Министерство образования Республики Беларусь
Белорусский государственный университет
Факультет прикладной математики и информатики
Кафедра компьютерных технологий и систем

СОГЛАСОВАНО
Заведующий кафедрой
_____ Казачёнок В. В.
«22» апреля 2025 г.

СОГЛАСОВАНО
Декан факультета
_____ Орлович Ю. Л.
«22» апреля 2025 г.
МП

Математические методы компьютерной графики

В 3-х частях.
Часть 2. Основные алгоритмы вычислительной геометрии

Электронный учебно-методический комплекс
для специальности:
6-05-0533-11 «Прикладная информатика»;
профилизация специальности: «Информационные аналитические системы»

Регистрационный № 2.4.2-24 / 618

Автор:
Шолтанюк С.В., старший преподаватель.

Рассмотрено и утверждено на заседании Научно-методического совета БГУ 24.04.2025 г., протокол № 9.

Минск 2025

УДК 004.925.8(075.8)
Ш 786

Утверждено на заседании Научно-методического совета БГУ.
Протокол № 9 от 24.04.2025 г.

Решение о депонировании вынес:
Совет факультета прикладной математики и информатики
Протокол № 8 от 22.04.2025 г.

Автор

Шолтанюк Станислав Витальевич, старший преподаватель кафедры компьютерных технологий и систем факультета прикладной математики и информатики Белорусского государственного университета.

Рецензенты:

кафедра программного обеспечения информационных технологий факультета компьютерных систем и сетей Белорусского государственного университета информатики и радиоэлектроники (заведующий кафедрой Лапицкая Н.В., кандидат технических наук, доцент),

Васильков Д.М., доцент кафедры биологической медицины и информатики факультета прикладной математики и информатики Белорусского государственного университета, кандидат физико-математических наук, доцент.

Шолтанюк, С.В. Математические методы компьютерной графики. В 3-х частях. Часть 2. Основные алгоритмы вычислительной геометрии : электронный учебно-методический комплекс для специальности 6-05-0533-11 «Прикладная информатика» / С.В. Шолтанюк ; БГУ, Фак. прикладной математики и информатики, Каф. компьютерных технологий и систем. – Минск : БГУ, 2025. – 500 с. : ил. – Библиогр.: с. 483–485.

Электронный учебно-методический комплекс (ЭУМК) предназначен для студентов специальности «Прикладная информатика». Настоящий ЭУМК посвящён базовым алгоритмам компьютерной графики и вычислительной геометрии, которые предназначены для решения различных задач: отсечения и удаления невидимых частей объектов, растеризация, разбиение объектов на элементарные части, построение оптических эффектов и некоторые другие. Для этих алгоритмов дано подробное описание и обоснование, а также приводятся примеры их работы. В практическом разделе приведены задачи для самостоятельного решения на использование описанных алгоритмов, а в ответах к большинству из них приведены чертежи, что наряду с числовыми результатами даёт лучшее понимание назначения и работы рассматриваемых алгоритмов. Сопровождается настоящий ЭУМК материалами для контроля знаний (варианты для контрольной работы, список вопросов для коллоквиума и экзамена) и перечнем рекомендуемой и дополнительной литературы, среди которой научные статьи – источники, где впервые описаны и обоснованы рассматриваемые алгоритмы.

ОГЛАВЛЕНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дисциплина «Математические методы компьютерной графики», знакомящая обучающихся с математическими моделями, применяемыми в компьютерной графике, предлагает, в частности, изучение некоторых базовых алгоритмов, используемых при построении компьютерной графики. Необходимость этого связана с тем, что построение изображения сцен, содержащих множество разнородных объектов и эффектов (освещение, окраска объектов, взаимное положение объектов, положение наблюдателя относительно сцены и её объектов и многие другие), требует решения многих подзадач, например отсечение и удаление невидимых объектов и их частей, упорядочение и сортировка геометрических объектов, разрезание и разбиение геометрических областей и т.д. Подобные задачи решаются при помощи методов и алгоритмов вычислительной геометрии – раздела теоретической информатики, изучающего алгоритмы и структуры данных для геометрических данных[1]. Настоящий ЭУМК, который является продолжением первой части [1], посвящён базовым алгоритмам вычислительной геометрии. Многие из них разработаны ещё во 2-й половине XX в., однако остаются актуальными и по сей день.

В настоящем ЭУМК алгоритмы вычислительной геометрии рассматриваются преимущественно с математической и алгоритмической точки зрения. В частности, для большинства алгоритмов оценивается временная асимптотика. При разработке алгоритмов вычислительной графики их временная оптимизация является одним из решающих этапов разработки, ведь сцена может состоять из тысяч и миллионов объектов, а её отрисовка должна проводиться за ограниченное время (зачастую за доли секунды). Для обозначения временной асимптотики используется широко принятая O-символика, знакомая обучающимся по дисциплинам «Математический анализ» и «Основы и методологии программирования». Более подробное объяснение O-символики, а также описание базовых особенностей реализации алгоритмов компьютерной графики на доступном языке изложено в [2, глава 1]. В настоящем ЭУМК не рассматриваются особенности программной реализации на том или ином языке программирования. Подобные вопросы обучающиеся изучат на дисциплине «Программирование компьютерной графики».

Также важно отметить такой момент, как описание алгоритмов. Автором было принято решение привести подробное, пошаговое описание алгоритмов в словесной форме, которое опирается также на математические формулы, выводимые при обосновании алгоритмов. Такой подход также позволяет давать пояснения для некоторых шагов здесь же, «на месте». Во многих классических учебниках по компьютерной графике наряду со словесным описанием для описания алгоритмов также часто используются блок-схемы (например, в [3–5]) или псевдокод (как в [2; 5]). При необходимости эти и другие источники могут

[1] Определение с сайта https://www.computational-geometry.org/

быть рекомендованы для лучшего понимания изучаемых алгоритмов (особенно при самостоятельной их программной реализации обучающимися).

С учётом того, что данная дисциплина является математической, ещё одной особенностью настоящего ЭУМК является наличие доказательств корректности для большинства приведённых алгоритмов. Большинство этих доказательств были взяты из оригинальных научных статей (перечень которых содержится в списке дополнительной литературы, и ссылки на которые приведены в начале подразделов, посвящённых соответствующим алгоритмам), где алгоритмы были представлены впервые самими авторами, и адаптированы для лучшего их понимания обучающимися. Для некоторых алгоритмов, с другой стороны, описаны доказательства, выведенные самим автором настоящего ЭУМК.

При преподавании раздела «Основные алгоритмы вычислительной графики», которому и посвящён настоящий ЭУМК, рекомендуется учитывать следующие особенности. Так, при чтении лекций весьма желательна демонстрация графических материалов, наглядно и пошагово иллюстрирующих работу изучаемых алгоритмов на конкретных примерах. Это могут быть заранее заготовленные рисунки и чертежи (например, которые приведены в настоящем ЭУМК) либо интерактивная графика, построенная при помощи систем компьютерной математики и алгебры (такие как Wolfram Mathematica, Mathcad) и/или языков программирования, и демонстрируемая в соответствующей среде программирования. Кроме того, важно обращать внимание на особенности работы алгоритмов, некоторые из которых приведены в замечаниях после их описания. Для каждого из алгоритмов рекомендуется хотя бы вкратце пояснять доказательство его корректности, за исключением случаев, когда корректность алгоритма целиком следует из его вывода или когда это доказательство является слишком громоздким.

Что касается лабораторных занятий, то здесь, как и при изучении первого раздела данной дисциплины, предлагается решение задач вычислительной геометрии при помощи изученных алгоритмов, которые выполняются вручную на бумаге или доске. В практическом разделе настоящего ЭУМК приведены подробные примеры решения задач, где продемонстрирована работа алгоритмов шаг за шагом, промежуточные и итоговые результаты их работы. Вместе с тем, ввиду затруднительных вычислений при решении некоторых из приведённых здесь задач, а также с учётом особенности специальности «Прикладная информатика» в качестве дополнительного задания обучающимся может быть предложено программно реализовать и протестировать некоторые из изучаемых алгоритмов. Программная реализация и тестирование алгоритмов может иногда указать на некоторые особенности их работы, которые могут быть упущены при их использовании вручную.

Текущая аттестация по данному разделу дисциплины предусмотрена в форме одного коллоквиума и одной контрольной работы. Коллоквиум представляет собой устный или письменный опрос по пройденному теоретическому материалу, который при необходимости может быть дополнен решением задачи. Контрольная работа проходит в форме самостоятельного

решения задач с применением подходящих алгоритмов. Как и в любой математической дисциплине, рекомендуется в первую очередь оценивать ход решения этих задач, что обусловлено, в частности, выбором нужного и наиболее подходящего для конкретного примера алгоритма.

Самостоятельная работа обучающихся, как и при изучении первого раздела данной дисциплины, предполагает в первую очередь работу с настоящим ЭУМК. Также может иметь место работа с источниками из приведённого списка литературы, в т.ч. первой части ЭУМК [1], или с другими материалами. Разумеется, при решении задач, в т.ч. предложенных в практическом разделе настоящего ЭУМК, возможно использование вычислительных средств для выполнения алгоритмов и визуализации их результатов.

Данная дисциплина в первую очередь предназначена для студентов, изучающих «Математические методы компьютерной графики» в университете, и для преподавателей этой дисциплины. Материал, представленный в настоящем ЭУМК, может быть полезен также для студентов, аспирантов, преподавателей и прочих людей, занимающихся или интересующихся компьютерной графикой, в т.ч. алгоритмами вычислительной геометрии.

В настоящем ЭУМК используются следующие сокращения и обозначения:

◆ – начало доказательства

■ – конец доказательства

& – логическое «и», побитовое «и»

$\vee$ – логическое «или», побитовое «или»

$\oplus$ – побитовая операция «исключающее или» (XOR)

$\Rightarrow$ – логическое следование, импликация

$\Leftrightarrow$ – эквивалентность

$\forall$ – квантор всеобщности

$\exists$ – квантор существования

$\mathbb{N}$ – множество натуральных чисел

$\mathbb{Z}$ – множество целых чисел

$\mathbb{Q}$ – множество рациональных чисел

$\mathbb{R}$ – множество действительных чисел

$\overline{a,b}$ – все натуральные числа от a до b (включительно) abs a – модуль (абсолютное значение) числа a; используется также обозначение $|a|$

sgn x («сигнум») – функция, определяющая знак числа x:

$$\operatorname{sgn} x = \begin{cases} -1, x < 0, \\ 0, x = 0, \\ 1, x > 0 \end{cases}$$

$M(x, y, z)$ – точка M с координатами x, y, z; иногда для наименования точки вместо прописных латинских букв используются строчные, либо используются только координаты точки (без её наименования)

$\vec{a}(a_x, a_y, a_z)$ – вектор $\vec{a}$ с координатами a_x, a_y, a_z; местами (особенно в практическом разделе) при вычислении векторных операций наименования векторов могут отсутствовать

$\vec{a} \parallel \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ коллинеарны

$\vec{a} \nparallel \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ не являются коллинеарными

$\vec{a} \uparrow\uparrow \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ сонаправлены

$\vec{a} \uparrow\downarrow \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ противонаправлены

$\vec{a} \perp \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ ортогональны

$\vec{a} \not\perp \vec{b}$ – векторы $\vec{a}$ и $\vec{b}$ не являются ортогональными

$\vec{a} \cdot \vec{b}$ – скалярное произведение векторов $\vec{a}$ и $\vec{b}$

$\vec{a} \times \vec{b}$ – векторное произведение[1] векторов $\vec{a}$ и $\vec{b}$

$\vec{a}\vec{b}\vec{c}$ – смешанное произведение векторов $\vec{a}$, $\vec{b}$ и $\vec{c}$: $\vec{a}\vec{b}\vec{c} = (\vec{a} \times \vec{b}) \cdot \vec{c}$

$|\vec{a}|$ – длина вектора $\vec{a}$

$\angle(\vec{a}, \vec{b})$ – угол между векторами $\vec{a}$ и $\vec{b}$

$p + \vec{a}$ – точка q, которая получается путём откладывания от точки p вектора $\vec{a}$ (т.е. $\overrightarrow{pq} = \vec{a}$)

$\mathbb{R}_{m,n}$ – множество действительных матриц с m строками и n столбцами

$\mathbb{R}^c_{m,n}$ – множество матриц с m строками и n столбцами, у которых каждый элемент является числовым действительным вектором размерности c

$A \succ 0$ – все элементы матрицы A больше нуля

$A = (a_{ij})$ – матрица A, у которой на пересечении i-ой строки и j-ого столбца расположен элемент a_{ij}

$\lfloor a \rfloor$ – (нижняя) целая часть числа a, т.е. максимальное целое число, не большее a

$\lceil a \rceil$ – верхняя целая часть числа a, т.е. минимальное целое число, не меньшее a

round a – функция, результатом которой является округлённое до ближайшего целого значения число a

$\{a\}$ – дробная часть числа a, вычисляемая по формуле $\{a\} = a - \lfloor a \rfloor$

$a := b$ – присвоить переменной a значение, равное b

[1] Кроме общепринятого понятия векторного произведения двух векторов трёхмерного пространства используется также понятие векторного произведения векторов на плоскости, введённое в первой части ЭУМК [1, с. 31, 48–49]

$O(f(x))$ («О большое от функции f») – некоторая функция $g(x)$, которая при $x \to \infty$ возрастает не быстрее, чем $Cf(x)$, где C – некоторая положительная константа, т.е.

$$\exists C > 0 \; \exists x_0 \; \forall x > x_0 : \frac{|g(x)|}{|f(x)|} \le C$$

$\Omega(f(x))$ («Омега большое от функции f») – некоторая функция $g(x)$, которая при $x \to \infty$ возрастает не медленнее, чем $Cf(x)$, где C – некоторая положительная константа, т.е.

$$\exists C > 0 \; \exists x_0 \; \forall x > x_0 : \frac{|g(x)|}{|f(x)|} \ge C$$

$\Theta(f(x))$ («Тэта большое от функции f») – некоторая функция $g(x)$, которая при $x \to \infty$ возрастает с той же скоростью, что и функция $Cf(x)$, где C – некоторая положительная константа, т.е.

$$\exists C > 0 \; \exists x_0 \; \forall x > x_0 : \frac{|g(x)|}{|f(x)|} = C$$

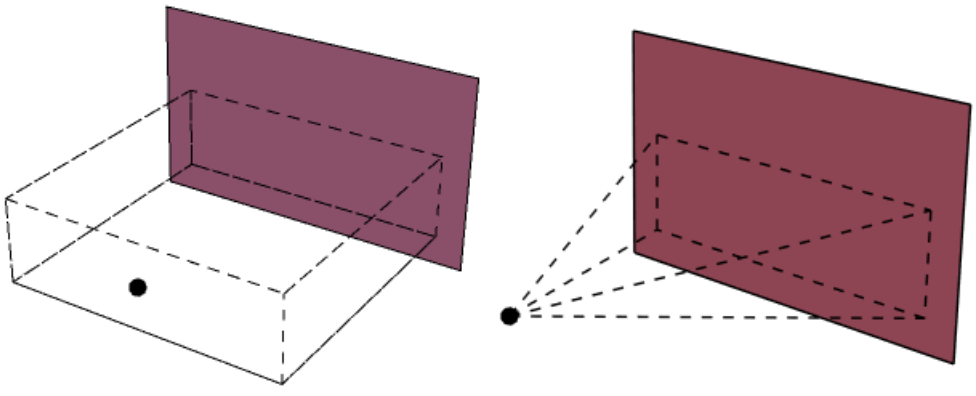
1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

1.1. Алгоритмы отсечения и удаления невидимых рёбер и граней

Для отображения графических объектов на экране, имеющем, как правило, прямоугольную форму, необходимо учитывать, будут ли видны эти объекты полностью, частично либо не будут видны вовсе. Если некоторый объект или его части не будут видны на экране, то нет смысла тратить время и ресурсы на их рисование. Кроме того, также не должны рисоваться объекты или их части, загороженные другими объектами от наблюдателя. Таким образом, наряду с проективными преобразованиями имеет место ещё один важный этап построения компьютерной графики, предшествующий выводу графики на экран, – **отсечение** и **удаление** невидимых объектов.

Существуют различные методы отсечения, которые в зависимости от назначения и реализации делятся на следующие категории:

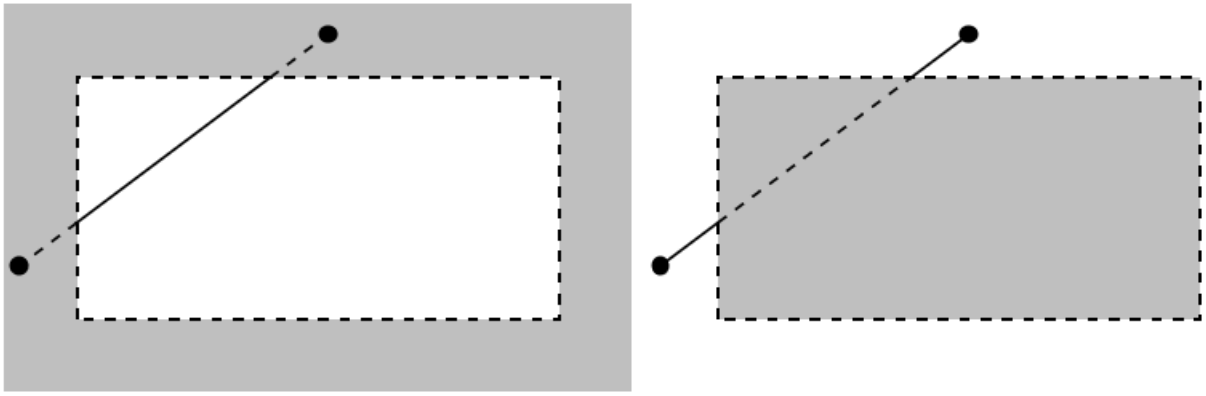
- Объёмные (Рисунок 1.1) и плоские (Рисунок 1.2). При использовании объёмных отсекателей отсечение предшествует проектированию объектов на картинную плоскость, при использовании плоских – наоборот.



а б

Рисунок 1.1 – Объёмные отсекатели. Параллелепипедальный (а) используется при параллельном проектировании объектов на картинную плоскость, пирамидальный (б) – при центральном. Точкой указана позиция наблюдателя.

- Внешние (экран) и внутренние (ширма) – отсекают объекты, расположенные соответственно вне и внутри отсекателя (Рисунок 1.2).



а б

Рисунок 1.2 – Плоские отсекатели (а – внешний, б – внутренний)

- Регулярные (прямоугольной и параллелепипедальной формы) и нерегулярные (имеющие произвольную форму).
- Аппаратные и программные.

1.1.1. Алгоритм Коэна-Сазерленда

Один из простейших алгоритмов отсечения, предложенный Р. Ф. Спруллом и А. Э. Сазерлендом в 1968 году[1] [6], проверяет положение отрезка относительно прямоугольного окна (экрана), на основе чего возвращает либо видимую в этом окне часть этого отрезка, либо пустой результат. Для этого выполняется проверка, находятся ли два конца данного отрезка выше, ниже, правее или левее границ окна. Пусть прямоугольное окно ограничено прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$. Тогда всякой точке (x, y) картинной плоскости можно поставить в соответствие четырёхбитное число: первый бит равен единице тогда и только тогда, когда точка расположена выше окна, т.е. когда справедливо строгое неравенство $y > y_{\max}$; второй бит – когда точка ниже окна: $y < y_{\min}$; третий – когда точка правее окна: $x > x_{\max}$; четвёртый – когда точка левее окна: $x < x_{\min}$. Таким образом, вся плоскость разбивается на девять областей (Рисунок 1.3).

Данный алгоритм опирается в первую очередь на побитовые операции. Пусть для двух концов отрезка AB с координатами $A(x_A, y_A)$ и $B(x_B, y_B)$ получили коды a и b. Тогда могут иметь место следующие случаи:

- Если отрезок полностью находится внутри окна, и, следовательно, не подлежит отсечению (отрезок 1 на рисунке 1.3), то обоим концам отрезка соответствует код 0000. Отсюда следует $a \vee b = 0000$. Несложно видеть, что верно и обратное, т.е. выполнение такого

[1] Хотя Д. Коэн, один из тогдашних аспирантов Сазерленда, не является соавтором этой статьи, алгоритм всё же назван в том числе в его честь, так как «его идеи оказали сильное влияние на проект» [7].

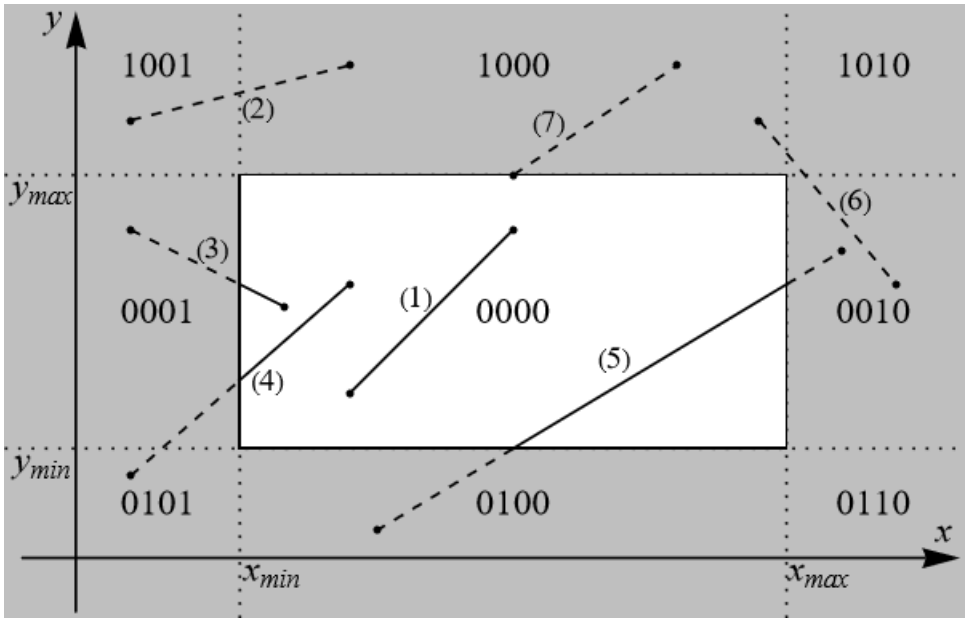


Рисунок 1.3 – Коды областей, ограниченных четырьмя прямыми

равенства означает расположение отрезка целиком в пределах окна. Значит, в этой ситуации алгоритм должен вернуть весь отрезок.

- Если оба конца отрезка находятся одновременно выше, ниже, правее либо левее окна, то оба соответствующих бита чисел a и b равны 1. Отсюда следует $a \,\&\, b \neq 0000$. Например, для отрезка 2 на рисунке 1.3, оба конца которого расположены выше окна, получится $a \,\&\, b = 1000$. Это достаточное условие для того, чтобы отрезок AB оказался полностью лежащим за пределами окна. Такой отрезок отсекается целиком – алгоритм возвращает пустой результат.
- При невыполнении предыдущих двух случаев отрезок может оказаться частично (отрезки 3, 4 и 5) либо полностью (отрезок 6) расположенным за пределами окна. В таких случаях происходит отсечение отрезка с одной либо с обеих сторон в зависимости от количества невидимых концов.
- Существует также вырожденный случай, когда весь отрезок, за исключением одной из граничных точек (отрезок 7), находится за пределами окна. Такой отрезок будем полагать подлежащим полному отсечению.

Одним из подходов обработки нетривиальных случаев расположения отрезка относительно окна является вычисление пересечений отрезка с прямыми $y = y_{\max}$, $y = y_{\min}$, $x = x_{\max}$, $x = x_{\min}$. Отрезок пересекает некоторую из этих прямых тогда и только тогда, когда два его конца находятся по разные от неё стороны (не считая случая, когда отрезок лежит на этой прямой), что эквивалентно разным значениям четырёхбитных кодов на соответствующих позициях (например, отрезок пересекает прямую $y = y_{\max}$ тогда и только тогда, когда его концам соответствуют коды, у одного из которых первый бит равен нулю, у второго тот же бит равен единице). Значит, для определения, с какими прямыми пересекается отрезок AB, достаточно вычислить $a \oplus b$. Например,

отрезок 3 на рисунке 1.3 пересекает только левую прямую, т.к. «исключающее “или”» кодов двух его концов равно $0001 \oplus 0000 = 0001$, а отрезок 6 пересекает уже две прямые: верхнюю и правую ($1000 \oplus 0010 = 1010$). Если же отрезок лежит на некоторой прямой – продолжении границы окна, то коды обоих концов содержат нули на соответствующей позиции, и пересечение с этой прямой вычислению не подлежит.

Зная, какие прямые пересекает данный отрезок, несложно найти точки пересечения. Так, если отрезок AB с концами в точках с координатами $A(x_A, y_A)$ и $B(x_B, y_B)$ пересекает некоторую горизонтальную прямую $y = y_0$, то найдётся некоторое значение параметра $t \in [0,1]$ такое, что выполняется равенство

$$y_A(1-t) + y_B t = y_0.$$

Это значение равно $t_0 = \dfrac{y_0 - y_A}{y_B - y_A}$. Зная его, получим точку пересечения $C(x_C, y_C)$ с координатами

$$\begin{gathered} x_C = x_A(1-t_0) + x_B t_0 = x_A + t_0(x_B - x_A) = x_A + \frac{(y_0 - y_A)(x_B - x_A)}{y_B - y_A}, \\ y_C = y_0. \end{gathered} \tag{1.1}$$

Вырожденный случай $y_A = y_B$ эквивалентен равенству $y_A = y_B = y_0$, при котором, как показано выше, пересечение вычисляться не будет. Таким образом, формулы (1.1) всегда будут корректно вычисляться. Аналогично получается формула для вычисления пересечения $D(x_D, y_D)$ отрезка AB с вертикальной прямой $x = x_0$:

$$\begin{gathered} x_D = x_0, \\ y_D = y_A + \frac{(x_0 - x_A)(y_B - y_A)}{x_B - x_A}. \end{gathered} \tag{1.2}$$

После вычисления координат точек пересечения по формулам (1.1) и (1.2) вычисляются четырёхбитные коды этих точек. Только те точки, для которых этот код равен 0000, лежат непосредственно на границе отсекающего окна.

Перед тем как привести формальное описание алгоритма, необходимо сделать ещё одно замечание. Так как по алгоритму Коэна-Сазерленда в первую очередь необходимо вычислять $a \,\&\, b$, и только в случае $a \,\&\, b = 0000$ проводить дальнейшие вычисления, то вместо кода $a \oplus b$ можно использовать код $a \vee b$. Это вытекает из того, что $a \,\&\, b = 0000 \Rightarrow a \oplus b = a \vee b$. Справедливость этого

утверждения несложно проверить исходя из таблицы ниже, где через a_i и b_i обозначены i-ые биты кодов a и b.

a_i	b_i	$a_i \,\&\, b_i$	$a_i \vee b_i$	$a_i \oplus b_i$
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Верхняя строка данной таблицы указывает на случай, когда $a \,\&\, b \neq 0000$, и алгоритм Коэна-Сазерленда сразу же возвращает пустой результат. Таким образом, вместо вычисления трёх кодов $a \vee b$, $a \,\&\, b$ и $a \oplus b$ достаточно обойтись первыми двумя, а при определении прямых, пересекаемых отрезком, использовать результат побитового «или» кодов a и b.

Итак, **программная реализация алгоритма Коэна-Сазерленда** отсечения отрезка AB с концами в точках $A(x_A, y_A)$ и $B(x_B, y_B)$ прямоугольным окном $\{(x, y) : x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}\}$ выглядит следующим образом:

1. Для концов A и B вычисляются коды a и b.
2. Если $a \,\&\, b \neq 0000$, то алгоритм завершает работу с возвращением пустого результата.
3. Вычисляется код $l = a \vee b$. Если $l = 0000$, то алгоритм завершает работу, возвращая два конца исходного отрезка.
4. Определяется множество точек – потенциальных концов искомого отрезка: $P := \{A, B\}$.
5. Если первый и/или второй бит l равен единице, то во множество P добавляется одна или две точки, координаты которых вычисляются по формулам (1.1) при $y_0 = y_{\max}$ и/или $y_0 = y_{\min}$ соответственно.
6. Если третий и/или четвёртый бит l равен единице, то во множество P добавляется одна или две точки, координаты которых вычисляются по формулам (1.2) при $x_0 = x_{\max}$ и/или $x_0 = x_{\min}$ соответственно.
7. Алгоритм возвращает те и только те точки из списка P, коды для которых равны 0000. Таких точек может быть две (концы искомого отрезка), одна (когда исходный отрезок, за исключением одного из концов, лежит за пределами окна) либо не оказаться вовсе.

По указанной ранее договорённости, в случае, когда этот алгоритм возвращает одну вершину, отрезок полностью отсекается, а следовательно, рисованию не подлежит.

Другая реализация этого алгоритма приведена в [7, с. 66–67].

При **аппаратной реализации** алгоритма Коэна-Сазерленда отсечение отрезка осуществляется при помощи средней точки $M(x_M, y_M)$. Тогда для двух отрезков AM и BM можно проверить выполнение тривиальных случаев. Если

они не выполняются, то два отрезка снова разбиваются на две половины и т.д., пока средние точки не попадут на границы окна.

Следует отметить, что для исходного отрезка *AB*, пересекающего границу окна, середина очередного отрезка с координатами (*x*, *y*) рано или поздно попадёт на границу окна с некоторой точностью $\varepsilon > 0$, т.е. будет выполняться хотя бы одно из следующих условий:

$$\begin{aligned} &|y - y_{\max}| < \varepsilon \;\&\; x_{\min} < x < x_{\max}, \\ &|y - y_{\min}| < \varepsilon \;\&\; x_{\min} < x < x_{\max}, \\ &|x - x_{\max}| < \varepsilon \;\&\; y_{\min} < y < y_{\max}, \\ &|x - x_{\min}| < \varepsilon \;\&\; y_{\min} < y < y_{\max}. \end{aligned} \tag{1.3}$$

Число ε может быть задано заранее, либо в качестве него может использоваться **машинный эпсилон** устройства, т.е. такое наименьшее число $\varepsilon_0 > 0$, что при вычислении числа $1 + \varepsilon_0$ на этом устройстве получается результат, отличный от единицы.

Таким образом, **аппаратная реализация алгоритма Коэна-Сазерленда** (известная также как **алгоритм средней точки**) отсечения отрезка *AB* с концами в точках $A(x_A, y_A)$ и $B(x_B, y_B)$ прямоугольным окном $\{(x, y) : x_{\min} \le x \le x_{\max},\ \ y_{\min} \le y \le y_{\max}\}$ с точностью $\varepsilon > 0$ представима в виде следующей последовательности команд:

1. Для концов отрезка *A* и *B* вычисляются их коды *a* и *b*.
2. Если выполняется равенство $a \vee b = 0000$, то полностью рисуется отрезок *AB*. Алгоритм завершает работу.
3. Если выполняется неравенство $a \,\&\, b \neq 0000$, то отрезок отбрасывается. Конец работы алгоритма.
4. Если хотя бы для одного из концов отрезка выполняется какое-то из условий (1.3)[1], то в четырёхбитном коде другого конца необходимо проверить бит, соответствующий той границе отсекателя, на которой лежит первая вершина. Если этот бит равен 1 (т.е. другой конец лежит по внешнюю сторону от нужной границы), то весь отрезок, кроме одной вершины, лежит строго снаружи отсекателя и подлежит полному отсечению, а алгоритм завершает свою работу. Так, на рисунке 1.4 изображены два отрезка с общим концом *A*, лежащим на верхней границе отсекателя, но один из них (AB_1) подлежит полному отсечению, так как точка B_1 лежит выше верхней границы, а у другого отрезка (AB_2) конец B_2 лежит нише верхней границы, значит, этот отрезок отчасти расположен внутри отсекателя.

[1] При использовании машинного эпсилона эти неравенства эквивалентны побитовому равенству координат точек соответствующим координатам границ окна

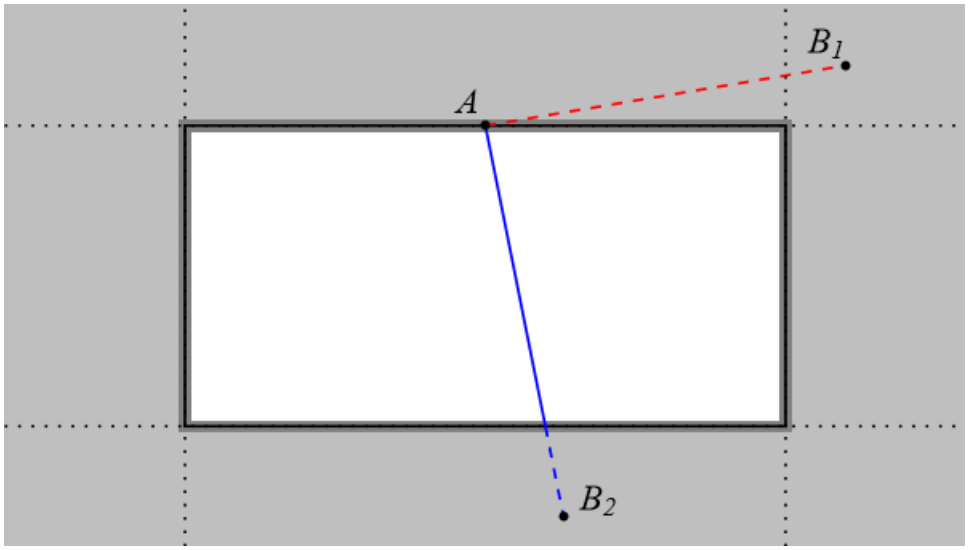


Рисунок 1.4 – Варианты расположения и отсечения отрезков с концом, лежащим на границе отсекателя. Тёмно-серым цветом обозначена ε-окрестность границы окна

5. Вычисляются координаты середины отрезка $M(x_M, y_M): M := \dfrac{A+B}{2}$.
6. Для отрезков AM и BM выполняется этот же алгоритм[1].

Замечание. Программная реализация алгоритма Коэна-Сазерленда использует ограниченное число операций, которые в большинстве приложений можно рассматривать как элементарные: побитовые «и», «или», а также четыре базовых арифметических операции. В то время как для аппаратной реализации число операций можно оценить асимптотикой $O(\log L)$, где L – длина отрезка.

◆ Для доказательства этого факта рассмотрим три случая:
- один из концов отрезка видимый,
- оба конца невидимы, отрезок полностью невидимый,
- оба конца невидимы, отрезок частично видимый.

В первом случае, если алгоритм не завершается на шагах 2–4, происходит деление отрезка на две части, у каждого из которых длина равна $L/2$. Если точка деления M_1 не попала на границу окна, то один из отрезков окажется либо полностью видимым, либо подлежит полному отсечению. Значит, далее алгоритм выполняется только для одной из половинок исходного отрезка. Например, для отрезка на рисунке 1.5 имеет место вначале деление средней точкой M_1, после чего отрезок BM_1 оказывается полностью лежащим внутри окна, а следовательно, подлежащим отрисовке. Отрезок AM_1 далее делится пополам точкой M_2, после чего отрезок AM_2 полностью отсекается, а отрезок M_1M_2 делится далее. Этот процесс продолжается до тех пор, пока очередная средняя точка не оказывается в ε-окрестности некоторой границы окна (в данном примере точка M_4 оказалась в окрестности левой границы). При этом

[1] Таким образом, имеет место рекурсия – вызов функции из неё же самой

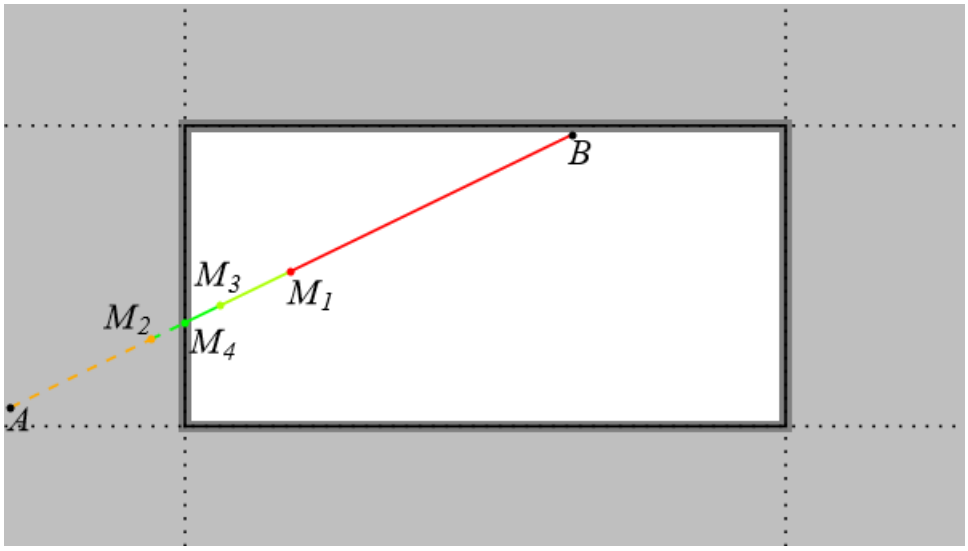


Рисунок 1.5 – Пример работы алгоритма средней точки

одна из половинок текущего отрезка полностью отрисовывается согласно шагу 2, другая – полностью отбрасывается согласно шагу 4.

Всё сказанное выше может выполняться до тех пор, пока у очередного отрезка длина превышает 2ε. Если же у очередного отрезка длина не больше 2ε, то для его середины непременно выполняется одно из неравенств (1.3).

Ясно, что для произвольного отрезка число операций на шагах 1–5 можно ограничить некоторым числом n. Обозначая через $N(L)$ число операций для отрезка длины L, получаем следующую цепочку неравенств:

$$N(L) \le n + N(L/2) \le 2n + N(L/4) \le 3n + N(L/8) \le$$
$$\le \ldots \le \left\lceil \log_2 \frac{L}{2\varepsilon} \right\rceil n + N(2\varepsilon) \le \left(\left\lceil \log_2 \frac{L}{2\varepsilon} \right\rceil + 1 \right) n.$$

Коэффициент, стоящий в правой части этой цепочки, равен $\lceil \log_2 L - \log_2 \varepsilon - 1 \rceil + 1 = \lceil \log_2 L - \log_2 \varepsilon \rceil = O(\log L)$. Домножение этого выражения на постоянный коэффициент n не изменяет этой асимптотики. Таким образом, утверждение доказано для первого случая.

Во втором случае алгоритм заканчивается, когда очередная точка деления попадает в «угловую» область картинной плоскости. Например, на рисунке 1.6 после вычисления точки M_4 отрезок M_1M_4 отбрасывается как находящийся целиком слева от окна, а M_3M_4 – как находящийся целиком сверху. Несложно видеть, что это происходит максимум за $\log_2(L/l)$ итераций, где l – длина части отрезка, лежащая в «угловой» области. Действительно, если точка деления попадает не в «угловую» область, то одна из половинок текущего отрезка должна быть сразу же отсечена. Значит, как и в первом случае, далее рассматривается только одна половинка отрезка. Как только длина текущего отрезка станет меньше $2l$, очередная средняя точка непременно попадёт в «угловую» область, и отсечению подлежат обе половинки отрезка.

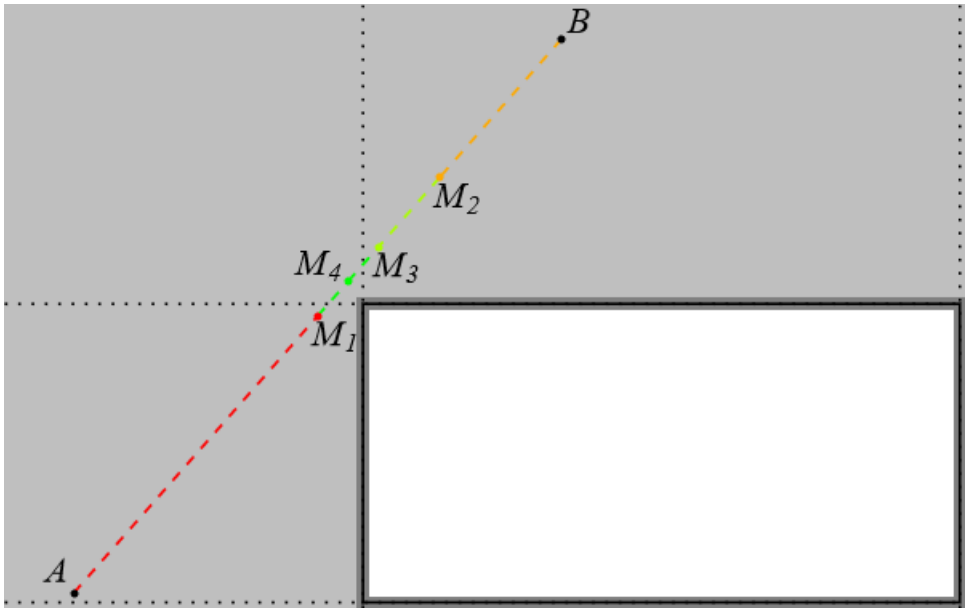


Рисунок 1.6 – Работа алгоритма средней точки для отрезка, полностью находящегося за пределами окна

Значит, максимальное число итераций возможно в ситуации, когда отрезок проходит через угол окна (Рисунок 1.7). При этом две половинки оставшейся части исходного отрезка $(M_2M_5$ и $M_4M_5)$ отбрасываются согласно шагу 4. Кроме, того, минимальная длина части отрезка, попадающей в ε-окрестность границы окна, равна $2\sqrt{2}\varepsilon$. Поэтому верхняя граница для числа операций равна $\left(\left\lceil \log_2 \frac{L}{2\sqrt{2}\varepsilon} \right\rceil + 2\right) n = O(\log L)$.

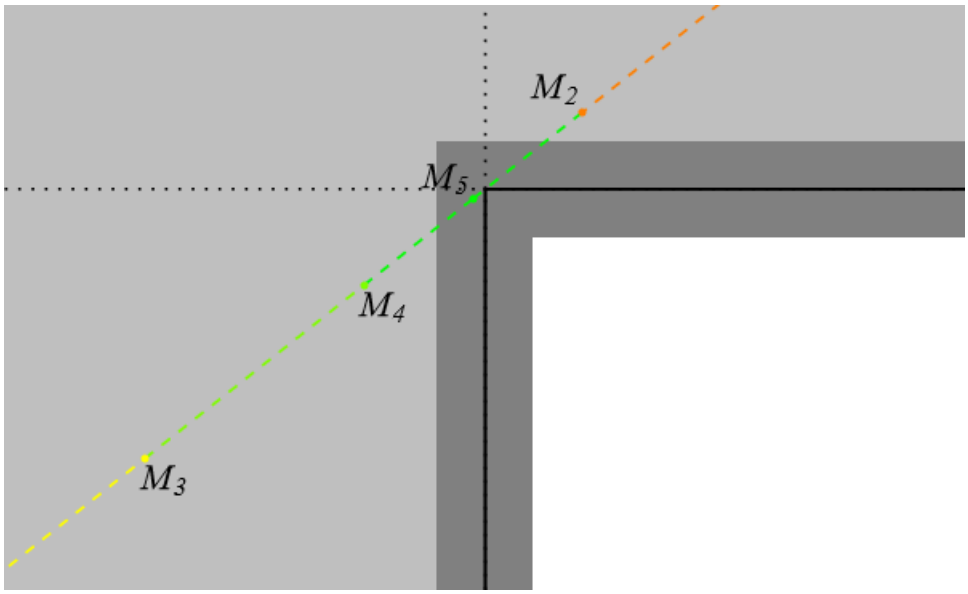
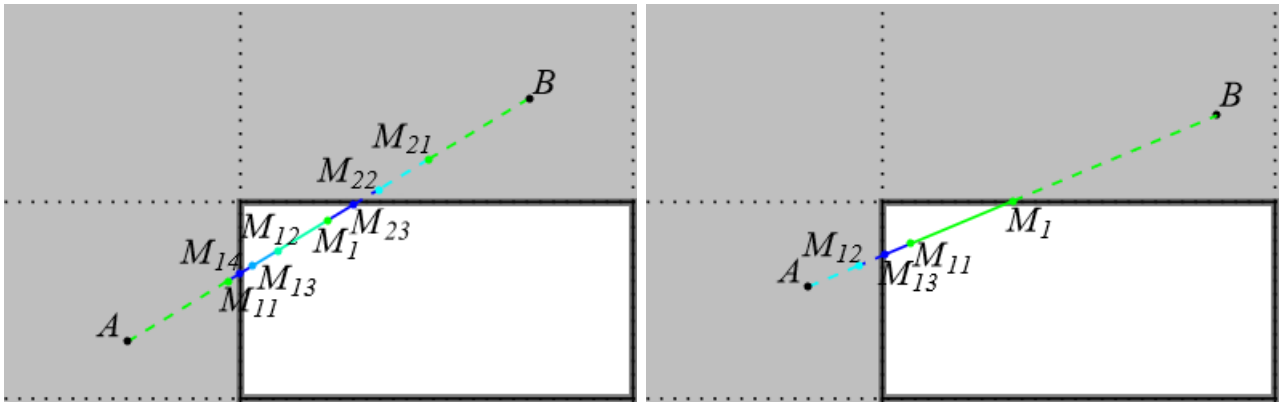


Рисунок 1.7 – Работа алгоритма для отрезка, проходящего через угол окна

Наконец, в третьем случае происходит отсечение отрезков до тех пор, пока очередная средняя точка не попадёт внутрь (Рисунок 1.8а) либо на границу окна (Рисунок 1.8б). Аналогично второму случаю, максимальное число



а б

Рисунок 1.8 – Обработка частично видимых отрезков

итераций осуществляется, когда отрезок проходит в окрестности угловой точки окна, и может быть оценено асимптотикой $O(\log L)$.

После этого рассматриваются отдельно не более двух отрезков, для каждого из которых имеет место первый случай. У каждого из них длина не превышает L, значит, число операций на этом этапе для каждого из отрезков также выражается асимптотикой $O(\log L)$. Таким образом, общее число операций в третьем случае выражается асимптотикой $O(\log L)$. ■

Итак, алгоритм средней точки, будучи реализованным программно, работал бы медленнее программной реализации алгоритма Коэна-Сазерленда. Однако аппаратная реализация благодаря быстрым побитовым операциям (побитовые «и», «или», сложение, а также побитовые сдвиги вправо для деления на 2) позволяет добиться значительного ускорения.

Другой вариант алгоритма нахождения пересечений отрезком границ регулярного окна, использующего среднюю точку, представлен в [7, с. 68].

1.1.2. Алгоритм Лианга-Барски

Другой алгоритм, который осуществляет отсечение отрезка регулярным окном, был предложен Ю. Ляном и Б. Барски в 1984 году [8].

Снова рассмотрим некоторый отрезок AB с концами в точках с координатами $A(x_A, y_A)$ и $B(x_B, y_B)$ и отсекатель, ограниченный прямыми $y = y_{\max}$, $y = y_{\min}$, $x = x_{\max}$, $x = x_{\min}$. Суть алгоритма Лианга-Барски заключается в использовании параметрического задания отрезка:

$$\begin{cases} x = x_A + (x_B - x_A)t, \\ y = y_A + (y_B - y_A)t. \end{cases} \qquad (1.4)$$

При таком задании параметру $t = 0$ соответствует точка A, при $t = 1$ получается точка B. Искомый отрезок, несложно заметить, может быть описан

этими же равенствами, только его концам будут отвечать уже другие значения *t*. Таким образом, задача отсечения отрезка *AB* сводится к нахождению значений для параметра *t*, соответствующих концам усечённого отрезка.

Для искомого отрезка получается следующая система неравенств:

$$\begin{cases} 0 \le t \le 1, \\ x_{\min} \le x \le x_{\max}, \\ y_{\min} \le y \le y_{\max}. \end{cases}$$

Подставим в два последних неравенства равенства (1.4):

$$\begin{gathered} x_{\min} \le x, \\ x_{\min} \le x_A + (x_B - x_A)t, \\ x_{\min} - x_A \le (x_B - x_A)t, \\ (x_A - x_B)t \le x_A - x_{\min}, \end{gathered} \qquad \begin{gathered} x \le x_{\max}, \\ x_A + (x_B - x_A)t \le x_{\max}, \\ (x_B - x_A)t \le x_{\max} - x_A, \end{gathered}$$

$$\begin{gathered} y_{\min} \le y, \\ y_{\min} \le y_A + (y_B - y_A)t, \\ y_{\min} - y_A \le (y_B - y_A)t, \\ (y_A - y_B)t \le y_A - y_{\min}, \end{gathered} \qquad \begin{gathered} y \le y_{\max}, \\ y_A + (y_B - y_A)t \le y_{\max}, \\ (y_B - y_A)t \le y_{\max} - y_A. \end{gathered}$$

Таким образом, получаются дополнительные условия для параметров *t*, соответствующих концам искомого отрезка:

$$p_i t \le q_i, \ \ i = \overline{1,4}, \tag{1.5}$$

где

$$\begin{aligned} p_1 &= x_A - x_B, & q_1 &= x_A - x_{\min}, \\ p_2 &= x_B - x_A, & q_2 &= x_{\max} - x_A, \\ p_3 &= y_A - y_B, & q_3 &= y_A - y_{\min}, \\ p_4 &= y_B - y_A, & q_4 &= y_{\max} - y_A. \end{aligned} \tag{1.6}$$

С учётом $t \ge 0$ и $t \le 1$ получается система из шести неравенств, решение которых даёт промежуток из параметров *t*, задающий искомый отрезок. Могут иметь место следующие три случая:

- При $p_i > 0$ формула (1.5) сводится к неравенству $t \le q_i / p_i$. Вместе с неравенством $t \le 1$ получается максимальное значение параметра *t*, при котором они все выполняются: $t_{\text{кон}} = \min\{\min\limits_{p_i>0} q_i / p_i, 1\}$.
- При $p_i < 0$ формула (1.5) сводится к неравенству $t \ge q_i / p_i$. Вместе с неравенством $t \ge 0$ получается наименьшее значение *t*, удовлетворяющее их системе: $t_{\text{нач}} = \max\{\max\limits_{p_i<0} q_i / p_i, 0\}$.
- При $p_i = 0$ имеем неравенство, не зависящее от *t*: $q_i \ge 0$. Если оно не выполняется хотя бы для одного из индексов *i*, при котором $p_i = 0$, это означает, что система из неравенств $0 \le t \le 1$ и (1.5) несовместна – алгоритм должен вернуть пустой результат. Если же $q_i \ge 0$ при всяких таких *i*, то они просто игнорируются.

Таким образом, **алгоритм Лианга-Барски** отсечения отрезка *AB* с концами в точках $A(x_A, y_A)$ и $B(x_B, y_B)$ прямоугольным окном $\{(x, y): x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}\}$ реализуется последовательностью следующих шагов:

1. Задаются начальные значения для параметров, отвечающих началу и концу текущего отрезка: $t_{\text{нач}} := 0$, $t_{\text{кон}} := 1$, а также итерационная переменная $i := 1$.
2. По формулам (1.6) вычисляются два значения p_i и q_i. При этом может возникнуть ровно одна из ситуаций:
   - 2а. При выполнении равенства $p_i = 0$ необходимо проверить неравенство $q_i < 0$. В случае его выполнения алгоритм немедленно завершает свою работу и возвращает пустой результат – отрезок **полностью отсекается**. В противном случае можно сразу перейти к шагу 4 алгоритма.
   - 2б. Если $p_i > 0$, то происходит **отсечение отрезка с конца**: $t_{\text{кон}} := \min\{q_i / p_i, t_{\text{кон}}\}$.
   - 2в. Если $p_i < 0$, то отрезок **отсекается с начала**: $t_{\text{нач}} := \max\{q_i / p_i, t_{\text{нач}}\}$.
3. Если оказалось справедливым неравенство $t_{\text{нач}} > t_{\text{кон}}$, то алгоритм немедленно возвращает пустой результат.
4. Если $i = 4$, то алгоритм возвращает отрезок с концами в точках, которые получаются при подстановке значений $t_{\text{нач}}$ и $t_{\text{кон}}$ в систему (1.4). Иначе алгоритм продолжается: $i := i + 1$, переход к шагу 2.

Иллюстрация работы этого алгоритма для разных случаев приведена на рисунке 1.9.

Этот алгоритм, как показали эксперименты самих авторов [8], работает в среднем на 36% быстрее, чем алгоритм Коэна-Сазерленда (реализация из [7, с. 66–67]). Однако в некоторых ситуациях, например, когда отсекатель имеет большие размеры, он оказался медленнее [9].

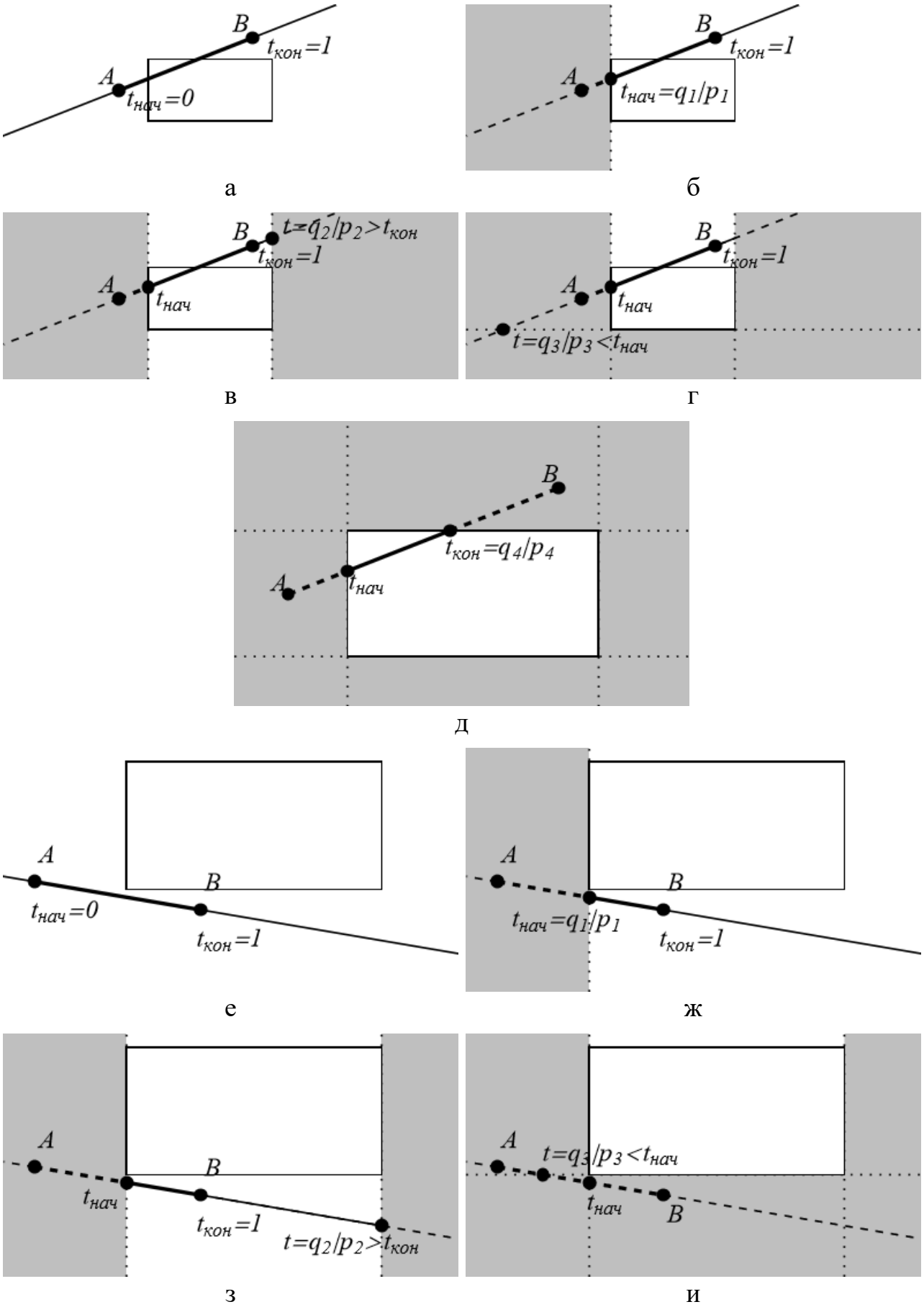


Рисунок 1.9 – Примеры работы алгоритма Лианга-Барски: а-д – для частично видимого отрезка, е-и – для отрезка, расположенного полностью снаружи окна

1.1.3. Алгоритм Кируса-Бека

Алгоритм, осуществляющий отсечение отрезка некоторым выпуклым многоугольником, был предложен М. Сайрусом и Дж. Беком в 1978 году [10].

Пусть есть некоторый отрезок AB с концами в точках с координатами $A(x_A, y_A)$ и $B(x_B, y_B)$, а также отсекатель в форме выпуклого полигона, заданный последовательностью вершин $C_1C_2...C_nC_1$, $n \geq 3$, имеющих координаты $C_i(x_i, y_i)$. Из определения выпуклого многоугольника ясно, что отрезок AB может пересечься с границей отсекателя не более, чем в двух точках.

Как и в алгоритме Лианга-Барски, отрезок AB задаётся в параметрической форме (1.4). Поиск пересечений осуществляется путём последовательного обхода рёбер отсекателя. Каждое ребро, по сути, само является отсекателем, которое отсекает полуплоскость, находящуюся за пределами полигона $C_1C_2...C_nC_1$ (Рисунок 1.10). В силу свойств выпуклых многоугольников при подобном отсечении внутренние точки исходного отсекателя отброшены не будут, ведь они все лежат по одну, неотсекаемую текущим ребром сторону. Таким образом, возможно последовательное отсекание отрезка рёбрами отсекателя $C_1C_2...C_nC_1$. Далее будем полагать, что имеет место левосторонний обход по периметру отсекателя.

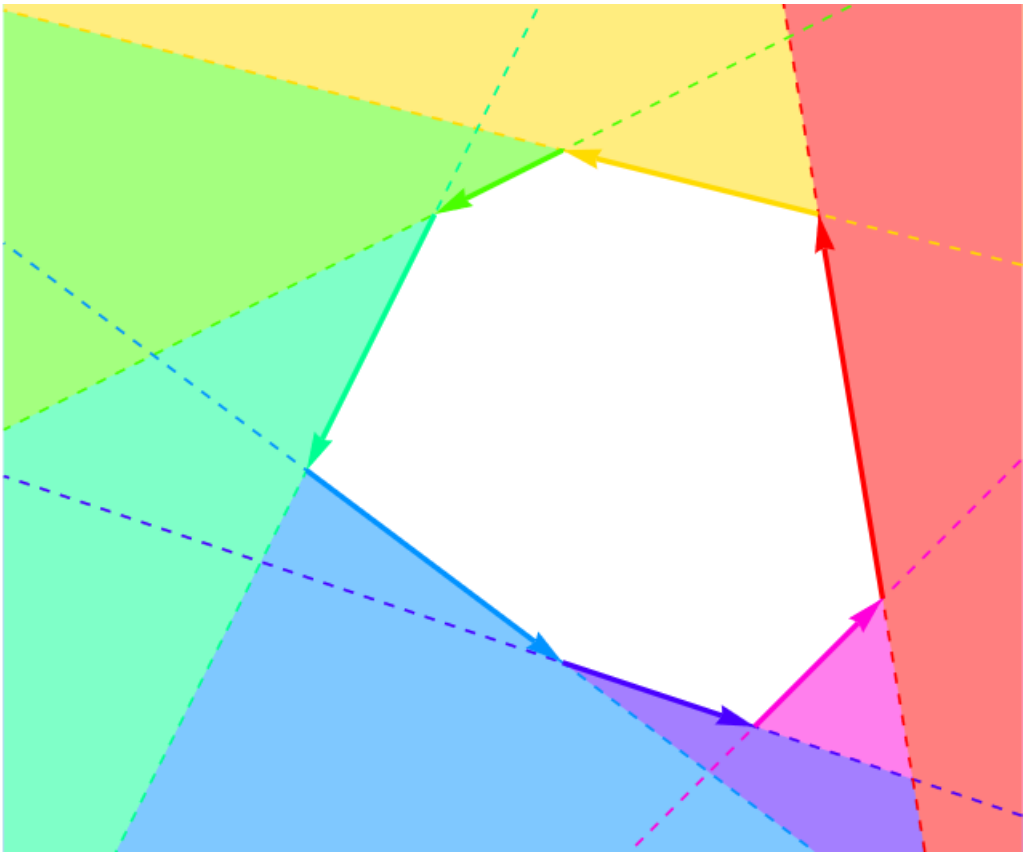


Рисунок 1.10 – Отсекатель в форме выпуклого полигона представи́м в виде совокупности отсекателей, порождаемых его рёбрами

Отсечение отрезка, задаваемого системой (1.4), ребром C_iC_{i+1}, $i=\overline{1,n}$, происходит следующим образом. Находится параметр *t*, соответствующий пересечению прямой C_iC_{i+1} с прямой, на которой лежит отсекаемый отрезок. Это можно сделать, например, задав общее уравнение прямой $C_iC_{i+1}:(y_{i+1}-y_i)(x-x_i)-(x_{i+1}-x_i)(y-y_i)=0$ и подставив в него координаты *x* и *y*, заданные системой (1.4):

$$(y_{i+1}-y_i)(x_A+(x_B-x_A)t-x_i)-(x_{i+1}-x_i)(y_A+(y_B-y_A)t-y_i)=0,$$
$$((y_{i+1}-y_i)(x_B-x_A)-(x_{i+1}-x_i)(y_B-y_A))t+$$
$$+(y_{i+1}-y_i)(x_A-x_i)-(x_{i+1}-x_i)(y_A-y_i)=0,$$
$$t=\frac{(x_{i+1}-x_i)(y_A-y_i)-(y_{i+1}-y_i)(x_A-x_i)}{(y_{i+1}-y_i)(x_B-x_A)-(x_{i+1}-x_i)(y_B-y_A)}=\frac{\overrightarrow{C_iC_{i+1}}\times\overrightarrow{C_iA}}{-\overrightarrow{C_iC_{i+1}}\times\overrightarrow{AB}}=\frac{\overrightarrow{AC_i}\times\overrightarrow{C_iC_{i+1}}}{\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}}.$$

Могут иметь место следующие случаи:

1. Параметр *t* не может быть вычислен в силу выполнения равенства $\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}=0$. Это эквивалентно коллинеарности векторов $\overrightarrow{AB}$ и $\overrightarrow{C_iC_{i+1}}$ – текущая сторона полигона параллельна отсекаемому отрезку (Рисунок 1.11, а-б). Тогда весь отрезок лежит по одну сторону от прямой C_iC_{i+1}, и либо **полностью отбрасывается** (когда отрезок справа от ребра, что эквивалентно неравенству $\overrightarrow{C_iC_{i+1}}\times\overrightarrow{C_iA}<0\Leftrightarrow\underline{\overrightarrow{AC_i}\times\overrightarrow{C_iC_{i+1}}<0}$, рисунок 1.11б), либо **остаётся неизменным** (когда отрезок слева от ребра $\Leftrightarrow$ $\underline{\overrightarrow{AC_i}\times\overrightarrow{C_iC_{i+1}}\geq 0}$ [1], рисунок 1.11а).

2. Отрезок **отсекается с начала** точкой A' при выполнении двух условий: ось *AB* направлена из правой полуплоскости от оси C_iC_{i+1} в левую, т.е. выполнено неравенство $\overrightarrow{C_iC_{i+1}}\times\overrightarrow{AB}>0\Leftrightarrow\underline{\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}<0}$ (Рисунок 1.11, г-д).

3. Отрезок **отсекается с конца** точкой B', если ось *AB* направлена из левой полуплоскости от оси C_iC_{i+1} в правую, т.е. $\underline{\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}>0}$ (Рисунок 1.11, ж-з).

Таким образом, для искомого отрезка имеет место неравенство $t\geq\frac{\overrightarrow{AC_i}\times\overrightarrow{C_iC_{i+1}}}{\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}}$, если $\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}<0$, либо $t\leq\frac{\overrightarrow{AC_i}\times\overrightarrow{C_iC_{i+1}}}{\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}}$, если $\overrightarrow{AB}\times\overrightarrow{C_iC_{i+1}}>0$. Их можно записать в виде одного неравенства:

$$p_it\leq q_i,\ i=\overline{1,n},\qquad(1.7)$$

[1] Будем полагать, что лежащие на границе внешнего отсекателя точки отсечению не подлежат

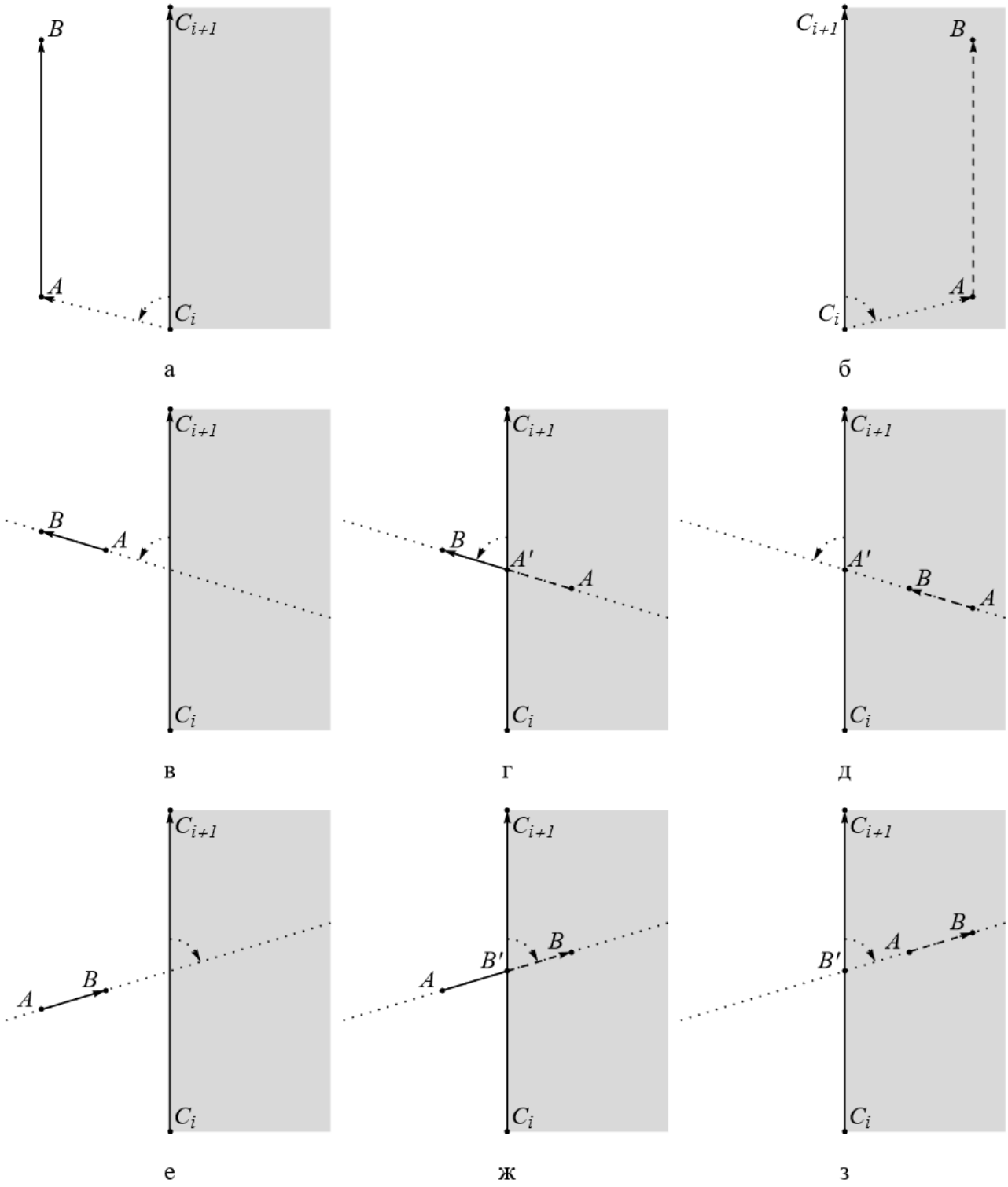


Рисунок 1.11 – Варианты отсечения отрезка *AB* ребром C_iC_{i+1} : а-б – при параллельности отрезка и ребра, в-д – при положительном угле между осями C_iC_{i+1} и *AB*, е-з – при отрицательном угле между ними. В левом столбце показаны случаи, когда отсечение не происходит, в среднем – отсечение части отрезка, в правом – полное отсечение.

где

$$p_i = \overrightarrow{AB} \times \overrightarrow{C_i C_{i+1}},\ q_i = \overrightarrow{AC_i} \times \overrightarrow{C_i C_{i+1}}. \qquad (1.8)$$

Заметим, что при $p_i = 0$ получается неравенство $q_i \geq 0$, при невыполнении которого происходит полное отсечение отрезка (Рисунок 1.11б), а иначе отрезок остаётся неизменным (Рисунок 1.11а). Таким образом, неравенства (1.7) учитывают все случаи, описанные выше.

Итак, **алгоритм Кируса-Бека** отсечения отрезка *AB* внешним отсекателем в форме выпуклого полигона $C_1C_2...C_nC_1$ с левосторонним обходом можно описать следующей последовательностью команд:

1. Задаются начальные значения для параметров, отвечающих началу и концу текущего отрезка: $t_{\text{нач}} := 0,\ t_{\text{кон}} := 1$, а также итерационная переменная $i := 1$.
2. По формулам (1.8) вычисляются значения p_i и q_i. При этом может возникнуть ровно одна из ситуаций:
   - 2а. При выполнении равенства $p_i = 0$ необходимо проверить неравенство $q_i < 0$. В случае его выполнения алгоритм немедленно завершает свою работу и возвращает пустой результат – отрезок **полностью отсекается**. В противном случае можно сразу перейти к шагу 4 алгоритма.
   - 2б. Если $p_i > 0$, то происходит **отсечение отрезка с конца**: $t_{\text{кон}} := := \min\{q_i / p_i, t_{\text{кон}}\}$.
   - 2в. Если $p_i < 0$, то отрезок **отсекается с начала**: $t_{\text{нач}} := \max\{q_i / p_i, t_{\text{нач}}\}$.
3. Если оказалось справедливым неравенство $t_{\text{нач}} > t_{\text{кон}}$, то алгоритм немедленно возвращает пустой результат.
4. Если $i = n$, то алгоритм возвращает отрезок с концами в точках, которые получаются при подстановке значений $t_{\text{нач}}$ и $t_{\text{кон}}$ в систему (1.4). Иначе алгоритм продолжается: $i := i + 1$, переход к шагу 2.

Таким образом, этот алгоритм можно рассматривать как обобщение алгоритма Лианга-Барски на случай произвольного выпуклого отсекателя.

Замечание 1. В случае правостороннего обхода по периметру отсекателя в неравенстве (1.7) следует полагать знак «больше либо равно», а подчёркнутые выше слова и неравенства заменить противоположными по смыслу. Если вместо внешнего отсекателя рассматривается внутренний, то приведённый выше алгоритм возвращает не усечённый отрезок, а часть отрезка, подлежащая отсечению.

Замечание 2. Вычислительная сложность алгоритма Кируса-Бека равна $O(n)$, так как, во-первых, каждое ребро отсекателя рассматривается не более

одного раза, во-вторых, максимальное число операций, выполняемых для каждого ребра, не зависит от входных данных.

1.1.4. Алгоритм Николла-Ли-Николла

Ещё один алгоритм отсечения отрезка *AB* регулярным окном предложили Т. М. Николла, Д. Т. Ли и Р. А. Николла в 1987 году. Этот алгоритм избегает вычисления пересечений отрезка с прямыми, которые не являются концами искомого отрезка, и использует аффинные преобразования плоскости [11].

Идея этого алгоритма заключается в следующем. Через $C_1(x_{\min}, y_{\max})$, $C_2(x_{\max}, y_{\max})$, $C_3(x_{\min}, y_{\min})$, $C_4(x_{\max}, y_{\min})$ обозначим вершины регулярного отсекателя (левую верхнюю, правую верхнюю, левую нижнюю и правую нижнюю соответственно). Тогда существует всего 9 случаев расположения конца *A* относительно окна $C_1C_2C_4C_3$, описываемого системой неравенств $x_{\min} \le x \le x_{\max}$ и $y_{\min} \le y \le y_{\max}$. Положение же конца *B* рассматривается как относительно окна, так и относительно конца *A*. Это позволяет вычислять ровно столько точек пересечений, сколько необходимо для определения искомого отрезка.

Разумеется, возникает необходимость каким-то образом обобщить эти ситуации для того, чтобы избежать чрезмерного ветвления алгоритма, что влечёт высокий риск ошибок при его программной реализации. Этого можно добиться при помощи следующих преобразований:

- поворот плоскости на 90°, 180° и 270° по часовой стрелке вокруг начала координат:

$$\varphi_{-\pi/2}\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} y \\ -x \end{pmatrix}, \tag{1.9}$$

$$\varphi_{\pi}\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} -x \\ -y \end{pmatrix}, \tag{1.10}$$

$$\varphi_{\pi/2}\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} -y \\ x \end{pmatrix}, \tag{1.11}$$

- симметрические отражения относительно прямой $x = -y$ и оси Ox:

$$\varphi_{xy}\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} -y \\ -x \end{pmatrix}, \tag{1.12}$$

$$\varphi_x\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} x \\ -y \end{pmatrix}. \qquad (1.13)$$

Таким образом, все случаи разделяются на следующие три принципиально разных категории:

- точка *A* находится левее и выше окна (вариант I),
- точка *A* находится левее, но не выше и не ниже окна (вариант II),
- точка *A* находится внутри окна (вариант III).

В зависимости от расположения точки *B* относительно границ окна и прямых, проведённых из *A* через углы окна, определяется, с какими границами имеет место пересечение отрезка *AB*, после чего происходит вычисление этих пересечений. При этом если на некотором этапе вычислений имело место применение хотя бы одного из преобразований (1.9) – (1.13), то после всех вычислений необходимо осуществить обратные преобразования.

Приведение к трём базовым вариантам. Все перечисленные выше случаи получаются исходя из следующих соображений. Для точки *A* существует три возможности:

- точка *A* находится левее окна: $x_A < x_{\min}$,
- точка *A* находится правее окна: $x_A > x_{\max}$,
- точка *A* не находится ни левее, ни правее окна: $x_{\min} \le x_A \le x_{\max}$.

Первый и третий случаи в алгоритме Николла-Ли-Николла рассматриваются отдельно. Второй же путём поворота на 180° (1.10) приводится к первому.

Если **точка *A* расположена левее окна**, то при нахождении точки *B* левее окна ($x_B < x_{\min}$) отрезок целиком лежит левее окна и подлежит полному отсечению. В противном случае ($x_B \ge x_{\min}$) опять возникают различные варианты:

- точка *A* находится выше окна: $y_A > y_{\max}$,
- точка *A* находится ниже окна: $y_A < y_{\min}$,
- точка *A* не находится ни выше, ни ниже окна: $y_{\min} \le y_A \le y_{\max}$.

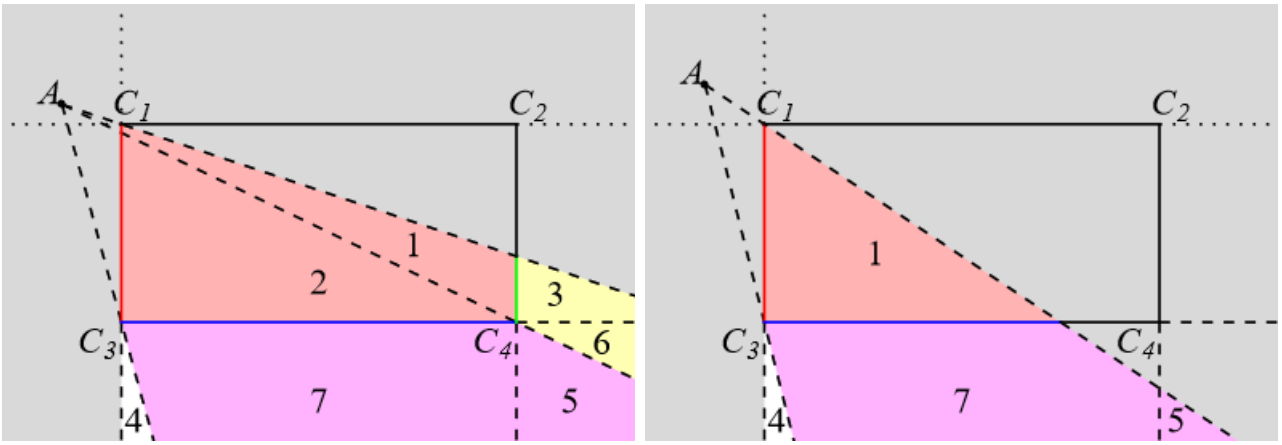
Первый и третий случаи дают соответственно варианты I и II. Второй же сводится к варианту I путём симметрического отражения (1.13).

Если же **точка *A* не расположена ни левее, ни правее окна**, то может иметь место ровно один из следующих трёх случаев:

- точка *A* находится выше окна: $y_A > y_{\max}$,
- точка *A* находится ниже окна: $y_A < y_{\min}$,
- точка *A* не находится ни выше, ни ниже окна: $y_{\min} \le y_A \le y_{\max}$.

Из первых двух случаев путём поворотов (1.11) и (1.9) соответственно получается вариант II. После этого нужно проверить выполнение неравенства $x'_B \ge x'_{\min}$, где x'_B и $x'_{\min}$ – значения абсциссы точки *B* и левой границы окна, получившиеся после таких поворотов. При невыполнении этого неравенства имеет место полное отсечение отрезка. Наконец, третий случай даёт вариант III.

Обработка трёх базовых вариантов. Вариант I удовлетворяет неравенствам $x_A < x_{\min}$, $y_A > y_{\max}$ и $x_B \geq x_{\min}$. Здесь вначале необходимо проверить, не находится ли точка B выше окна. Если справедливо неравенство $y_B > y_{\max}$, то весь отрезок лежит выше окна и полностью отсекается – алгоритм завершается и даёт пустой результат. Если же $y_B \leq y_{\max}$, то проверяется, по какую сторону от оси AC_1 расположена точка B. Это определяется знаком угла $\angle\left(\overrightarrow{AC_1}, \overrightarrow{AB}\right)$, равным знаку векторного произведения $\overrightarrow{AC_1} \times \overrightarrow{AB}$. Случай $\overrightarrow{AC_1} \times \overrightarrow{AB} > 0$ сводится к $\overrightarrow{AC_1} \times \overrightarrow{AB} \leq 0$ при помощи симметрического отражения (1.12). После этого преобразования (если оно необходимо) выполняется следующий алгоритм (на рисунке 1.12 красным, жёлтым и пурпурным цветами обозначены соответственно области, при попадании точки B в которые происходит отсечение отрезка AB только слева, слева и справа, слева и снизу):



а б

Рисунок 1.12 – Обработка варианта I алгоритма Николла-Ли-Николла:
а – луч AC_1 пересекает правую границу окна,
б – луч AC_1 пересекает нижнюю границу окна

1. Если $y_B \geq y_{\min}$, то происходит отсечение левой границей по формуле (1.2) при $x_0 = x_{\min}$ (области 1–3).
   1a. После этого проверяется выполнение неравенства $x_B > x_{\max}$. При его выполнении (область 3) происходит отсечение отрезка AB правой границей окна по той же формуле при $x_0 = x_{\max}$.

   Таким образом, отсечение отрезка завершено, и осталось только выполнить преобразования, обратные тем из (1.9) – (1.13), которые были применены ранее.
2. Иначе необходимо проверить, по какую сторону расположена точка B относительно оси AC_3. Если $\overrightarrow{AC_3} \times \overrightarrow{AB} \leq 0$, то отрезок отсекается полностью (область 4). Алгоритм возвращает пустой результат.
3. Проверяется неравенство $x_B > x_{\max}$. Если оно оказалось верным, то далее необходимо проверить, по какую сторону расположена точка B

от оси AC_4. При этом может получиться ровно один из следующих случаев:

3а. Точка B расположена по правую сторону от этой оси: $\overrightarrow{AC_4} \times \overrightarrow{AB} \le 0$ (область 5). Тогда происходит отсечение отрезка нижней границей окна по формуле (1.1) при $y_0 = y_{\min}$.

3б. Точка B расположена по левую сторону от этой оси: $\overrightarrow{AC_4} \times \overrightarrow{AB} > 0$ (область 6). В этом случае происходит отсечение отрезка правой границей окна по формуле (1.2) при $x_0 = x_{\max}$.

Иначе происходит отсечение нижней границей окна (область 7).

4. Происходит отсечение отрезка левой границей окна по формуле (1.2) при $x_0 = x_{\min}$. С учётом шага 3 получили два конца искомого отрезка, таким образом, отсечение отрезка AB завершено, и как и в шаге 1, нужно проделать преобразования, обратные ранее осуществлённым.

Следует отметить, что алгоритм не проверяет, который из двух случаев имеет место: представленный на рисунке 1.12а или же на рисунке 1.12б. Поэтому в случае, когда луч AC_1 пересекает нижнюю границу окна (Рисунок 1.12б), осуществляются лишние вычисления. Но для их предотвращения пришлось бы проверять ещё больше условий, что привело бы к ещё более сильному ветвлению алгоритма.

Вариант II имеет место при выполнении неравенств $x_A < x_{\min}$, $y_{\min} \le y_A \le y_{\max}$, $x_B \ge x_{\min}$. Для точки B существует три варианта её расположения относительно окна:

- точка B не находится ни выше, ни ниже окна: $y_{\min} \le y_B \le y_{\max}$.
- точка B находится выше окна: $y_B > y_{\max}$,
- точка B находится ниже окна: $y_B < y_{\min}$,

Первый случай обрабатывается согласно следующему алгоритму (Рисунок 1.13, области 1–3):

1. Происходит отсечение левой границей окна по формуле (1.2) при $x_0 = x_{\min}$. Получается один из концов искомого отрезка.

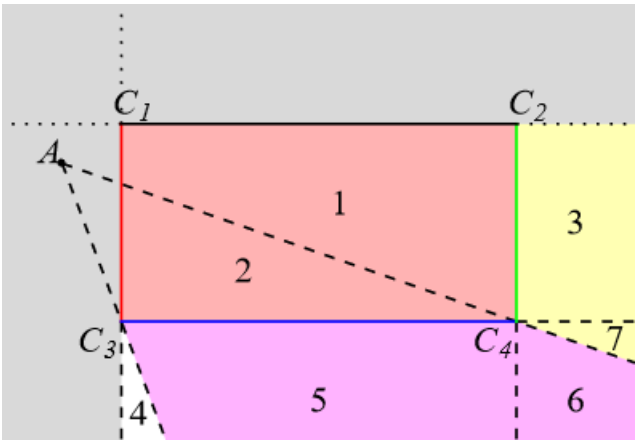


Рисунок 1.13 – Обработка варианта II

2. При выполнении неравенства $x_B > x_{\max}$ (область 3) выполняется вычисление другого конца искомого отрезка как пересечение отрезка AB с правой границей окна по этой же формуле при $x_0 = x_{\max}$. В противном случае (области 1–2) точка B сама является этим другим концом.
3. Выполняются преобразования, обратные тем из (1.9) – (1.13), которые были применены для приведения исходного отрезка к варианту II.
4. Алгоритм возвращает координаты концов, полученных на шагах 1 и 2 и преобразованных с учётом шага 3.

Второй случай приводится к третьему симметрическим отражением (1.13). После применения этого преобразования (если оно необходимо) выполняется следующий алгоритм (области 4–7 на рисунке 1.13):

1. Происходит проверка, по какую сторону расположена точка B относительно оси AC_3. Если $\overrightarrow{AC_3} \times \overrightarrow{AB} \le 0$, то отрезок отсекается полностью (область 4). Алгоритм возвращает пустой результат.
2. Если $x_B \le x_{\max}$ (область 5), то имеет место отсечение отрезка нижней границей по формуле (1.1) при $y_0 = y_{\min}$.
3. В противном случае необходимо проверить, по какую сторону расположена точка B относительно оси AC_4. Может иметь место ровно один из двух вариантов:
   3а. Точка B расположена по правую сторону от этой оси: $\overrightarrow{AC_4} \times \overrightarrow{AB} \le 0$ (область 6). Тогда происходит отсечение отрезка нижней границей окна по формуле (1.1) при $y_0 = y_{\min}$.
   3б. Точка B расположена по левую сторону от этой оси: $\overrightarrow{AC_4} \times \times \overrightarrow{AB} > 0$ (область 7). В этом случае происходит отсечение отрезка правой границей окна по формуле (1.2) при $x_0 = x_{\max}$.
4. Происходит отсечение отрезка левой границей окна по формуле (1.2) при $x_0 = x_{\min}$.
5. Как и в предыдущих алгоритмах, осуществляются обратные преобразования. Возвращаются два полученных конца искомого отрезка.

Вариант III соответствует ситуации, когда выполняются неравенства $x_{\min} \le x_A \le x_{\max}$ и $y_{\min} \le y_A \le y_{\max}$. Тогда все варианты расположения точки B относительно окна также можно свести к вариантам I, II и III. Отсюда имеем следующий алгоритм (Рисунок 1.14):

I-1. Проверяется, по какую сторону от оси AC_1 расположена точка B.

I-2. Если $\overrightarrow{AC_1} \times \overrightarrow{AB} < 0$ (область 1) то происходит отсечение отрезка верхней границей окна по формуле (1.1) при $y_0 = y_{\max}$. Иначе (область 2) осуществляется его отсечение левой границей по формуле (1.2) при $x_0 = x_{\min}$.

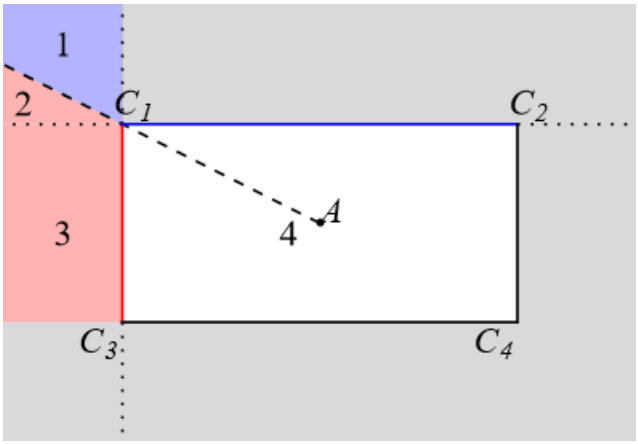


Рисунок 1.14 – Обработка варианта III

I-3. При необходимости применяются обратные преобразования. Точка A и вычисленная на шаге I-2 точка возвращаются как концы искомого отрезка.

II-1. (Область 3.) Вычисляется точка пересечения отрезка с левой границей окна по формуле (1.2) при $x_0 = x_{\min}$.

II-2. При необходимости применяются обратные преобразования. Точка A и вычисленная на шаге II-1 точка возвращаются как концы искомого отрезка.

III. (Область 4.) Никаких отсечений не происходит. Алгоритм возвращает точки A и B.

Замечание 1. Для запоминания всех проделанных аффинных преобразований может быть использована структура данных **стек**. Например, каждое из преобразований (1.9) – (1.13) можно обозначить числами от 1 до 5 и эти числа заносить в стек при выполнении соответствующих преобразований. Затем, после того как найдены концы искомого отрезка, проделывать обратные преобразования в порядке, обратном добавлению чисел в стек: $\varphi_{-\pi/2}^{-1} = \varphi_{\pi/2}$, $\varphi_{\pi/2}^{-1} = \varphi_{-\pi/2}$, $\varphi_{\pi}^{-1} = \varphi_{\pi}$, $\varphi_{xy}^{-1} = \varphi_{xy}$, $\varphi_{x}^{-1} = \varphi_{x}$.

Замечание 2. Как и алгоритмы Коэна-Сазерленда и Лианга-Барски, алгоритм Николла-Ли-Николла выполняется за время $O(1)$. Однако благодаря уменьшению числа операций (особенно сравнения и деления) он работает несколько быстрее [11].

1.1.5. Алгоритм Сазерленда-Ходжмана

Алгоритм, осуществляющий отсечение самонепересекающегося полигона выпуклым окном, предложен А. Э. Сазерлендом и Г. Ходжманом в 1974 году [12]. Суть данного алгоритма, как и алгоритма Кируса-Бека, заключается в последовательном отсечении заданного полигона $A_1A_2...A_lA_1$ сторонами выпуклого отсекателя $C_1C_2...C_nC_1$ (см. рисунок 1.10).

Предположим, что у отсекателя имеет место левосторонний обход. Координаты вершин текущего (частично усечённого некоторыми рёбрами отсекателя) полигона обозначим через $A'_i(x_i, y_i),\ i = \overline{1, l_1}$, а координаты вершин отсекателя – $C_k(\tilde{x}_k, \tilde{y}_k),\ k = \overline{1, n}$. Для каждого ребра отсекателя $C_k C_{k+1}$ последовательно рассматриваются рёбра текущего полигона $A'_1 A'_2 ... A'_{l_1} A'_1$. Для каждой его вершины несложно определить, должна ли она отсекаться на данном шаге: вершина A'_i расположена слева от ребра $C_k C_{k+1}$ тогда и только тогда, когда выполняется неравенство $\overrightarrow{C_k C_{k+1}} \times \overrightarrow{C_k A'_i} \geq 0$. Могут быть следующие четыре случая:

1. Ребро $A'_i A'_{i+1}$ полностью лежит по левую сторону от ребра $C_k C_{k+1}$ (Рисунок 1.15а). Тогда вершина A'_{i+1} должна остаться в списке вершин усечённого полигона.

2. Вершина A'_i лежит по левую сторону от ребра $C_k C_{k+1}$, а A'_{i+1} – по правую (Рисунок 1.15б). Тогда вместо вершины A'_{i+1} в усечённом полигоне будет вершина, равная пересечению отрезка $A'_i A'_{i+1}$ с прямой $C_k C_{k+1}$. Она может быть вычислена, например, как пересечение двух прямых, заданных по двум точкам:

$$\begin{cases} \dfrac{x - x_i}{x_{i+1} - x_i} = \dfrac{y - y_i}{y_{i+1} - y_i}, \\ \dfrac{x - \tilde{x}_k}{\tilde{x}_{k+1} - \tilde{x}_k} = \dfrac{y - \tilde{y}_k}{\tilde{y}_{k+1} - \tilde{y}_k}. \end{cases}$$

Эту систему можно решить, например, по правилу Крамера. В результате получим точку пересечения S_i со следующими координатами:

$$\overline{x}_i = \frac{\Delta_x}{\Delta},\ \overline{y}_i = \frac{\Delta_y}{\Delta}, \tag{1.14}$$

где

$$\Delta_x = \begin{vmatrix} x_i y_{i+1} - x_{i+1} y_i & x_i - x_{i+1} \\ \tilde{x}_k \tilde{y}_{k+1} - \tilde{x}_{k+1} \tilde{y}_k & \tilde{x}_k - \tilde{x}_{k+1} \end{vmatrix}, \Delta_y = \begin{vmatrix} y_{i+1} - y_i & x_i y_{i+1} - x_{i+1} y_i \\ \tilde{y}_{k+1} - \tilde{y}_k & \tilde{x}_k \tilde{y}_{k+1} - \tilde{x}_{k+1} \tilde{y}_k \end{vmatrix},$$
$$\Delta = \begin{vmatrix} y_{i+1} - y_i & x_i - x_{i+1} \\ \tilde{y}_{k+1} - \tilde{y}_k & \tilde{x}_k - \tilde{x}_{k+1} \end{vmatrix}. \tag{1.15}$$

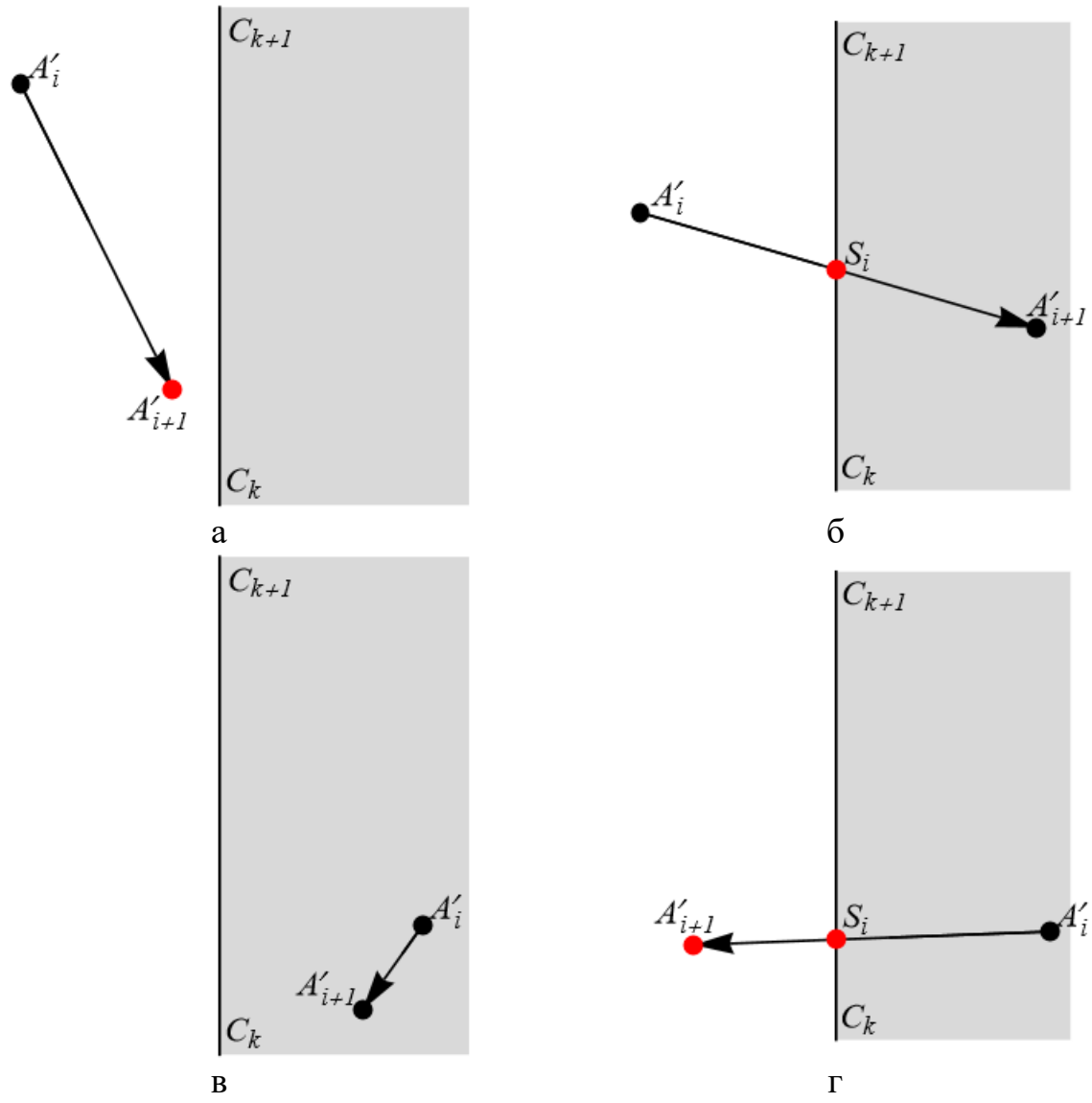


Рисунок 1.15 – Варианты размещения ребра отсекаемого полигона относительно отсекателя

3. Ребро $A'_iA'_{i+1}$ полностью лежит по правую сторону от ребра C_kC_{k+1} (Рисунок 1.15в). Тогда оно подлежит полному отсечению, и никакие из этих вершин не будут вершинами усечённого полигона.

4. Вершина A'_i лежит по правую сторону от ребра C_kC_{k+1}, а A'_{i+1} – по левую (Рисунок 1.15г). Тогда, во-первых, в усечённом полигоне появится вершина S_i, координаты которой вычисляются по формулам (1.14) и (1.15), во-вторых, точка A'_{i+1} также является его вершиной.

На рисунке 1.15 красным цветом обозначены точки, подлежащие внесению в список новых вершин усечённого полигона. В случаях 1 и 2 начальная точка текущего ребра не учитывается во избежание дублирований точек в списке.

Таким образом, по мере обхода текущего полигона по рёбрам $A'_1A'_2$, $A'_2A'_3$, …, $A'_{l_1-1}A'_{l_1}$, $A'_{l_1}A'_1$ составляется список из новых вершин A''_1, A''_2, …, A''_{l_2}, образующих полигон, усечённый текущим ребром C_kC_{k+1}. **Алгоритм Сазерленда-Ходжмана** для самонепересекающегося полигона $A_1A_2...A_lA_1$ и

выпуклого внешнего отсекателя $C_1C_2...C_nC_1$, с левосторонним обходом каждый, выглядит следующим образом:

1. Отсечение начинается с ребра C_1C_2 : $k := 1$.
2. Инициализация списка вершин текущего полигона: $\mathcal{A} := \{A_1, A_2, ..., A_l\}$.
3. Обозначим вершины из списка $\mathcal{A}$ через A'_1, A'_2, …, A'_{l_1}. Для текущего ребра отсекателя инициализируется список вершин усечённого полигона: $\mathcal{A}' := \varnothing$.
4. Обход текущего полигона начинается с ребра $A'_1A'_2$: $i := 1$.
5. Определяется, находится ли точка A'_1 слева либо справа от текущего отсекателя. Для этого заведём логическую переменную: $V := \underline{\overrightarrow{C_kC_{k+1}} \times \overrightarrow{C_kA'_1} \geq 0}$. В дальнейшем она будет хранить информацию о том, находится ли начало текущего ребра внутри отсекателя или нет.
6. Выполняется проверка расположения концевой вершины A'_{i+1} текущего ребра относительно ребра C_kC_{k+1} : $V' := \underline{\overrightarrow{C_kC_{k+1}} \times \overrightarrow{C_kA'_{i+1}} \geq 0}$. При этом может иметь место ровно один из следующих вариантов:
   - 6а. Если $V = истина$ и $V' = истина$, то в список $\mathcal{A}'$ добавляется вершина A'_{i+1} (Рисунок 1.15а).
   - 6б. Если $V = истина$ и $V' = ложь$, то в список $\mathcal{A}'$ добавляется вершина S_i (Рисунок 1.15б).
   - 6в. Если $V = ложь$ и $V' = истина$, то в список $\mathcal{A}'$ добавляются две вершины: вначале S_i, потом A'_{i+1} (Рисунок 1.15г).
7. Если $i < l_1$, то происходит переход к следующему ребру полигона: переопределяются переменные $i := i + 1$, $V := V'$, а потом происходит переход к шагу 6 алгоритма. Иначе – переопределение текущего полигона: $\mathcal{A} := \mathcal{A}'$, а после этого – переход к шагу 8.
8. Если $k < n$ и список $\mathcal{A}$ не пустой, то происходит переход к следующему ребру отсекателя: $k := k + 1$, затем переход к шагу 3. Иначе отсечение завершается, алгоритм возвращает полигон с вершинами $\mathcal{A}$.

При правостороннем обходе отсекателя $C_1C_2...C_nC_1$ подчёркнутые слова и неравенства следует заменить на противоположные по смыслу.

На рисунках 1.16, а-д, синими точками обозначены вершины исходного полигона и промежуточных результатов отсечений, на рисунках 1.16, б-е, красной линией – текущее ребро отсекателя.

Замечание 1. Для хранения списков вершин $\mathcal{A}$ и $\mathcal{A}'$ целесообразно использовать структуру данных **массив**, так как именно он позволяет добавлять новые элементы в конец списка за время $O(1)$.

Замечание 2. Несложно проверить, что алгоритм работает при отсутствии пересечений границ исходного полигона и отсекателя, т.е. в случаях, когда

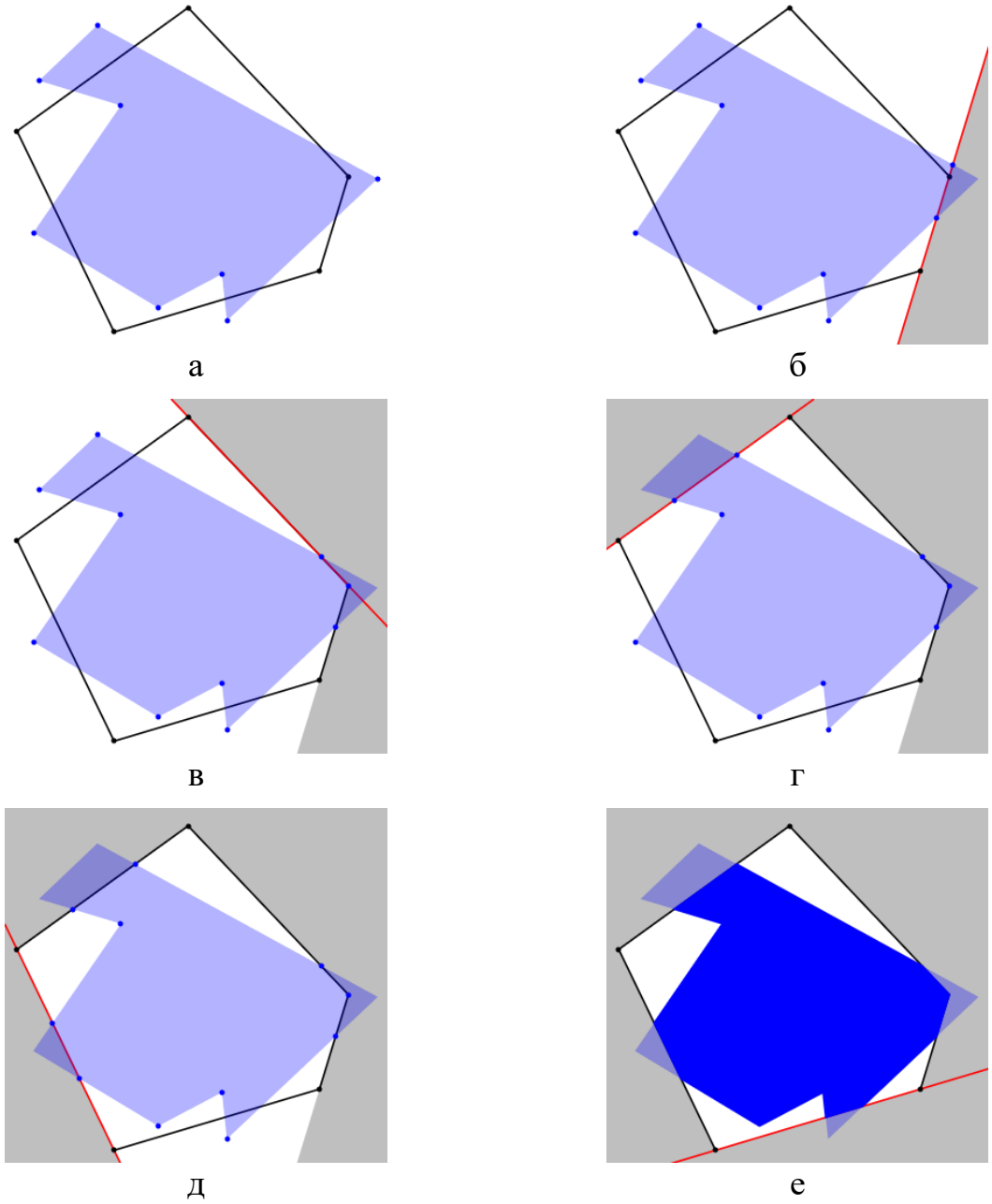


Рисунок 1.16 – Пример работы алгоритма Сазерленда-Ходжмана: а – исходный полигон и отсекатель, б-д – отсечение полигона рёбрами отсекателя, е – отсечение последним ребром и окончательный результат

полигон $A_1A_2...A_iA_1$ целиком лежит внутри либо снаружи отсекателя $C_1C_2...C_nC_1$. В первом случае список $\mathcal{A}$ не будет изменяться ни на какой итерации, во втором – в результате очередной итерации список $\mathcal{A}$ окажется пустым.

Замечание 3. Оценивая сложность алгоритма, рассмотрим количество вершин искомого полигона. У каждого ребра исходного полигона не более двух точек (две его вершины, либо одна из вершин и пересечение с ребром отсекателя, либо два таких пересечения) могут служить вершинами результата отсечения. Кроме того, вершинами усечённого полигона могут оказаться вершины отсекателя, охваченные рёбрами исходного полигона. Следовательно,

число вершин искомого полигона, а значит, и число вершин текущего полигона после каждой итерации можно оценить асимптотикой $O(2l + n) = O(l + n)$. Таким образом, сложность алгоритма Сазерленда-Ходжмана оценивается асимптотикой $O(ln + n^2)$. Обычно на практике достигается сложность $O(ln)$, так как число вершин отсекаемого полигона, как правило, сильно превышает число вершин отсекателя. На рисунке 1.17 представлен случай достижения максимального числа вершин при $l = 24$, $n = 10$, равный $\frac{3}{2}l + n = 46$.

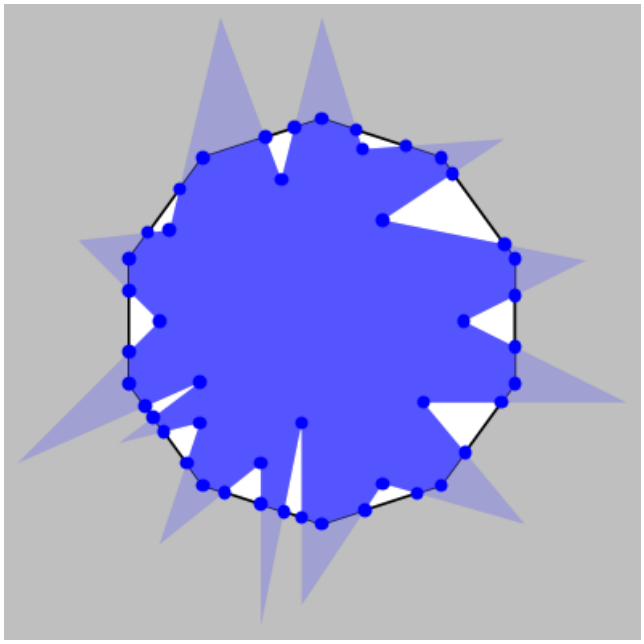


Рисунок 1.17 – Полигон, каждое из рёбер которого пересекает границы отсекателя, и охватывающий все его вершины

Замечание 4. Недостатком алгоритма Сазерленда-Ходжмана является то, что он возвращает последовательность вершин, которые, вообще говоря, могут задавать самопересекающийся полигон. На рисунке 1.18 у усечённого полигона $A'_1A'_2A'_3A'_4A'_5A'_6A'_7A'_8A'_9A'_1$ ребро $A'_4A'_5$ лежит на ребре $A'_9A'_1$. Эта проблема может быть решена при помощи некоторых вспомогательных алгоритмов [12, приложение В].

Кроме того, существует алгоритм, позволяющий из самопересекающегося усечённого полигона получить список вершин для каждого самонепересекающегося полигона, являющегося частью отсечения. Его основная идея – обработка рёбер, лежащих на рёбрах отсекателя. Они, вообще говоря, могут накладываться или перекрываться друг с другом произвольным образом (Рисунок 1.19б). Для того, чтобы избавиться от самопересечений вдоль того или иного ребра отсекателя, для каждой вершины на нём необходимо переназначить соседей таким образом, чтобы новые рёбра соединяли вершины, расположенные по соседству на ребре отсекателя, начиная с крайних вершин (Рисунок 1.19в).

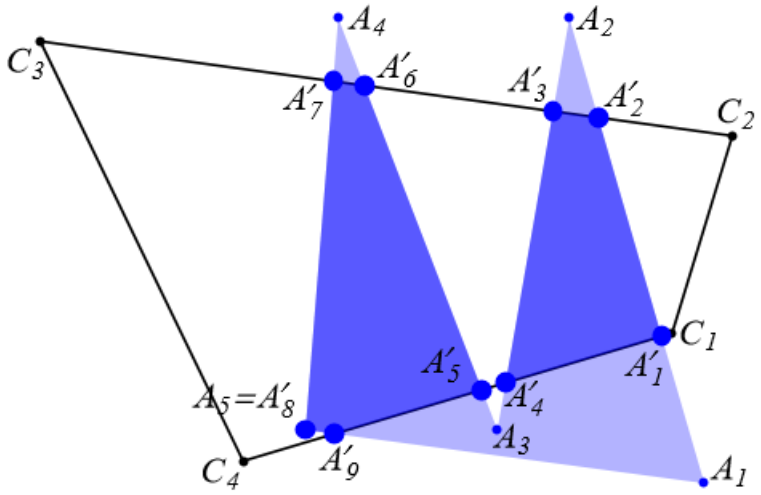
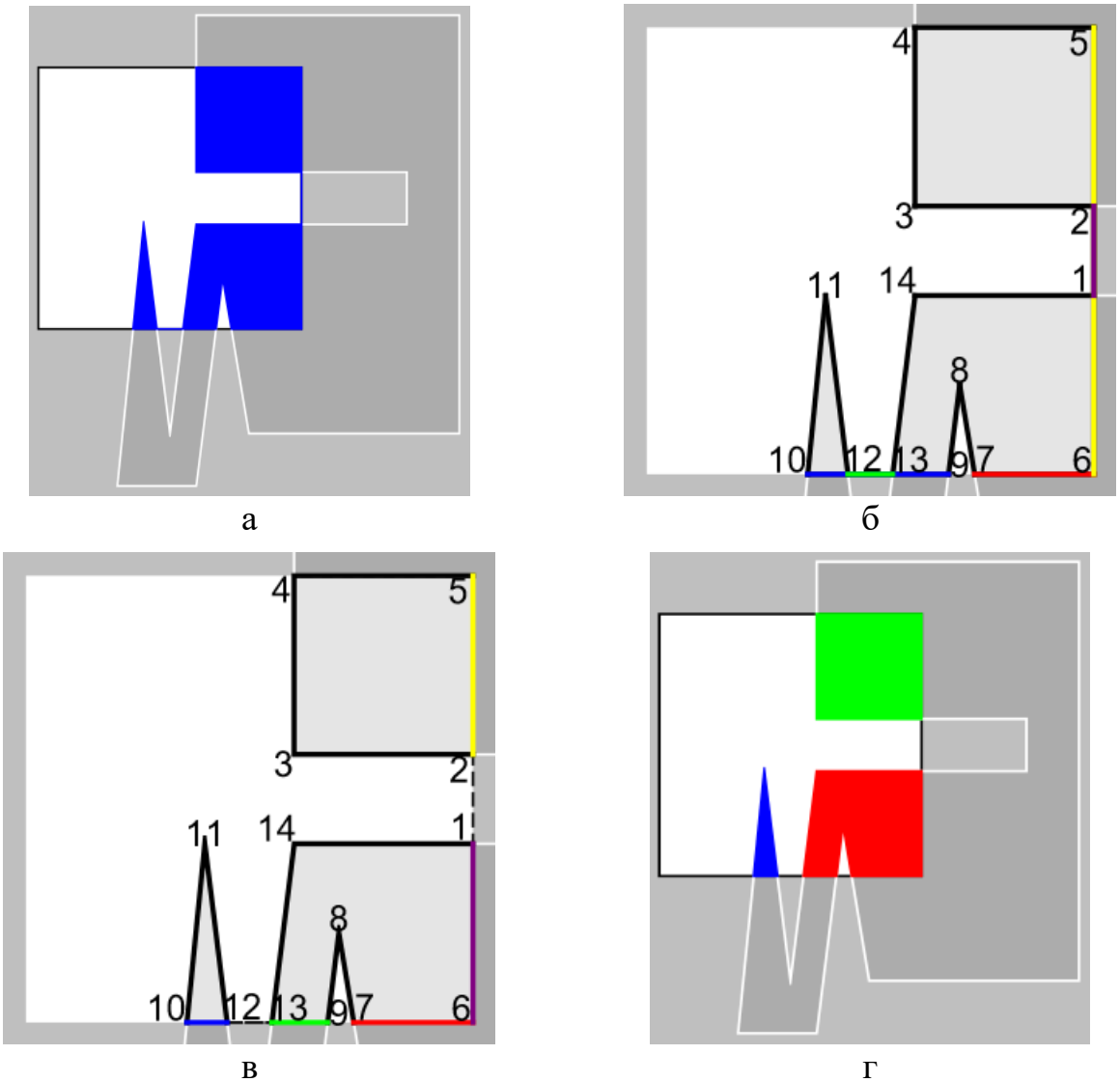


Рисунок 1.18 – Если результатом отсечения является несвязная область, то алгоритм Сазерленда-Ходжмана возвращает самопересекающийся полигон



а б в г

Рисунок 1.19 – Обработка полигона после отсечения алгоритмом Сазерленда-Ходжмана: а – результат работы алгоритма, б – накладывающиеся рёбра этого полигона, в – редактирование рёбер, г – разбиение полигона на самонепересекающиеся части

Покажем, что подобное редактирование рёбер даст новые рёбра, ограничивающие самонепересекающиеся полигоны (в случае, если изначально имел место самонепересекающийся полигон, и полученный полигон является минимальным). ◆ Для этого рассмотрим некоторое ребро отсекателя, на котором по порядку расположены вершины A'_1, A'_2, …, A'_{2n} n рёбер усечённого полигона, попавших на ребро отсекателя. Ясно, что каждая из этих точек является вершиной только для одного из таких рёбер, так как в противном случае имели бы место фиктивные вершины. Следовательно, для каждой из точек A'_i найдётся смежная вершина B_i, не лежащая на ребре отсекателя. Рассматривая вершину A'_1, имеем, что всякий отрезок $A'_1A'_i$, $i = \overline{2,2n}$, проходит через вершину A'_2. Однако если $i \neq 2$, то непременно получается пересечение несмежных рёбер $A'_1A'_i$ и A'_2B_2 (Рисунок 1.20). Значит, вершины A'_1 и A'_2 необходимо соединить ребром. Аналогично рассматривается вершина A'_3, которая становится новой крайней вершиной и подлежит соединению с A'_4, A'_5 – с A'_6, …, а A'_{2n-1} соединяется с вершиной A'_{2n}. ■

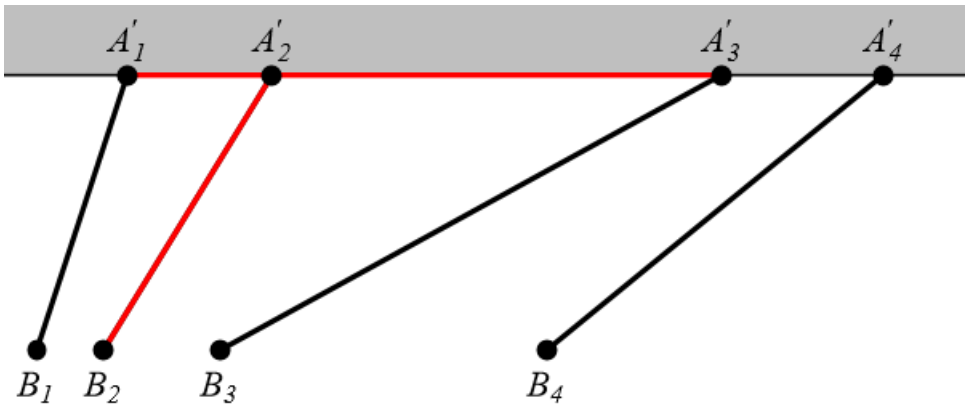
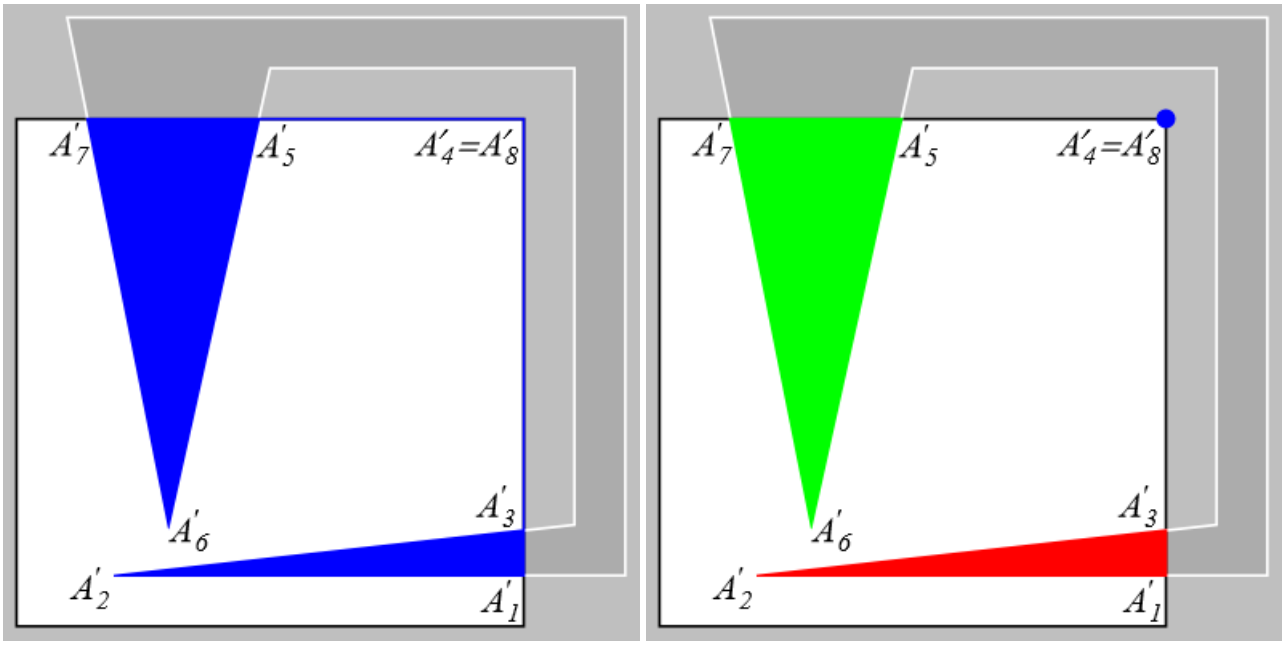


Рисунок 1.20 – Пересечение несмежных рёбер при неверном выборе смежных вершин

Отметим, что в этом доказательстве не рассмотрен случай, когда имеют место совпадающие вершины, образующиеся в результате алгоритма Сазерленда-Ходжмана в вершинах отсекателя (Рисунок 1.21а). Тем не менее, и в такой ситуации данный подход даёт верные последовательности вершин, при этом имеют место фиктивные изолированные рёбра, в конечном итоге подлежащие удалению (Рисунок 1.21б).

Исходя из этих рассуждений можно сформулировать **алгоритм редактирования усечённого полигона** $A'_1A'_2...A'_{l'}A'_1$ при выпуклом отсекателе $C_1C_2...C_nC_1$:

1. Происходит минимизация полигона $A'_1A'_2...A'_{l'}A'_1$: для всякого $i = \overline{1,l'}$ при выполнении $\overrightarrow{A'_{i-1}A'_i} \parallel \overrightarrow{A'_iA'_{i+1}}$ вершина A'_i удаляется из списка вершин исходного полигона. Обозначим новый полигон через $B_1B_2...B_{l''}B_1$.



а б

Рисунок 1.21 – Обработка полигона с совпадающими вершинами, расположенными в вершине отсекателя

2. Если после минимизации число вершин полигона стало меньше 3, то алгоритм немедленно завершается с возвращением пустого списка.
3. Определяются списки $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n\}$, $\mathcal{B}_i := \varnothing$.
4. Для каждого ребра отсекателя $C_i C_{i+1}$, $i = \overline{1,n}$, необходимо проверить, какие из рёбер $B_j B_{j+1}$, $j = \overline{1,l''}$, лежат на нём. Для этого проверяются соотношения $\overrightarrow{C_i C_{i+1}} \uparrow\uparrow \overrightarrow{C_i B_j} \uparrow\uparrow \overrightarrow{C_i B_{j+1}}$. При их выполнении в список $\mathcal{B}_i$ необходимо добавить обе вершины B_j и B_{j+1}.
5. Все списки $\mathcal{B}_i$, оставшиеся пустыми, можно удалить из $\mathcal{B}$.
6. Определяются отображения, ставящие в соответствие номеру вершины номера его соседей:

$$f_{\text{pre}}, f_{\text{next}} : \{B_1, B_2, ..., B_{l''}\} \to \{B_1, B_2, ..., B_{l''}\},$$

$$f_{\text{pre}}(B_j) := \begin{cases} B_{l''}, j = 1, \\ B_{j-1}, 2 \le j \le l'', \end{cases} \qquad f_{\text{next}}(B_j) := \begin{cases} B_{j+1}, 1 \le j \le l'' - 1, \\ B_1, j = l''. \end{cases}$$

7. Для каждого непустого списка $\mathcal{B}_i$ выполняется следующая последовательность шагов:
   7а. Список сортируется, например, в порядке возрастания абсцисс или ординат содержащихся в нём точек. Обозначим его элементы $\mathcal{B}_i = \{B'_{i1}, B'_{i2}, ..., B'_{il_i}\}$, $l_i > 0$.

7б. Для каждой вершины B'_{ij}, $j=\overline{1,l_i}$, необходимо определить, какое из двух отношений имеет место: $f_{\text{pre}}(B'_{ij})\in\mathcal{B}_i$ или $f_{\text{next}}(B'_{ij})\in\mathcal{B}_i$. Если $f_{\text{pre}}(B'_{ij})\in\mathcal{B}_i$, то нужно переопределить отображение f_{pre}: $f_{\text{pre}}(B'_{ij}):=B'_{i,j\pm1}$. Аналогично и в другом случае: $f_{\text{next}}(B'_{ij})\in\mathcal{B}_i\Rightarrow$ $\Rightarrow f_{\text{next}}(B'_{ij}):=B'_{i,j\pm1}$. При нечётном j в индексе $j\pm1$ нужно выбирать знак «плюс», при чётном – «минус».

8. По новым соседним вершинам $f_{\text{pre}}(B_j)$ и $f_{\text{next}}(B_j)$ восстанавливаются списки самонепересекающихся полигонов, составляющих исходный полигон $B_1B_2...B_{l''}B_1$. Следует обратить внимание, что после шагов 7б точка $f_{\text{pre}}(B_j)$ может оказаться на самом деле следующей, а $f_{\text{next}}(B_j)$ – предыдущей по отношению к B_j. Это означает, что необходимо составить список смежности вершин, по которому затем восстанавливаются их последовательности.
9. Последовательности из двух или менее точек удаляются. Список из остальных последовательностей является результатом работы этого алгоритма.

Данный алгоритм позволяет избавиться от фиктивных рёбер усечённого полигона, расположенных на рёбрах отсекателя. Однако он не работает, если исходный полигон является самопересекающимся (Рисунок 1.22), или при касании двух внутренних смежных рёбер границы отсекателя (Рисунок 1.23).

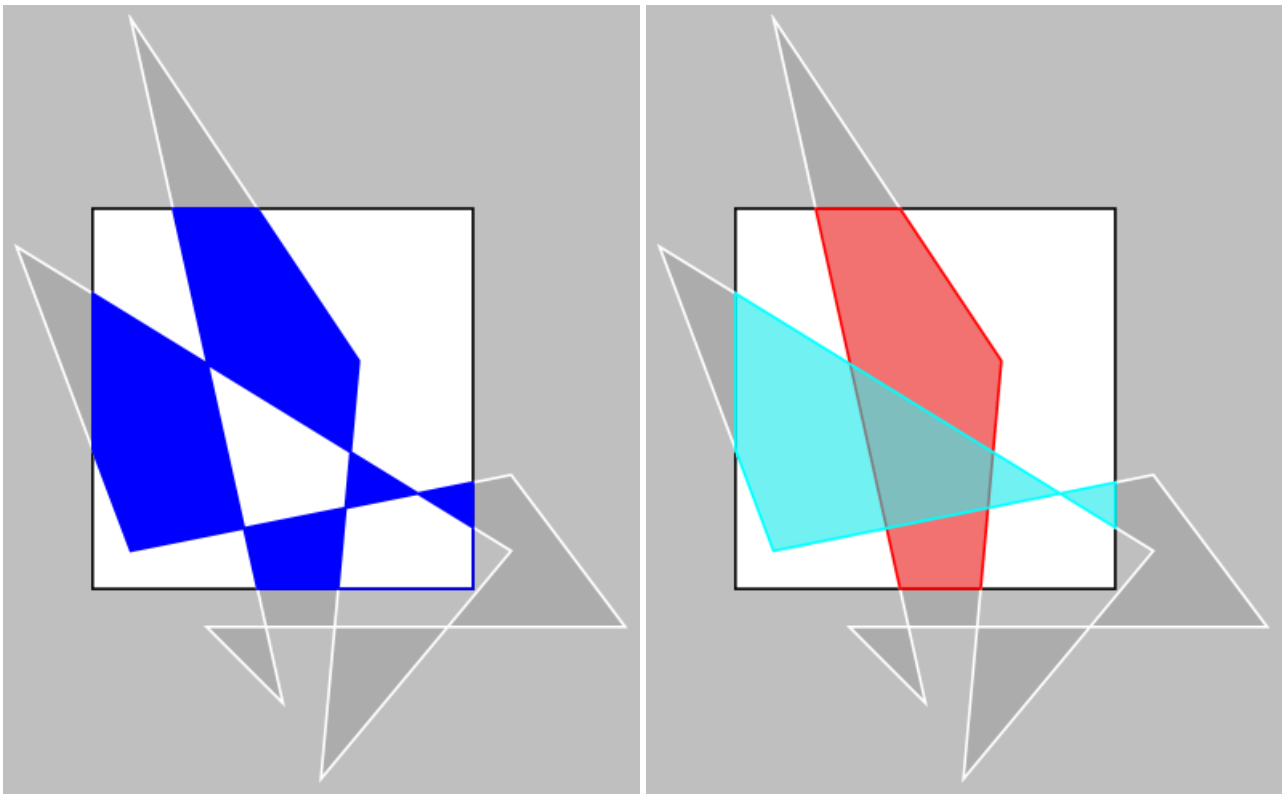


Рисунок 1.22 – Попытка убрать самопересекающиеся рёбра при отсечении самопересекающегося полигона

Рисунок 1.23 – Результат отсечения полигона
с вершиной, касающейся границы отсекателя

Замечание 5. Время работы алгоритма можно оценить той же асимптотикой, что и время работы алгоритма Сазерленда-Ходжмана. Ясно, что шаг 4 выполнется за время $O(l''n)$, что исходя из замечания 3 даёт $O((l+n)n)$. Время выполнения остальных шагов (кроме 7б) также можно оценить этой асимптотикой. Что касается шага 7б, то его можно выполнить за $O(l''+n)=O(l+n)$ (с учётом возможного дублирования точек в вершинах отсекателя), если объединить шаги 4 и 6: при добавлении вершин B_j и B_{j+1} в список $\mathcal{B}_i$ помечать номером i вершину B_{j+1} в списке смежности для B_j и наоборот, после чего на шаге 7б достаточно в нужном списке смежности заменить помеченную вершину на новую.

1.1.6. Алгоритм Вейлера-Азертона. Алгоритм Бентли-Оттманна нахождения пересечения отрезков

При моделировании и отображении компьютерной графики очень часто возникает ситуация, когда необходимо проводить отсечение невыпуклым окном (например, один объект произвольной формы загораживает другой). Алгоритм Вейлера-Азертона (Уайлера-Атертона) отсечения невыпуклого полигона отсекателем, также имеющим форму невыпуклого полигона, был предложен в 1977 году К. Уайлером и П. Атертоном [13]. Этот алгоритм находит пересечение отсекателя $C_1C_2...C_nC_1$ с отсекаемым полигоном $A_1A_2...A_lA_1$, которое в общем случае является совокупностью непересекающихся между собой полигонов (Рисунок 1.24) и заключается в следующем:

1. Для двух полигонов составляется по одному списку вершин в порядке правостороннего обхода. Далее подразумевается, что обходы $A_1A_2...A_lA_1$ и $C_1C_2...C_nC_1$ – правосторонние.
2. Вершины отсекаемого полигона $A_1A_2...A_lA_1$ помечаются в зависимости от того, находятся ли они внутри или снаружи полигона $C_1C_2...C_nC_1$.

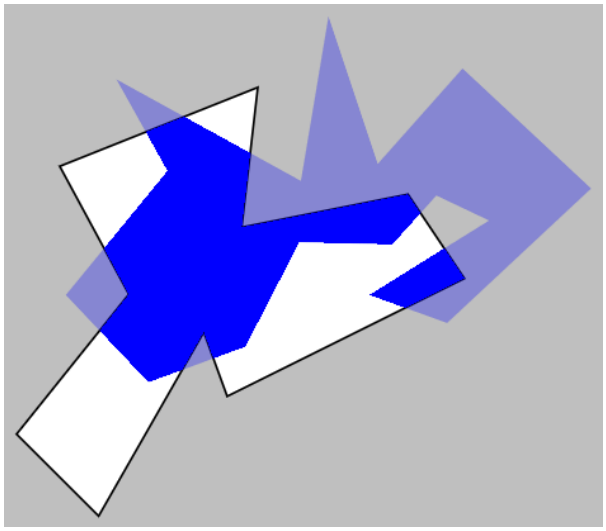
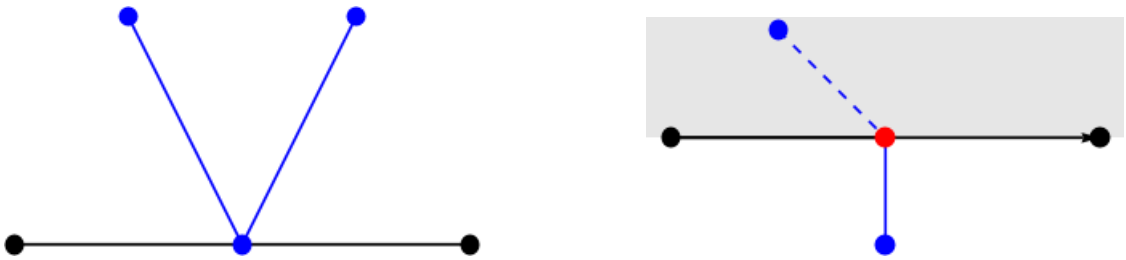


Рисунок 1.24 – Результатом отсечения полигона $A_1A_2...A_lA_1$ внешним отсекателем $C_1C_2...C_nC_1$ является пересечение этих полигонов

3. Находятся точки пересечений контуров обоих полигонов, которые затем помещаются в оба списка. Между списками устанавливаются двусторонние связи, отвечающие точкам пересечения.
4. Составляется список точек пересечения, в которых при обходе контура полигона $A_1A_2...A_lA_1$ происходит вход во внутреннюю область отсекателя $C_1C_2...C_nC_1$.
5. Из каждой точки, найденной на предыдущем шаге, начинается обход по контуру $A_1A_2...A_lA_1$ до следующей точки пересечения. От следующей точки пересечения осуществляется переход к контуру отсекателя $C_1C_2...C_nC_1$. Обход по нему происходит до достижения следующей точки пересечения, от которой переходим опять к полигону $A_1A_2...A_lA_1$ и т.д. до возврата к изначальной точке пересечения.
6. Алгоритм возвращает все полигоны, полученные на шаге 5.

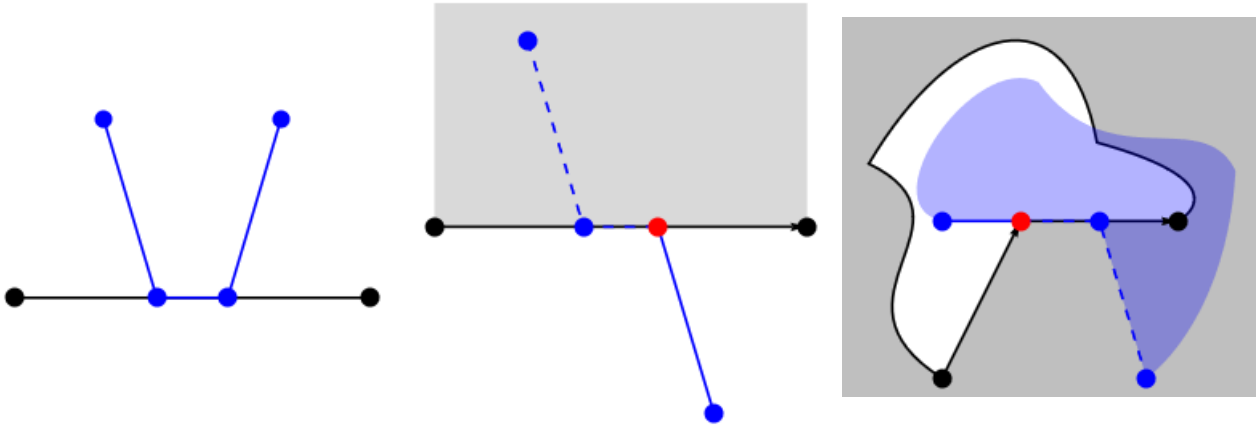
Замечание 1. Вырожденные пересечения обрабатываются следующим образом (на рисунках 1.25 и 1.26 синей ломаной обозначен фрагмент контура отсекаемого полигона, чёрной – граница отсекателя, красная точка – точка пересечения, учитываемая в алгоритме Вейлера-Азертона):

- Касание двух смежных рёбер полигона $A_1A_2...A_lA_1$ некоторого ребра отсекателя $C_1C_2...C_nC_1$ считается пересечением тогда и только тогда, когда эти смежные рёбра находятся по разные стороны от этого ребра отсекателя (Рисунок 1.25).
- Если ребро A_iA_{i+1} полностью лежит на ребре C_kC_{k+1}, то в случае расположения вершин A_{i-1} и A_{i+2} по одну сторону от ребра C_kC_{k+1} считается, что пересечений нет (Рисунок 1.26а). Если же они лежат по разные стороны, то пересечением считается та из вершин A_i либо A_{i+1}, которая является соседней для вершины, находящейся по правую



а б

Рисунок 1.25 – Случаи касания двух смежных рёбер отсекаемого полигона ребра отсекателя



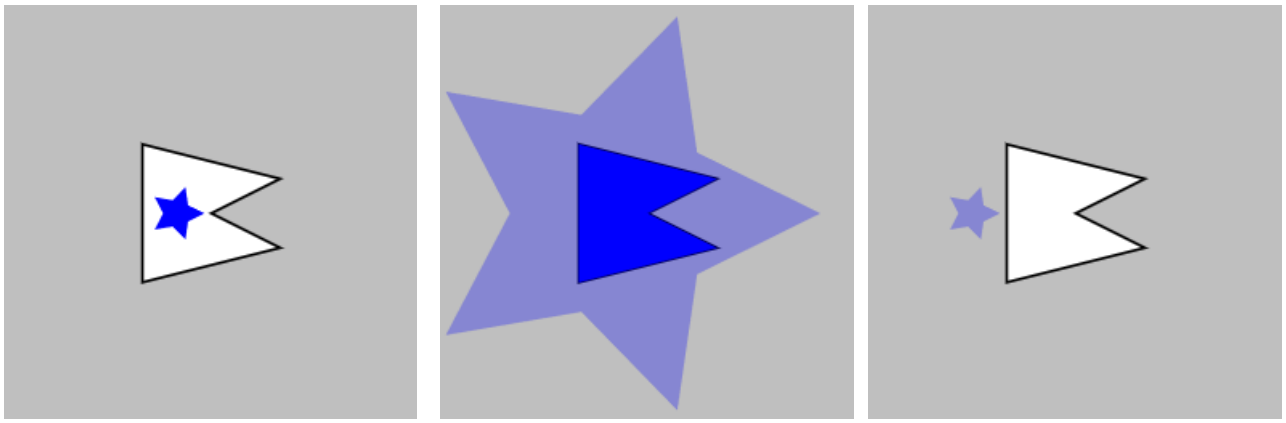
а б в

Рисунок 1.26 – Случаи, когда ребро отсекаемого полигона целиком или частично лежит на ребре отсекателя

сторону от оси C_kC_{k+1} (Рисунок 1.26б). Аналогично рассматривается случай частичного наложения двух рёбер A_iA_{i+1} и C_kC_{k+1} друг на друга (Рисунок 1.26в) и случай, когда ребро C_kC_{k+1} полностью лежит на ребре A_iA_{i+1}.

Замечание 2. Если пересечения отсутствуют, то результатом отсечения является весь полигон $A_1A_2...A_iA_1$, вся внутренняя область отсекателя $C_1C_2...C_nC_1$ либо пустое множество (Рисунок 1.27). В первом случае все вершины полигона $A_1A_2...A_iA_1$ находятся внутри окна $C_1C_2...C_nC_1$, в двух других – снаружи. Следовательно, если точек пересечения не оказалось, и вершины отсекаемого полигона лежат снаружи окна $C_1C_2...C_nC_1$, то нужно определить, лежат ли вершины окна внутри либо снаружи полигона $A_1A_2...A_iA_1$.

Одной из главных проблем при отсечении по алгоритму Вейлера-Азертона является нахождение всех пересечений границ двух полигонов. Очевидно, можно перебрать все ln пар рёбер и выяснить, какие из них пересекаются, затем найти координаты всех точек пересечения, после чего поместить их в оба списка согласно пункту 3 изложенного выше алгоритма. На выполнение такой последовательности операций уйдёт $O(ln)$ времени. Однако в общем случае это время может быть улучшено.



а б в

Рисунок 1.27 – Случаи, когда пересечения отсутствуют (а – полигон не подвергается отсечению, б – результатом отсечения является окно, в – полное отсечение)

Алгоритм, предложенный Дж. Л. Бэнтли и Т. А. Оттманном [14], позволяет найти все пересечения n отрезков при помощи **заметающей прямой**, параллельной оси ординат и движущейся слева направо (Рисунок 1.28). Будем полагать, что выполняются следующие три условия:

- среди рассматриваемых отрезков нет вертикальных,
- никакие три отрезка не пересекаются в одной точке,
- каждая пара отрезков имеет не более одной общей точки.

Во время выполнения данного алгоритма рассматривается последовательность отрезков, пересекаемых заметающей прямой, упорядоченная, например, по возрастанию ординаты точки пересечения. Очевидно, заметающая прямая $x = x_0$ пересекает некоторый отрезок AB с концами в точках $A(x_A, y_A)$ и $B(x_B, y_B)$ тогда и только тогда, когда $x_0 \in [x_A, x_B]$.

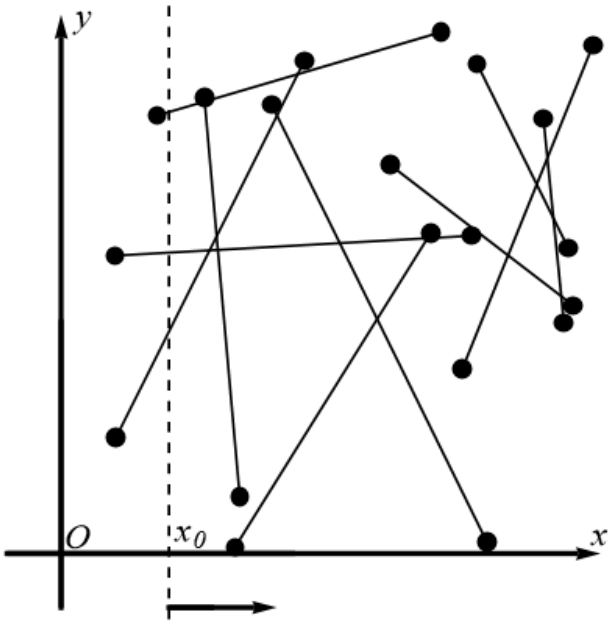
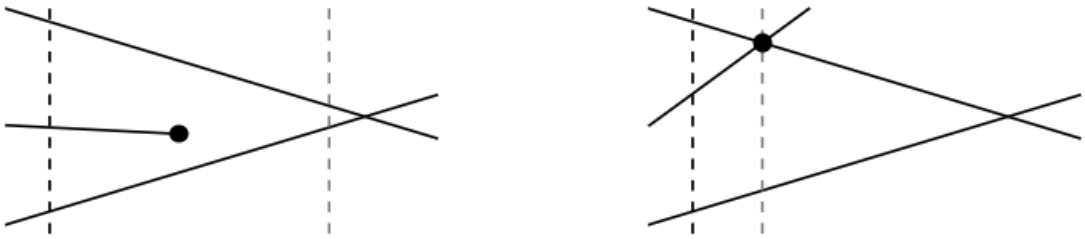


Рисунок 1.28 – Заметающая прямая (вертикальная пунктирная линия) сканирует все отрезки, двигаясь слева направо

По мере движения заметающей прямой слева направо эта последовательность меняется следующим образом:

- Когда заметающая прямая достигает левого конца нового отрезка, новый отрезок добавляется в текущую последовательность пересекаемых отрезков в нужное место согласно выбранному способу сортировки.
- Когда заметающая прямая достигает конца отрезка, этот отрезок подлежит удалению из последовательности.
- Когда заметающая прямая проходит через пересечение двух отрезков, их нужно поменять местами в последовательности. При этом два отрезка из последовательности могут пересекаться на данном этапе только тогда, когда между ними нет других отрезков, так как в случае наличия такового он либо кончится раньше пересечения (Рисунок 1.29а), либо он бы пересёк один из двух отрезков раньше (Рисунок 1.29б).



а б

Рисунок 1.29 – Если между двумя отрезками есть третий, пересекающий заметающую прямую, то либо найдётся дальнейшее её положение, где два отрезка станут соседними, либо один из них пересечётся с третьим раньше

Отсюда следует последовательность шагов **алгоритма Бентли-Оттманна** нахождения всех пересечений отрезков A_iB_i, $i=\overline{1,m}$, с концами в точках $A_i(x_{i1}, y_{i1})$ и $B_i(x_{i2}, y_{i2})$, $x_{i1} < x_{i2}$:

1. Точки A_i и B_i записываются в структуру данных P в порядке возрастания абсцисс. В структуру P также будут записываться все найденные точки пересечений, причём их тоже надо размещать в соответствии с заданным порядком. Также объявляется структура данных S, где хранятся отрезки, пересекаемые заметающей прямой, в порядке возрастания ординат их точек пересечения.
2. Заметающая прямая движется от самой первой точки структуры P к самой последней. Текущая точка удаляется из структуры P. Если она является:
   2а. точкой начала A_i нового отрезка, то этот отрезок добавляется в структуру S в соответствии с заданным порядком. После добавления нового отрезка необходимо проверить, пересекается ли он с отрезками, оказавшимися его соседями в структуре S. В

случае наличия таких пересечений их нужно поместить в структуру P.

2б. точкой пересечения двух отрезков, то эти два отрезка меняются местами в структуре S. После этого снова нужно проверить наличие пересечений у двух пар отрезков, ставших новыми соседями. Если найдены новые пересечения, они добавляются в P.

2в. точкой конца B_i некоторого отрезка структуры S, то этот отрезок удаляется из S, после чего его два соседа становятся соседними между собой, и нужно проверить, пересекаются ли они. Если найдено новое пересечение, оно добавляется в P.

3. Шаг 2 повторяется, пока структура P содержит хотя бы одну точку.
4. Алгоритм возвращает все пересечения, найденные на шаге 2, с указанием отрезков, которым принадлежат эти пересечения.

Особенности совместного использования алгоритмов Вейлера-Азертона и Бентли-Оттманна. С учётом замечания 1 ясно, что каждое из найденных пересечений разделяет две смежные вершины полигона $A_1A_2...A_lA_1$ на внешнюю и внутреннюю. Следовательно, на шаге 2 алгоритма Вейлера-Азертона достаточно лишь для некоторой одной вершины полигона $A_1A_2...A_lA_1$ определить её положение относительно окна $C_1C_2...C_nC_1$. Если она окажется лежащей на ребре отсекателя, то согласно рисунку 1.25 определяется, не является ли она точкой пересечения.

Из постановки задачи отсечения полигона $A_1A_2...A_lA_1$ окном $C_1C_2...C_nC_1$ ясно, что для нахождения нужных пересечений по алгоритму Бентли-Оттманна нужно проверять пересекаемость только тех пар отрезков, которые являются рёбрами разных полигонов. При реализации алгоритма Бентли-Оттманна можно считать, что никакие три отрезка не пересекутся в одной точке (за исключением случая пересечения в вершине одного из полигонов), так как в противном случае получим, что по крайней мере два пересекающихся отрезка являются рёбрами одного полигона, а следовательно, он является самопересекающимся. По той же причине можно также считать, что никакие три отрезка не накладываются друг на друга.

Сложность двух алгоритмов. Покажем, что шаг 2 алгоритма Бентли-Оттманна выполняется за время $O(\max(\log m, \log p))$, где p – число текущих точек в структуре P. Действительно, все операции, за исключением добавления отрезка в структуру S и новых точек пересечения в структуру P, выполняются за ограниченное время, добавление отрезка в S и новой точки в P – за время $\Omega(\log m)$ и $\Omega(\log p)$ соответственно. Эти оценки являются достижимыми, если в качестве структуры для P взять **приоритетную очередь**, а для S – **сбалансированное дерево**. В структуру P в худшем случае будут добавлены две новые точки за время $\Theta(\log(p-1)) + \Theta(\log p) = O(\log p)$.

Оценим теперь количество итераций алгоритма и число p. Легко видеть, что всего будет выполнено $2m + k$ итераций, где k – общее число искомых пересечений. Так как структура P содержит только точки концов исходных отрезков и пересечения отрезков, являющихся соседями в структуре S, то P

всегда содержит не более $3m - 1$ точки. Таким образом, доказано, что $p = O(m)$, а значит, общая сложность алгоритма может быть оценена асимптотикой $O((2m+k)\log m) = O((m+k)\log m)$.

Временная сложность алгоритма Вейлера-Азертона совпадает с алгоритмом Бентли-Оттманна, так как все остальные шаги выполняются за линейное время $O(l + n)$.

Замечание. Для числа пересечений m отрезков справедлива оценка $k = O(m^2)$, которая достигается в случае, когда все отрезки попарно пересекаются. В таких ситуациях алгоритм Бентли-Оттманна выполняется медленнее, чем обычный перебор всех пар отрезков. Однако на практике в большинстве случаев число k пропорционально m, и тогда алгоритм Бентли-Оттманна предпочтительнее попарного перебора.

1.1.7. Удаление невидимых рёбер выпуклого полигона на плоскости

Наряду с рассмотренными выше алгоритмами отсечения, где подразумевается, что наблюдатель находится за пределами картинной плоскости, также часто используются алгоритмы отсечения при наблюдении с рассматриваемой плоскости в конкретном направлении или в разные стороны. Как и в пространстве, различают случаи дальнего (Рисунок 1.30а) и ближнего наблюдения (Рисунок 1.30б). Дальнее наблюдение характеризуется вектором $\vec{s}$, указывающим направление на наблюдателя, ближнее наблюдение – точкой S, где расположен наблюдатель.

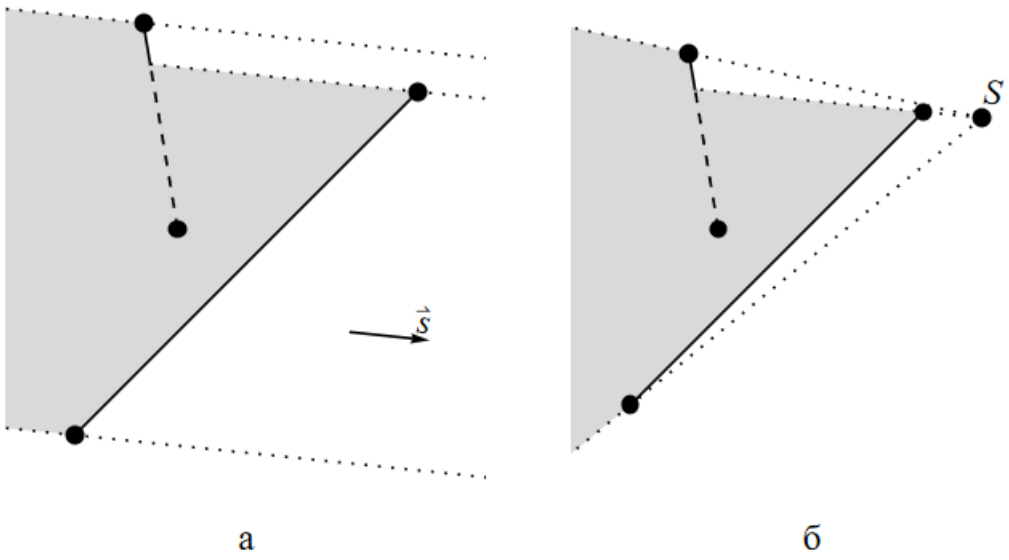


Рисунок 1.30 – Отсечение отрезка, заслоняемого другим отрезком, при ближнем и дальнем наблюдении на плоскости

Рассмотрим алгоритмы удаления невидимых сторон для выпуклых полигонов. Так как всякая прямая и луч, проведённые через выпуклый полигон, пересекает его границу не более, чем в двух точках, то граница выпуклого полигона делится на две ломаные, одна из которых является полностью видимой, другая заслоняется ей и подлежит отсечению (Рисунок 1.31).

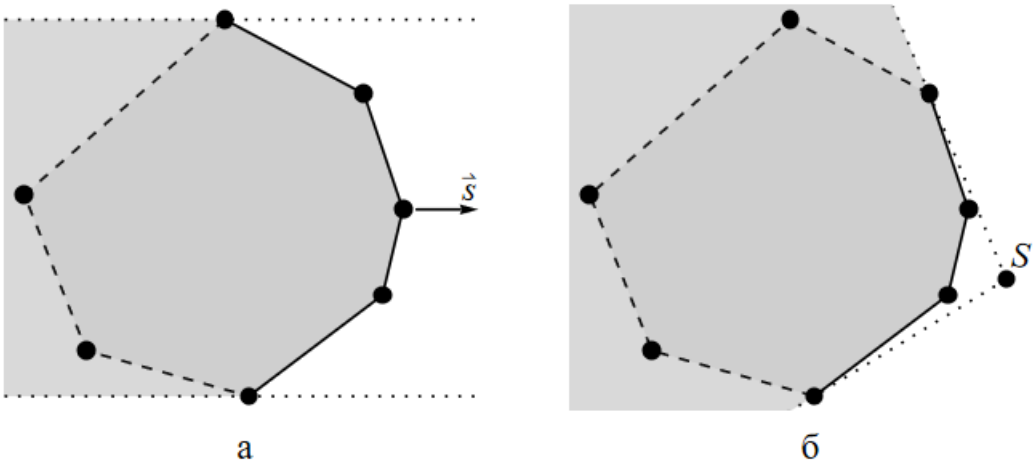


Рисунок 1.31 – Отсечение невидимых рёбер выпуклого полигона при дальнем (а) и ближнем (б) наблюдении

Алгоритм удаления невидимых рёбер выпуклого полигона $A_1A_2...A_nA_1$ с левосторонним обходом **при дальнем наблюдении**, характеризующемся направлением $\vec{s}(s_x, s_y)$ на наблюдателя, состоит из следующих шагов:

1. Происходит преобразование координат таким образом, чтобы направление на наблюдателя стало сонаправленным оси абсцисс. Такое преобразование обеспечивает, например, умножение координатных столбцов точек A_i, $i=\overline{1,n}$, слева на матрицу $\begin{pmatrix} s_x & s_y \\ -s_y & s_x \end{pmatrix}$.

   Эта матрица отвечает за поворот на угол, противоположный углу между осью абсцисс и вектором $\vec{s}$ с последующим растяжением плоскости вокруг начала координат в $\sqrt{s_x^2 + s_y^2}$ раз.

2. Среди точек A_i, $i=\overline{1,n}$, находятся точки с минимальным и максимальным значением ординат в новой системе координат. Если таких точек оказалось несколько, то среди точек как с наименьшей, так и с наибольшей ординатой выбираются две точки с наибольшими абсциссами. Обозначим их $A_{i_{\min}}$ и $A_{i_{\max}}$.
3. Часть границы $A_{i_{\min}}A_{i_{\min}+1}...A_{i_{\max}-1}A_{i_{\max}}$ является <u>видимой</u>, а $A_{i_{\max}}A_{i_{\max}+1}...A_{i_{\min}-1}A_{i_{\min}}$ <u>не видна</u>. Таким образом, все рёбра <u>последней</u> ломаной подлежат отсечению.

При правостороннем обходе исходного полигона подчёркнутые слова следует изменить на противоположные по смыслу. Нетрудно видеть, что

данный алгоритм работает за время $O(n)$, так как требует нахождения самой нижней и самой высокой точки во множестве из n точек.

В случае же **ближнего наблюдения**, которое характеризуется точкой наблюдения $S(s_x, s_y)$, алгоритм отсечения невидимых рёбер выпуклого полигона $A_1A_2...A_nA_1$ с левосторонним обходом выглядит следующим образом:

1. Выполняется проверка на принадлежность точки S заданному полигону. Если она оказалось внутренней, то все точки являются видимыми – отсечения не происходит. Если точка S оказалась лежащей на границе полигона, то в зависимости от постановки задачи видимыми полагаются либо все вершины полигона (наблюдение изнутри), либо сторона полигона, на которой лежит точка S (наблюдение снаружи).
2. Если точка S оказалась внешней для полигона, то из неё проводятся векторы $\overrightarrow{SA_i}$, $i = \overline{1,n}$.
3. Среди полученных векторов нужно найти два крайних. Сравнение двух произвольных векторов $\vec{a}$ и $\vec{b}$ происходит по следующему правилу: при знаке угла $\operatorname{sgn}\angle\left(\vec{a},\vec{b}\right) > 0$ вектор $\vec{b}$ находится левее вектора $\vec{a}$ (поворот против часовой стрелки), если же $\operatorname{sgn}\angle\left(\vec{a},\vec{b}\right) < 0$, то $\vec{b}$ расположен правее $\vec{a}$ (по направлению часовой стрелки). Если среди векторов $\overrightarrow{SA_i}$ найдётся несколько крайних векторов, то из них выбираются векторы наименьшей длины. Концы крайних в направлении против и по часовой стрелке векторов обозначим $A_{i_{\min}}$ и $A_{i_{\max}}$ соответственно.
4. Часть границы $A_{i_{\min}}A_{i_{\min}+1}...A_{i_{\max}-1}A_{i_{\max}}$ является <u>видимой</u>, а $A_{i_{\max}}A_{i_{\max}+1}...A_{i_{\min}-1}A_{i_{\min}}$ <u>не видна</u>. Таким образом, все рёбра <u>последней</u> ломаной подлежат отсечению.

При правостороннем обходе исходного полигона подчёркнутые слова следует изменить на противоположные по смыслу. Как и у предыдущего алгоритма, трудоёмкость составляет $O(n)$.

<u>Замечание.</u> Данный алгоритм можно объединить с проверкой принадлежности наблюдателя полигону при помощи октантного критерия [1, с. 66–70 и разд. 2.3.3], в котором также вычисляются векторы $\overrightarrow{SA_i}$.

1.2. Алгоритмы отсечения в трёхмерном пространстве

Наряду с отсечениями и удалениями невидимых объектов на картинной плоскости часто стоит задача отсечения в трёхмерном пространстве. В отличие от отсечений на картинной плоскости, которое осуществляется **после**

проективных преобразований, отсечения в трёхмерной сцене **предшествуют** им. Оно осуществляется в первую очередь в ситуациях наложения и перекрывания множества объектов на рассматриваемой сцене, что приводит к невозможности или затруднению отсечения объектов, следующего после проективных преобразований.

Несмотря на кажущуюся простоту задач отсечения (избавление от объектов, выходящих за пределы видимости наблюдателя – так называемую **пирамиду видимости**, см. рисунок 1.1) и удаления (в случае, когда объект перекрывается, или **экранируется** другим) трёхмерных объектов и их элементов, эти задачи характеризуются большим объёмом и сложностью вычислений. Поэтому существует множество различных алгоритмов отсечения, применяемых к трёхмерным объектам и различающихся алгоритмически, а также в плане аппаратной и программной реализации.

1.2.1. Выпуклые области и полиэдры в трёхмерном пространстве. Отсечение отрезков трёхмерными выпуклыми отсекателями

Пожалуй, одни из наиболее простых методов и алгоритмов отсечения трёхмерной графики были предложены Л. Г. Робертсом, американским инженером в области компьютерного зрения и компьютерных сетей, в диссертации [15, с. 62–69], которую он защитил в 1963 году. В ней он рассматривал, в частности, вопрос визуализации выпуклых **полиэдров**, для чего необходимо осуществлять отсечение и удаление некоторых рёбер или их частей.

Полиэдры характеризуются своими вершинами, рёбрами и гранями, которые ограничивают некоторую область в пространстве. **Выпуклый полиэдр** – это полиэдр, который целиком лежит по одну сторону от плоскости, проведённой через любую из его граней (Рисунок 1.32). Для отрисовки выпуклых полиэдров, которые, вообще говоря, могут выходить за пределы видимости наблюдателя или перекрывать друг друга, Робертс решал задачу отсечения и удаления их рёбер, для чего выделил следующие три подзадачи:

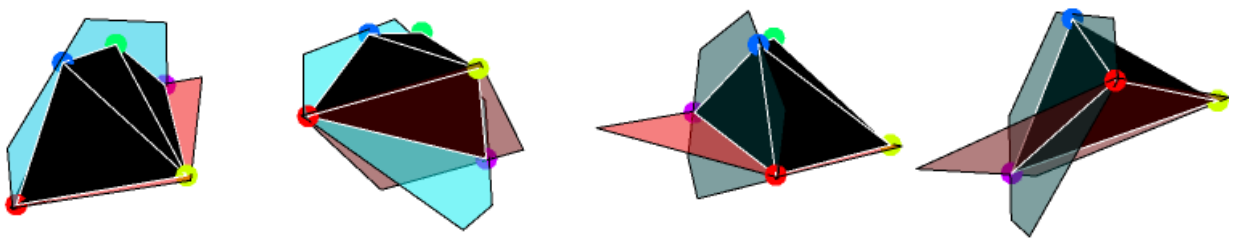


Рисунок 1.32 – Выпуклый полиэдр и две плоскости, проведённые через его грани, изображённые с разных ракурсов

1. отсечение рёбер, оказавшихся за пределами видимости наблюдателя (Рисунок 1.33б),
2. удаление рёбер, невидимых из-за граней того же полиэдра (Рисунок 1.33в),
3. удаление рёбер или их частей, невидимых из-за граней других полиэдров (Рисунок 1.33г).

На рисунке 1.33 изображено два полиэдра, у которых отсекаемые и удаляемые на каждом из перечисленных этапов рёбра изображены пунктиром.

Для адекватного графического представления полиэдров важно знать, по какую из сторон от каждой грани расположена внутренняя область полиэдра. В случае выпуклого полиэдра это легко выяснить исходя из следующих соображений. Обозначим через A_1, A_2, …, A_n последовательные вершины некоторой грани, N – вершина полиэдра, не лежащая на этой грани. Тогда вектор нормали, проведённый от полигона $A_1A_2...A_nA_1$ и направленный внутрь полиэдра (**внутренняя нормаль**), должен образовывать острый угол с любым вектором, проведённым от плоскости полигона $A_1A_2...A_nA_1$ к любой внутренней или граничной точке полиэдра, в том числе и к вершине N (Рисунок 1.34). Таким образом, для внутренней нормали $\vec{n}$ получаем следующие условия:

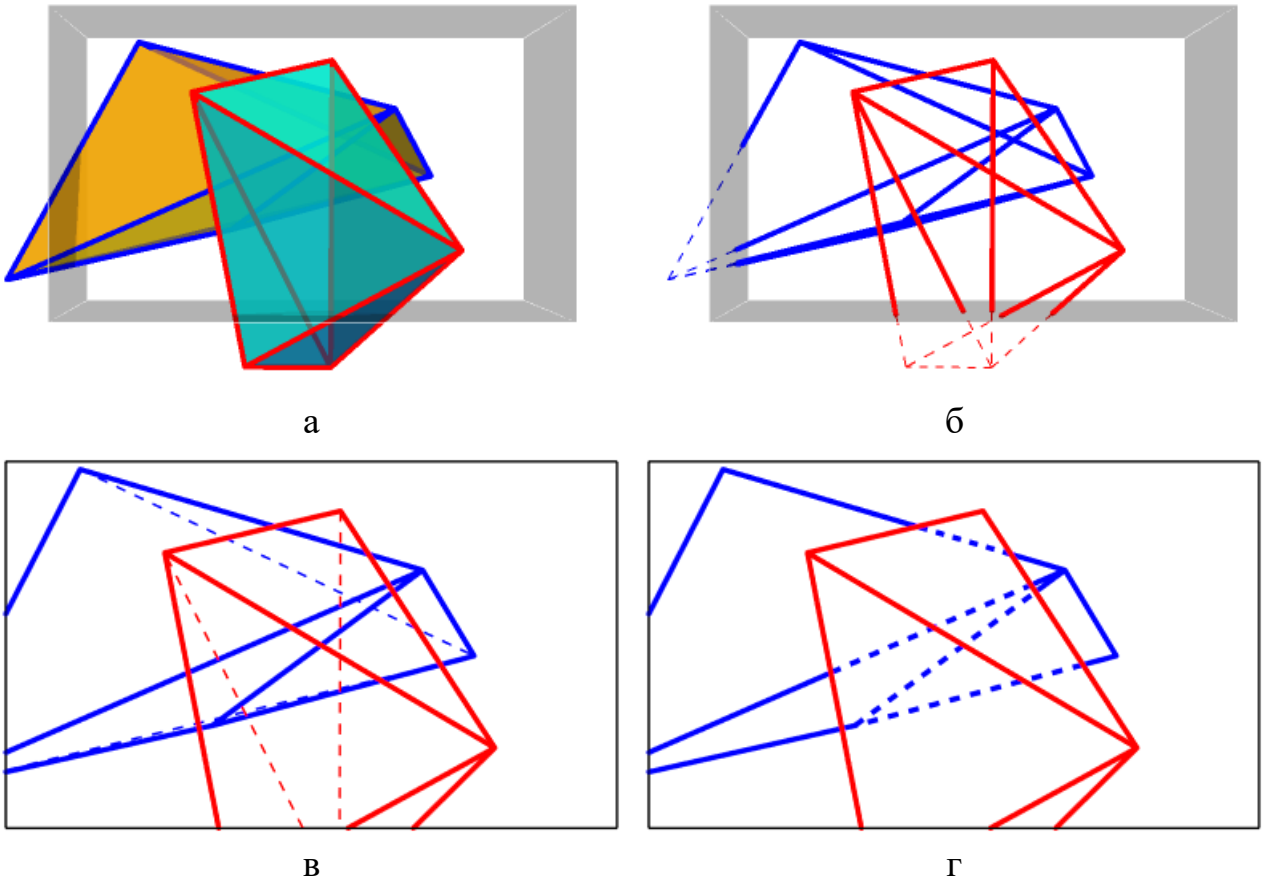


Рисунок 1.33 – Отсечение и удаление рёбер полиэдров: а – два полиэдра внутри пирамиды видимости дальнего наблюдателя, б – отсечение невидимых рёбер, в – удаление невидимых (нелицевых) рёбер, г – удаление рёбер, экранируемых другими объектами

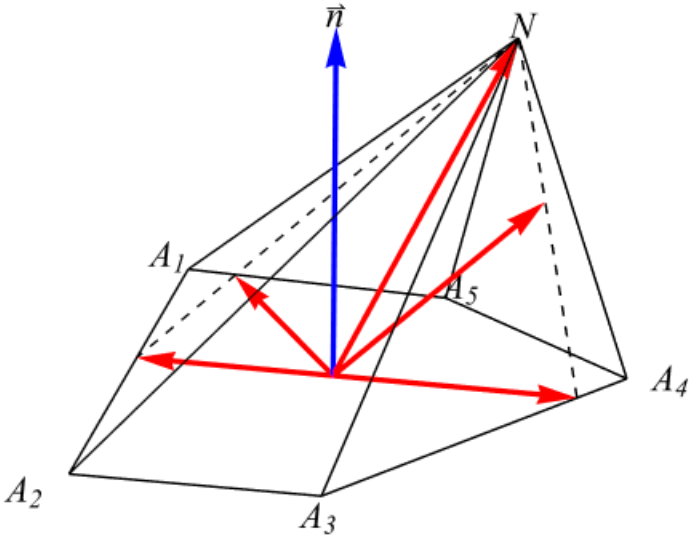


Рисунок 1.34 – Внутренняя нормаль $\vec{n}$ грани $A_1A_2A_3A_4A_5A_1$ образует острые и прямые углы со всеми красными векторами, проведёнными от центра грани к граничным точкам пирамиды $NA_1A_2A_3A_4A_5$

$$\vec{n} \perp \overrightarrow{A_1A_2},$$
$$\vec{n} \perp \overrightarrow{A_2A_3},$$
$$\vec{n} \cdot \overrightarrow{A_1N} > 0.$$

Отсюда имеем, во-первых, что $\vec{n} \parallel \overrightarrow{A_1A_2} \times \overrightarrow{A_2A_3}$, во-вторых, что из двух направлений $\overrightarrow{A_1A_2} \times \overrightarrow{A_2A_3}$ и $-\overrightarrow{A_1A_2} \times \overrightarrow{A_2A_3}$ следует выбирать то, которое даёт положительное скалярное произведение с вектором $\overrightarrow{A_1N}$, что приводит к вычислению смешанного произведения: $\left(\overrightarrow{A_1A_2} \times \overrightarrow{A_2A_3}\right) \cdot \overrightarrow{A_1N} = \overrightarrow{A_1A_2}\,\overrightarrow{A_2A_3}\,\overrightarrow{A_1N}$. Таким образом, обозначая $\overrightarrow{A_1A_2} = \vec{a}$, $\overrightarrow{A_2A_3} = \vec{b}$, $\overrightarrow{A_1N} = \vec{c}$, окончательно получаем

$$\vec{n} = \operatorname{sgn}(\vec{a}\vec{b}\vec{c})(\vec{a} \times \vec{b}). \qquad (1.16)$$

Для получения **внешней нормали** знак в правой части формулы (1.16) следует поменять на противоположный. Отметим, что в качестве векторов $\vec{a}$ и $\vec{b}$ могут выступать любые два неколлинеарные вектора, параллельные рассматриваемой грани, а вектор $\vec{c}$ может быть проведён от любой точки грани к любой внутренней или граничной точке полиэдра, не лежащей на этой грани. В частности, в качестве точки N может быть использован геометрический центр (**барицентр**) полиэдра, который заведомо является его внутренней точкой и координаты которого равны среднему арифметическому координат всех вершин полиэдра.

Кроме выпуклых полиэдров, в компьютерной графике часто используются бесконечные выпуклые области, например, для задания пирамид видимости.

Такие области обычно задаются множеством ограничивающих их плоскостей с заданными векторами внутренних либо внешних нормалей. Каждая из этих плоскостей ограничивает некоторое подпространство

$$A_i x + B_i y + C_i z + D_i > 0, \;\; i = \overline{1,n}, \tag{1.17}$$

а вектор с координатами (A_i, B_i, C_i) задаёт внутреннюю нормаль плоскости. Так, следующие две системы задают разные области (Рисунок 1.35):

$$\begin{cases} y > 0, \\ x > 0, \\ z > 0, \\ -x + 1 > 0, \\ -z + 1 > 0, \end{cases} \qquad \begin{cases} -y > 0, \\ x > 0, \\ z > 0, \\ -x + 1 > 0, \\ -z + 1 > 0. \end{cases}$$

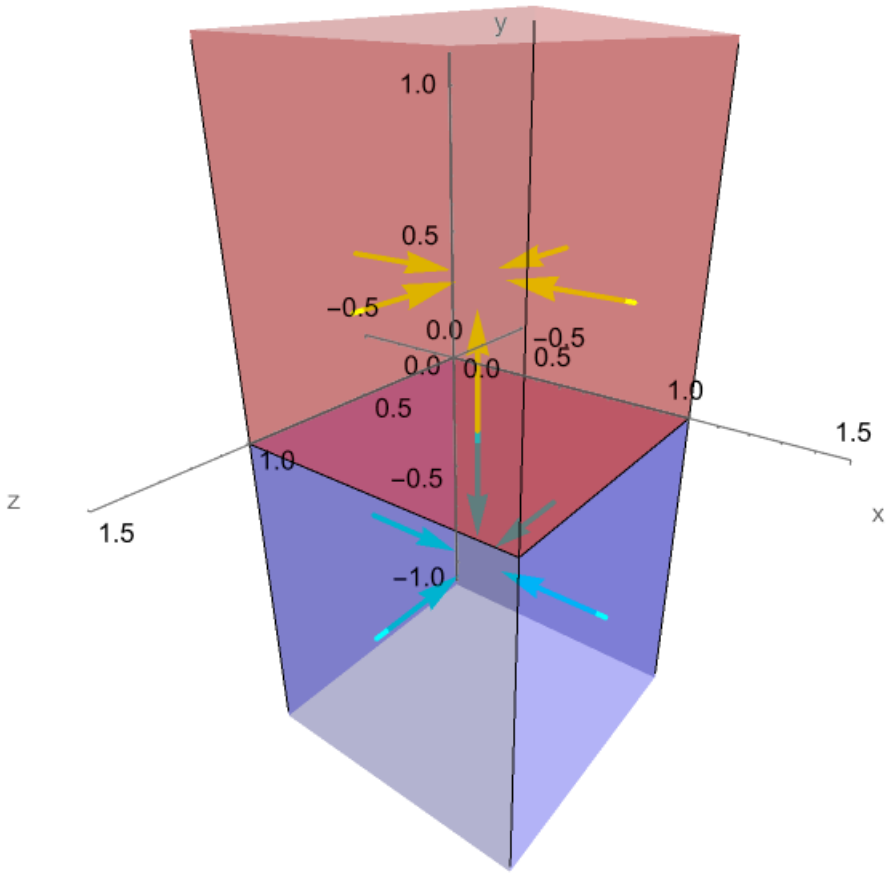


Рисунок 1.35 – Две бесконечных выпуклых области, задаваемых плоскостями и их внутренними нормалями

Отсечение отрезков трёхмерными выпуклыми отсекателями. Пусть некоторый трёхмерный отсекатель задан в виде системы неравенств (1.17).

Тогда точка с координатами (x_0, y_0, z_0) расположена внутри отсекателя тогда и только тогда, когда эти неравенства выполняются для координат этой точки. Это можно представить в следующем виде:

$$A_i x_0 + B_i y_0 + C_i z_0 + D_i = A_i x_0 + B_i y_0 + C_i z_0 + D_i \cdot 1 = (A_i, B_i, C_i, D_i) \cdot (x_0, y_0, z_0, 1)^T.$$

Тогда получается, что если выпуклый отсекатель представить в виде матрицы

$$\mathcal{C} = \begin{pmatrix} A_1 & B_1 & C_1 & D_1 \\ A_2 & B_2 & C_2 & D_2 \\ \dots & \dots & \dots & \dots \\ A_n & B_n & C_n & D_n \end{pmatrix}, \qquad (1.18)$$

в которой каждая строка соответствует некоторой одной ограничивающей плоскости (её иногда называют **матрицей тела**), и умножить её справа на координатный столбец некоторой точки M в однородных координатах $\tilde{M} = (x_0, y_0, z_0, 1)^T$, то получится столбец, исходя из которого можно судить о расположении этой точки относительно отсекателя: она находится **внутри** него тогда и только тогда, когда в столбце $\mathcal{C}\tilde{M}$ **все элементы положительны**. В противном случае отрицательные значения соответствуют граням отсекателя, за пределами которых оказалась рассматриваемая точка. Если какое-то из значений оказалось равно нулю, то тогда точка лежит на соответствующей грани. Эти утверждения остаются справедливыми при использовании произвольного столбца $(\omega x_0, \omega y_0, \omega z_0, \omega)^T$ при $\omega > 0$.

Как и в алгоритмах двухмерного отсечения, отрезок AB представляется параметрически по формуле (1.4), что эквивалентно следующей векторной формуле: $\vec{r}(t) = \overrightarrow{OA} + t\overrightarrow{AB}$, $t \in [0,1]$, где $\vec{r}(t)$ – радиус-вектор точки отрезка AB, которой соответствует заданный параметр t. Тогда координатный столбец в однородных координатах точки M такой, что $\overrightarrow{OM} = \vec{r}(t)$, равен $\tilde{A} + t\left(\tilde{B} - \tilde{A}\right)$, где $\tilde{A} = (x_A, y_A, z_A, 1)^T$, $\tilde{B} = (x_B, y_B, z_B, 1)^T$ – координатные столбцы в однородных координатах точек A и B, и тогда получаем следующее условие:

$$\mathcal{C}\tilde{M} = \mathcal{C}\left(\tilde{A} + t\left(\tilde{B} - \tilde{A}\right)\right) = \mathcal{C}\tilde{A} + t\mathcal{C}\left(\tilde{B} - \tilde{A}\right) \succ 0,$$

где символом $\succ$ обозначено, что все элементы полученного столбца должны быть больше нуля. Обозначая теперь

$$P = (p_1, p_2, ..., p_n)^T = C\tilde{A},$$
$$Q = (q_1, q_2, ..., q_n)^T = C\left(\tilde{B} - \tilde{A}\right), \tag{1.19}$$

где

$$p_i = A_i x_A + B_i y_A + C_i z_A + D_i,$$
$$q_i = A_i(x_B - x_A) + B_i(y_B - y_A) + C_i(z_B - z_A) \tag{1.20}$$

при $i = \overline{1,n},$ получим систему неравенств

$$p_i + tq_i > 0, \tag{1.21}$$

откуда

$$q_i > 0 \Rightarrow t > -p_i/q_i,$$
$$q_i < 0 \Rightarrow t < -p_i/q_i.$$

Если для некоторого отсекателя получим $q_i = 0,$ то тогда неравенство (1.21) сведётся к $p_i > 0,$ и если оно не выполняется, то отрезок лежит целиком за пределами отсекателя.

Таким образом, получили **алгоритм отсечения отрезка трёхмерным выпуклым внешним отсекателем**, заданным в виде системы линейных алгебраических неравенств вида (1.17):

1. Для первоначального отрезка *AB* задаются значения параметра *t* для начала и конца: $t_{\text{нач}} := 0,$ $t_{\text{кон}} := 1.$ Задаётся также итерационная переменная $i := 1.$
2. Для *i*-ой грани отсекателя определяется, какую часть отрезка она отсекает. Для этого по формулам (1.20) вычисляются значения p_i и $q_i.$
3. Далее может возникнуть одна из трёх ситуаций:
   - 3а. Если $q_i = 0,$ то проверяется выполнение неравенства $p_i < 0.$ При его выполнении алгоритм немедленно завершает свою работу, возвращая **пустой результат**. Иначе происходит переход к шагу 5.
   - 3б. Если $q_i > 0,$ то отрезок **отсекается с начала**: $t_{\text{нач}} := \max\{-p_i/q_i, t_{\text{нач}}\}.$
   - 3в. Если $q_i < 0,$ то отрезок **отсекается с конца**: $t_{\text{кон}} := \min\{-p_i/q_i, t_{\text{кон}}\}.$
4. Если выполняется неравенство $t_{\text{нач}} > t_{\text{кон}},$ то алгоритм немедленно возвращает пустой результат.

5. Если $i = n$, то алгоритм возвращает отрезок с концами в точках, которые получаются при подстановке значений $t_{нач}$ и $t_{кон}$ в систему (1.4). Это часть исходного отрезка, расположенная внутри отсекателя. Иначе алгоритм продолжается: $i := i + 1$, переход к шагу 2.

Замечание 1. Несложно видеть, что данный алгоритм выполняется за время $O(n)$.

Замечание 2. Вместо шага 2 и использования формул (1.20) можно воспользоваться матричными формулами (1.19). Хотя такой подход может привести к лишним вычислениям (когда отрезок отсекается не всеми гранями отсекателя), однако матричные вычисления часто используются в различных задачах и приложениях компьютерной графики, что оказалось возможным благодаря аппаратной и программной оптимизации, а также параллельным вычислениям.

1.2.2. Алгоритм Робертса

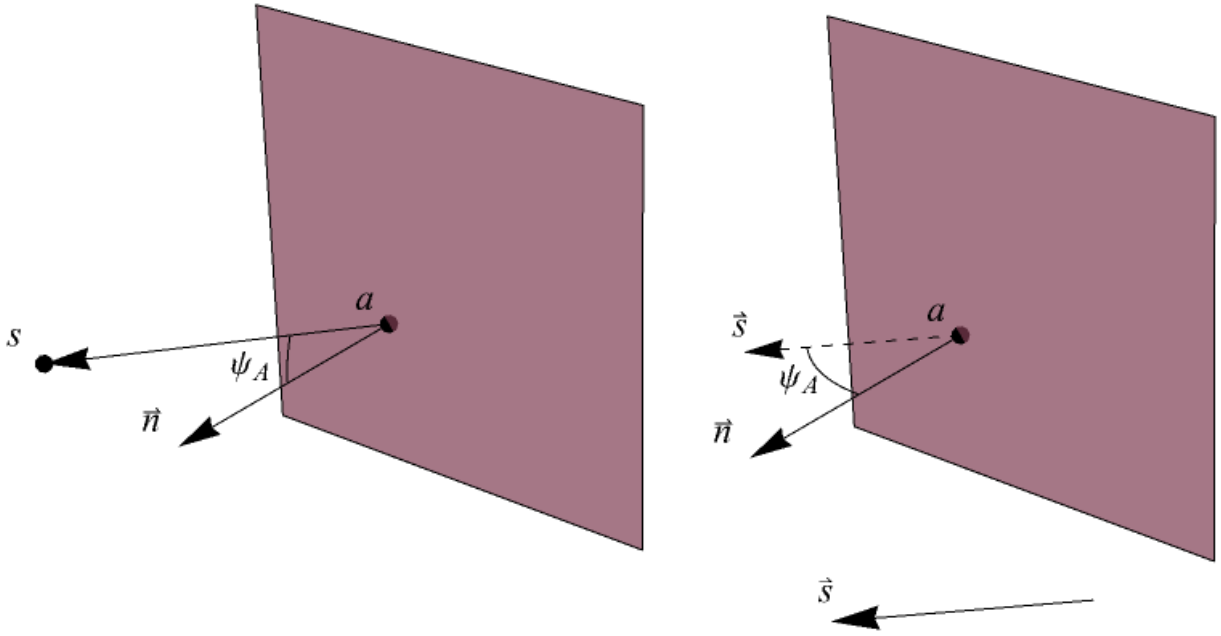
Алгоритм Робертса, идея которого мельком описана в [15, с. 65–66], определяет, какие из граней заданного выпуклого полиэдра являются видимыми, или **лицевыми**, что позволяет удалить рёбра невидимых граней при его рисовании. Как и при наблюдении полигона в плоскости (см. пункт 1.1.7), у выпуклого полиэдра в пространстве всякая грань либо целиком видима, либо целиком перегораживается другими гранями. Для точки a, лежащей на некоторой плоской поверхности, можно ввести величину, называемую **углом видимости** и определяемую как угол между вектором внешней нормали $\vec{n}$ этой поверхности и вектором, проведённым от точки к наблюдателю (Рисунок 1.36):

$$\psi_A = \angle(\vec{n}, \vec{s}),$$

где вектор направления на наблюдателя $\vec{s}$ является постоянным при дальнем наблюдении, а при ближнем полагается $\vec{s} = \overrightarrow{as}$, где s – позиция наблюдателя. Легко видеть, что угол видимости может принимать значения от 0 до π[1]. **Углом видимости** всей **поверхности** будем называть угол видимости некоторой её точки.

Грань выпуклого полиэдра является **лицевой** тогда и только тогда, когда угол её видимости является **острым**; **невидимой** – когда этот угол **тупой**; **вырожденной** – когда этот угол **прямой** (Рисунок 1.37). Значит, для

[1] Хотя согласно определению, данному в первой части ЭУМК [1, п. 1.4.1], величина угла между двумя векторами может принимать значения из промежутка $(-\pi, \pi]$, в трёхмерном пространстве, как правило, в качестве величины угла рассматривают только положительные значения, так как знак угла зависит от позиции и направления наблюдения за этими векторами



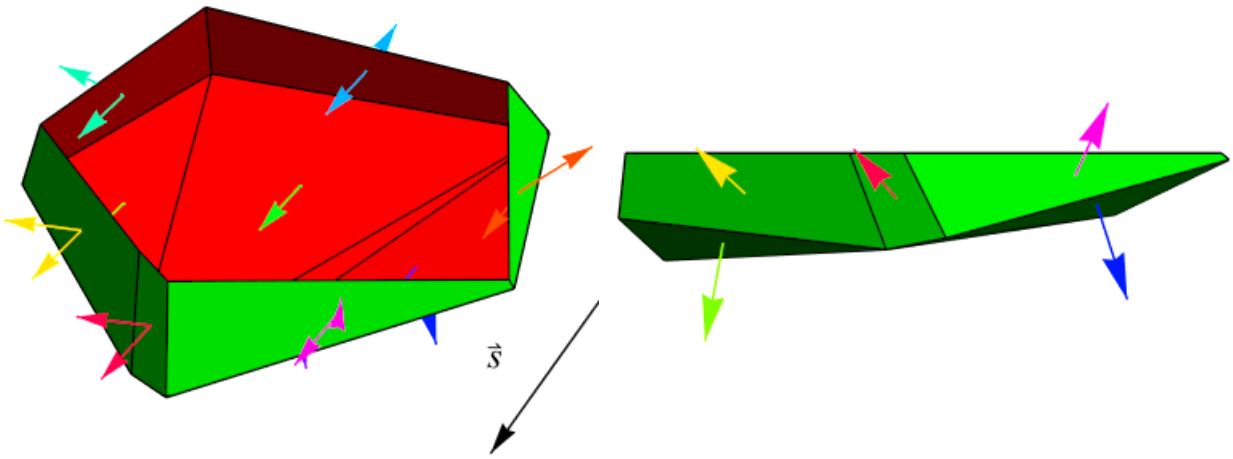
а б

Рисунок 1.36 – Угол видимости точки на поверхности: а – при ближнем наблюдателе (точка s), б – при дальнем наблюдателе (вектор $\vec{s}$)

определения видимости грани можно вычислить скалярное произведение v между двумя векторами:

$$v = \vec{n} \cdot \vec{s} = N^T S', \tag{1.22}$$

где N и S' – координатные столбцы векторов $\vec{n}$ и $\vec{s}$. Эту формулу можно обобщить для ближнего и дальнего наблюдения, для чего введём величину h_s – **индикатор близости наблюдателя** – которая равна 0 при дальнем наблюдателе и 1 – при ближнем. Тогда формула (1.22) переписывается в следующем виде:



а б

Рисунок 1.37 – Полиэдр в разрезе и углы между внешними нормалями его граней и вектором $\vec{s}$ направления на дальнего наблюдателя (а) и его вид с точки зрения наблюдателя (б)

$$v = N^T (S - h_s A), \qquad (1.23)$$

где S – координатный столбец вектора направления на дальнего наблюдателя $\vec{s}$ при $h_s = 0$, либо координаты ближнего наблюдателя s при $h_s = 1$; A – координатный столбец точки a, лежащий на рассматриваемой грани. С учётом того, что всякая грань либо полностью видима, либо полностью невидима, точка a может быть выбрана произвольно, например, можно взять геометрический центр грани, координаты которого вычисляются как среднее арифметическое всех её вершин, либо просто одну из её вершин.

Итак, положим, что имеется некоторый выпуклый полиэдр, заданный в виде списка своих граней G_i, $i = \overline{1,m}$, которые, в свою очередь, представлены списком своих вершин A_{ij}, $j = \overline{1,m_i}$. Тогда **алгоритм Робертса** удаления невидимых рёбер этого полиэдра может быть описан следующим образом:

1. Вначале необходимо определить индикатор близости наблюдателя $h_s \in \{0,1\}$ и координатный столбец S, характеризующий наблюдателя (направление на него либо его позицию). Для удобства вычисления нормалей граней можно также вычислить геометрический центр M заданного полиэдра.
2. Вначале рассматривается первая грань: $i := 1$.
3. Для текущей грани вычисляется вектор внешней нормали $\vec{n}_i$, который противоположен вектору внутренней нормали, вычисляемому по формуле (1.16), где можно положить, например, $\vec{a} = \overrightarrow{A_{i1}A_{i2}}$, $\vec{b} = \overrightarrow{A_{i2}A_{i3}}$, $\vec{c} = \overrightarrow{A_{i1}M}$.
4. По формуле (1.23), куда вместо N подставляется координатный столбец вектора $\vec{n}_i$, а вместо A – координатный столбец любой из вершин грани G_i (например A_{i1}), вычисляется скалярное произведение v векторов, образующих угол видимости текущей грани.
5. Текущая грань может оказаться либо лицевой, либо невидимой, либо вырожденной. Это определяется следующим образом:
   5а. При $v > 0$ грань G_i является **лицевой**, и все её рёбра $A_{i1}A_{i2}$, $A_{i2}A_{i3}$, …, $A_{i,m_i-1}A_{im_i}$, $A_{im_i}A_{i1}$ подлежат рисованию.
   5б. При $v \le 0$ грань G_i является **невидимой** либо **вырожденной**. Ребра такой грани могут быть изображены, например, пунктирными линиями, либо не рисоваться вовсе.
6. Если $i = m$, то алгоритм завершается, при этом отрисованы все видимые грани и их рёбра. Если же $i < m$, то происходит переход к следующей грани: $i := i + 1$, переход к шагу 3.

Основное достоинство алгоритма Робертса заключается в том, что порядок граней может быть выбран произвольно – он не влияет ни на результат, ни на скорость вычислений. Реализация некоторых шагов данного алгоритма

(вычисление геометрического центра, рисование видимых рёбер), впрочем, может сильно зависеть от структуры данных, которой представляется полиэдр.

Замечание. Алгоритма Робертса обладает линейной трудоёмкостью относительно числа граней (так как проверяется видимость каждой грани), вершин (для вычисления внутренней точки полиэдра используются все его вершины) и рёбер (их нужно отрисовать) полиэдра.

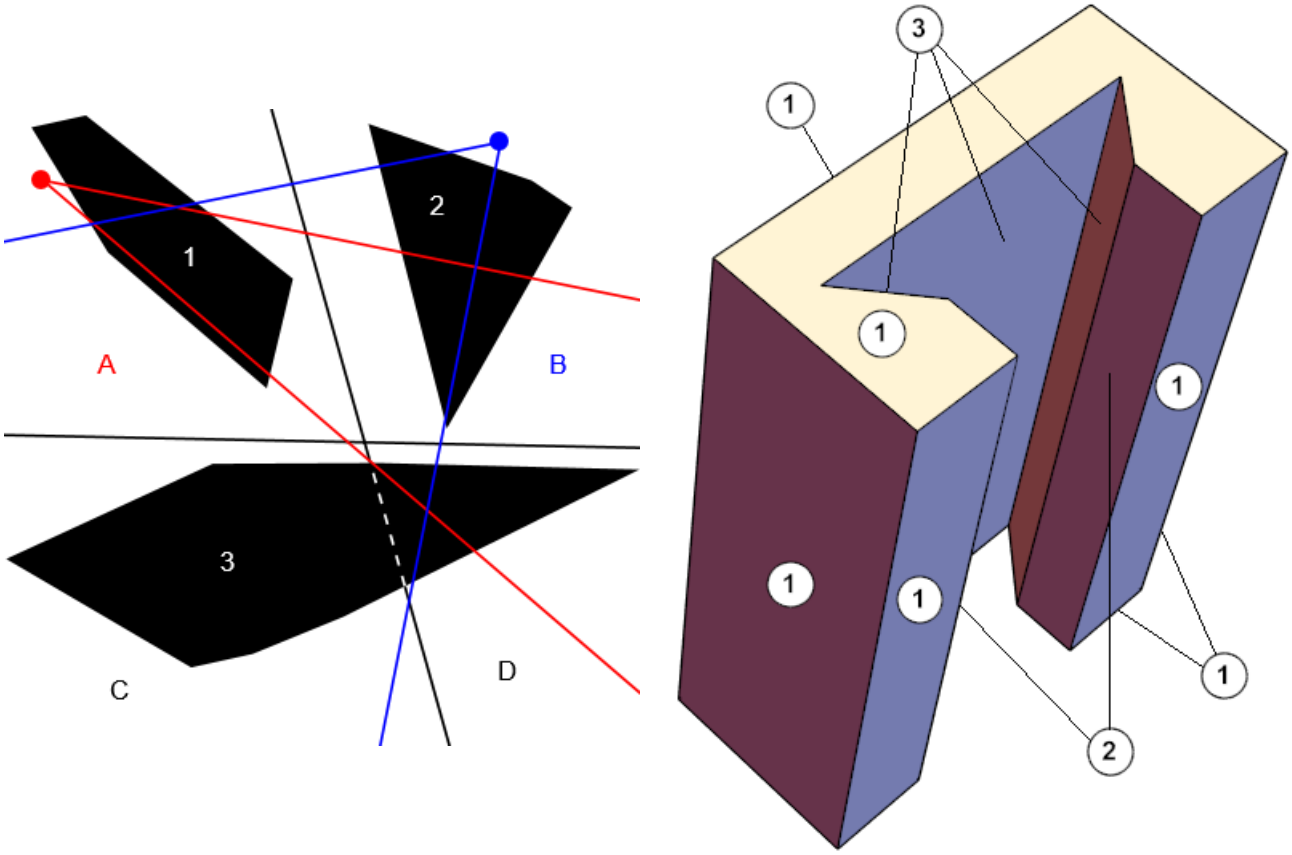
Модификация алгоритма Робертса для невыпуклого полиэдра. Идея метода **приоритетных списков**, предложенного Р. А. Шумакером и др. [16; 17, с. 24–27], заключается в следующем. Будем говорить, что один объект (полиэдр, грань и т.д.) имеет приоритет перед другим, если он перекрывает его. Каждому объекту может быть поставлено в соответствие натуральное число – приоритет – таким образом, что если один объект имеет приоритет перед другим, то ему должно быть присвоено меньшее число, нежели второму. При отрисовке сцены видимые объекты или их части изображаются в порядке убывания приоритетов.

При рассмотрении сцены с множеством полиэдров её нужно разбить на подмножества (кластеры), в пределах каждого из которых приоритеты объектов не меняются. Так, на рисунке 1.38а изображён горизонтальный вид сцены, разделённой на четыре кластера. Для всякой позиции наблюдателя в кластере A (красная точка и лучи) объект 1 имеет приоритет над объектами 2 и 3, а для наблюдателя в кластере B (синяя точка и лучи) объект 2 имеет наименьший приоритет.

Если же рассматривается один полиэдр, то приоритеты его граней зачастую можно прикинуть исходя из следующих принципов (Рисунок 1.38б):

- грани, которые не могут экранировать друг друга, имеют равный приоритет;
- чем глубже и ближе к центру полиэдра расположена грань, тем больший приоритет она имеет.

Для определения приоритетов граней достаточно сложных полиэдров могут использоваться различные структуры данных, например BSP-деревья [18]. Отрисовка полиэдра с BSP-деревом для его граней происходит посредством обхода дерева, осуществляемого в зависимости от позиции наблюдателя. По мере обхода для каждой грани определяется её видимость согласно формуле (1.22) или (1.23), и если текущая грань является видимой, то она подлежит рисованию. Таким образом, по мере отрисовки полиэдра происходит перекрытие менее приоритетных граней более приоритетными (Рисунок 1.39).



а б

Рисунок 1.38 – Примеры разбиения сцены на кластеры (а) и расстановки приоритетов граней невыпуклого полиэдра (б)

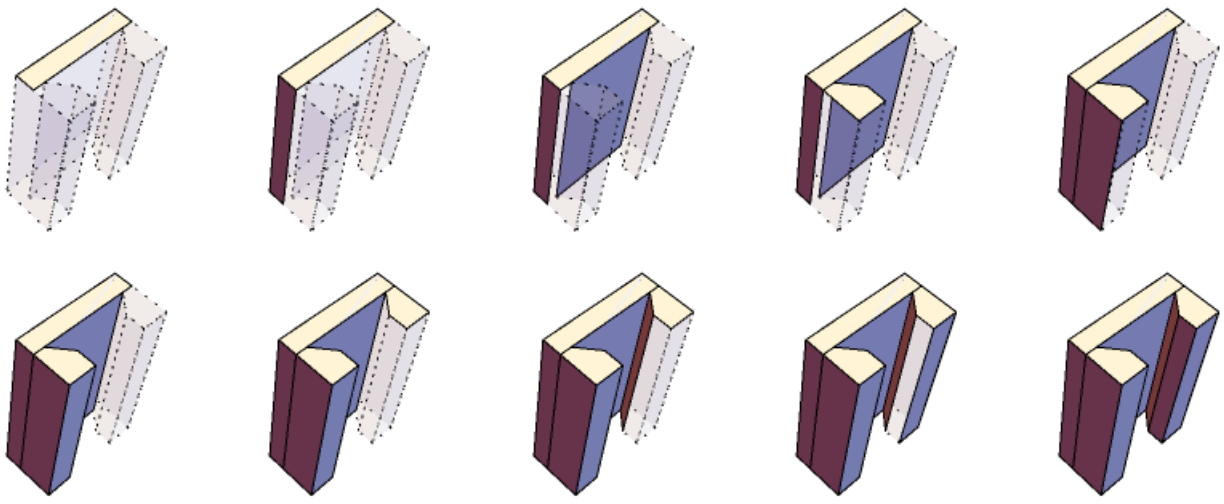


Рисунок 1.39 – Пример отрисовки полиэдра на основе BSP-дерева

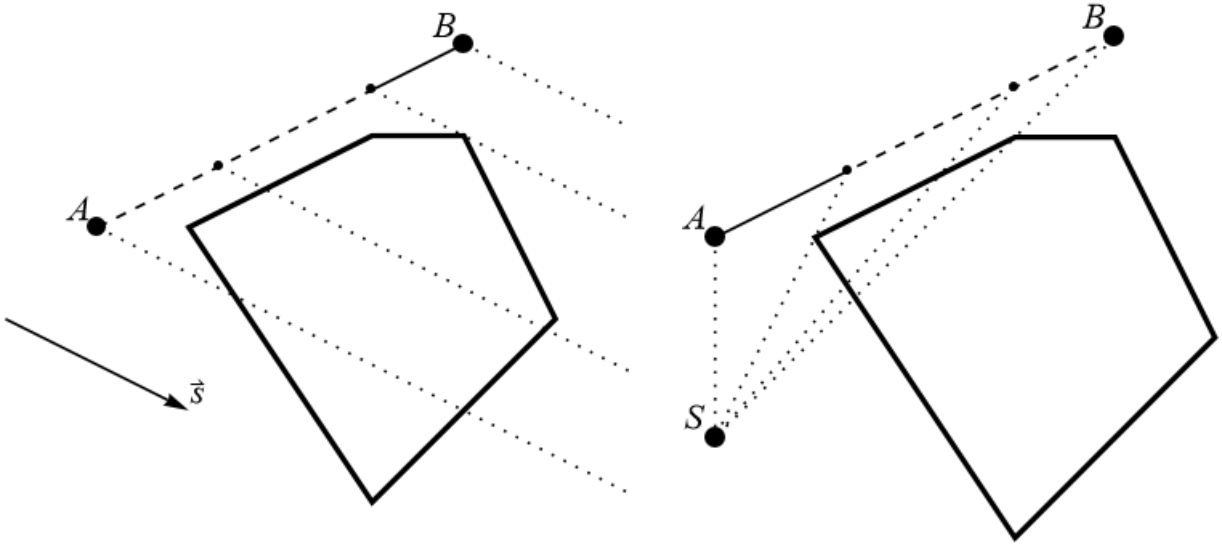
1.2.3. Удаление отрезка при экранировании выпуклым полиэдром

Для отрисовки полиэдров, экранируемых другими полиэдрами, для каждого из его видимых рёбер на основании его матрицы тела вычисляется, какая часть ребра перекрывается тем или иным полиэдром [15, с. 66–69].

Рассмотрим отрезок *AB*, который снова предста́вим в параметрическом виде: $\vec{r}(t) = \overrightarrow{OA} + t\overrightarrow{AB},\ t \in [0,1]$. Рассмотрим также полиэдр с матрицей тела $\mathcal{C}$ в виде (1.18), который для заданного отрезка служит внутренним отсекателем. Идея Робертса для нахождения невидимой части этого отрезка заключается в том, чтобы отрезок или какую-то его отдельную точку приближать к наблюдателю, и если рано или поздно произойдёт проникновение внутрь полиэдра, то это означает, что отрезок или рассматриваемая точка загораживается этим полиэдром. Это представимо лучом либо отрезком, проведённым от точки отрезка *AB* в сторону наблюдателя: при дальнем наблюдении имеет место проведение луча с постоянным направляющим вектором (Рисунок 1.40а), при ближнем – проводится отрезок от точки отрезка *AB* к позиции наблюдателя (Рисунок 1.40б). На рисунке 1.40 видно, что фактически имеет место экранирование отрезка *AB* полигоном, представляющим собой сечение полиэдра плоскостью, проходящей через этот отрезок и наблюдателя.

Уравнения для проведённых лучей и отрезков имеют следующий вид:

$$\begin{aligned} \vec{l}_{\text{дал}}(t,\alpha) &= \overrightarrow{OA} + t\overrightarrow{AB} + \alpha\vec{s}, \alpha \geq 0, \\ \vec{l}_{\text{ближ}}(t,\alpha) &= \overrightarrow{OA} + (1-\alpha)t\overrightarrow{AB} + \alpha\overrightarrow{AS}, 0 \leq \alpha < 1, \end{aligned} \tag{1.24}$$



а б

Рисунок 1.40 – Экранирование отрезка полиэдром при дальнем (а) и ближнем (б) наблюдении (пунктиром обозначена невидимая часть отрезка)

где $\vec{l}_{\text{дал}}(t,\alpha)$ и $\vec{l}_{\text{ближ}}(t,\alpha)$ – радиус-векторы точек, лежащих на них, $\vec{s}$ – вектор направления на дальнего наблюдателя, S – позиция ближнего наблюдателя. В матричном виде уравнения (1.24) переписываются в следующем виде:

$$\begin{aligned}\tilde{L}_{\text{дал}}(t,\alpha) &= \tilde{A} + t\left(\tilde{B} - \tilde{A}\right) + \alpha\tilde{S}', \alpha \geq 0,\\ \tilde{L}_{\text{ближ}}(t,\alpha) &= \tilde{A} + (1-\alpha)t\left(\tilde{B} - \tilde{A}\right) + \alpha\left(\tilde{S} - \tilde{A}\right), 0 \leq \alpha < 1,\end{aligned} \tag{1.25}$$

где $\tilde{L}_{\text{дал}}(t,\alpha)$ и $\tilde{L}_{\text{ближ}}(t,\alpha)$ – координатные столбцы точек с радиус-векторами $\vec{l}_{\text{дал}}(t,\alpha)$ и $\vec{l}_{\text{ближ}}(t,\alpha)$, $\tilde{A}$ и $\tilde{B}$ – координатные столбцы точек A и B, $\tilde{S}'$ – координатный столбец вектора $\vec{s}$ при дальнем наблюдении, $\tilde{S}$ – координатный столбец точки S при ближнем наблюдении (все в однородных координатах, причём последний элемент у координатных столбцов точек должен быть равным 1, векторов – 0). Обобщить формулы (1.25) можно при помощи индикатора близости наблюдателя:

$$\tilde{L}(t,\alpha) = \tilde{A} + (1-h_s\alpha)t\left(\tilde{B} - \tilde{A}\right) + \alpha\left(\tilde{S} - h_s\tilde{A}\right),\ 0 \leq \alpha < \alpha_{\max}, \tag{1.26}$$

где $\tilde{S} = (s_x, s_y, s_z, h_s)^T$ теперь обозначает координатный столбец в однородных координатах вектора $\vec{s}$ либо точки S, $\alpha_{\max} = +\infty$ при $h_s = 0$ и $\alpha_{\max} = 1$ при $h_s = 1$.

Таким образом, задачу внутреннего отсечения отрезка AB выпуклым полиэдром можно поставить следующим образом: необходимо найти все значения $t \in [0,1]$, для каждого из которых найдётся такое значение $\alpha \in [0, \alpha_{\max})$, что точки, задаваемые радиус-векторами (1.24) и координатными столбцами (1.25) и (1.26), окажутся внутри этого полиэдра. При задании матрицы тела $\mathcal{C}$ полиэдра в виде (1.18) значения параметра t для внутренних точек должны удовлетворять следующему неравенству:

$$\mathcal{C}\tilde{A} + (1-h_s\alpha)t\mathcal{C}\left(\tilde{B} - \tilde{A}\right) + \alpha\mathcal{C}\left(\tilde{S} - h_s\tilde{A}\right) \succ 0.$$

Это неравенство можно переписать в виде системы, если наряду с обозначениями (1.19) и (1.20) ввести столбец $W = (w_1, w_2, ..., w_n)^T = \mathcal{C}\left(\tilde{S} - h_s\tilde{A}\right)$:

$$L_i(t,\alpha) = p_i + (1-h_s\alpha)tq_i + \alpha w_i \geq 0,\ i = \overline{1,n}, \tag{1.27}$$

где

$$w_i = A_i(s_x - h_s x_A) + B_i(s_y - h_s y_A) + C_i(s_z - h_s z_A). \tag{1.28}$$

В первую очередь для каждого из неравенств (1.27) необходимо проверить, выполняются ли они для всех значений $t \in [0,1]$ и $\alpha \in [0, \alpha_{\max})$ либо, напротив, нарушаются при всех этих значениях. Достаточным условием для этого является их выполнение (нарушение) для граничных значений заданного множества (т.е. $t = 0$, $t = 1$, $\alpha = 0$ и $\alpha \to \alpha_{\max}$), а также стационарных точек, вычисляемых исходя из частных производных функций $L_i(t,\alpha)$:

$$\frac{\partial L_i(t,\alpha)}{\partial t} = \frac{\partial}{\partial t}(p_i + (1-h_s\alpha)tq_i + \alpha w_i) = (1-h_s\alpha)q_i, \qquad (1.29)$$
$$\frac{\partial L_i(t,\alpha)}{\partial \alpha} = \frac{\partial}{\partial \alpha}(p_i + (1-h_s\alpha)tq_i + \alpha w_i) = -h_s t q_i + w_i.$$

Обе производные равны нулю тогда и только тогда, когда $h_s = 1$, $t = w_i/q_i$, $\alpha = 1$, либо когда $q_i = w_i = 0$. В последнем случае получается $L_i(t,\alpha) = p_i$, и его дальнейшее рассмотрение тривиально. Значит, достаточно проверить неравенства (1.27) только при граничных значениях параметров t и α.

Заметим также, что знак производной (1.29) зависит от знака q_i: при $q_i > 0$ получаем, что функция $L_i(t,\alpha)$ является возрастающей при неизменном значении α, при $q_i < 0$ – убывающей.

Пусть $q_i > 0$. Тогда неравенство (1.27) выполняется при всех значениях t и α из заданных промежутков, если оно выполняется при $t = 0$:

$$\forall \alpha \in [0, \alpha_{\max}): p_i + \alpha w_i > 0. \qquad (1.30)$$

При ближнем наблюдении $\alpha_{\max} = 1$, и для выполнения неравенства (1.30) необходимо и достаточно потребовать $p_i > 0$ и $p_i + w_i \geq 0$. При дальнем же наблюдении $\alpha_{\max} = +\infty$, поэтому выполняется $p_i > 0$ и $w_i \geq 0$. Два этих случая можно обобщить неравенствами $p_i > 0$ и $h_S p_i + w_i \geq 0$.

Пусть теперь $q_i < 0$. Тогда проверим неравенство (1.27) при $t = 1$:

$$\forall \alpha \in [0, \alpha_{\max}): p_i + (1-h_s\alpha)q_i + \alpha w_i = p_i + q_i + \alpha(w_i - h_s q_i) > 0. \qquad (1.31)$$

При ближнем наблюдении имеем $h_s = 1$ и $\alpha_{\max} = 1$, значит, при подстановке $\alpha = 0$ и $\alpha = 1$ получим соответственно $p_i + q_i > 0$ и $p_i + w_i \geq 0$. При дальнем наблюдении $h_s = 0$ и $\alpha_{\max} = +\infty$, и тогда неравенство (1.31) эквивалентно $p_i + q_i > 0$ и $w_i \geq 0$. Два случая обобщаются двумя неравенствами $p_i + q_i > 0$ и $h_s p_i + w_i \geq 0$.

Таким образом, получаем условия, при которых неравенство (1.27) выполняется при всех значениях параметров:

$$
\begin{gathered}
q_i > 0 \,\&\, p_i > 0 \,\&\, h_S p_i + w_i \geq 0, \\
q_i < 0 \,\&\, p_i + q_i > 0 \,\&\, h_s p_i + w_i \geq 0, \\
q_i = 0 \,\&\, p_i > 0.
\end{gathered} \tag{1.32}
$$

При выполнении хотя бы одного из этих условий неравенство (1.27) при заданном значении индекса i может быть отброшено из системы. Таким образом, в системе остаётся всего несколько неравенств, участвующих в вычислении искомых значений t.

Аналогично определяются условия, при которых неравенство (1.27) не выполняется ни при каких значениях параметров:

$$
\begin{gathered}
q_i < 0 \,\&\, p_i < 0 \,\&\, h_S p_i + w_i \leq 0, \\
q_i > 0 \,\&\, p_i + q_i < 0 \,\&\, h_s p_i + w_i \leq 0, \\
q_i = 0 \,\&\, p_i < 0.
\end{gathered} \tag{1.33}
$$

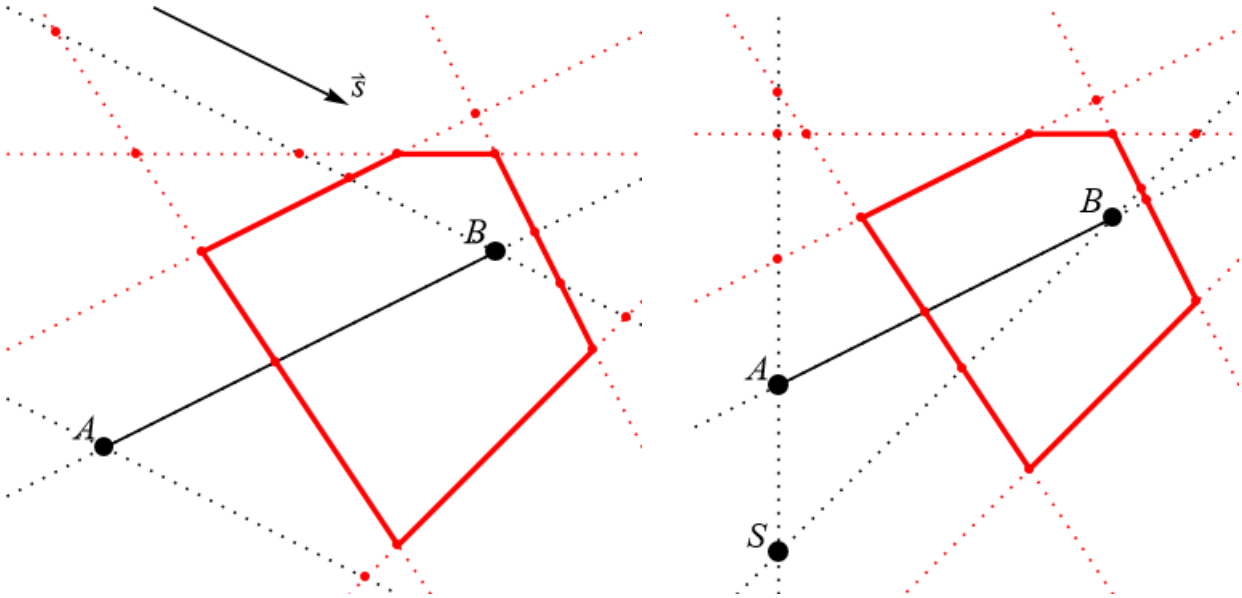
При выполнении хотя бы одного из этих условий одно из неравенств, а значит, и вся система (1.27) нарушается при всяких $t \in [0,1]$ и $\alpha \in [0, \alpha_{\max})$ – **отрезок не экранируется заданным полиэдром и не подлежит удалениям**.

Предположим теперь, что в системе (1.27) ни одно из неравенств не удовлетворяет условиям (1.32) и (1.33). Эту систему неравенств можно решить множеством способов и методов, однако Робертс предложил самый простой подход. Каждому из неравенств ставится в соответствие уравнение $L_i(t,\alpha) = 0$, которое описывает сечение некоторой грани исходного полиэдра плоскостью, определяемой отрезком AB и наблюдателем. Очевидно, это сечение представляет собой некоторую прямую (красные пунктирные прямые на рисунке 1.41). В каждой паре таких прямых они либо являются параллельными (что эквивалентно несовместности системы из двух уравнений $L_i(t,\alpha) = 0$ и $L_j(t,\alpha) = 0$), либо пересекается в точке, которой отвечают параметры (t_{ij}, α_{ij}). Для определения того, является ли эта точка граничной для исходного полиэдра, необходимо эти параметры подставить в систему (1.27). Вычисляются также решения систем каждого из уравнений $L_i(t,\alpha) = 0$ с одним из уравнений $t = 0$, $t = 1$, $\alpha = 0$ (чёрные пунктирные прямые на рисунке 1.41). Совокупность всех решений, удовлетворяющих системе (1.27), даст искомый промежуток t, соответствующий части исходного отрезка, подлежащей отсечению. Если ни одно из них не удовлетворяет системе (1.27), то отрезок не подлежит отсечению.

- При $t = 0$ получаем следующее решение уравнения $L_i(t,\alpha) = 0$:

$$
L_i(0,\alpha) = 0 \Rightarrow p_i + \alpha w_i = 0 \Rightarrow \underline{\alpha}_i = -p_i / w_i .
$$

При это должно выполняться



а б

Рисунок 1.41 – Для вычисления части отрезка *AB*, подлежащей удалению, достаточно вычислить все пересечения прямых (красные точки) и затем проверить для них неравенства исходной системы (а – при дальнем наблюдении, б – при ближнем)

$$\underline{\alpha}_i > 0 \Rightarrow -p_i / w_i > 0 \Rightarrow w_i \neq 0 \,\&\, \operatorname{sgn} p_i = -\operatorname{sgn} w_i,$$
$$h_s \underline{\alpha}_i < 1 \Rightarrow -h_s p_i / w_i < 1 \Rightarrow -\operatorname{sgn} w_i \cdot h_s p_i < |w_i|,$$

что эквивалентно

$$\begin{gathered} \sout{w_i \neq 0,} \\ \operatorname{sgn} p_i = -\operatorname{sgn} w_i, \\ h_s |p_i| < |w_i|. \end{gathered} \tag{1.34}$$

Первое условие зачёркнуто, так как оно следует из третьего. Если какое-то из двух условий (1.34) не выполняется, то это означает, что значение $\underline{\alpha}_i$ выходит за пределы промежутка $(0, \alpha_{\max})$, и это решение не учитывается. Если же условия (1.34) выполняются, и значения $t = 0$ и $\alpha = \underline{\alpha}_i$ удовлетворяют всем неравенствам (1.27), то тогда точка исходного отрезка, которой соответствует значение $t = 0$ (т.е. конец A), экранируется гранью, которой отвечает i-ое неравенство системы (1.27).

При подстановке $t = 0$ и $\alpha = \underline{\alpha}_i$ в неравенство (1.27) с произвольным индексом j получаем следующее:

$$L_j(0, \underline{\alpha}_i) = p_j + \underline{\alpha}_i w_j = p_j - \frac{p_i}{w_i} w_j \geq 0,$$

что с учётом (1.34) эквивалентно

$$\begin{gathered} |w_i| p_j - \operatorname{sgn} w_i \cdot p_i w_j \geq 0, \\ |w_i| p_j + |p_i| w_j \geq 0. \end{gathered} \tag{1.35}$$

Значит, при выполнении условий (1.34) и (1.35) при некотором i и любом $j \neq i$ точка A экранируется некоторой гранью полиэдра-отсекателя, а значит, исходный отрезок подлежит по крайней мере частичному удалению с этого конца.

- При $t = 1$ имеем следующее:

$$L_i(1,\alpha) = 0 \Rightarrow p_i + q_i + \alpha(w_i - h_s q_i) = 0 \Rightarrow \bar{\alpha}_i = -\frac{p_i + q_i}{w_i - h_s q_i}.$$

Для этого значения должны выполняться следующие условия:

$$\bar{\alpha}_i > 0 \Rightarrow -\frac{p_i + q_i}{w_i - h_s q_i} > 0 \Rightarrow w_i - h_s q_i \neq 0 \;\&\; \operatorname{sgn}(p_i + q_i) = -\operatorname{sgn}(w_i - h_s q_i),$$

$$h_s \bar{\alpha}_i < 1 \Rightarrow -h_s \frac{p_i + q_i}{w_i - h_s q_i} < 1 \Rightarrow -\operatorname{sgn}(w_i - h_s q_i) h_s (p_i + q_i) < |w_i - h_s q_i|.$$

Это эквивалентно следующему набору условий:

$$\begin{gathered} \sout{w_i - h_s q_i \neq 0,} \\ \operatorname{sgn}(p_i + q_i) = -\operatorname{sgn}(w_i - h_s q_i), \\ h_s |p_i + q_i| < |w_i - h_s q_i|. \end{gathered} \tag{1.36}$$

Аналогично предыдущему случаю, подставим $t = 1$ и $\alpha = \bar{\alpha}_i$ в неравенство (1.27) с произвольным индексом j:

$$L_j(1,\bar{\alpha}_i) = p_j + q_j + \bar{\alpha}_i(w_j - h_s q_j) = p_j + q_j - \frac{p_i + q_i}{w_i - h_s q_i}(w_j - h_s q_j) \geq 0.$$

После преобразований с использованием (1.36) получается следующее неравенство:

$$|w_i - h_s q_i|(p_j + q_j) + |p_i + q_i|(w_j - h_s q_j) \geq 0. \tag{1.37}$$

При выполнении условий (1.36) и неравенств (1.37) при некотором *i* и всех $j \neq i$ точка исходного отрезка, которой отвечает значение параметра $t = 1$ (т.е. конец *B*), экранируется соответствующей гранью исходного полиэдра.

- Для $\alpha = 0$ имеем следующее:

$$L_i(t,0) = 0 \Rightarrow p_i + tq_i = 0 \Rightarrow t_i = -p_i/q_i.$$

При подстановке в систему (1.27) этих значений получаются следующие неравенства:

$$L_j(t_i,0) = p_j + t_i q_j = p_j - \frac{p_i}{q_i} q_j \geq 0 \Rightarrow |q_i| p_j - \operatorname{sgn} q_i \cdot p_i q_j \geq 0.$$

Так как имеет смысл проверять эти неравенства, только если выполняется неравенство $t_i > 0,$ что эквивалентно $\operatorname{sgn} q_i = -\operatorname{sgn} p_i,$ то можно записать

$$|q_i| p_j + |p_i| q_j \geq 0. \tag{1.38}$$

Значит, если для некоторого i и любого $j \neq i$ выполняется условие $\operatorname{sgn} q_i = -\operatorname{sgn} p_i$ и неравенство (1.38), то точка отрезка *AB*, которой отвечает значение параметра $t_i = -p_i/q_i,$ экранируется некоторой гранью полиэдра-отсекателя.

- Пусть теперь имеется система двух уравнений $L_i(t,\alpha) = 0$ и $L_j(t,\alpha) = 0$:

$$\begin{cases} p_i + (1 - h_s\alpha)tq_i + \alpha w_i = 0, \\ p_j + (1 - h_s\alpha)tq_j + \alpha w_j = 0. \end{cases} \tag{1.39}$$

Найдём её решения на множестве $t \in (0,1),\ \ \alpha \in (0, \alpha_{\max})$. При $q_i = q_j = 0$ система (1.39) является несовместной, так как два сечения, описываемые уравнениями этой системы, параллельны. Если теперь положить $q_i \neq 0,\ q_j \neq 0,$ то из обоих уравнений системы (1.39) можно выразить *t*:

$$t = -\frac{p_i + \alpha w_i}{(1 - h_s\alpha)q_i} = -\frac{p_j + \alpha w_j}{(1 - h_s\alpha)q_j}.$$

Решая полученное уравнение относительно α, имеем $\alpha = \dfrac{q_i p_j - q_j p_i}{q_j w_i - q_i w_j}$. Если $q_j w_i - q_i w_j = 0$, то тогда система (1.39) является несовместной, так как два сечения оказались параллельными. Далее получаем значение для t: $t = -\left.\dfrac{p_i + \alpha w_i}{(1 - h_s \alpha) q_i}\right|_{\alpha = \frac{q_i p_j - q_j p_i}{q_j w_i - q_i w_j}} = \dfrac{p_j w_i - p_i w_j}{q_i w_j - q_j w_i + h_s (q_i p_j - q_j p_i)}$. Несложно видеть, что обе формулы также выполняются в случае, когда какой-то один из коэффициентов q_i либо q_j равен нулю. Если $h_s = 1$ и $q_i w_j - q_j w_i = -q_i p_j + q_j p_i$, то тогда $\alpha = 1$, а значение t не определено.

Таким образом, решением системы (1.39) является следующая пара значений параметров:

$$\alpha_{ij} = \frac{\Delta_{pq}}{\Delta_{qw}}, \quad t_{ij} = \frac{-\Delta_{pw}}{\Delta_{qw} - h_s \Delta_{pq}}, \tag{1.40}$$

где $\Delta_{pq} = \begin{vmatrix} p_i & p_j \\ q_i & q_j \end{vmatrix}$, $\Delta_{pw} = \begin{vmatrix} p_i & p_j \\ w_i & w_j \end{vmatrix}$, $\Delta_{qw} = \begin{vmatrix} q_i & q_j \\ w_i & w_j \end{vmatrix}$. Из них вычислению теперь подлежат оба значения α_{ij} и t_{ij}, хотя для α_{ij} можно также предварительно проверять следующие условия:

$$\alpha_{ij} > 0 \Rightarrow \frac{\Delta_{pq}}{\Delta_{qw}} > 0 \Rightarrow \Delta_{qw} \neq 0 \,\&\, \operatorname{sgn} \Delta_{pq} = \operatorname{sgn} \Delta_{qw},$$

$$h_s \alpha_{ij} < 1 \Rightarrow h_s \frac{\Delta_{pq}}{\Delta_{qw}} < 1 \Rightarrow \operatorname{sgn} \Delta_{qw} \cdot h_s \Delta_{pq} < \left|\Delta_{qw}\right|,$$

что эквивалентно

$$\begin{gathered} \cancel{\Delta_{qw} \neq 0,} \\ \operatorname{sgn} \Delta_{pq} = \operatorname{sgn} \Delta_{qw}, \\ h_s \left|\Delta_{pq}\right| < \left|\Delta_{qw}\right|. \end{gathered} \tag{1.41}$$

Кроме того, найденные значения α_{ij} и t_{ij} подлежат подстановке во все неравенства (1.27) с тем, чтобы проверить, действительно ли точка пересечения двух заданных сечений является граничной точкой исходного полиэдра.

Таким образом, **алгоритм удаления отрезка** *AB* **внутренним отсекателем в виде выпуклого полиэдра** с матрицей тела $\mathcal{C}$ вида (1.18) при

наблюдателе, характеризующимся координатным столбцом $\tilde{S} = (s_x, s_y, s_z, h_s)^T$, выглядит следующим образом:

1. Алгоритм начинается с первой грани: $i := 1$. Можно ввести также переменную – счётчик граней, не удовлетворяющих тривиальным случаям (1.32) и (1.33): $n' := 0$. Переменные, отвечающие этим граням, будут записываться в отдельный список $\mathcal{L}$.
2. Для текущей грани по формулам (1.20) и (1.28) вычисляются коэффициенты p_i, q_i и w_i.
3. Если выполняется по крайней мере одно из условий (1.33), то алгоритм немедленно завершается с возвращением пустого результата – **отрезок не подлежит удалению**.
4. Если выполняется по крайней мере одно из условий (1.32), то *i*-ая грань в дальнейшем игнорируется. В противном случае коэффициенты p_i, q_i, w_i необходимо записать в список $\mathcal{L}$ (как единый элемент этого списка), $n' := n' + 1$.
5. Если $i < n$, то происходит переход к следующей грани: $i := i + 1$, переход к шагу 2.
6. В результате выполнения шагов 2–5 в списке $\mathcal{L}$ всего n' элементов. Значения параметров для концов сегмента, подлежащего удалению, инициализируются следующим образом: $t_{\text{нач}} := 1$, $t_{\text{кон}} := 0$. Рассматривается первый элемент списка: $i := 1$.
7. Если $t_{\text{нач}} > 0$ и выполняются условия (1.34), а также неравенства (1.35) при всех значениях индекса *j*, отвечающих оставшимся после шагов 2–5 граням, то удаляется часть отрезка с конца *A*: $t_{\text{нач}} := 0$.
8. Если $t_{\text{кон}} < 1$ и выполняются оба условия (1.36), а также неравенства (1.37) при всех *j*, то удаляется часть отрезка с конца *B*: $t_{\text{кон}} := 1$.
9. Если выполнено равенство $\operatorname{sgn} q_i = -\operatorname{sgn} p_i$ и неравенства (1.38) для всякого *j*, то вычисляется значение $t_i := -p_i / q_i$. При этом может иметь место один или оба следующих случая:
   9а. Если оказалось верным неравенство $0 < t_i < t_{\text{нач}}$, то происходит расширение отсекаемого множества: $t_{\text{нач}} := t_i$.
   9б. Если $t_{\text{кон}} < t_i < 1$, то тогда $t_{\text{кон}} := t_i$.
10. Можно осуществлять проверку, выполняются ли равенства $t_{\text{нач}} = 0$ и $t_{\text{кон}} = 1$. Если да, то алгоритм немедленно завершается, возвращая весь исходный отрезок – **он полностью удаляется**.
11. Если $i < n'$, то происходит переход к следующей грани: $i := i + 1$, переход к шагу 7.
12. Теперь необходимо вычислять решения систем вида (1.39). Начинается рассмотрение всех пар элементов списка $\mathcal{L}$: $i := 1$.

13. Индекс второго элемента инициализируется номером, следующим за первым: $j := i + 1$.
14. При невыполнении условий (1.41) происходит переход к шагу 18.
15. По формулам (1.40) вычисляются значения α_{ij} и t_{ij}. Если нарушается хотя бы одно из неравенств (1.27), оставшихся после шагов 2–5, то происходит переход к шагу 18.
16. Если оказалось верным неравенство $0 < t_{ij} < t_{\text{нач}}$, то происходит расширение отсекаемого множества: $t_{\text{нач}} := t_{ij}$.
17. Если $t_{\text{кон}} < t_{ij} < 1$, то тогда $t_{\text{кон}} := t_{ij}$.
18. Если $j < n'$, то переопределяется индекс второго элемента: $j := j + 1$, переход к шагу 14.
19. Если $i < n' - 1$, то переопределяется индекс первого элемента: $i := i + 1$, переход к шагу 13.
20. Если выполняется неравенство $t_{\text{нач}} > t_{\text{кон}}$ (а такое возможно, только если до этого не найдено ни одного подходящего решения какой-либо из рассмотренных систем уравнений), то алгоритм возвращает **пустой результат**.
21. Алгоритм возвращает пару точек, которые вычисляются при помощи подстановки значений $t_{\text{нач}}$ и $t_{\text{кон}}$ в систему (1.4). Отрезок с концами в этих точках подлежит удалению.

Данный алгоритм выполняется за время $O(n) + O(n'^3)$. Возведение в куб в этой оценке возникает из-за шага 15, где в двойном цикле по переменным i и j, $1 \le i < j \le n'$, на каждом его шаге проверяется выполнение $O(n')$ неравенств.

Замечание 1. Представленный алгоритм может быть улучшен, если на шагах 12–19 рассматривать не все пересечения, удовлетворяющие системе (1.39), а только пересечения смежных сечений, т.е. вершины полигона – сечения полиэдра плоскостью, в которой проходит исходный отрезок при его приближении к наблюдателю. Для этого необходимо отсортировать список $\mathcal{L}$ по нормальным векторам соответствующих сечений, проведённых в секущей плоскости. Нетрудно заметить, что при замене $t' = (1 - h_s\alpha)t$ уравнение сечения можно переписать в виде $p_i + t'q_i + \alpha w_i = 0$, что соответствует прямой с нормальным вектором (q_i, w_i), направленным внутрь полигона. Эти векторы сортируются в порядке поворота по либо против часовой стрелки, что даёт последовательность уравнений $L_i(t, \alpha) = 0$, $i = \overline{1, n'}$, задающую соответственно право- либо левосторонний обход полигона. Сама эта сортировка выполняется за время $O(n' \log n')$, а дальнейшее вычисление параметров $t_{i,i+1}$ – за время $O(n')$.

Замечание 2. При подобном улучшении алгоритма значения $\alpha_{i,i+1}$ можно не вычислять, а только проверять условия (1.41). Кроме того, отпадает необходимость проверять неравенства (1.27) на шаге 15 данного алгоритма.

Замечание 3. При практической реализации данного алгоритма (как и многих других в компьютерной графике) следует учитывать тот факт, что при операциях с действительными числами возможна потеря точности, и вместо нуля может получиться число из достаточно малой его окрестности. Поэтому во всех неравенствах, где происходит сравнение с нулём (особенно в неравенстве (1.27)), вместо нуля рекомендуется задавать достаточно малое число ε такое, что всякое число из интервала $(-\varepsilon, \varepsilon)$ можно было бы считать равным нулю.

1.2.4. Алгоритм художника

Ещё одним алгоритмом, выводящим различные полигоны в порядке их приоритетов (см. подраздел 1.2.2), является **алгоритм художника**. Подобно художнику, который рисует вначале фон, потом поверх него – наиболее удалённые объекты, затем более близкие и т.д., этот алгоритм размещает полигоны (например, грани полиэдров и прочих объектов рассматриваемой сцены) в порядке их приближения к наблюдателю, причём по мере их отрисовки возможно перекрытие (экранирование) дальних полигонов ближними, которые отрисовываются позже.

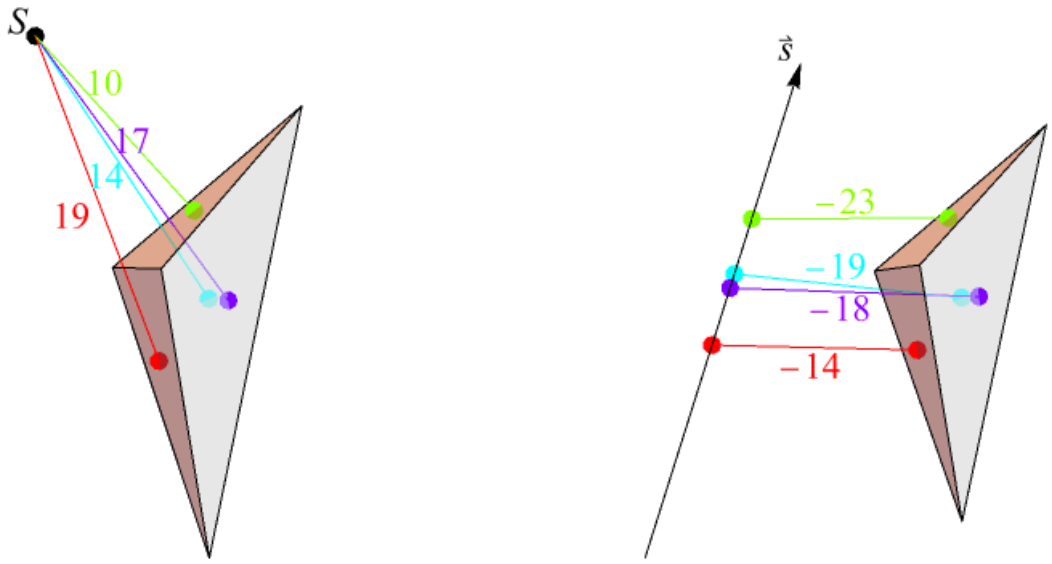
Для определения удалённости плоских полигонов от наблюдателя используется понятие **глубины**, определяемой для некоторой точки пространства. Так, для ближнего наблюдателя, расположенного в точке S, глубину точки M естественно определить как её расстояние от наблюдателя (Рисунок 1.42а):

$$d_S(M) = \left|\overrightarrow{SM}\right|.$$

При сравнении глубин разных точек, разумеется, более практичным является вычисление их квадратов:

$$(d_S(M))^2 = \left|\overrightarrow{SM}\right|^2 = \overrightarrow{SM}^2. \qquad (1.42)$$

В случае же дальнего наблюдателя, направление на которого выражается вектором $\vec{s}$, в качестве глубины точки M может выступать число, противоположное координате её ортогональной проекции M' на ось, проходящая в направлении вектора $\vec{s}$, например, через начало координат (Рисунок 1.42б). Как известно, величина направленного отрезка OM' может быть вычислена по формуле



а б

Рисунок 1.42 – Вычисление глубин точек при ближнем (а) и дальнем (б) наблюдении

$$OM' = -\frac{\vec{s} \cdot \overrightarrow{OM}}{|\vec{s}|}.$$

Заметим, что домножение обеих частей этой формулы на положительный множитель $|\vec{s}|$ даёт величину, которую также можно использовать в качестве глубины точки, ведь эта операция не влияет на сортировку точек по убыванию их глубин, а следовательно, на порядок их рисования. Таким образом, при дальнем наблюдателе может использоваться следующая формула:

$$d_{\vec{s}}(M) = -\vec{s} \cdot \overrightarrow{OM}. \qquad (1.43)$$

На рисунке 1.42 для каждой точки указана её глубина, вычисленная по формулам (1.42) и (1.43).

Зная глубины точек, формирующих подлежащие рисованию полигоны, можно некоторым образом определить глубины этих полигонов. В самом простом случае в качестве глубины полигона полагают глубину её средней точки, после чего происходит рисование всех граней в порядке убывания их глубин. Подобный подход, однако, характеризуется некоторыми недостатками.

Одним из них является возможность рисования полигонов в неверном порядке, особенно граней одного и того же полиэдра. На рисунке 1.43 представлен полиэдр, содержащий пять вершин и шесть граней, а также значения глубин этих граней. При последовательной отрисовке граней в порядке убывания их глубин вначале изображается красная грань (лицевая), которая затем перекрывается жёлтой, зелёной и бирюзовой гранями (невидимые). После этого рисуется синяя грань (лицевая), частично перекрывающая предыдущие три. Наконец, проекция пурпурной грани

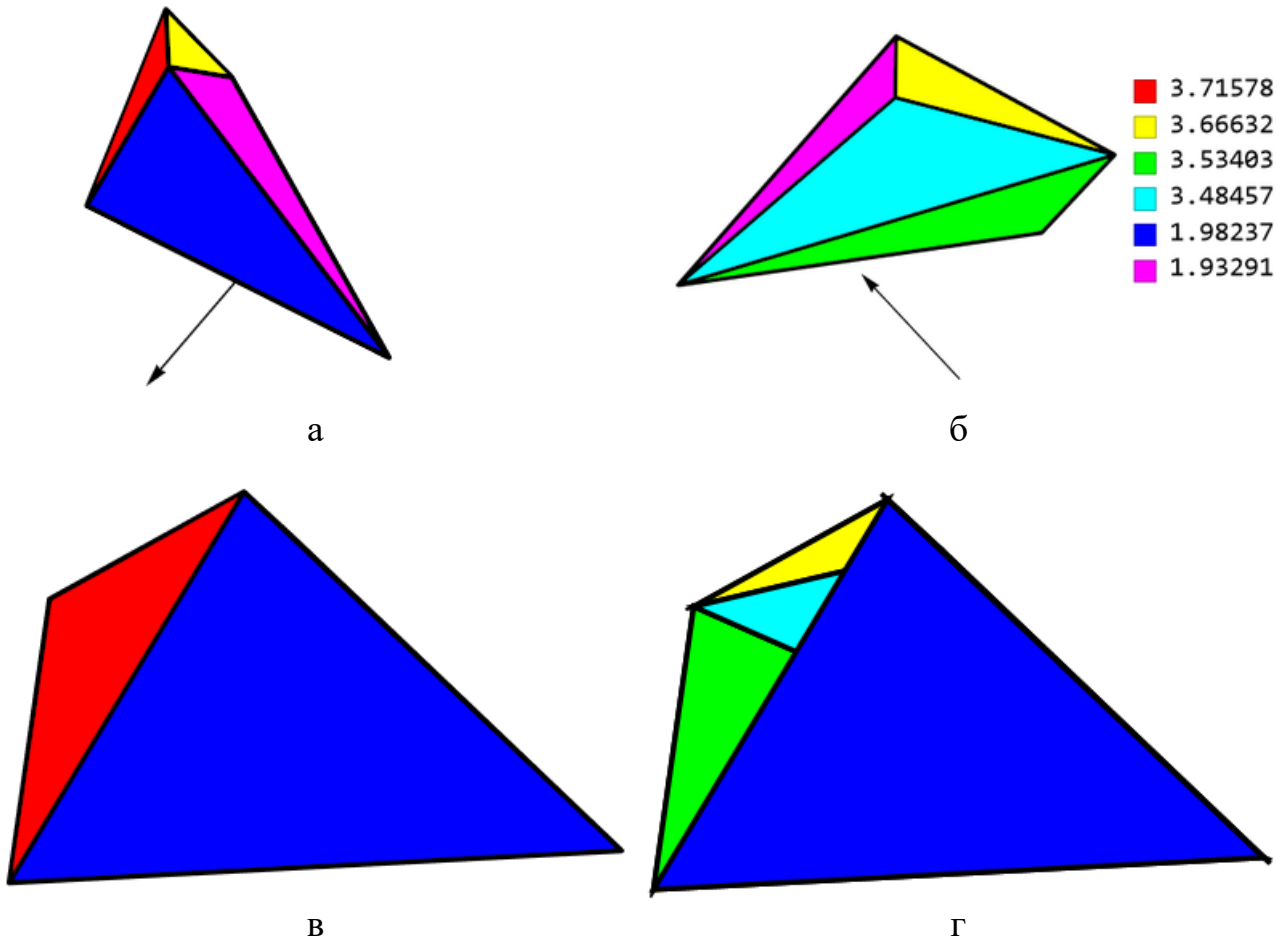


Рисунок 1.43 – Отрисовка граней полиэдра: а – вид спереди, б – вид сзади, в – правильная отрисовка граней полиэдра, г – отрисовка граней алгоритмом художника

вырождается в отрезок, поэтому на рисунке 1.43г она не видна. Данная проблема может быть решена отрисовкой только лицевых граней полиэдров либо разбиением каждой грани на достаточно мелкие ячейки примерно равных размеров.

Вторым, пожалуй, более серьёзным недостатком является невозможность отрисовки взаимно экранирующих друг друга полигонов (Рисунок 1.44). При использовании алгоритма художника для их визуализации один из полигонов обязательно перекроет все остальные, что приведёт к построению неверного

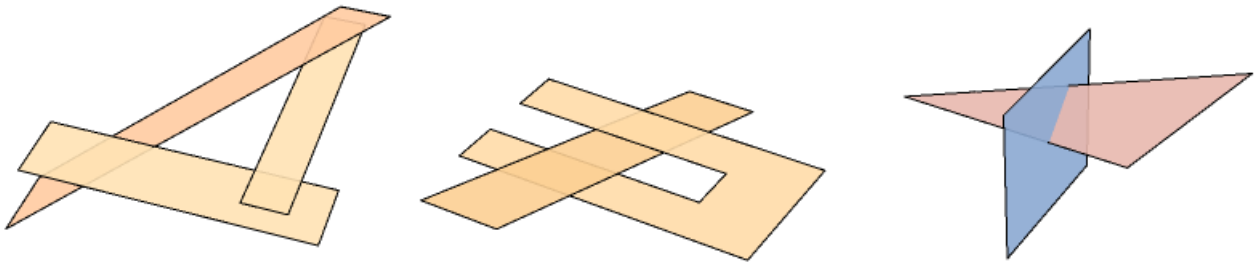


Рисунок 1.44 – Примеры взаимно экранирующих полигонов

изображения. Наиболее простым решением этой проблемы опять же является разбиение полигонов на мелкие ячейки.

Другим подходом для обработки взаимно экранирующих полигонов и граней является **алгоритм Ньюэлла (Ньюэлла-Ньюэлла-Санча)**, предложенный М. Ньюэллом, Р. Ньюэллом и Т. Санча в 1972 году [цит. по: 5, с. 330–333]. Суть данного алгоритма заключается в том, что для каждого исходного полигона G_i вычисляются максимальная и минимальная глубина (соответствующие некоторым его вершинам), что даёт габаритную глубину $[d_{i\min}, d_{i\max}]$ этого полигона. После этого список всех полигонов сортируется в порядке убывания их максимальных глубин $d_{i\max}$. Обозначим полигоны этого отсортированного списка G_1, G_2, …, G_N. Далее рассматривается первый полигон этого списка. Могут иметь место следующие ситуации:

1. Габаритные глубины полигонов G_1 и G_2 не пересекаются: $d_{1\min} \geq d_{2\max}$. Это означает, что полигон G_1 не экранирует ни G_2, ни остальные полигоны (на рисунке 1.45а представлена сцена с двумя полигонами, удовлетворяющими этому условию, а также ось, направленная на дальнего наблюдателя, с отмеченными на ней проекциями полигонов), что позволяет сразу же вывести проекцию G_1 на экран. Полигон G_1 подлежит удалению из списка.

2. В случае, если имеют место неравенства $d_{1\min} < d_{i\max}$ при $i = 2, 3, \ldots, k$, возможно частичное или полное экранирование полигоном G_1 произвольного числа (в том числе ни одного) полигонов G_2, G_3, …, G_k. Следовательно, для каждого из них необходимо проверить ещё некоторые условия.

3. Обозначим проекцию полигона G_i через G_i'. При задании в картинной плоскости экранной системы координат Oxy всякая проекция G_i' характеризуется своим габаритным прямоугольником $\{(x, y): x_{i\min} \leq x \leq x_{i\max}, y_{i\min} \leq y \leq y_{i\max}\}$. Достаточным условием того, чтобы габаритные прямоугольники проекций G_1' и G_i' не пересекались (а следовательно, и сами проекции тоже), является непересечение числовых отрезков $[x_{1\min}, x_{1\max}]$ с $[x_{i\min}, x_{i\max}]$ или $[y_{1\min}, y_{1\max}]$ с $[y_{i\min}, y_{i\max}]$. В случае, если хотя бы одна из этих пар отрезков не пересекается (или, быть может, примыкают своими концами друг к другу), полигон G_1 заведомо не экранирует полигон G_i (например, на рисунке 1.45б габаритные прямоугольники не пересекаются вследствие выполнения неравенства $y_{1\max} < y_{i\min}$).

4. Если условие из пункта 3 нарушено, то далее проверяется, находится ли исходный полигон G_1 целиком за плоскостью полигона G_i (Рисунок 1.45в). Это легко проверить, восстановив общее уравнение плоскости и затем вычислив расположение всех вершин G_1 и наблюдателя относительно неё. Если все вершины полигона G_1 находятся по одну, а наблюдатель – по другую сторону от плоскости полигона G_i, то это означает, что G_1 отделяется плоскостью полигона G_i от наблюдателя, а следовательно, не экранирует G_i.

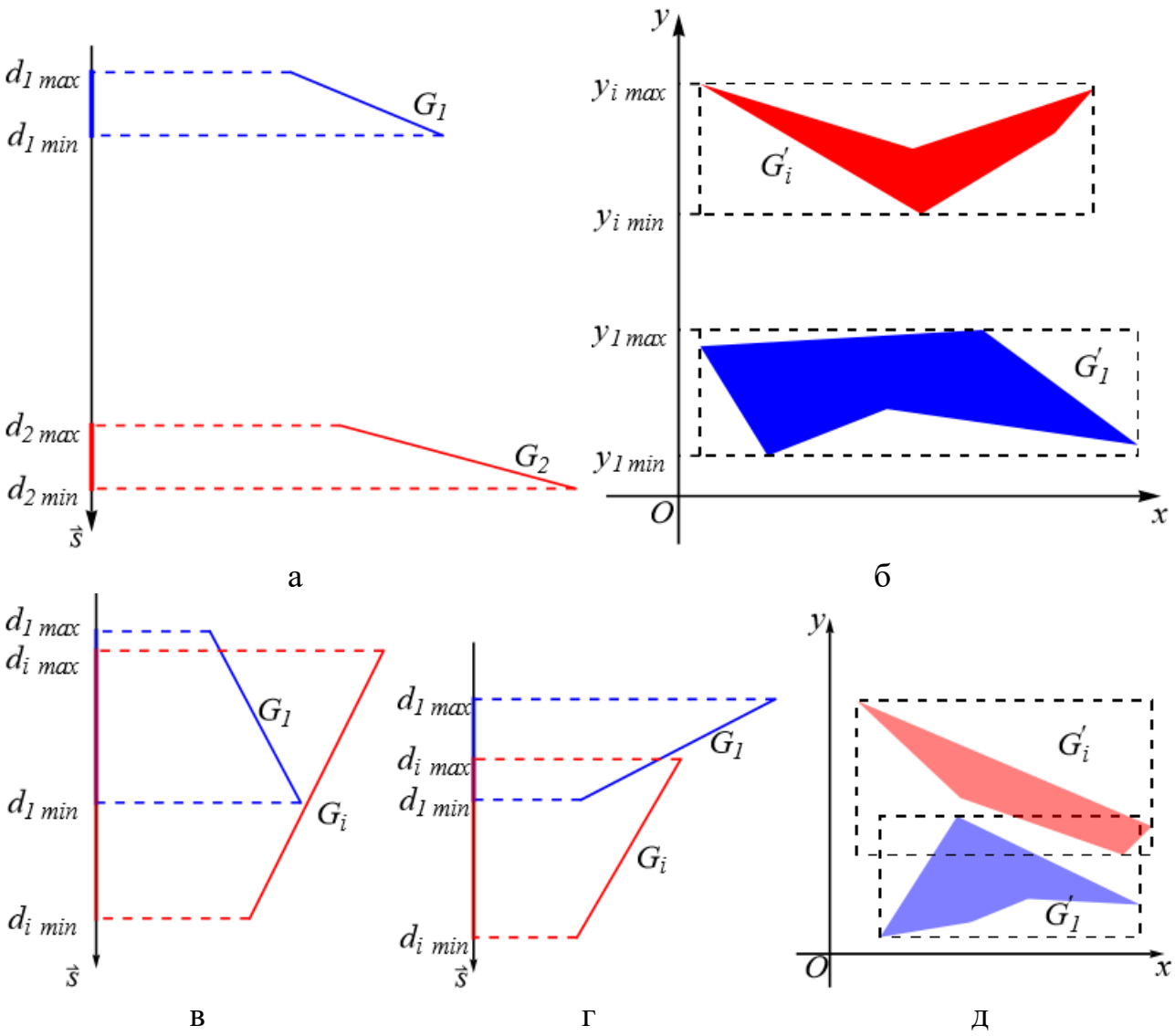


Рисунок 1.45 – Различные варианты расположения полигона G_1 относительно других (а, в, г – вид сверху, все полигоны лежат в вертикальных плоскостях; б, д – картинная плоскость с проекциями исходных полигонов)

5. При нарушении условия из пункта 4 осуществляется проверка того, расположен ли полигон G_i целиком перед плоскостью G_1 (Рисунок 1.45г). Если это так, то снова получаем, что G_1 не экранирует G_i.

6. При невыполнении всех предыдущих условий вычисляется пересечение проекций G_1' и G_i'. Если оно пустое либо имеет нулевую площадь (как на рисунке 1.45д), то G_1 не экранирует G_i.

7. Если для каждого $i = \overline{2,k}$ выполняется хотя бы одно из условий 3–6, то полигон G_1 не экранирует ни один другой, и его проекция может быть немедленно изображена на экране.

8. При нарушении всех вышеперечисленных условий для некоторого i полигон G_i должен быть переставлен в списке перед G_1. При этом полигон G_1

помечается как уже рассмотренный и в дальнейшем не может быть переставлен при рассмотрении первого полигона обновленного списка.

Таким образом, после выполнения этих шагов в новом списке будут представлены вначале полигоны, которые по крайней мере частично перекрываются плоскостью полигона G_1, потом уже помеченный полигон G_1, а затем полигоны, перекрывающие G_1. При этом в обеих частях этого списка сохраняется ранее определённый порядок сортировки, а сам помеченный полигон G_1, вообще говоря, его нарушает. Значит, при рассмотрении списка, содержащего помеченные полигоны, в представленные выше шаги следует внести следующие поправки:

- На шаге 2 выбираются для дальнейшего рассмотрения не только полигоны G_i, $i = \overline{2,k}$, но и **все помеченные полигоны**.
- На шаге 8 если полигон G_i является помеченным, а следовательно, как указано выше, не может быть переставлен в начало списка, то происходит разрезание полигона G_1 плоскостью полигона G_i. Результатом этого разрезания является некоторое множество новых полигонов, подлежащих добавлению в текущий список. Перед этим добавлением исходный полигон G_1 удаляется из списка.
- Каждый из новых полигонов вставляется либо **до первого помеченного полигона** в списке (если новый полигон находится **за** секущей плоскостью), либо **между** G_i **и следующим помеченным полигоном** (при расположении нового полигона **перед** секущей плоскостью). При этом во всех частях списка, разделённых помеченными ребрами, должен сохраняться исходный порядок сортировки.
- Новые полигоны после вставки в список не помечаются, так как может потребоваться их разрезание плоскостями других помеченных полигонов.
- После разрезания рассматривается новый список с новым первым полигоном.

Для разрезания полигона G_1 плоскостью полигона G_i могут использоваться различные алгоритмы отсечения произвольного полигона на плоскости (например, видоизменённый алгоритм Сазерленда-Ходжмана, где рассматривается отсекатель в виде прямой). При этом первоначально необходимо найти секущую прямую – пересечение плоскостей, в которых лежат полигоны G_1 и G_i. Кроме того, в плоскости полигона G_1 необходимо определить новую систему координат, в которой затем применяется выбранный алгоритм. В результате отсечения получаются новые полигоны, координаты вершин которых переводятся обратно в исходную систему координат трёхмерного пространства [3, с. 226].

Более простым, однако, является подход, согласно которому проверяется, по какую сторону от секущей плоскости расположены вершины рассматриваемого полигона. **Алгоритм разрезания плоского полигона**

$A_1A_2...A_nA_1$ **в трёхмерном пространстве плоскостью** П, проходящей через точку P перпендикулярно вектору $\vec{n}$, выглядит следующим образом:

1. Определяются два пустых списка C_F и C_B для хранения вершин исходного полигона, расположенных соответственно перед плоскостью П (т.е. в той же стороне, куда направлен вектор $\vec{n}$, отложенный от некоторой точки плоскости) и за ней. Если вершина лежит на самой плоскости, то она должна быть добавлена в оба списка.

2. Вначале рассматривается первая вершина: $i := 1$. Вычисляется скалярное произведение $a := \vec{n} \cdot \overrightarrow{PA_1}$.

3. Если $a \geq 0$, то текущая вершина A_i добавляется в список C_F.

4. Если $a \leq 0$, то текущая вершина A_i добавляется в список C_B.

5. Вычисляется скалярное произведение для следующей вершины: $a' := \vec{n} \cdot \overrightarrow{PA_{i+1}}$.

6. Если два скалярных произведения a и a' разных знаков (т.е. выполняется равенство $\operatorname{sgn} a \cdot \operatorname{sgn} a' = -1$), то это означает, что две точки A_i и A_{i+1} лежат по разные стороны от плоскости П, а следовательно, отрезок A_iA_{i+1} пересекает её в некоторой новой точке A'. С учётом того, что скалярное умножение векторов является линейной операцией, координаты A' можно вычислить, основываясь на том, что должно выполняться $\vec{n} \cdot \overrightarrow{PA'} = 0$. Это осуществимо, например, при помощи следующей формулы:

$$\vec{r}' = \left|\frac{a'}{a'-a}\right|\vec{r}_i + \left|\frac{a}{a'-a}\right|\vec{r}_{i+1}, \qquad (1.44)$$

где $\vec{r}'$, $\vec{r}_i$ и $\vec{r}_{i+1}$ – радиус-векторы точек A', A_i и A_{i+1}.

7. Полученная точка A' записывается в оба списка C_F и C_B.

8. Если $i = n$, то алгоритм завершается и возвращает два списка C_F и C_B. В противном случае происходит переход к следующей вершине: $a := a'$, $i := i+1$, возврат к шагу 3.

Следует отметить, что результатом представленного алгоритма, как и в алгоритме Сазерленда-Ходжмана, являются, вообще говоря, самопересекающиеся полигоны (Рисунок 1.46). Тем не менее, и с такими полигонами алгоритм Ньюэлла работает корректно.

Итак, с учётом всех вышеуказанных замечаний **алгоритм Ньюэлла при дальнем наблюдателе, выраженном вектором** $\vec{s}(0,0,1)$, может быть реализован, например, следующим образом:

1. Пусть изначально имеются полигоны $G_i = A_{i1}A_{i2}...A_{i,n_i}A_{i1}$, $i = \overline{1,N}$. Для каждого из них вычисляются значения $d_{i\min} = \min\limits_{j=1,n_i}(-z_{ij})$, $d_{i\max} = \max\limits_{j=1,n_i}(-z_{ij})$, где z_{ij} – аппликата вершины A_{ij}. Также объявляется

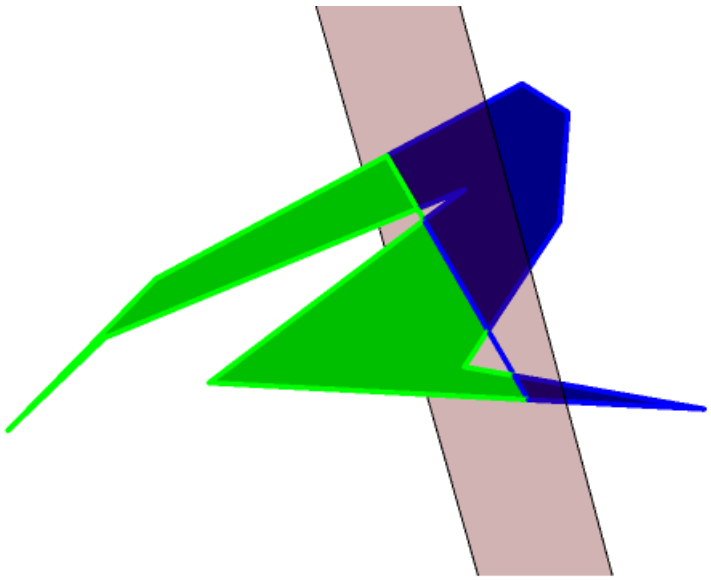


Рисунок 1.46 – Самопересекающиеся части полигона,
полученные в результате разрезания

список $\mathcal{G}$, куда будут записываться исходные полигоны или их части в порядке их рисования.

2. Составляется список D из полигонов G_i, отсортированных в порядке убывания соответствующих значений $d_{i\max}$. В дальнейшем через G_i будем обозначать i-ый элемент текущего списка D.
3. Проверяется, является ли первый полигон G_1 помеченным. Если да, то он удаляется из списка и помещается в результирующий список $\mathcal{G}$. Кроме того, происходит переход к шагу 10.
4. Если G_1 не помечен, то составляется новый список $D' := \varnothing$ из полигонов, которые потенциально могут хотя бы частично экранироваться им.
   - 4а. Осуществляется проход по списку D с его второго элемента: $i := 2$.
   - 4б. Если $i \le N$, $d_{1\min} < d_{i\max}$, и G_i не помечен, то G_i заносится в список D'. Происходит переход к следующему элементу: $i := i + 1$, этот шаг повторяется.
   - 4в. Если G_i является помеченным, то в начало списка D' добавляются все помеченные полигоны в том же порядке, в каком они представлены в списке D. Тем самым гарантируется, что вначале будет проверяться положение полигона G_1 относительно помеченных граней, затем при необходимости – относительно непомеченных.
5. Теперь необходимо выяснить, какие из полигонов списка $D' = \{G'_1, G'_2, ..., G'_{N'}\}$ по крайней мере частично перекрываются полигоном G_1. Для их хранения нужно составить ещё один новый список $D'' := \varnothing$. Если $N' = 0$, то сразу происходит переход к шагу 7.

5а. Начинается обход списка D' с его первого элемента: $i := 1$.

5б. Проверяется, пересекаются ли числовые отрезки $[x_{1\min}, x_{1\max}]$ и $[x'_{i\min}, x'_{i\max}]$ – минимальные и максимальные абсциссы вершин полигонов G_1 и G'_i соответственно. Если нет (в т.ч. если они касаются), то происходит переход к шагу 5к.

5в. Проверяется, пересекаются ли числовые отрезки $[y_{1\min}, y_{1\max}]$ и $[y'_{i\min}, y'_{i\max}]$ – минимальные и максимальные ординаты вершин полигонов G_1 и G'_i соответственно. Если нет (в т.ч. если они касаются), то происходит переход к шагу 5к.

5г. Проверяется, находятся ли все вершины G'_i перед плоскостью полигона G_1. Для этого восстанавливается нормальный вектор этой плоскости, направленный в полуплоскость наблюдателя: $\vec{n}_1 = \overrightarrow{A_{11}A_{12}} \times \overrightarrow{A_{12}A_{13}}$, а если аппликата этого вектора окажется отрицательной, то нужно положить $\vec{n}_1 := -\vec{n}_1$.

5д. Если для всякой вершины A'_{ij} полигона G'_i выполняется неравенство $\vec{n}_1 \cdot \overrightarrow{A_{11}A'_{ij}} \geq 0$, то происходит переход к шагу 5к.

5е. Проверяется, находятся ли все вершины G_1 за плоскостью полигона G'_i. Выполняется это аналогично шагам 5г и 5д: $\vec{n}'_i = \overrightarrow{A'_{i1}A'_{i2}} \times \overrightarrow{A'_{i2}A'_{i3}}$, при отрицательной аппликате полученного вектора необходимо положить $\vec{n}'_i := -\vec{n}'_i$.

5ж. Если для всякой вершины A_{1j} полигона G_1 выполняется $\vec{n}'_i \cdot \overrightarrow{A'_{i1}A_{1j}} \leq 0$, то происходит переход к шагу 5к.

5з. Проверяется, имеет ли пересечение проекций полигонов G_1 и G'_i ненулевую площадь. Для дальнего наблюдателя, характеризуемого вектором $\vec{s}(0,0,1)$, проекции легко получить, просто взяв абсциссы и ординаты всех вершин обоих полигонов. Для проверки достаточно выяснить, лежит ли проекция какой-либо вершины G_1 строго внутри проекции G'_i, лежат ли проекции вершин G'_i строго внутри проекции G_1, и не пересекаются ли между собой хотя бы по одному ребру обеих проекций. Если хотя бы одно из этих утверждений верно, это означает, что проекции пересекаются. Иначе нужно делать переход к шагу 5к.

5и. Если полигон G'_i является помеченным, то происходит переход к шагу 6. При этом список D'' остаётся пустым. Иначе G'_i добавляется в список D''.

5к. Если $i = N'$, то список D' закончился, и нужно перейти к шагу 7. Иначе происходит переход к следующему полигону: $i := i + 1$, возврат к шагу 5б.

6. Происходит разрезание полигона G_1 плоскостью полигона G_i', для которой на шаге 5е уже вычислен вектор нормали $\vec{n}_i'$, а также известна точка A_{i1}'. В результате получаются два полигона C_F и C_B. Далее редактируется список D.
   - 6а. Происходит поиск нужной позиции в списке D для добавления переднего полигона C_F. Для этого первоначально вычисляется его максимальная глубина $d_{F\max}$ аналогично шагу 1.
   - 6б. Обход списка D начинается с индекса $i' := i+1$.
   - 6в. Если $i' \le N$, $d_{F\max} < d_{i'\max}$, и $G_{i'}$ не является помеченным, то нужно перейти к следующему полигону: $i' := i'+1$, после чего этот же шаг повторяется. В противном случае происходит вставка полигона C_F в список D перед полигоном $G_{i'}$ (либо в самый конец списка при $i' = N+1$).
   - 6г. Первый элемент списка D заменяется на C_B.
   - 6д. Переход к шагу 10.
7. Полигон G_1 помечается. Он уже не может быть разрезан на шаге 6, но может сам оказаться отсекателем для любого полигона, который окажется спереди него в списке D в результате следующих шагов.
8. Из списка D удаляются все элементы списка D''.
9. В начало списка D добавляются все элементы D'' с сохранением их взаимного порядка.
10. Если в списке D осталось не менее двух элементов, происходит возврат к шагу 3. В противном случае единственный оставшийся элемент подлежит занесению в список $\mathcal{G}$, и алгоритм завершается.

Замечание 1. Алгоритм Ньюэлла не требует проверки того, являются ли все полигоны лицевыми гранями для своих полиэдров или других тел, однако подобная проверка позволяет уменьшить число первоначальных полигонов, оставив только лицевые, и упростить вычисления на шагах 5г и 5е, где используются уже вычисленные вектора их внешних нормалей.

Замечание 2. Алгоритм Ньюэлла используется для определения порядка рисования полигонов, граней и прочих поверхностей при конкретном положении наблюдателя и для отображения конкретной сцены. В ситуациях, когда имеет место динамическое наблюдение некоторой статической сцены, и меняется только положение наблюдателя, может использоваться подход Шумакера, согласно которому сцена разбивается на несколько кластеров, при нахождении наблюдателя в пределах каждого из которых приоритеты всех поверхностей остаются неизменными (см. рисунок 1.38а).

Замечание 3. При наблюдателе, отличном от дальнего наблюдателя с вектором $\vec{s}(0,0,1)$, перед использованием алгоритма Ньюэлла (как и при большинстве других алгоритмов) обычно производятся проективные предыскажения, переводящие наблюдателя в дальнее положение, направление на которого совпадает с положительным направлением оси аппликат мировой

системы координат [1, с. 180–181; 3, с. 436–438]. Это позволяет не затрачивать дальнейшее время и вычислительные ресурсы на проектирование всех полигонов – для нахождения проекций достаточно только взять абсциссы и ординаты всех вершин. Другим подходом является вычисление проекций для всех полигонов G_i, как исходных, так и получаемых в результате разрезаний, однако для этого необходимо вначале задать конкретную картинную плоскость, а также экранную систему координат на ней.

1.2.5. Алгоритм z-буфера

Одним из самых распространённых и важных алгоритмов компьютерной графики, связанных с удалением объектов и отрисовкой сцены, является **алгоритм z-буфера** [5, с. 321–329; 19], предложенный Э. Э. Кэтмулом в своей диссертации 1974 года [20], где он рассматривал разные вопросы визуализации криволинейных поверхностей. Этот алгоритм использует две матрицы, которые хранят информацию о проекциях объектов рассматриваемой сцены. Их размеры совпадают с разрешением экрана, а каждый элемент соответствует некоторому его **пикселю** (см. подраздел 1.3). Первая матрица – **буфер глубины** – является числовой матрицей, где каждый элемент равен глубине точки, представленной на экране соответствующим пикселем. При дальнем наблюдателе с вектором $\vec{s}(0,0,1)$ глубина точки определяется только его аппликатой – координатой z, откуда и происходит название алгоритма. Вторая матрица – **буфер кадра** – содержит цвета (которые могут быть представлены числами или числовыми векторами) точек, видимых наблюдателю и подлежащих выводу на экран.

Обработка произвольной поверхности некоторого объекта состоит из следующих шагов:

1. проектирование поверхности на картинную плоскость;
2. отсечение и растеризация полученной проекции, т.е. её приближение некоторым множеством пикселей;
3. сопоставление каждому пикселю точки исходной поверхности, которая отображается в него в результате проектирования на картинную плоскость;
4. обработка найденной точки с тем, чтобы в случае её видимости внести её глубину и цвет соответственно в буферы глубины и кадра.

Проектирование поверхности осуществляется в зависимости от типа и положения наблюдателя при помощи ортографического, косоугольного или центрального проектирования, которые были подробно рассмотрены в первой части данного ЭУМК [1, разд. 1.6].

Во втором из представленных шагов происходит отсечение регулярным окном, которое может быть проведено различными алгоритмами, рассмотренными ранее в подразделе 1.1. Для растеризации прямо- и криволинейных поверхностей также могут использоваться различные

алгоритмы. Суть одного из них, предложенных самим Кэтмулом, заключается в разбиении исходной поверхности на части, проекции каждой из которых содержат не более одного пикселя. Например, параметрически заданная поверхность $\vec{r} = \vec{r}(u,v) = (x(u,v), y(u,v), z(u,v)), \quad u \in (u_{\min}, u_{\max}), \quad v \in (v_{\min}, v_{\max})$ разбивается на четыре части двумя средними линиями $\vec{r} = \vec{r}\left(\frac{u_{\min} + u_{\max}}{2}, v\right)$ и $\vec{r} = \vec{r}\left(u, \frac{v_{\min} + v_{\max}}{2}\right)$. Если проекция какой-то из полученных частей содержит более одного пикселя, то к ней применяется та же операция. На рисунке 1.47 показан пример разбиения поверхности $\vec{r} = (u^2 - v^2, uv, v^2 + 2uv - u + v)$, $u \in (0,1.9), \ v \in (0,1.8)$.

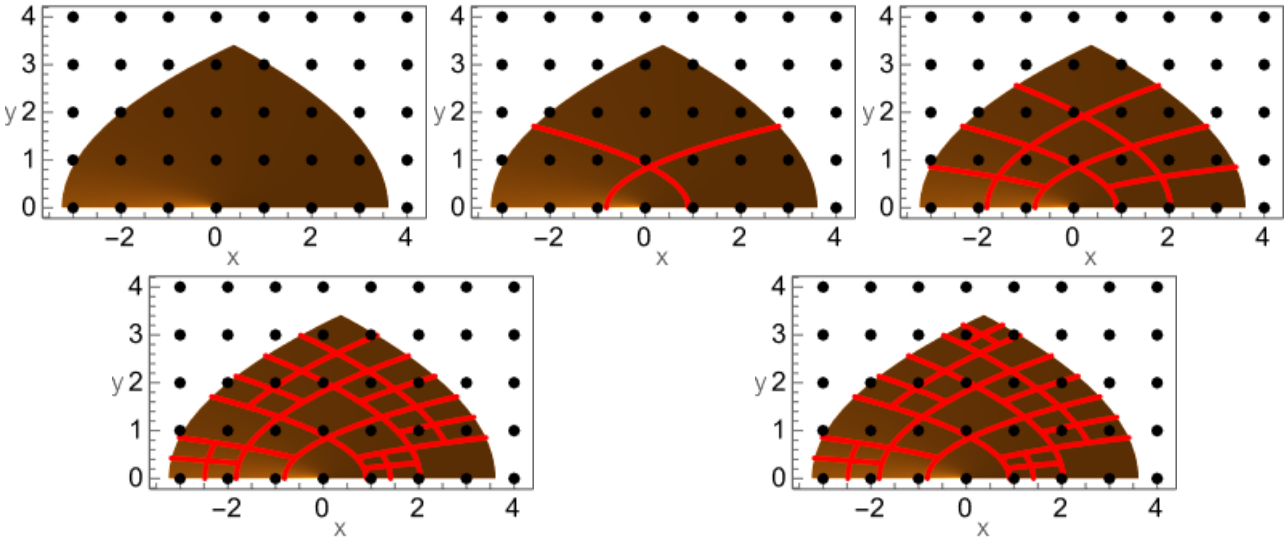


Рисунок 1.47 – Разбиение криволинейной поверхности для сопоставления каждому пикселю проекции некоторой части исходной поверхности

После растеризации каждому пикселю необходимо поставить в соответствие некоторую точку исходной поверхности. Выполняется это посредством проведения луча, направленного в сторону наблюдателя. Этот луч лежит на прямой, описываемой следующей системой параметрических уравнений:

$$\begin{cases} x = x_0 + (s_x - h_s x_0)t, \\ y = y_0 + (s_y - h_s y_0)t, \\ z = z_0 + (s_z - h_s z_0)t, \end{cases} \tag{1.45}$$

где (x_0, y_0, z_0) – координаты пикселя в системе координат сцены, (s_x, s_y, s_z) – координаты, которыми характеризуется наблюдатель (вектор направления на дальнего наблюдателя либо точка расположения ближнего наблюдателя), h_s – индикатор близости наблюдателя, равный 0, если наблюдатель дальний, либо 1,

если он является ближним. Заметим, что при таком задании луча бóльшему значению параметра t соответствует более близкая к наблюдателю точка, следовательно, в качестве её глубины может выступать величина, равная $-t$.

Пересечение прямой (1.45) с заданной поверхностью вычисляется исходя из её модели:

- Если поверхность задана неявным уравнением $f(x, y, z) = C$, то происходит подстановка в него координат точек прямой (1.45), после чего получается уравнение относительно t:

$$f(x_0 + (s_x - h_s x_0)t, y_0 + (s_y - h_s y_0)t, z_0 + (s_z - h_s z_0)t) = C. \qquad (1.46)$$

Оно решается для всех $t \in \mathbb{R}$ при дальнем наблюдателе и для $t \in (-\infty, 1)$ при ближнем, что даёт некоторое множество решений, среди которых нужно выбрать наибольшее значение. Этому значению соответствует одна точка поверхности, лежащая на прямой (1.45) и видимая наблюдателю (если только какая-то другая поверхность не загораживает её).

- Если поверхность задана параметрическими уравнениями $\vec{r} = \vec{r}(u, v) =$ $= (x(u, v), y(u, v), z(u, v))$, то имеет место следующая система уравнений:

$$\begin{cases} x(u,v) = x_0 + (s_x - h_s x_0)t, \\ y(u,v) = y_0 + (s_y - h_s y_0)t, \\ z(u,v) = z_0 + (s_z - h_s z_0)t, \end{cases} \qquad (1.47)$$

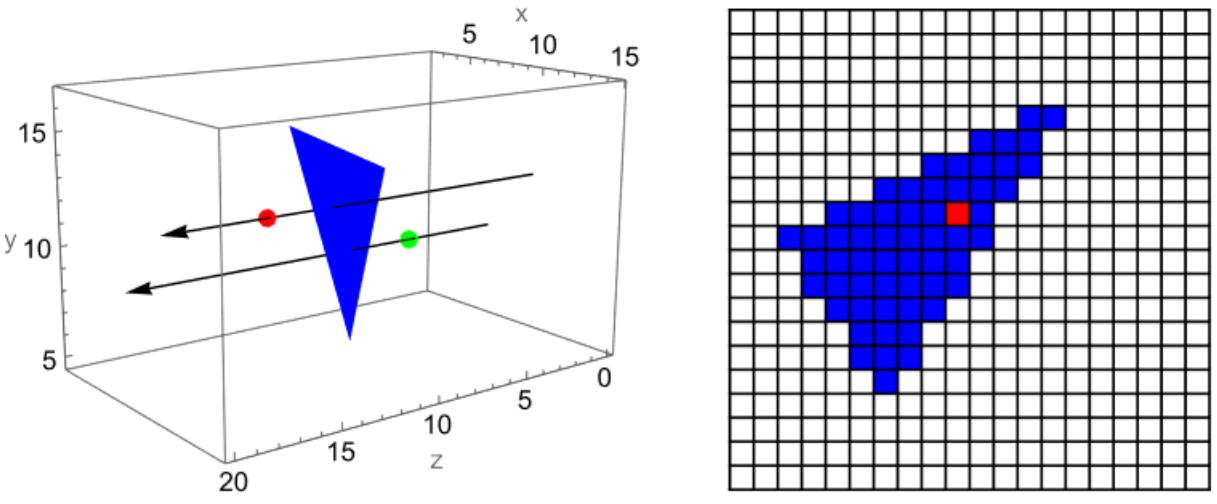
где в качестве неизвестных выступают параметры t, u и v. Опять же, среди полученных параметров t выбирается наибольший.

В частности, пересечение с плоскостью, которая, как известно, может быть задана обоими из этих способов, даёт ровно одну точку пересечения. Её параметр t вычисляется по одной из следующих формул:

$$\Pi : Ax + By + Cz + D = 0 \Rightarrow$$
$$\Rightarrow t = -\frac{Ax_0 + By_0 + Cz_0 + D}{As_x + Bs_y + Cs_z - h_s(Ax_0 + By_0 + Cz_0)}, \qquad (1.48)$$
$$\Pi : \begin{cases} x = x_1 + a_x u + b_x v, \\ y = y_1 + a_y u + b_y v, \\ z = z_1 + a_z u + b_z v \end{cases} \Rightarrow t = \frac{\Delta_t}{\Delta},$$

где $\Delta_t = \begin{vmatrix} x_1 - x_0 & a_x & b_x \\ y_1 - y_0 & a_y & b_y \\ z_1 - z_0 & a_z & b_z \end{vmatrix}$, $\Delta = \begin{vmatrix} s_x - h_s x_0 & a_x & b_x \\ s_y - h_s y_0 & a_y & b_y \\ s_z - h_s z_0 & a_z & b_z \end{vmatrix}$ – определители, вычисляемые согласно правилу Крамера решения системы линейных алгебраических уравнений.

После нахождения искомой точки поверхности нужно выяснить две вещи: является ли она видимой наблюдателю (точнее, не экранируется ли она какой-либо другой ранее рассмотренной поверхностью) и в какой цвет нужно окрасить соответствующий пиксель в случае её видимости. Первое проверяется путём сравнения глубины текущей точки с глубиной, записанной в соответствующем элементе буфера глубины: если глубина текущей точки меньше ранее записанной, то это означает, что текущая точка перекрывает соответствующий пиксель, ранее записанный в буфере кадра, что влечёт необходимость переопределения нужных элементов буферов глубины и кадра. Например, предположим, что в ходе визуализации сцены на рисунке 1.48а вначале отрисовывается синий треугольник, а затем – красная и зелёная точки. Пиксель, соответствующий красной точке, будет перекрашен, а пиксель, в который проектируется зелёная точка – нет (Рисунок 1.48б), так как при сравнении глубин этих точек с глубинами точек синего треугольника, с которыми они отображаются в общий пиксель (на рисунке 1.48а они лежат на осях, проведённых через красную и зелёную точки в направлении к наблюдателю), красная точка оказывается менее глубокой, зелёная – более глубокой (Рисунок 1.49). При визуализации более сложных объектов пиксели могут окрашиваться в усреднённые цвета соответствующих участков исходной поверхности, полученных ранее в результате её разбиения при растеризации.



а б

Рисунок 1.48 – Визуализация сцены с несколькими объектами (а) алгоритмом z-буфера, в результате которого в буфере кадра (б) сохраняется информация только о видимых объектах

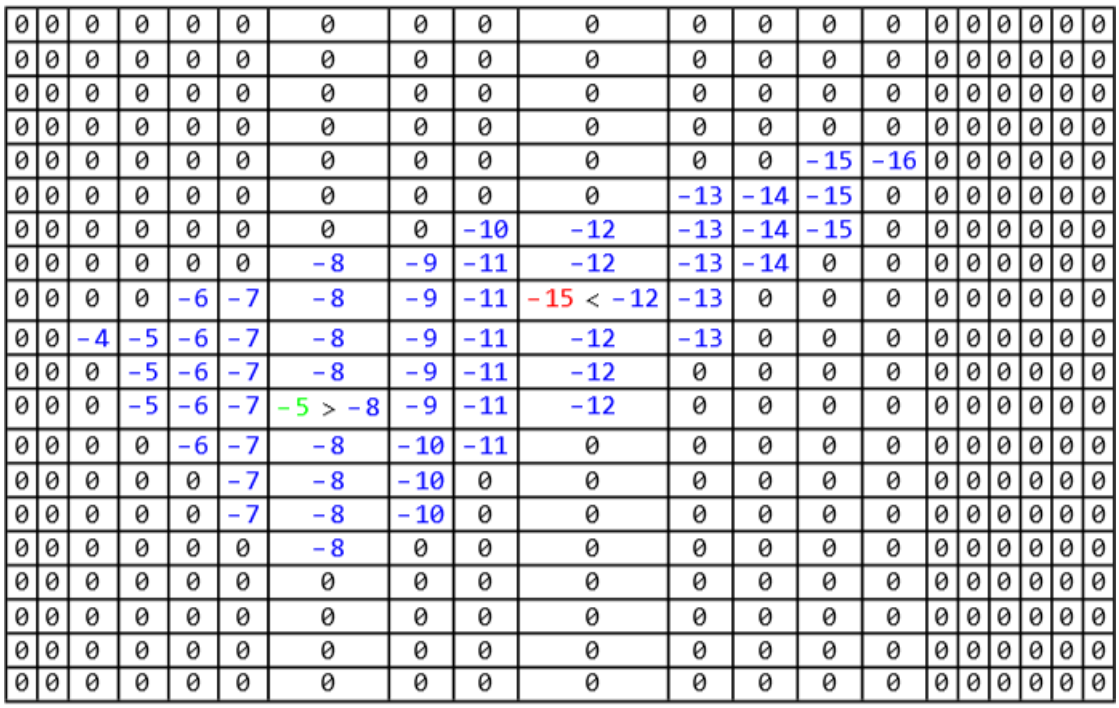


Рисунок 1.49 – Буфер глубины и сравнение глубин новых точек с его элементами (приведены округлённые значения)

Таким образом, **алгоритм z-буфера** отрисовки некоторого множества криволинейных поверхностей $\Xi = \{\Xi_1, \Xi_2, ..., \Xi_n\}$ можно представить следующей последовательностью шагов:

1. Исходя из разрешения экрана $w \times h$ пикселей (w – число пикселей в ширину, h – их число в высоту) определяются буфер глубины $Z = (z_{ij}) \in \mathbb{R}_{h,w}$ и буфер кадра $F = (f_{ij}) \in \mathbb{R}^c_{h,w}$ (где c – размерность числовых векторов, представляющих цвета), причём их необходимо инициализировать соответственно некоторым максимальным значением глубины (скажем, $d_0 = +\infty$) и фоновым цветом $\vec{c}_ф$: $z_{ij} := d_0$, $f_{ij} := \vec{c}_ф$, $i = \overline{1,h}$, $j = \overline{1,w}$.
2. Поверхности рассматриваются в произвольном порядке, например, начиная с первой: $k := 1$.
3. Текущая поверхность Ξ_k проектируется на картинную плоскость, в результате чего получается её проекция Ξ'_k.
4. Для проекции Ξ'_k в ходе её отсечения и растеризации вычисляется множество пикселей $P_k \subset \{(i, j) : 1 \le i \le h, 1 \le j \le w\}$.
5. Для каждого из двойных индексов $(i, j) \in P_k$ выполняются следующие действия:
   5а. Для пикселя с индексами (i, j) исходя из заданной картинной плоскости, экранной системы координат и отсекающего окна восстанавливаются координаты (x_0, y_0, z_0) в системе координат рассматриваемой сцены.

5б. В зависимости от способа задания исходной поверхности Ξ_k составляются и решаются относительно t уравнения (1.46) либо (1.47).

5в. Среди найденного множества решений выбирается наибольшее значение $t_{\max}$. Глубину соответствующей точки можно положить равной $-t_{\max}$.

5г. Если $-t_{\max} < z_{ij}$, то необходимо обновление обоих буферов: $z_{ij} := -t_{\max}$, $f_{ij} := \vec{c}$, где $\vec{c}$ – цвет точки, соответствующий параметру $t_{\max}$.

6. Если $k = n$, то алгоритм возвращает полученный буфер кадра F, который может быть немедленно выведен на экран. В противном случае происходит переход к следующей поверхности: $k := k + 1$, возврат к шагу 3.

Несложно видеть, что представленный алгоритм работает за линейное время относительно числа поверхностей и разрешения экрана.

Основным достоинством данного алгоритма является его простота. Он применим вне зависимости от сложности сцены или представленных в ней поверхностей. Кроме того, как уже было отмечено, поверхности можно обрабатывать в произвольном порядке, что избавляет нас от необходимости их предварительной сортировки по глубине, которая бы заняла время $\Theta(n \log n)$. Недостатками же этого алгоритма являются:

- Большой объём занимаемой памяти. Например, для экрана размерностью 1920 × 1080 с глубиной цвета в 24 бит необходимо выделить примерно 6 Мбайт на буфер кадра и около 16 Мбайт на буфер глубины (если для каждого его элемента выделить по 8 байт). На сегодняшний день, однако, это не представляет настолько большой проблемы, так как благодаря аппаратной реализации алгоритма практически на всех видеокартах, а также ёмкости видеопамяти, которая теперь исчисляется гигабайтами, этот объём данных может обрабатываться за ограниченное время.

- Результат в виде изображения с «лесенками», т.е. ступенчатыми растрированными линиями и границами поверхностей. Для преодоления этого эффекта при помощи растеризации со сглаживанием, или антиалиасингом (см. подраздел 1.3.5), важно понимать, на фоне которого из объектов рисуется текущий. В случае рисования поверхностей в произвольном порядке выяснить это бывает затруднительно.

- Сложность реализации полупрозрачности в силу произвольного порядка отображения поверхностей, ведь для видимости полупрозрачной поверхности она должна рисоваться после того, как изображены непрозрачные объекты позади неё.

- Сложность растеризации проекций криволинейных поверхностей.

Основными методами, в некоторой степени устраняющими эти недостатки, является использование сокращённого буфера глубины, содержащего одну или несколько строк, для построчного сканированию сцены;

а также отказ от точного вычисления множества пикселей для каждой проекции Ξ'_k [3, с. 229–233]. Последнее может быть осуществлено одной из двух модификаций представленного выше алгоритма:

- Вместо двойных индексов P_k на шаге 4 вычисляется габаритный прямоугольник $\{(i,j): i_{\min} \le i \le i_{\max}, j_{\min} \le j \le j_{\max}\}$, ограничивающий растрированную проекцию Ξ'_k. При этом на шаге 5б возможна ситуация, когда для очередного пикселя уравнения (1.46) и (1.47) могут не иметь решения. Тогда шаги 5в и 5г просто пропускаются, и происходит переход к другому пикселю. При этом необходимость в z-буфере всё ещё остаётся, так как габаритные прямоугольники для проекций разных поверхностей могут пересекаться.

- Можно вовсе отказаться от растеризации или её приближения для каждой поверхности, рассматривая вместо множеств P_k объединённый габаритный прямоугольник (или даже весь экран). Тогда уже фактически имеет место **метод трассировки лучей**, который сводится к проведению прямых (от картинной плоскости к наблюдателю, как на рисунке 1.50) или обратных (от наблюдателя к картинной плоскости) лучей и вычислению их пересечений с объектами сцены. Несложно видеть, что при такой организации алгоритма можно перейти от перебора поверхностей к перебору пикселей. После шага 1 выполняются шаги 5а-5г для каждого пикселя экрана, где вычисляются пересечения прямой (1.45) со всеми поверхностями и выбирается ровно одна точка с наименьшей глубиной (при её наличии). **Это даёт возможность не использовать z-буфер**. Однако для экономии вычислений рекомендуется по возможности проводить предварительную сортировку исходных поверхностей по глубине (особенно если имеют место непересекающиеся поверхности),

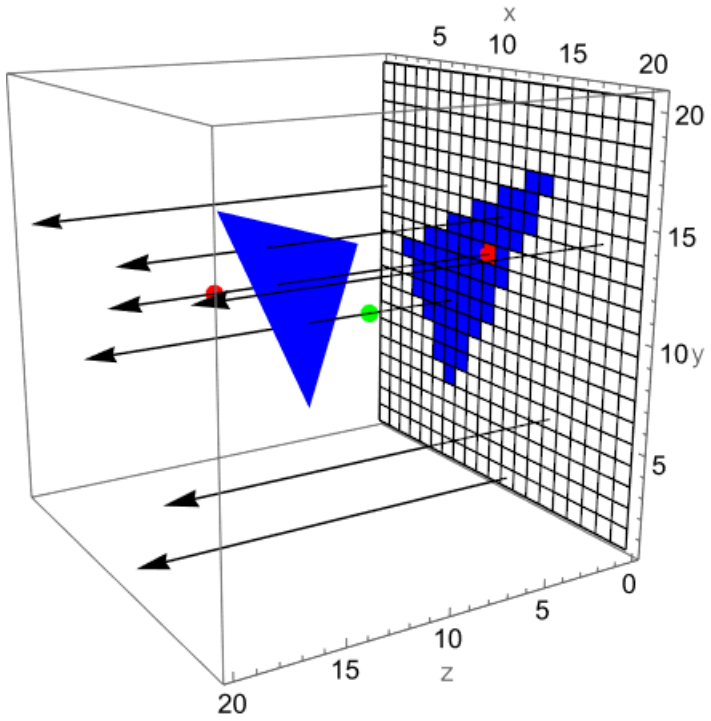


Рисунок 1.50 – Трассировка лучей в алгоритме z-буфера

благодаря чему происходит отрисовка пикселя после вычисления первого же пересечения соответствующей прямой с рассматриваемыми поверхностями.

1.2.6. Алгоритм Варнока

Ещё один алгоритм, удаляющий невидимые линии и поверхности, предложил Дж. Э. Уорнок в 1969 году [21]. Основная его идея заключается в том, что при наблюдении некоторой сцены человек не заостряет внимание на крупные участки, свободные от объектов. Напротив, представляют интерес мелкие участки сцены, изобилующие множеством различных графических объектов, например разбросанные на столе карандаши, ваза с цветами на столе, птицы в небе и т.д. Рассматривая стол, человек замечает карандаши, затем рассматривает, как они лежат относительно друг друга, их цвет, длину, насколько хорошо они заточены, какая фирма-производитель указана на них и прочие свойства. Таким образом, происходит перенос внимания со всей сцены на всё меньшие и меньшие участки. По сути, имеет место деление всей сцены на однородные участки разных размеров, например, поверхность стола и безоблачное небо можно считать визуально однородными. Такое свойство графических объектов получило название **когерентности**. В компьютерной графике под когерентностью часто понимается схожесть или близость характеристик соседних пикселей: принадлежность к некоторой области, цвет и другие.

В алгоритме Варнока, предназначенном для визуализации сцены с некоторым множеством плоских полигонов, происходит деление картинной плоскости на несколько окон, после чего для каждого окна определяется, является ли однородной ограниченная им часть проекции сцены. В случае, если ограниченная текущим окном часть сцены признаётся однородной, а также если окно имеет минимальный возможный размер (например, один пиксель), происходит отрисовка части сцены в пределах этого окна, и оно больше не рассматривается. В противном случае окно делится на подокна, и каждое из них рассматривается по тому же алгоритму. В зависимости от решаемых задач могут использоваться разные критерии однородности содержимого окон, чем обусловлено многообразие реализаций алгоритма Варнока [5, с. 290–315].

Прежде чем приступить к описанию этих критериев, рассмотрим разные случаи положения полигона относительно регулярного окна (Рисунок 1.51):

- **внешний полигон** расположен полностью снаружи окна,
- **пересекающий полигон** имеет общую область с окном,
- **внутренний полигон** расположен полностью внутри окна,
- **охватывающий полигон** полностью содержит окно.

Эти понятия будем также применять к полигонам трёхмерной сцены, которые, вообще говоря, не лежат на картинной плоскости. При этом будем

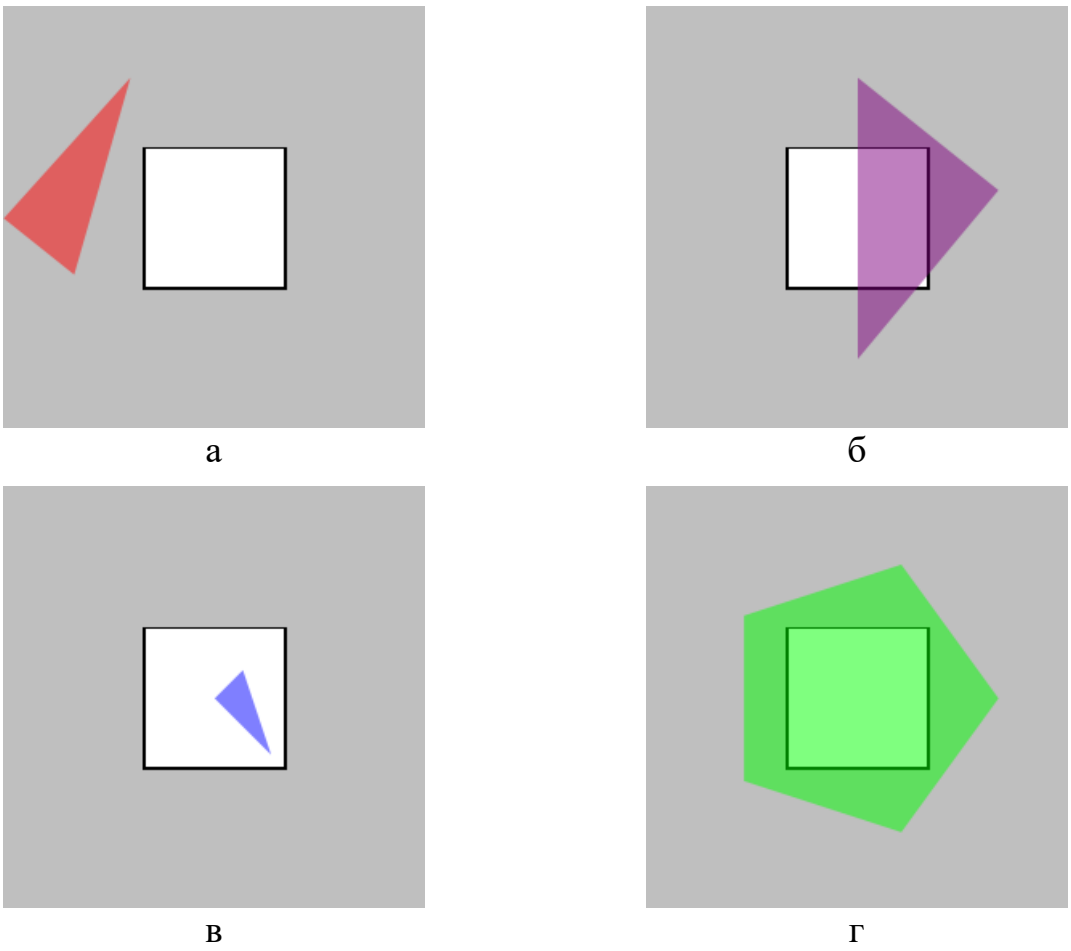


Рисунок 1.51 – Различные варианты расположения полигона относительно окна:
а – внешний полигон, б – пересекающий, в – внутренний, г – охватывающий

понимать, что те или иные условия выполняются по отношению к проекции полигона на картинную плоскость.

Для определения категории, к которой относится тот или иной полигон, в первую очередь используется его габаритный прямоугольник. Он, в свою очередь, также может быть внешним, пересекающим, внутренним или охватывающим. Пусть окно ограничено четырьмя прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а габаритный прямоугольник – прямыми $x = x'_{\min}$, $x = x'_{\max}$, $y = y'_{\min}$, $y = y'_{\max}$. Тогда могут иметь место следующие случаи:

- Если габаритный прямоугольник является внутренним по отношению к окну, что эквивалентно системе неравенств

$$\begin{cases} x_{\min} \le x'_{\min} < x'_{\max} \le x_{\max}, \\ y_{\min} \le y'_{\min} < y'_{\max} \le y_{\max}, \end{cases} \tag{1.49}$$

то и сам полигон также является внутренним (Рисунок 1.52а). Однако если габаритный прямоугольник и сам полигон совпадают с окном, то его лучше считать охватывающим.

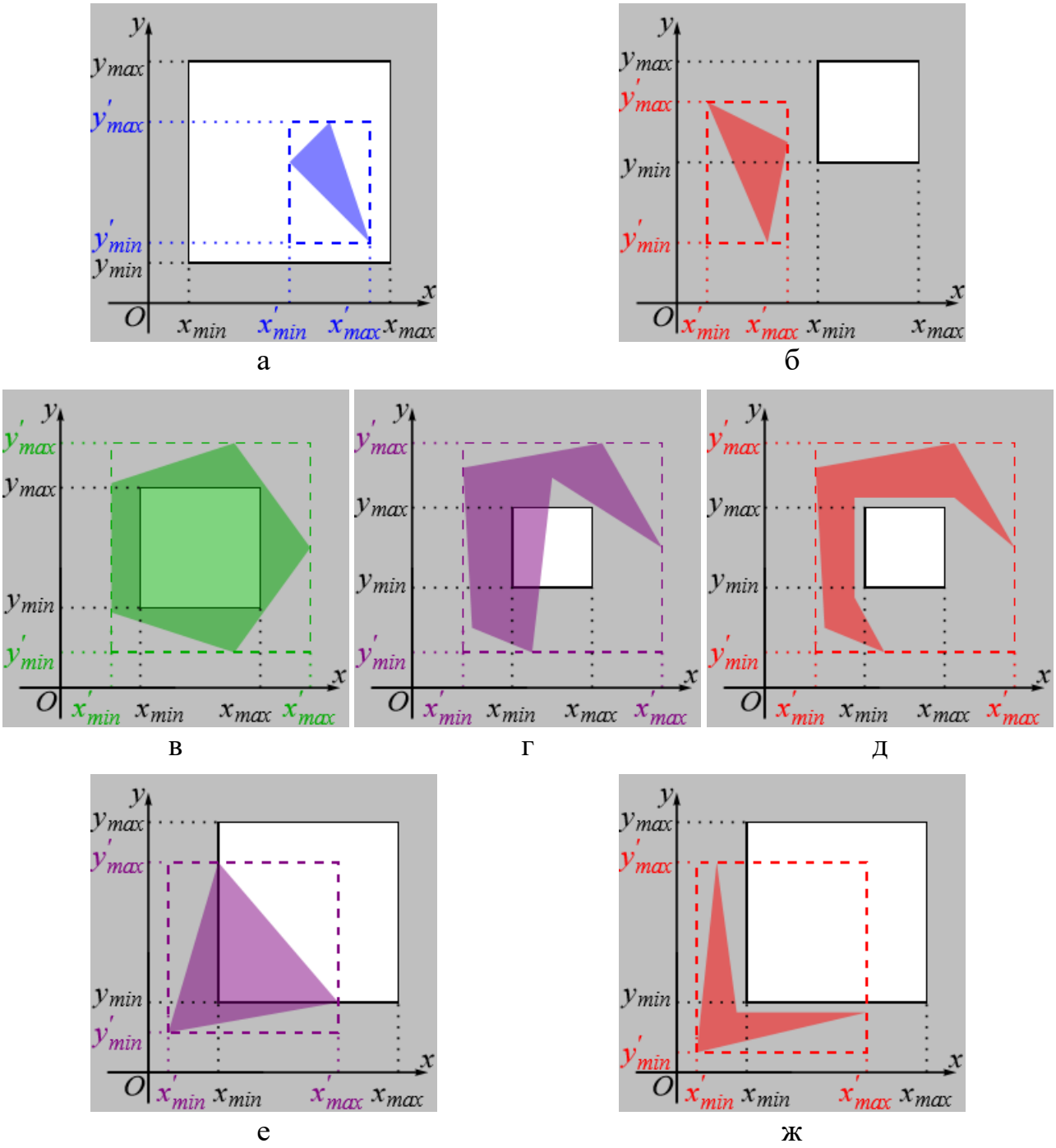


Рисунок 1.52 – Варианты расположения полигона и его габаритного прямоугольника относительно окна

- Если габаритный прямоугольник является внешним, для чего необходимо и достаточно выполнение хотя бы одного из следующих неравенств:

$$\begin{cases} x'_{\min} > x_{\max}, \\ x'_{\max} < x_{\min}, \\ y'_{\min} > y_{\max}, \\ y'_{\max} < y_{\min}, \end{cases} \qquad (1.50)$$

то исходный полигон также является внешним (Рисунок 1.52б).

- Если габаритный прямоугольник является охватывающим, что равносильно выполнению следующей системы:

$$\begin{cases} x'_{\min} \le x_{\min} < x_{\max} \le x'_{\max}, \\ y'_{\min} \le y_{\min} < y_{\max} \le y'_{\max}, \end{cases} \tag{1.51}$$

то полигон может оказаться охватывающим (Рисунок 1.52в), пересекающим (Рисунок 1.52г) либо внешним (Рисунок 1.52д).

- Наконец, если не выполняются ни системы (1.49) и (1.51), ни какое-либо из неравенств (1.50), то габаритный прямоугольник является пересекающим, а исходный полигон может быть либо пересекающим (Рисунок 1.52е), либо внешним (Рисунок 1.52ж).

Таким образом, если выполняется система (1.49) либо одно из неравенств (1.50), то можно сразу определить категорию, к которой относится данный полигон, что значительно сокращает вычисления. В противном случае для её определения приходится прибегать к более сложным алгоритмам. Так, может быть применён алгоритм Сазерленда-Ходжмана для отсечения полигона окном (причём необходима реализация, которая даёт полигоны без накладывания рёбер, например, через редактирование рёбер, см. алгоритм на с. 40–42). В зависимости от результата отсечения можно сделать следующие выводы:

- если полигон подлежит полному отсечению, то он является внешним;
- если результат отсечения совпадает с окном, то это охватывающий полигон;
- в противном случае полигон является пересекающим.

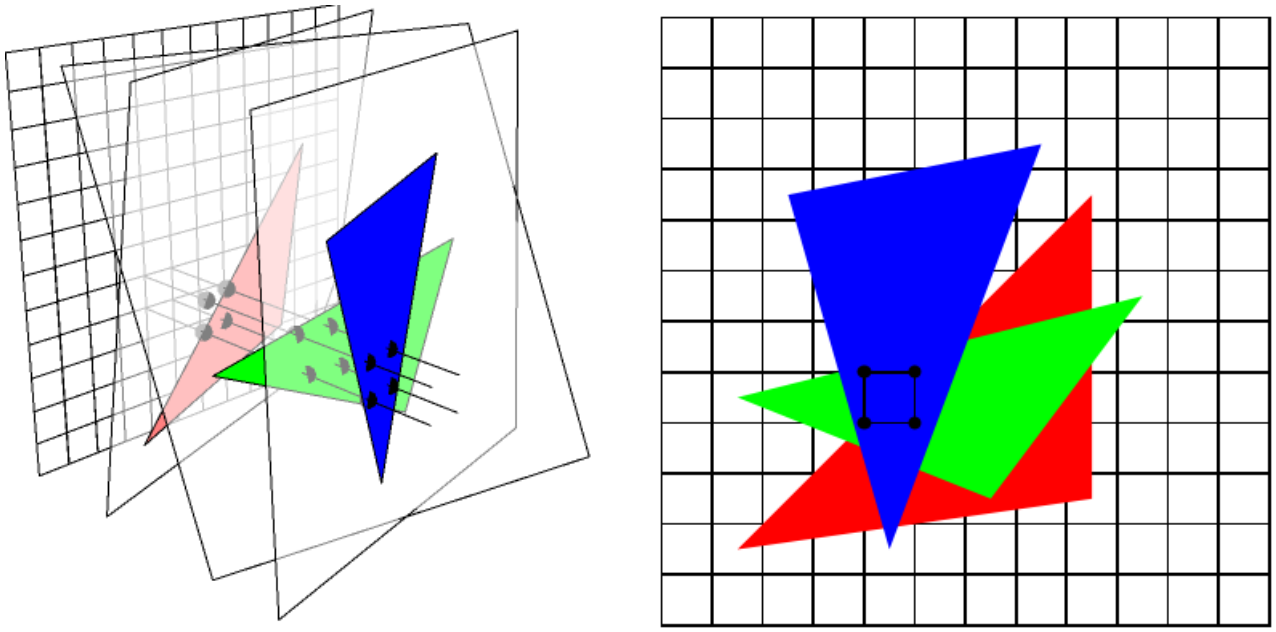
Кроме того, после деления окна в некоторых ситуациях для получившихся мелких окон можно не выполнять дальнейших вычислений для определения категорий некоторых полигонов. Так, ясно, что если полигон является внешним для текущего окна, то он же будет внешним и для всех подокон. Соответственно, его можно просто отбросить. Аналогично, полигон, охватывающий некоторое окно, является охватывающим и для всех его подокон. Что касается пересекающего полигона, то после деления окна при рассмотрении подокон можно вместо этого полигона рассматривать его части, полученные в результате отсечения исходным окном. При этом по отношению к подокнам эти части, будучи внутренними полигонами исходного окна, могут оказаться в любой категории.

Удаление невидимых линий. Оригинальный алгоритм [21] позволяет осуществлять рисование видимых линий путём выявления окон наименьшего размера, не отвечающих определённым условиям однородности. Уорнок рассматривал два таких условия:

- окно не содержит ни одного полигона, т.е. все полигоны по отношению к нему являются внешними;
- для окна найдётся охватывающий полигон, экранирующий все остальные полигоны, не являющиеся внешними для этого же окна.

Если для окна минимального размера эти оба условия нарушаются, то оно должно быть помещено в список, называемый **дисплейным файлом**. После выполнения алгоритма все окна из дисплейного файла подлежат закраске в некоторый цвет, и получаются растрированные проекции видимых границ полигонов, а также их пересечений.

После определения категорий всех полигонов для текущего окна легко проверить выполнение первого условия. При проверке второго условия используются глубины точек – пересечений прямых, проведённых от углов окна к наблюдателю, с плоскостями, на которых лежат нужные полигоны (Рисунок 1.53а). Если для каждой из четырёх прямых получилось по точке с наименьшими глубинами, лежащими на плоскости одного и того же охватывающего полигона, то в пределах окна видимой является часть проекции только одного этого полигона (Рисунок 1.53б). Для вычисления глубин пересечений четырёх прямых, которые описываются системами уравнений (1.45), с плоскостями, описываемыми общими уравнениями, может использоваться формула (1.48), которая даёт величину, противоположную искомой глубине. Таким образом, для каждого полигона необходимо также знать уравнение, описывающее плоскость, в которой он лежит.



а б

Рисунок 1.53 – Обработка окна при наличии охватывающего полигона, экранирующего все плоскости в его пределах

Таким образом, реализация **алгоритма Варнока, удаляющая невидимые линии** самонепересекающихся полигонов $P_i = A_{i1}A_{i2}...A_{i,l_i}A_{i1}$, $i = \overline{1,N}$, при их выводе в окно B картинной плоскости, описываемое экранными координатами $\{(x,y): x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}\}$, выглядит следующим образом:

1. Выбирается максимальное число делений окна $n \in \mathbb{N}$.

2. Для каждого полигона P_i вычисляется проекция P_i' на картинную плоскость. При ортографическом проектировании на картинную плоскость Oxy это выполняется элементарным отбрасыванием аппликаты каждой вершины A_{ij}.
3. Для каждого полигона P_i вычисляется габаритный прямоугольник, ограничивающий проекцию P_i':

$$B_i = \{(x,y) : x'_{i\min} \le x \le x'_{i\max}, y'_{i\min} \le y \le y'_{i\max}\},$$
$$x'_{i\min} = \min_{j=1,N_i} x'_{ij},\ \ x'_{i\max} = \max_{j=1,N_i} x'_{ij},\ \ y'_{i\min} = \min_{j=1,N_i} y'_{ij},\ \ y'_{i\max} = \max_{j=1,N_i} y'_{ij},$$

где (x'_{ij}, y'_{ij}) – координаты проекции вершины A_{ij} в экранной системе координат.
4. Для каждого полигона вычисляется уравнение плоскости $P_i \subset \Pi_i$: $A_i x + B_i y + C_i z + D_i = 0$. При условии минимальности полигона P_i это осуществимо по следующим формулам:

$$\vec{n}_i(A_i, B_i, C_i) = \overrightarrow{A_{i1}A_{i2}} \times \overrightarrow{A_{i2}A_{i3}},$$
$$D_i = -\vec{n}_i \cdot \vec{r}_{i1},$$

где $\vec{r}_{i1}$ – радиус-вектор точки A_{i1}. При направлении на дальнего наблюдателя $\vec{s}(0,0,1)$ целесообразно хранить коэффициенты для формулы $z = \dfrac{-A_i}{C_i}x + \dfrac{-B_i}{C_i}y + \dfrac{-D_i}{C_i}$, по которой при $C_i \neq 0$ легко найти глубину точки, проектирующейся в заданный пиксель картинной плоскости.
5. Списки проекций P_i', габаритных прямоугольников B_i и плоскостей Π_i используются для составления нового списка полигонов $\mathcal{P} := \varnothing$, подлежащего при необходимости передаче дочерним окнам. Для этого определяется, к какой категории относится каждый из полигонов P_i:
   5а. Вначале проверяется первый полигон: $i := 1$.
   5б. Если полигон P_i помечен как охватывающий (в результате применения данного алгоритма к предыдущему, родительскому окну), то он помещается в список $\mathcal{P}$. При этом он остаётся помеченным. Переход к шагу 5и.
   5в. Если выполняется система неравенств (1.49), то проверяется, совпадает ли проекция P_i' с текущим окном B. Если да, то P_i помечается и помещается в $\mathcal{P}$. В противном случае он помещается туда без пометки. Переход к шагу 5и.

5г. Если выполняется по крайней мере одно из неравенств (1.50), то полигон P_i является внешним, и просто пропускается. Переход к шагу 5и.

5д. К проекции P_i' и отсекателю B применяется алгоритм Сазерленда-Ходжмана, возвращающий, вообще говоря, множество $\mathcal{P}_i' = \{Q_{i1}', Q_{i2}', ..., Q_{i,l_i'}'\}$ самонепересекающихся полигонов – частей полигона P_i', которые являются внутренними для окна B.

5е. Если $l_i' = 0$, то полигон P_i является внешним – список $\mathcal{P}$ не подвергается редактированию. Переход к шагу 5и.

5ж. Если $l_i' = 1$, и единственный полигон из множества $\mathcal{P}_i'$ совпадает с окном B, то полигон P_i является охватывающим, помечается и добавляется в $\mathcal{P}$. Переход к шагу 5и.

5з. Полигон P_i является пересекающим. В список $\mathcal{P}$ помещаются все полигоны Q_{ij}, $j = \overline{1, l_i'}$, координаты вершин которых восстанавливаются исходя из уравнения плоскости Π_i и вершин полигонов Q_{ij}'. Новым полигонам ставятся в соответствие их проекции Q_{ij}', габаритные прямоугольники, вычисляемые согласно шагу 2, и одна и та же плоскость Π_i, в которой они все лежат.

5и. Если $i < N$, то происходит переход к следующему полигону: $i := i + 1$, возврат к шагу 5б.

6. Если $|\mathcal{P}| = 0$, то все полигоны оказались внешними, и алгоритм завершает свою работу, возвращая пустой дисплейный файл.

7. Если в списке $\mathcal{P}$ есть по крайней мере один помеченный полигон, то для каждого из углов окна B проводятся прямые (1.45), и для каждой из этих прямых вычисляется, пересечение с какой плоскостью имеет наименьшую глубину. При ортографическом проектировании на плоскость Oxy справедливы следующие формулы для вычисления параметра точки с наименьшей глубиной и нахождения соответствующей плоскости:

$$t(x,y) = \max_{i:P_i \in \mathcal{P} \vee Q_{ij} \in \mathcal{P}} \frac{A_i x + B_i y + D_i}{C_i},$$
$$i(x,y) = \arg\max_{i:P_i \in \mathcal{P} \vee Q_{ij} \in \mathcal{P}} \frac{A_i x + B_i y + D_i}{C_i}, \quad (1.52)$$

которые получаются при подстановке в (1.48) значений $(x_0, y_0, z_0) = (x, y, 0)$, $(s_x, s_y, s_z) = (0,0,1)$, $h_s = 0$.

8. Если $i(x_{\min}, y_{\min}) = i(x_{\min}, y_{\max}) = i(x_{\max}, y_{\min}) = i(x_{\max}, y_{\max}) = I$, и полигон P_I является помеченным, то тогда имеем охватывающий полигон, экранирующий все остальные, и алгоритм завершается с возвращением пустого дисплейного файла.
9. Если $n = 0$, то алгоритм возвращает дисплейный файл с одним окном B.
10. Окно B делится на четыре дочерних окна прямыми $x = \frac{x_{\min} + x_{\max}}{2}$ и $y = \frac{y_{\min} + y_{\max}}{2}$. Шаги 5–10 повторяются с множеством полигонов из $\mathcal{P}$ для каждого из новых окон, при этом нужно положить $n := n - 1$. В результате получаются четыре дисплейных файла, объединение которых даёт результат алгоритма.

Пример работы алгоритма представлен на рисунке 1.54. Алгоритм работает за время, которое в первую очередь оценивается исходя из количества рассматриваемых окон. Максимальное число окон не превышает $1 + 4^1 + ... + 4^n = \frac{4^{n+1} - 1}{3}$. Фактически число разбиений нет смысла выбирать больше, чем $n_{\max} = \log_2 L$, где L – число пикселей по ширине и высоте квадратного экрана. Значит, максимальное число окон может быть выражено через число пикселей экрана $p = L^2$: $\frac{4^{n+1} - 1}{3} \le \frac{4^{n_{\max}+1} - 1}{3} = \frac{4 \cdot L^2 - 1}{3} = O(p)$. Кроме того, время выполнения шагов для каждого конкретного окна можно оценить как $O(V)$, где $V = \sum_{i=1}^{N} l_i$ – суммарное число вершин всех полигонов, так как в худшем случае к ним всем приходится применять алгоритм Сазерленда-Ходжмана, работающий за линейное время от числа вершин отсекаемого полигона (при постоянном числе вершин отсекателя).

Замечание 1. Данная реализация алгоритма Варнока не работает корректно в случае, когда плоскость некоторого полигона является параллельной направлению на наблюдателя, т.е. когда ближний наблюдатель лежит на этой плоскости, либо когда направление на дальнего наблюдателя выражается вектором, ортогональным нормальному вектору плоскости. Для этих плоскостей вместо формул (1.52) следует проверять условие $A_i x + B_i y + D_i = 0$. Если оно не выполняется, то такая плоскость не пересекается с прямой (1.45), проведённой через рассматриваемый угол окна. Если же условие выполнено, то следует вычислять наименьшую глубину точек – пересечений прямой с самим полигоном. Кроме того, такие полигоны могут вообще не рисоваться.

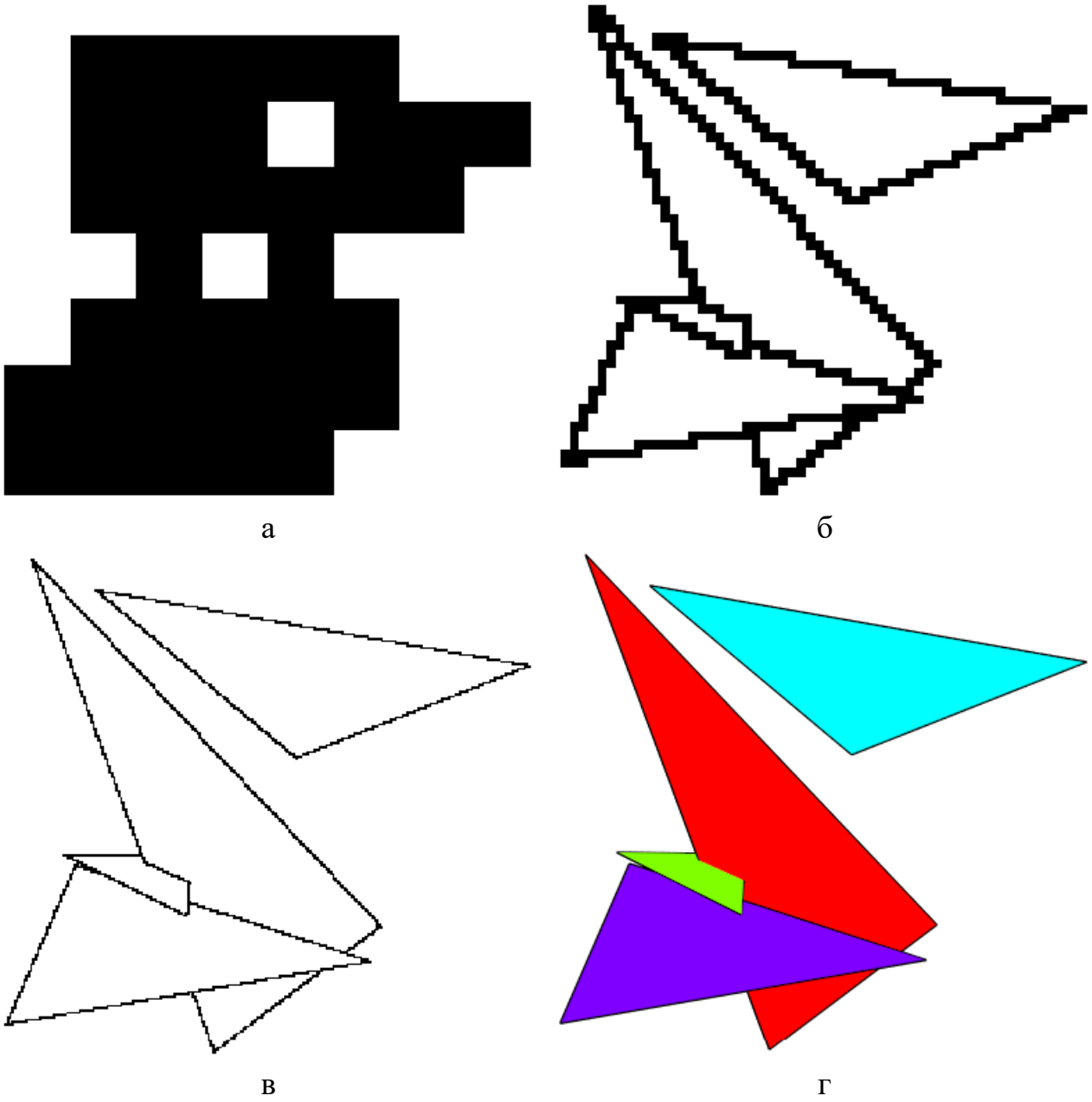
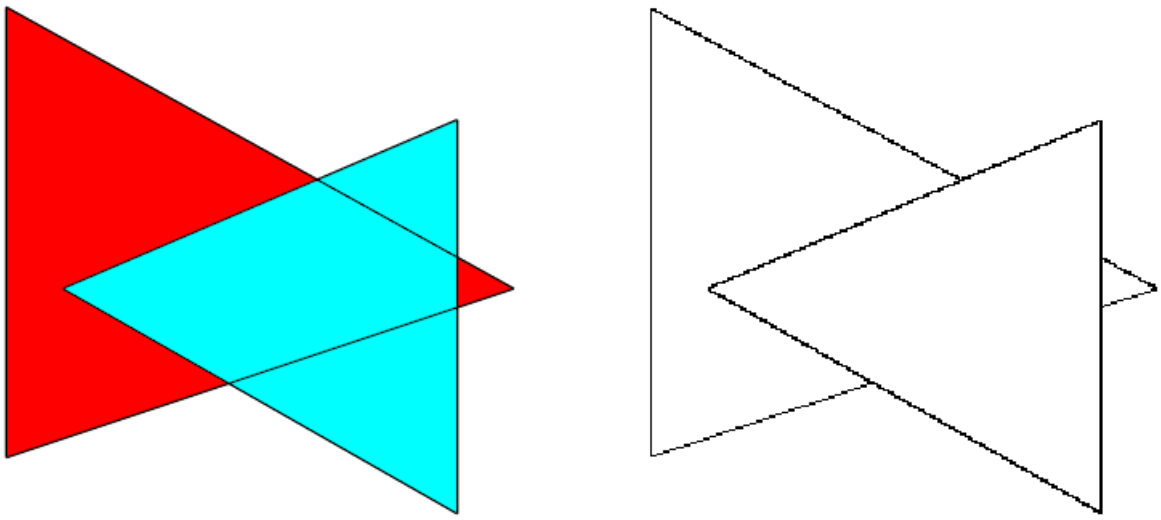


Рисунок 1.54 – Результаты реализации алгоритма Варнока, рисующей видимые линии: а – при n = 3 разбиениях окна, б – при n = 6, в – при n = 8, г – проекция исходной сцены

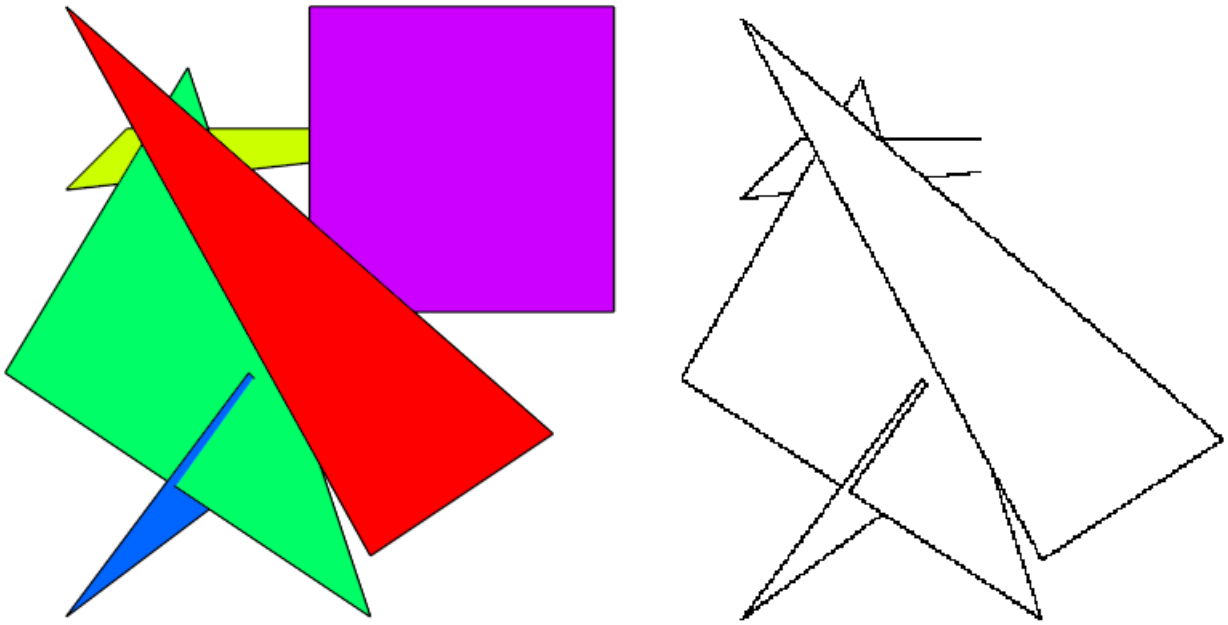
Замечание 2. В ситуации, когда имеют место два пересекающихся полигона, лежащих на одной плоскости, экранирующим является тот из них, который идёт позже в списке (Рисунок 1.55).



а б

Рисунок 1.55 – Удаление линий при рисовании полигонов на одной плоскости
(а – проекция исходной сцены, б – удаление невидимых линий алгоритмом Варнока)

Замечание 3. Данная реализация также некорректно работает при попадании рёбер на границы окон. Например, на рисунке 1.56а изображена сцена с несколькими полигонами, при попытке изображения которой при помощи алгоритма Варнока сиреневый квадрат оказался совпадающим с верхним правым окном, полученным в результате деления исходного окна картинной плоскости. Это верхнее правое окно в последствии разделилось ещё на четыре окна, три из которых согласно шагу 8 дали пустые дисплейные файлы. Таким образом, границы этого квадрата не оказались в итоговом



а б

Рисунок 1.56 – Рисование сцены с полигоном, ребра проекции которого совпали с границами одного из окон

дисплейном файле (Рисунок 1.56б). Решается эта проблема посредством достаточно мелких сдвигов вершин таких полигонов или вершин окна.

Удаление невидимых поверхностей. Удаление невидимых и визуализация видимых поверхностей может быть реализована алгоритмом, который в целом повторяет шаги вышеуказанной реализации алгоритма Варнока. Здесь используются некоторые дополнительные условия однородности окна:

- Если для текущего окна есть ровно один внутренний полигон, а остальные являются внешними, то происходит отрисовка этого окна с внутренним полигоном.

- Если для текущего окна есть ровно один пересекающий полигон (а остальные – внешние), то происходит отсечение полигона, а затем – отрисовка окна с найденным содержимым.

- Если для текущего окна есть ровно один охватывающий полигон (а остальные – внешние), то всё окно окрашивается в цвет этого полигона.

В первых двух случаях могут использоваться различные алгоритмы растеризации видимой в пределах окна части рассматриваемого полигона. Например, имеет место следующий **алгоритм визуализации одного полигона** в пределах заданного окна:

1. Выбирается максимальное число делений окна $n \in \mathbb{N}$.

2. Если $n = 0$, то алгоритм завершается с возвратом дисплейного файла, содержащим только одно текущее окно.

3. Если полигон совпадает с окном, то алгоритм завершается с возвратом дисплейного файла, содержащим текущее окно.

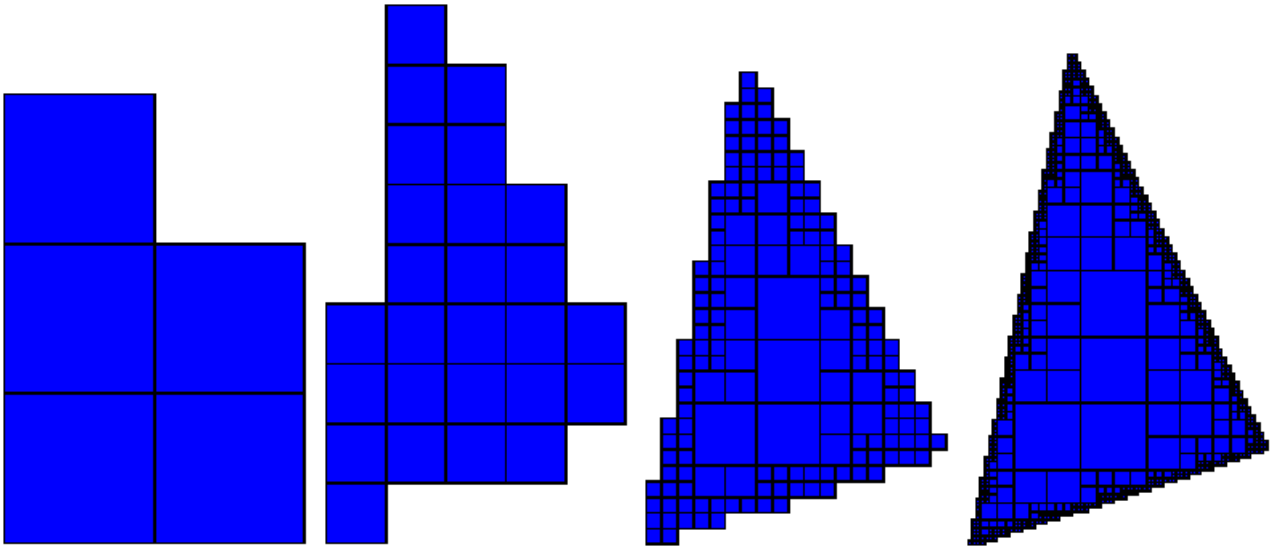
4. Во всех остальных случаях происходит деление окна на четыре части. Происходит отсечение полигона каждым из получившихся дочерних окон, в результате чего для каждого окна получается, вообще говоря, список новых полигонов.

5. Для каждого нового полигона и соответствующего дочернего окна повторяются шаги 2–5 при $n := n - 1$, в результате чего получается множество дисплейных файлов. Их объединение даёт искомый дисплейный файл.

Варианты растеризации при различном числе разбиений представлены на рисунке 1.57. Как и для представленной выше реализации алгоритма Варнока, асимптотика времени выполнения этого алгоритма является линейной относительно числа пикселей и вершин исходного полигона.

Кроме того, важным дополнением является необходимость отрисовки разных поверхностей разными цветами. Следовательно, каждому окну дисплейного файла должен быть поставлен в соответствие определённый цвет. Простейшим подходом является назначение каждому исходному полигону некоторого цвета, в которые затем будут окрашиваться соответствующие окна.

Таким образом, реализация **алгоритма Варнока, удаляющая невидимые поверхности** самонепересекающихся полигонов P_i, $i = \overline{1, N}$, аналогична представленной выше реализации, удаляющей невидимые линии, со следующими изменениями:



а б в г

Рисунок 1.57 – Растеризация треугольника при разном числе разбиений окна: $n = 2$ (а), $n = 4$ (б), $n = 6$ (в), $n = 8$ (г)

- Наряду с шагами 1–4 происходит также назначение каждому полигону своего цвета $\vec{c}_i$.
- Выполняются шаги 5а-5и, при этом при выполнении шага 5з каждому новому полигону Q_{ij} ставится в соответствие один и тот же цвет $\vec{c}_i$.
- После выполнения шага 6 проверяется, всем ли полигонам из списка $\mathcal{P}$ поставлен в соответствие один и тот же цвет. Если да, то проверяется также, есть ли в списке помеченные полигоны.
  - Если и это условие выполняется, то это означает, что имеет место охватывающий полигон. Он либо является целиком видимым, либо частично или полностью экранируется другими полигонами того же цвета. Как бы то ни было, всё окно должно быть окрашено одним цветом. Алгоритм завершается с возвратом дисплейного файла, содержащим текущее окно, которому соответствует цвет этого полигона.
  - В противном случае ко всем полигонам из $\mathcal{P}$ применяется приведённый выше алгоритм растеризации, возвращающий дисплейные файлы для каждого из них. Алгоритм завершается с возвратом объединённого дисплейного файла, где каждому окну поставлен в соответствие один и тот же цвет.
- Если $n = 0$, то по формулам (1.52) при $x = \frac{x_{\min} + x_{\max}}{2}$, $y = \frac{y_{\min} + y_{\max}}{2}$ проверяется, для какого из полигонов списка $\mathcal{P}$ точка его плоскости, проектируемая в середину окна, имеет наименьшую глубину. Алгоритм возвращает дисплейный файл с текущим окном, которое должно быть окрашено в цвет найденного полигона.

- Выполняются шаги 7 и 8. Если на шаге 8 необходимо завершить алгоритм, то возвращается дисплейный файл, содержащий текущее окно, окрашиваемое в цвет найденного охватывающего полигона.
- Окно делится на четыре дочерних окна. Для них повторяются описанные выше шаги (кроме самого первого и включая этот), после которых получаются четыре дисплейных файла. Их объединение является результатом данного алгоритма.

Примеры использования этой реализации представлены на рисунке 1.58.

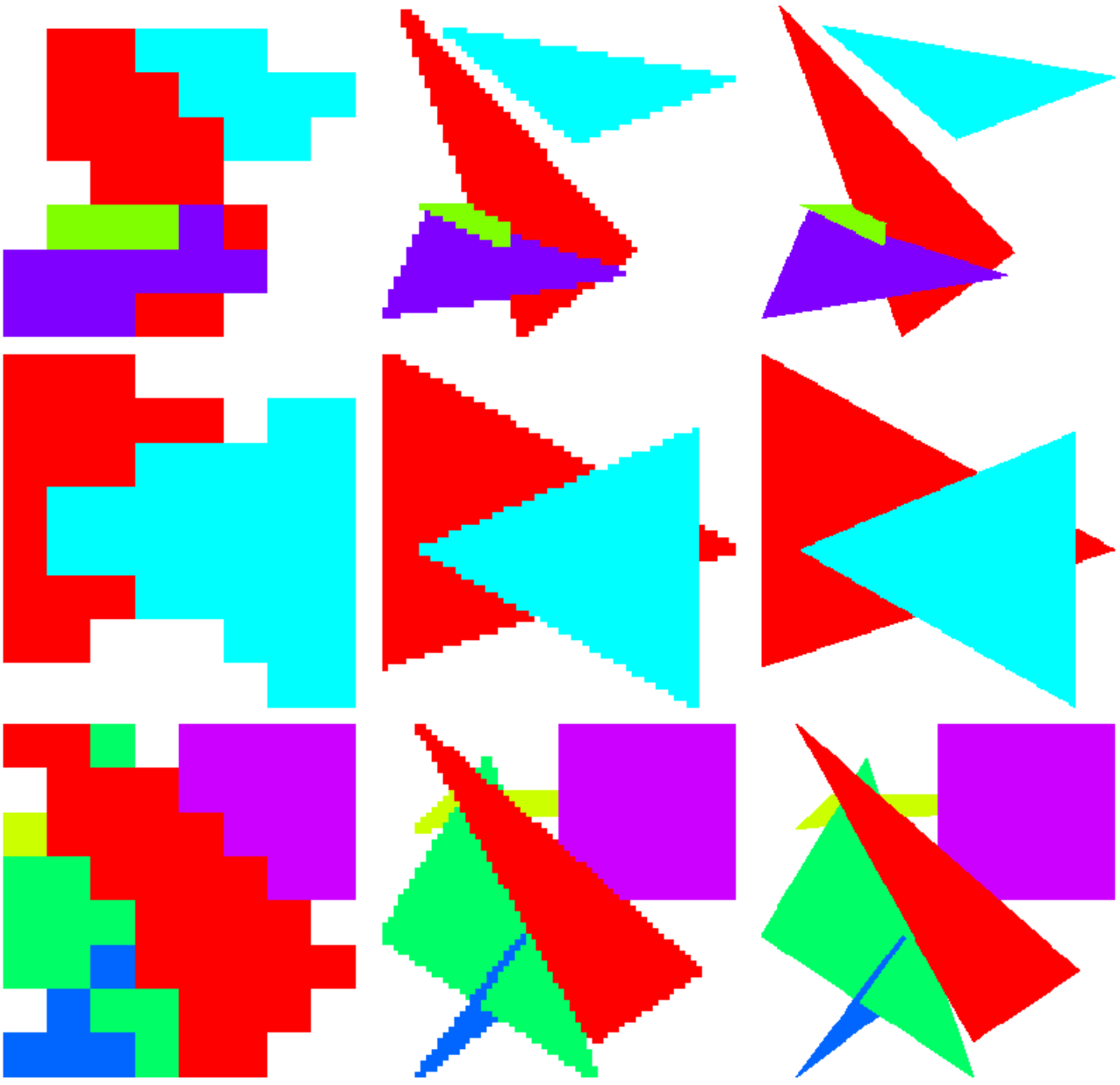


Рисунок 1.58 – Примеры отрисовки трёхмерных сцен реализацией алгоритма Варнока, удаляющей невидимые поверхности: n = 3 (а), n = 6 (а), n = 8 (в)

1.3. Базовые растровые алгоритмы

Моделирование сцены и расположенных в ней геометрических объектов предшествует формированию конечного изображения, которое затем должно быть выведено на экран или распечатано. На первом этапе, по сути, происходит работа с векторной графикой, на последнем – с растровой. Таким образом, одной из важных задач компьютерной графики является преобразование векторной графики в растровую. **Растеризацией** называется процесс нахождения множества пикселей, наилучшим образом приближающих некоторый геометрический объект.

Растром называется прямоугольная сетка точек, формирующих изображение на экране компьютера. Каждая точка растра (**пиксел**, **пиксель**) характеризуется двумя целочисленными координатами и цветом. При растеризации происходит дискретизация картинной плоскости на пиксели, после которой они окрашиваются в различные цвета в зависимости от положения и окраски объектов исходной картинной плоскости. Растровые алгоритмы предназначены для определения принадлежности пикселей к исходным геометрическим объектам.

1.3.1. Растеризация отрезка. Основные понятия и свойства

Одной из простейших задач растровой графики является рисование отрезка. Растровые алгоритмы рисования отрезка вычисляют координаты пикселей, наиболее близких к нему (Рисунок 1.59). Для определённости будем считать, что у отрезка AB точка $A(x_A, y_A)$ является начальной, точка $B(x_B, y_B)$ – конечной. Также будем говорить, что **отрезок *AB* принадлежит *I*-ому октантному углу**, $I = \overline{1,8}$, если вектор $\overrightarrow{AB}$ лежит в этом же октантном угле. Определить октант отрезка можно, например, сравнив числа $x_B - x_A$ и $y_B - y_A$ с нулём и друг с другом согласно схеме из первой части ЭУМК [1, рисунок 1.31].

Для объединения некоторой совокупности пикселей в единое множество используется понятие связности, позволяющее определять, какие пиксели являются соседними. Два пикселя с координатами (x_1, y_1), (x_2, y_2), $x_1, x_2, y_1, y_2 \in \mathbb{Z}$, называются **4-связными**, или **4-соседями**, если выполняется неравенство $|x_2 - x_1| + |y_2 - y_1| \le 1$, т.е. они соседствуют по горизонтали либо вертикали растровой сетки (Рисунок 1.60а). Два пикселя называются **8-связными**, или **8-соседями**, если выполняются два неравенства $|x_2 - x_1| \le 1$ и $|y_2 - y_1| \le 1$, т.е. они соседствуют либо по горизонтали, либо по вертикали, либо по диагонали растровой сетки (Рисунок 1.60б).

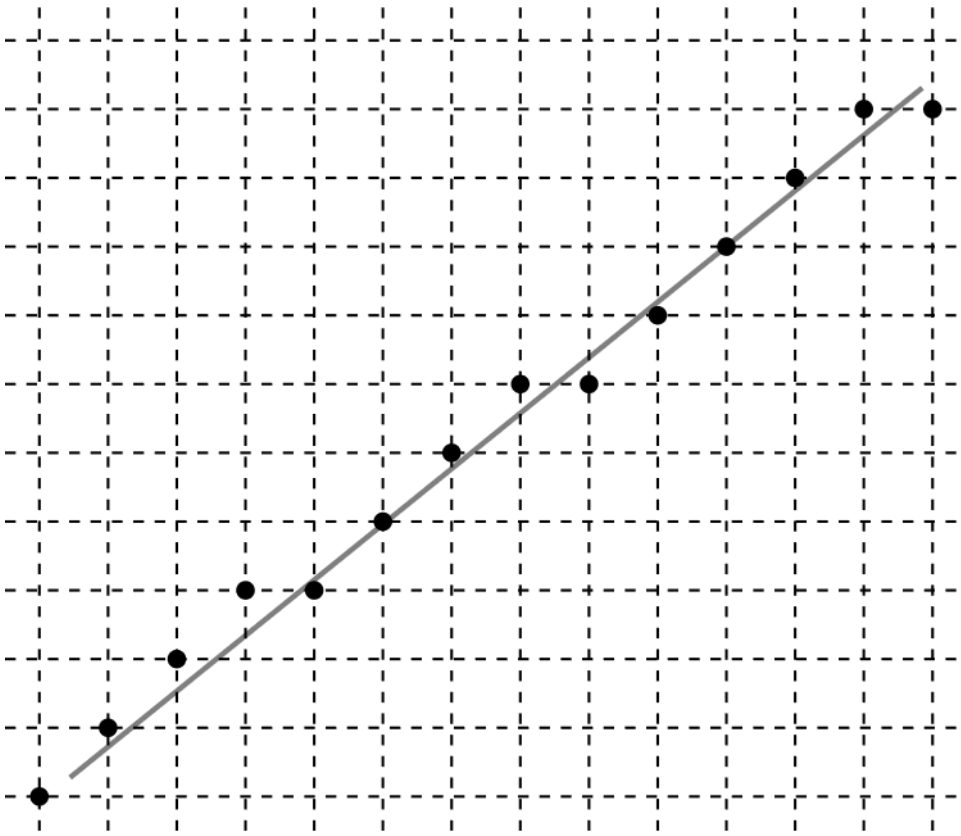


Рисунок 1.59 – Вариант растеризации отрезка. Чёрными точками обозначены пиксели, образующие искомый растрированный отрезок

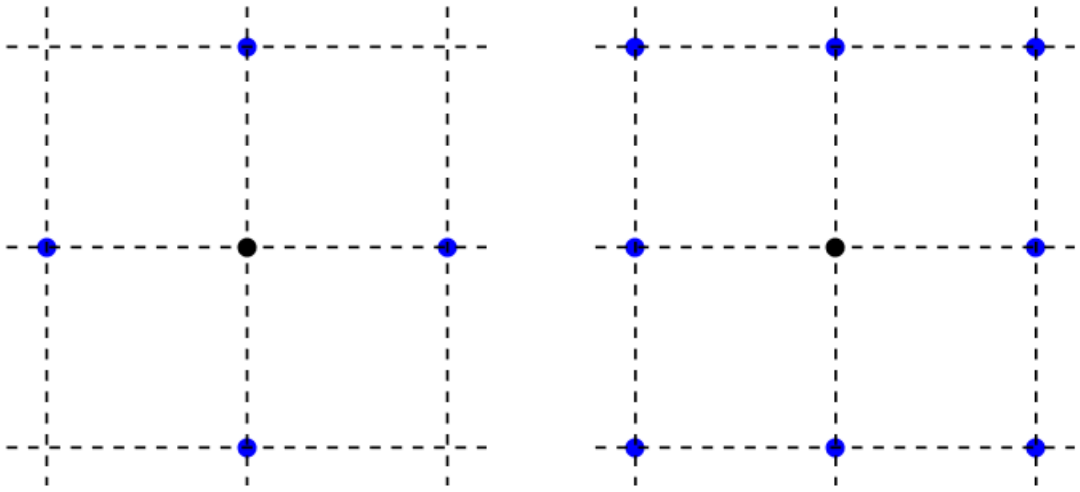


Рисунок 1.60 – Для произвольного пикселя есть всего четыре 4-соседа (а) и восемь 8-соседей (б)

Множество пикселей P называется **4-связным (8-связным)**, если для всяких двух пикселей $p, q \in P$ существует последовательность пикселей $p_i \in P$, $i = \overline{1,n}$, такая, что $p_1 = p$, $p_n = q$, и пиксели p_i и p_{i+1}, $i = \overline{1,n-1}$, являются 4-соседями (8-соседями) (Рисунок 1.61).

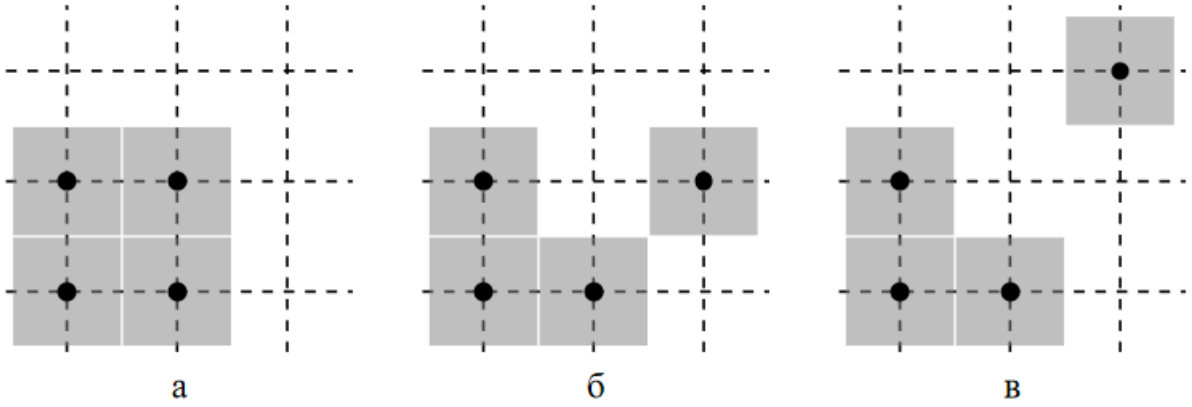


Рисунок 1.61 – 4-связная (а), 8-связная (б) и несвязная (в) области

Пусть у растрированного отрезка концы расположены в точках с целочисленными координатами $(x_{нач}, y_{нач})$ и $(x_{кон}, y_{кон})$. Тогда для построения 8-связного отрезка необходимо $L + 1$ пикселей, где L вычисляется по формуле

$$L = \max\left\{\left|x_{кон} - x_{нач}\right|, \left|y_{кон} - y_{нач}\right|\right\}. \qquad (1.53)$$

◆ Покажем, что потребуется не менее $\left|x_{кон} - x_{нач}\right| + 1$ пикселя. Предположим противное, т.е. растрированный отрезок содержит не более $\left|x_{кон} - x_{нач}\right|$ пикселей. Если положить $x_{нач} < x_{кон}$, то ясно, что горизонтальная координата каждого из пикселей отрезка находится во множестве $\{x_{нач}, x_{нач} + 1, \ldots, x_{кон} - 1, x_{кон}\}$ мощности $\left|x_{кон} - x_{нач}\right| + 1$. Значит, найдётся некоторое значение x' из этого множества, которое не является горизонтальной координатой ни для какого из пикселей отрезка (Рисунок 1.62).

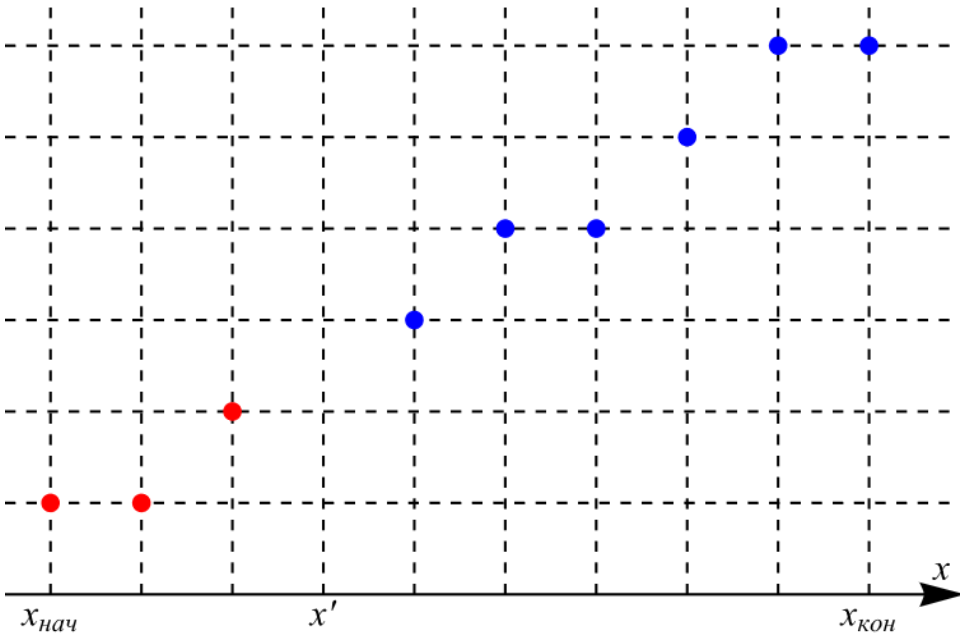


Рисунок 1.62 – При недостаточном количестве пикселей отрезок становится несвязным

Соответственно, у всех пикселей координата x удовлетворяет одному из неравенств:

$$\begin{aligned} x &\le x' - 1, \\ x &\ge x' + 1. \end{aligned} \tag{1.54}$$

При этом каждому из этих неравенств удовлетворяет хотя бы по одному пикселю: первому – начальный, второму – конечный. Таким образом, все пиксели делятся на два непустых подмножества P_1 и P_2 в зависимости от того, какому именно из неравенств (1.54) удовлетворяют их координаты (на рисунке 1.62 пиксели из двух множеств отмечены красным и синим цветом).

Теперь, если взять два произвольных пикселя $(x_1, y_1) \in P_1$ и $(x_2, y_2) \in P_2$, имеем следующее: $x_2 - x_1 \ge x' + 1 - (x' - 1) = 2$, откуда следует невыполнение неравенства $|x_2 - x_1| \le 1$, а значит, и то, что эти два пикселя не являются 8-связными. Таким образом, объединение двух множеств P_1 и P_2 не является 8-связным – противоречие тому, что исходный растрированный отрезок 8-связен.

Аналогично доказывается, что нужно не менее $|y_{\text{кон}} - y_{\text{нач}}| + 1$, откуда следует, что необходимо по крайней мере $L + 1$ пикселей. ■

1.3.2. Алгоритм DDA-линии

Алгоритм DDA-линии[1] состоит в следующем:

1. Координаты конечных точек округляются: $x_{\text{нач}} = \text{round}(x_A)$, $y_{\text{нач}} = \text{round}(y_A)$, $x_{\text{кон}} = \text{round}(x_B)$, $y_{\text{кон}} = \text{round}(y_B)$.
2. Вычисляется количество шагов L по формуле (1.53).
3. Случай $L = 0$ возникает при вырождении растрированного отрезка, т.е. когда его начало и конец совпадают. В этой ситуации алгоритм завершает свою работу, возвращая единственную точку $(x_{\text{нач}}, y_{\text{нач}})$.
4. Пусть теперь $L > 0$. На первом шаге вспомогательным переменным $x, y \in \mathbb{R}$ присваиваются значения $x := x_A$, $y := y_A$.
5. На очередном, l-ом шаге, $l = \overline{2, L+1}$, значения этих переменных увеличиваются:

$$x := x + \frac{x_B - x_A}{L}, y := y + \frac{y_B - y_A}{L}. \tag{1.55}$$

[1] Аббревиатура DDA означает digital differential analyzer – цифровой дифференциальный анализатор, т.е. устройство или программа, используемая для генерации отрезков и векторов при помощи приближённых значений, расположенных на некотором числовом интервале

Полученные значения округляются без их присваивания переменным x и y, и получаются координаты l-ого пикселя, принадлежащего растрированному отрезку.

Таким образом, результатом алгоритма DDA-линии является множество из $L + 1$ пикселя, при этом l-ый пиксель имеет координаты

$$(x_l, y_l) = \left(\text{round}\left(x_A + (l-1)\frac{x_B - x_A}{L} \right), \text{round}\left(y_A + (l-1)\frac{y_B - y_A}{L} \right) \right). \qquad (1.56)$$

Заметим, что по формуле (1.56) начальный и конечный пиксель искомого отрезка имеют координаты $(x_{\text{нач}}, y_{\text{нач}})$ и $(x_{\text{кон}}, y_{\text{кон}})$, ранее полученные на первом шаге алгоритма. В ходе работы алгоритма, по сути, происходит разбиение исходного отрезка на L равных частей, после чего полученные узлы округляются до искомых пикселей (на рисунке 1.63 изображён серый отрезок с началом в точке $A(8.91, 3.16)$ и с концом в точке $B(14.53, 6.64)$, разноцветные точки – промежуточные узлы, разбивающие отрезок AB на $L = 6$ равных частей, чёрные точки с цветными центрами – искомые пиксели).

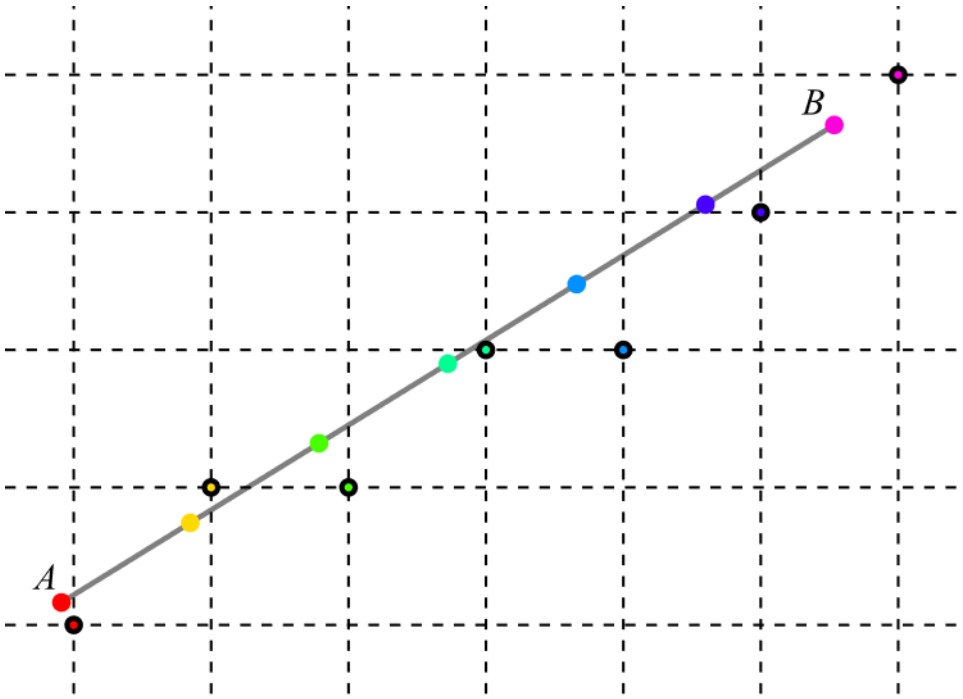


Рисунок 1.63 – Пример работы симметричного алгоритма DDA-линии

В результате работы алгоритма DDA-линии получается 8-связный отрезок.

◆ Докажем это утверждение для отрезка из I октантного угла, т.е. для которого выполнены следующие неравенства и равенство:

$$x_B > x_A, \;\; y_B > y_A, \;\; x_B - x_A > y_B - y_A, \;\; L = \text{round}(x_B) - \text{round}(x_A).$$

Достаточно показать, что два соседних пикселя с координатами (1.56) являются 8-соседями. Оценим слагаемое $\frac{x_B - x_A}{L}$ из формулы (1.55):

$$x_B < \text{round}(x_B) + \frac{1}{2} \,\&\, x_A \geq \text{round}(x_A) - \frac{1}{2} \Rightarrow$$
$$\Rightarrow \frac{x_B - x_A}{L} < \frac{\text{round}(x_B) - \text{round}(x_A) + 1}{L} = \frac{L+1}{L},$$
$$x_B \geq \text{round}(x_B) - \frac{1}{2} \,\&\, x_A < \text{round}(x_A) + \frac{1}{2} \Rightarrow \frac{x_B - x_A}{L} > \frac{L-1}{L}.$$

Таким образом, выполняются неравенства $0 < \frac{L-1}{L} < \frac{x_B - x_A}{L} < \frac{L+1}{L} < 2.$ Далее рассмотрим два случая:

- $1 < \frac{x_B - x_A}{L} < 2.$ Тогда округление нового значения *x*, полученного по формуле (1.55), будет превышать предыдущее либо на 1, либо на 2, т.е. $\Delta x_i \in \{1, 2\},$ где $\Delta x_i = x_{i+1} - x_i$ – приращение абсциссы *i*-ого пикселя, $i = \overline{1, L}.$ С другой стороны, справедливо равенство:

  $$x_{\text{кон}} = x_{\text{нач}} + \Delta x_1 + \ldots + \Delta x_L.$$

  Если предположить, что $\Delta x_i = 2$ для некоторого индекса *i*, то получится следующее:

  $$L = x_{\text{кон}} - x_{\text{нач}} = \Delta x_1 + \ldots + \Delta x_L \geq \underbrace{1 + \ldots + 1}_{L-1 \text{ слагаемое}} + 2 = L + 1$$

  – противоречие. Т.е. $\Delta x_i = 1$ для всякого $i = \overline{1, L}.$

- $0 < \frac{x_B - x_A}{L} < 1.$ В этом случае $\Delta x_i \in \{0, 1\},$ и если $\exists i : \Delta x_i = 0,$ то тогда

  $$L = x_{\text{кон}} - x_{\text{нач}} = \Delta x_1 + \ldots + \Delta x_L \leq \underbrace{1 + \ldots + 1}_{L-1 \text{ слагаемое}} + 0 = L - 1$$

  – снова противоречие. Т.е. опять $\Delta x_i = 1$ для всякого $i = \overline{1, L}.$

Осталось ещё доказать $\Delta y_i \leq 1,$ где $\Delta y_i = y_{i+1} - y_i$ – приращение ординаты *i*-ого пикселя, $i = \overline{1, L}.$ Действительно, если $\exists i : \Delta y_i \geq 2,$ то тогда имеем

$$L \geq y_{\text{кон}} - y_{\text{нач}} = \Delta y_1 + \ldots + \Delta y_L \geq \underbrace{1 + \ldots + 1}_{L-1 \text{ слагаемое}} + 2 = L + 1,$$

что, очевидно, является неверным. Итак, для соседних пикселей, полученных алгоритмом DDA-линии, выполнены неравенства $|\Delta x_i| \leq 1$ и $|\Delta y_i| \leq 1$, что и означает их 8-связность, откуда вытекает 8-связность всего растрированного отрезка. Для отрезков из других октантных углов утверждение доказывается аналогично. ■

Замечание 1. В приведённом выше доказательстве также доказано, что для отрезка из I октантного угла выполнено равенство $\Delta x_i = 1,\ i = \overline{1, L}$. Для других октантных углов выполняются равенства и неравенства, приведённые на рисунке 1.64. Эта схема справедлива и для других алгоритмов растеризации отрезка.

Замечание 2. Иногда вместо формул (1.55) используются следующие, в которых координаты начала и конца исходного отрезка заменены на округлённые значения:

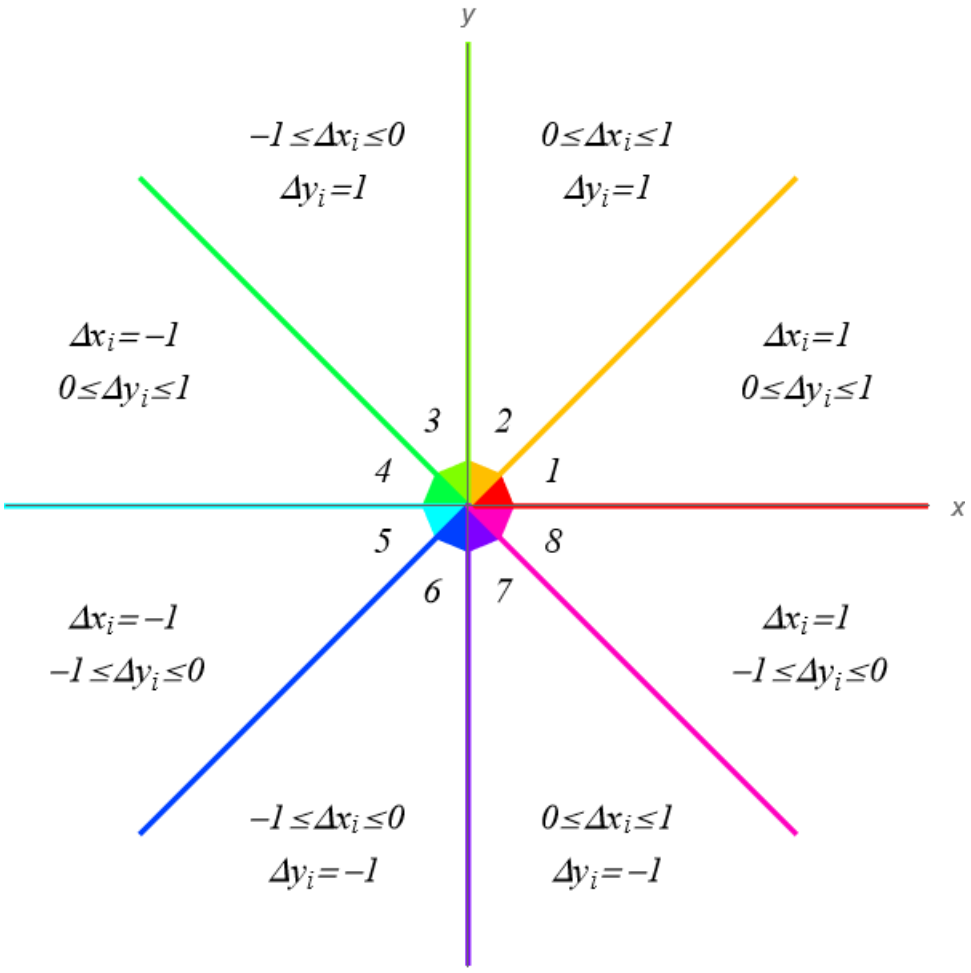


Рисунок 1.64 – Приращения координат пикселей отрезка в зависимости от его октантного угла

$$x := x + \frac{x_{кон} - x_{нач}}{L}, y := y + \frac{y_{кон} - y_{нач}}{L}. \quad (1.57)$$

При этом одно из слагаемых, $\frac{x_{кон} - x_{нач}}{L}$ или $\frac{y_{кон} - y_{нач}}{L}$, всегда равно ±1. Реализация, использующая формулы (1.55), называется **симметричным алгоритмом DDA-линии**, а формулы (1.57) используются в **несимметричном алгоритме DDA-линии**. Несимметрический алгоритм обычно реализуется для частных случаев, например, когда отрезок принадлежит I октантному углу. По сути, в несимметричном алгоритме происходит деление на равные части отрезка с концами в точках с координатами $(x_{нач}, y_{нач})$ и $(x_{кон}, y_{кон})$, при этом промежуточные узлы имеют целочисленные абсциссы или ординаты (Рисунок 1.65).

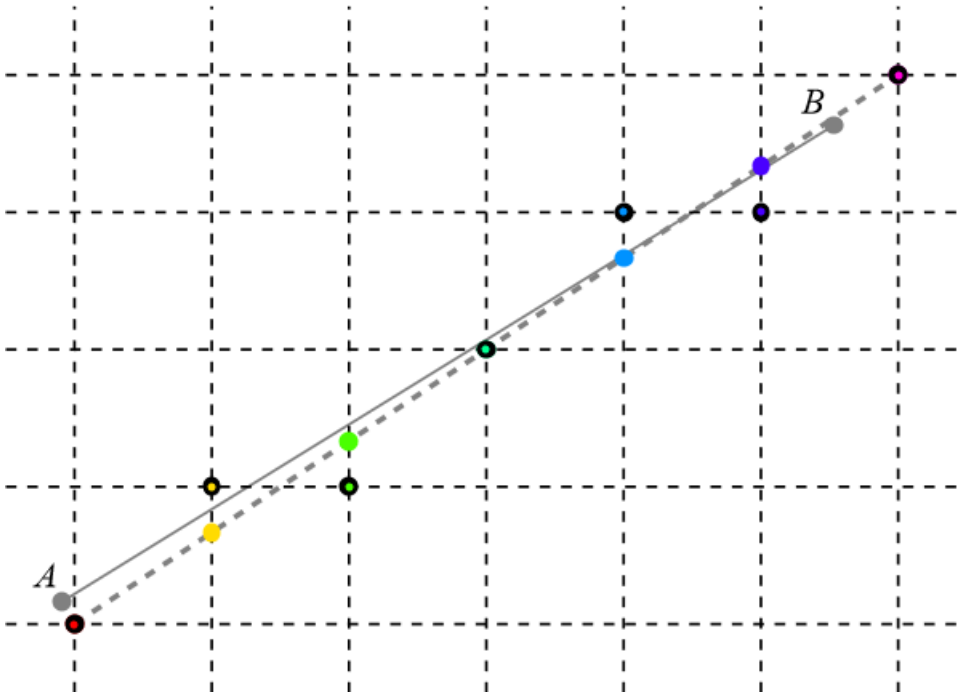


Рисунок 1.65 – Пример работы несимметричного алгоритма DDA-линии (тёмно-зелёная точка оказалась целочисленной)

Как и симметричная реализация, несимметричный алгоритм DDA-линии строит 8-связный отрезок (доказывается аналогично). Однако результаты обоих алгоритмов, вообще говоря, отличаются (на рисунке 1.65, демонстрирующем результат работы для того же отрезка, что и на рисунке 1.63, синий пиксель расположен выше, чем на рисунке 1.63).

Несмотря на простоту идеи, в настоящее время алгоритм DDA-линии растеризации отрезка используется редко. Он не является оптимальным, так как используются операции с действительными числами, в том числе операция деления и округления.

1.3.3. Алгоритм Брезенхема

Алгоритм растеризации отрезка, оперирующий целыми числами, разработан Дж. Э. Брезенхемом в компании IBM в 1962 году [22]. Основная идея этого алгоритма заключается в следующем. Пусть AB – отрезок из I октантного угла с целочисленными координатами начала $A(0,0)$ и конца $B(x_B, y_B)$. В этом случае, согласно замечанию 1 из пункта 1.3.2, на каждом столбце растровой сетки $x = x_i$, $x_i \in \{x_A, x_A + 1, ..., x_B - 1, x_B\}$, должно находиться по одному пикселю искомого растрированного отрезка. Алгоритм Брезенхема вычисляет ординаты пикселей, наиболее близких к исходному отрезку, для каждого столбца.

Отрезок AB лежит на прямой, заданной уравнением

$$y = \frac{y_B}{x_B} x. \tag{1.58}$$

Пусть $(i-1)$-ый пиксель искомого отрезка имеет координаты (x_{i-1}, y_{i-1}) (Рисунок 1.66). Пересечение столбца $x = x_{i-1} + 1$ и прямой (1.58) имеет координаты $\left(x_{i-1} + 1, \frac{y_B}{x_B}(x_{i-1} + 1) \right)$. Обозначим также $\Delta_+ = \left\lceil \frac{y_B}{x_B}(x_{i-1} + 1) \right\rceil -$

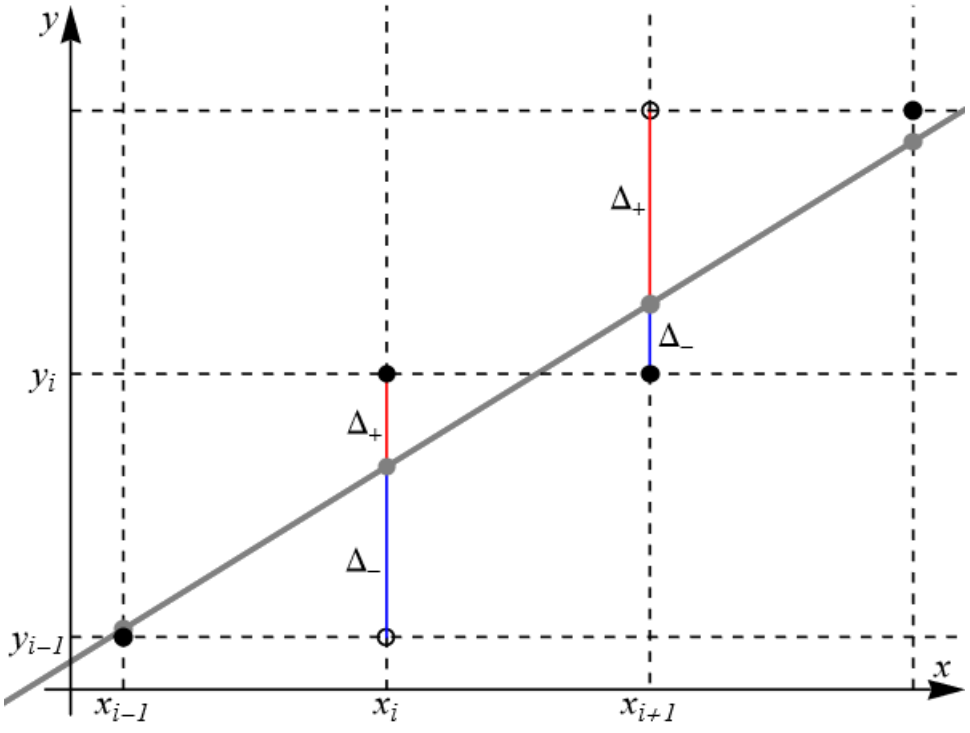


Рисунок 1.66 – Шаг алгоритма Брезенхема

$-\frac{y_B}{x_B}(x_{i-1}+1)$, $\Delta_- = \frac{y_B}{x_B}(x_{i-1}+1) - \left\lfloor \frac{y_B}{x_B}(x_{i-1}+1) \right\rfloor$. В зависимости от того, в какую сторону округляется ордината этой точки, возможны два варианта:

- если $\text{round}\left(\frac{y_B}{x_B}(x_{i-1}+1)\right) = \left\lfloor \frac{y_B}{x_B}(x_{i-1}+1) \right\rfloor$, то координаты *i*-ого пикселя равны $(x_i, y_i) = (x_{i-1}+1, y_{i-1}+1)$, другими словами, приращение ординаты $\Delta y_{i-1} = 1$;
- если $\text{round}\left(\frac{y_B}{x_B}(x_{i-1}+1)\right) = \left\lfloor \frac{y_B}{x_B}(x_{i-1}+1) \right\rfloor$, то тогда $(x_i, y_i) = (x_{i-1}+1, y_{i-1})$, $\Delta y_{i-1} = 0$.

Заметим, что выбор из этих двух вариантов зависит от разности $\Delta_- - \Delta_+$: если она больше либо равна нулю, то имеет место первый случай (на рисунке 1.66 в столбце $x = x_i$ синий отрезок больше красного), а иначе – второй случай (столбец $x = x_{i+1}$, красный отрезок больше синего). Вычислим эту разность:

$$\Delta_- = \frac{y_B}{x_B}(x_{i-1}+1) - \left\lfloor \frac{y_B}{x_B}(x_{i-1}+1) \right\rfloor = \frac{y_B}{x_B}(x_{i-1}+1) - y_{i-1},$$

$$\Delta_+ = \left\lceil \frac{y_B}{x_B}(x_{i-1}+1) \right\rceil - \frac{y_B}{x_B}(x_{i-1}+1) = y_{i-1}+1 - \frac{y_B}{x_B}(x_{i-1}+1),$$

$$\Delta_- - \Delta_+ = 2\frac{y_B}{x_B}(x_{i-1}+1) - 2y_{i-1} - 1. \qquad (1.59)$$

Во избежание операций с действительными числами домножим обе части (1.59) на x_B:

$$x_B(\Delta_- - \Delta_+) = 2y_B(x_{i-1}+1) - x_B(2y_{i-1}+1). \qquad (1.60)$$

Так как $x_B > 0$, то знак левой части равенства (1.60) совпадает со знаком разности $\Delta_- - \Delta_+$, значит, можно рассматривать знак правой части. Обозначим её через d_{i-1}. Выведем рекуррентную формулу для этого выражения, для чего выразим d_i. С одной стороны, используя рассуждения выше, можно вывести равенство, аналогичное (1.60), откуда получится $d_i = 2y_B(x_i+1) - x_B(2y_i+1)$. С другой стороны, используя равенство $x_i = x_{i-1}+1$, имеем следующее:

$$d_i = 2y_B(x_i+1) - x_B(2y_i+1) = 2y_B(x_{i-1}+2) - x_B(2y_i+1),$$

$$d_i - d_{i-1} = \underline{2y_B(x_{i-1}+2)} - \underline{\underline{x_B(2y_i+1)}} - \underline{2y_B(x_{i-1}+1)} +$$

$$+\underline{\underline{x_B(2y_{i-1}+1)}} = \underline{2y_B} - \underline{\underline{2x_B(y_i - y_{i-1})}},$$

$$d_i = d_{i-1} + 2y_B - 2x_B\Delta y_{i-1}.$$

$\Delta y_{i-1} = 0$ в случае, если на предыдущем шаге получили округление вниз, т.е. разность $\Delta_- - \Delta_+$, а значит, d_{i-1} имели отрицательный знак. Аналогично, если $d_{i-1} \geq 0$, то $\Delta y_{i-1} = 1$. Таким образом, рекуррентная формула выглядит следующим образом:

$$d_i = \begin{cases} d_{i-1} + 2(y_B - x_B), & d_{i-1} \geq 0, \\ d_{i-1} + 2y_B, & d_{i-1} < 0. \end{cases} \tag{1.61}$$

Для того, чтобы можно было её использовать, необходимо задать начальное значение d_1. Полагая в равенстве (1.59) $x_{i-1} = y_{i-1} = 0$, имеем

$$\Delta_- - \Delta_+ = 2\frac{y_B}{x_B} - 1 \Rightarrow d_1 = x_B(\Delta_- - \Delta_+) = 2y_B - x_B. \tag{1.62}$$

Итак, вывели **алгоритм Брезенхема** растеризации отрезка <u>из I октантного угла</u>. Пусть начало отрезка расположено в начале координат, а его конец имеет координаты (x_B, y_B), $x_B > y_B > 0$. Тогда выполняются следующие шаги:

1. Начало координат отмечается как первый пиксель искомого растрированного отрезка.
2. Итерационная переменная: i := 1.
3. Для нового, (i + 1)-ого пикселя, вычисляется d_i по рекуррентным формулам (1.62) и (1.61).
4. Вычисляется приращение ординаты текущего пикселя: $\Delta y_i = \begin{cases} 0, d_i < 0, \\ 1, d_i \geq 0. \end{cases}$
5. Вычисляются координаты следующего пикселя: $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + \\ + \Delta y_i)$, где (x_i, y_i) – координаты текущего пикселя.
6. Если $i = x_B$, то алгоритм завершает свою работу, возвращая все полученные пиксели. Иначе i := i + 1, возврат к шагу 3.

В результате работы алгоритма Брезенхема получается 8-связный отрезок, так как приращения координат пикселей по модулю не превышают единицы.

<u>Замечание 1.</u> Для растеризации при помощи алгоритма Брезенхема отрезков из других октантных углов, а также в случае, когда начало отрезка не совпадает с началом координат, достаточно вначале применить нужные аффинные преобразования плоскости (симметрические отражения и параллельный перенос), а после растеризации применить обратные преобразования.

Замечание 2. Если исходный отрезок имеет концы в точках с нецелочисленными координатами, то можно поступить двумя способами: первоначально осуществить их округление, либо осуществить растеризацию с субпиксельной точностью.

1.3.4. Код Ротштейна. Алгоритм Кастла-Питвея

Как было показано выше (пункты 1.3.2 и 1.3.3), при растеризации отрезка из I октантного угла очередной пиксель рисуется со смещением абсциссы $\Delta x_i = 1$ и смещением ординаты Δy_i, равным 0 или 1. Таким образом, при движении от предыдущего пикселя растрированного отрезка к следующему происходит сдвиг либо вправо, либо по диагонали. Эти движения можно закодировать следующим образом: сдвиг вправо обозначается буквой S (англ. *square* – квадрат), сдвиг по диагонали – буквой D (*diagonal*). Весь растрированный отрезок представи́м в виде строки из этих двух букв, которая называется **кодом Ротштейна** этого отрезка. Например, код Ротштейна для отрезка с координатами конца, равными (22, 9), выглядит следующим образом: S D S D SS D S D S D SS D S D SS D S D S (Рисунок 1.67).

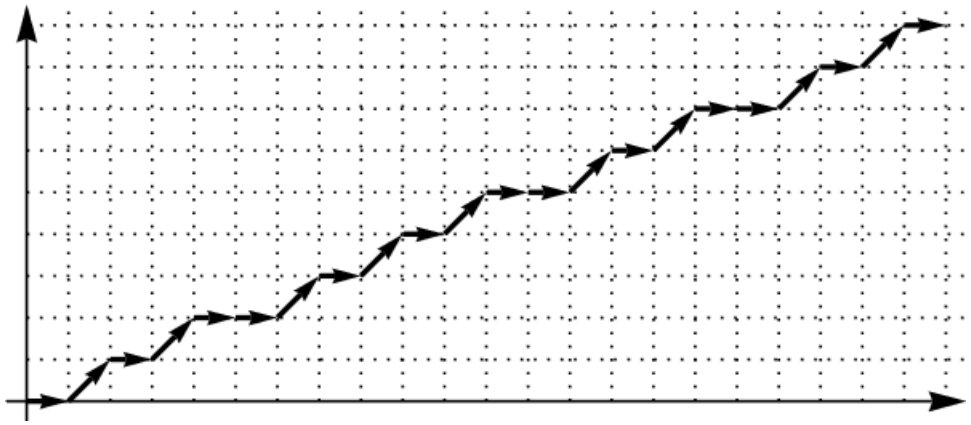


Рисунок 1.67 – Пример рисования растрированного отрезка путём горизонтальных и диагональных смещений

Одним из свойств кода Ротштейна является его квазисимметричность. В большинстве случаев код Ротштейна симметричен, что объясняется независимостью растеризации от порядка вычисления пикселей (от начала до конца отрезка либо наоборот). Однако если отрезок проходит через точку $\left(x', y' + \frac{1}{2}\right)$ при некоторых целых x' и y', то переворачивание кода Ротштейна даст растеризацию отрезка, которая бы получилась в случае, если бы имело место округление вниз, т.е. если бы в алгоритме Брезенхема выбиралось приращение $\Delta y_i = 0$ при $d_i = 0$ (Рисунок 1.68).

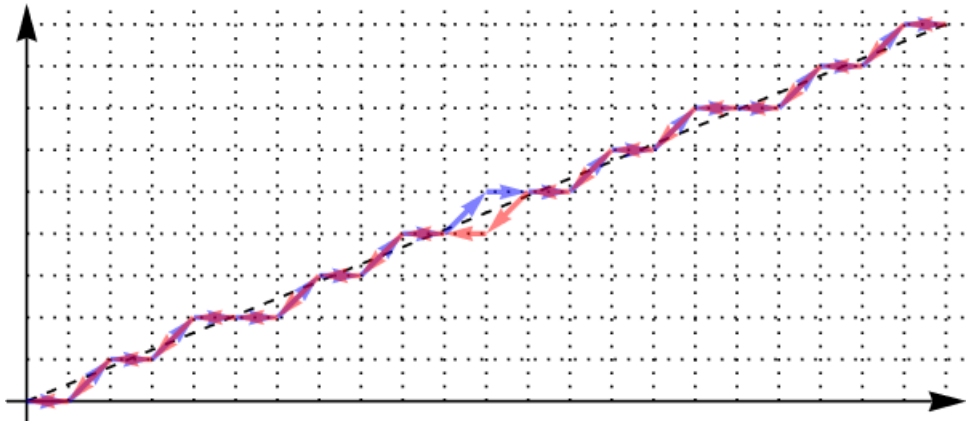


Рисунок 1.68 – Переворачивание исходного кода Ротштейна (синяя ломаная) даёт немного отличающуюся красную ломаную

К 1985 году К. Кастл и М. Питвей при содействии вышеупомянутого Брезенхема разработали эффективный алгоритм построения кода Ротштейна для произвольного отрезка из I октанта [23]. Он основан на построении приближений исходного отрезка, которые также выражаются при помощи кода Ротштейна; по мере работы алгоритма они становятся всё ближе и ближе к исходному отрезку, пока не совпадут с ним.

Перед описанием алгоритма необходимо ввести некоторые операции с кодами Ротштейна:

- Переворот строки обозначается тильдой. Например, если m = SSSD, то $\sim m$ = DSSS.
- Конкатенация строк обозначается символом $\oplus$. Например, если m_1 = = SDS, m_2 = DDS, то $m_1 \oplus m_2$ = SDSDDS.

Алгоритм Кастла-Питвея, выполняющий построение кода Ротштейна для отрезка с концами в точках с координатами (0, 0) и (x_B, y_B), $x_B > y_B > 0$, выглядит следующим образом:

1. Вводятся две переменные, в которых хранятся коды Ротштейна для приближений искомого отрезка: $m_1 := \mathrm{S}$, $m_2 := \mathrm{D}$. Строка m_1 представляет приближение снизу, m_2 – приближение сверху.
2. Вводятся также две вспомогательные переменные $x, y \in \mathbb{N}$: $y := y_B$, $x := x_B - y_B$.
3. Сравниваем x и y. Если $x > y$, то происходит переход к шагу 4, если $x < y$ – к шагу 5, $x = y$ – к шагу 6.
4. В случае $x > y$ происходит уточнение верхнего приближения m_2 и переменной x: $m_2 := m_1 \oplus \sim m_2$, $x := x - y$. Возврат к шагу 3.
5. Если $x < y$, то уточняется нижнее приближение m_1 и переменная y: $m_1 := m_2 \oplus \sim m_1$, $y := y - x$. Возврат к шагу 3.
6. Если $x = y$, то окончательным результатом является строка $m_2 \oplus \sim m_1$, повторённая x раз.

Пример работы алгоритма Кастла-Питвея при $x_B = 11$, $y_B = 3$ приведён на рисунке 1.69. Красными и синими жирными ломаными линиями обозначены растрированные отрезки, которым отвечают коды m_1 и m_2 соответственно (на всех рисунках, кроме первого и последнего, один из отрезков накладывается поверх другого), пунктирные линии – лучи, проведённые через начало координат и концы этих отрезков. Синий пунктирный луч отсекает x пикселей справа на верхней границе прямоугольника, построенного на растровой сетке, с вершинами в точках (0, 0) и (x_B, y_B), красный – y пикселей сверху на правой стороне этого же прямоугольника.

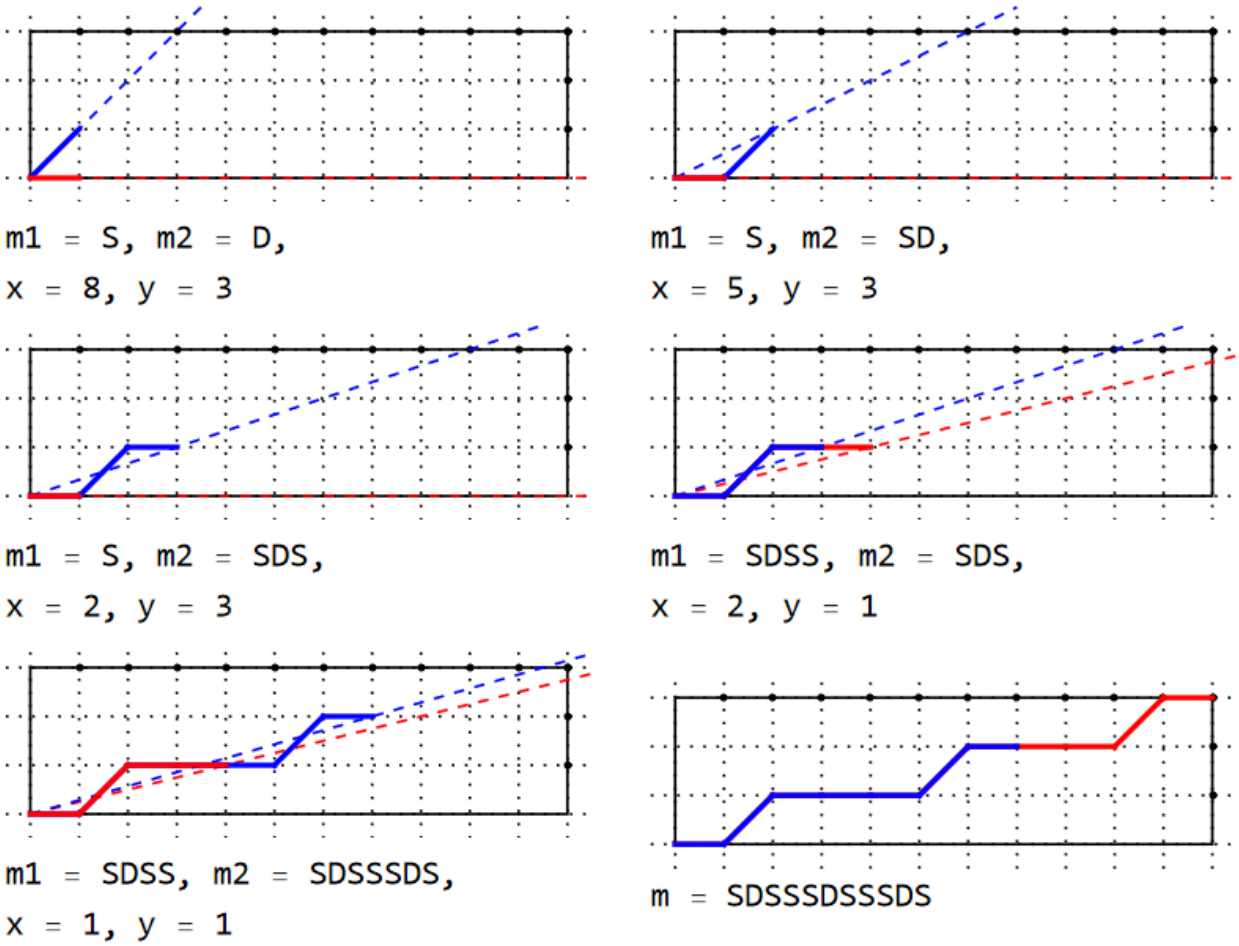


Рисунок 1.69 – Итерации алгоритма Кастла-Питвея и его результат

Алгоритм Кастла-Питвея представляет интерес не столько в практических приложениях, сколько с точки зрения алгоритмики. В результате этого алгоритма получается та же растеризация, что и после алгоритма Брезенхема. Доказательство этого факта вынесено в Приложение 1.

1.3.5. Растеризация отрезка со сглаживанием. Алгоритм Ву

Под **интенсивностью** пикселя (x, y) будем понимать степень влияния расположения некоторого геометрического объекта на его окраску. Обозначим интенсивность через $I(x, y)$ и положим $I(x, y) \in [0,1]$. Для окрашивания пикселя с интенсивностью i его цвет вычисляется по следующей формуле:

$$\vec{c} = (1-i)\vec{c}_{\text{ф}} + i\vec{c}_{\text{о}}, \tag{1.63}$$

где $\vec{c}_{\text{ф}}$ – цвет фона, поверх которого происходит рисование, $\vec{c}_{\text{о}}$ – цвет объекта. В частности, при $i = 0$ получится $\vec{c} = \vec{c}_{\text{ф}}$ – пиксель останется незакрашенным в результате рисования объекта, $i = 1 \Rightarrow \vec{c} = \vec{c}_{\text{о}}$ – пиксель окрасится в цвет объекта.

Если цвет фона – белый, а исходный отрезок имеет чёрный цвет, то формулу (1.63) можно переписать, используя оттенки серого. Оттенок серого обычно выражается действительным числом $g(x, y)$ из отрезка [0, 1] или целым числом из некоторого отрезка $[0, M-1]$, где нулю соответствует абсолютно чёрный цвет, максимальному значению – абсолютно белый. Другими словами, чем больше значение g, тем ярче сам пиксель. Если обозначить через $g_{\text{ф}}$ оттенок фона (который является максимальным числом), через $g_{\text{о}}$ – оттенок отрезка (равный нулю), то тогда получим следующее:

$$g(x, y) = (1-i)g_{\text{ф}} + ig_{\text{о}} = (1-i)g_{\text{ф}}, \tag{1.64}$$

где $g(x, y)$ – оттенок серого, в который окрашивается пиксель с координатами (x, y). Сравнивая формулы (1.63) и (1.64), отметим, что при рисовании чёрного отрезка более интенсивному пикселю соответствует более тёмный цвет (Рисунок 1.70).

При рисовании отрезка из I октантного угла интенсивность ближайших к нему пикселей рассчитывается по следующей формуле:

$$I(X, Y) = 1 - \left|Y - Y'\right|, \tag{1.65}$$

где Y' – ордината точки пересечения исходного отрезка со столбцом $x = X$ растровой сетки. На рисунке 1.70 интенсивность закрашенных пикселей снизу исходного отрезка равна расстояниям, отмеченным красными линиями, пикселей, находящихся выше отрезка – длинам синих отрезков. Нетрудно видеть, что сумма интенсивностей двух окрашенных пикселей из одного столбца равна единице.

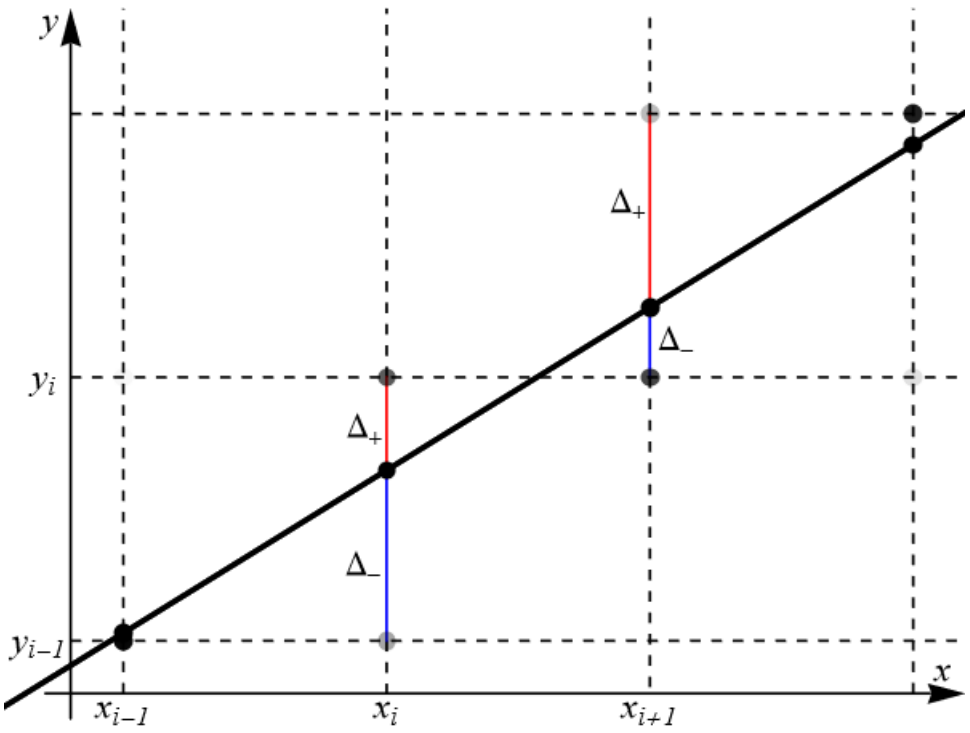


Рисунок 1.70 – Пример рисования сглаженного отрезка

Целочисленный алгоритм рисования сглаженного отрезка. Один из подходов, минимизирующих количество операций с действительными числами, предложен Сяолинем У в 1991 году [24]. Наряду с целочисленной растровой сеткой рассматривается также сетка, образованная делением единичного отрезка на N равных частей (на рисунке 1.71б добавлена вспомогательная сетка, разделяющая единичный отрезок основной сетки на $N = 10$ равных частей, а исходные пиксели с рисунка 1.71а – на $N^2 = 100$ субпикселей).

Пусть необходимо растрировать чёрный отрезок из I октантного угла, начало которого находится в начале координат, а конец – в точке $B(x_B, y_B)$.

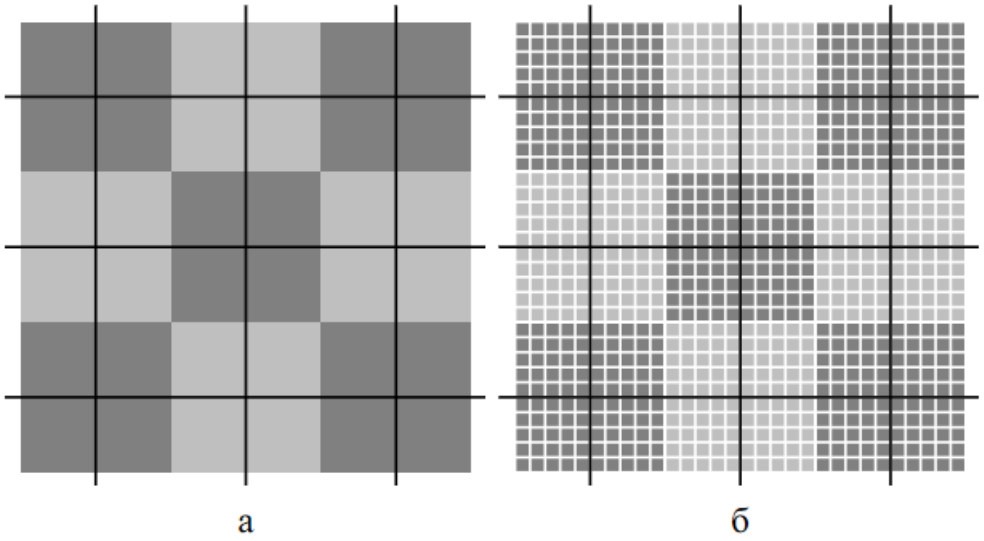


Рисунок 1.71 – Деление пикселей на субпиксели

Пусть $P_{i-1}(x_{i-1}, y_{i-1})$, $i = \overline{1, x_B}$, – координаты пересечения этого отрезка с очередным столбцом основной растровой сетки. Через x'_{i-1} и y'_{i-1} обозначим координаты этой же точки в субпикселях. Тогда $x'_i = Nx_i$, $y'_i = Ny_i$, $i = \overline{0, x_B}$.

Следующее пересечение P_i получится, если точке P_{i-1} дать приращение абсциссы $\Delta x = 1$, или $\Delta x' = N$ субпикселей, и приращение ординаты $\Delta y = \frac{y_B}{x_B}$, что составляет $\Delta y' = \frac{y_B}{x_B} N$ субпикселей. Округляя $\Delta y'$, можно получить целочисленный шаг

$$d = \text{round}\left(\frac{y_B}{x_B} N\right), \tag{1.66}$$

на который будет увеличиваться ордината точки пересечения отрезка со столбцом целочисленной сетки, выраженная в субпикселях: $y'_i = y'_{i-1} + d$. Для отрезка с началом в начале координат будем иметь $y'_i \in \mathbb{Z}$, $i = \overline{0, x_B}$.

Нужно принять во внимание, что при увеличении субпиксельной ординаты на d имеет место погрешность $\Delta = \frac{y_B}{x_B} N - \text{round}\left(\frac{y_B}{x_B} N\right)$. Это приводит к тому, что фактически растеризации подвергается другой отрезок (на рисунке 1.72 это пунктирный отрезок, который строится по концам красных отрезков с длинами,

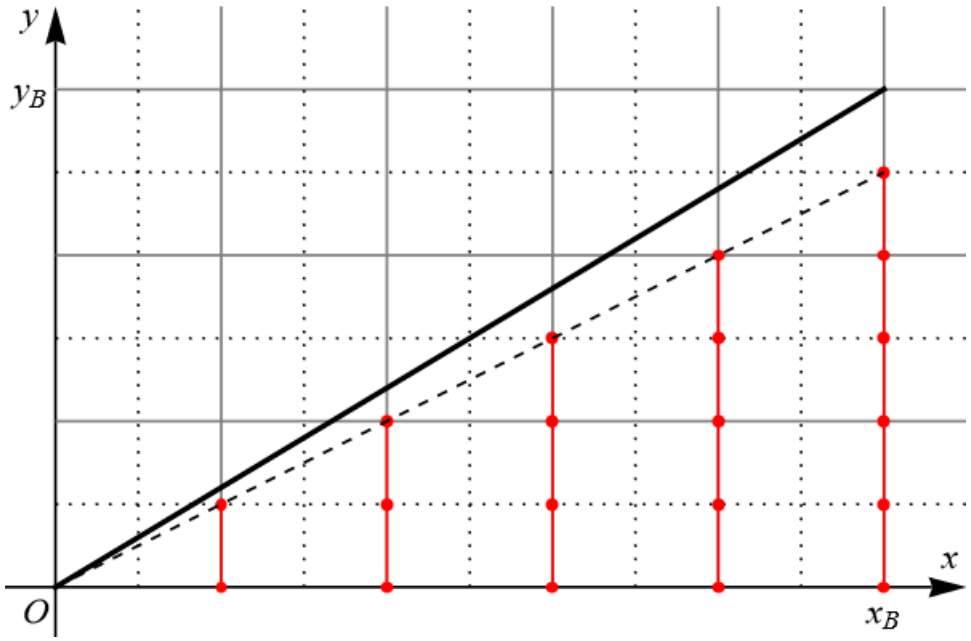


Рисунок 1.72 – Исходный отрезок (жирная линия) с концом в точке $B(x_B, y_B)$ при $x_B = 5$ и $y_B = 3$ и фактически растрируемый отрезок (пунктирная линия) при делении растровой сетки на $N = 2$

кратными d). Следовательно, необходимо оценить эту погрешность для того, чтобы задать условия для выбора N с тем, чтобы оба отрезка не сильно отличались друг от друга. Согласно правилам округления имеет место неравенство

$$|\Delta| = \left| \frac{y_B}{x_B} N - \text{round}\left(\frac{y_B}{x_B} N \right) \right| \le \frac{1}{2}. \qquad (1.67)$$

После рисования всех пикселей искомого отрезка накопится погрешность, в x_B раз бóльшая, чем Δ. Домножая неравенство (1.67) на x_B, получим верхнюю границу, равную $\frac{1}{2} x_B$ субпикселей, или $\frac{1}{2N} x_B$ пикселей. Таким образом, для рисования растрированного отрезка по алгоритму By необходимо выполнение неравенства $N > \frac{x_B}{2\Delta_{\max}}$, где $\Delta_{\max}$ – максимальная допустимая погрешность в пикселях.

После очередного смещения нужно найти координаты пикселей $L_i(x_{li}, y_{li})$ и $H_i(x_{hi}, y_{hi})$, наиболее близких к точке P_i (Рисунок 1.73). Их абсциссы будут равны x_i, а для вычисления ординат достаточно найти целые числа, ближайшие к $y_i = \frac{y'_i}{N}$. Но во избежание операций с дробными числами лучше оперировать не самой субпиксельной ординатой y'_i, а числом, равным расстоянию в субпикселях от точки P_i до ближайшей строки целочисленной сетки, расположенной ниже этой точки. Обозначим это число D_i (на рисунке 1.73 эти расстояния отмечены синими отрезками). Тогда по определению получим $D_i = \lfloor N\{y_i\} \rfloor$. Из этой формулы следует $D_i \in [0, N)$.

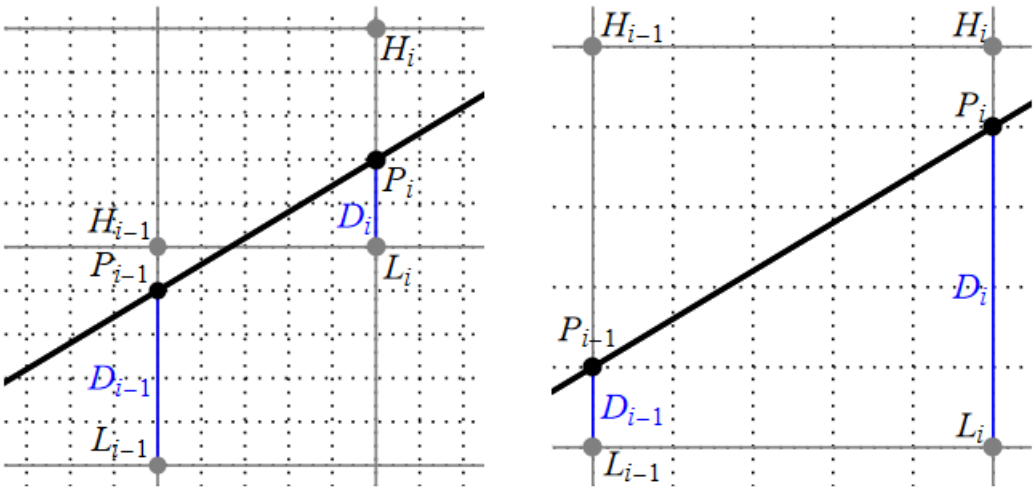


Рисунок 1.73 – Вычисление пикселей, ближайших к точке P_i

На очередной итерации может возникнуть один из двух случаев:

- При движении по отрезку от точки P_{i-1} к P_i происходит пересечение целочисленной строки (Рисунок 1.73а). Тогда $D_{i-1} + d \geq N$, а $D_i = D_{i-1} + d - N$. Кроме того, ордината точки L_i увеличивается: $y_{li} = y_{l,i-1} + 1$. Ордината точки H_i всегда на единицу больше: $y_{hi} = y_{li} + 1$.
- Пересечения строки не происходит (Рисунок 1.73б). В этом случае $D_i = D_{i-1} + d$, $y_{li} = y_{l,i-1}$, $y_{hi} = y_{li} + 1$.

Наконец, после нахождения координат точек L_i и H_i вычисляются их цвета, являющиеся оттенками серого. Для этого воспользуемся формулами (1.64) и (1.65):

$$g(L_i) = (1 - I(L_i))(M-1) = (M-1)(y_i - y_{li}) = (M-1)\{y_i\} \approx M\frac{D_i}{N} \approx \left\lfloor \frac{D_i}{N/M} \right\rfloor,$$

$$g(H_i) = (1 - I(H_i))(M-1) = (M-1)(y_{hi} - y) = (M-1)(1 - \{y_i\}) =$$

$$= M - 1 - M\{y_i\} + \{y_i\} \approx M - 1 - \left\lfloor \frac{D_i}{N/M} \right\rfloor,$$

где M – число оттенков серого, выраженных целыми числами (чёрному цвету соответствует 0, белому – $M - 1$).

Заметим, что растрировать отрезок AB можно, начиная одновременно с обеих точек A и B и продолжая до тех пор, пока не достигнута середина. **Алгоритм By** растеризации чёрного отрезка AB с началом в точке $A(0, 0)$ и концом в точке $B(x_B, y_B)$, $x_B > y_B > 0$, состоит из следующих шагов:

1. Определяется коэффициент N дробления растровой сетки и количество оттенков серого M.
2. Итерационной переменной $D \in \mathbb{N}_0$ присваивается начальное значение $D := 0$. Каждый шаг она будет увеличиваться на число d, вычисляемое по формуле (1.66).
3. Вспомогательным переменным $x_1, y_1, x_2, y_2 \in \mathbb{N}_0$, где будут храниться соответственно абсцисса и ордината точки текущего начала, абсцисса и ордината точки текущего конца, присваиваются значения $x_1 := 0$, $y_1 := 0$, $x_2 := x_B$, $y_2 := y_B$.
4. Пиксели начала и конца растрированного отрезка являются абсолютно чёрными: $g(0,0) = g(x_B, y_B) = 0$.
5. Абсциссы текущего начала и конца сближаются друг к другу: $x_1 := x_1 + 1$, $x_2 := x_2 - 1$. Если в результате оказалось справедливым неравенство $x_1 > x_2$, то алгоритм завершает свою работу, так как дошли до середины отрезка.
6. Увеличивается итерационная переменная: $D := D + d$.

7. Если $D \geq N$, тогда происходит сближение ординат точек текущего начала и конца: $y_1 := y_1 + 1$, $y_2 := y_2 - 1$, а сама итерационная переменная переопределяется: $D := D - N$.
8. Определяется оттенок новых точек текущего начала и конца:

$$g(x_1, y_1) = g(x_2, y_2) = \left\lfloor \frac{D}{N/M} \right\rfloor, \tag{1.68}$$

а также двух смежных с ними пикселей:

$$g(x_1, y_1 + 1) = g(x_2, y_2 - 1) = M - 1 - g(x_1, y_1). \tag{1.69}$$

9. Возврат к шагу 5.

Примеры растеризации отрезка с концом в точке с координатами $x_B = 17$ и $y_B = 6$ со сглаживанием и без приведены на рисунке 1.74. На рисунках 1.74б, 1.74в использованы только $M = 2$ оттенка серого – просто чёрный и белый цвета. При этом при недостаточно большом значении N (Рисунок 1.74в) происходит разрыв отрезка. От выбора количества оттенков серого M зависит степень сглаженности отрезка (рисунки 1.74, г-е).

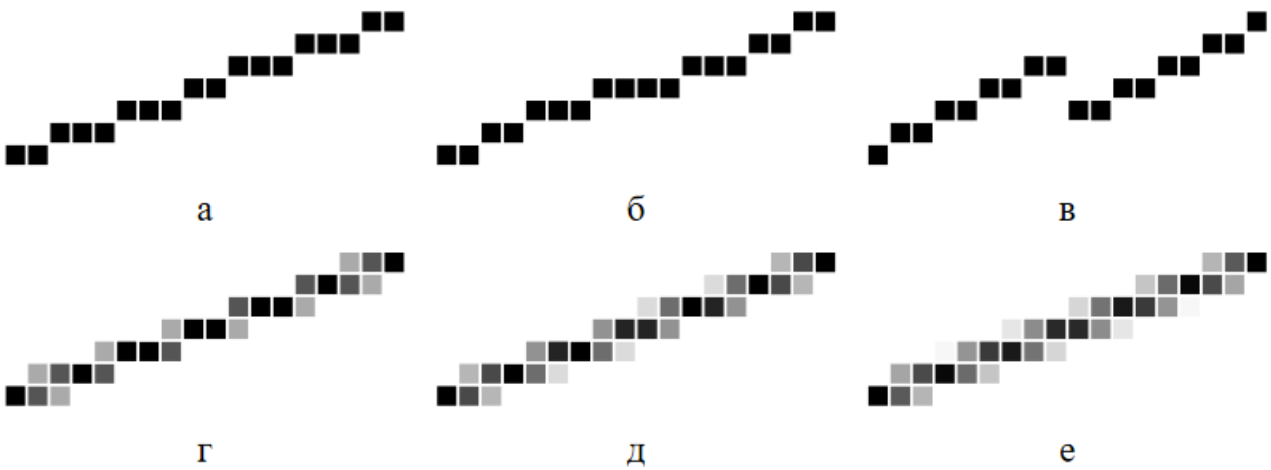


Рисунок 1.74 – Растеризация отрезка по алгоритму Брезенхема (а) и по алгоритму Ву при разных значениях параметров: $N = 8$, $M = 2$ (б); $N = 2$, $M = 2$ (в); $N = 256$, $M = 4$ (г); $N = 256$, $M = 8$ (д); $N = 256$, $M = 32$ (е);

Замечание 1. Описанный выше алгоритм, вообще говоря, не является целочисленным. Обычно в качестве параметров N и M используются степени двойки: $N = 2^n$, $M = 2^m$, $n > m$. Тогда число D можно представить n-битным числом, оттенки серого, вычисляемые по формулам (1.68) – (1.69) – в виде m-битных чисел. Отсюда получается, что проверку неравенства $D \geq N$ можно заменить на проверку переполнения n-битного числа (для чего нужен ещё один бит), формулу (1.68) можно заменить на операцию взятия первых m бит числа

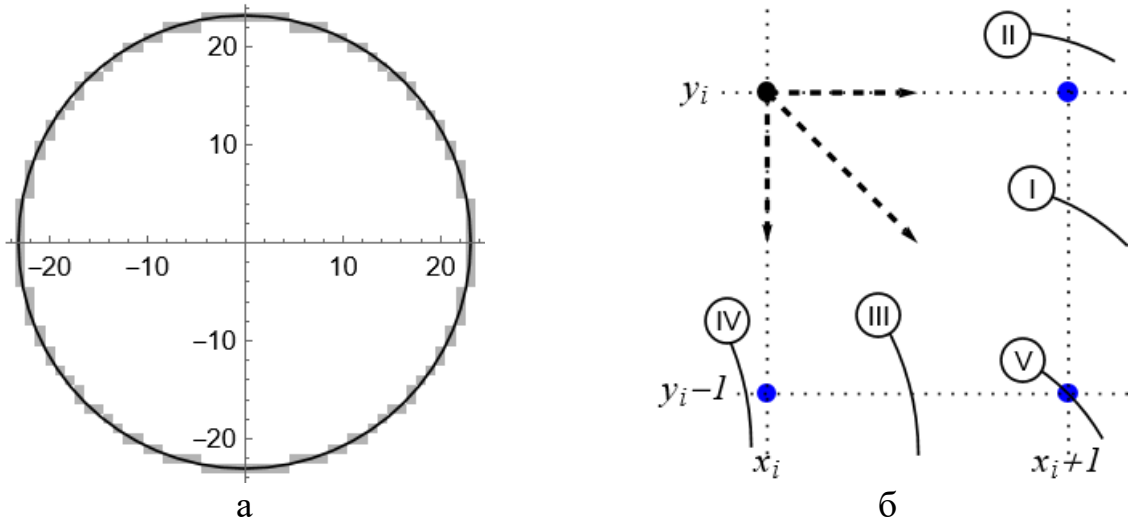
D, формулу (1.69) – на операцию инвертирования M-битного числа $g(x_1, y_1)$. Таким образом, единственная оставшаяся нецелочисленная операция – это вычисление шага d.

Замечание 2. Как было показано выше, наименьшее значение для параметра N, необходимое для корректного рисования растрированного отрезка, вообще говоря, зависит от размеров отрезка. На практике значение N не зависит от длины отрезка, потому что выбирается таким образом, чтобы погрешность растеризации была заведомо допустимой для любого отрезка. Например, при выводе на дисплей с расширением 1024×1024 пикселей для растеризации отрезка со сглаживанием при помощи 32 оттенков серого достаточно использования $n = 15$ бит, т.е. для хранения переменной D достаточно двух байт [24].

1.3.6. Растеризация окружностей и эллипсов

Для нахождения множества пикселей, наилучшим образом приближающих некоторую окружность, Брезенхемом был разработан ещё один алгоритм, в последствии названный в его честь [25].

Окружность с центром в точке $A(x_A, y_A)$ радиуса R задаётся уравнением $(x - x_A)^2 + (y - y_A)^2 = R^2$. При $x_A \in \mathbb{Z}$, $y_A \in \mathbb{Z}$ можно осуществить параллельный перенос на вектор $\overrightarrow{AO}(-x_A, -y_A)$, который переведёт центр окружности в начало координат. Растеризация исходной окружности при таком преобразовании перейдёт в растеризацию окружности с центром в начале координат. Легко видеть, что она должна быть симметричной относительно обеих координатных осей (Рисунок 1.75а). Следовательно, достаточно построить растеризацию дуги, расположенной в I квадранте.



а б

Рисунок 1.75 – Растеризация окружности с центром в начале координат

При последовательной отрисовке дуги окружности $x^2 + y^2 = R^2,\ x \geq 0,$ $y \geq 0,$ от верхней левой точки $(0, R)$ до нижней правой $(R, 0)$ происходит движение вправо и/или вниз. Таким образом, относительно текущего пикселя (x_i, y_i) следующий пиксель (x_{i+1}, y_{i+1}) может иметь смещение вправо: $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$, вниз: $(x_{i+1}, y_{i+1}) = (x_i, y_i - 1)$, либо диагональное: $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$ (Рисунок 1.75б) Из этих трёх пикселей необходимо выбрать тот, для которого выражение $x^2 + y^2 - R^2$ минимально по модулю, что означает наибольшую близость этого пикселя к рассматриваемой окружности. Значит, можно сравнить абсолютные значения трёх выражений

$$\begin{gathered} (x_i + 1)^2 + y_i^2 - R^2, \\ (x_i + 1)^2 + (y_i - 1)^2 - R^2, \\ x_i^2 + (y_i - 1)^2 - R^2 \end{gathered} \tag{1.70}$$

и выбрать пиксель, для которого соответствующее значение меньше других. Но для упрощения вычислений на каждой итерации достаточно рассматривать только две из этих разностей. Действительно, рассмотрим это выражение для смещённого по диагонали пикселя:

$$\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2.$$

В зависимости от знака этого числа имеет место один из следующих случаев:

1. Неравенство $\Delta_i < 0$ означает, что точка с координатами $(x_i + 1, y_i - 1)$ расположена внутри круга $x^2 + y^2 - R^2 < 0$. Значит, окружность в этом случае будет проходить либо между точками $(x_i + 1, y_i - 1)$ и $(x_i + 1, y_i)$ (вариант I на рисунке 1.75б) либо между точками $(x_i + 1,\ y_i)$ и $(x_i + 1, y_i + 1)$ (вариант II). В варианте I выбор нужного пикселя определяется следующей разностью:

$$\delta_- = \left|(x_i + 1)^2 + y_i^2 - R^2\right| - \left|(x_i + 1)^2 + (y_i - 1)^2 - R^2\right|.$$

С учётом положения этих двух пикселей относительно окружности имеем неравенства

$$\begin{gathered} (x_i + 1)^2 + y_i^2 - R^2 \geq 0, \\ (x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0, \end{gathered}$$

откуда получаем

$$\delta_- = (x_i+1)^2 + y_i^2 - R^2 - (-(x_i+1)^2 - (y_i-1)^2 + R^2) = 2(x_i+1)^2 + (y_i-1)^2 - 2R^2 + y_i^2.$$

Отнимая и добавляя $2y_i - 1,$ это значение можно привести к виду

$$\begin{aligned}\delta_- &= 2(x_i+1)^2 + (y_i-1)^2 - 2R^2 + y_i^2 = \underline{2(x_i+1)^2 +} \\ &\underline{+(y_i-1)^2 - 2R^2 + (y_i-1)^2} + 2y_i - 1 = \underline{2\Delta_i} + 2y_i - 1.\end{aligned} \tag{1.71}$$

Таким образом, по знаку выражения (1.71), вычисляемого с использованием только одной из разностей (1.70), можно определить следующий пиксель для растеризации окружности: при $\delta_- \le 0$ необходимо выбирать пиксель $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i),$ а если $\delta_- > 0,$ тогда $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1).$ Что касается варианта II, то здесь непременно должен быть выбран пиксель $(x_i + 1, y_i).$ Значение $\delta_-,$ вычисляемое по формуле (1.71), в этой ситуации всегда меньше нуля. Действительно, в варианте II оба пикселя $(x_i + 1, y_i)$ и $(x_i + 1, y_i - 1)$ расположены внутри круга, значит, выполняются неравенства

$$\begin{gathered}(x_i+1)^2 + (y_i-1)^2 - R^2 < 0, \\ (x_i+1)^2 + y_i^2 - R^2 < 0,\end{gathered}$$

которые в сумме дают $\delta_- < 0.$ Значит, варианты I и II можно объединить и сформулировать следующее правило:

$$(x_{i+1}, y_{i+1}) = \begin{cases}(x_i+1, y_i-1), \Delta_i < 0 \,\&\, \delta_- > 0, \\ (x_i+1, y_i), \Delta_i < 0 \,\&\, \delta_- \le 0.\end{cases} \tag{1.72}$$

2. Неравенство $\Delta_i > 0$ эквивалентно прохождению окружности между точками $(x_i + 1, y_i - 1)$ и $(x_i, y_i - 1)$ (вариант III) либо между точками $(x_i, y_i - 1)$ и $(x_i - 1, y_i - 1)$ (вариант IV). Аналогично выводится правило для этих двух вариантов:

$$(x_{i+1}, y_{i+1}) = \begin{cases}(x_i, y_i-1), \Delta_i > 0 \,\&\, \delta_+ > 0, \\ (x_i+1, y_i-1), \Delta_i > 0 \,\&\, \delta_+ \le 0,\end{cases} \tag{1.73}$$

где

$$\delta_+ = 2\Delta_i - 2x_i - 1. \tag{1.74}$$

3. Если же $\Delta_i = 0$, то тогда окружность проходит через точку $(x_i + 1, y_i - 1)$, и это и будет следующий пиксель искомой растеризации:

$$(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1), \text{ если } \Delta_i = 0. \quad (1.75)$$

Таким образом, в ходе текущей итерации происходят два действия:

- По текущему значению Δ_i и текущему пикселю (x_i, y_i) определяется положение следующего пикселя. Объединяя правила (1.72), (1.73) и (1.75), получаем следующие формулы:

$$\Delta_i < 0 \,\&\, \delta_- \le 0 \Rightarrow (x_{i+1}, y_{i+1}) = (x_i + 1, y_i), \quad (1.76)$$

$$\Delta_i > 0 \,\&\, \delta_+ > 0 \Rightarrow (x_{i+1}, y_{i+1}) = (x_i, y_i - 1), \quad (1.77)$$

$$(\Delta_i \le 0 \,\&\, \delta_- > 0) \vee (\Delta_i \ge 0 \,\&\, \delta_+ \le 0) \Rightarrow (x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1). \quad (1.78)$$

- Для следующей итерации вычисляется значение Δ_{i+1}:

$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2. \quad (1.79)$$

В случае, когда происходит увеличение абсциссы пикселя на 1, формула (1.79) переписывается в следующем виде:

$$\begin{aligned} \Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 1 + 1)^2 + (y_i - 1)^2 - R^2 = \\ &= \underline{(x_i + 1)^2} + 2(x_i + 1) + 1 + \underline{(y_i - 1)^2 - R^2} = \underline{\Delta_i} + 2(x_i + 1) + 1 = \Delta_i + 2x_{i+1} + 1. \end{aligned} \quad (1.80)$$

Если происходит уменьшение ординаты на 1, то тогда (1.79) преобразуется следующим образом:

$$\begin{aligned} \Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 1)^2 + (y_i - 1 - 1)^2 - R^2 = \\ &= \underline{(x_i + 1)^2 + (y_i - 1)^2} - 2(y_i - 1) + 1 - \underline{R^2} = \underline{\Delta_i} - 2(y_i - 1) + 1 = \Delta_i - 2y_{i+1} + 1. \end{aligned} \quad (1.81)$$

Если происходит и то, и другое, то получается следующая формула:

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2. \quad (1.82)$$

Таким образом, вывели формулы (1.76) – (1.78), (1.80) – (1.82), рекуррентно вычисляющие координаты очередного пикселя и значения Δ_{i+1} по аналогичным значениям с предыдущей итерации. В качестве первоначальных значений этих величин перед первой итерацией необходимо положить

$$(x_1, y_1) = (0, R),$$
$$\Delta_1 = (x_1 + 1)^2 + (y_1 - 1)^2 - R^2 = 1^2 + (R-1)^2 - R^2 = 2 - 2R.$$

Из всего вышесказанного следует **алгоритм Брезенхема растеризации окружности** с центром в начале координат и радиусом $R \in \mathbb{Z}_+$:

1. Происходит инициализация переменных: $i := 1,\ x_1 = 0,\ y_1 = R,\ \Delta_1 = 2 - 2R$.
2. Если $y_i < 0$, то алгоритм заканчивает работу и возвращает все пиксели, ранее записанные в результат.
3. Пиксели с координатами $(x_i, y_i),\ (-x_i, y_i),\ (-x_i, -y_i),\ (x_i, -y_i)$ из I, II, III и IV квадрантов записываются в результат.
4. Вычисляются значения δ_- и δ_+ по формулам (1.71) и (1.74).
5. Далее может возникнуть ровна одна из трёх ситуаций:
   5а. Если $\Delta_i < 0 \,\&\, \delta_- \le 0$, тогда происходит вычисление координат следующего пикселя и значения Δ_{i+1} по формулам (1.76) и (1.80).
   5б. Если $\Delta_i > 0 \,\&\, \delta_+ > 0$, то используются формулы (1.77) и (1.81).
   5в. В противном случае используются формулы (1.78) и (1.82).
6. Переход к следующей итерации: $i := i + 1$, возврат к шагу 2.

Таким образом, для растеризации окружностей может применяться подход, использующий только операции сложения, вычитания и сравнения. Зачастую этот алгоритм применяется для отрисовки дуги не в отдельном квадранте, а в октантном угле, ведь окружность с центром в начале координат симметрична также относительно прямых $y = x$ и $y = -x$. Например, могут вычисляться пиксели искомой растеризации, расположенные во II октантном угле, при этом условие из шага 2 меняется на $y_i < x_i$, а на шаге 3 необходимо вычислять координаты 8 пикселей из всех октантных углов.

Аналогично может быть выведен алгоритм растеризации эллипса. Рассмотрим некоторый эллипс с центром в начале координат и с целочисленными значениями длин полуосей, которые направлены параллельно координатным осям. Как известно, эта линия может быть выражена следующим уравнением:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

которое легко привести к виду, не использующему операцию деления: $b^2x^2 + a^2y^2 = a^2b^2$. Используя рассуждения, аналогичные приведённым выше, можно вывести следующий **алгоритм растеризации эллипса**:

1. Инициализация переменных: $i := 1,\ x_1 = 0,\ y_1 = b,\ \Delta_1 = a^2 + b^2 - 2a^2b$.

2. Если $y_i < 0$, то алгоритм заканчивает работу и возвращает все пиксели, ранее записанные в результат.
3. Пиксели с координатами (x_i, y_i), $(-x_i, y_i)$, $(-x_i, -y_i)$, $(x_i, -y_i)$ из I, II, III и IV квадрантов записываются в результат.
4. Вычисляются значения δ_- и δ_+ по следующим формулам:

$$\delta_- = 2\Delta_i + a^2(2y_i - 1), \tag{1.83}$$

$$\delta_+ = 2\Delta_i - b^2(2x_i + 1). \tag{1.84}$$

5. В зависимости от знака значений Δ_i, δ_- и δ_+ вычисляются координаты следующего пикселя и значения Δ_{i+1}:

$$\Delta_i < 0 \,\&\, \delta_- \le 0 \Rightarrow \begin{cases} x_{i+1} = x_i + 1, \\ y_{i+1} = y_i, \\ \Delta_{i+1} = \Delta_i + b^2(2x_{i+1} + 1), \end{cases} \tag{1.85}$$

$$\Delta_i > 0 \,\&\, \delta_+ > 0 \Rightarrow \begin{cases} x_{i+1} = x_i, \\ y_{i+1} = y_i - 1, \\ \Delta_{i+1} = \Delta_i + a^2(-2y_{i+1} + 1), \end{cases}$$

$$(\Delta_i \le 0 \,\&\, \delta_- > 0) \vee (\Delta_i \ge 0 \,\&\, \delta_+ \le 0) \Rightarrow \begin{cases} x_{i+1} = x_i + 1, \\ y_{i+1} = y_i - 1, \\ \Delta_{i+1} = \Delta_i + b^2(2x_{i+1} + 1) + a^2(-2y_{i+1} + 1). \end{cases} \tag{1.86}$$

6. Переход к следующей итерации: $i := i + 1$, возврат к шагу 2.

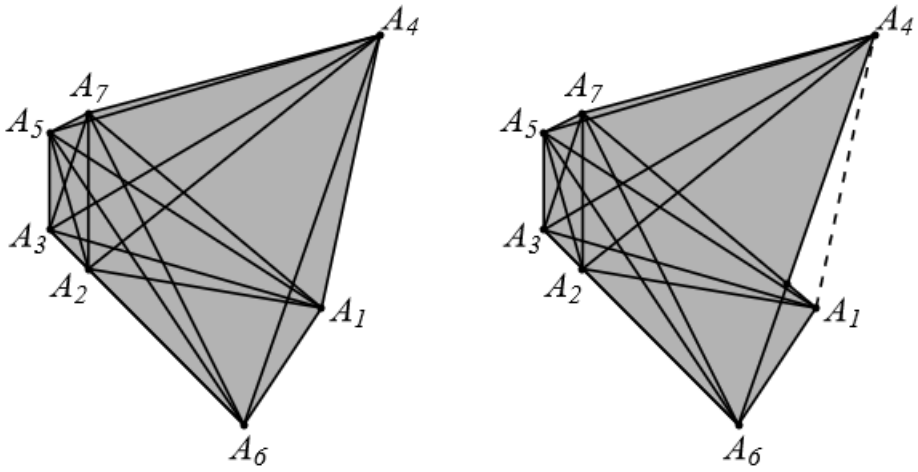
Заметим, что этот алгоритм можно использовать и для растеризации окружностей, положив $a = b = R$. Однако по сравнению с оригинальным алгоритмом Брезенхема здесь будут иметь место ненужные домножения на R^2 вспомогательных переменных Δ_i.

1.4. Алгоритмы построения выпуклой оболочки множества точек на плоскости

Выпуклой оболочкой некоторого геометрического множества Ω называется выпуклое множество S, содержащее множество Ω. **Минимальной выпуклой оболочкой** множества Ω называется такая его выпуклая оболочка

$S_{\min}$, что для всякой выпуклой оболочки S выполняется $S_{\min} \subseteq S$. Часто под выпуклой оболочкой понимается именно минимальная выпуклая оболочка.

Пусть $\Omega = \{A_1, A_2, ..., A_n\}$ – некоторое множество точек на плоскости, не лежащих на одной прямой. По определению всякое выпуклое множество S, содержащее все эти точки, должно также содержать любой отрезок A_iA_j, $i, j = \overline{1,n}$. Все такие отрезки ограничивают некоторый полигон, который, несложно видеть, является выпуклым (Рисунок 1.76а). Действительно, если предположить, что он является невыпуклым, то тогда найдётся некоторый отрезок A_iA_j, не содержащийся в нём (Рисунок 1.76б). Таким образом, минимальной выпуклой оболочкой множества точек на плоскости является выпуклый полигон с вершинами в некоторых из этих точек, содержащий все остальные точки.



а б

Рисунок 1.76 – Выпуклая оболочка множества точек является некоторым выпуклым полигоном (а). Всякий невыпуклый полигон с вершинами в этих точках можно дополнить до выпуклой оболочки (б)

1.4.1. Алгоритм Джарвиса

Алгоритм Джарвиса, или **алгоритм заворачивания подарка** [26], заключается в нахождении границ искомой выпуклой оболочки путём поворота ограничивающих прямых до тех пор, пока не будет достигнута некоторая точка из множества Ω. Пусть нужно найти выпуклую оболочку, причём определить последовательность её вершин, обеспечивающих левосторонний обход. Тогда нужно выполнить следующую последовательность шагов:

1. Выбирается какая-либо крайняя точка множества Ω, например, самая левая (если таковых несколько, то можно взять, например, самую нижнюю из них). Эта точка заведомо является вершиной искомой выпуклой оболочки. Обозначим её через P_1.

2. Среди векторов $\overrightarrow{P_1A_i}$, $A_i \neq P_1$, нужно найти вектор $\overrightarrow{P_1P_2}$, являющийся последним при обходе этих векторов по часовой стрелке. Такой вектор должен удовлетворять неравенству $\angle\left(\overrightarrow{P_1P_2}, \overrightarrow{P_1A_i}\right) \geq 0$ или, что то же самое, $\angle\left(\overrightarrow{P_1A_i}, \overrightarrow{P_1P_2}\right) \leq 0$. Если таких крайних векторов оказывается несколько, то необходимо выбрать вектор наибольшей длины. Найденный вектор $\overrightarrow{P_1P_2}$ определяет первое ребро искомого полигона.
3. Шаг 2 повторяется для векторов $\overrightarrow{P_2A_i}$, $A_i \neq P_1$, $A_i \neq P_2$.Новую найденную вершину обозначим за P_3.
4. Инициализация итерационной переменной: $i := 3$.
5. Для очередной вершины P_i, $i \geq 3$, шаг 2 выполняется для векторов $\overrightarrow{P_iA_j}$, $A_j \notin \{P_2, P_3, ..., P_i\}$. Нужно обратить внимание на то, что в отличие от шагов 2 и 3, здесь нужно рассматривать, в частности, вектор $\overrightarrow{P_iP_1}$, так как ломаная, ограничивающая искомую выпуклую оболочку, **должна рано или поздно замкнуться**.
6. Если в результате шага 5 получили $P_{i+1} = P_1$, то алгоритм завершает свою работу, возвращая полигон $P_1P_2...P_iP_1$ в качестве результата. В противном случае – переход к следующей вершине: $i := i + 1$, возврат к шагу 5.

Докажем, что данный алгоритм действительно возвращает минимальную выпуклую оболочку множества точек Ω. ◆ Для этого необходимо и достаточно показать, что полигон $P_1P_2...P_mP_1$, полученный в результате алгоритма Джарвиса, является выпуклым, и что все точки из Ω являются либо его вершинами, либо граничными, либо внутренними.

Для выпуклости полигона $P_1P_2...P_mP_1$, в свою очередь, необходимо и достаточно выполнения двух свойств: поворота налево в каждой вершине при обходе по периметру полигона, а также самонепересекаемость полигона. Первое напрямую следует из шага 2 представленного выше алгоритма: для всякой вершины P_i, $i = 2, 3, \ldots, m, 1$, выполняются неравенства $\angle\left(\overrightarrow{P_{i-1}P_i}, \overrightarrow{P_{i-1}P_{i+1}}\right) \geq 0 \Rightarrow \angle\left(\overrightarrow{P_{i-1}P_i}, \overrightarrow{P_iP_{i+1}}\right) \geq 0$. Доказывая же второе свойство, предположим противное: пусть имеются два пересекающихся ребра P_iP_{i+1} и P_jP_{j+1}. Тогда одна из вершин P_j и P_{j+1} лежит по левую сторону от оси P_iP_{i+1}, другая – по правую, и отсюда имеем, что должно выполняться одно из неравенств $\angle\left(\overrightarrow{P_iP_{i+1}}, \overrightarrow{P_iP_j}\right) < 0$ либо $\angle\left(\overrightarrow{P_iP_{i+1}}, \overrightarrow{P_iP_{j+1}}\right) < 0$, что противоречит выбору вершины P_{i+1}. Теперь предположим, что ребро P_jP_{j+1} не пересекает ребро P_iP_{i+1}, а примыкает к нему. Тогда следующая точка, P_{j+2}, должна располагаться по левую сторону от обеих осей P_iP_{i+1} и P_jP_{j+1} (Рисунок 1.77а). Рассматривая

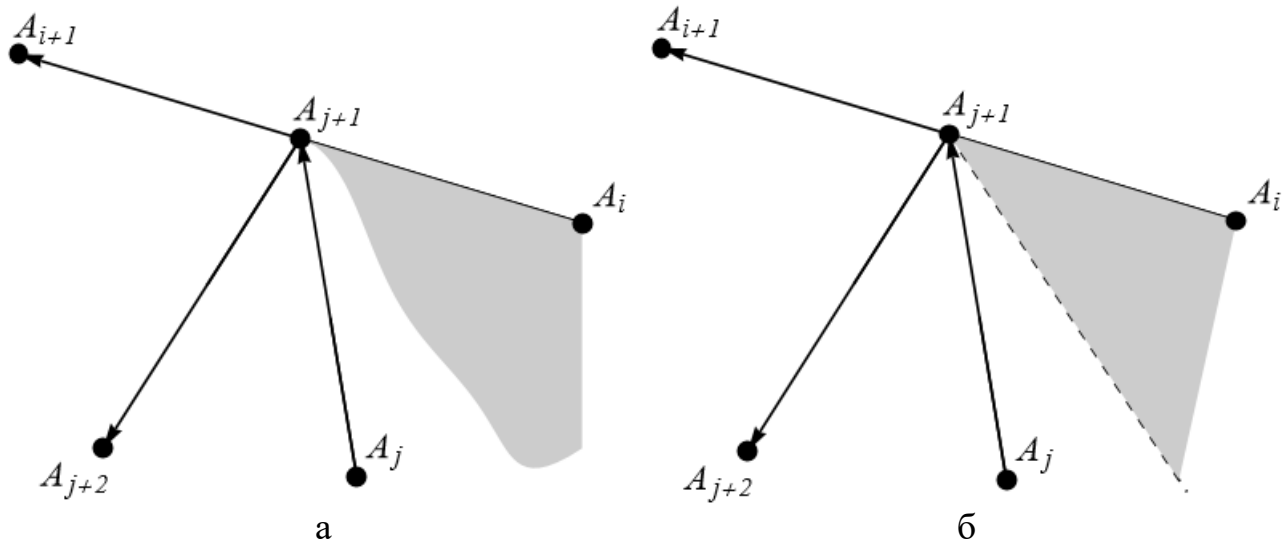


Рисунок 1.77 – Предположение о примыкании рёбер искомого полигона (а) приводит к необходимости построения дополнительного ребра (б)

теперь угол $\angle P_i P_{j+1} P_j$, заметим, что некоторая его часть является внутренней по отношению к искомому полигону (так как угол расположен по левую сторону от оси $P_i P_{i+1}$), а оставшаяся часть – внешней (ведь угол расположен по правую сторону от оси $P_j P_{j+1}$). Более того, данное высказывание справедливо для части угла $\angle P_i P_{j+1} P_j$, лежащей в сколь угодно малой окрестности его вершины P_{j+1}. Следовательно, существует по крайней мере ещё одно ребро внутри этого угла, выходящее из точки P_{j+1} (Рисунок 1.77б). Таким образом, получили, что от точки P_{j+1} проведено три ребра, что, нетрудно видеть, является невозможным по алгоритму Джарвиса.

Из всего этого следует, что два ребра полигона либо являются смежными, либо не имеют общих точек, т.е. искомый полигон является самонепересекающимся. Таким образом, имеем, что полигон $P_1 P_2 ... P_m P_1$ является самонепересекающимся, и что у каждой вершины при его обходе происходит поворот налево, что и означает его выпуклость.

Что касается расположения точек относительно полигона, то некоторая точка A_j может оказаться для него граничной при условии $\overrightarrow{P_i A_j} \uparrow\uparrow \overrightarrow{P_i P_{i+1}}$ для некоторого i, что выясняется на шагах 2, 3 или 5. Рассмотрим теперь случай, когда точка A_j не является ни вершиной полигона $P_1 P_2 ... P_m P_1$, ни его граничной точкой. Тогда получается, что исходя из шага 2 должно выполняться неравенство $\angle\left(\overrightarrow{P_1 P_2}, \overrightarrow{P_1 A_j}\right) > 0$, из шага 3 имеем $\angle\left(\overrightarrow{P_2 P_3}, \overrightarrow{P_2 A_j}\right) > 0$, а из шага 5 получаем аналогичные неравенства для каждого ребра: $\angle\left(\overrightarrow{P_i P_{i+1}}, \overrightarrow{P_i A_j}\right) > 0$. Другими словами, выполняется критерий принадлежности точки A_j выпуклому полигону $P_1 P_2 ... P_m P_1$. Таким образом, каждая точка из Ω является внутренней либо граничной точкой для полигона $P_1 P_2 ... P_m P_1$, либо его вершиной. ■

Замечание 1. В шаге 1 в качестве первой точки можно взять также самую нижнюю, правую или верхнюю. Алгоритм при этом никак не поменяется.

Замечание 2. Если нужно найти полигон с правосторонним обходом, то подчёркнутые слова и неравенства нужно заменить противоположными по смыслу.

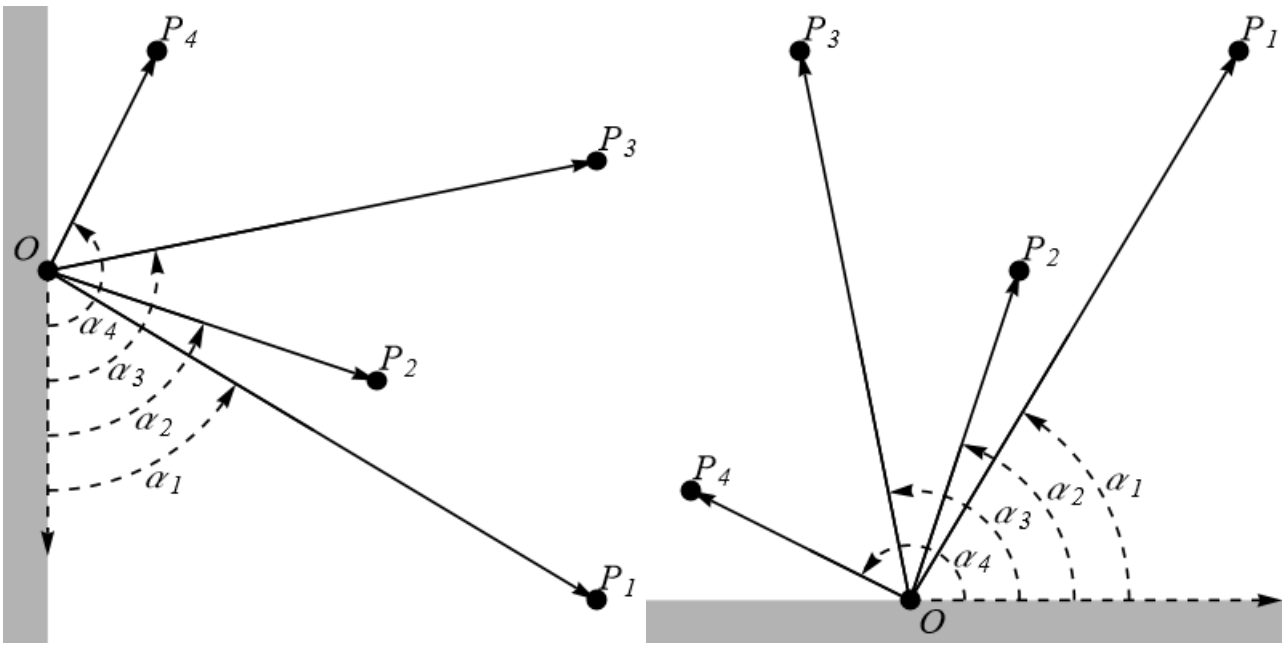
Замечание 3. В худшем случае на шаге 4 множество точек, куда не нужно проводить векторы $\overrightarrow{P_iA_j}$, будет пополняться точками P_2, P_3, …, P_i, …, P_n, и останется только точка P_1. Таким образом, в случае, когда выпуклой оболочкой множества Ω является полигон с вершинами во всех точках, выполняется наибольшее число операций. Его можно оценить асимптотикой $O(n^2)$, так как на первой итерации происходит нахождение крайнего среди $n-1$ вектора, на второй – среди $n-2$ векторов, на третьей – также среди $n-2$ векторов, …, на $(n-1)$-ой остаётся только один вектор, и нужно выполнить $1 + 2 + \ldots + (n-3) + (n-2) + (n-3) = n-3+\frac{(n-1)(n-2)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n - 2 = O(n^2)$. Остальные операции не ухудшают этой асимптотики.

Замечание 4. Для хранения последовательности искомых вершин можно использовать **массив**, т.к., во-первых, вершины добавляются последовательно, во-вторых, найденные вершины являются окончательными и не подлежат пересмотру или удалению (в отличие от других алгоритмов).

1.4.2. Алгоритм Грэхема

Алгоритм, предложенный Р. Л. Грэмом в 1972 году [27], позволяет строить выпуклую оболочку исходя из сортировки векторов, проведённых из некоторой точки ко всем остальным, и последующего рассмотрения этих точек в нужном порядке. Он описывается следующей последовательностью шагов:

1. Как и в алгоритме Джарвиса, выбирается крайняя точка множества Ω. Эта точка заведомо является вершиной искомой выпуклой оболочки. Обозначим её O.
2. Из вершины O проводятся векторы ко всем остальным точкам. Эти векторы сортируются в порядке прохождения через них заметающего луча с началом в точке O против часовой стрелки. Такой порядок существует и является единственным, так как эта сортировка сводится к сортировке по возрастанию величин α_i углов между начальным положением заметающего луча и этими векторами (на рисунке 1.78 это начальное положение указано пунктирной стрелкой). Сортировать векторы можно, например, одним из способов, представленных в подразделе 2.4.2. В результате получается последовательность



а б

Рисунок 1.78 – Сортировка векторов и точек в зависимости от выбора крайней точки: а – крайняя левая точка, б – крайняя нижняя точка

векторов $\overrightarrow{OP_1}$, $\overrightarrow{OP_2}$, …, $\overrightarrow{OP_k}$, удовлетворяющих следующим свойствам:

- $\forall i = \overline{1,k-1}: \underline{\operatorname{sgn} \angle(\overrightarrow{OP_i}, \overrightarrow{OP_{i+1}}) > 0}$;

- для всякого $i = \overline{1,k-1}$ внутри угла $\angle P_i O P_{i+1}$ не расположены никакие точки из Ω;

- среди векторов $\overrightarrow{OP_i}$, $i = \overline{1,k}$, нет коллинеарных. Если на шаге 2 сравниваются два коллинеарных вектора, то более короткий из них отбрасывается, а конец этого вектора является либо граничной, либо внутренней точкой искомой выпуклой оболочки.

3. Среди точек P_1, P_2, …, P_k, вообще говоря, могут оказаться внутренние либо граничные точки. Следовательно, список этих вершин подлежит редактированию. Это осуществляется следующим образом:
   - 3а. Начинается обработка списка с точки с индексом $i := 2$.
   - 3б. Если выполняется равенство $i = k$, то это означает, что обход списка закончен, и происходит переход к шагу 4.
   - 3в. Вычисляется знак угла $\angle(\overrightarrow{P_{i-1}P_i}, \overrightarrow{P_iP_{i+1}})$. Если имеет место неравенство $\underline{\operatorname{sgn} \angle(\overrightarrow{P_{i-1}P_i}, \overrightarrow{P_iP_{i+1}}) > 0}$, тогда происходит переход к следующей точке: $i := i + 1$. В противном же случае следует исключить точку P_i из списка вершин выпуклой оболочки. При этом число вершин уменьшается: $k := k - 1$. Кроме того, нужно вернуться к предыдущей вершине: $i := i - 1$, так как **направление поворота в ней может измениться**.

3г. Если оказывается верным равенство $i = 1$, то необходимо перейти к следующей вершине: $i := 2$, так как точка P_1 заведомо является вершиной искомой выпуклой оболочки. Возврат к шагу 3б.

4. После завершения обхода в новом списке P_1, P_2, …, P_k остались только точки – вершины искомой выпуклой оболочки. Этот список необходимо дополнить вершиной O, и получится выпуклая оболочка $OP_1P_2...P_kO$ с левосторонним обходом.

Докажем теперь корректность приведённого алгоритма. ◆ Как и при доказательстве корректности алгоритма Джарвиса, нужно показать, что выполняются три свойства:

- В каждой вершине имеет место поворот в одну и ту же сторону при обходе полигона. Для вершин P_i, $i = \overline{1,k}$, это выполняется в силу шага 3в. Предположим теперь, что $\operatorname{sgn}\angle(\overrightarrow{P_kO}, \overrightarrow{OP_1}) \le 0$. Тогда величину внутреннего угла $\angle P_1OP_k$ можно вычислить как сумму абсолютных величин углов $\sum_{i=1}^{k-1} \left|\angle(\overrightarrow{OP_i}, \overrightarrow{OP_{i+1}})\right| = \alpha_k - \alpha_1$. С другой стороны, получается, что $\operatorname{sgn}\angle(\overrightarrow{OP_1}, \overrightarrow{OP_k}) = -\operatorname{sgn}\angle(\overrightarrow{OP_k}, \overrightarrow{OP_1}) = \operatorname{sgn}\angle(\overrightarrow{P_kO}, \overrightarrow{OP_1}) \le 0$ согласно ранее выдвинутому предположению. Всё это означает, что угол $\angle P_1OP_k$ искомого полигона больше или равен развёрнутому. Однако вместе с этим угол величиной $\alpha_k - \alpha_1$ не может быть больше π, так как все точки P_i расположены в развёрнутом угле с центром в крайней точке O (см. рисунок 1.78). Таким образом, имеем противоречие, т.е. на самом деле имеет место неравенство $\operatorname{sgn}\angle(\overrightarrow{P_kO}, \overrightarrow{OP_1}) > 0$.

- Полигон $OP_1P_2...P_kO$ является самонепересекающимся. Прежде всего, все точки из Ω расположены внутри угла $\angle P_1OP_k$, следовательно, рёбра OP_1 и P_kO не содержат самопересечений найденного полигона. Остальные рёбра тоже не пересекаются друг с другом, так как в каждом углу $\angle P_iOP_{i+1}$, $i = \overline{1,k-1}$, нетрудно видеть, расположено ровно одно ребро – P_iP_{i+1}.

- Все точки из Ω либо являются вершинами полигона $OP_1P_2...P_kO$, либо расположены на его границе или внутри. Так, на шаге 2 представленного выше алгоритма могут быть выявлены точки, лежащие непосредственно на диагоналях OP_i, $i = \overline{2,k-1}$, или на рёбрах OP_1 и OP_k. Кроме того, точки, отбрасываемые на шаге 3в, являются граничными либо внутренними. Чтобы показать это, докажем, что при $\operatorname{sgn}\angle(\overrightarrow{P_{i-1}P_i}, \overrightarrow{P_iP_{i+1}}) \le 0$ отбрасываемая точка P_i лежит внутри или на границе треугольника $OP_{i-1}P_{i+1}$. Действительно, с учётом того, что выполняются также неравенства $\operatorname{sgn}\angle(\overrightarrow{OP_{i-1}}, \overrightarrow{OP_i}) > 0$ и $\operatorname{sgn}\angle(\overrightarrow{OP_i}, \overrightarrow{OP_{i+1}}) > 0$, имеем следующее (на рисунке 1.79 углы с известными

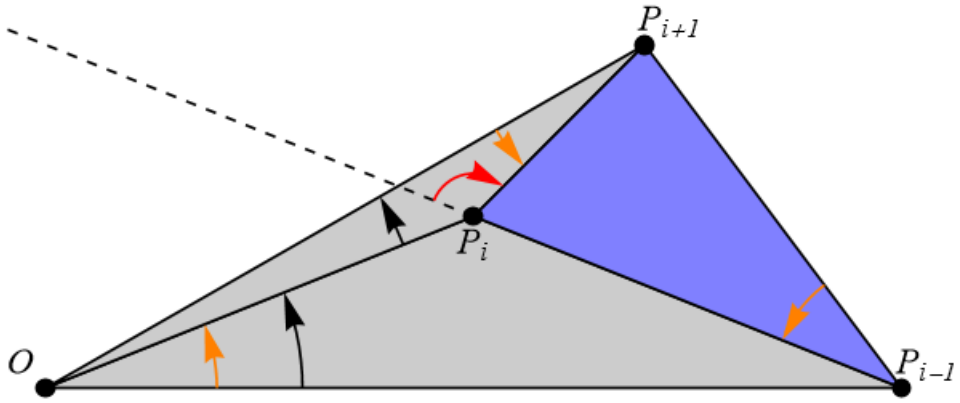


Рисунок 1.79 – Увеличение искомого полигона при отрицательном угле поворота в очередной вершине

знаками отмечены красной и чёрными стрелками, а углы, знаки которых нужно вычислить для проверки критерия принадлежности точки P_i треугольнику $OP_{i-1}P_{i+1}$ – тёмно-оранжевым):

$$\begin{gathered}\underline{\operatorname{sgn}\angle(\overrightarrow{OP_{i-1}},\overrightarrow{OP_i})>0},\\ \operatorname{sgn}\angle(\overrightarrow{P_{i-1}P_{i+1}},\overrightarrow{P_{i-1}P_i})=[\overrightarrow{P_{i-1}P_i}=\vec{a},\overrightarrow{P_iP_{i+1}}=\vec{b}]=\operatorname{sgn}\angle(\vec{a}+\vec{b},\vec{a})=\\ =-\operatorname{sgn}\angle(\vec{a},\vec{a}+\vec{b})=\underline{-\operatorname{sgn}\angle(\vec{a},\vec{b})\ge 0},\\ \operatorname{sgn}\angle(\overrightarrow{P_{i+1}O},\overrightarrow{P_{i+1}P_i})=-\operatorname{sgn}\angle(\overrightarrow{OP_{i+1}},\overrightarrow{P_{i+1}P_i})=\\ =-\operatorname{sgn}\angle(\overrightarrow{OP_{i+1}},\overrightarrow{OP_i})=\underline{\operatorname{sgn}\angle(\overrightarrow{OP_i},\overrightarrow{OP_{i+1}})>0}.\end{gathered}\qquad(1.87)$$

Выполнение этих неравенств означает принадлежность точки P_i треугольнику $OP_{i-1}P_{i+1}$ (причём при равенстве $\operatorname{sgn}\angle(\overrightarrow{P_{i-1}P_i},\overrightarrow{P_iP_{i+1}})=0$ имеем $P_i\in$ $\in P_{i-1}P_{i+1}$). Значит, при $\underline{\operatorname{sgn}\angle(\overrightarrow{P_{i-1}P_i},\overrightarrow{P_iP_{i+1}})\le 0}$ на шаге 3в, по сути, происходит увеличение текущего полигона путём добавления к нему треугольника $P_{i-1}P_iP_{i+1}$ (синий треугольник на рисунке 1.79). Заметим также, что уменьшения текущего полигона не происходит ни на каком шаге алгоритма. Следовательно, из отношения $P_i\in\triangle OP_{i-1}P_{i+1}$ следует принадлежность P_i и всему искомому полигону. ■

Замечание 1. Выше приведён алгоритм Грэхема для нахождения выпуклой оболочки с левосторонним обходом. Для получения правостороннего обхода подчёркнутые слова и неравенства нужно заменить противоположными по смыслу.

Замечание 2. Время работы алгоритма Грэхема оценивается асимптотикой $O(n\log n)$. ◆ Для доказательства этого факта оценим временную асимптотику каждого шага этого алгоритма. Шаг 1 – нахождение точки с экстремальной абсциссой либо ординатой – выполняется за время $O(n)$. Далее, на шаге 2, имеет место сортировка $n-1$ вектора, которая при использовании подходящих

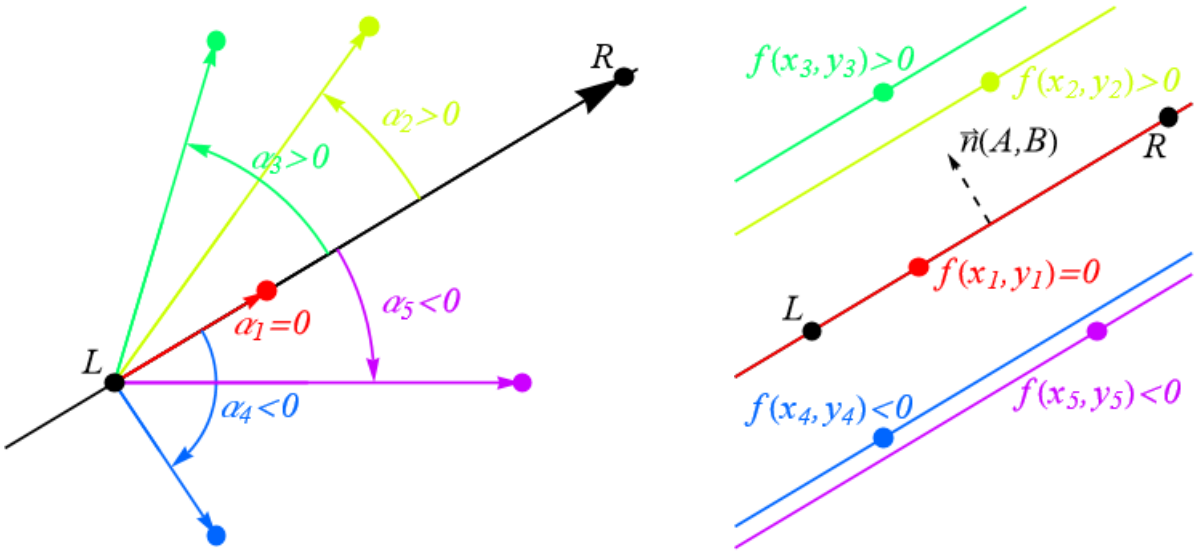
структур данных и алгоритмов выполняется за время $O(n \log n)$. Рассматривая же теперь шаг 3, заметим, что каждая точка P_i может быть удалена не более одного раза, после чего шаг 3в выполняется ещё один раз (так как надо вернуться к предыдущей точке). Следовательно, шаг 3в выполняется не больше $2k$ раз, т.е. имеет место асимптотика $O(2k) = O(k) = O(n)$. Таким образом, алгоритм Грэхема работает за линейно-логарифмическое время, причём худшим случаем является ситуация, когда искомая выпуклая оболочка является треугольником. ■

Замечание 3. Для формирования списка искомых вершин вместо редактируемого списка может использоваться **стек**. При этом в данном стеке должна быть реализована операция просмотра *двух* последних элементов, чтобы осуществлять проверку, аналогичную шагу 3в. После удаления всех элементов стека, которые дают не тот знак угла поворота, добавляется текущая вершина из списка, полученного на шаге 2, и происходит переход к следующей.

1.4.3. Алгоритм Эндрю

Ещё один алгоритм построения выпуклой оболочки точек был предложен А. М. Эндрю в 1979 году и в целом опирается на те же идеи, что и алгоритм Грэхема [28]. Разница заключается в другом подходе к сортировке точек исходного множества Ω. Состоит данный алгоритм из следующих шагов:

1. Выбираются две крайние точки: самая левая и самая правая. Если самых левых и/или самых правых точек оказывается несколько, то из левых точек обычно выбирают самую нижнюю, среди правых – самую верхнюю. Обозначим выбранные левую и правую точки соответственно через L и R.
2. Определяется расположение остальных точек относительно прямой LR. Это можно осуществить множеством различных способов, например, через углы между векторами $\overrightarrow{LR}$ и $\overrightarrow{LA_i}$ либо исходя из общего уравнения прямой LR (Рисунок 1.80). Точки, оказавшиеся лежащими на прямой LR, дальше не рассматриваются – это внутренние точки искомой выпуклой оболочки.
3. Если точек в верхней полуплоскости не оказалось, то происходит переход к шагу 7. При этом список вершин выпуклой оболочки точек верхней полуплоскости является пустым: $H_{\text{top}} := \varnothing$.
4. Рассматривается верхняя полуплоскость относительно прямой LR. При построении левосторонней выпуклой оболочки точки, оказавшиеся в результате шага 2 лежащими в этой полуплоскости, сортируются в порядке убывания их абсцисс. При равенстве абсцисс у нескольких точек выбирается точка с максимальной ординатой, а



а б

Рисунок 1.80 – Определение положения точек относительно прямой: а – через углы между векторами, б – через функцию $f(x, y) = Ax + By + C$, которой задаётся прямая LR

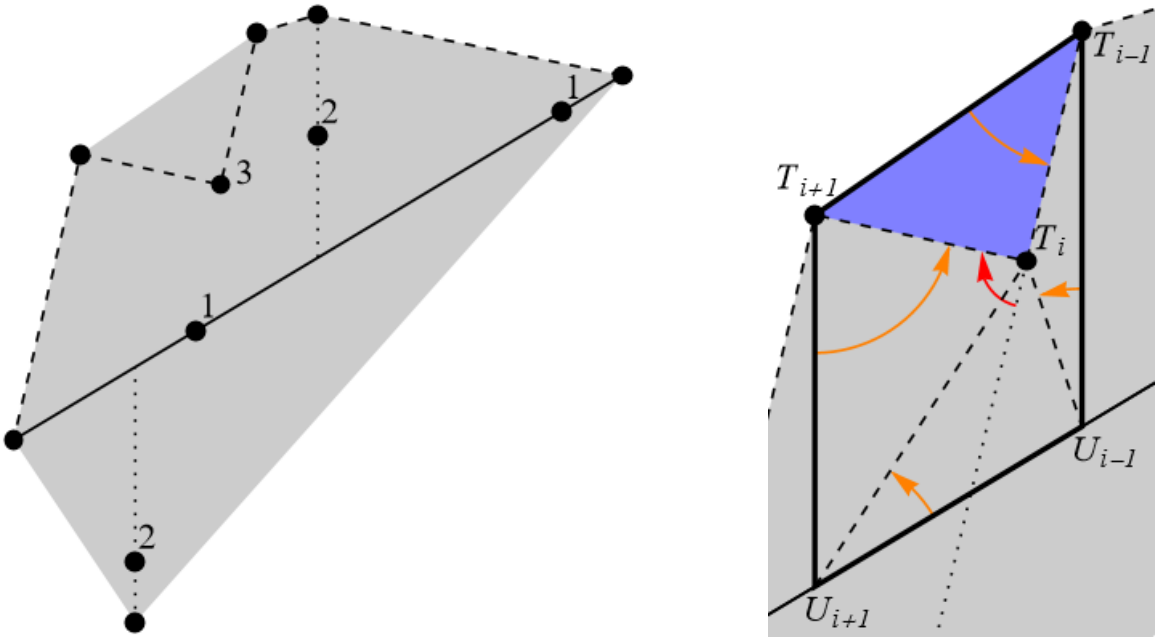
остальные отбрасываются. Полученную последовательность точек обозначим $T_0 = R,\ T_1,\ T_2,\ \ldots,\ T_k,\ T_{k+1} = L$.

5. Среди этих точек вершинами искомой выпуклой оболочки заведомо являются точки L, R и по крайней мере одна точка из множества $\{T_i\}_{i=1}^{k}$. Обработка этого списка происходит аналогично алгоритму Грэхема:
   - 5а. Начинается обработка списка с точки с индексом $i := 1$.
   - 5б. Если выполняется равенство $i = k + 1$, то это означает, что обход списка закончен, и происходит переход к шагу 6.
   - 5в. Вычисляется знак угла $\angle(\overrightarrow{T_{i-1}T_i}, \overrightarrow{T_iT_{i+1}})$. Если имеет место неравенство $\operatorname{sgn}\angle(\overrightarrow{T_{i-1}T_i}, \overrightarrow{T_iT_{i+1}}) > 0$, тогда происходит переход к следующей точке: $i := i + 1$. В противном же случае следует исключить точку T_i из списка вершин выпуклой оболочки точек верхней полуплоскости. При этом число вершин уменьшается: $k := k - 1$. Кроме того, нужно вернуться к предыдущей вершине: $i := i - 1$, так как **направление поворота в ней может измениться**.
   - 5г. Если оказывается верным равенство $k = 1$, то три оставшиеся точки являются вершинами выпуклой оболочки точек верхней полуплоскости, и происходит переход к шагу 6.
   - 5д. Если оказывается верным равенство $i = 0$, то необходимо перейти к следующей вершине: $i := 1$. Возврат к шагу 5б.
6. Оставшиеся точки образуют список вершин выпуклой оболочки для верхней полуплоскости: $H_{\text{top}} := \{T_1, T_2, \ldots, T_k, L\}$.
7. Нижняя полуплоскость рассматривается аналогично. Если точек в нижней полуплоскости не оказалось, то список вершин выпуклой

оболочки для неё полагается пустым: $H_{\text{bottom}} := \varnothing$, и происходит переход к шагу 11.

8. Точки в нижней полуплоскости сортируются в порядке возрастания их абсцисс (при равенстве абсцисс у некоторых из них остаётся только точка с минимальной ординатой), в результате чего получается последовательность $B_0 = L,\ B_1,\ B_2,\ \ldots,\ B_l,\ B_{l+1} = R$.
9. Список, полученный на предыдущем шаге, подлежит редактированию, аналогичному шагу 5:
   9а. Начинается обработка списка с точки с индексом $i := 1$.
   9б. Если выполняется равенство $i = l + 1$, то это означает, что обход списка закончен, и происходит переход к шагу 10.
   9в. Вычисляется знак угла $\angle(\overrightarrow{B_{i-1}B_i}, \overrightarrow{B_iB_{i+1}})$. Если имеет место неравенство $\operatorname{sgn}\angle(\overrightarrow{B_{i-1}B_i}, \overrightarrow{B_iB_{i+1}}) > 0$, тогда происходит переход к следующей точке: $i := i + 1$. В противном же случае следует исключить точку B_i из списка вершин выпуклой оболочки точек верхней полуплоскости. При этом число вершин уменьшается: $l := l - 1$. Кроме того, нужно вернуться к предыдущей вершине: $i := i - 1$, так как **направление поворота в ней может измениться**.
   9г. Если оказывается верным равенство $l = 1$, то три оставшиеся точки являются вершинами выпуклой оболочки точек нижней полуплоскости, и происходит переход к шагу 10.
   9д. Если оказывается верным равенство $i = 0$, то необходимо перейти к следующей вершине: $i := 1$. Возврат к шагу 9б.
10. После выполнения шага 9 получается список вершин выпуклой оболочки для нижней полуплоскости: $H_{\text{bottom}} := \{B_1, B_2, ..., B_l, R\}$.
11. Объединение списков H_{top} и H_{bottom} даёт искомый список вершин единой выпуклой оболочки исходного множества Ω. Если один из этих списков оказался пустым, то в искомый список добавляется та из крайних вершин L либо R, которая отсутствует в другом, непустом списке.

Доказательство корректности алгоритма Эндрю в целом повторяет доказательство для алгоритма Грэхема. ◆ Самонепересекаемость следует из того, что списки H_{top} и H_{bottom} отсортированы, а значит, точки рёбер, проведённые в одной из полуплоскостей относительно прямой LR, не могут иметь равных абсцисс (кроме их общих концов). Кроме того, точки, отброшенные на шагах 2, 4, 5в, 8, 9в, являются внутренними или граничными для искомой выпуклой оболочки (Рисунок 1.81а). Для некоторой точки T_i, отброшенной на шаге 5в, для доказательства её принадлежности полученной выпуклой оболочке достаточно рассмотреть четырёхугольник $T_{i-1}T_{i+1}U_{i+1}U_{i-1}$, где U_{i-1} и U_{i+1} – пересечения прямых, проведённых соответственно через точки T_{i-1}



а б

Рисунок 1.81 – Внутренние точки выпуклой оболочки, полученной по алгоритму Эндрю: а – точки на прямой *LR* (1), точки, отброшенные в результате сортировки (2), и точки, отброшенные при обходе отсортированных списков вершин (3); б –увеличение искомой выпуклой оболочки при отрицательном угле поворота в очередной вершине

и T_{i+1} перпендикулярно оси абсцисс, с прямой *LT* (Рисунок 1.81б). В силу сортировки на шаге 4 она расположена между прямыми $T_{i+1}U_{i+1}$ и $T_{i-1}U_{i-1}$, а также выше прямой *LT*. Кроме того, она лежит по левую сторону от оси $T_{i-1}T_{i+1}$, что доказывается по формуле, аналогичной (1.87). ■

Замечание 1. Выше приведён алгоритм Эндрю для построения выпуклой оболочки с левосторонним обходом. Для построения правосторонней оболочки подчёркнутые слова и неравенства следует заменить противоположными по смыслу, а в подчёркнутых равенствах точки *L* и *R* поменять местами.

Замечание 2. Временная асимптотика алгоритма Эндрю совпадает с алгоритмом Грэхема и равна $O(n \log n)$. Тем не менее, благодаря упрощённой сортировке точек, которой как раз и обусловлено линейно-логарифмическое время работы обоих алгоритмов, алгоритм Эндрю обычно работает несколько быстрее для достаточно большого числа точек *n* [29, sec. 3.1, pp. 5–7].

1.4.4. Алгоритм быстрой оболочки

Ещё один алгоритм был описан Дж. С. Гринфилдом в его статье [30] в 1990 году. Его суть заключается в том, что рекурсивно, добавлением шаг за шагом новых вершин находятся приближения выпуклой оболочки до тех пор, пока не получится полигон, содержащий все точки из исходного множества.

Одна из реализаций этого алгоритма предполагает его применение к двум полуплоскостям относительно некоторой секущей прямой. Полученные две выпуклые оболочки, как и в алгоритме Эндрю, объединяются, и получается единая выпуклая оболочка целого множества Ω. Проводится секущая прямая обычно через крайние точки Ω, например, через самую левую L и правую R. Таким образом, рассматривается некоторый отрезок и множество точек, расположенных от него по одну сторону. **Алгоритм построения выпуклой оболочки для отрезка $L'R'$ и** множества Ω' **точек, расположенных от него по правую сторону** (вспомогательный алгоритм), имеет следующий вид:

1. Если $\Omega' = \varnothing$, то алгоритм возвращает пустой список вершин.
2. Из множества Ω' выбирается наиболее отдалённая от отрезка точка. Расстояние от некоторой точки $A_i(x_i, y_i)$ до прямой $L'R'$ прямо пропорционально модулю векторного произведения векторов $\overrightarrow{L'R'}(x, y)$ и $\overrightarrow{L'A_i}(x_i', y_i')$, равному $\text{abs}\begin{vmatrix} x & y \\ x_i' & y_i' \end{vmatrix}$. Также это расстояние пропорционально величине $Ax_i + By_i + C$, где $Ax_i + By_i + C = 0$ – уравнение прямой $L'R'$. Найденную точку обозначим F.
3. После проведения рёбер $L'F$ и FR' множество Ω' разбивается на три подмножества: точки внутри $\triangle L'FR'$, точки по правую сторону от $L'F$ и точки по правую сторону от FR' (Рисунок 1.82). Внутренние точки, очевидно, являются внутренними точками искомой выпуклой оболочки. Два других множества обозначим через $\Omega_L := \varnothing$ и $\Omega_R := \varnothing$ соответственно. Процесс нахождения двух этих множеств формально можно описать последовательностью следующих шагов, применяемых к каждой точке A_i из Ω':
   - 3а. Если $\text{sgn}\angle(\overrightarrow{L'F}, \overrightarrow{L'A_i}) < 0$, то точка A_i является внешней относительно ребра $L'F$, и она добавляется во множество Ω_L.
   - 3б. В противном случае вычисляется знак угла $\angle(\overrightarrow{FR'}, \overrightarrow{FA_i})$. Если $\text{sgn}\angle(\overrightarrow{FR'}, \overrightarrow{FA_i}) < 0$, то точка A_i добавляется во множество Ω_R. Иначе эта точка просто пропускается.
4. Данный алгоритм применяется к ребру $L'F$ и множеству точек Ω_L, в результате чего получается список вершин H_L.
5. Этот же алгоритм применяется к ребру FR' и множеству точек Ω_R, что даёт список вершин H_R.
6. Алгоритм завершается возвратом упорядоченного списка $H_L \cup \{F\} \cup \cup H_R$.

В целом же **алгоритм выпуклой оболочки** (основной алгоритм) выглядит следующим образом:

1. Как и в алгоритме Эндрю, находятся крайняя левая точка L и крайняя правая точка R. Прямая LR разбивает множество Ω на точки верхней и

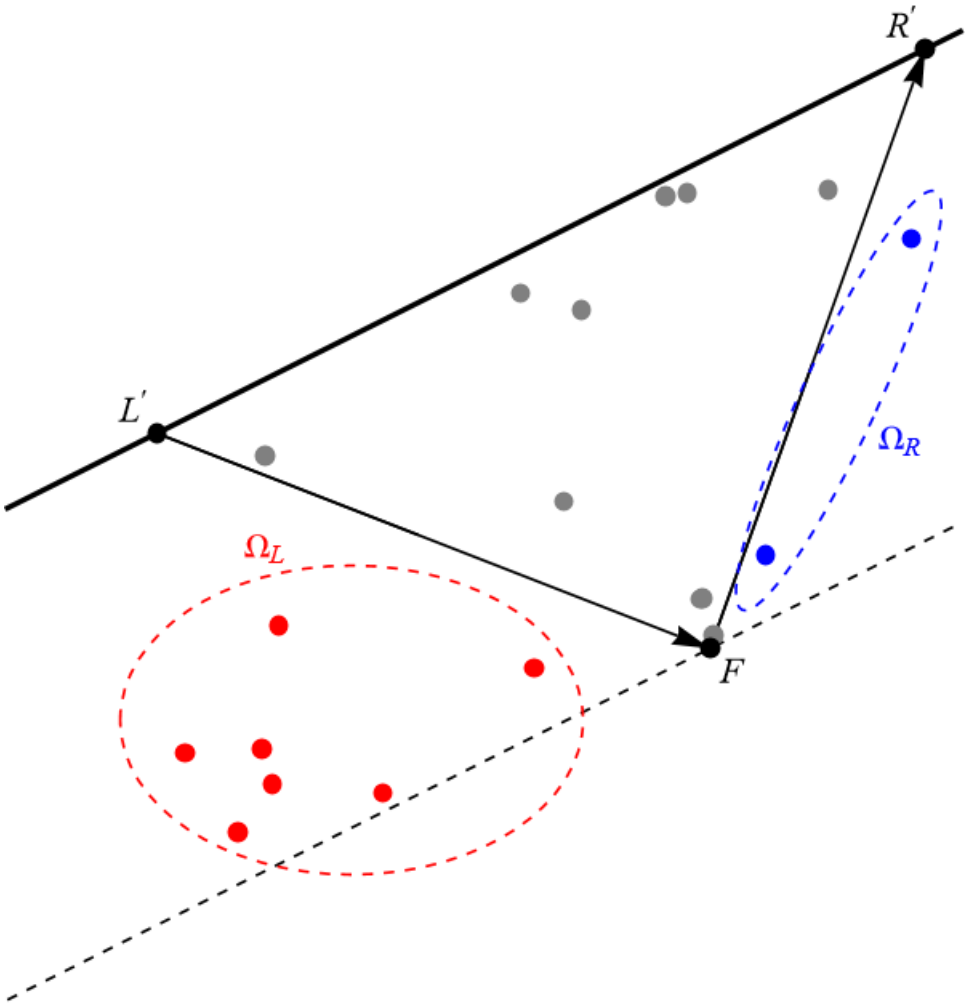


Рисунок 1.82 – Разбиение множества Ω' на подмножества внутренних и внешних точек

нижней полуплоскости, которые можно обозначить через Ω_T и Ω_B соответственно. Точки, оказавшиеся на самой прямой, далее не рассматриваются.

2. Первой вершиной искомой выпуклой оболочки объявляется точка L: $H := \{L\}$.
3. Вспомогательный алгоритм применяется к ребру LR и множеству Ω_B, находящемуся от него по правую сторону. Вершины полученного списка добавляются в конец списка H.
4. Затем в H добавляется вершина R.
5. Вспомогательный алгоритм применяется к ребру RL и другому множеству. Итоговый список добавляется к H.
6. Алгоритм завершается возвратом списка H, содержащего вершины выпуклой оболочки с левосторонним обходом.

Доказательство этого алгоритма проведём по той же схеме, что и для предыдущих. ◆ Вначале покажем самонепересекаемость полученного полигона. Для этого достаточно показать, что на шаге 3 вспомогательного алгоритма получаются непересекающиеся множества Ω_L и Ω_R. Действительно,

пересечение двух правых полуплоскостей относительно осей $L'F$ и FR' представляет собой угол, вертикальный для $\angle L'FR'$ (фиолетовый угол на рисунке 1.83). Однако этот вертикальный угол расположен по другую сторону от прямой Δ, параллельной $L'R'$ и проходящей через F. Прямая Δ разделяет полуплоскость по правую сторону от оси $L'R'$ на две части: точки между двумя прямыми расположены ближе к $L'R'$, чем F; точки же другой части (где и расположен этот угол) расположены дальше от прямой $L'R'$. Следовательно, в силу выбора точки F в этом углу не лежит ни одной точки из Ω'. Таким образом, рёбра, проведённые между точками внутри двух множеств, пересекаться не могут.

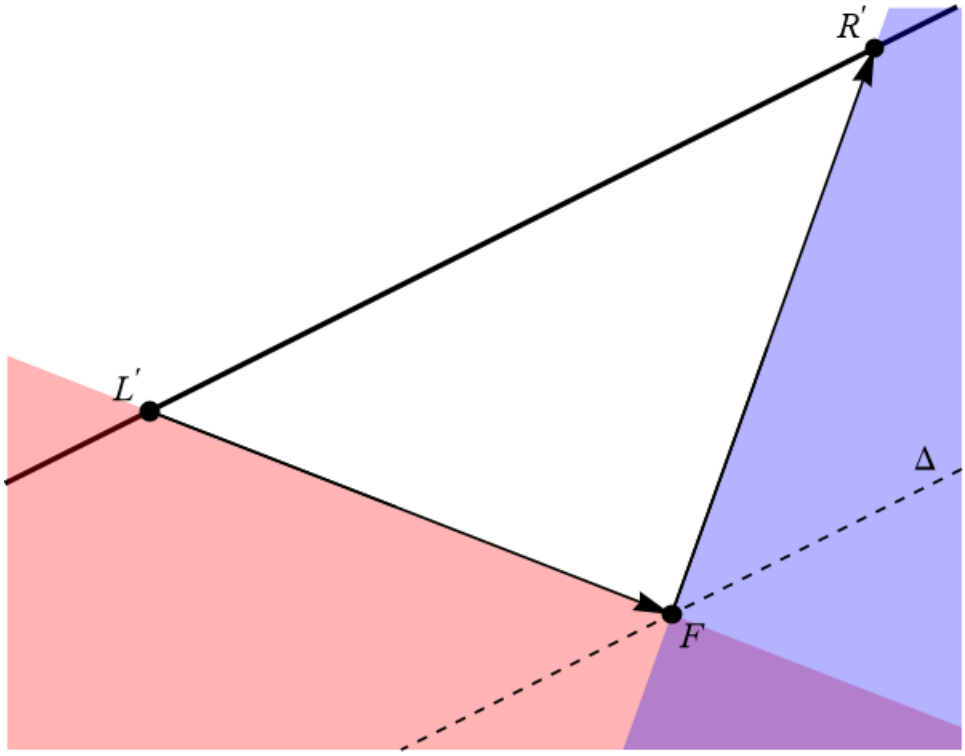


Рисунок 1.83 – Пересечение двух полуплоскостей, в которых расположены внешние точки

Кроме того, в каждой вершине имеет место поворот на положительный угол. Рассмотрим вершину F и две соседние вершины L'' и R'' (Рисунок 1.84). Тогда имеют место следующие равенства и неравенство:

$$\operatorname{sgn}\angle(\overrightarrow{FL''},\overrightarrow{FL'}) = -\operatorname{sgn}\angle(\overrightarrow{FL'},\overrightarrow{FL''}) = \operatorname{sgn}\angle(\overrightarrow{L'F},\overrightarrow{FL''}) = \underline{-1}, \qquad (1.88)$$

так как точка L'' расположена по правую сторону от ребра $L'F$;

$$\begin{aligned}\operatorname{sgn}\angle(\overrightarrow{FL'},\overrightarrow{FR'}) = \operatorname{sgn}\angle(\overrightarrow{FL'},\overrightarrow{L'R'}) = -\operatorname{sgn}\angle(\overrightarrow{L'R'},\overrightarrow{FL'}) = \\ = \operatorname{sgn}\angle(\overrightarrow{L'R'},\overrightarrow{L'F}) = \underline{-1},\end{aligned} \qquad (1.89)$$

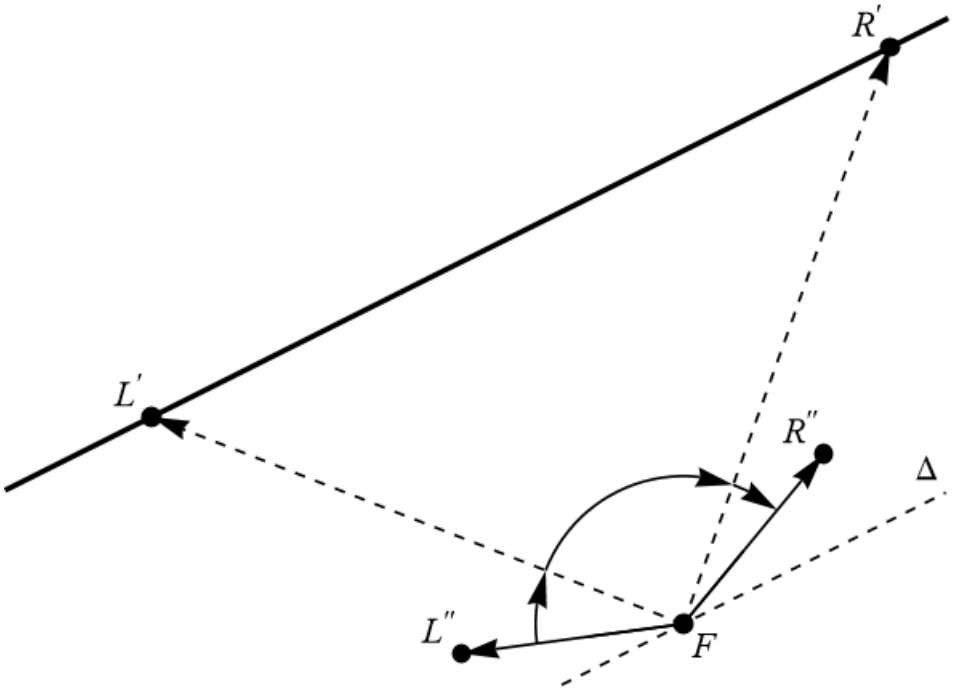


Рисунок 1.84 – Угол поворота при вершине F

так как точка F расположена по правую сторону от оси $L'R'$;

$$\angle(\overrightarrow{FR'}, \overrightarrow{FR''}) < 0$$

в силу расположения точки R'' по правую сторону от ребра FR'. По аналогии с алгоритмом Грэхема нетрудно видеть, что так как все три угла расположены внутри развёрнутого угла с вершиной в точке F и сторонами, образованными прямой Δ, то их сумма даст угол между крайними векторами без добавления полных углов: $\angle(\overrightarrow{FL''}, \overrightarrow{FL'}) + \angle(\overrightarrow{FL'}, \overrightarrow{FR'}) + \angle(\overrightarrow{FR'}, \overrightarrow{FR''}) = \angle(\overrightarrow{FL''}, \overrightarrow{FR''}) < 0$, что окончательно даёт $\angle(\overrightarrow{FL''}, \overrightarrow{FR''}) < 0 \Rightarrow \angle(\overrightarrow{L''F}, \overrightarrow{FR''}) > 0$.

Наконец, все точки, не попавшие в итоговый список H, являются внутренними либо граничными точками итоговой выпуклой оболочки. Точки, отброшенные на шаге 3б вспомогательного алгоритма, являются внутренними или граничными точками треугольника $L'FR'$, а значит, и найденного полигона. Кроме того, ясно, что после работы основного алгоритма внешних вершин не остаётся, так как после каждого вызова вспомогательного алгоритма число вершин в текущем множестве уменьшается по крайней мере на 1 (во вложенных вызовах вспомогательного алгоритма точка F уже не принадлежит множеству Ω'). ■

Замечание 1. Другое доказательство корректности этого алгоритма изложено самим автором [30, sec. 5] и состоит из трёх лемм: алгоритм возвращает все вершины выпуклой оболочки; все точки в искомом списке являются вершинами выпуклой оболочки; в списке приведены вершины в правильном порядке.

Замечание 2. Выше приведён алгоритм, в результате которого получается список вершин, обеспечивающий левосторонний обход искомого полигона. Для нахождения правосторонней выпуклой оболочки подчёркнутые слова и неравенства следует заменить противоположными по смыслу, в шаге 3 общего алгоритма множество Ω_B заменить на Ω_T, а в равенствах (1.88) и (1.89) отрицательную единицу заменить на положительную. Другая реализация алгоритма изложена в практическом разделе (подраздел 2.4.4).

Замечание 3. Сложность алгоритма в худшем случае составляет $O(n^2)$ из-за шага 2 вспомогательного алгоритма, где для каждой точки нужно вычислять величину, пропорциональную расстоянию до нового ребра. Эта асимптотика имеет место в ситуации, когда при каждом новом вызове вспомогательного алгоритма одно из множеств Ω_L либо Ω_R оказывается пустым, и при следующем вызове мощность множества Ω' уменьшается ровно на 1 (Рисунок 1.85). Впрочем, такое происходит крайне редко, поэтому среднее (ожидаемое) время выполнения алгоритма оценивается асимптотикой $O(n \log n)$ или даже $O(n)$ в ситуациях, когда число вершин выпуклой оболочки примерно равно n или же гораздо меньше него соответственно [30, sec. 6].

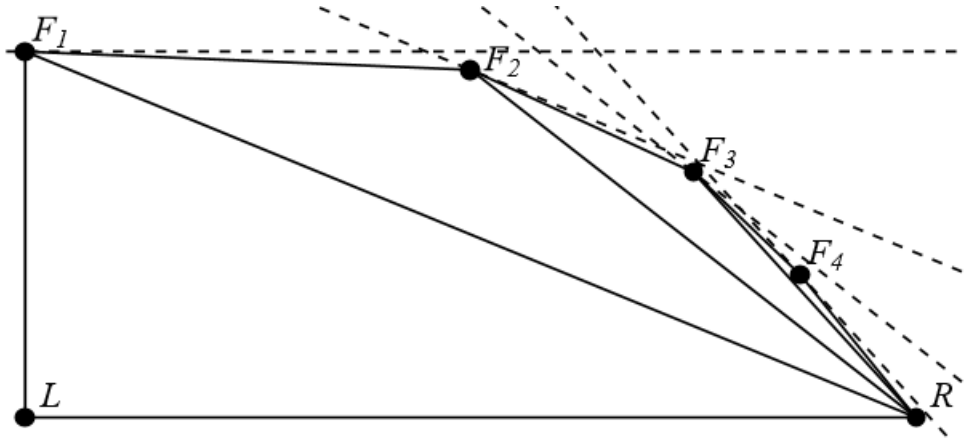


Рисунок 1.85 – Случай, который даёт худшую асимптотику

Замечание 4. Если в основном алгоритме задать дополнительный параметр – максимальное число вложенных вызовов вспомогательного алгоритма, то тогда можно получать неполные последовательности вершин, представляющие полигоны, которые, вообще говоря, не содержат некоторых точек из исходного множества Ω, однако могут использоваться как приближения искомой выпуклой оболочки (Рисунок 1.86). Другой способ построения приблизительной выпуклой оболочки приведён в [2, с. 57–59].

1.5. Триангуляция

В самом широком смысле в компьютерной графике и прочих приложениях геометрии под **триангуляцией** на плоскости понимается планарный граф (т.е.

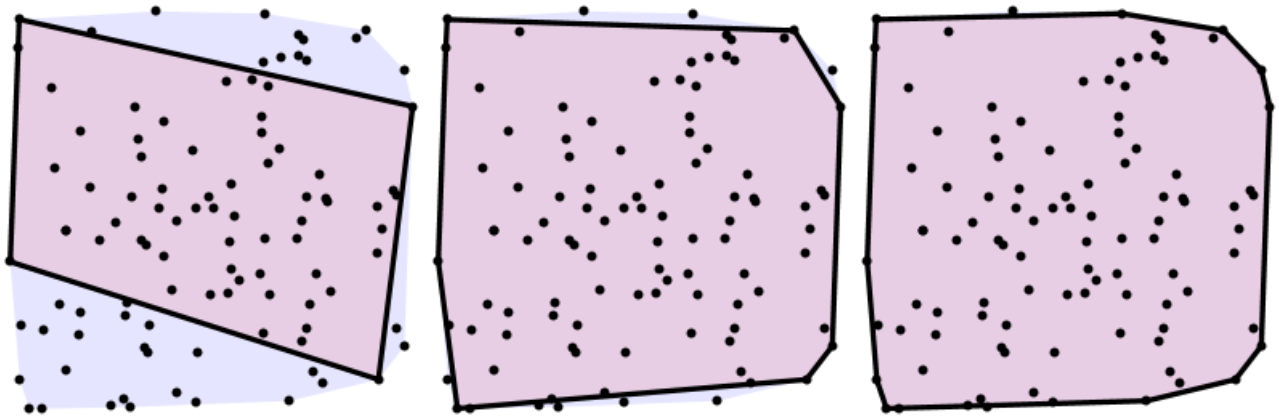


Рисунок 1.86 – Приближения выпуклой оболочки (синий полигон) промежуточными результатами (сиреневые полигоны)

такой, у которого рёбра, не имеющие общей вершины, не пересекаются), все внутренние области которого являются треугольниками [31, с. 7]. При таком определении также следует учитывать, что значимым является положение вершин, т.е. изоморфные графы, строго говоря, считаются разными триангуляциями. С точки зрения же геометрии триангуляция – это множество треугольников Δ_i, $i=\overline{1,T}$, пересечения которых $\Delta_i \cap \Delta_j$ при $i \neq j$ имеют нулевую площадь (т.е. разные треугольники могут иметь общее ребро, общую вершину, либо не пересекаться вовсе). Обычно постановка задачи триангуляции подразумевает, что вначале задано либо некоторое геометрическое множество, либо множество точек. Тогда говорят соответственно о **триангуляции геометрического множества** и **триангуляции множества точек**.

Триангуляция геометрического множества Ω на плоскости подразумевает его разбиение на треугольники Δ_i такие, что также выполняются следующие свойства:

- $\forall i : \Delta_i \subseteq \Omega$.
- $\bigcup_{i=1}^{T} \Delta_i \approx \Omega$.

В последнем свойстве полное совпадение триангуляции и исходного множества возможно только в случае, когда граница множества Ω состоит только из прямолинейных участков, т.е. Ω представляет собой полигон. В прочих же случаях имеет место, строго говоря, приближение множества Ω некоторым множеством треугольников.

Триангуляция множества точек – это такая триангуляция, при которой все вершины треугольников Δ_i расположены в этих точках.

Триангуляция имеет широкое применение в компьютерной графике [3, с. 114–115; 31, главы 9, 13]. Благодаря ей значительно упрощается решение, например, следующих задач:

1. Локализация точки относительно заданного множества, которая сводится к локализации точки относительно треугольников Δ_i: точка A является

внутренней или граничной для множества Ω тогда и только тогда, когда она является внутренней или граничной для некоторого из треугольников Δ_i.

2. Отсечение и удаление невидимых частей поверхностей, что сводится к перебору составляющих её треугольников и применению алгоритмов отсечения и удаления выпуклых полигонов.

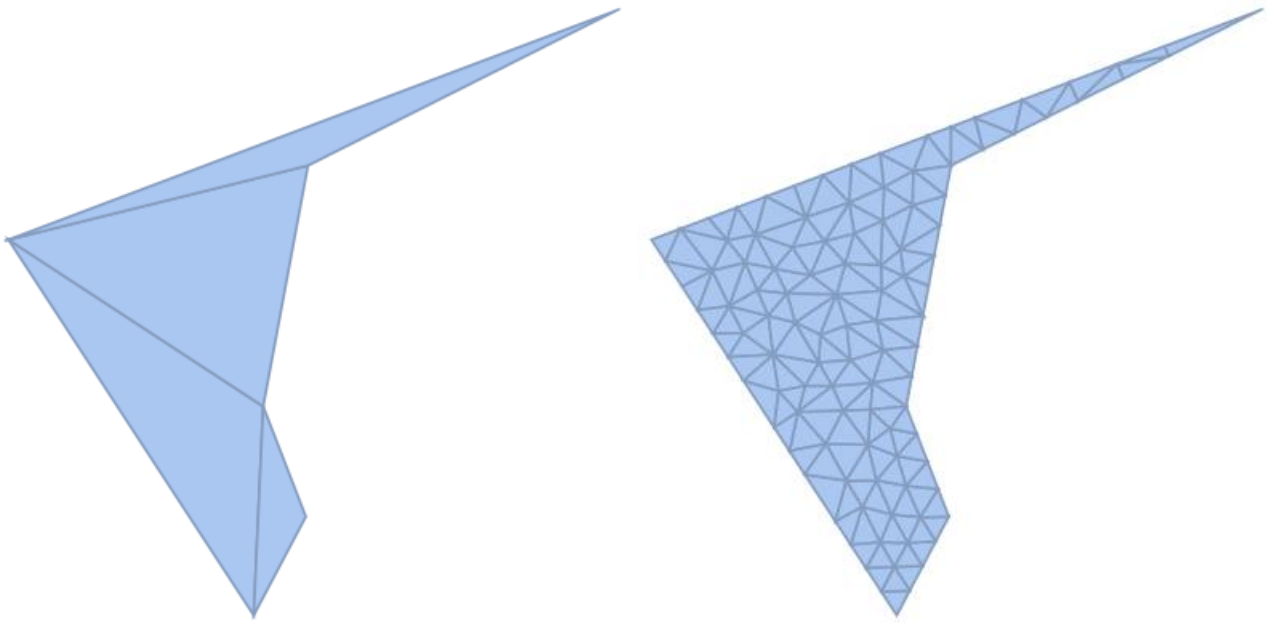
3. Значительная часть алгоритмов визуализации, в первую очередь окраски криволинейных поверхностей, которые можно путём триангуляции приблизить множеством треугольников. Так как три вершины треугольника однозначно задают плоскость (для которой вектор нормали является постоянным в любой её точке), то её закраска значительно упрощается.

4. Кроме того, благодаря триангуляции упрощается задача хранения геометрических объектов в памяти компьютера, что сводится к хранению информации о расположении вершин, рёбер треугольников и/или самих треугольников и связей между ними.

1.5.1. Триангуляция полигона диагоналями

Как отмечено выше, точная триангуляция некоторого геометрического множества на плоскости возможна только в случае, когда это множество является полигоном. Верно и обратное, т.е. всякий полигон можно разбить на некоторое конечное число треугольников, удовлетворяющее условиям выше. Ясно, что рёбра исходного полигона служат также сторонами для треугольников той или иной триангуляции, а вершины полигона – вершинами треугольников. Если для построения триангуляции не используются никакие другие вершины, то она называется **минимальной** (Рисунок 1.87а). Часто под триангуляцией полигона понимают именно его минимальную триангуляцию. Однако иногда используются дополнительные вершины для того, чтобы получить более удобную форму треугольников, например, для предотвращения вытянутых треугольников, максимизации минимального угла, ограничения их площадей и т.д. (Рисунок 1.87б).

Нетрудно видеть, что минимальная триангуляция полигона получается путём проведения некоторых его диагоналей. Более того, для всякого самонепересекающегося полигона с n вершинами существует минимальная триангуляция, состоящая ровно из $n-2$ треугольников. ◆ Докажем это утверждение методом индукции. Базис индукции очевиден: при $n = 3$ имеем один треугольник ($n - 2 = 1$), который сам образует простейшую триангуляцию. Теперь докажем это же утверждение для $n = k$ при предположении, что оно выполняется при $n \in \{3,4,...,k-1\}$. Для этого нужно провести некоторую диагональ в полигоне, тем самым разбив его на два полигона с меньшим числом вершин в каждом. Провести диагональ в полигоне можно разными способами:



а б

Рисунок 1.87 – Триангуляции полигона: а – минимальная триангуляция, б – триангуляция с дополнительными вершинами

- Из любой **вогнутой вершины**, т.е. вершины, угол при которой больше развёрнутого, возможно проведение по крайней мере одной диагонали. Действительно, если предположить, что из некоторой вершины A_i нельзя провести диагональ (т.е. отрезок, соединяющий её с некоторой другой вершиной и **целиком** лежащий **внутри** полигона), то получается, что всякий луч, проведённый из такой вершины внутрь угла $\angle A_{i-1}A_iA_{i+1}$, пересекает одно и то же ребро полигона, другими словами, из данной вершины видна часть только одного ребра (Рисунок 1.88). Обозначая пересечения контура полигона лучами A_iA_{i-1} и A_iA_{i+1} через A'_{i-1} и A'_{i+1}, получим треугольник с суммой углов $\angle A_{i-1}A_iA_{i+1} + \angle A_iA'_{i-1}A'_{i+1} + \angle A_iA'_{i+1}A'_{i-1} = \pi,$ откуда следует $\angle A_{i-1}A_iA_{i+1} < \pi.$

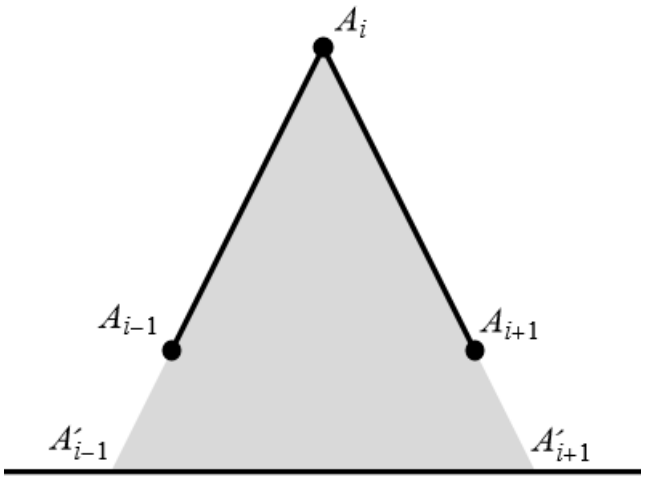


Рисунок 1.88 – Вершина полигона, из которой нельзя провести диагональ

- При рассмотрении же **выпуклой вершины**, т.е. той, угол при которой меньше развёрнутого, возможны различные варианты. Если при некоторой выпуклой вершине A_i треугольник $A_{i-1}A_iA_{i+1}$ не содержит других вершин этого же полигона, то тогда можно провести диагональ $A_{i-1}A_{i+1}$ (Рисунок 1.89а). В противном же случае диагональ можно провести из A_i к вершине внутри этого треугольника, которая оказалась наиболее удалённой от прямой $A_{i-1}A_{i+1}$ (Рисунок 1.89б).

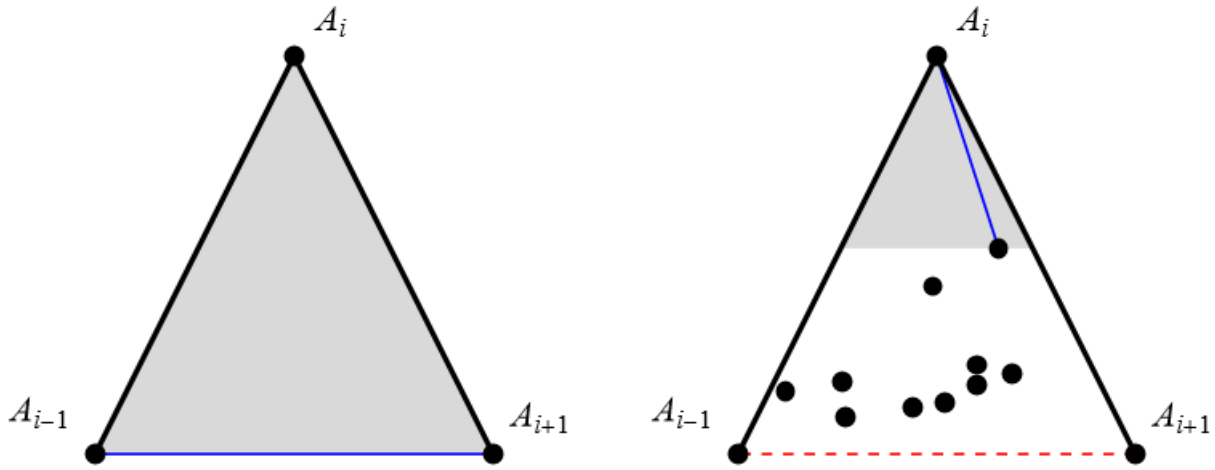


Рисунок 1.89 – Варианты проведения диагонали из выпуклой вершины

Итак, во всяком самонепересекающемся полигоне можно провести по крайней мере одну диагональ. Она разбивает весь исходный k-угольник на два полигона, число вершин у которых обозначим через k_1 и k_2 (Рисунок 1.90). Так как $k_1 < k$ и $k_2 < k$, то по предположению индукции получаем, что для двух новых полигонов выполняется доказываемое утверждение, т.е. их минимальные триангуляции содержат соответственно $k_1 - 2$ и $k_2 - 2$ треугольника. В сумме это даёт $k_1 + k_2 - 4$ треугольника. Но число $k_1 + k_2$, как нетрудно видеть из рисунка 1.90, равно $k + 2$, так как две фиолетовые вершины являются вершинами обоих полигонов. Поэтому окончательно получаем, что минимальная триангуляция исходного k-угольника содержит ровно $k_1 + k_2 - 4 =$ $= k - 2$ треугольника. Таким образом, утверждение полностью доказано. ■

Итак, всякий самонепересекающийся n-угольник при проведении достаточного числа диагоналей (нетрудно вычислить их количество: оно должно быть на единицу меньшим, чем число треугольников в искомой триангуляции, т.е. $n - 3$) можно разбить на $n - 2$ треугольника. Однако поиск пары вершин, отрезок между которыми образует диагональ, – нетривиальная задача. Для того, чтобы в полигоне $A_1A_2...A_nA_1$ отрезок A_iA_j, $i \neq j$, являлся диагональю, необходимо выполнение следующих условий:

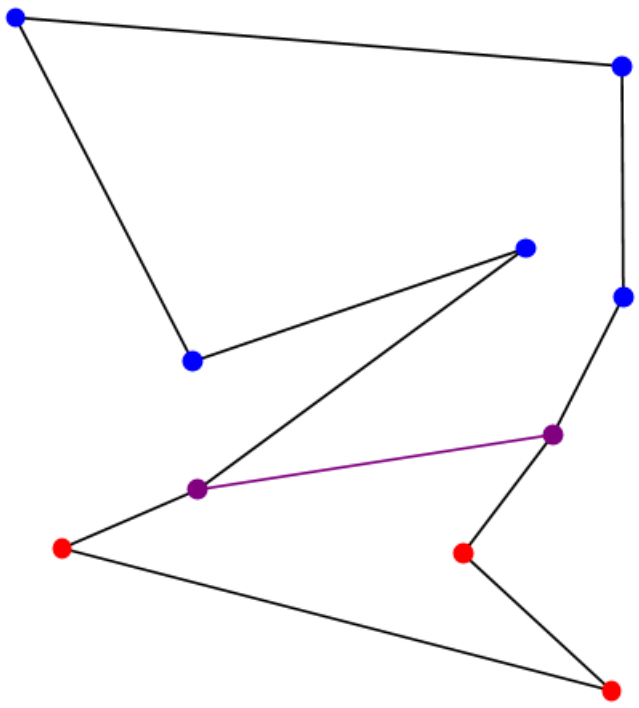


Рисунок 1.90 – Разбиение полигона диагональю на две части

- Этот отрезок не пересекается ни с каким из рёбер полигона и не касается их: $A_iA_j \cap A_kA_{k+1} = \varnothing$, $k \notin \{i, j, i-1, j-1\}$, а также $\overrightarrow{A_iA_j} \nuparrow\!\!\!\nearrow \overrightarrow{A_iA_{i-1}}$, $\overrightarrow{A_iA_j} \nuparrow\!\!\!\nearrow \overrightarrow{A_iA_{i+1}}$, $\overrightarrow{A_jA_i} \nuparrow\!\!\!\nearrow \overrightarrow{A_jA_{j-1}}$, $\overrightarrow{A_jA_i} \nuparrow\!\!\!\nearrow \overrightarrow{A_jA_{j+1}}$.

- Этот отрезок расположен внутри полигона: хотя бы один из углов $\angle\left(\overrightarrow{A_{i-1}A_i}, \overrightarrow{A_iA_j}\right)$ или $\angle\left(\overrightarrow{A_iA_{i+1}}, \overrightarrow{A_iA_j}\right)$ положительный, если полигон имеет левостороннее направление обхода, либо отрицательный, если имеет место правосторонний обход.

Существует множество алгоритмов триангуляции полигонов. В настоящее время в основном используются те из них, время выполнения которых оценивается асимптотикой $O(n \log n)$, где n – число вершин полигона. Ниже приведены только самые простые алгоритмы, которые опираются лишь на проверку возможности проведения тех или иных диагоналей.

Алгоритм триангуляции диагоналями состоит в проведении диагоналей из вогнутых вершин. Для нахождения вогнутых и выпуклых вершин полигона первоначально определяется его направление обхода. Для этого воспользуемся следующим равенством, справедливым для всякого самонепересекающегося полигона $A_1A_2...A_nA_1$ [1, с. 50]:

$$\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_2A_3}\right) + \angle\left(\overrightarrow{A_2A_3}, \overrightarrow{A_3A_4}\right) + ... + \angle\left(\overrightarrow{A_{n-1}A_n}, \overrightarrow{A_nA_1}\right) + \angle\left(\overrightarrow{A_nA_1}, \overrightarrow{A_1A_2}\right) = \pm 2\pi,$$

причём знак «плюс» свидетельствует о левостороннем обходе, «минус» – о правостороннем. Чтобы избежать вычисления величин всех углов из этого равенства, можно воспользоваться октантным подходом, описанным в первой

части ЭУМК [1, с. 66–70 и разд. 2.3.3]. **Октантный алгоритм определения вогнутых и выпуклых вершин** выглядит следующим образом:

1. Вычисляются направляющие векторы для каждого ребра $\overrightarrow{A_iA_{i+1}},\ i=\overline{1,n}.$

2. Для каждого вектора $\overrightarrow{A_iA_{i+1}}$ вычисляется номер октантного угла I_i, в котором он лежит.

3. Каждой вершине A_i ставится в соответствие разность октантных углов векторов $\overrightarrow{A_iA_{i+1}}$ и $\overrightarrow{A_{i-1}A_i}$: $\Delta_i = I_i - I_{i-1}$. Она указывает на величину угла $\angle\left(\overrightarrow{A_{i-1}A_i}, \overrightarrow{A_iA_{i+1}}\right)$ поворота в этой вершине.

4. Разности подлежат корректировке:

$$\Delta_i' = \begin{cases} \Delta_i, |\Delta_i| < 4, \\ \Delta_i - 8\operatorname{sgn}\Delta_i, |\Delta_i| > 4, \\ 4\operatorname{sgn}\angle\left(\overrightarrow{A_{i-1}A_i}, \overrightarrow{A_iA_{i+1}}\right), \Delta_i = \pm 4. \end{cases}$$

5. Вычисляется сумма скорректированных разностей. Если $\Delta = \sum_{i=1}^{n} \Delta_i' = +8,$ то получается, что полигон $A_1A_2...A_nA_1$ – левосторонний, если $\Delta = -8,$ то тогда имеем дело с правосторонним полигоном.

6. Вычисляется знак поворота при каждой вершине:

$$\delta_i = \begin{cases} \operatorname{sgn}\Delta_i', \Delta_i' \neq 0, \\ \operatorname{sgn}\angle\left(\overrightarrow{A_{i-1}A_i}, \overrightarrow{A_iA_{i+1}}\right), \Delta_i' = 0. \end{cases}$$

7. Определяются вогнутые и выпуклые вершины: вершина A_i является выпуклой, если $\operatorname{sgn}\delta_i = \operatorname{sgn}\Delta$; вогнутой, если $\operatorname{sgn}\delta_i = -\Delta$; либо фиктивной, если $\operatorname{sgn}\delta_i = 0.$

Конечно, в полигоне может вовсе не оказаться вогнутых вершин. В этом случае он является выпуклым, и можно просто по очереди провести все диагонали, соединяющие произвольные вершины (при этом отслеживая, чтобы они не пересекались друг с другом), например, провести все $n - 3$ диагонали из одной вершины. Если же нашлась некоторая вогнутая вершина, то для неё осуществляется поиск другой вершины, к которой можно провести диагональ. После проведения найденной диагонали дальнейшие диагонали ищутся уже в частях, полученных после разбиения исходного полигона этой диагональю.

Таким образом, может быть дано следующее формальное описание **алгоритма триангуляции** самонепересекающегося минимального полигона $A_1A_2...A_nA_1$ **диагоналями**, проводимыми из вогнутых вершин:

1. Вычисляется направление обхода полигона, а также выпуклость каждой из вершин. **Вогнутые** вершины записываются в отдельный список $C = \{A_i \mid \delta_i = -\operatorname{sgn}\Delta\}$.
2. Если список C пустой, то алгоритм возвращает список треугольников $\{\triangle A_1 A_i A_{i+1}\}_{i=2}^{n-1}$.
3. Пусть первой вершиной из списка C является A_k. Тогда для неё нужно найти второй конец искомой диагонали.
   3а. Перебор начинается с первой несмежной вершины: $l := k + 2$ (в случае $k + 2 > n$ нужно положить значение $k - n + 2$).
   3б. Проверяется расположение отрезка $A_k A_l$ относительно ребра $A_k A_{k+1}$. Если $\operatorname{sgn}\angle\left(\overrightarrow{A_{k-1}A_k}, \overrightarrow{A_k A_l}\right) = \operatorname{sgn}\angle\left(\overrightarrow{A_k A_{k+1}}, \overrightarrow{A_k A_l}\right) = -\operatorname{sgn}\Delta$, тогда $A_k A_l$ не является диагональю рассматриваемого полигона – нужно перейти к шагу 3д.
   3в. Проверяется пересекаемость отрезком $A_k A_l$ рёбер полигона. Для этого могут использоваться модификации различных алгоритмов нахождения пересечений отрезков, которые не находят сами пересечения, а только проверяют их наличие. При наличии таких пересечений (сами концы A_k и A_l пересечениями считать не следует) нужно перейти к шагу 3д.
   3г. Отрезок $A_k A_l$ является диагональю полигона $A_1 A_2 ... A_n A_1$. Проводя её, получим два новых полигона: $\mathcal{A}_1 = A_k A_l A_{l+1} ... A_{k-1} A_k$ и $\mathcal{A}_2 = A_l A_k A_{k+1} ... A_{l-1} A_l$, оба имеют то же направление обхода $\operatorname{sgn}\Delta$, что и исходный полигон. Переход к шагу 4.
   3д. Отрезок $A_k A_l$ не образует диагонали полигона. Значит, нужно перейти к следующей вершине: если $l = n$, то $l := 1$, в противном случае $l := l + 1$. Переход к шагу 3б.
4. Каждому из новых полигонов ставится в соответствие список их вогнутых вершин, которые берутся из списка C: $C_j = \{A_i \mid A_i \in C \,\&\, \&A_i \in \mathcal{A}_j\}$, $j = \overline{1,2}$.
5. В обоих полигонах вершины A_k и A_l могут оказаться выпуклыми или фиктивными, так как уже провели диагональ. Поэтому выполняются следующие проверки:
   5а. Если $\operatorname{sgn}\angle\left(\overrightarrow{A_{k-1}A_k}, \overrightarrow{A_k A_l}\right) \in \{\operatorname{sgn}\Delta, 0\}$, то вершина A_k перестала быть вогнутой в полигоне $\mathcal{A}_1$, и её нужно убрать из списка C_1: $C_1 := C_1 \setminus \{A_k\}$.
   5б. При $\operatorname{sgn}\angle\left(\overrightarrow{A_l A_k}, \overrightarrow{A_k A_{k+1}}\right) \in \{\operatorname{sgn}\Delta, 0\}$ редактируется список вогнутых вершин полигона $\mathcal{A}_2$: $C_2 := C_2 \setminus \{A_k\}$.

5в. При наличии вершины A_l в списке C_1 и выполнении равенства $\operatorname{sgn}\angle\left(\overrightarrow{A_kA_l}, \overrightarrow{A_lA_{l+1}}\right) \in \{\operatorname{sgn}\Delta, 0\}$ эта вершина удаляется из списка C_1.

5г. При наличии вершины A_l в списке C_2 и выполнении равенства $\operatorname{sgn}\angle\left(\overrightarrow{A_{l-1}A_l}, \overrightarrow{A_lA_k}\right) \in \{\operatorname{sgn}\Delta, 0\}$ эта вершина удаляется из списка C_2.

5д. Если на каком-то из шагов 5а–5г получился ноль, то это означает, что соответствующая вершина оказалась фиктивной, и её нужно также удалить из списка вершин полигона $\mathcal{A}_1$ или $\mathcal{A}_2$.

6. Шаги 2–5 выполняются для полигонов $\mathcal{A}_1$ и $\mathcal{A}_2$ и соответствующих им списков C_1 и C_2.

7. Алгоритм возвращает объединение двух списков, полученных для этих двух полигонов.

Пример выполнения этого алгоритма приведён на рисунке 1.91. Порядок проведения диагоналей обозначен индексами. После проведения диагонали e_1 получился полигон $A_1...A_4A_6A_7...A_{12}A_{11}$, в котором вершина A_6 оказалась фиктивной. Это привело к тому, что вместо $n-3=9$ диагоналей проведено только 8, и получилось 7 треугольников.

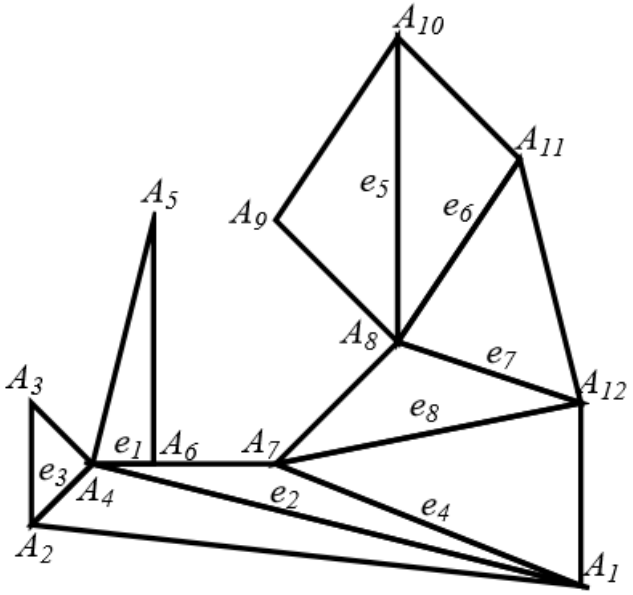


Рисунок 1.91 – Триангуляция полигона
диагоналями, проведёнными из вогнутых вершин

Легко видеть, что в ходе этого алгоритма происходит уменьшение числа вогнутых вершин до тех пор, пока все части исходного полигона не окажутся выпуклыми. В итоге будет проведено не более $n-3$ диагоналей. При проведении каждой из диагоналей для некоторой фиксированной вогнутой вершины будут рассматриваться в худшем случае все $n-3$ несмежные с ней вершины для того, чтобы среди них найти конец искомой диагонали. При этом, в свою очередь, для текущего отрезка проверяется её пересекаемость не более,

чем с n рёбрами полигона. Таким образом, верхняя оценка асимптотики времени работы этого алгоритма составляет $O(n^3)$, которая может быть улучшена, например, при помощи эффективного перебора возможных концов для искомой диагонали или использованием оптимальных алгоритмов нахождения пересечений отрезков. Кроме того, из-за разбиения полигона на две части фактически имеет место среднее время, куда меньшее этой оценки, особенно при делении на два полигона с примерно равным числом вершин.

Алгоритм «отрезания ушей». Ещё один алгоритм заключается в том, чтобы найти некоторое ухо полигона, т.е. такую выпуклую вершину A_i, что отрезок $A_{i-1}A_{i+1}$, соединяющий две смежные вершины, образует диагональ. Как отмечено выше, такое происходит тогда и только тогда, когда треугольник $A_{i-1}A_iA_{i+1}$ не содержит других вершин этого же полигона.

Всякий самонепересекающийся полигон имеет по крайней мере одно ухо.

◆ Доказательство этого факта опирается на представление минимальной триангуляции самонепересекающегося полигона в виде графа $G = (V, E)$, где V – множество вершин, каждая из которых обозначает некоторый треугольник, E – множество рёбер, каждое из которых соединяет две вершины, соответствующие граничащим треугольникам (Рисунок 1.92). Ясно, что такой граф содержит $n-2$ вершины, $n-3$ ребра (так как общим ребром для двух треугольников служит некоторая из $n-3$ проведённых диагоналей), а также должен быть связным (откуда следует, что из каждой вершины исходит по крайней мере одно ребро). Предположим, что степень каждой вершины (т.е. число рёбер, исходящих из неё) равна 2 или выше. Вычислим сумму всех степеней: она равна числу концов всех рёбер, т.е. в два раза превышает число всех рёбер. Отсюда получаем формулу суммы степеней всех вершин графа:

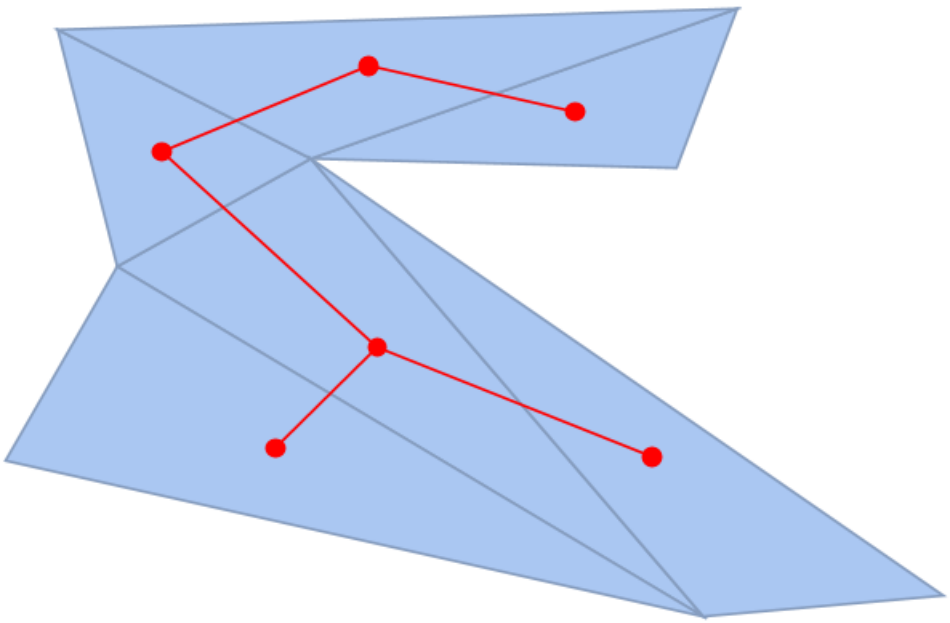


Рисунок 1.92 – Граф для представления триангуляции полигона

$$\sum_{v\in V}\deg(v)=2|E|.$$

С другой стороны, имеем $\sum_{v\in V}\deg(v)\geq 2|V|$. Это приводит к неверному неравенству $2(n-3)\geq 2(n-2)$. Из этого противоречия следует вывод о том, что существует вершина со степенью 1, т.е. найдётся некоторый треугольник, граничащий только с одним другим треугольником по одной из проведённых диагоналей. Две другие стороны у этого треугольника – рёбра исходного полигона, что и означает, что он является ухом данного полигона. ■

Таким образом, при нахождении уха $\triangle A_{i-1}A_iA_{i+1}$ у полигона $A_1A_2...A_nA_1$ можно провести диагональ $A_{i-1}A_{i+1}$, тем самым отрезав это ухо, и останется полигон $A_1A_2...A_{i-1}A_{i+1}...A_nA_1$. У нового полигона, в свою очередь, тоже имеется некоторое ухо, которое подлежит отрезанию, что приводит к образованию ещё одного полигона и т.д.

Отсюда следует **алгоритм «отрезания ушей»** для триангуляции самонепересекающегося полигона $A_1A_2...A_nA_1$:

1. Как и в предыдущем алгоритме, нужно выяснить, какие вершины являются выпуклыми, а какие – вогнутыми. Все **выпуклые** вершины запишем в отдельный список: $C=\{A_i\mid \delta_i=\operatorname{sgn}\Delta\}$. Также определяется список, куда будут вноситься все искомые уши: $T:=\varnothing$. Вершины текущего полигона также можно записать в отдельный список: $\mathcal{A}:=\{A_i\}_{i=1}^n$.

2. Происходит перебор всех вершин из C, пока не найдено ухо. Вначале выбирается номер индекса i такой, что вершина A_i является первой в списке C. Её соседние вершины в текущем полигоне обозначим A' и A''.

3. Для треугольника при текущей вершине, $\triangle A'A_iA''$, определяется, не попадает ли в него какая-либо из вершин $A_j\in\mathcal{A}\setminus\{A',A_i,A''\}$. Если $\exists j: A_j\in\triangle A'A_iA''$, то индекс i переопределяется таким образом, что A_i – это следующая точка из C. Для новой точки A_i определяются её соседи, которые снова обозначаются через A' и A'', и текущий шаг повторяется.

4. $\triangle A'A_iA''$ не содержит внутри никаких вершин полигона, значит, образует ухо. Таким образом, его нужно внести в список T. Кроме того, вершина A_i удаляется из списков $\mathcal{A}$ и C.

5. Для вершин A' и A'', если они отсутствуют в списке C, проверяется, окажутся ли они выпуклыми в новом полигоне. Выполняется это аналогично предыдущему алгоритму. Новые выпуклые вершины вносятся в список C. Если же они оказываются фиктивными, то они удаляются из $\mathcal{A}$.

6. Шаги 2–5 повторяются до тех пор, пока в списке T не окажется ровно $n-2$ треугольника.

Пример работы этого алгоритма приведён на рисунке 1.93. Порядок отсечения ушей обозначен индексами.

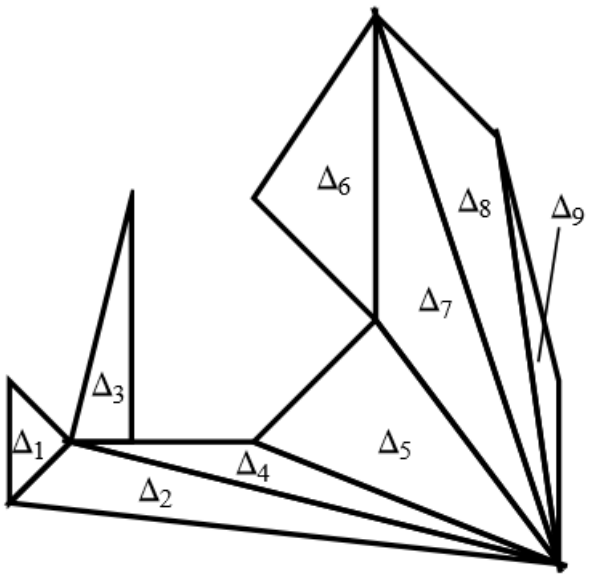


Рисунок 1.93 – Триангуляция полигона «отрезанием ушей»

Как и предыдущий алгоритм, алгоритм «отрезания ушей» работает за время $O(n^3)$. Действительно, при «отрезании» каждого из $n-2$ ушей проверка того, образует ли некоторая из $O(n)$ выпуклых вершин ухо, проводится за время $O(n)$ в силу перебора всех вершин текущего полигона, кроме этой текущей вершины и её двух соседей.

При следующей модификации этого алгоритма удаётся достичь асимптотики $O(n^2)$. Суть в том, что на шаге 3 переход к следующей выпуклой вершине в случае наличия точек внутри треугольника $\triangle A'A_iA''$ заменяется проведением диагонали от вершины A_i к вершине, которая среди всех внутренних точек этого треугольника является наиболее удалённой от прямой $A'A''$ (см. рисунок 1.89б). При такой реализации всегда проводится диагональ либо от вершины A_i, либо между вершинами A' и A'', что предотвращает перебор выпуклых вершин на шаге 3. Таким образом, получается, что проведение некоторой диагонали в любом самонепересекающемся полигоне выполняется за время $O(n)$:

- вычисление направления обхода полигона и выпуклости всех вершин выполняется за $O(n)$;
- поиск некоторой выпуклой вершины в списке вершин полигона также осуществляется за $O(n)$;
- для определения возможности «отсечения уха» при найденной вершине для $O(n)$ точек определяется, лежит ли каждая из них внутри треугольника $\triangle A'A_iA''$;
- наконец, в случае наличия внутренних точек наиболее удалённая от прямой $A'A''$ также находится за время $O(n)$.

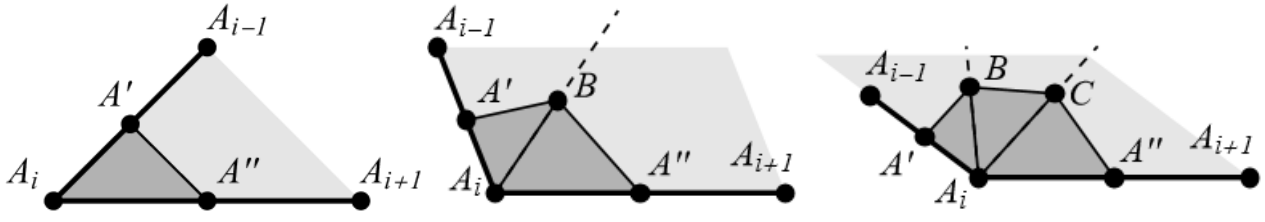
1.5.2. Контролируемая триангуляция

Для триангуляции геометрических множеств на треугольники с требуемой формой нередко используются специальные алгоритмы, в ходе которых порождаются новые вершины. В данном подразделе приводится алгоритм контролируемой триангуляции, предложенный Е. А. Никулиным [3, с. 121–125; 32]. В результате получаются треугольники, у каждого из которых стороны имеют длину, не превышающую заданное значение h.

Суть этого алгоритма, как и алгоритма «отрезания ушей», заключается в отрезании треугольников от некоторых вершин текущего полигона. Вначале выбирается некоторая выпуклая вершина полигона A_i. Может иметь место один из следующих вариантов:

- $\angle A_{i-1}A_iA_{i+1} \le \pi/3$ (Рисунок 1.94а). Это означает, что отложив на лучах A_iA_{i-1} и A_iA_{i+1} по отрезку длины не более h каждый, получим две точки A' и A'', расстояние между которыми также не превосходит h. Этот факт следует из того, что во всяком треугольнике напротив бóльшего угла лежит более длинная сторона, и если $\angle A_{i-1}A_iA_{i+1} \le \pi/3$, то в $\triangle A'A_iA''$ какой-то из двух других углов имеет бóльшую величину. Но напротив этого угла лежит сторона длины, не большей h, откуда и следует то, что напротив $\angle A_{i-1}A_iA_{i+1}$ лежит сторона, длина которой также не превышает h. Итак, в треугольнике $A'A_iA''$ все три стороны имеют длину не больше h, и его можно отсечь от исходного полигона.

- Если же $\pi/3 < \angle A_{i-1}A_iA_{i+1} \le 2\pi/3$, тогда при аналогичном откладывании отрезков на рёбрах исходного полигона может оказаться справедливым неравенство $\left|\overrightarrow{A'A''}\right| > h$. В этом случае из угла $\angle A_{i-1}A_iA_{i+1}$ проводится биссектриса, на которой отмечается такая новая вершина B, что $\left|\overrightarrow{A_iB}\right| = \frac{1}{2}\left|\overrightarrow{A_iA'}\right| + \frac{1}{2}\left|\overrightarrow{A_iA''}\right|$ (Рисунок 1.94б). Это условие, кроме выполнения заданного ограничения на стороны треугольников, даёт также треугольники, форма которых достаточно близка к равностороннему треугольнику. Нетрудно проверить, что если $\left|\overrightarrow{A_iA'}\right| \le h$ и $\left|\overrightarrow{A_iA''}\right| \le h$, то и для трёх других отрезков выполняется это же условие: $\left|\overrightarrow{A_iB}\right| \le h$, $\left|\overrightarrow{A'B}\right| \le h$, $\left|\overrightarrow{BA''}\right| \le h$.



а б в

Рисунок 1.94 – Варианты отсечения треугольников

- При $2\pi/3 < \angle A_{i-1}A_iA_{i+1} < \pi$ и $\left|\overrightarrow{A'A''}\right| > h$ проводятся две трисектрисы, на которых отмечаются такие точки B и C, что $\left|\overrightarrow{A_iB}\right| = \frac{2}{3}\left|\overrightarrow{A_iA'}\right| + \frac{1}{3}\left|\overrightarrow{A_iA''}\right|$ и $\left|\overrightarrow{A_iC}\right| = \frac{1}{3}\left|\overrightarrow{A_iA'}\right| + \frac{2}{3}\left|\overrightarrow{A_iA''}\right|$ (Рисунок 1.94в).

Таким образом, при рассмотрении очередной вершины происходит отсечение одного, двух или трёх треугольников. Дальнейшему рассмотрению подлежит полигон, оставшийся после этого отсечения. Однако при подобном отсечении может оказаться пересечение новых рёбер этого полигона со старыми рёбрами, либо они окажутся за его пределами. В этом случае необходимо вернуться к старому полигону и у той же вершины A_i подобрать новые, более близкие к ней положения для точек A' и A''.

Перед формулировкой алгоритма определим функцию, которая для некоторых векторов $\vec{a}(a_x, a_y)$ и $\vec{b}(b_x, b_y)$ и $\lambda \in (0,1)$ вычисляет вектор $\vec{c}$, который делит угол между этими векторами, проведёнными из одной точки, в отношении $\lambda : (1-\lambda)$, а также имеет длину, вычисляемую по формуле среднего взвешенного длин двух исходных векторов (коэффициент при длине вектора $\vec{a}$ тем больше, чем меньше угол между $\vec{a}$ и $\vec{c}$, также, как и для вектора $\vec{b}$):

$$\left|\vec{c}\right| = (1-\lambda)\left|\vec{a}\right| + \lambda\left|\vec{b}\right|. \tag{1.90}$$

Для определения направления искомого вектора $\vec{c}$ нужно вычислить величины углов α и β между положительным направлением оси Ox и соответственно векторами $\vec{a}$ и $\vec{b}$. Для произвольного вектора с координатами (x, y) имеет место следующая формула:

$$\omega(x,y) = \begin{cases} \operatorname{arctg}\dfrac{y}{x}, x > 0, \\ \operatorname{arctg}\dfrac{y}{x} + \operatorname{sgn} y \cdot \pi, x < 0 \;\&\; y \neq 0, \\ \pi, x < 0 \;\&\; y = 0, \\ \operatorname{sgn} y \cdot \dfrac{\pi}{2}, x = 0. \end{cases} \tag{1.91}$$

Областью значений этой функции является полуинтервал $(-\pi, \pi]$. После вычисления углов $\alpha = \omega(a_x, a_y)$ и $\beta = \omega(b_x, b_y)$ может оказаться один из двух следующих вариантов:

- Если $\left|\alpha - \beta\right| < \pi$, то тогда всякое значение между α и β соответствует некоторому вектору, лежащему внутри угла между векторами $\vec{a}$ и $\vec{b}$.

- Если же $|\alpha-\beta|>\pi$, то необходима корректировка одного из этих значений, например, меньший из этих углов увеличивается на 2π. Тогда он станет больше другого угла, но не более, чем на π.

Обозначим через α' и β' откорректированные значения углов для векторов $\vec{a}$ и $\vec{b}$. Тогда искомому вектору $\vec{c}$ соответствует значение угла, равное

$$\gamma=(1-\lambda)\alpha'+\lambda\beta'. \qquad (1.92)$$

Действительно, если предположить, что $\alpha'<\beta'$, то тогда имеем

$$\gamma-\alpha'=-\lambda\alpha'+\lambda\beta'=\lambda(\beta'-\alpha')>0,$$
$$\beta'-\gamma=(\lambda-1)\alpha'+(1-\lambda)\beta'=(1-\lambda)(\beta'-\alpha')>0,$$
$$\frac{\gamma-\alpha'}{\beta'-\gamma}=\frac{\lambda(\beta'-\alpha')}{(1-\lambda)(\beta'-\alpha')}=\frac{\lambda}{1-\lambda},$$

т.е. имеет место деление угла между двумя исходными векторами в заданном отношении. Аналогичное равенство получается при $\alpha'>\beta'$. Отсюда с учётом (1.90) нетрудно вычислить координаты итогового вектора:

$$\vec{c}(\vec{a},\vec{b},\lambda)=\vec{c}\left(\cos\gamma\left((1-\lambda)|\vec{a}|+\lambda|\vec{b}|\right),\sin\gamma\left((1-\lambda)|\vec{a}|+\lambda|\vec{b}|\right)\right). \qquad (1.93)$$

Определённая на отрезке $\lambda\in[0,1]$, эта функция даёт непрерывную линию, соединяющая концы векторов $\vec{a}$ и $\vec{b}$ (Рисунок 1.95). В частности, при $\lambda=1/2$ получается вектор, отложенный на биссектрисе, а при $\lambda=1/3$ и $\lambda=2/3$ – два вектора на трисектрисах.

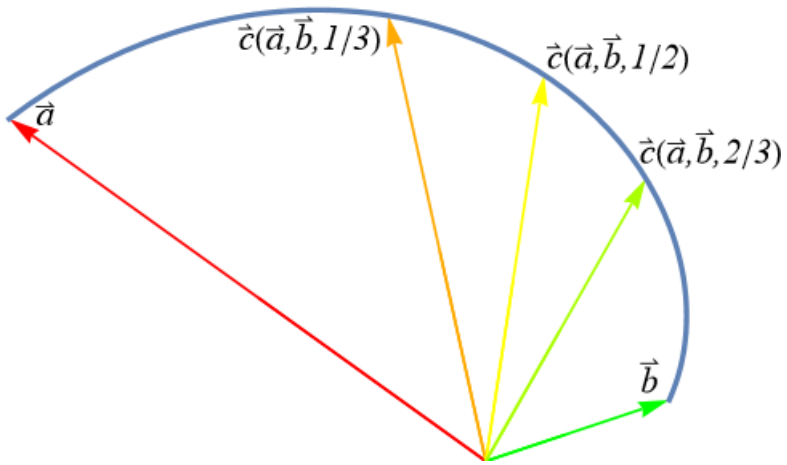


Рисунок 1.95 – Проведение промежуточных векторов внутри угла

Итак, **алгоритм контролируемой триангуляции** самонепересекающегося полигона $A_1A_2...A_nA_1$ на треугольники с длинами сторон не более h выглядит следующим образом:

1. Инициализация текущего полигона: $\mathcal{A} := A_1A_2...A_nA_1$, а также списка отсечённых треугольников: $T := \varnothing$.
2. Определение направления обхода полигона sgn Δ и списка выпуклых вершин $C := \{A_i \mid \delta_i = \operatorname{sgn}\Delta\}$.
3. Для нахождения выпуклой вершины с минимальным углом достаточно рассмотреть косинусы углов при всех выпуклых вершинах: так как косинус является функцией, убывающей на интервале $(0, \pi)$, то

$$\angle A_{i-1}A_iA_{i+1} < \angle A_{j-1}A_jA_{j+1} \Leftrightarrow \cos\angle A_i = \frac{\overrightarrow{A_iA_{i-1}} \cdot \overrightarrow{A_iA_{i+1}}}{\left|\overrightarrow{A_iA_{i-1}}\right|\left|\overrightarrow{A_iA_{i+1}}\right|} > \frac{\overrightarrow{A_jA_{j-1}} \cdot \overrightarrow{A_jA_{j+1}}}{\left|\overrightarrow{A_jA_{j-1}}\right|\left|\overrightarrow{A_jA_{j+1}}\right|}.$$

Таким образом, выбирается вершина с индексом $i = \arg\max\limits_{\{i|A_i \in C\}} \left(\cos\angle A_i\right)$.

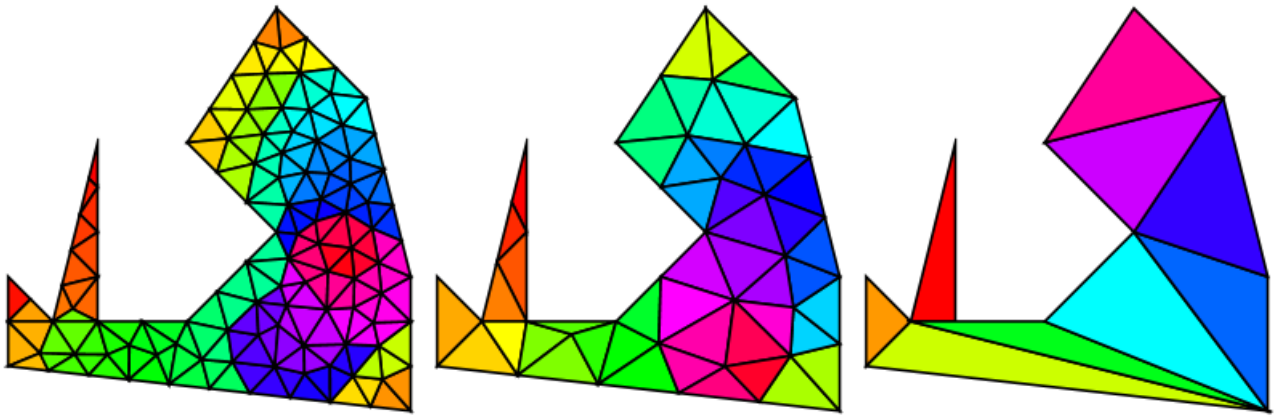
4. Вычисляется минимальное число отрезков длины не более h, на которые можно разбить два ребра $A_{i-1}A_i$ и A_iA_{i+1}:

$$k' := \left\lceil \frac{\left|\overrightarrow{A_iA_{i-1}}\right|}{h} \right\rceil, \quad k'' := \left\lceil \frac{\left|\overrightarrow{A_iA_{i+1}}\right|}{h} \right\rceil. \tag{1.94}$$

5. Откладываются точки $A' := A_i + \frac{1}{k'}\overrightarrow{A_iA_{i-1}}$ и $A'' := A_i + \frac{1}{k''}\overrightarrow{A_iA_{i+1}}$.
6. Далее происходит *предварительное* отсечение согласно одному из следующих вариантов:
   - 6а. Если $\left|\overrightarrow{A'A''}\right| \le h$, то отсекается **один** треугольник: $T' := \{\triangle A'A_iA''\}$. После этого получается полигон $\mathcal{A}' := A_1A_2...A_{i-1}A'A''A_{i+1}...A_nA_1$.
   - 6б. Если $\left|\cos\angle A_i\right| \le 1/2$, что эквивалентно $\pi/3 < \angle A_i \le 2\pi/3$, то тогда отмечается точка $B := A_i + \vec{c}\left(\overrightarrow{A_iA'}, \overrightarrow{A_iA''}, 1/2\right)$, после чего отсекаются **два** треугольника: $T' := \{\triangle A'A_iB, \triangle BA_iA''\}$, $\mathcal{A}' := A_1A_2...A_{i-1}A'BA''A_{i+1}...A_nA_1$.
   - 6в. Если $\cos\angle A_i < -1/2$, что эквивалентно $2\pi/3 < \angle A_i < \pi$, то отмечаются две точки: $B := A_i + \vec{c}\left(\overrightarrow{A_iA'}, \overrightarrow{A_iA''}, 1/3\right)$ и $C := A_i + \vec{c}\left(\overrightarrow{A_iA'}, \overrightarrow{A_iA''}, 2/3\right)$. Отсекаются **три** треугольника: $T' := \{\triangle A'A_iB, \triangle BA_iC, \triangle CA_iA''\}$, $\mathcal{A}' := A_1A_2...A_{i-1}A'BCA''A_{i+1}...A_nA_1$.

7. В зависимости от результатов предыдущего шага определяется, выполняется ли одно из следующих равенств: $\triangle A'A_iA'' = \mathcal{A}$, $A'A_iA''BA' = \mathcal{A}$ или $A'A_iA''CBA' = \mathcal{A}$. Его выполнение означает, что весь текущий полигон разбит на треугольники, и алгоритм завершает работу, возвращая список $T := T \cup T'$.
8. Для полученного полигона $\mathcal{A}'$ проверяется, не оказался ли он самопересекающимся и сохранил ли он направление обхода. Если какое-то из этих условий нарушено (что эквивалентно тому, что внутрь хотя бы одного из отсекаемых треугольников попадает некоторая вершина из $\mathcal{A}$), то происходит переопределение переменных: $k' := k' + 1$, $k'' := k'' + 1$, а также повтор шагов 5–7.
9. Предварительное отсечение, проведённое на шаге 6, становится окончательным. Вершина A_i удаляется из списка C. Список T дополняется всеми треугольниками из T'. Происходит переопределение текущего полигона: $\mathcal{A} := \mathcal{A}'$.
10. Обработка вершины A':
    - 10а. Если $k' > 1$, то это означает, что A' является новой вершиной, которая к тому же всегда является выпуклой и подлежит добавлению в список C. Переход к шагу 11.
    - 10б. Если $k' = 1$, то это означает, что $A' = A_{i-1}$, и из списка вершин $\mathcal{A}$ нужно удалить повторную вершину A'. Кроме того, в новом полигоне вершина A_{i-1} может оказаться вогнутой, выпуклой или фиктивной.
    - 10в. Если вершины A_{i-1} нет в списке C, то вычисляется знак угла $\angle\left(\overrightarrow{A_{i-2}A_{i-1}}, \overrightarrow{A_{i-1}A''}\right)$ либо $\angle\left(\overrightarrow{A_{i-2}A_{i-1}}, \overrightarrow{A_{i-1}B}\right)$ (в зависимости от результатов шага 6). В случае его совпадения со знаком Δ обхода исходного (а значит, и текущего) полигона вершину A_{i-1} нужно добавить в список C. Если же знак оказывается нулевым, то тогда A_{i-1} – фиктивная вершина, и её нужно удалить из списка вершин $\mathcal{A}$.
11. Шаг 10 повторяется по отношению к вершине A'', при этом k' заменяется на k'', вершина A_{i-1} – на A_{i+1}, а указанные углы – на один из углов $\angle\left(\overrightarrow{A'A_{i+1}}, \overrightarrow{A_{i+1}A_{i+2}}\right)$, $\angle\left(\overrightarrow{BA_{i+1}}, \overrightarrow{A_{i+1}A_{i+2}}\right)$ либо $\angle\left(\overrightarrow{CA_{i+1}}, \overrightarrow{A_{i+1}A_{i+2}}\right)$.
12. Повтор шагов 3–11.

Результаты алгоритма при различных значениях h приведены на рисунке 1.96, где треугольники окрашены в разные цвета радуги в зависимости от порядка их отсечения (от красного к пурпурному). На рисунке 1.96в получилась минимальная триангуляция в силу задания параметра h, превышающего длину всех рёбер исходного полигона, что избавило от необходимости отмечать дополнительные узлы.



а б в

Рисунок 1.96 – Контролируемая триангуляция полигона при разных значениях параметра: h = 1 (а), h = 2 (б), h = 10 (в)

Благодаря указанным методам построения новых вершин получаются треугольники, форма большинства из которых близка к равностороннему треугольнику (если только в исходном полигоне нет слишком малых углов). Кроме того, процедура разбиения рёбер исходного полигона, описанная на шаге 4, предотвращает генерацию слишком маленьких треугольников и чрезмерного разбиения полигона. Недостатком данного алгоритма, очевидно, является необходимость трудоёмких вычислений по формулам (1.91) – (1.93) для добавления дополнительных вершин на шаге 6.

Замечание 1. Оценим трудоёмкость представленного алгоритма. Ясно, что число треугольников в итоговой триангуляции зависит от площади полигона. Если обозначить её через S, то тогда имеем, что число треугольников должно быть не меньше $\frac{4S}{\sqrt{3}h^2}$, ведь максимальная площадь каждого из треугольников, полученных по данному алгоритму, равна $\frac{1}{2}\sin\frac{\pi}{3}\cdot h^2 = \frac{\sqrt{3}}{4}h^2$. Значит, число треугольников можно оценить асимптотикой $\Omega\left(S/h^2\right)$, и оценка эта тем точнее, чем меньше h. Общее число итераций оценивается такой же асимптотикой, ведь число треугольников, отсекаемых на каждой итерации, является ограниченным. Кроме того, эта асимптотика ухудшается проверкой, проводимой на шаге 8, – она осуществляется каждую итерацию за время $O(n)$ (не говоря уже о том, что её результат может привести к повторному выполнению шагов 5–8). Обработка же списка C и выпуклых вершин текущего полигона не ухудшает общей асимптотики алгоритма: при использовании **приоритетной очереди** выполняются следующие операции:

- формирование списка C для исходного полигона $A_1A_2...A_nA_1$ происходит за время $O(n \log n)$,
- извлечение минимальной вершины – за время $O(1)$,
- удаление и добавление вершин – $O(\log n)$.

Значит, формирование исходной очереди выполняется за время $O(n \log n)$, которое при $n << S/h^2$ гораздо меньше асимптотики $\Omega\left(S/h^2\right)$, а редактирование очереди на каждой итерации происходит быстрее, чем выполнение шага 8.

Замечание 2. Для предотвращения лишних повторений шагов 5–8 можно определить отдельную структуру данных, в которой каждому ребру текущего полигона поставлено в соответствие число k разбиений этого ребра на отрезки длины не более h. Первоначально это число вычисляется по формулам (1.94), а после каждого отсечения треугольника со стороной, лежащей на этом ребре, оно уменьшается на 1. При необходимости значение k также может быть увеличено на шаге 8.

Замечание 3. Этот алгоритм можно также использовать, если ограничена не длина стороны, а площадь каждого треугольника. Для этого достаточно исходя из заданного значения максимальной площади s вычислить верхнюю границу для параметра, равную $h_{\max} = \dfrac{2\sqrt{s}}{\sqrt[4]{3}}$ и положить некоторое значение $h \le h_{\max}$.

1.5.3. Триангуляция множества точек. Триангуляция Делоне

Триангуляция множества точек подразумевает проведение отрезков, соединяющих некоторые из пар этих точек таким образом, чтобы эти отрезки образовали множество треугольников. Как и любой связный планарный граф, всякая триангуляция множества точек на плоскости удовлетворяет следующему равенству (**формула Эйлера**):

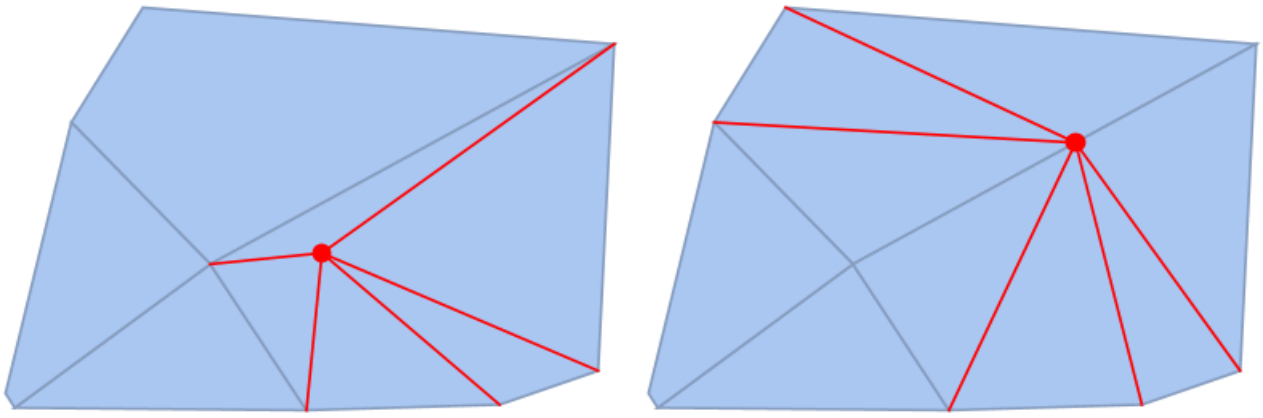
$$|V| - |E| + |F| = 2, \tag{1.95}$$

где V – множество вершин графа, E – множество его рёбер, F – множество его граней, т.е. всех областей, на которые разбивается плоскость в результате проведения всех рёбер (включая внешнюю, бесконечную область). ◆ Данное равенство можно доказать методом индукции по числу вершин $|V| = n$. При $n = 1$ получаем единственную точку, от которой никакие рёбра никуда не проводятся, и имеет место одна-единственная грань – вся плоскость. В этом случае $|V| - |E| + |F| = 1 - 0 + 1 = 2$.

Предположим, что формула (1.95) выполняется для всякого графа, у которого $n = k - 1$, $k \in \mathbb{N}$. Докажем её справедливость для $n = k$. Для этого к графу $G = (V, E)$ с $k - 1$ вершиной добавим ещё одну. Чтобы получился связный граф $G' = (V', E')$, необходимо от новой вершины провести рёбра к старым вершинам. Могут возникнуть следующие ситуации:

- Новая вершина лежит в одной из граней графа G (Рисунок 1.97а). Тогда при проведении e новых рёбер, $e \geq 1$, эта грань разделится на e новых подграни. Таким образом, для нового графа будут выполнены следующие равенства: $|V'| = |V| + 1, \quad |E'| = |E| + e, \quad |F'| = |F| + e - 1$. Подставляя эти значения в (1.95), имеем $|V'| - |E'| + |F'| = |V| + 1 - (|E| + e) + |F| + e - 1 = |V| - |E| + |F| = 2$.

- Новая вершина лежит на некотором ребре графа G (Рисунок 1.97б). Тогда это ребро делится на два новых. Кроме того, так как это ребро является граничным ровно для двух граней графа G, то из новой вершины возможно проведение новых рёбер в каждой из них. Если в первой грани провести e_1 новых рёбер, а во второй – e_2, то тогда произойдёт их деление на $e_1 + 1$ и $e_2 + 1$ новую грань. Отсюда имеем $|V'| = |V| + 1, \quad |E'| = |E| + 1 + e_1 + e_2, \quad |F'| = |F| + e_1 + e_2$, $|V'| - |E'| + |F'| = |V| + 1 - (|E| + 1 + e_1 + e_2) + |F| + e_1 + e_2 = |V| - |E| + |F| = 2$.



а б

Рисунок 1.97 – Добавление новой вершины и проведение новых рёбер из неё

Итак, при добавлении новой вершины равенство (1.95) сохраняется. Это и означает, что оно выполняется для всякого связного планарного графа, в том числе для триангуляции точек на плоскости. ■

Из формулы (1.95) следует ещё одно важное утверждение: число треугольников всякой выпуклой триангуляции n точек (т.е. такой, треугольники которой образуют их выпуклую оболочку) равно $2n - k - 2$, где k – число точек, попавших на границу выпуклой оболочки, в том числе её вершин. Например, на рисунке 1.98 число таких точек равно 5 (четыре вершины выпуклой оболочки и точка A_6, лежащая на отрезке A_2A_3). ◆ Это свойство легко доказывается подсчётом рёбер триангуляции: если посчитать рёбра в каждом треугольнике, то все внешние рёбра триангуляции будут учтены по одному разу, а внутренние – по два (Рисунок 1.98). С учётом введённых обозначений отсюда имеем $3(|F| - 1) = k + 2(|E| - k)$, что даёт $|E| = \frac{1}{2}(3|F| + k - 3)$. При подстановке этого выражения в (1.95) получается $|V| - |E| + |F| = 2 \Rightarrow |V| - \frac{1}{2}(3|F| + k - 3) + |F| =$

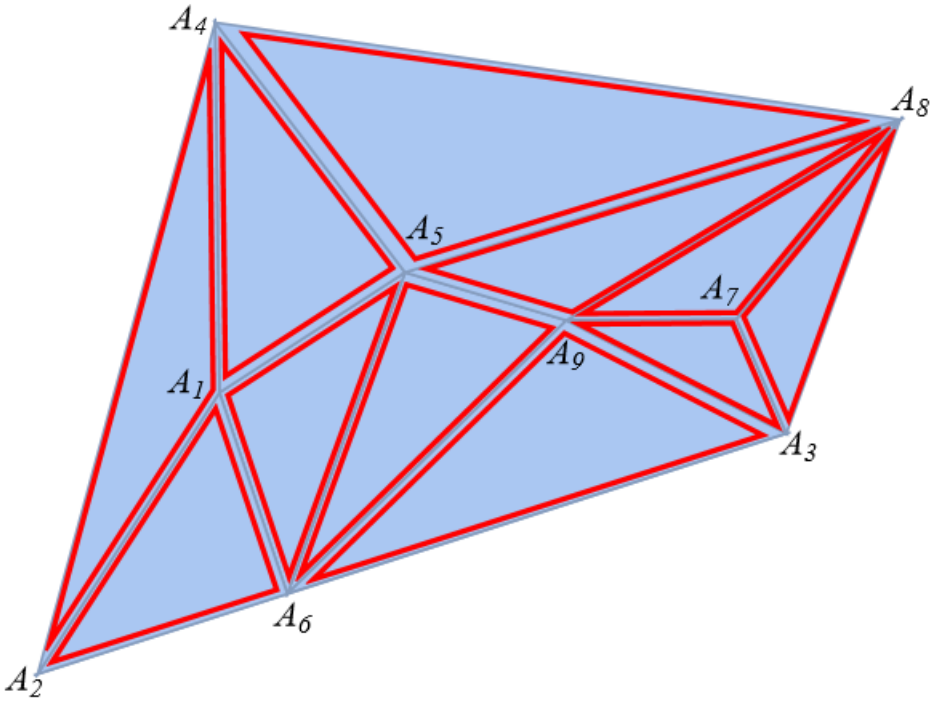


Рисунок 1.98 – Подсчёт рёбер и треугольников триангуляции

$= |V| - \frac{1}{2}|F| - \frac{k}{2} + \frac{3}{2} = 2 \Rightarrow |F| = 2|V| - k - 1$, а число треугольников на 1 меньше (отбрасывается внешняя грань планарного графа), т.е. равно $2n - k - 2$. ■ Из этого доказательства также нетрудно вычислить число рёбер произвольной триангуляции: $|E| = \frac{1}{2}(3|F| + k - 3) = \frac{1}{2}(6|V| - 3k - 3 + k - 3) = 3n - k - 3$, из них внутренних всего $|E| - k = 3n - 2k - 3$.

Задача построения триангуляции точек имеет множество решений, которые можно получить различными алгоритмами. Так, имеет место оценка сверху для числа всех возможных триангуляций n точек на плоскости, равная $O(59^n n^{-6})$ [33]. Отсюда возникает вопрос выбора нужной триангуляции и предъявляемых ей условий.

Одной из используемых на практике триангуляций является **оптимальная**, т.е. та, у которой сумма длин всех рёбер является минимальной из всех триангуляций заданного множества точек. Задача построения этой триангуляции является NP-трудной [34], поэтому большинство используемых на практике алгоритмов построения оптимальной триангуляции дают приблизительное решение. Обзор некоторых из них приведён в [31, глава 7].

Гораздо чаще используется **триангуляция Делоне**, которая впервые упоминается в статье советского учёного Б. Н. Делоне [35]. Триангуляция Делоне – это такая триангуляция множества точек на плоскости, в которой для всякого треугольника описанная около него окружность не содержит внутри себя точек из исходного множества (Рисунок 1.99). Триангуляция Делоне обладает множеством полезных свойств:

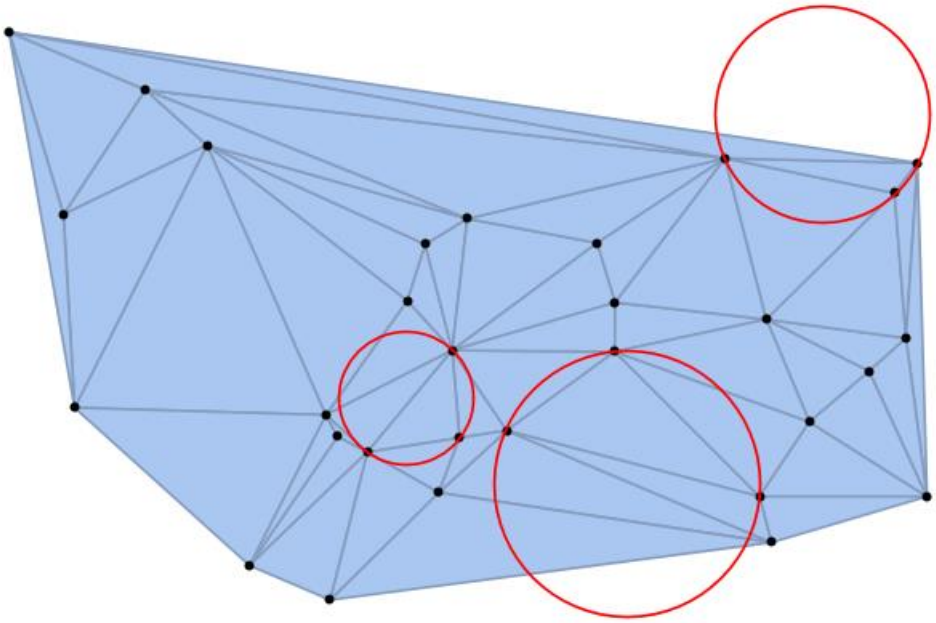


Рисунок 1.99 – Триангуляция Делоне

1. Для заданного множества точек триангуляция Делоне определяется единственным образом (если только никакие четыре точки не лежат на одной окружности).

2. Минимальный угол в триангуляции Делоне имеет максимальную величину по сравнению с минимальным углом любой другой триангуляции. Это приводит к предотвращению построения узких треугольников.

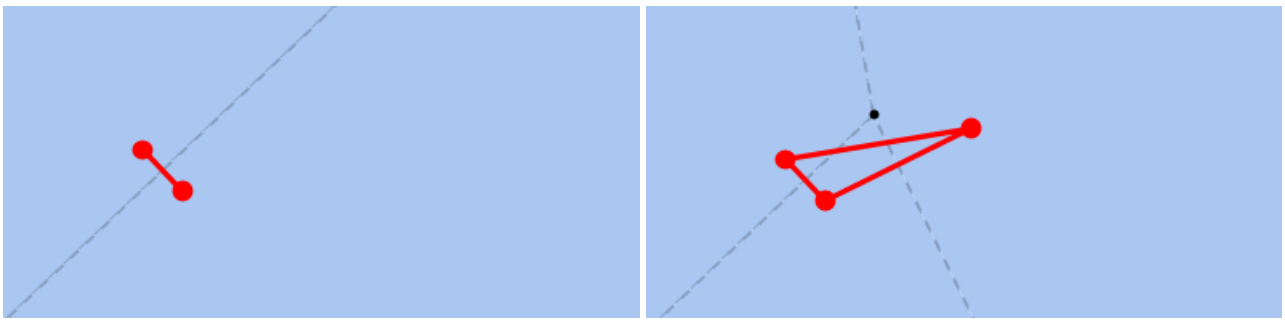
3. Триангуляция Делоне имеет максимальную сумму радиусов вписанных в треугольники окружностей.

4. Триангуляция Делоне имеет минимальную сумму радиусов описанных около треугольников окружностей.

5. В триангуляции Делоне всякая точка исходного множества соединена ребром со своим ближайшим соседом.

6. Триангуляция Делоне является **остовным графом**: длина пути между любыми двумя вершинами по рёбрам триангуляции превышает расстояние между ними не более, чем в 1.998 раз [36].

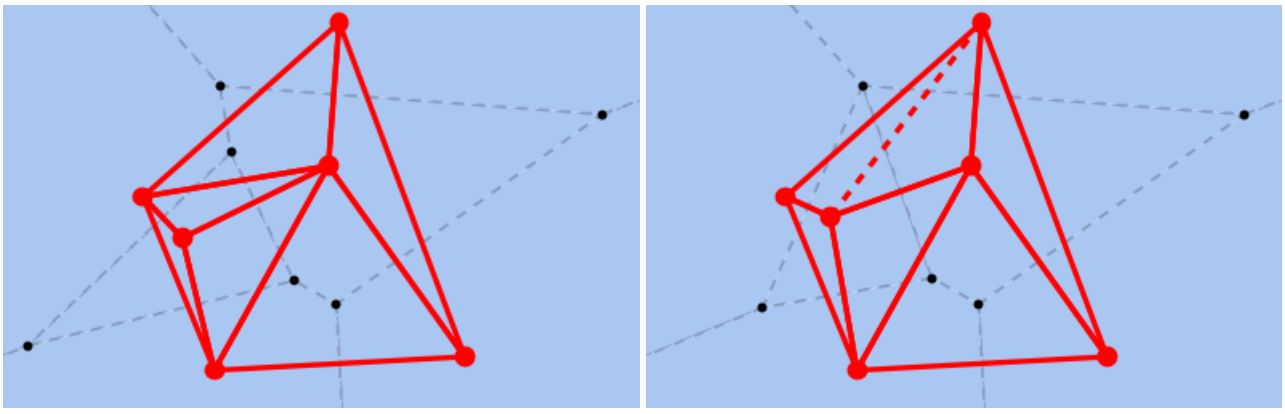
Существование триангуляции Делоне для всякого множества точек $\Omega = \{A_i\}_{i=1}^{n}$ следует из **диаграммы Вороного** – разбиения плоскости на такие области Ω_i, что каждая из них содержит точки, для которых A_i является ближайшей среди всех точек из Ω: $\Omega_i = \left\{ P \mid \arg\min_{j=\overline{1,n}} \left| \overrightarrow{PA_j} \right| = i \right\}$. Например, для двух точек диаграмма Вороного получается проведением срединного перпендикуляра к отрезку с концами в этих точках (Рисунок 1.100а), для трёх точек – проведением трёх лучей из центра описанной окружности через середины трёх отрезков (Рисунок 1.100б). Основные алгоритмы построения и области применения диаграммы Вороного изложены в [2, глава 10].



а б

Рисунок 1.100 – Построение диаграммы Вороного для двух (а) и трёх точек (б)

В общем случае три попарно граничащие области Ω_i, Ω_j и Ω_k сходятся в точке, являющейся центром окружности, описанной около $\triangle A_i A_j A_k$. Следовательно, при соединении всех точек из Ω, которым соответствуют области диаграммы Вороного, имеющие общую сторону, получится граф, в котором вокруг всех внутренних граней можно описать окружность, причём внутри неё не лежит ни одна точка из исходного множества Ω. Если никакие четыре точки из Ω не лежат на одной окружности, то по определению имеем триангуляцию Делоне (Рисунок 1.101а), в противном случае для получения триангуляции Делоне достаточно каждую грань, не являющуюся треугольником, разбить произвольным образом диагоналями на треугольники (Рисунок 1.101б). Таким образом, при отсутствии четырёх точек, лежащих на одной окружности, триангуляция Делоне является **двойственным графом** для диаграммы Вороного.

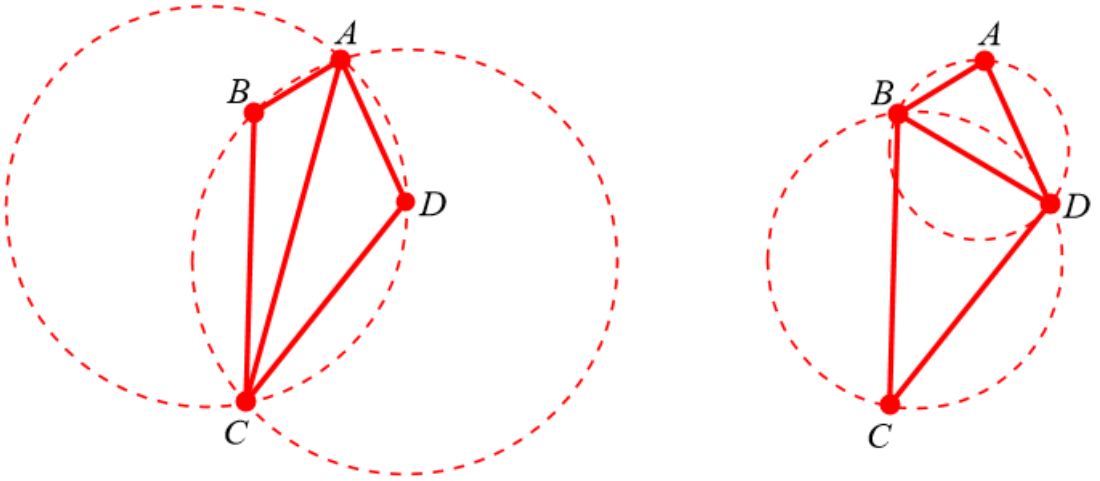


а б

Рисунок 1.101 – Построение триангуляции Делоне по диаграмме Вороного (а – при отсутствии четырёх точек на одной окружности, б – при их наличии)

Алгоритмы построения триангуляции Делоне. Существует множество алгоритмов построения триангуляции Делоне, различающихся по своей сути, используемым структурам данных [31, раздел 1.3] и реализацией отдельных процедур. Большинство из них основываются на следующем факте: если для четырёх точек A, B, C и D, образующих выпуклый четырёхугольник, триангуляция из двух треугольников $\triangle ABC$ и $\triangle ACD$ не является триангуляцией

Делоне (будем говорить, что эти два треугольника **не удовлетворяют условию Делоне**, рисунок 1.102а), то удаление ребра AC и проведение вместо него ребра BD даёт триангуляцию Делоне $\triangle ABD$ и $\triangle BCD$ (Рисунок 1.102б). Такая операция замены одного ребра другим часто называется **флипом**.



а б

Рисунок 1.102 – Флип ребра в триангуляции, не удовлетворяющей условию Делоне (а), приводит к триангуляции Делоне (б)

Для проверки условия Делоне для двух смежных треугольников могут использоваться разные подходы. Один из них заключается в том, чтобы рассматривать противолежащие углы $\angle ABC$ и $\angle CDA$, опирающиеся на ребро AC. Если имеет место нарушение условия Делоне (скажем, точка D лежит внутри окружности, описанной около $\triangle ABC$), то тогда имеет место неравенство $\angle CDA > \pi - \angle ABC \Rightarrow \angle ABC + \angle CDA > \pi$. При флипе ребра AC для новых треугольников получим $\angle DAB + \angle BCD = 2\pi - \angle ABC - \angle CDA < \pi$. Отсюда имеем следующее **условие Делоне**: в выпуклом четырёхугольнике $ABCDA$ проведение диагонали AC даёт триангуляцию Делоне тогда и только тогда, когда выполняется неравенство

$$\angle ABC + \angle CDA \le \pi,$$

причём равенство достигается в ситуации, когда все четыре точки лежат на одной окружности, а значит, проведение любой из двух диагоналей даст триангуляцию Делоне. На практике проще оперировать тригонометрическими функциями от этих углов:

$$\begin{aligned} &\angle ABC + \angle CDA \le \pi \Leftrightarrow \sin\left(\angle ABC + \angle CDA\right) \ge 0 \Leftrightarrow \\ &\Leftrightarrow \sin\angle ABC \cos\angle CDA + \cos\angle ABC \sin\angle CDA \ge 0. \end{aligned} \tag{1.96}$$

Как известно, значения этих функций выражаются через векторные операции:

$$\sin\angle ABC = \frac{\left|\overrightarrow{BA}\times\overrightarrow{BC}\right|}{\left|\overrightarrow{BA}\right|\left|\overrightarrow{BC}\right|}, \cos\angle ABC = \frac{\overrightarrow{BA}\cdot\overrightarrow{BC}}{\left|\overrightarrow{BA}\right|\left|\overrightarrow{BC}\right|},$$
$$\sin\angle CDA = \frac{\left|\overrightarrow{DC}\times\overrightarrow{DA}\right|}{\left|\overrightarrow{DC}\right|\left|\overrightarrow{DA}\right|}, \cos\angle CDA = \frac{\overrightarrow{DC}\cdot\overrightarrow{DA}}{\left|\overrightarrow{DC}\right|\left|\overrightarrow{DA}\right|}.$$

Подставляя эти значения в (1.96) и умножая обе части неравенства на $\left|\overrightarrow{BA}\right|\left|\overrightarrow{BC}\right|\left|\overrightarrow{DC}\right|\left|\overrightarrow{DA}\right|$, получим следующее неравенство:

$$\left|\overrightarrow{BA}\times\overrightarrow{BC}\right|\left(\overrightarrow{DC}\cdot\overrightarrow{DA}\right)+\left(\overrightarrow{BA}\cdot\overrightarrow{BC}\right)\left|\overrightarrow{DC}\times\overrightarrow{DA}\right|\geq 0. \qquad (1.97)$$

Разумеется, перед проверкой неравенства (1.97) можно вычислить знаки скалярных произведений. Если они оба положительны, то условие Делоне автоматически выполняется, если оба отрицательны, то оно сразу же нарушается. И только если два скалярных произведения имеют разные знаки, то тогда дополнительно вычисляются векторные произведения, и все эти числа подставляются в (1.97).

С учётом экстремальных свойств триангуляции Делоне (например, свойство 4) получаем одно важное следствие, использующее операцию флипа. Так, если в некоторой триангуляции G_1 точек множества Ω есть два смежных треугольника $\triangle ABC$ и $\triangle ACD$, $\{A,B,C,D\}\subseteq\Omega$, не удовлетворяющих условию Делоне, то обозначая через $R(\Delta)$ радиус окружности, описанной вокруг некоторого треугольника Δ, имеем $R(\triangle ABC)+R(\triangle ACD)>R(\triangle ABD)+R(\triangle BCD)$. С учётом неизменности остальных треугольников в триангуляции получим, что в результате флипа одного ребра получится новая триангуляция G_2, для которой справедливо неравенство $\mathcal{F}(G_1)>\mathcal{F}(G_2)$, где $\mathcal{F}(G)$ – общая сумма радиусов окружностей, описанных вокруг всех треугольников триангуляции G. С учётом конечного числа всех возможных триангуляций точек из Ω имеем, что при наличии смежных треугольников, нарушающих условие Делоне, флип за флипом будем получать новые триангуляции: $\mathcal{F}(G_1)>\mathcal{F}(G_2)>\mathcal{F}(G_3)>\ldots>$ $>\mathcal{F}(G_N)$, что рано или поздно приведёт к построению такой триангуляции, где всякая пара смежных треугольников удовлетворяет условию Делоне. Это означает, что получена триангуляция Делоне.

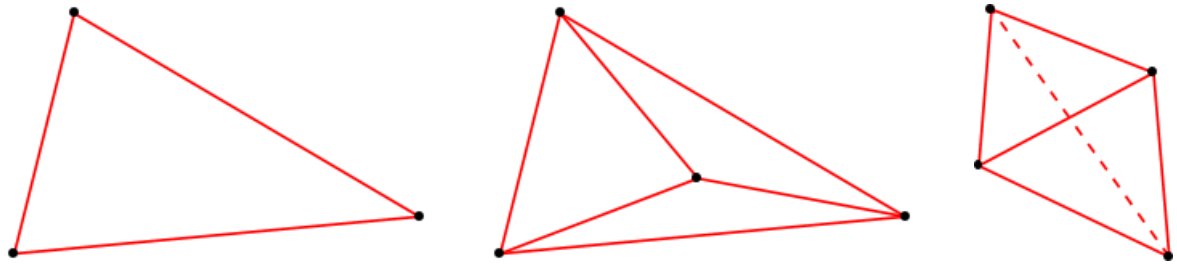
Большинство известных алгоритмов построения триангуляции Делоне можно отнести к одной из двух больших категорий: **итеративные** [31, глава 2] и **алгоритмы слияния** [31, глава 3].

Итеративные алгоритмы. Идея алгоритмов данной категории заключается в следующем. Первоначально легко строится триангуляция Делоне для некоторых 3 или 4 точек из исходного множества Ω:

- Для произвольных трёх точек треугольник с вершинами в них образует триангуляцию Делоне (Рисунок 1.103а).

- Для четырёх точек, из которых одна находится внутри треугольника, образованного тремя другими точками, также несложно построить триангуляцию Делоне (Рисунок 1.103б).

- Для четырёх точек, образующих выпуклый четырёхугольник, существует две триангуляции, из которых по крайней мере одна является триангуляцией Делоне. Проводя в этом четырёхугольнике некоторую диагональ и затем проверяя условие Делоне для двух полученных треугольников, а также при необходимости проводя флип этой диагонали, приходим к триангуляции Делоне этих четырёх точек (Рисунок 1.103в).



а б в

Рисунок 1.103 – Примеры простейших триангуляций Делоне

После этого для текущей триангуляции Делоне G проводятся следующие действия:

- Добавляется новая точка из Ω.

- Определяется её расположение относительно триангуляции G. Исходя из него проводятся новые рёбра, и получается новая триангуляция G'.

- В полученной триангуляции при необходимости проводятся флипы рёбер, что приводит к её трансформации в триангуляцию Делоне с учётом новой точки.

Таким образом, по мере добавления точек из Ω на каждой новой итерации происходит построение триангуляций Делоне для всё большего и большего числа точек, пока наконец не будет добавлена последняя точка из Ω и не получена окончательная триангуляция Делоне для всего множества (Рисунок 1.104).

Ниже приведён **простой итеративный алгоритм** построения триангуляции Делоне для множества точек $\{A_i\}_{i=1}^{n}$, $n \geq 4$:

1. Вначале составляется триангуляция из одного треугольника: $G := \{\triangle A_1 A_2 A_3\}$.
2. Также для текущей триангуляции нужно сохранить границу её выпуклой оболочки с заданным направлением обхода, например, левосторонним: $H := \begin{cases} A_1 A_2 A_3 A_1, \angle\left(\overrightarrow{A_1 A_2}, \overrightarrow{A_2 A_3}\right) > 0, \\ A_1 A_3 A_2 A_1, \angle\left(\overrightarrow{A_1 A_2}, \overrightarrow{A_2 A_3}\right) < 0. \end{cases}$

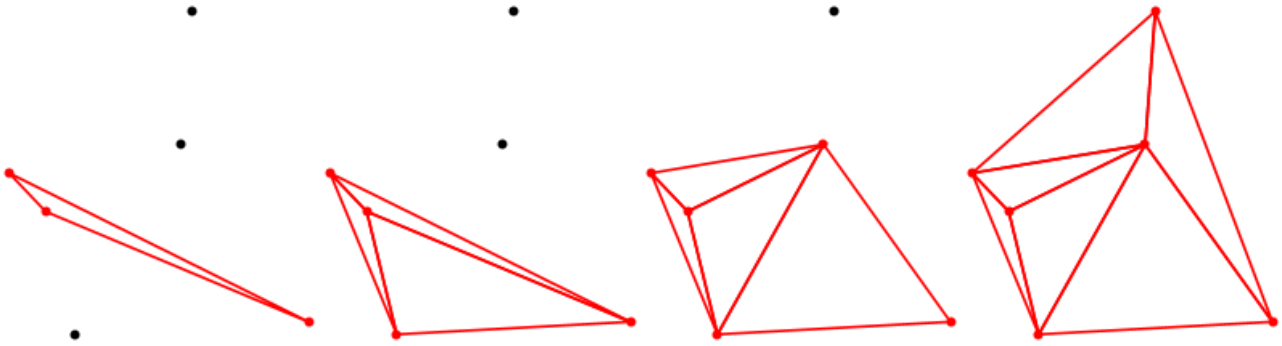


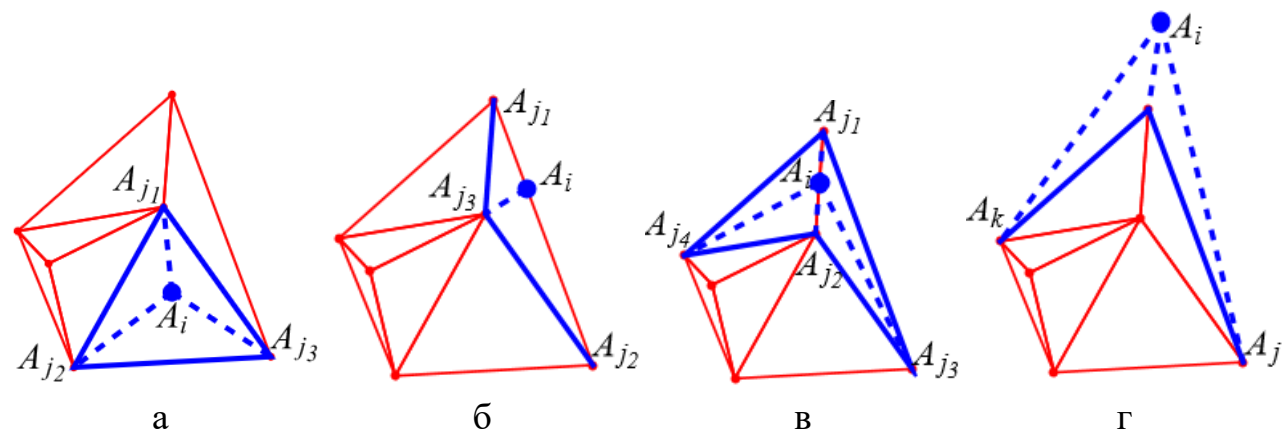
Рисунок 1.104 – Построение триангуляции Делоне итеративным алгоритмом

3. Цикл начинается с точки с индексом $i := 4$.
4. Для каждого треугольника $\Delta_j \in G$ проверяется условие $A_i \in \Delta_j$. В результате может оказаться справедливым одно из условий:

   4а. $A_i \in \triangle A_{j_1} A_{j_2} A_{j_3}$ для некоторого $\triangle A_{j_1} A_{j_2} A_{j_3} \in G$ (Рисунок 1.105а). Тогда для получения новой триангуляции необходимо провести рёбра $A_i A_{j_1}$, $A_i A_{j_2}$, $A_i A_{j_3}$, что даст редактирование множества всех треугольников: $G := \left(G \setminus \{\triangle A_{j_1} A_{j_2} A_{j_3}\}\right) \cup \{\triangle A_i A_{j_1} A_{j_2}, \triangle A_i A_{j_2} A_{j_3}, \triangle A_i A_{j_3} A_{j_1}\}$. Также определяется список рёбер, видимых из новой вершины: $V := \{A_{j_1} A_{j_2}, A_{j_2} A_{j_3}, A_{j_3} A_{j_1}\}$.

   4б. Новая точка оказалась лежащей на некотором ребре: $A_i \in A_{j_1} A_{j_2}$. Тогда проверяется, является ли это ребро граничным для всей текущей триангуляции, что эквивалентно принадлежности этого ребра замкнутой ломаной линии H.

   - Если ребро оказалось граничным для триангуляции G, то существует ровно один треугольник, для которого это ребро является граничным. Обозначая третью его вершину A_{j_3}, имеем, что нужно провести ребро $A_i A_{j_3}$, тем самым разбив треугольник $A_{j_1} A_{j_2} A_{j_3}$ на две части: $G := \left(G \setminus \{\triangle A_{j_1} A_{j_2} A_{j_3}\}\right) \cup \{\triangle A_i A_{j_1} A_{j_3},$



а б в г

Рисунок 1.105 – Обработка разных случаев в зависимости от положения новой точки: а – точка лежит внутри треугольника, б – на внешнем ребре, в – на внутреннем ребре, г – снаружи триангуляции

$\triangle A_i A_{j_2} A_{j_3}\}$, а также определить список видимых рёбер: $V :=$ $:= \{A_{j_1} A_{j_3}, A_{j_2} A_{j_3}\}$ (Рисунок 1.105б).

- В противном случае ребро является граничным для двух треугольников. Обозначив их $\triangle A_{j_1} A_{j_2} A_{j_3}$ и $\triangle A_{j_1} A_{j_2} A_{j_4}$, получаем следующие действия с нужными множествами: $G := (G \setminus$ $\setminus \{\triangle A_{j_1} A_{j_2} A_{j_3}, \triangle A_{j_1} A_{j_2} A_{j_4}\}) \cup \{\triangle A_i A_{j_1} A_{j_3}, \triangle A_i A_{j_2} A_{j_3}, \triangle A_i A_{j_1} A_{j_4}, \triangle A_i A_{j_2} A_{j_4}\}$, $V := \{A_{j_1} A_{j_3}, A_{j_2} A_{j_3}, A_{j_1} A_{j_4}, A_{j_2} A_{j_4}\}$ (Рисунок 1.105в).

4в. Точка A_i лежит снаружи триангуляции G (Рисунок 1.105г). Это означает, что из A_i виден некоторый участок ломаной линии H. Для его нахождения можно воспользоваться, например, алгоритмом со стр. 51. В результате получается некоторая ломаная линия $A_j A_{j+1} ... A_k$. После этого нужно провести рёбра от новой точки A_i ко всем вершинам этой линии, и получатся новые треугольники: $G := G \cup \{\triangle A_i A_j A_{j+1}, ..., \triangle A_i A_{k-1} A_k\}$. Кроме того, все рёбра этой ломаной нужно записать в список видимых рёбер: $V := \{A_j A_{j+1}, ..., A_{k-1} A_k\}$.

5. Если список V пуст, то происходит переход к шагу 7. Иначе из списка V извлекается первое ребро. Обозначим его $A_{j_1} A_{j_2}$. Оно является граничным для треугольника $A_i A_{j_1} A_{j_2}$, а также может оказаться граничным для некоторого другого треугольника из G.

5а. Если $A_{j_1} A_{j_2} \in H$, то есть это ребро является граничным для триангуляции, то оно никак дальше не обрабатывается и просто удаляется из V.

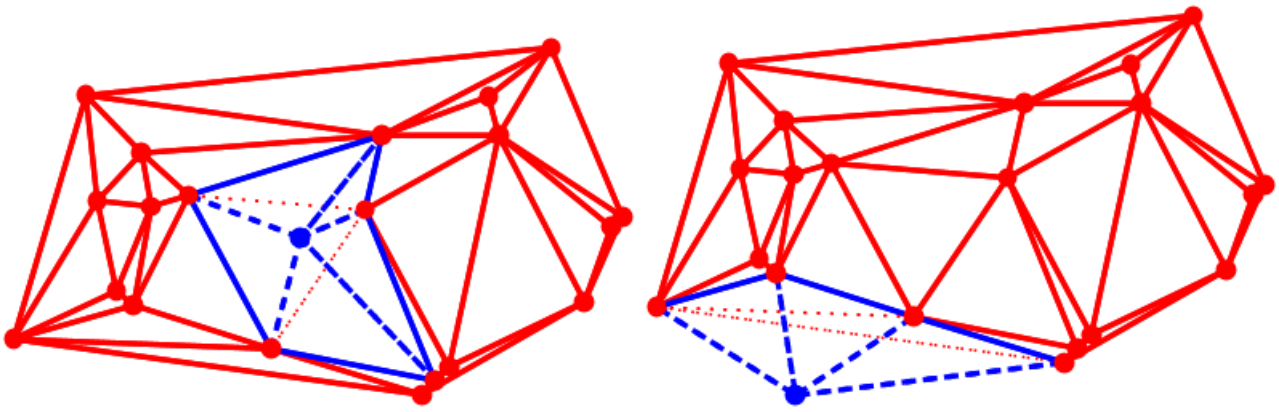
5б. Иначе существует некоторый треугольник, граничащий с $\triangle A_i A_{j_1} A_{j_2}$ по этому ребру: $\triangle A_{j_1} A_{j_2} A_k \in G$, $A_k \neq A_i$. Для этих двух треугольников необходимо проверить условие Делоне, например, по формуле (1.97). При его выполнении текущее ребро $A_{j_1} A_{j_2}$ удаляется из V, а при нарушении необходимо сделать флип этого ребра на ребро $A_i A_k$, что заменяет два треугольника на новые: $G := \left(G \setminus \{\triangle A_i A_{j_1} A_{j_2}, \triangle A_{j_1} A_{j_2} A_k\}\right) \cup$ $\cup \{\triangle A_i A_{j_1} A_k, \triangle A_i A_{j_2} A_k\}$. Кроме того, обновляется список видимых рёбер: $V := \left(V \setminus \{A_{j_1} A_{j_2}\}\right) \cup \{A_{j1} A_k, A_k A_{j_2}\}$.

6. Возврат к шагу 5.

7. После рассмотрения всех видимых рёбер в G записана триангуляция Делоне для точек $\{A_j\}_{j=1}^{i}$. Если $i = n$, то алгоритм завершается с возвратом списка G. В противном случае происходит переход к следующей точке: $i := i + 1$, возврат к шагу 4.

Для доказательства корректности этого алгоритма достаточно показать, что после добавления каждой точки A_i и выполнения шагов 4–7 действительно получается триангуляция Делоне точек $\{A_j\}_{j=1}^{i}$. ◆ Ясно, что после добавления новых треугольников на шаге 4 не нужно рассматривать пары старых треугольников, которые согласно предыдущей итерации удовлетворяют условию Делоне. Но также не нужно проверять условие Делоне для двух смежных новых треугольников. Действительно, для всякой пары таких смежных треугольников имеет место два острых угла, опирающихся на общее ребро (на рисунках 1.105 на каждое из синих пунктирных рёбер, оказавшихся внутри новой триангуляции, опираются по два острых угла).

Кроме того, несложно видеть, что в ходе выполнения шага 5 видимые рёбра образуют полигон (если точка A_i оказалась внутренней для G, рисунок 1.106а) или ломаную линию, которую также можно дополнить до полигона новыми внешними рёбрами (если A_i оказалась граничной или внешней для G, рисунок 1.106б). На рисунках 1.106 новая вершина и текущие видимые рёбра отмечены синим цветом, новые рёбра – синими пунктирными линиями. Внутри этого полигона лежат треугольники с общей вершиной A_i, причём всякая пара смежных треугольников $\triangle A_i A_{j_1} A_{j_2}$ и $\triangle A_i A_{j_2} A_{j_3}$ заведомо удовлетворяют условию Делоне, потому что иначе получилось бы, что условию Делоне удовлетворяют треугольники $\triangle A_i A_{j_1} A_{j_3}$ и $\triangle A_{j_1} A_{j_2} A_{j_3}$, и ребро $A_{j_1} A_{j_3}$, которое было видимым на предыдущем шаге, не было бы удалено (на рисунках 1.106 такие рёбра отмечены красным пунктиром).



а б

Рисунок 1.106 – Редактирование триангуляции Делоне при добавлении новой точки (а – новая точка лежит внутри исходной триангуляции, б – снаружи неё)

Таким образом, имеем следующее:

- В ходе выполнения шага 5 все рёбра можно разделить на четыре категории: граничные (внешние) рёбра всей триангуляции, рёбра с концом в точке A_i, видимые рёбра из списка V и другие внутренние рёбра.

- Для каждого внешнего ребра существует только один треугольник, для которого оно является граничным.

- Рёбра с концом в точке A_i, как показано выше, являются граничными для пар треугольников, заведомо удовлетворяющих условию Делоне.

- Другие внутренние рёбра находятся за пределами полигона, образованного видимыми рёбрами, значит, они являются граничными для пар треугольников, пока что не изменявшихся на текущей итерации, а следовательно, удовлетворяющих условию Делоне по построению триангуляции для предыдущих i – 1 точек.

Значит, проверять условие Делоне необходимо только для треугольников, граничащих по видимым рёбрам, что и происходит в изложенном выше алгоритме.

Также важно отметить, что список V рано или поздно окажется пустым, ведь на шаге 5 происходит удаление внутренних рёбер триангуляции, не инцидентных точке A_i. При удалении всех таких рёбер в списке V окажутся только граничные рёбра триангуляции G, которые согласно шагу 5а подлежат окончательному удалению из списка V (Рисунок 1.107). Таким образом, на шагах 4–7 рано или поздно действительно получается триангуляция Делоне для точек $\{A_j\}_{j=1}^{i}$ на основе триангуляции Делоне, полученной на предыдущей итерации. ■

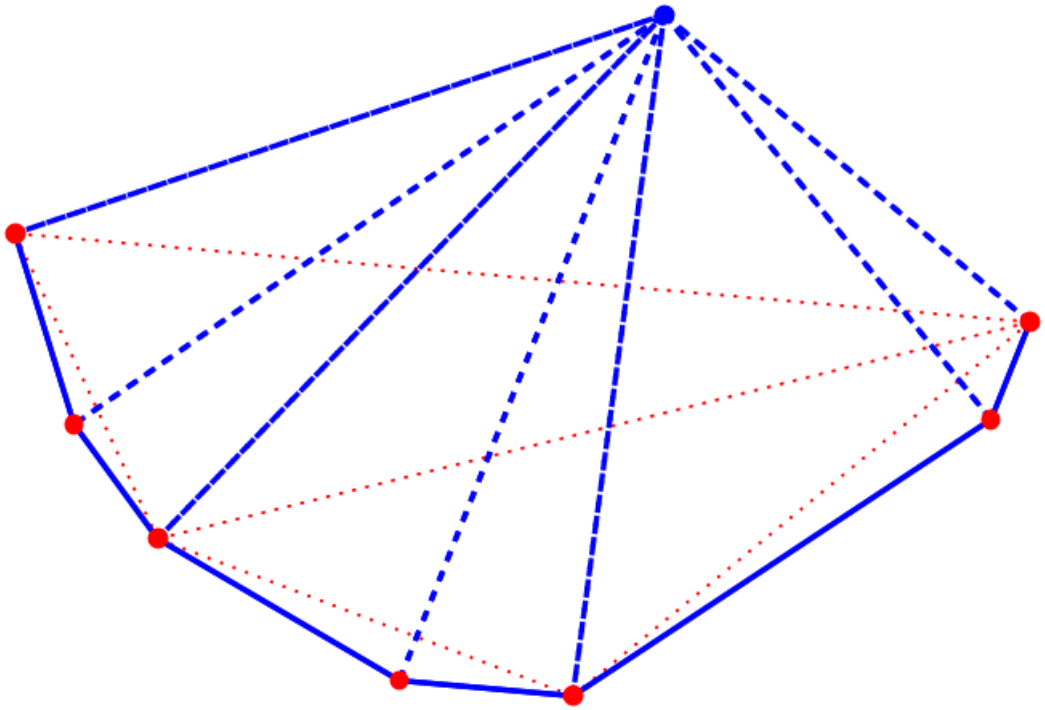


Рисунок 1.107 – Ситуация, когда удаляется максимальное число рёбер

Оценивая временную асимптотику, имеем, что для каждой из новых точек A_i, $i = \overline{4,n}$, происходят следующие операции:

- перебор всех треугольников текущей триангуляции, число которых равно $O(i)$;

- вычисление начального списка V, котором в худшем случае (когда новая точка оказывается внешней для текущей триангуляции) вычисляется за время $O(k) = O(i)$ точек, где k – число точек в списке H;

- перебор всех видимых рёбер, в худшем случае среди которых могут оказаться все внутренние рёбра триангуляции, происходит за время $O(i)$.

Также важно учитывать операции, связанные с нахождением нужных элементов триангуляции. Так, в данном алгоритме на шаге 4 для каждого треугольника из G необходимо получать список его вершин, а на шаге 5 для каждого ребра из V – список вершин, углы при которых опираются на это ребро. При использовании подходящих структур данных эти операции выполняются за время $O(1)$, однако для хранения этих данных необходим линейный объём памяти $O(i)$. Более подробный обзор возможных структур данных, которыми можно представить триангуляцию, изложен в [31, разд. 1.3].

Отсюда получается, что в ходе каждой из $O(n)$ итераций происходят операции, общее время выполнения которых можно оценить линейным временем $O(i)$, где i – номер текущей точки. Это в итоге даёт асимптотику $O(n^2)$.

Алгоритмы слияния. Алгоритмы этой категории основаны на делении исходного множества точек на некоторые подмножества, для которых легко построить триангуляции Делоне, и последующем слиянии этих триангуляций в одну целую. Основная сложность в таких алгоритмах заключается именно в реализации этапа слияния, который заключается в проведении новых и, при необходимости, удалении некоторых старых рёбер.

Как и в любом другом алгоритме, действующем по принципу «разделяй и властвуй», целесообразно осуществлять деление всего множества точек на примерно равные части. При рассмотрении n точек, $n \geq 3$, $n \neq 5$, может использоваться, например, следующая **схема разделения** [31, с. 49]:

- При $n \leq 4$ просто выполняется построение триангуляции Делоне для всех точек (см. рисунок 1.103).

- При $6 \leq n \leq 11$ всё множество точек делится на два подмножества (из 4 точек каждое при $n = 8$ либо из 3 и $n - 3$ точек в противном случае), для одного из которых сразу можно построить триангуляцию, второе же при необходимости разбивается ещё на два подмножества. После этого происходит слияние полученных триангуляций.

- При $n \geq 12$ происходит деление множества точек на примерно равные части из $\lfloor n/2 \rfloor$ и $\lceil n/2 \rceil$ точек, для которых рекурсивно выполняется эта же схема деления, построения триангуляции для каждой части и слияния этих триангуляций. В итоге происходит слияние двух больших триангуляций.

Нетрудно видеть, что при такой схеме деления и слияния время работы алгоритма в целом оценивается асимптотикой $O(f(n) \log n)$, где $f(n)$ – общая трудоёмкость всех операций деления и слияния, выражаемая функцией от числа точек в разделяемом или объединяемом множестве.

Один из алгоритмов слияния предложили Д.-Т. Ли и Б. Дж. Шектер в 1980 году [37]. В нём происходит слияние триангуляций G_L и G_R левого $\{L_i\}_{i=1}^{n_1}$ и правого множества точек $\{R_j\}_{j=1}^{n_2}$, представленных списком своих рёбер. Оно происходит за несколько этапов:

I. Объединение выпуклых оболочек двух множеств. Пусть H_L – это выпуклая оболочка для левого множества, H_R – выпуклая оболочка для правого, причём обе имеют левосторонний обход. Тогда имеет место следующий **алгоритм их объединения**:

1. В выпуклой оболочке H_L левого множества точек L_i и выпуклой оболочке H_R правого множества точек R_j выбираются соответственно самая правая и самая левая вершины. Обозначим их индексы через i_1 и j_1.

2. В качестве исходных концов **нижней** касательной L' и R' возьмём две крайние точки: $i' := i_1,\ j' := j_1,\ L' := L_{i'},\ R' := R_{j'}$.

3. Пока для следующей после R' вершины правой выпуклой оболочки выполняется неравенство $\angle\left(\overrightarrow{L'R'}, \overrightarrow{L'R_{j'+1}}\right) < 0$, нижняя касательная сдвигается к ней: $j' := j' + 1,\ R' := R_{j'}$.

4. Как только оказалось справедливым неравенство $\angle\left(\overrightarrow{L'R'}, \overrightarrow{L'R_{j'+1}}\right) \geq 0$, необходимо проверить аналогичное неравенство для предыдущей перед L' вершины левой выпуклой оболочки: $\angle\left(\overrightarrow{L'R'}, \overrightarrow{L'L_{i'-1}}\right) < 0$. Если оно выполняется, то происходит сдвиг касательной к этой вершине: $i' := i' + 1,\ L' := L_{i'}$, а также возврат к шагу 3. Иначе алгоритм даёт нижнюю касательную $L_B R_B := L'R'$.

5. Для нахождения **верхней** касательной повторяются шаги 2–4. При этом подчёркнутые слова и неравенства следует заменить на противоположные по смыслу, а знаки «плюс» и «минус» поменять местами. Обозначим верхнюю касательную через $L_T R_T$.

6. Последовательность вершин итоговой выпуклой оболочки получается путём объединения последовательностей $R_B ... R_T$ и $L_T ... L_B$, взятых из исходных выпуклых оболочек: $H_L \cup H_R = R_B ... R_T L_T ... L_B R_B$.

В результате этого алгоритма получается выпуклая оболочка для объединения двух множеств, а также две касательные между выпуклыми оболочками двух исходных множеств $\{L_i\}_{i=1}^{n_1}$ и $\{R_j\}_{j=1}^{n_2}$, являющиеся крайними рёбрами, которыми можно соединить точки разных множеств (Рисунок 1.108). Несложно видеть, что он работает за время $O(n)$, где n – общее число точек: $n = n_1 + n_2$, так как при нахождении нижней касательной в ходе шага 3 осуществляется проход через вершины правой выпуклой оболочки $R_{j'}$, $R_{j'+1}$, …, R_B, а на шаге 4 – через вершины левой выпуклой оболочки $L_{i'}$, $L_{i'-1}$, …, L_B. Аналогично рассматривается и построение верхней касательной.

II. Объединение двух триангуляций, которое заключается в проведении новых рёбер, соединяющих две исходные триангуляции, а также удалении тех из старых рёбер, которые дают нарушение условия Делоне для новых треугольников. Зная из предыдущего этапа нижнюю и верхнюю касательные $L_B R_B$ и $L_T R_T$, можно выполнить следующий **алгоритм объединения**:

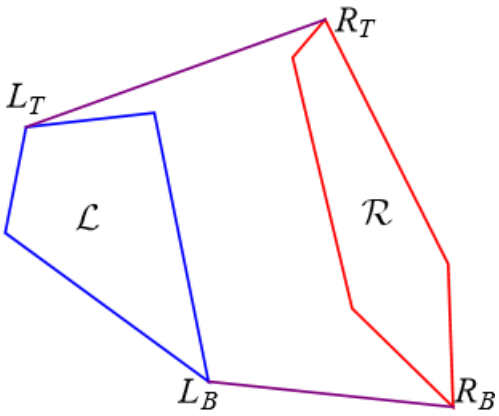


Рисунок 1.108 – Объединение двух
выпуклых оболочек при помощи касательных

1. Объявляется множество рёбер итоговой триангуляции: $G := G_L \cup G_R$.
2. В G добавляется новое ребро, соответствующее текущей нижней касательной $L_B R_B$. Также задаются две вспомогательные логические переменные: $A :=$ *ложь*, $B :=$ *ложь*.
3. Для правого конца R_B составляется список всех смежных с ним вершин $\{R_1, R_2, ..., R_k\}$ в таком порядке, чтобы векторы $\overrightarrow{R_B R_1}$, $\overrightarrow{R_B R_2}$, …, $\overrightarrow{R_B R_k}$ были отсортированы в порядке поворота **по часовой стрелке**, начиная от вершины $R_1 = L_B$.
4. Рассматривается первая вершина после L_B: $i := 2$. Если точка R_2 расположена правее ребра $L_B R_B$, то происходит переопределение вспомогательной переменной: $A :=$ *истина*, а также переход к шагу 6.
5. Если $i < k$, точка R_{i+1} расположена левее ребра $L_B R_B$, а также нарушается условие Делоне для треугольников $\triangle L_B R_B R_i$ и $\triangle R_B R_i R_{i+1}$, то из множества G удаляется ребро $R_B R_i$, происходит переход к следующей вершине: $i := i + 1$, и этот же шаг повторяется. Таким образом, происходит последовательное удаление рёбер, пока для следующего не станет выполняться условие Делоне, либо оно не окажется правее текущей касательной, либо не окажется последним.
6. Для левого конца L_B составляется список всех смежных с ним вершин $\{L_1, L_2, ..., L_l\}$ в таком порядке, чтобы векторы $\overrightarrow{L_B L_1}$, $\overrightarrow{L_B L_2}$, …, $\overrightarrow{L_B L_l}$ были отсортированы в порядке поворота **против часовой стрелки**, начиная от вершины $L_1 = R_B$.
7. Рассматривается первая вершина после R_B: $j := 2$. Если точка L_2 расположена правее ребра $L_B R_B$, то происходит переопределение вспомогательной переменной: $B :=$ *истина*, а также переход к шагу 9.
8. Если $j < l$, точка L_{j+1} лежит левее ребра $L_B R_B$, а также нарушается условие Делоне для треугольников $\triangle R_B L_B L_j$ и $\triangle L_B L_j L_{j+1}$, то из

множества G удаляется ребро $L_B L_j$, происходит переход к следующей вершине: $j := j + 1$, и этот же шаг повторяется.

9. Если A = *истина*, то сдвигается левый конец нижней касательной: $L_B := L_j$, а также происходит переход к шагу 12.
10. Если B = *истина*, то сдвигается правый конец нижней касательной: $R_B := R_i$, а также происходит переход к шагу 12.
11. Если шаги 9 и 10 не выполнены, то может сдвинуться как правый конец, так и левый. Может иметь место ровно один из следующих случаев:
    - 11а. $R_i \in \triangle L_B R_B L_j \Rightarrow$ происходит сдвиг правого конца: $R_B := R_i$.
    - 11б. $L_j \in \triangle L_B R_B R_i \Rightarrow$ происходит сдвиг левого конца: $L_B := L_j$.
    - 11в. Два смежных треугольника $\triangle L_B R_B L_j$ и $\triangle R_B R_i L_j$ удовлетворяют условию Делоне. Тогда сдвигается левый конец касательной: $L_B := L_j$.
    - 11г. Два смежных треугольника $\triangle L_B R_B L_j$ и $\triangle R_B R_i L_j$ не удовлетворяют условию Делоне. Тогда сдвигается правый конец касательной: $R_B := R_i$.
12. Если произошло совпадение новой нижней касательной с верхней касательной $L_T R_T$, то она добавляется во множество G, и алгоритм завершается возвратом множества G. В противном случае происходит возврат к шагу 2.

Таким образом, происходит постепенное заполнение пространства между двумя триангуляциями путём проведения рёбер, идущих снизу вверх от нижней касательной к верхней. На рисунке 1.109 рёбра двух исходных триангуляций окрашены в красный цвет, новые – в синий, причём удалённые рёбра обозначены пунктиром.

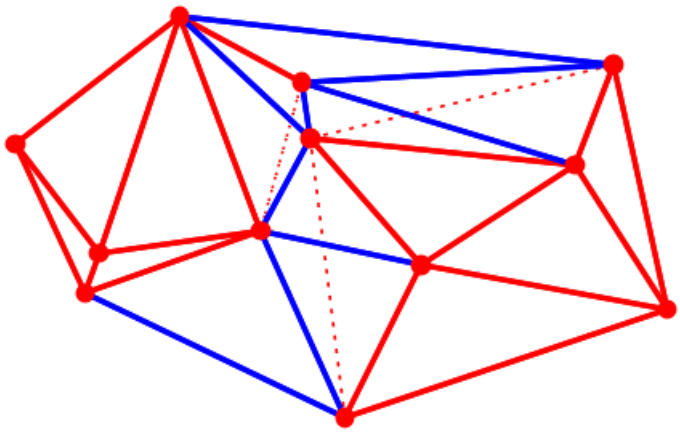


Рисунок 1.109 – Слияние двух триангуляций

Время работы этого алгоритма оценивается следующим образом. Каждое ребро удаляется не более одного раза на шагах 5 и 8, следовательно, за всё время выполнения алгоритма эти шаги повторяются $O(n)$ раз. Кроме того,

число новых рёбер также равно $O(n)$. При грамотной организации структуры данных, где хранятся последовательности вершин-соседей, получаемых на шагах 3 и 6, можно добиться того, что эти шаги будут выполняться за время $O(k)$ и $O(l)$. Действительно, для каждой вершины некоторой триангуляции можно хранить список её соседей, отсортированных в порядке поворота по или против часовой стрелки относительно неё. Инициализация этого списка происходит в простейших триангуляциях, обычно выполняющихся для множеств из трёх или четырёх точек (см. рисунок 1.103) за время $O(1)$. Также необходима реализация следующих операций:

- поиск нужного вектора, который может быть произведён за время $O(\log d)$, где d – текущее число соседей для заданной вершины;
- проход по всему списку в обоих направлениях за время $O(d)$;
- добавление нового соседа за время $O(\log d)$.

Таким образом, окончательно получаем, что представленный выше алгоритм работает за время $O(n)$. А весь алгоритм построения триангуляции n точек, использующий такое слияние, работает за время $O(n \log n)$.

1.6. Алгоритмы геометрической оптики

Неотъемлемой частью моделирования реалистичных изображений является моделирование света и световых эффектов. Как известно из школьного курса физики, свет имеет двойственную природу: волновую и корпускулярную. В приложениях, связанных с компьютерной графикой, как правило, используется корпускулярная природа света, заключающаяся в прямолинейном движении фотонов. Таким образом, для моделирования света используются **лучи**, испускаемые из некоторого источника света и характеризующиеся прежде всего направлением.

Для моделирования света и световых эффектов используются в первую очередь следующие законы геометрической оптики:

1. Прямолинейность распространения света в однородной среде: свет всегда распространяется по некоторому лучу (**принцип Ферма**).
2. Отражение от зеркальной поверхности.
3. Преломление на границе разделения сред.
4. Поглощение света.
5. Обратимость хода светового луча.
6. Независимость хода световых лучей.

В различных приложениях компьютерной графики рассматриваются, в частности, следующие задачи геометрической оптики:

1. Удаление (отсечение) невидимых частей.
2. Построение тени (полутени) одного объекта на поверхность другого (на подложку).
3. Обнаружение прямого облучения видимой точки источником света.
4. Построение зеркального отражения.

1.6.1. Моделирование световых лучей

Первичные световые лучи, т.е. те, которые непосредственно испускаются источником, либо проходят мимо всех объектов и попадают неизменёнными на приёмник (глаз человека, видеокамера, другие сенсоры), либо покидают сцену незамеченными, либо попадают на поверхность некоторого объекта. В последнем случае имеют место три процесса (в некоторой степени): отражение, преломление, поглощение. После этого имеет место распределение непоглощённой световой энергии на отражённые (направленные, или идеальные, и диффузные, или рассеянные) и преломлённые (то же самое) лучи, которые отслеживаются независимо и желательно параллельно. На пути вторичных лучей тоже могут оказаться объекты, в результате чего происходит дальнейшее поглощение световой энергии и порождаются третичные лучи и т.д. А ещё поглощение световой энергии происходит при распространении света в неабсолютно прозрачной среде. Таким образом, на некотором этапе световые лучи могут сохранить так мало энергии, что дальнейшее отслеживание методом трассировки лучей оказывается нецелесообразным.

Благодаря закону независимости распространения световых лучей широкое распространение получил **метод трассировки лучей**, заключающийся в параллельном отслеживании некоторого множества световых лучей, которые, отражаясь от разных поверхностей и попадая в конечном итоге на приёмник, формируют в нём видимое изображение. Крайне важна оценка количества моделируемых лучей, чтобы выдержать компромисс между построением адекватного изображения наблюдаемой сцены и вычислительной сложностью и временем всех расчётов. В целом, при **прямой трассировке лучей** траектория отдельно взятого луча вычисляется по следующему алгоритму:

1. **Первичный луч** характеризуется некоторым порождающим его источником света, который, как и наблюдатель, может быть дальним либо ближним, а также обладает конкретной формой. Точечный источник света характеризуется либо вектором $\vec{l}$ направления на него (дальний источник), либо своим положением L (ближний источник). Соответственно, все первичные лучи, испускаемые дальним источником, обладают одним и тем же направляющим вектором, но разными началами, а для ближнего источника всё наоборот: лучи исходят из одного начала, но в разных направлениях. Таким образом, для определения конкретного луча необходимо задать недостающий элемент луча. Лучу также может быть поставлена в соответствие его **интенсивность** – количество световой энергии, переносимой лучом за единицу времени при пересечении некоторой поверхности единичной площади.

2. Происходит вычисление точек пересечения текущего луча со всеми поверхностями сцены, в т.ч. некоторым участком проективной плоскости, моделирующим приёмник. Выполняется это обычно с использованием параметрического уравнения луча: $p(t) = p_0 + t\vec{l}$. Вместе с уравнениями каждой поверхности Ξ_i получаются системы, решаемые различными способами, что в

общем случае даёт некоторое множество решений T_i. Из всех параметров $\bigcup_i T_i$ выбирается минимальный положительный, который соответствует **первому пересечению** первичного луча с некоторой поверхностью.

3. Полученная точка является началом **вторичных** лучей, которые являются результатом различных оптических процессов, в первую очередь отражения от поверхности и преломления через неё. Вторичные лучи продолжают переносить часть исходной световой энергии, другая же часть теряется перед падением первичного луча, если имеет место его распространение в поглощающей среде, а также в результате поглощения поверхностью. Долю интенсивности светового луча от начальной в зависимости от пройденного им расстояния можно вычислить исходя из **закона Бугера**:

$$a(d) = e^{-\alpha d}, \tag{1.98}$$

где d – пройденное лучом расстояние, $\alpha \geq 0$ – коэффициент поглощения среды. Энергия, попавшая на поверхность, делится на поглощённую, отражённую и преломлённую, что вычисляется исходя из соответствующих коэффициентов $k_{\text{погл}}$, $k_{\text{отр}}$ и $k_{\text{прел}}$, зависящих от оптических свойств поверхности и сумма которых равна 1. В свою очередь, отражённая и преломлённая энергия делятся между направленными и рассеянными (диффузно отражёнными) лучами: $k_{\text{отр}} = k_{\text{но}} + k_{\text{до}}$, $k_{\text{прел}} = k_{\text{нп}} + k_{\text{дп}}$.

4. Для каждого из вторичных лучей повторяются шаги 2–3, пока не выполнится одно из следующих условий:

- Луч попадает на приёмник. Тогда наблюдатель видит точку – начало этого луча. Если это точка некоторой поверхности, то на изображении сцены она окрашивается в цвет, зависящий от оптических характеристик светового луча и самой поверхности, если же произошло попадание первичного луча, то происходит построение блика на изображении.
- На шаге 2 получается пустое решение. Это означает, что луч вышел за пределы сцены, не попав на приёмник, тем самым оставшись незаметен.
- На шаге 3 при вычислении интенсивности луча при падении на поверхность по формуле (1.98) либо при её распределении между отражёнными и преломлёнными лучами после её частичного поглощения получаются значения ниже некоторого порога. В этом случае отслеживание луча прекращается в силу отсутствия видимого его влияния на итоговое изображение сцены.

Основным недостатком описанного подхода является невозможность подбора лучей, которые бы порождали лучи, непременно падающие на приёмник. Вместо этого получается множество траекторий, большинство из которых оканчивается за пределами сцены, либо отслеживание которых после отражения от разных поверхностей прекращается по причине потери слишком большой доли интенсивности. Поэтому чаще используется **обратная**

трассировка лучей, возможная благодаря принципу обратимости световых лучей, т.е. моделированию траекторий распространения световой энергии в обратном направлении.

1. Участок проективной плоскости, в котором происходит построение итогового изображения, разбивается на ячейки, из центра каждой из которых испускается луч в направлении, *противоположном* направлению на наблюдателя. На практике в качестве таких ячеек выступают пиксели дисплея, на котором происходит построение изображения. Направление на наблюдателя выражается вектором с координатным столбцом $S - h_s P$, где S – координатный столбец наблюдателя, h_s – индикатор близости наблюдателя, P – координатный столбец некоторой точки.

2. Каждый из лучей обрабатывается аналогично шагам 2–4 предыдущего алгоритма. В результате достигается одно из условий, указанных в шаге 4, что приводит либо к прекращению отслеживания луча, либо при попадании на источник света – окрашиванию ячейки, в которой лежит начало исходного луча, в некоторый цвет с учётом оптических свойств этого источника света и точки первого пересечения луча с некоторой поверхностью сцены.

Таким образом, возможно построение обратных траекторий от наблюдателя к одному или нескольким источникам света, которые вместе образуют итоговое изображение. Как и в методе прямого распространения лучей, получается множество вторичных, третичных и т.д. лучей, не влияющих на построение изображения в результате того, что они не попадают ни на какой источник света.

Отражение света. Пусть некоторый световой луч с направляющим вектором $\vec{v}$ падает в точку p некоторой поверхности (вообще говоря, криволинейной) (Рисунок 1.110). Тогда направляющий вектор отражённого луча $\vec{v}_{\text{отр}}$ удовлетворяет следующим трём положениям:

1. Три вектора $\vec{v}$, $\vec{v}_{\text{отр}}$ и вектор нормали $\vec{n} = \vec{n}(p)$, проведённый к поверхности из точки p, лежат в одной плоскости. Вектор $\vec{n}(p)$ для неявно заданной поверхности $F(x,y,z) = C$ равен $\left(\frac{\partial F(p)}{\partial x}, \frac{\partial F(p)}{\partial y}, \frac{\partial F(p)}{\partial z} \right)$, а для параметрически заданной поверхности $\vec{r} = \vec{r}(u,v)$ он равен $\frac{\partial \vec{r}(u_0, v_0)}{\partial u} \times$ $\times \frac{\partial \vec{r}(u_0, v_0)}{\partial v}$, где u_0 и v_0 – такие значения параметров, что радиус-вектор точки p равен $\vec{r}(u_0, v_0)$.

2. Падающий и отражённый лучи лежат **по одну сторону** от отражающей поверхности.

3. Угол падения равен углу отражения: $\left| \angle(\vec{n}, \vec{v}) \right| = \left| \angle(\vec{n}, \vec{v}_{\text{отр}}) \right| = \alpha$.

Положим также $\left| \vec{v}_{\text{отр}} \right| = \left| \vec{v} \right|$, т.е. будем искать именно такой направляющий вектор отражённого луча, длина которого равна длине заданного

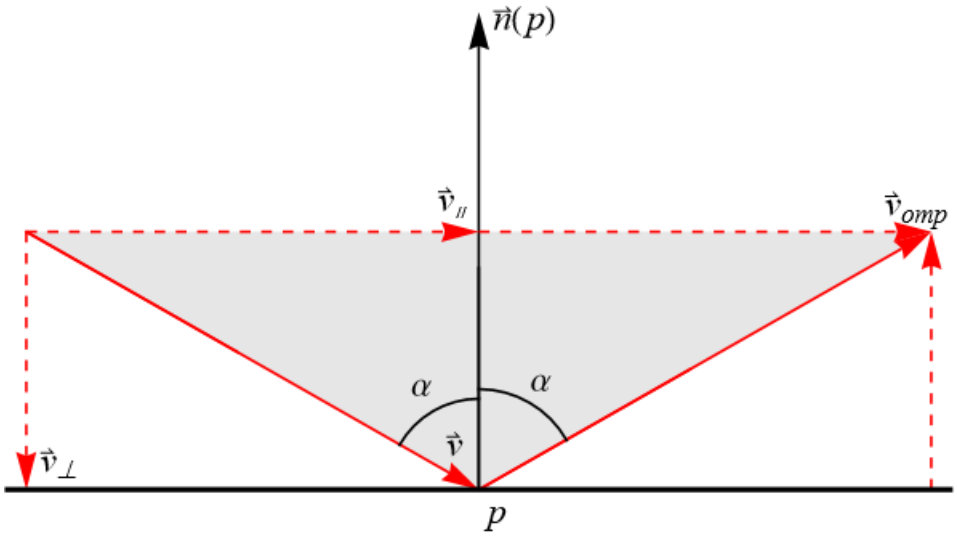


Рисунок 1.110 – Моделирование отражённого светового луча

направляющего вектора падающего луча. Для этого разложим $\vec{v}$ на две составляющие: $\vec{v} = \vec{v}_{\perp} + \vec{v}_{||}$, где $\vec{v}_{\perp} \,||\, \vec{n}$ – **нормальная составляющая**, $\vec{v}_{||} \perp \vec{n}$ – **тангенциальная составляющая** вектора $\vec{v}$. Нормальная составляющая равна проекции $\vec{v}$ на ось N с началом в точке p и направлением, задаваемым вектором $\vec{n}$. Величина вектора $\vec{v}_{\perp}$ (как величина направленного отрезка), таким образом, может быть вычислена следующим образом:

$$\operatorname{sgn}(\cos\alpha)\left|\vec{v}_{\perp}\right| = \operatorname{sgn}(\cos\alpha)\left|\cos\alpha \cdot \vec{v}\right| = \cos\alpha\left|\vec{v}\right| = \frac{\vec{v}\cdot\vec{n}}{\left|\vec{v}\right|\left|\vec{n}\right|}\left|\vec{v}\right| = \frac{\vec{v}\cdot\vec{n}}{\left|\vec{n}\right|}.$$

Отсюда сам вектор равен $\vec{v}_{\perp} = \operatorname{sgn}(\cos\alpha)\left|\vec{v}_{\perp}\right|\frac{\vec{n}}{\left|\vec{n}\right|} = \frac{\vec{v}\cdot\vec{n}}{\left|\vec{n}\right|}\frac{\vec{n}}{\left|\vec{n}\right|} = \frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}$, а $\vec{v}_{||} = \vec{v} - \frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}$. С учётом равенства серых треугольников на рисунке 1.110 приходим к выводу, что искомый вектор можно представить в виде

$$\vec{v}_{\text{отр}} = \vec{v}_{\text{отр}\perp} + \vec{v}_{\text{отр}||} = -\vec{v}_{\perp} + \vec{v}_{||} = \vec{v} - 2\frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}. \qquad (1.99)$$

Отметим следующие свойства этой формулы:

1. По ней действительно имеем $\left|\vec{v}_{\text{отр}}\right| = \left|\vec{v}\right|$, что доказывается путём нахождения скалярного квадрата обеих частей формулы (1.99).
2. Результат этой формулы не зависит от направления $\vec{n}$.
3. Результат не зависит от длины $\vec{n}$.

Таким образом, **алгоритм построения отражённого луча** выглядит следующим образом:

1. Для данного светового луча $p = p_0 + t\vec{v}$, $t > 0$, вычисляется его ближайшее к началу луча пересечение со всеми поверхностями объектов рассматриваемой сцены. Это даёт некоторый параметр t, при подстановке которого в уравнение луча получается точка p его падения на некоторую поверхность.

2. Вычисляется вектор нормали $\vec{n} = \vec{n}(p)$, проведённый из точки p, исходя из вида поверхности, на которой эта точка лежит.

3. По формуле (1.99) вычисляется направляющий вектор отражённого луча. Таким образом, получается отражённый луч с началом в точке p и направляющим вектором $\vec{v}_{\text{отр}}$, $\left|\vec{v}_{\text{отр}}\right| = \left|\vec{v}\right|$.

Преломление света. Пусть некоторый световой луч с направляющим вектором $\vec{v}$ падает в точку p некоторой поверхности (вообще говоря, криволинейной) (Рисунок 1.111). Тогда направляющий вектор преломлённого луча $\vec{v}_{\text{прел}}$ удовлетворяет следующим трём положениям:

1. Три вектора $\vec{v}$, $\vec{v}_{\text{прел}}$ и вектор нормали $\vec{n} = \vec{n}(p)$, проведённый к поверхности из точки p, лежат в одной плоскости.

2. Падающий и преломлённый лучи лежат **по разные стороны** от преломляющей поверхности.

3. Угол падения α и угол преломления γ связаны **законом Снеллиуса-Декарта**:

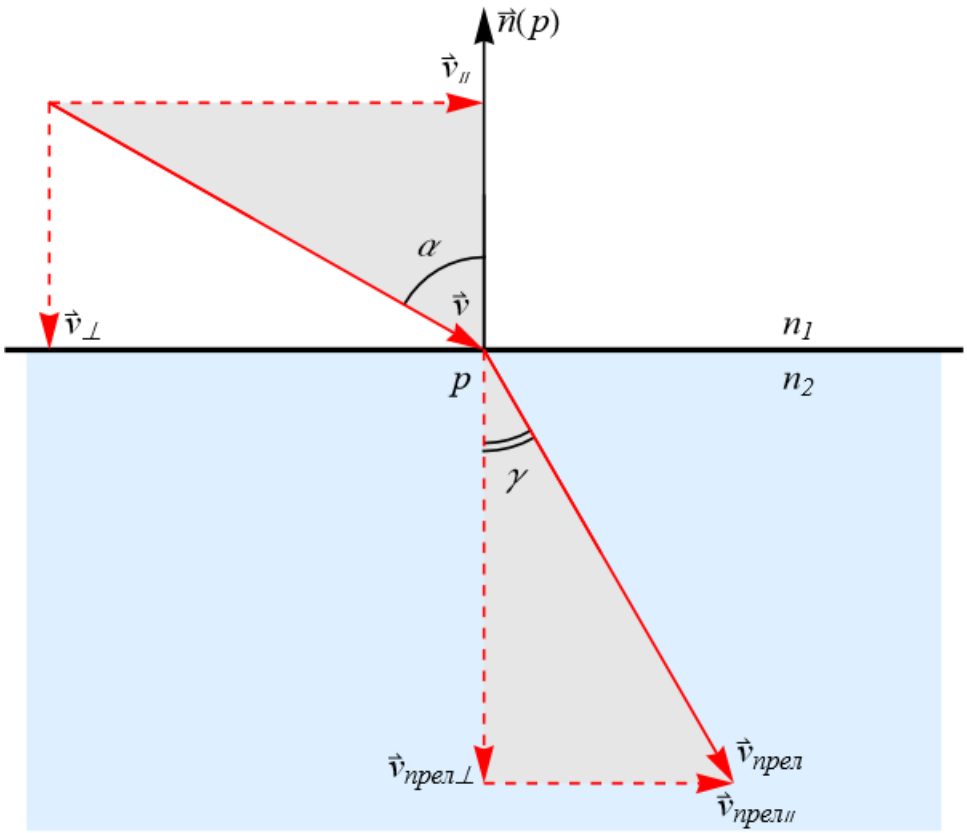


Рисунок 1.111 – Моделирование преломлённого светового луча

$$\frac{\sin\gamma}{\sin\alpha}=\frac{n_1}{n_2}=n_{\text{прел}},\qquad(1.100)$$

где n_1 и n_2 – **абсолютные показатели преломления** среды, в которой проходит первичный луч, и среды, в которой проходит преломлённый луч; $n_{\text{прел}}$ – **относительный показатель преломления** для этих двух сред.

Также положим $\left|\vec{v}_{\text{прел}}\right|=\left|\vec{v}\right|$. Разложим направляющий вектор преломлённого луча на две составляющие: $\vec{v}_{\text{прел}}=\vec{v}_{\text{прел}\perp}+\vec{v}_{\text{прел}||}$. Должно выполняться $\vec{v}_{\text{прел}\perp}\uparrow\uparrow\vec{v}_{\perp}$, $\vec{v}_{\text{прел}||}\uparrow\uparrow\vec{v}_{||}$. Таким образом, осталось только найти длины этих векторов.

На рисунке 1.111 у серых треугольников гипотенузы равны. Отсюда можно найти отношения длин векторов:

$$\frac{\left|\vec{v}_{\text{прел}\perp}\right|}{\left|\vec{v}_{\perp}\right|}=\frac{\left|\cos\gamma\right|}{\left|\cos\alpha\right|}=\frac{\left|\vec{v}\right|\left|\vec{n}\right|}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{1-\sin^2\gamma}=\frac{\left|\vec{v}\right|\left|\vec{n}\right|}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{1-n_{\text{прел}}^2\sin^2\alpha}=$$

$$=\frac{\left|\vec{v}\right|\left|\vec{n}\right|}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{1-n_{\text{прел}}^2\frac{\left|\vec{v}\times\vec{n}\right|^2}{\left|\vec{v}\right|^2\left|\vec{n}\right|^2}}=\frac{1}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2},$$

$$\frac{\left|\vec{v}_{\text{прел}||}\right|}{\left|\vec{v}_{||}\right|}=\frac{\sin\gamma}{\sin\alpha}=n_{\text{прел}}.$$

Отсюда получаем сами векторы $\vec{v}_{\text{прел}\perp}$ и $\vec{v}_{\text{прел}||}$:

$$\vec{v}_{\text{прел}\perp}=\left(\frac{1}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2}\right)\vec{v}_{\perp}=\left(\frac{1}{\left|\vec{v}\cdot\vec{n}\right|}\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2}\right)\frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}=$$

$$=\frac{\operatorname{sgn}(\vec{v}\cdot\vec{n})\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2}}{\vec{n}^2}\vec{n},$$

$$\vec{v}_{\text{прел}||}=n_{\text{прел}}\vec{v}_{||}=n_{\text{прел}}\vec{v}-n_{\text{прел}}\frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}.$$

Таким образом, окончательная формула имеет вид

$$\vec{v}_{\text{прел}}=\vec{v}_{\text{прел}\perp}+\vec{v}_{\text{прел}||}=\frac{\operatorname{sgn}(\vec{v}\cdot\vec{n})\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2}}{\vec{n}^2}\vec{n}+n_{\text{прел}}\vec{v}-$$

$$-n_{\text{прел}}\frac{\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}=n_{\text{прел}}\vec{v}+\frac{\operatorname{sgn}(\vec{v}\cdot\vec{n})\sqrt{\left|\vec{v}\right|^2\left|\vec{n}\right|^2-n_{\text{прел}}^2\left|\vec{v}\times\vec{n}\right|^2}-n_{\text{прел}}\vec{v}\cdot\vec{n}}{\vec{n}^2}\vec{n}.\qquad(1.101)$$

Как и формула (1.99), эта формула даёт направляющий вектор, длина которого равна длине исходного вектора $\vec{v}$ и который не зависит от направления и длины нормального вектора $\vec{n}$.

Замечание. Исходя из формулы (1.100), при $n_1 < n_2$, т.е. когда свет из менее оптически плотной среды попадает в более плотную, преломление будет происходить всегда. Если же $n_1 > n_2$, то тогда имеем $\sin\alpha = \dfrac{\sin\gamma}{n_{\text{прел}}} \le \dfrac{1}{n_{\text{прел}}}$. Подобное ограничение приводит к тому, что в случае, когда подкоренное выражение в формуле (1.101) является отрицательным, т.е. когда нарушается условие

$$\left|\vec{v}\right|^2 \left|\vec{n}\right|^2 - n_{\text{прел}}^2 \left|\vec{v} \times \vec{n}\right|^2 \ge 0, \qquad (1.102)$$

преломления не происходит – имеет место **полное отражение**.

Отсюда имеем **алгоритм построения преломлённого луча**:

1. Для данного светового луча $p = p_0 + t\vec{v},\ t > 0$, вычисляется точка падения p.

2. Вычисляется вектор нормали $\vec{n} = \vec{n}(p)$, проведённый из точки p исходя из вида поверхности, на которой эта точка лежит.

3. Проверяется условие (1.102) наличия преломления светового луча. Если оно не выполняется, то преломления не происходит.

4. Если условие выполнено, то по формуле (1.101) вычисляется направляющий вектор преломлённого луча. Таким образом, получается преломлённый луч с началом в точке p и направляющим вектором $\vec{v}_{\text{прел}}$, $\left|\vec{v}_{\text{прел}}\right| = \left|\vec{v}\right|$.

1.6.2. Лучевые методы моделирования оптических эффектов

Под **оптическими эффектами** понимаются зрительно наблюдаемые явления, возникающие под влиянием освещения сцены и её объектов и подлежащими визуализации. Они не представляют собой настоящие геометрические объекты, но их моделирование позволяет строить реалистичное изображение сцены, приближая его к восприятию сцены наблюдателем. К основным оптическим эффектам относятся тень, зеркальное отражение и преломлённое изображение некоторого геометрического объекта. Для их построения необходимо задать поверхность, на которой наблюдается тот или иной эффект. Имеют место следующие особенности при моделировании оптических эффектов [4, с. 56–57]:

- В зависимости от положения наблюдателя эффект может оказаться **невидимым** (например, в силу нахождения наблюдателя по другую сторону от поверхности или экранирования полученной точки эффекта), а следовательно,

не подлежащим вычислению и визуализации (при моделировании статической, не изменяющейся со временем сцены).

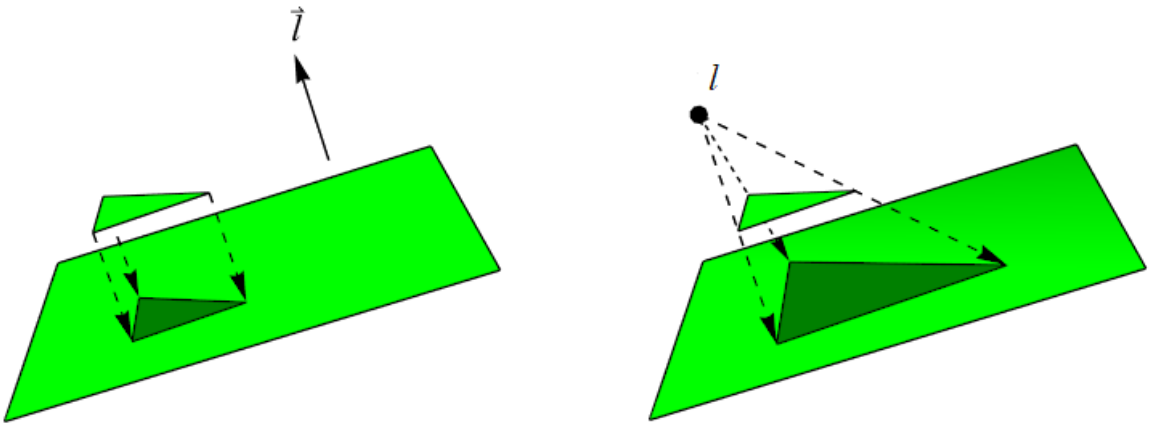
- Эффект может быть **фрагментарным**, т.е. быть порождённым не всеми точками объекта, а только теми, для которых выполнены все условия существования рассматриваемого эффекта.

- Эффект может быть **разрывным**, особенно если он строится на криволинейной поверхности. Это означает, что двум достаточно близким точкам некоторого объекта могут соответствовать точки эффекта, расположенные на разных участках поверхности или на значительном расстоянии друг от друга.

- Эффект может быть **множественным**, т.е. одна точка исходного объекта может порождать множество точек эффекта, расположенных на разных участках поверхности, для каждой из которых выполнены все условия существования этого эффекта.

Как и при моделировании световых лучей, при моделировании оптических эффектов используются параллельные вычисления, позволяющие сразу и независимо рассматривать несколько точек заданного объекта, для которого на некоторой поверхности необходимо построить тот или иной эффект.

Тень точки и видимость тени. Тень от заданной точки на заданную поверхность (**подложка** тени) порождается некоторым источником света. При дальнем освещении, характеризуемом вектором $\vec{l}$ направления на источник света, через точку p проходит световой луч с направляющим вектором $-\vec{l}$. Тенью, отбрасываемой точкой p на заданную подложку, является первое её пересечение луча $q = p - t\vec{l}$, $t > 0$. Аналогично, при ближнем освещении из точки l имеет место следующее уравнение светового луча: $q = p + t\overrightarrow{lp}$, $t > 0$. Тень некоторого объекта образуется совокупностью теней всех его точек, например, для расчёта тени полигона на плоскую подложку достаточно вычислить тени его вершин (Рисунок 1.112).



а б

Рисунок 1.112 – Построение тени, отбрасываемой на плоскую подложку:
а – при дальнем освещении, б – при ближнем освещении

Заметим, что если P – координатный столбец точки p, а L – координатный столбец точки l либо вектора $\vec{l}$, то можно записать обобщённую, универсальную формулу:

$$Q = P + t(h_l P - L), \tag{1.103}$$

где $h_l = 0$ при дальнем освещении и $h_l = 1$ при ближнем (**индикатор близости источника света**).

Важно также учитывать видимость полученной тени исходя из типа наблюдения и положения наблюдателя. В общем случае необходимым условием видимости тени является такое расположение наблюдателя, чтобы он видел освещаемую сторону подложки. Это условие эквивалентно тому, что направление на наблюдателя из точки тени q должно являться вектором, направленным от поверхности в ту же сторону, откуда падал луч (1.103). На рисунке 1.113 через $\vec{l}'$ обозначен направляющий вектор падающего светового луча, координатный вектор которого, как было показано выше, равен $h_l P - L$, а через $\vec{s}'$ – вектор направления на наблюдателя, который является постоянным для дальнего наблюдателя: $\vec{s}' = \vec{s}$ и зависит от точки q для ближнего наблюдателе: $\vec{s}' = \overrightarrow{qs}$. Несложно видеть, что координатный столбец вектора $\vec{s}'$ можно вычислить по формуле $S - h_s Q$, где S – координатный столбец точки наблюдения s либо вектора $\vec{s}$, $h_s \in \{0,1\}$ – индикатор близости наблюдателя, Q – координатный столбец точки тени.

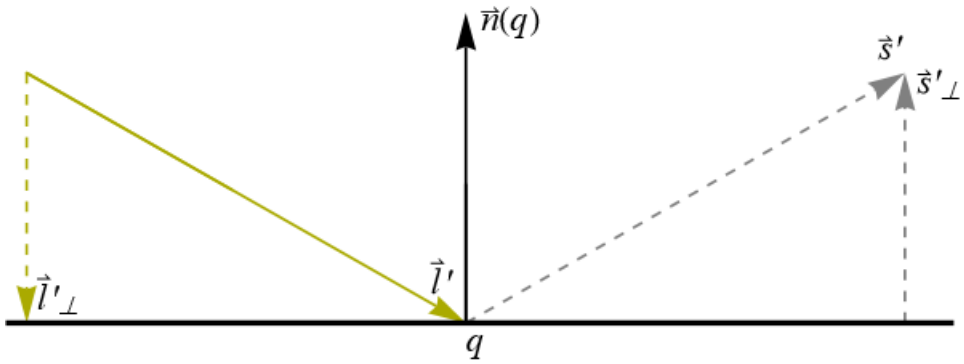


Рисунок 1.113 – Взаимное расположение направляющего вектора светового луча и вектора направления на наблюдателя при видимой тени q

Два вектора $\vec{l}'$ и $\vec{s}'$ лежат по одну сторону от подложки тогда и только тогда, когда их нормальные составляющие $\vec{l}'_\perp$ и $\vec{s}'_\perp$ противонаправлены друг другу. В свою очередь, их направление определяется скалярными произведениями $\vec{l}' \cdot \vec{n}$ и $\vec{s}' \cdot \vec{n}$. Таким образом, окончательно получаем следующее **условие видимости тени**: при отсутствии других объектов, которые могли бы экранировать точку q (в том числе другие участки самой

подложки), тень видна тогда и только тогда, когда числа $\vec{l}' \cdot \vec{n}$ и $\vec{s}' \cdot \vec{n}$ имеют разные знаки. Это можно также записать в виде следующей формулы:

$$\operatorname{sgn}(N^T(h_l P - L)) = -\operatorname{sgn}(N^T(S - h_s Q)), \tag{1.104}$$

где N – координатный столбец вектора нормали $\vec{n} = \vec{n}(q)$, отложенного от точки тени. Легко видеть, что этот вектор может иметь произвольную длину и одно из двух возможных направлений. Также в формуле (1.104) важно, чтобы ни одно из двух выражений не оказалось равным нулю (особенно в случае плоской подложки): равенство $N^T(h_l P - L) = 0$ означает, что световой луч направлен параллельно плоской подложке, а $N^T(S - h_s Q) = 0$ свидетельствует о том, что для наблюдателя плоская подложка вырождается в линию (наблюдатель смотрит в её торец).

Зеркальное отражение точки. Рассмотрим точку p и идеально отражающую поверхность П при дальнем наблюдателе, направление на которого выражается вектором $\vec{s}$. Пусть некоторый световой луч, проходя через точку p, падает на поверхность П в точке q. Тогда точка q является **наблюдаемым отражением** точки p тогда и только тогда, когда отражённый луч попадает на дальнего наблюдателя (Рисунок 1.114). Для этого отражённый луч должен обладать направляющим вектором, сонаправленным $\vec{s}$. Отсюда и из формулы отражённого луча (1.99) следует

$$\lambda \vec{s} = \overrightarrow{pq} - 2\frac{\overrightarrow{pq} \cdot \vec{n}}{\vec{n}^2}\vec{n}, \tag{1.105}$$

где $\lambda > 0$, $\vec{n} = \vec{n}(q)$ – вектор нормали, проведённый к П из точки q. Для произвольной поверхности это уравнение относительно λ и q часто решается численными методами и имеет, вообще говоря, множество решений. Отсюда

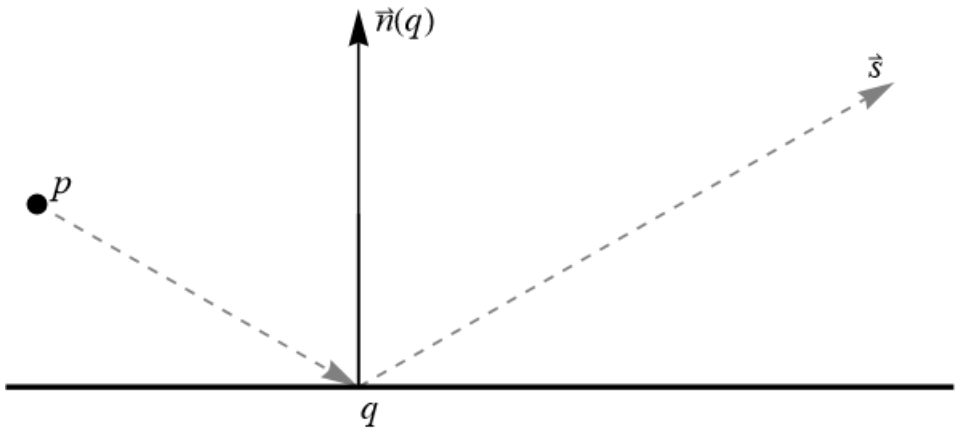


Рисунок 1.114 – Построение зеркального отражения точки

следует возможность множества отражений одного и того же объекта [4, рисунок 40].

Рассмотрим случай, когда П – это плоскость. Тогда $\vec{n} = \text{const}$ можно считать известным. Пусть также известна некоторая точка $o \in \Pi$. Умножим скалярно обе части уравнения (1.105) на $\vec{n}$:

$$\lambda\vec{s}\cdot\vec{n} = \overrightarrow{pq}\cdot\vec{n} - 2\frac{\overrightarrow{pq}\cdot\vec{n}}{\vec{n}^2}\vec{n}\cdot\vec{n},$$

$$\lambda\vec{s}\cdot\vec{n} = \overrightarrow{pq}\cdot\vec{n} - 2\overrightarrow{pq}\cdot\vec{n},$$

$$\lambda\vec{s}\cdot\vec{n} = -\overrightarrow{pq}\cdot\vec{n},$$

$$\lambda\vec{s}\cdot\vec{n} = -(\overrightarrow{po} + \overrightarrow{oq})\cdot\vec{n}.$$

Так как обе точки o и q лежат на плоскости П, то $\overrightarrow{oq} \perp \vec{n}$, откуда получим $\lambda\vec{s}\cdot\vec{n} = \overrightarrow{op}\cdot\vec{n} \Rightarrow \lambda = \dfrac{\overrightarrow{op}\cdot\vec{n}}{\vec{s}\cdot\vec{n}}$. Отметим, что для наблюдения отражения необходимо выполнение неравенства $\lambda > 0$ (так называемое **действительное** отражение). Это эквивалентно тому, что **для видимости зеркального отражения точка *p* и дальний наблюдатель должны располагаться по одну сторону от П**. В противном случае имеет место либо наблюдение из-за другой стороны плоскости ($\lambda < 0$, **мнимое** отражение), либо наблюдение параллельно плоскости (в результате чего плоскость с точки зрения такого наблюдателя **вырождается** в прямую), и отражение точки построению не подлежит.

Зная значение λ, несложно вычислить искомую точку q. Используя равенство $\overrightarrow{pq}\cdot\vec{n} = -\overrightarrow{op}\cdot\vec{n}$, выразим q из (1.105):

$$\lambda\vec{s} = \overrightarrow{pq} + 2\frac{\overrightarrow{op}\cdot\vec{n}}{\vec{n}^2}\vec{n},$$

$$\overrightarrow{pq} = \lambda\vec{s} - 2\frac{\overrightarrow{op}\cdot\vec{n}}{\vec{n}^2}\vec{n},$$

$$q = p + \lambda\vec{s} - 2\frac{\overrightarrow{op}\cdot\vec{n}}{\vec{n}^2}\vec{n}.$$

Записывая последнее равенство в матричном виде, получим формулу для координатного столбца искомой точки:

$$Q = P + \lambda S + 2\frac{N^T(O-P)}{N^T N}N.$$

Для ближнего наблюдения из точки s уравнение (1.105) переписывается в виде

$$\lambda\overrightarrow{qs} = \overrightarrow{pq} - 2\frac{\overrightarrow{pq}\cdot\vec{n}}{\vec{n}^2}\vec{n}.$$

Аналогично тому, как сделано выше, вычисляется коэффициент в левой части: $\lambda = \dfrac{\overrightarrow{op}\cdot\vec{n}}{\overrightarrow{os}\cdot\vec{n}}$. Если получилось, что $\lambda > 0$, то искомая точка q вычисляется согласно следующим формулам:

$$\lambda\overrightarrow{qs} = \overrightarrow{pq} + 2\frac{\overrightarrow{op}\cdot\vec{n}}{\left|\vec{n}\right|^2}\vec{n},$$

$$\lambda(S-Q) = Q - P + 2\frac{N^T(P-O)}{N^TN}N,$$

$$(\lambda+1)Q = \lambda S + P + 2\frac{N^T(O-P)}{N^TN}N,$$

$$Q = \frac{1}{\lambda+1}\left(P + \lambda S + 2\frac{N^T(O-P)}{N^TN}N\right).$$

Случаи дальнего и ближнего наблюдения можно обобщить, используя индикатор близости наблюдателя:

$$\lambda = \frac{N^T(P-O)}{N^T(S-h_sO)}. \qquad (1.106)$$

Если значение λ оказалось положительным, то далее вычисляется координатный столбец Q:

$$Q = \frac{1}{h_s\lambda+1}\left(P + \lambda S + 2\frac{N^T(O-P)}{N^TN}N\right). \qquad (1.107)$$

Окончательно **алгоритм вычисления видимого отражения** точки p от плоскости П выглядит следующим образом:

1. Проверяется видимость отражения: отражение является видимым тогда и только тогда, когда числа $N^T(P-O)$ и $N^T(S-h_sO)$ одного знака и не равны нулю. Если это утверждение не выполняется, то отражение не является видимым и построению не подлежит.

2. Если отражение оказывается видимым, то оно вычисляется по формулам (1.106) и (1.107).

Преломлённое изображение точки. Рассмотрим две среды с показателями преломления n_1 и n_2, $\frac{n_1}{n_2} = n_{\text{прел}}$, разделённые идеально преломляющей поверхностью П. Пусть в первой среде расположена точка p, и имеет место дальнее наблюдение с наблюдателем, направление на которого выражается вектором $\vec{s}$. Пусть также некоторый световой луч, проходя через точку p, падает на поверхность П в точке q. Тогда точка q является **наблюдаемым преломлённым изображением** точки p тогда и только тогда, когда преломлённый луч сонаправлен $\vec{s}$ (Рисунок 1.115). Пользуясь принципом обратимости световых лучей, направим световой луч в противоположном направлении $-\vec{s}$ из второй среды в точку q. Тогда направляющий вектор полученного преломлённого луча сонаправлен $\overrightarrow{qp}$, что можно выразить при помощи формулы преломления светового луча (1.101):

$$\lambda\overrightarrow{qp} = -n'_{\text{прел}}\vec{s} - \frac{\operatorname{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} - n'_{\text{прел}}\vec{s}\cdot\vec{n}}{\vec{n}^2}\vec{n}, \qquad (1.108)$$

где $n'_{\text{прел}}$ – относительный показатель преломления, равный $n_2/n_1 = 1/n_{\text{прел}}$.

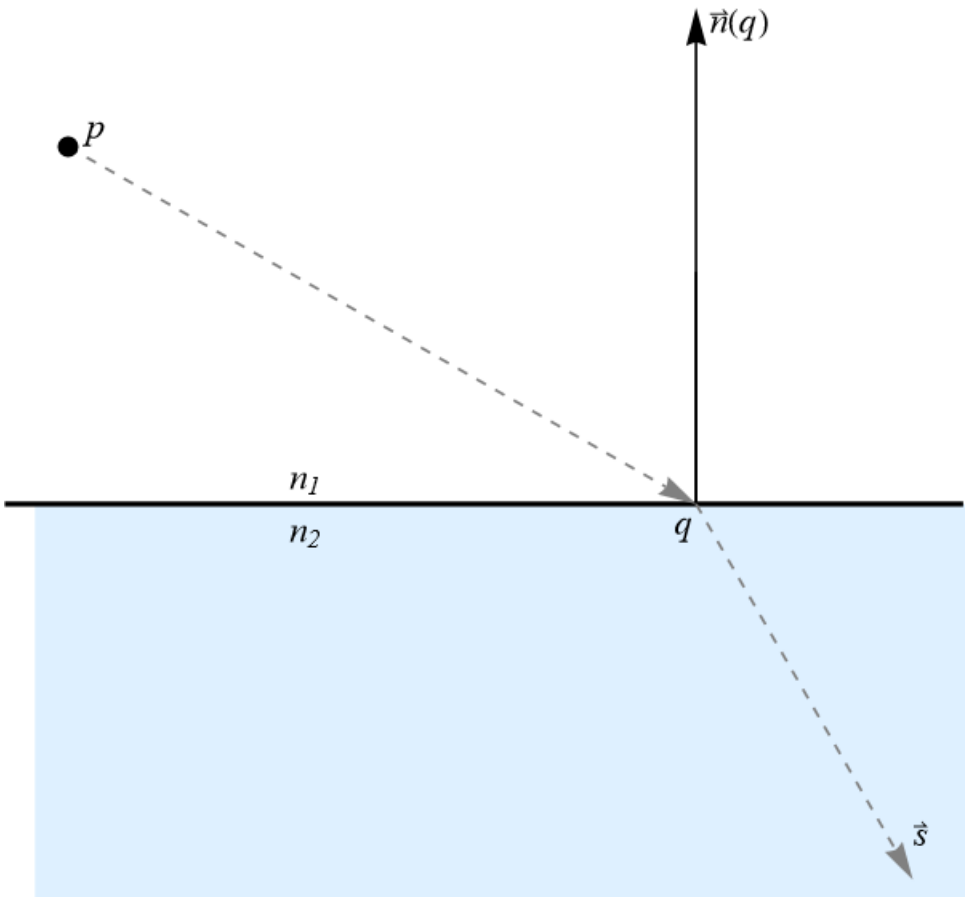


Рисунок 1.115 – Построение преломлённого изображения точки для дальнего наблюдателя

Рассмотрим случай, когда П является плоскостью. Она характеризуется вектором $\vec{n} = \text{const}$ и некоторой лежащей на ней точкой o. Умножим скалярно обе части равенства (1.108) на $\vec{n}$:

$$\lambda\overrightarrow{qp}\cdot\vec{n} = -n'_{\text{прел}}\vec{s}\cdot\vec{n} - \frac{\text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} - n'_{\text{прел}}\vec{s}\cdot\vec{n}}{\vec{n}^2}\vec{n}^2,$$

$$\lambda\overrightarrow{qp}\cdot\vec{n} = \cancel{-n'_{\text{прел}}\vec{s}\cdot\vec{n}} - \text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} + \cancel{n'_{\text{прел}}\vec{s}\cdot\vec{n}},$$

$$\lambda\overrightarrow{qp}\cdot\vec{n} = -\text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2}.$$

Аналогично тому, как сделано выше, можно заменить произведение $\overrightarrow{qp}\cdot\vec{n}$ на $\overrightarrow{op}\cdot\vec{n},$ после чего получается

$$\lambda = \frac{-\text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2}}{\overrightarrow{op}\cdot\vec{n}}. \qquad (1.109)$$

Как и в случае с отражением, различают случаи **действительного** ($\lambda > 0$), **мнимого** ($\lambda < 0$) и **вырожденного** ($\lambda = 0$) преломления. Видимое преломление подлежит построению только при $\lambda > 0$. Для проверки этого неравенства достаточно проверить, чтобы скалярные произведения $\vec{s}\cdot\vec{n}$ и $\overrightarrow{op}\cdot\vec{n}$ были разных знаков. Получается следующее:

$$\overrightarrow{qp} = -\frac{1}{\lambda}\left(n'_{\text{прел}}\vec{s} + \frac{\text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} - n'_{\text{прел}}\vec{s}\cdot\vec{n}}{|\vec{n}|^2}\vec{n}\right),$$

$$q = p + \frac{1}{\lambda}\left(n'_{\text{прел}}\vec{s} + \frac{\text{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} - n'_{\text{прел}}\vec{s}\cdot\vec{n}}{|\vec{n}|^2}\vec{n}\right). \qquad (1.110)$$

Таким образом, **алгоритм вычисления видимого преломления** точки p через плоскость П при **дальнем** наблюдателе выглядит следующим образом:

1. Проверка условия существования преломления (1.102) для векторов $\vec{s}$ и $\vec{n}$ и относительного показателя преломления $n'_{\text{прел}} = 1/n_{\text{прел}}$. Его нарушение означает, что такой дальний наблюдатель не сможет наблюдать преломление какого бы то ни было объекта через заданную плоскость.

2. Преломление является видимым тогда и только тогда, когда скалярные произведения $\vec{s}\cdot\vec{n}$ и $\overrightarrow{op}\cdot\vec{n}$ разного знака и не равны нулю. В противном случае

имеет место мнимое либо вырожденное преломление, не подлежащее вычислению.

3. В случае существования и видимости преломления оно вычисляется по формулам (1.109) и (1.110).

В случае же **ближнего** наблюдателя, расположенного в точке s, имеют место следующие рассуждения. Основания $p_\perp$ и $s_\perp$ перпендикуляров, опущенных из точек p и s на плоскость П, в силу расположения в одной плоскости векторов $\overrightarrow{pq}$, $\overrightarrow{qs}$ и $\vec{n}(q)=\vec{n}=\text{const}$ должны лежать на одной прямой с искомой точкой q (Рисунок 1.116). Таким образом, вычисление точки q сводится к задаче деления отрезка $p_\perp s_\perp$ в таком отношении, чтобы образовавшиеся углы падения $\alpha=\left|\angle(\vec{n},\overrightarrow{pq})\right|$ и преломления $\gamma=\left|\angle(-\vec{n},\overrightarrow{qs})\right|$ удовлетворяли закону Снеллиуса-Декарта (1.100). Обозначая теперь $\left|\overrightarrow{p_\perp p}\right|=P$, $\left|\overrightarrow{s_\perp s}\right|=S$, $\left|\overrightarrow{p_\perp s_\perp}\right|=D$, имеем

$$\left|\overrightarrow{p_\perp q}\right|=P\,\mathrm{tg}\,\alpha,\ \left|\overrightarrow{qs_\perp}\right|=S\,\mathrm{tg}\,\gamma.$$

Выразим tg γ через tg α:

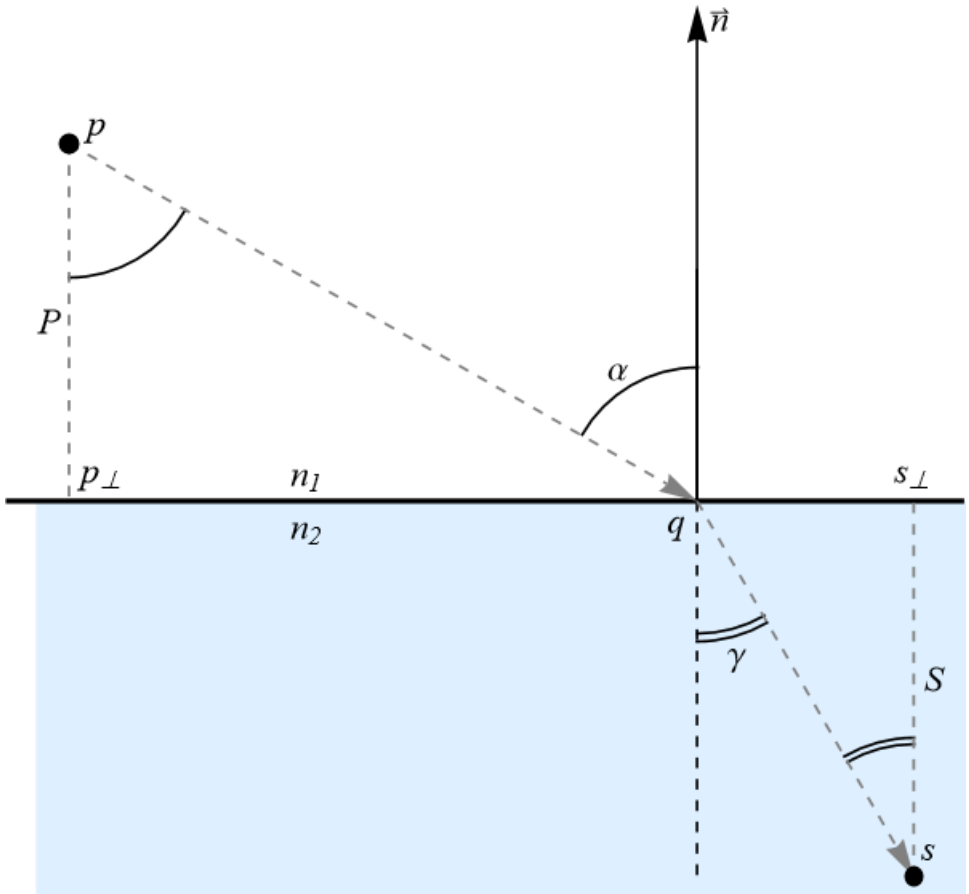


Рисунок 1.116 – Построение преломлённого изображения точки для ближнего наблюдателя

$$\operatorname{tg}\gamma=\frac{\sin\gamma}{\cos\gamma}=\frac{n_1}{n_2}\frac{\sin\alpha}{\sqrt{1-\sin^2\gamma}}=\frac{n_1}{n_2}\frac{\sin\alpha}{\sqrt{1-\dfrac{n_1^2}{n_2^2}\sin^2\alpha}}=\frac{n_1\sin\alpha}{\sqrt{n_2^2-n_1^2\sin^2\alpha}}=$$

$$=\frac{n_1\dfrac{\operatorname{tg}\alpha}{\sqrt{1+\operatorname{tg}^2\alpha}}}{\sqrt{n_2^2-n_1^2\dfrac{\operatorname{tg}^2\alpha}{1+\operatorname{tg}^2\alpha}}}=\frac{n_1\operatorname{tg}\alpha}{\sqrt{n_2^2+n_2^2\operatorname{tg}^2\alpha-n_1^2\operatorname{tg}^2\alpha}}=\frac{n_1\operatorname{tg}\alpha}{\sqrt{n_2^2+(n_2^2-n_1^2)\operatorname{tg}^2\alpha}}.$$

Тогда с учётом $\left|\overrightarrow{p_\perp q}\right|+\left|\overrightarrow{qs_\perp}\right|=\left|\overrightarrow{p_\perp s_\perp}\right|$ можно вывести уравнение относительно $t=\operatorname{tg}\alpha,\ t\in(0,+\infty)$:

$$P\operatorname{tg}\alpha+S\operatorname{tg}\gamma=D,$$

$$Pt+S\frac{n_1t}{\sqrt{n_2^2+(n_2^2-n_1^2)t^2}}=D. \qquad (1.111)$$

Так как на промежутке $[0,\pi/2)$ тангенс является возрастающей функцией, то при увеличении α будет увеличиваться tg α, а также и tg γ, ведь сам угол γ также увеличивается (что следует из закона Снеллиуса-Декарта). Отсюда следует, что функция $f(t)=Pt+S\dfrac{n_1t}{\sqrt{n_2^2+(n_2^2-n_1^2)t^2}}-D$ является возрастающей, и уравнение (1.111) имеет не более одного решения. Более того, проверяя значения функции f на концах области определения, имеем, что это уравнение имеет ровно одно решение:

$$f(0)=P\cdot0+S\frac{n_1\cdot0}{\sqrt{n_2^2+(n_2^2-n_1^2)\cdot0^2}}-D=-D<0,$$

$$n_1<n_2\Rightarrow\lim_{t\to+\infty}f(t)=\left[\lim_{t\to+\infty}Pt=+\infty,\lim_{t\to+\infty}S\frac{n_1t}{\sqrt{n_2^2+(n_2^2-n_1^2)t^2}}=\right.$$

$$\left.\lim_{t\to+\infty}\frac{Sn_1}{\sqrt{\dfrac{n_2^2}{t^2}+n_2^2-n_1^2}}=\frac{Sn_1}{\sqrt{n_2^2-n_1^2}}\right]=+\infty,n_1>n_2\Rightarrow\lim_{t\to\frac{n_2}{\sqrt{n_1^2-n_2^2}}-0}f(t)=$$

$$=\left[\lim_{t\to\frac{n_2}{\sqrt{n_1^2-n_2^2}}-0}Pt=\frac{Pn_2}{\sqrt{n_1^2-n_2^2}},\lim_{t\to\frac{n_2}{\sqrt{n_1^2-n_2^2}}-0}S\frac{n_1t}{\sqrt{n_2^2+(n_2^2-n_1^2)t^2}}=+\infty\right]=+\infty.$$

Уравнение (1.111) сводится к уравнению четвёртой степени:

$$Pt + S\frac{n_1 t}{\sqrt{n_2^2 + (n_2^2 - n_1^2)t^2}} = D \Leftrightarrow \sqrt{n_2^2 + (n_2^2 - n_1^2)t^2} =$$

$$= \frac{Sn_1 t}{D - Pt} \overset{\text{неэквивалентный переход!}}{\Rightarrow} n_2^2 + (n_2^2 - n_1^2)t^2 = \frac{S^2 n_1^2 t^2}{(D - Pt)^2},$$

$$(D - Pt)^2 (n_2^2 + (n_2^2 - n_1^2)t^2) - S^2 n_1^2 t^2 = 0. \qquad (1.112)$$

В силу того, что на одном из шагов обе части были возведены в квадрат, возможно появление посторонних корней. Поэтому нужно наложить ещё одно требование: $\frac{Sn_1 t}{D - Pt} > 0 \Leftrightarrow D - Pt > 0 \Leftrightarrow t < \frac{D}{P}$.

Таким образом, **алгоритм вычисления видимого преломления** точки p через плоскость П при **ближнем** наблюдателе может быть сформулирован следующим образом:

1. Проверка наличия эффекта: преломлённое изображение точки p на плоскости П, наблюдаемое из точки s, имеет место тогда и только тогда, когда p и s находятся по разные стороны от П, что эквивалентно разным знакам чисел $\overrightarrow{op} \cdot \vec{n}$ и $\overrightarrow{os} \cdot \vec{n}$, где o – некоторая точка плоскости П. При нарушении этого условия построения точки q не происходит.

2. Находятся ортогональные проекции $p_\perp$ и $s_\perp$:

$$p_\perp = p - \frac{\overrightarrow{op} \cdot \vec{n}}{\vec{n}^2}\vec{n},\ s_\perp = s - \frac{\overrightarrow{os} \cdot \vec{n}}{\vec{n}^2}\vec{n}. \qquad (1.113)$$

3. Вычисляются расстояния P, S, D:

$$P = \frac{\left|\overrightarrow{op} \cdot \vec{n}\right|}{\left|\vec{n}\right|},\ S = \frac{\left|\overrightarrow{os} \cdot \vec{n}\right|}{\left|\vec{n}\right|},\ D = \left|\overrightarrow{ps} - \frac{\overrightarrow{ps} \cdot \vec{n}}{\vec{n}^2}\vec{n}\right|. \qquad (1.114)$$

Если получилось $D = 0$, то тогда сразу же получается искомая точка $q = p_\perp = s_\perp$.

4. Эти значения, а также абсолютные показатели преломления n_1 (для среды, где находится точка p) и n_2 (среды, где расположена точка s), поставляются в уравнение (1.112) которое имеет единственное решение $0 < t < D/P$.

5. Откладывается искомая точка q:

$$q = p_\perp + \frac{P}{D}t\overrightarrow{p_\perp s_\perp}. \qquad (1.115)$$

2. ПРАКТИЧЕСКИЙ РАЗДЕЛ

2.1. Алгоритмы отсечения и удаления невидимых рёбер и граней

2.1.1. Алгоритм Коэна-Сазерленда

Алгоритм Коэна-Сазерленда (стр. 15) определяет результат отсечения заданного отрезка регулярным окном, которое обычно задаётся системой неравенств

$$y \le y_{\max},\ \ y \ge y_{\min},\ \ x \le x_{\max},\ \ x \ge x_{\min}. \qquad (2.1)$$

Пример 1.1. Осуществить внешнее отсечение отрезка *AB* с концами в точках с координатами *A*(12, –7) и *B*(–6, 10) регулярным окном (2.1), где $y_{\max} = 5,\ y_{\min} = 0,\ x_{\max} = 5,\ x_{\min} = 0.$

Вычисляем четырёхбитные коды для обоих концов отрезка. Для этого их координаты подставим в систему (2.1). Если для какой-то из точек нарушается *i*-ое неравенство, $i = \overline{1,4}$, то *i*-ый бит кода этой точки приравнивается единице, в противном случае – нулю:

$$A(12,-7):$$
$$(y > 5) = \textit{ложь},$$
$$(y < 0) = \textit{истина},$$
$$(x > 5) = \textit{истина},$$
$$(x < 0) = \textit{ложь},$$
$$a = 0110,$$

$$B(-6,10):$$
$$(y > 5) = \textit{истина},$$
$$(y < 0) = \textit{ложь},$$
$$(x > 5) = \textit{ложь},$$
$$(x < 0) = \textit{истина},$$
$$b = 1001.$$

Проверка тривиальных случаев даёт следующие результаты: $a \,\&\, b = 0000$, что означает, что отрезок не находится целиком выше/ниже/правее/левее окна, $a \vee b = 1111 \ne 0000$, что означает, что отрезок не лежит целиком внутри окна. Значит, алгоритм Коэна-Сазерленда продолжает свою работу.

Далее необходимо определить точки пересечения отрезка *AB* с прямыми

$$y = y_{\max},\ \ y = y_{\min},\ \ x = x_{\max},\ \ x = x_{\min}, \qquad (2.2)$$

на которых лежат границы окна. Из этих прямых исходный отрезок пересекается только с теми, для которых соответствующий бит в коде $a \vee b$ равен 1. В данном примере $a \vee b = 1111$, значит, следует вычислять пересечения отрезка со всеми четырьмя прямыми. Это можно сделать, например, при помощи общего уравнения прямой *AB*:

$$\overrightarrow{AB}(-18,17) \perp (17,18),$$
$$AB: 17(x+6)+18(y-10)=0,$$
$$17x+18y-180+102=0,$$
$$17x+18y=78.$$

Это уравнение дополняется каким-то одним из уравнений (2.2), и получается система для нахождения координат соответствующего пересечения:

$$C_1: \begin{cases} 17x+18y=78, \\ y=5 \end{cases} \Rightarrow y=5, x=\frac{78-18y}{17}=-\frac{12}{17},$$
$$C_2: \begin{cases} 17x+18y=78, \\ y=0 \end{cases} \Rightarrow y=0, x=\frac{78-18y}{17}=\frac{78}{17},$$
$$C_3: \begin{cases} 17x+18y=78, \\ x=5 \end{cases} \Rightarrow x=5, y=\frac{78-17x}{18}=-\frac{7}{18},$$
$$C_4: \begin{cases} 17x+18y=78, \\ x=0 \end{cases} \Rightarrow x=0, y=\frac{78-17x}{18}=\frac{13}{3}.$$

Итак, получили четыре точки, не более двух из которых, возможно, являются концами искомого отрезка. Концом искомого отрезка, вообще говоря, может оказаться также точка *A* либо *B*. Определить, которые из всех точек являются концами усечённого отрезка, можно по их четырёхбитным кодам:

$C_1\left(-\frac{12}{17},5\right)$:	$C_2\left(\frac{78}{17},0\right)$:	$C_3\left(5,-\frac{7}{18}\right)$:	$C_4\left(0,\frac{13}{3}\right)$:
$(y>5)=$ *ложь*,	$(y>5)=$ *ложь*,	$(y>5)=$ *ложь*,	$(y>5)=$ *ложь*,
$(y<0)=$ *ложь*,	$(y<0)=$ *ложь*,	$(y<0)=$ *истина*,	$(y<0)=$ *ложь*,
$(x>5)=$ *ложь*,	$(x>5)=$ *ложь*,	$(x>5)=$ *ложь*,	$(x>5)=$ *ложь*,
$(x<0)=$ *истина*,	$(x<0)=$ *ложь*,	$(x<0)=$ *ложь*,	$(x<0)=$ *ложь*,
$c_1=0001$,	$c_2=0000$,	$c_3=0100$,	$c_4=0000$.

Из всех шести точек только у точек C_2 и C_4 коды равны 0000, значит, это и будут концы искомого отрезка (Рисунок 2.1).

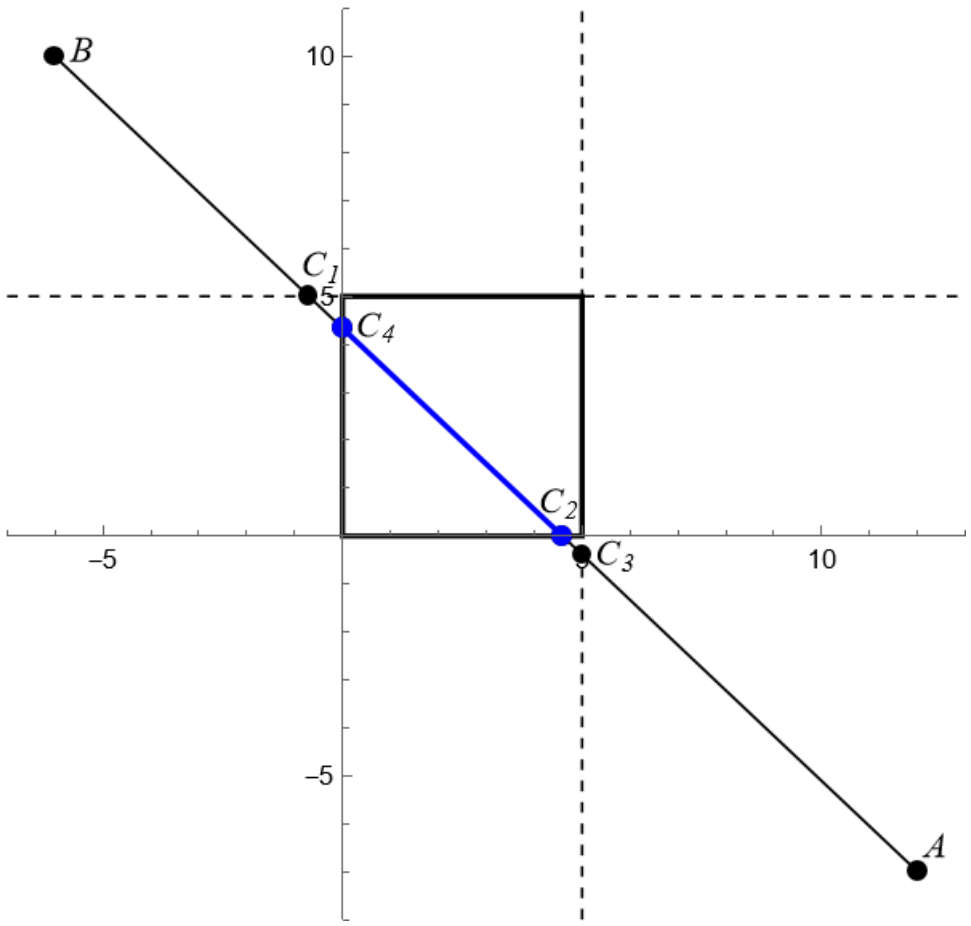


Рисунок 2.1 – Отсечение отрезка регулярным внешним отсекателем

Пример 1.2. Осуществить внешнее отсечение отрезка AB с концами в точках с координатами $A(10, 2)$ и $B(4, 3)$ регулярным окном (2.1), где $y_{\max} = 5$, $y_{\min} = 0$, $x_{\max} = 8$, $x_{\min} = 0$.

В данном примере получаем следующие коды: $a = 0010$, $b = 0000$. Так как $a \,\&\, b = 0000$, $a \vee b = 0010 \neq 0000$, то снова получаем, что имеет место нетривиальный случай. Значит, нужно искать пересечение отрезка AB с прямыми (2.2). Здесь уже необходимо искать только одно пересечение с третьей прямой (т.к. только третий бит у кода $a \vee b$ равен 1):

$$\overrightarrow{AB}(-6,1) \perp (1,6),$$
$$AB: x - 4 + 6(y - 3) = 0,$$
$$x + 6y = 22,$$
$$C: \begin{cases} x + 6y = 22, \\ x = 8 \end{cases} \Rightarrow C\left(8, \frac{7}{3}\right).$$

Нетрудно проверить, что код для этой точки равен 0000. Поэтому из трёх точек A, B, C конечными точками усечённого отрезка являются B и C (Рисунок 2.2).

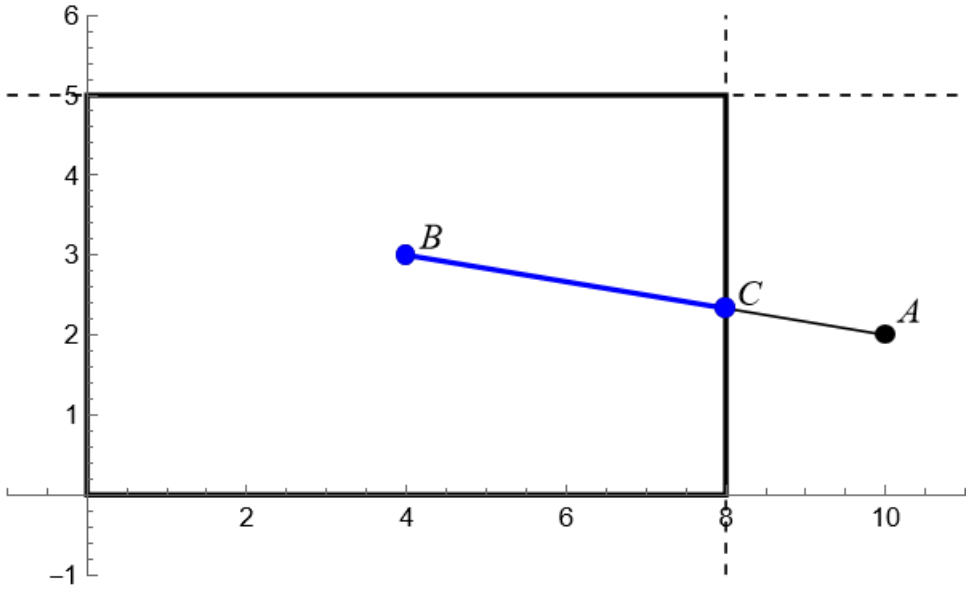


Рисунок 2.2 – Отсечение отрезка с видимым концом

Пример 1.3. Осуществить внешнее отсечение отрезка AB с концами в точках с координатами $A(-1, 1)$ и $B(1, 4)$ регулярным окном (2.1), где $y_{\max} = 2$, $y_{\min} = 0$, $x_{\max} = 5$, $x_{\min} = 0$.

Для концов отрезка AB получаются коды, равные $a = 0001$, $b = 1000$. Так как $a \,\&\, b = 0000$, $a \vee b = 1001$, то снова получаем, что это нетривиальный случай. Судя по значению кода $a \vee b$, здесь необходимо вычислить пересечение отрезка AB с двумя прямыми: верхней и левой:

$$\overrightarrow{AB}(2,3) \perp (3,-2),$$

$$AB: 3(x+1) - 2(y-1) = 0,$$

$$3x - 2y = -5,$$

$$C_1: \begin{cases} 3x - 2y = -5, \\ y = 2 \end{cases} \Rightarrow C_1\left(-\frac{1}{3}, 2\right),$$

$$C_2: \begin{cases} 3x - 2y = -5, \\ x = 0 \end{cases} \Rightarrow C_2\left(0, \frac{5}{2}\right).$$

Для полученных точек коды равны $c_1 = 0001$, $c_2 = 1000$. Таким образом, ни одна из точек A, B, C_1 и C_2 не лежит на границе окна (так как у них у всех коды отличны от 0000). Значит, отрезок полностью отсекается (Рисунок 2.3).

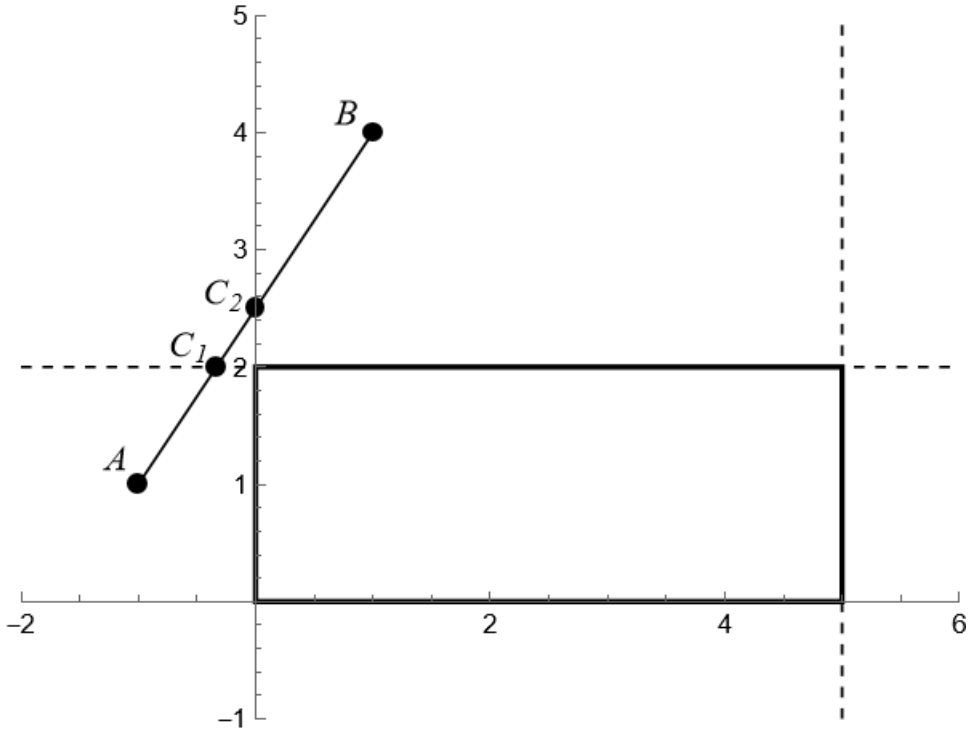


Рисунок 2.3 – Пример полного отсечения отрезка

2.1.2. Алгоритм Лианга-Барски

Алгоритм Лианга-Барски (стр. 22) позволяет провести отсечение отрезка регулярным отсекателем при помощи его параметрического представления (1.4). Решим примеры 1.1–1.3 с использованием этого алгоритма.

Пример 1.4. Осуществить внешнее отсечение отрезка *AB* с концами в точках с координатами *A*(12, –7) и *B*(–6, 10) регулярным окном (2.1), где $y_{\max} = 5,\ y_{\min} = 0,\ x_{\max} = 5,\ x_{\min} = 0.$

Для первоначального, неусечённого отрезка *AB* имеем значения параметров стартовой и конечной точки, равные соответственно $t_{\text{нач}} = 0$ и $t_{\text{кон}} = 1$. Рассмотрим последовательно результаты отсечения этого отрезка верхней, нижней, правой и левой границами (2.2). Для каждой из них вычисляются свои значения p_i и q_i согласно формулам (1.6), которые затем подставляются в неравенство (1.5), что может привести к усечению числового отрезка $[t_{\text{нач}}, t_{\text{кон}}]$.

Так, для верхней границы имеют место следующие вычисления:

$$p_4 = y_B - y_A = 17,\ \ q_4 = y_{\max} - y_A = 12.$$

Отсюда имеем неравенство $17t \le 12 \Rightarrow t \le 12/17$. Значит, происходит отсечение отрезка $[t_{\text{нач}}, t_{\text{кон}}]$ с конца, так как из двух начальных неравенств $t \ge 0$

и $t \le 1$ последнее заменяется более сильным $t \le 12/17$. Таким образом, в этой итерации необходимо переопределить один из параметров: $t_{кон} := 12/17$. Легко видеть, что это значение по-прежнему больше $t_{нач}$, значит, полного отсечения пока не происходит. На рисунке 2.4 красной точкой указана начальная точка A исходного отрезка, которой отвечает значение параметра $t = t_{нач} = 0$, синими точками указана конечная точка B исходного отрезка со значением параметра $t = 1$, а также точка C_4, которой отвечает только что вычисленное значение $t = t_{кон} = 12/17$.

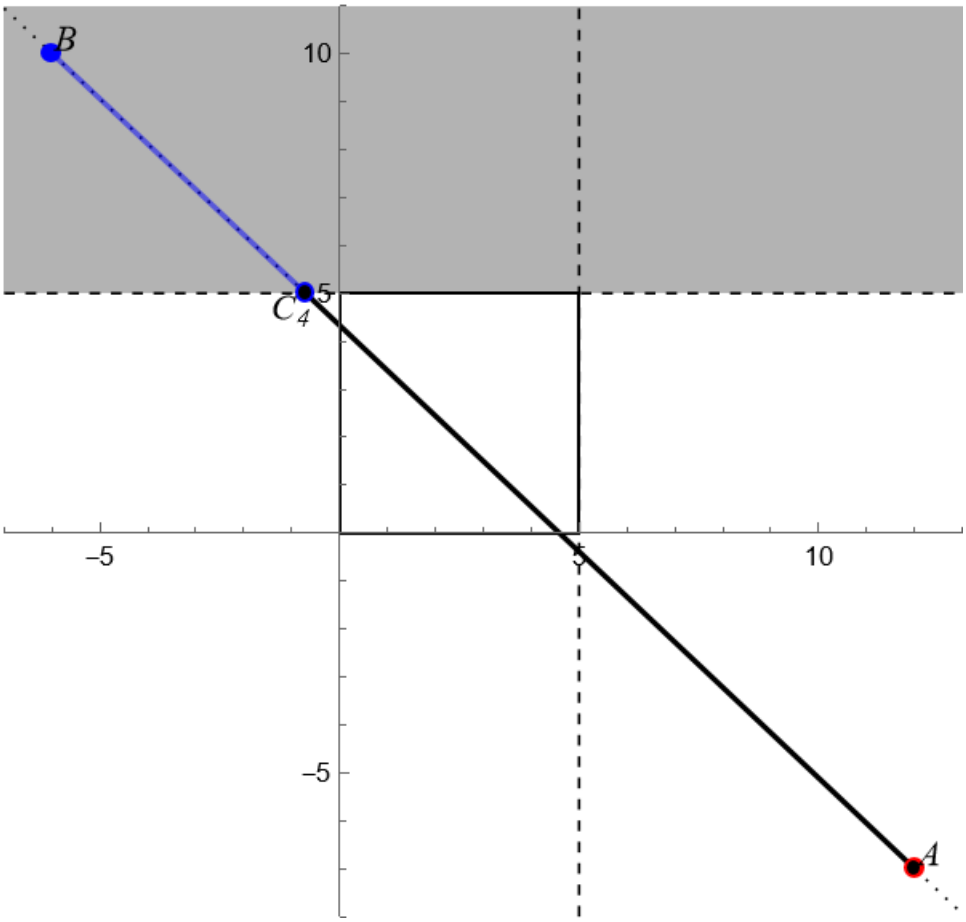


Рисунок 2.4 – Отсечение отрезка верхней границей в ходе алгоритма Лианга-Барски

Для нижней границы имеем

$$p_3 = y_A - y_B = -17, q_3 = y_A - y_{\min} = -7,$$
$$-17t \le -7 \Rightarrow t \ge 7/17.$$

Значит, имеет место отсечение с начала: $t_{нач} := 7/17$ (красная точка C_3 на рисунке 2.5) После этого переопределения снова выполняется $t_{нач} < t_{кон}$, значит, опять не происходит полного отсечения.

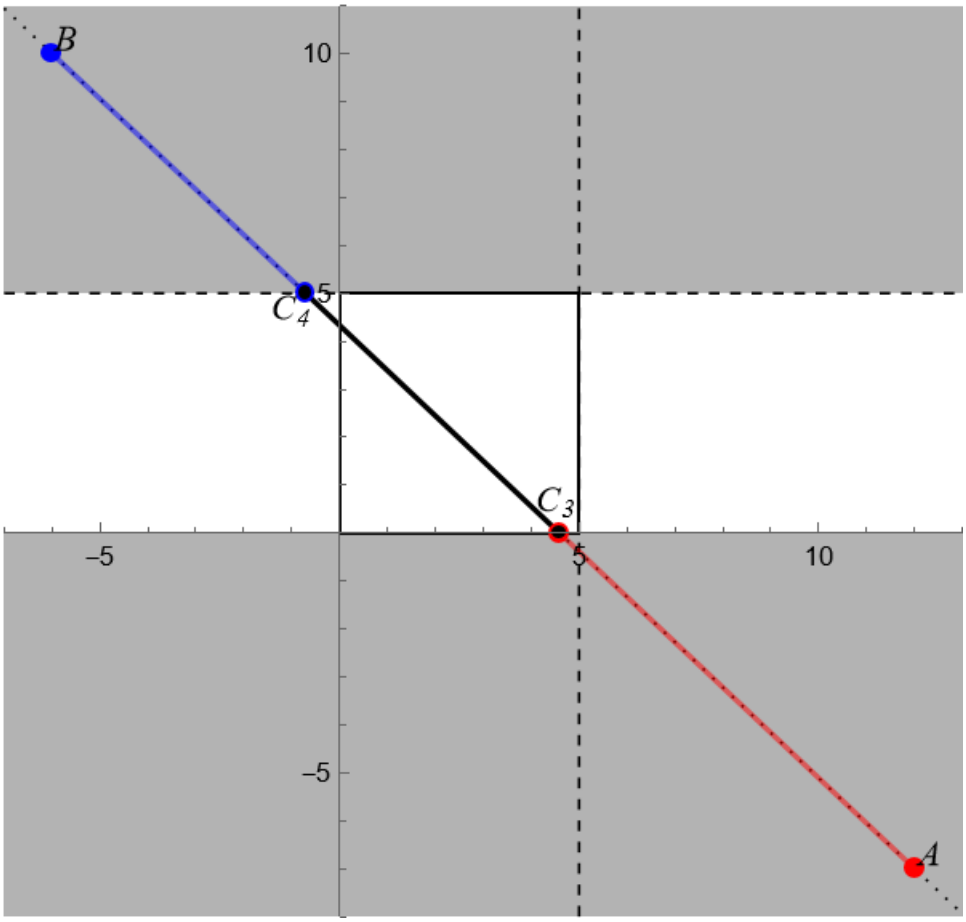


Рисунок 2.5 – Отсечение нижней границей

Правая граница:

$$p_2 = x_B - x_A = -18, q_2 = x_{\max} - x_A = -7,$$
$$-18t \le -7 \Rightarrow t \ge 7/18.$$

Это значение не превышает $t_{\text{нач}} = 7/17$ – отсечения на данной итерации не происходит. На рисунке 2.6 значение параметра $t = 7/18$ отвечает красной точке C_2, которая оказывается на ранее усечённой части отрезка AB между точкой A с параметром $t = 0$ и точкой C_3, которой соответствует значение $t_{\text{нач}} = 7/17$.

Левая граница:

$$p_1 = x_A - x_B = 18, q_2 = x_A - x_{\min} = 12,$$
$$18t \le 12 \Rightarrow t \le 2/3.$$

Полученное значение меньше $t_{\text{кон}} = 12/17$, поэтому имеет место отсечение: $t_{\text{кон}} := 2/3$.

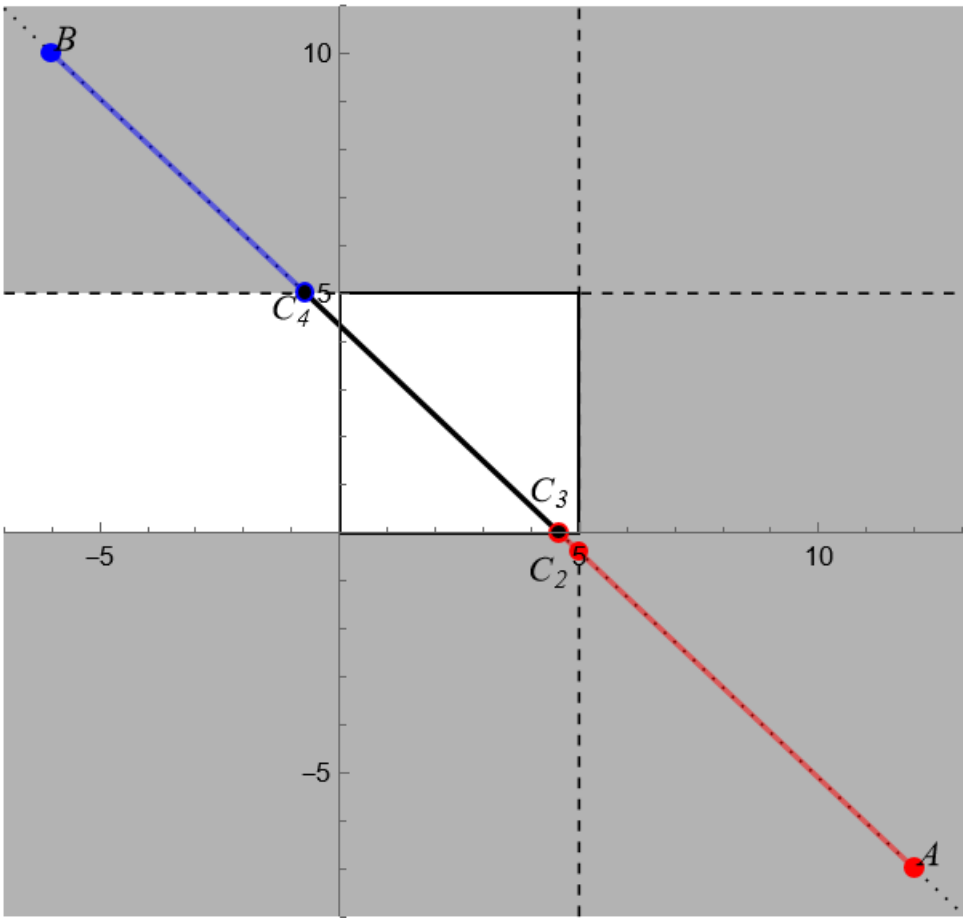


Рисунок 2.6 – Отсечения правой границей не произошло

Окончательно получаем отрезок $[t_{\text{нач}}, t_{\text{кон}}] = [7/17, 2/3]$. Подставим эти значения в параметрические уравнения исходного отрезка AB, и получим координаты концов искомого, усечённого отрезка:

$$\begin{cases} x = x_A + t(x_B - x_A) = 12 - 18t, \\ y = y_A + t(y_B - y_A) = -7 + 17t, \end{cases}$$

$$t_{\text{нач}} : C_3(12 - 18t, -7 + 17t)\big|_{t=7/17} = C_3\left(\frac{12 \cdot 17 - 18 \cdot 7}{17}, -7 + \frac{17 \cdot 7}{17}\right) = C_3\left(\frac{78}{17}, 0\right),$$

$$t_{\text{кон}} : C_1(12 - 18t, -7 + 17t)\big|_{t=2/3} = C_1\left(12 - \frac{18 \cdot 2}{3}, \frac{-7 \cdot 3 + 17 \cdot 2}{3}\right) = C_1\left(0, \frac{13}{3}\right).$$

Результат отсечения отрезка AB по алгоритму Лианга-Барски представлен на рисунке 2.7: красным выделена часть отрезка AB, усечённая со стороны точки начала отрезка A, синим – со стороны точки конца B.

Пример 1.5. Осуществить внешнее отсечение отрезка AB с концами в точках с координатами $A(10, 2)$ и $B(4, 3)$ регулярным окном (2.1), где $y_{\max} = 5$, $y_{\min} = 0$, $x_{\max} = 8$, $x_{\min} = 0$.

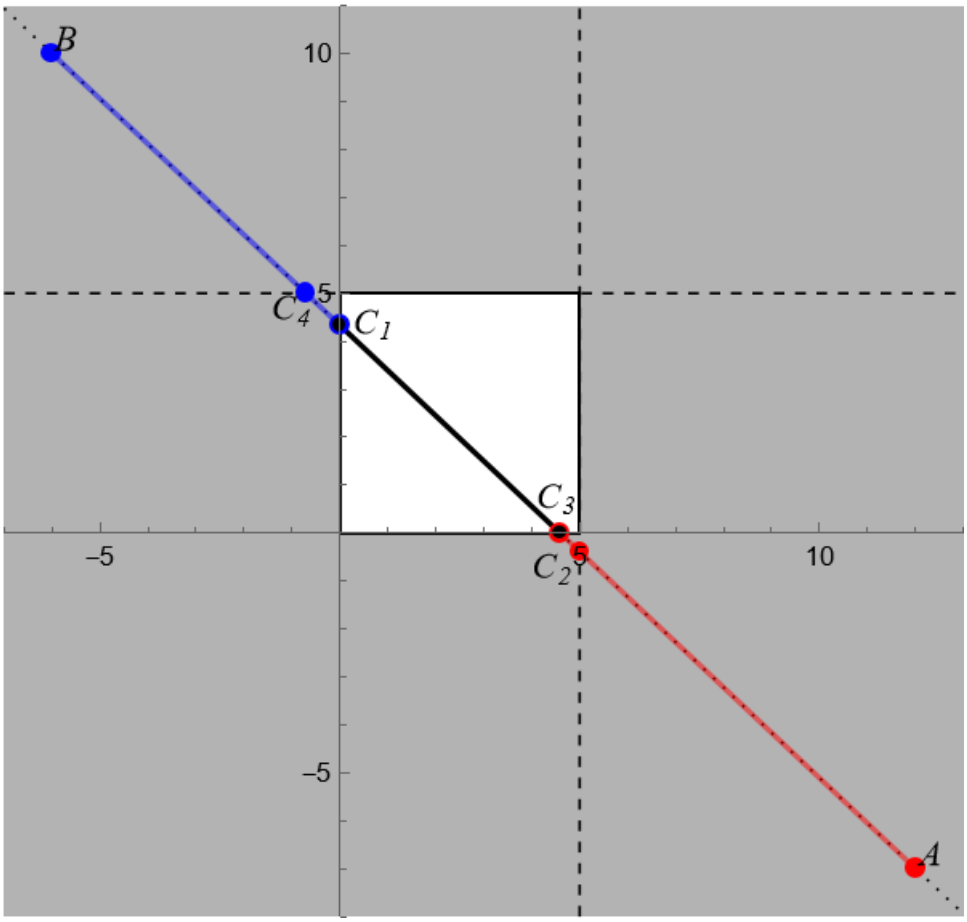


Рисунок 2.7 – Отсечение отрезка левой границей и результат работы алгоритма Лианга-Барски

Применим алгоритм Лианга-Барски аналогично примеру выше. Вначале положим $t_{\text{нач}} = 0,\ t_{\text{кон}} = 1$, затем проведём отсечение числового отрезка $[t_{\text{нач}}, t_{\text{кон}}]$ четырьмя прямыми:

$$p_4 = y_B - y_A = 1, q_4 = y_{\max} - y_A = 6,$$
$$t \le 6$$

– отсечения не происходит, так как уже имеет место неравенство $t \le t_{\text{кон}} = 1$;

$$p_3 = y_A - y_B = -1, q_3 = y_A - y_{\min} = 2,$$
$$-t \le 2 \Rightarrow t \ge -2$$

– отсечения опять не происходит, т.к. выполняется $t \ge t_{\text{нач}} = 0$;

$$p_2 = x_B - x_A = -6, q_2 = x_{\max} - x_A = -2,$$
$$-6t \le -2 \Rightarrow t \ge 1/3 > t_{\text{нач}},$$

значит, происходит отсечение с начала: $t_{\text{нач}} = 1/3$. Числовой отрезок $[t_{\text{нач}}, t_{\text{кон}}] = [1/3, 1]$ определён корректно, значит, полного отсечения пока не происходит;

$$p_1 = x_A - x_B = 6, q_1 = x_A - x_{\min} = 10,$$
$$6t \le 10 \Rightarrow t \le 5/3,$$

значит, отсечения не происходит, ведь $t_{\text{кон}} = 1 \le 5/3$. Итак, имеем числовой отрезок $[t_{\text{нач}}, t_{\text{кон}}] = [1/3, 1]$, начало которого отвечает следующей точке:

$$\begin{cases} x = 10 - 6t, \\ y = 2 + t, \end{cases}$$
$$t_{\text{нач}} : C(10 - 6t, 2 + t)\big|_{t=1/3} = C\left(10 - \frac{6}{3}, 2 + \frac{1}{3}\right) = C\left(8, \frac{7}{3}\right),$$

а параметру $t_{\text{кон}} = 1$ соответствует исходная концевая точка B. Работа алгоритма Лианга-Барски для этого примера проиллюстрирована на рисунке 2.8.

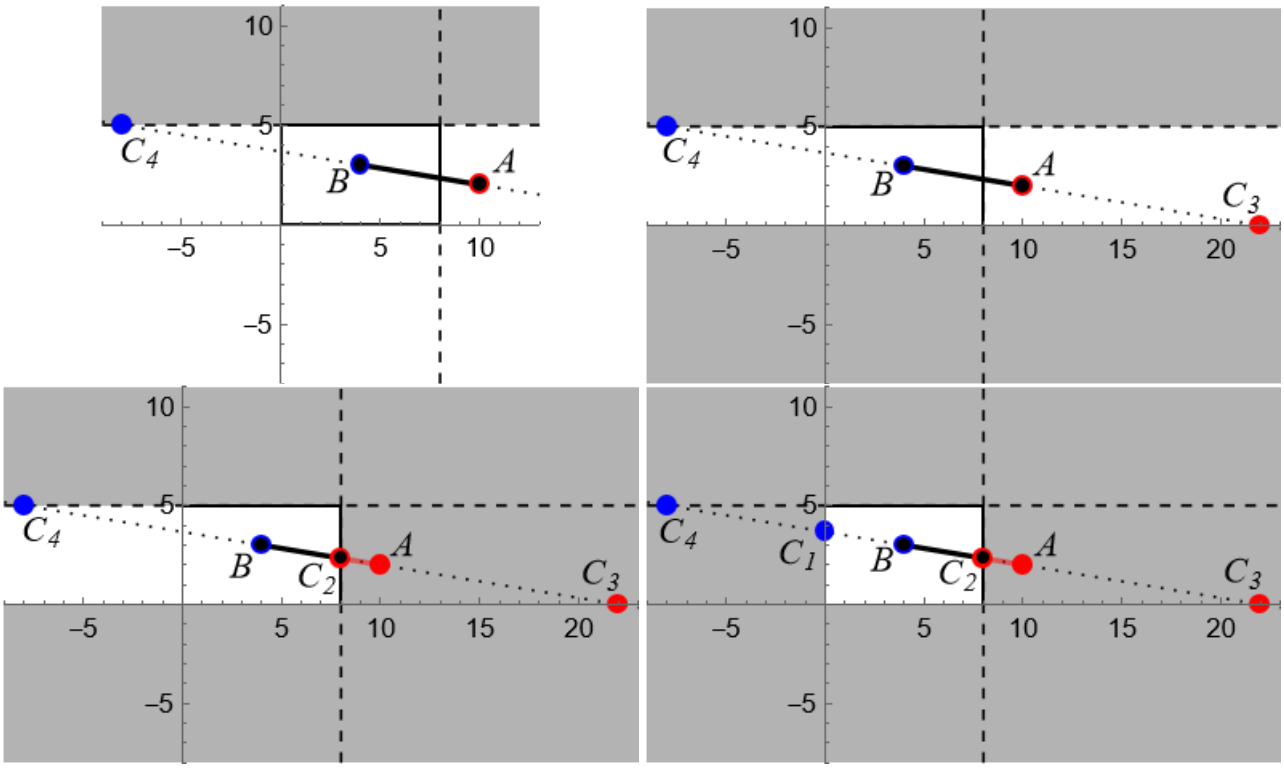


Рисунок 2.8 – Последовательное отсечение отрезка границами регулярного окна

Пример 1.6. Осуществить внешнее отсечение отрезка AB с концами в точках с координатами $A(-1, 1)$ и $B(1, 4)$ регулярным окном (2.1), где $y_{\max} = 2$, $y_{\min} = 0$, $x_{\max} = 5$, $x_{\min} = 0$.

Применение алгоритма Лианга-Барски даст следующие вычисления:

$$p_4 = y_B - y_A = 3, q_4 = y_{\max} - y_A = 1,$$
$$3t \le 1 \Rightarrow t \le 1/3 < t_{\text{кон}}$$

– имеет место отсечение отрезка с конца: $t_{\text{кон}} := 1/3$. Пока остаётся числовой отрезок $[t_{\text{нач}}, t_{\text{кон}}] = [0, 1/3]$.

$$p_3 = y_A - y_B = -3, q_3 = y_A - y_{\min} = 1,$$
$$-3t \le 1 \Rightarrow t \ge -1/3$$

– отсечения не происходит, так как $t_{\text{нач}} > -1/3$.

$$p_2 = x_B - x_A = 2, q_2 = x_{\max} - x_A = 6,$$
$$2t \le 6 \Rightarrow t \le 3.$$

Здесь отсечения тоже не происходит, т.к. $t_{\text{кон}} = 1/3 > 3$.

Наконец, для левой границы имеем следующее:

$$p_1 = x_A - x_B = -2, q_1 = x_A - x_{\min} = -1,$$
$$-2t \le -1 \Rightarrow t \ge 1/2 > t_{\text{нач}},$$

значит, происходит отсечение отрезка с начала: $t_{\text{нач}} := 1/2$. Однако это значение превышает $t_{\text{кон}} = 1/3$, откуда следует, что имеет место полное отсечение отрезка. Результаты работы алгоритма Лианга-Барски после каждой из итераций приведены на рисунке 2.9.

2.1.3. Алгоритм Кируса-Бека

Алгоритм Кируса-Бека (стр. 27) позволяет найти результат отсечения отрезка нерегулярным отсекателем, имеющим форму выпуклого полигона. Он опирается на свойство выпуклого полигона, согласно которому он лежит целиком по одну сторону от прямой, проведённой через любое из его рёбер (см. рисунок 1.10). Таким образом, можно совершать поочерёдное отсечение отрезка каждым из рёбер отсекателя.

Пример 1.7. Провести внешнее отсечение отрезка AB выпуклым полигоном $C_1C_2C_3C_4C_5C_1$:

$$A(0,-3), B(-2,5),$$
$$C_1(-3,-4), C_2(0,-1), C_3(3,4), C_4(-1,5), C_5(-6,1).$$

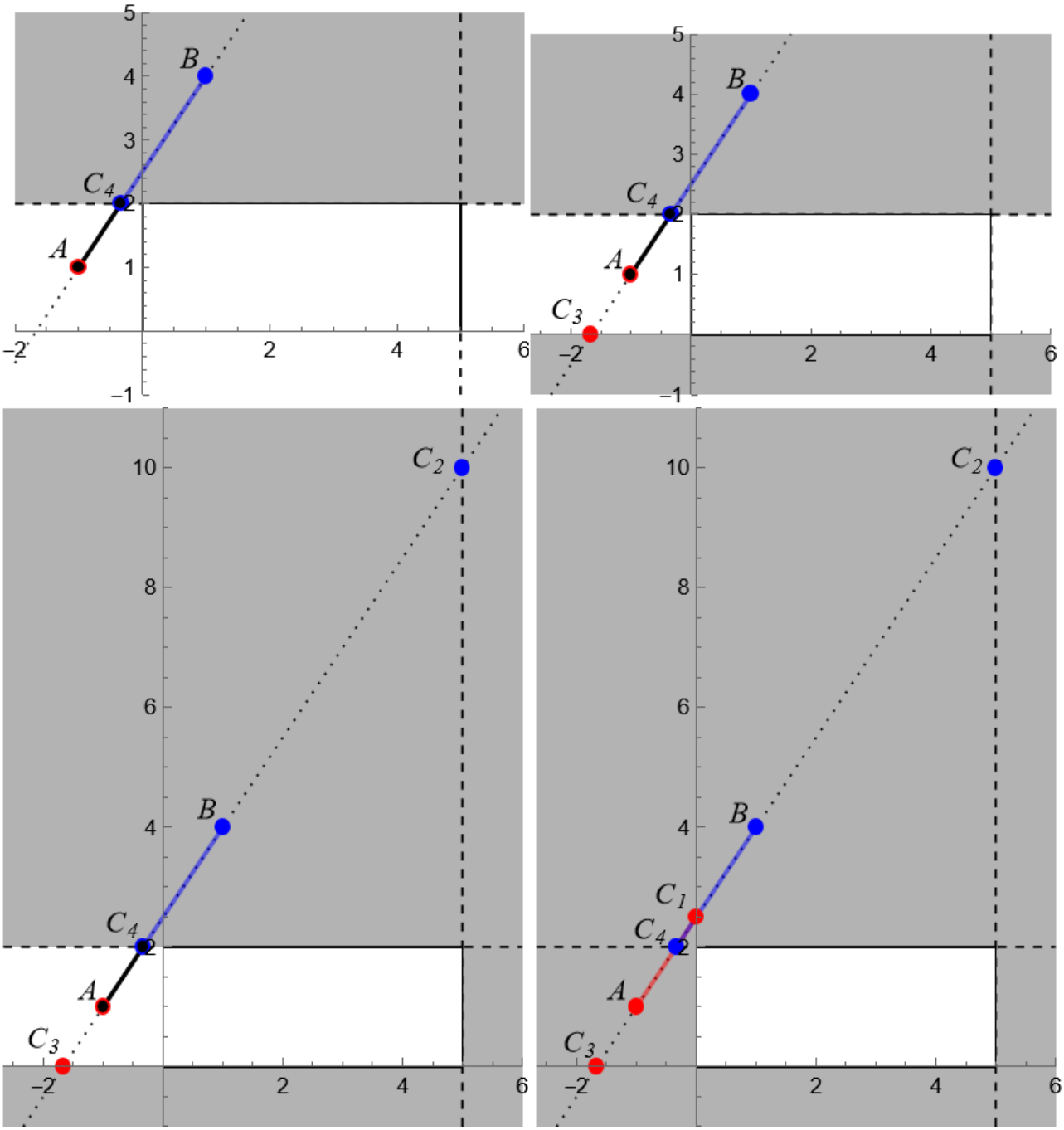


Рисунок 2.9 – Полное отсечение отрезка в ходе алгоритма Лианга-Барски

Вначале необходимо выяснить направление обхода полигона $C_1C_2C_3C_4C_5C_1$ (лево- или правосторонний). Для этого достаточно вычислить знак угла между направляющими векторами каких-либо из двух рёбер:

$$\operatorname{sgn}\angle(\overrightarrow{C_1C_2},\overrightarrow{C_2C_3}) = \operatorname{sgn}\begin{vmatrix} 3 & 3 \\ 3 & 5 \end{vmatrix} = +1.$$

Значит, последовательность вершин $C_1C_2C_3C_4C_5C_1$ обеспечивает левосторонний обход полигона.

Как и в алгоритме Лианга-Барски, полагаем, что у исходного отрезка AB точке начала A отвечает значение параметра $t_{\text{нач}} = 0$, а точке конца B – $t_{\text{кон}} = 1$. В ходе первой итерации рассмотрим отсечение этого отрезка границей отсекателя C_1C_2. По формулам (1.8) получаются следующие вычисления:

$$\overrightarrow{AB}(-2,8), \overrightarrow{C_1C_2}(3,3), \overrightarrow{AC_1}(-3,-1),$$
$$p_1 = \begin{vmatrix} -2 & 8 \\ 3 & 3 \end{vmatrix} = -30, q_1 = \begin{vmatrix} -3 & -1 \\ 3 & 3 \end{vmatrix} = -6.$$

Подставим эти значения в неравенство (1.7):

$$-30t \le -6 \Rightarrow t \ge 1/5.$$

Значит, нужно переопределить значение параметра начала усечённого отрезка: $t_{\text{нач}} := 1/5$, и получится отсечение отрезка AB ребром C_1C_2, которое даст отрезок P_1B (Рисунок 2.10). При этом неравенство $t_{\text{нач}} < t_{\text{кон}}$ сохраняется, т.е. полного отсечения не происходит.

На следующей итерации рассматривается ребро C_2C_3:

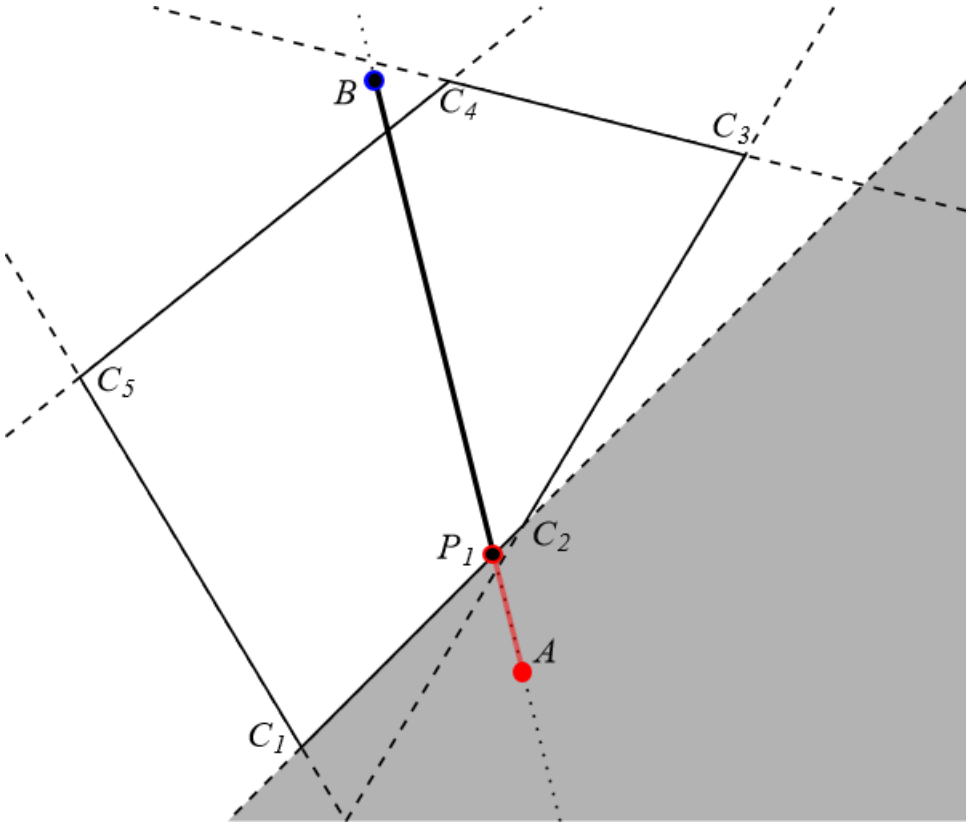


Рисунок 2.10 – Отсечение отрезка одним ребром выпуклого внешнего отсекателя

$$\overrightarrow{C_2C_3}(3,5), \overrightarrow{AC_2}(0,2),$$
$$p_2 = \begin{vmatrix} -2 & 8 \\ 3 & 5 \end{vmatrix} = -34, q_2 = \begin{vmatrix} 0 & 2 \\ 3 & 5 \end{vmatrix} = -6,$$
$$-34t \le -6 \Rightarrow t \ge 3/17.$$

Сравнивая полученное значение с текущим значением $t_{\text{нач}}$, приходим к выводу, что на данной итерации отсечения не происходит (на рисунке 2.11 точка P_2, которой отвечает только что вычисленное значение параметра t, расположена за пределами текущего отрезка P_1B).

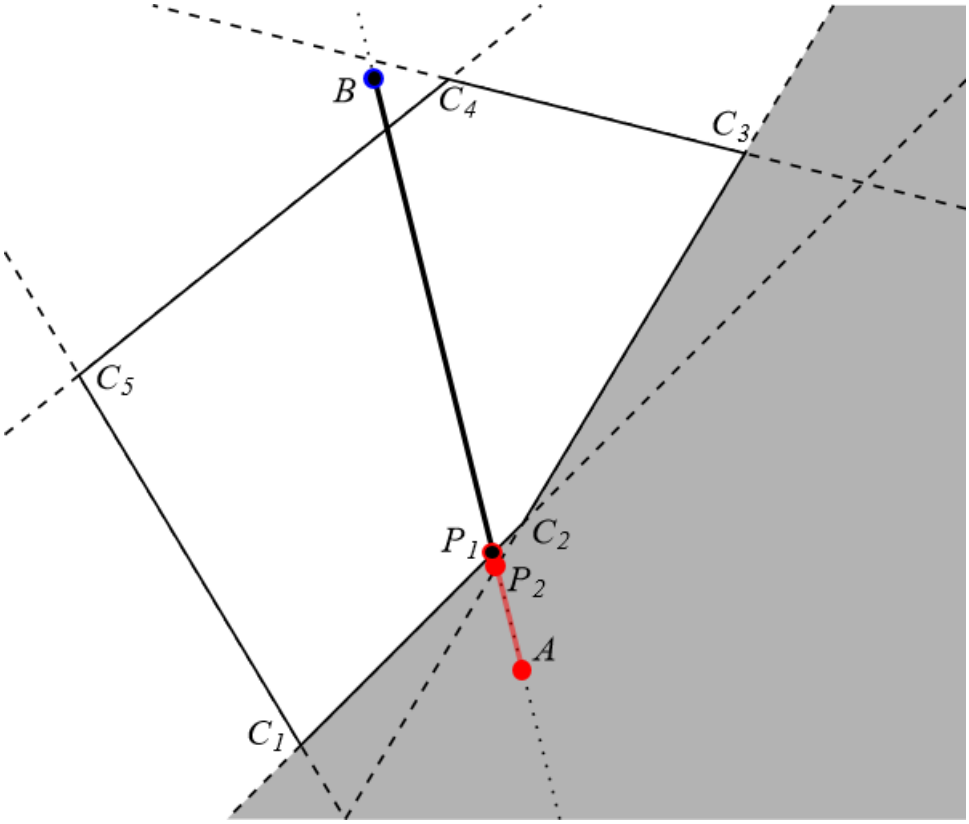


Рисунок 2.11 – На очередной итерации алгоритма Кируса-Бека отсечения не происходит

Переходим далее к ребру C_3C_4:

$$\overrightarrow{C_3C_4}(-4,1), \overrightarrow{AC_3}(3,7),$$
$$p_3 = \begin{vmatrix} -2 & 8 \\ -4 & 1 \end{vmatrix} = 30, q_3 = \begin{vmatrix} 3 & 7 \\ -4 & 1 \end{vmatrix} = 31,$$
$$30t \le 31 \Rightarrow t \le 31/30.$$

Здесь тоже не происходит отсечений, ведь полученное значение превышает $t_{\text{кон}} = 1$ (Рисунок 2.12).

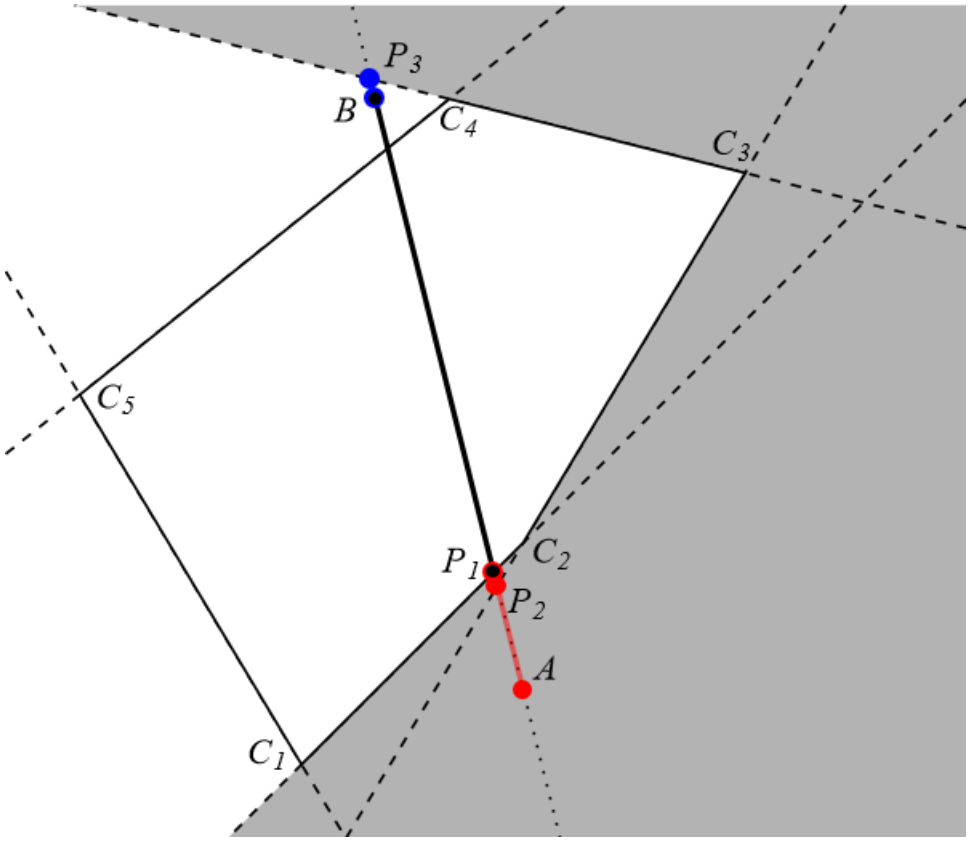


Рисунок 2.12 – Третья итерация алгоритма Кируса-Бека

Для ребра C_4C_5 получаются следующие вычисления:

$$\overrightarrow{C_4C_5}(-5,-4), \overrightarrow{AC_4}(-1,8),$$

$$p_4 = \begin{vmatrix} -2 & 8 \\ -5 & -4 \end{vmatrix} = 48, q_4 = \begin{vmatrix} -1 & 8 \\ -5 & -4 \end{vmatrix} = 44,$$

$$48t \le 44 \Rightarrow t \le 11/12.$$

На данной итерации уже имеет место отсечение с конца: $t_{\text{кон}} := 11/12$ (Рисунок 2.13). Проверяя неравенство $t_{\text{нач}} < t_{\text{кон}}$, убеждаемся, что полного отсечения снова не происходит.

Наконец, для последнего ребра C_5C_1 получаются следующие результаты:

$$\overrightarrow{C_5C_1}(3,-5), \overrightarrow{AC_5}(-6,4),$$

$$p_5 = \begin{vmatrix} -2 & 8 \\ 3 & -5 \end{vmatrix} = -14, q_5 = \begin{vmatrix} -1 & 8 \\ -6 & 4 \end{vmatrix} = 44,$$

$$-14t \le 44 \Rightarrow t \ge -22/7.$$

Отсечения с начала отрезка не происходит, так как $-22/7 < t_{\text{нач}} = 1/5$ (Рисунок 2.14). Таким образом, результатом алгоритма Кируса-Бека является числовой отрезок $[t_{\text{нач}}, t_{\text{кон}}] = [1/5, 11/12]$. Подставляя эти значения в параметрические уравнения отрезка AB:

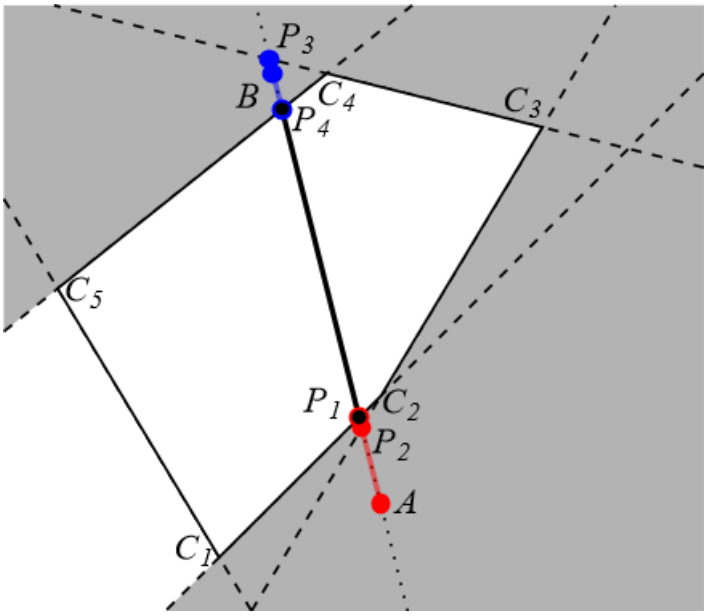


Рисунок 2.13 – Четвёртая итерация алгоритма Кируса-Бека

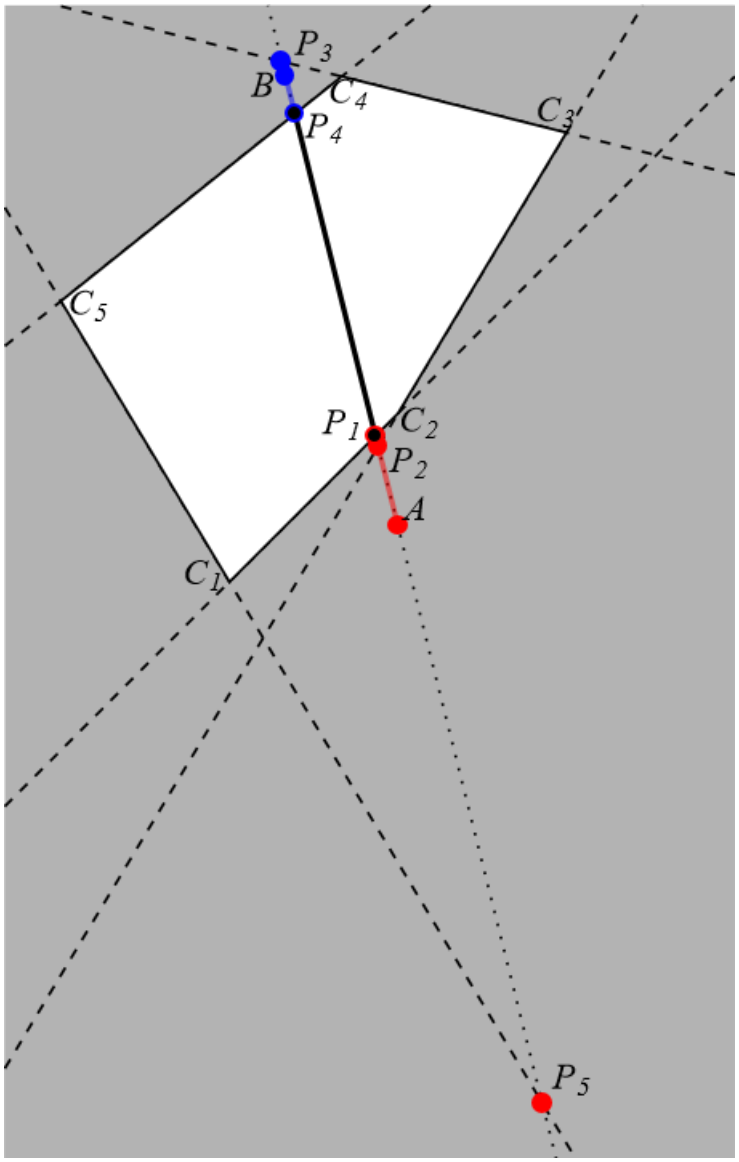


Рисунок 2.14 – Заключительная итерация алгоритма Кируса-Бека

$$\begin{cases} x = -2t, \\ y = -3 + 8t, \end{cases}$$

получим координаты концов искомого отрезка: $P_1\left(-\frac{2}{5}, -\frac{7}{5}\right)$ и $P_4\left(-\frac{11}{6}, \frac{13}{3}\right)$.

2.1.4. Алгоритм Сазерленда-Ходжмана

Алгоритм Сазерленда-Ходжмана (стр. 35) предназначен для отсечения произвольного полигона отсекателем, имеющим форму выпуклого полигона.

Пример 1.8. Осуществить отсечение полигона $A_1A_2A_3A_4A_1$ выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$, где вершины имеют следующие координаты:

$$A_1(1,-4), A_2(11,-1), A_3(2,-1), A_4(0,5),$$
$$C_1(6,-2), C_2(11,5), C_3(9,13), C_4(1,12), C_5(-3,3).$$

Будем проводить последовательное отсечение рёбрами C_1C_2, C_2C_3, C_3C_4, C_4C_5 и C_5C_1. Вначале необходимо определить направление обхода отсекателя:

$$\operatorname{sgn} \angle(\overrightarrow{C_1C_2}, \overrightarrow{C_2C_3}) = \operatorname{sgn} \begin{vmatrix} 5 & 7 \\ -2 & 8 \end{vmatrix} = +1.$$

Значит, заданный порядок вершин обеспечивает левосторонний обход отсекателя.

Рассматривая первое ребро C_1C_2, определим, по какую сторону (внутреннюю или внешнюю) лежат от него вершины отсекаемого полигона $A_1A_2A_3A_4A_1$. Для этого составим общее уравнение прямой C_1C_2, причём в качестве вектора нормали возьмём внутреннюю нормаль отсекателя. Для полигона с левосторонним обходом он получается, если направляющий вектор ребра повернуть на прямой угол против часовой стрелки. Матрица такого линейного преобразования равна $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, и тогда вектор с координатами (x, y) переходит в вектор с координатным столбцом $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}$. Итак, для ребра C_1C_2 имеем следующие вычисления:

$$\overrightarrow{C_1C_2}(5,7) \perp (-7,5),$$
$$C_1C_2 : -7x + 5y + 52 = 0. \qquad (2.3)$$

Это уравнение позволяет определить, по какую сторону от ребра C_1C_2 расположена некоторая точка: если после подстановки её координат в левую часть уравнения (2.3) получается значение, большее нуля, то точка находится на внутренней стороне, если меньше – на внешней, если получился ноль – точка лежит на прямой.

Подставим координаты точек A_1 и A_2 в левую часть уравнения (2.3):

$$A_1 : -7 \cdot 1 + 5(-4) + 52 = -7 - 20 + 52 > 0,$$
$$A_2 : -7 \cdot 11 + 5(-1) + 52 = -77 - 5 + 52 < 0.$$

Значит, ребро A_1A_2 отсекаемого полигона пересекает текущее ребро отсекателя, причём имеет место переход изнутри отсекателя наружу. Значит, в список новых вершин необходимо добавить точку пересечения двух отрезков. Для её вычисления составим уравнение прямой A_1A_2:

$$\overrightarrow{A_1A_2}(10,3) \perp (-3,10),$$
$$A_1A_2 : -3x + 10y + 43 = 0.$$

Составим из уравнений двух прямых систему, которую затем можно решить, например, по правилу Крамера:

$$A_1A_2 \cap C_1C_2 : \begin{cases} -3x + 10y = -43, \\ -7x + 5y = -52, \end{cases}$$
$$\Delta = \begin{vmatrix} -3 & 10 \\ -7 & 5 \end{vmatrix} = 55, \Delta_x = \begin{vmatrix} -43 & 10 \\ -52 & 5 \end{vmatrix} = -200 - 15 + 520 = 305,$$
$$\Delta_y = \begin{vmatrix} -3 & -43 \\ -7 & -52 \end{vmatrix} = 156 - 280 - 21 = -145,$$
$$x = \frac{\Delta_x}{\Delta} = \frac{305}{55} = \frac{61}{11}, y = \frac{\Delta_y}{\Delta} = \frac{-145}{55} = -\frac{29}{11}.$$

Значит, первой вершиной усечённого после первой итерации полигона является точка $A_1^1\left(\frac{61}{11}, -\frac{29}{11}\right)$.

Переходим затем к ребру A_2A_3:

$$A_2 : -7 \cdot 11 + 5(-1) + 52 = -77 - 5 + 52 < 0,$$
$$A_3 : -7 \cdot 2 + 5(-1) + 52 = -14 - 5 + 52 > 0.$$

Значит, ребро A_2A_3 также пересекает текущее ребро отсекателя, но теперь уже происходит переход из внешней области во внутреннюю. Следовательно, в новый список вершин нужно добавить вначале точку пересечения, затем концевую точку A_3. Вычислим пересечение двух отрезков:

$$\overrightarrow{A_2A_3}(-9,0) \perp (0,-1),$$
$$A_2A_3 : -y - 1 = 0,$$
$$A_2A_3 \cap C_1C_2 : \begin{cases} -y = 1, \\ -7x + 5y = -52. \end{cases}$$

Решение этой системы даёт значения $x = 47/7$, $y = -1$. Таким образом, получаем следующие вершины для усечённого полигона: $A_2^1(47/7, -1)$, $A_3^1 = A_3(2,-1)$.

Ребро A_3A_4 оказывается целиком лежащим на внутренней стороне относительно текущего ребра отсекателя:

$$A_3 : -7 \cdot 2 + 5(-1) + 52 = -14 - 5 + 52 > 0,$$
$$A_4 : -7 \cdot 0 + 5 \cdot 5 + 52 = 25 + 52 > 0.$$

Это означает, что нужно добавить конечную вершину этого ребра в список новых вершин: $A_4^1 = A_4(0,5)$.

Наконец, ребро A_4A_1 также находится целиком на внутренней стороне:

$$A_4 : -7 \cdot 0 + 5 \cdot 5 + 52 = 25 + 52 > 0,$$
$$A_1 : -7 \cdot 1 + 5(-4) + 52 = -7 - 20 + 52 > 0,$$

и добавляется последняя вершина $A_5^1 = A_1(1,-4)$. Таким образом, после первой итерации произошло отсечение полигона $A_1A_2A_3A_4A_1$ ребром C_1C_2, в результате чего получился полигон $A_1^1A_2^1A_3^1A_4^1A_5^1A_1^1$, вершины которого имеют координаты

$$A_1^1\left(\frac{61}{11}, -\frac{29}{11}\right), A_2^1\left(\frac{47}{7}, -1\right), A_3^1(2,-1), A_4^1(0,5), A_5^1(1,-4).$$

Результат работы первой итерации алгоритма Сазерленда-Ходжмана представлен на рисунке 2.15.

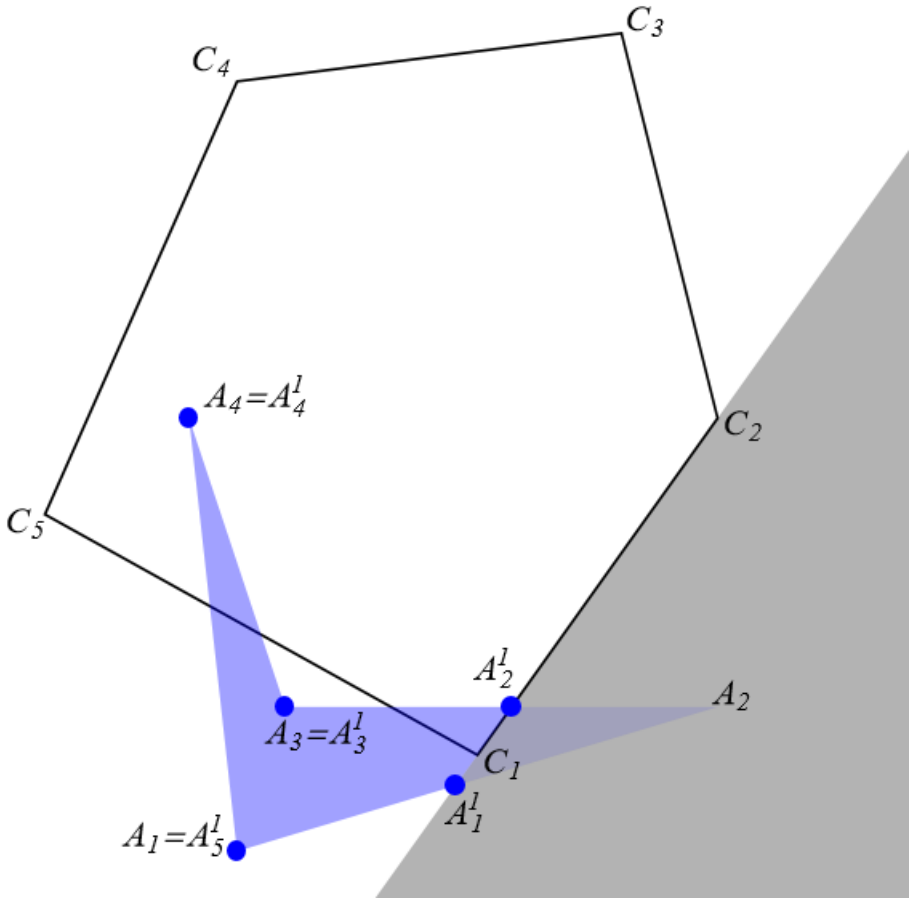


Рисунок 2.15 – Отсечение полигона одним ребром выпуклого отсекателя

Во время выполнения второй итерации алгоритма рассматривается ребро отсекателя C_2C_3:

$$\overrightarrow{C_2C_3}(-2,8) \perp (-8,-2) \uparrow\uparrow (-4,-1),$$
$$C_2C_3 : -4x - y + 49 = 0.$$

Все рёбра полигона $A_1^1A_2^1A_3^1A_4^1A_5^1A_1^1$ оказываются лежащими на внутренней стороне от текущего ребра отсекателя:

$$A_1^1 : -4 \cdot \frac{61}{11} - \left(-\frac{29}{11}\right) + 49 = \frac{1}{11}(-4 \cdot 61 + 29 + 49 \cdot 11) = \frac{1}{11}(-244 + 29 + 490 + 49) > 0,$$
$$A_2^1 : -4 \cdot \frac{47}{7} - (-1) + 49 = \frac{1}{7}(-4 \cdot 47 + 7 + 49 \cdot 7) > 0,$$
$$A_3^1 : -4 \cdot 2 - (-1) + 49 > 0,$$
$$A_4^1 : -4 \cdot 0 - 5 + 49 > 0,$$
$$A_5^1 : -4 \cdot 1 - (-4) + 49 > 0,$$

$$A_1^1 : \frac{1}{11}(-244 + 29 + 490 + 49) > 0.$$

При этом в новый список добавляется вначале вторая вершина, затем третья, четвёртая, пятая и в последнюю очередь – первая:

$$A_1^2\left(\frac{47}{7}, -1\right), A_2^2(2,-1), A_3^2(0,5), A_4^2(1,-4), A_5^2\left(\frac{61}{11}, -\frac{29}{11}\right).$$

На третьей итерации снова получается, что относительно ребра отсекателя C_3C_4 все рёбра усечённого полигона расположены на внутренней стороне:

$$\overrightarrow{C_3C_4}(-8,-1) \perp (1,-8),$$
$$C_3C_4 : x - 8y + 95 = 0,$$
$$A_1^2 : \frac{47}{7} - 8(-1) + 95 > 0,$$
$$A_2^2 : 2 - 8(-1) + 95 > 0,$$
$$A_3^2 : 0 - 8 \cdot 5 + 95 > 0,$$
$$A_4^2 : 1 - 8(-4) + 95 > 0,$$
$$A_5^2 : \frac{61}{11} - 8\left(-\frac{29}{11}\right) + 95 > 0,$$
$$A_1^2 : \frac{47}{7} - 8(-1) + 95 > 0.$$

Список вершин после третьей итерации:

$$A_1^3(2,-1), A_2^3(0,5), A_3^3(1,-4), A_4^3\left(\frac{61}{11}, -\frac{29}{11}\right), A_5^3\left(\frac{47}{7}, -1\right).$$

Четвёртое ребро отсекателя C_4C_5 также не осуществляет никаких отсечений:

$$\overrightarrow{C_4C_5}(-4,-9) \perp (9,-4),$$
$$C_4C_5 : 9x - 4y + 39 = 0,$$
$$A_1^3 : 9 \cdot 2 - 4(-1) + 39 > 0,$$
$$A_2^3 : 9 \cdot 0 - 4 \cdot 5 + 39 = -20 + 39 > 0,$$
$$A_3^3 : 9 \cdot 1 - 4(-4) + 39 > 0,$$

$$A_4^3 : 9 \cdot \frac{61}{11} - 4\left(-\frac{29}{11}\right) + 39 > 0,$$

$$A_5^3 : 9 \cdot \frac{47}{7} - 4(-1) + 39 > 0,$$

$$A_1^3 : 9 \cdot 2 - 4(-1) + 39 > 0.$$

Список вершин после четвёртой итерации:

$$A_1^4(0,5), A_2^4(1,-4), A_3^4\left(\frac{61}{11}, -\frac{29}{11}\right), A_4^4\left(\frac{47}{7}, -1\right), A_5^4(2,-1).$$

На пятой, заключительной итерации рассматривается ребро C_5C_1, описываемое следующим уравнением:

$$\overrightarrow{C_5C_1}(9,-5) \perp (5,9),$$
$$C_5C_1 : 5x + 9y - 12 = 0.$$

Определим расположение ребра $A_1^4A_2^4$ относительно него:

$$A_1^4 : 5 \cdot 0 + 9 \cdot 5 - 12 = 45 - 12 > 0,$$
$$A_2^4 : 5 \cdot 1 + 9(-4) - 12 = 5 - 36 - 12 < 0,$$

т.е. происходит пересечение отсекателя и переход из внутренней стороны на внешнюю. Значит, необходимо в новый список добавить одну вершину – точку пересечения:

$$\overrightarrow{A_1^4A_2^4}(1,-9) \perp (9,1),$$
$$A_1^4A_2^4 : 9x + y - 5 = 0,$$
$$A_1^4A_2^4 \cap C_5C_1 : \begin{cases} 9x + y = 5, \\ 5x + 9y = 12, \end{cases}$$
$$\Delta = \begin{vmatrix} 9 & 1 \\ 5 & 9 \end{vmatrix} = 76, \Delta_x = \begin{vmatrix} 5 & 1 \\ 12 & 9 \end{vmatrix} = 33, \Delta_y = \begin{vmatrix} 9 & 5 \\ 5 & 12 \end{vmatrix} = 83,$$
$$x = \frac{\Delta_x}{\Delta} = \frac{33}{76}, y = \frac{\Delta_y}{\Delta} = \frac{83}{76}.$$

Таким образом, вычислили первую вершину нового полигона $A_1^5\left(\frac{33}{76}, \frac{83}{76}\right)$.

Для следующего ребра отсекаемого полигона имеют место следующие вычисления:

$$A_2^4 : 5\cdot 1 + 9(-4) - 12 = 5 - 36 - 12 < 0,$$

$$A_3^4 : 5\cdot\frac{61}{11} + 9\left(-\frac{29}{11}\right) - 12 = \frac{1}{11}(5\cdot 61 - 9\cdot 29 - 12\cdot 11) = \frac{1}{11}(305 - 290 + 29 - 132) < 0.$$

Значит, ребро целиком лежит за пределами отсекателя и не порождает новых вершин.

Ребро $A_3^4 A_4^4$ уже пересекает ребро отсекателя:

$$A_3^4 : \frac{1}{11}(305 - 290 + 29 - 132) < 0.$$

$$A_4^4 : 5\cdot\frac{47}{7} + 9(-1) - 12 = \frac{1}{7}(5\cdot 47 - 9\cdot 7 - 12\cdot 7) = \frac{1}{7}(235 - 147) > 0.$$

Так как теперь происходит переход с внешней на внутреннюю сторону относительно текущего ребра отсекателя, то необходимо добавить две вершины: точку пересечения двух отрезков и концевую вершину A_4^4. Вычислим пересечение:

$$\overrightarrow{A_3^4 A_4^4}\left(\frac{47}{7} - \frac{61}{11}, -1 + \frac{29}{11}\right) = \left(\frac{47\cdot 11 - 61\cdot 7}{77}, \frac{18}{11}\right) = \left(\frac{90}{77}, \frac{18}{11}\right) \| (5,7) \perp (-7,5),$$

$$A_3^4 A_4^4 : -7x + 5y + 52 = 0,$$

$$A_3^4 A_4^4 \cap C_5 C_1 : \begin{cases} -7x + 5y = -52, \\ 5x + 9y = 12, \end{cases}$$

$$\Delta = \begin{vmatrix} -7 & 5 \\ 5 & 9 \end{vmatrix} = -88, \Delta_x = \begin{vmatrix} -52 & 5 \\ 12 & 9 \end{vmatrix} = -520 + 52 - 60 = -528,$$

$$\Delta_y = \begin{vmatrix} -7 & -52 \\ 5 & 12 \end{vmatrix} = -84 + 260 = 176,$$

$$x = \frac{\Delta_x}{\Delta} = \frac{-528}{-88} = 6, y = \frac{\Delta_y}{\Delta} = \frac{176}{-88} = -2.$$

Добавляются в список новых вершин две точки с координатами $A_2^5(6,-2)$ и $A_3^5 = A_4^4\left(\frac{47}{7}, -1\right)$.

Переходим далее к ребру $A_4^4 A_5^4$:

$$A_4^4 : \frac{1}{7}(235-147) > 0,$$

$$A_5^4 : 5 \cdot 2 + 9(-1) - 12 = 10 - 9 - 12 < 0.$$

Значит, на данном шаге необходимо добавить одну вершину – точку пересечения двух отрезков:

$$\overrightarrow{A_4^4 A_5^4}\left(2 - \frac{47}{7}, 0\right) \| (1,0) \perp (0,1),$$

$$A_4^4 A_5^4 : y + 1 = 0,$$

$$A_3^4 A_4^4 \cap C_5 C_1 : \begin{cases} y = -1, \\ 5x + 9y = 12, \end{cases}$$

$$x = \frac{21}{5}, y = -1, A_4^5\left(\frac{21}{5}, -1\right).$$

Наконец, рассмотрим ребро $A_5^4 A_1^4$:

$$A_5^4 : 10 - 9 - 12 < 0,$$

$$A_1^4 : 5 \cdot 0 + 9 \cdot 5 - 12 = 45 - 12 > 0,$$

поэтому нужно добавить точку пересечения двух отрезков и концевую вершину:

$$\overrightarrow{A_5^4 A_1^4}(-2,6) \perp (3,1),$$

$$A_5^4 A_1^4 : 3x + y - 5 = 0,$$

$$A_5^4 A_1^4 \cap C_5 C_1 : \begin{cases} 3x + y = 5, \\ 5x + 9y = 12, \end{cases}$$

$$\Delta = \begin{vmatrix} 3 & 1 \\ 5 & 9 \end{vmatrix} = 22, \Delta_x = \begin{vmatrix} 5 & 1 \\ 12 & 9 \end{vmatrix} = 33, \Delta_y = \begin{vmatrix} 3 & 5 \\ 5 & 12 \end{vmatrix} = 11,$$

$$A_5^5\left(\frac{33}{22}, \frac{11}{22}\right) = \left(\frac{3}{2}, \frac{1}{2}\right), A_6^5 = A_1^4(0,5).$$

Таким образом, окончательно получаем полигон $A_1^5 A_2^5 A_3^5 A_4^5 A_5^5 A_6^5 A_1^5$, вершины которого имеют следующие координаты:

$$A_1^5\left(\frac{33}{76},\frac{83}{76}\right), A_2^5(6,-2), A_3^5\left(\frac{47}{7},-1\right), A_4^5\left(\frac{21}{5},-1\right), A_5^5\left(\frac{3}{2},\frac{1}{2}\right), A_6^5(0,5). \qquad (2.4)$$

Результат отсечения представлен на рисунке 2.16. Как видно, вершины (2.4) образуют несвязную область, представимую в виде объединения двух треугольников $\triangle A_1^5 A_5^5 A_6^5$ и $\triangle A_2^5 A_3^5 A_4^5$. Получить эти треугольники из списка (2.4) можно, например, воспользовавшись алгоритмом со страницы 40.

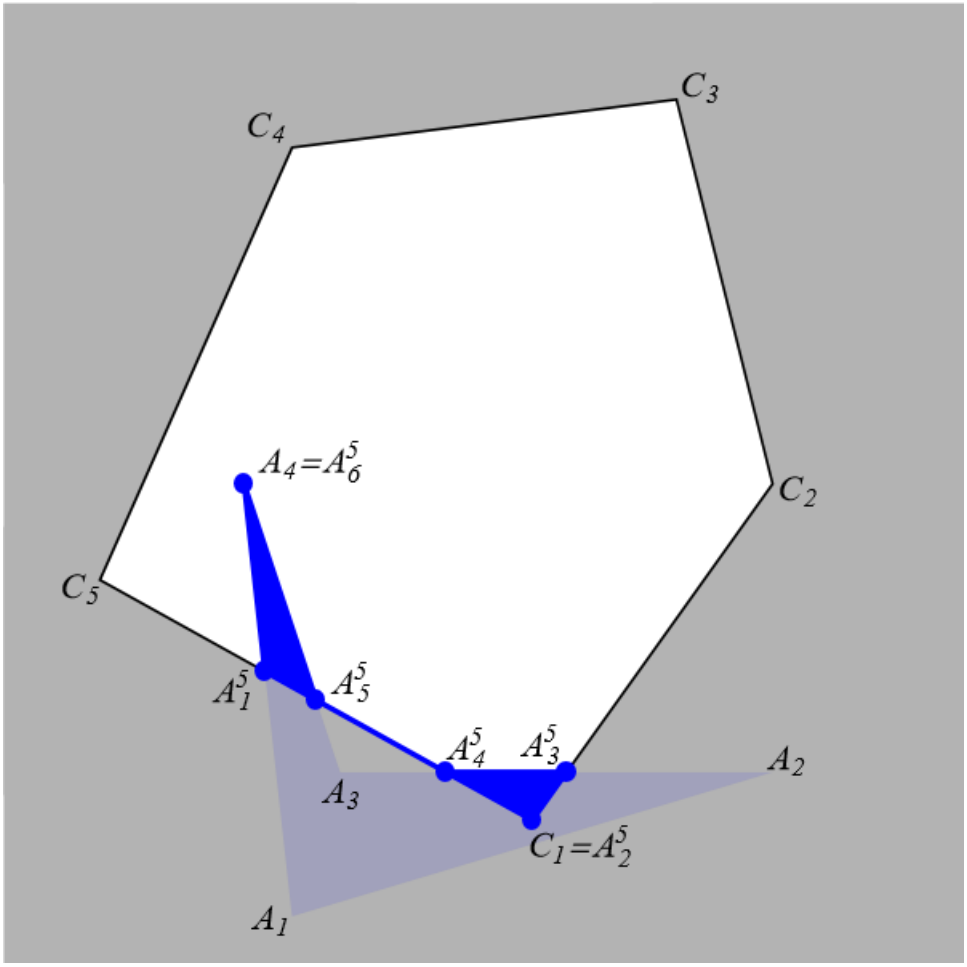


Рисунок 2.16 – Результат отсечения полигона выпуклым отсекателем по алгоритму Сазерленда-Ходжмана

2.1.5. Алгоритм Бентли-Оттманна

Алгоритм Бентли-Оттманна (стр. 47) предназначен для нахождения пересечения отрезков из заданного множества. Как было отмечено в замечаниях к этому алгоритму, в среднем он работает быстрее, чем нахождение пересечений для всех возможных пар отрезков. При предположении, что $k = O(m)$, где k – число искомых пересечений, m – число самих отрезков, временная трудоёмкость этого алгоритма равна $O(m \log m)$.

Пример 1.9. Найти все пересечения отрезков A_iB_i, $i=\overline{1,4}$:

$$A_1(2,9), \quad B_1(10,10),$$
$$A_2(0,9), \quad B_2(8,2),$$
$$A_3(5,2), \quad B_3(6,6),$$
$$A_4(0,7), \quad B_4(8,5).$$

Для начала составим из всех этих точек приоритетную очередь P, где они расположены в порядке возрастания абсцисс. При равенстве абсцисс рекомендуется полагать точки начал более приоритетными по сравнению с точками пересечений, которые, в свою очередь, имеют приоритет перед точками концов.

$$P := \{A_4, A_2, A_1, A_3, B_3, B_2, B_4, B_1\}.$$

Другая структура данных, S, пока является пустой: $S := \varnothing$. В ней будем располагать отрезки в порядке возрастания ординаты их точки пересечения с заметающей прямой.

Теперь начнём процесс сканирования точек и отрезков. Из очереди P вначале извлекается точка A_4. Это точка начала отрезка A_4B_4, поэтому добавляем его в структуру S. При этом в силу отсутствия в S соседей у нового отрезка пересечения пока не вычисляются. После этой итерации две структуры выглядят следующим образом:

$$P := \{\cancel{A_4}, A_2, A_1, A_3, B_3, B_2, B_4, B_1\},$$
$$S := \{4\}.$$

Далее идёт начало другого отрезка, точка A_2. При добавлении нового отрезка в S необходимо определить, пересекает ли он заметающую прямую, которая теперь проходит через A_2, выше или ниже 4-го отрезка, записанного в S на предыдущей итерации. Это можно определить исходя из положения A_2 относительно отрезка A_4B_4 (Рисунок 2.17):

$$\overrightarrow{A_4B_4}(8,-2), \overrightarrow{A_4A_2}(0,2),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_4B_4}, \overrightarrow{A_4A_2}\right) = \operatorname{sgn}\begin{vmatrix} 8 & -2 \\ 0 & 2 \end{vmatrix} = +1.$$

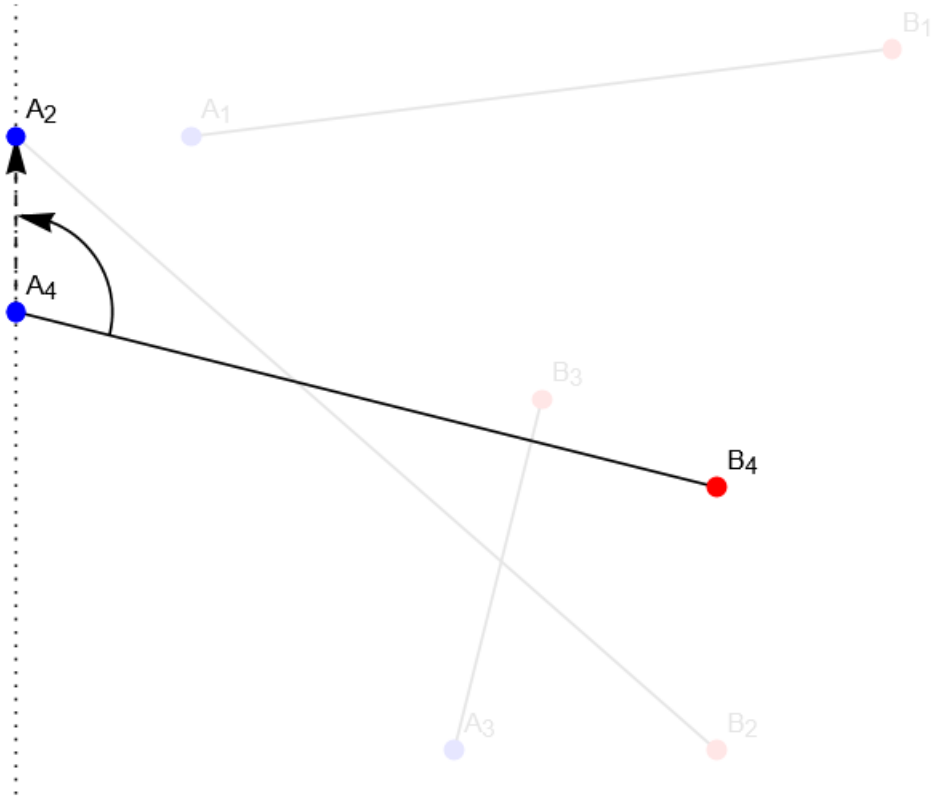


Рисунок 2.17 – Определение положения нового отрезка для его размещения в структуре S

Значит, 2-й отрезок пересекает заметающую прямую выше 4-го, и располагаться в S он должен после него. Кроме того, необходимо выяснить, пересекаются ли эти два отрезка:

$$\begin{gathered}\overrightarrow{A_2B_2}(8,-7)\perp(7,8),\\ A_2B_2: 7x+8y-72=0,\\ \overrightarrow{A_4B_4}(8,-2)\perp(1,4),\\ A_4B_4: x+4y-28=0,\\ \begin{cases}7x+8y-72=0,\\ x+4y-28=0\end{cases}\Rightarrow x=\frac{16}{5}, y=\frac{31}{5}.\end{gathered} \tag{2.5}$$

Сравнивая полученные координаты с координатами начал и концов обоих отрезков, нетрудно убедиться в том, что точка $C_{24}\left(\frac{16}{5},\frac{31}{5}\right)$ пересечения прямых A_2B_2 и A_4B_4 действительно лежит на обоих отрезках. Значит, отрезки A_2B_2 и A_4B_4 пересекаются в точке C_{24}, которую нужно теперь занести в P с учётом соблюдения приоритетов:

$$P := \{\cancel{A_2}, A_1, \boxed{C_{24}}, A_3, B_3, B_2, B_4, B_1\},$$
$$S := \{4, \boxed{2}\}.$$

Далее из P извлекаем точку A_1 – начало ещё одного отрезка. Поступая аналогично предыдущей итерации, определяем положение этой точки относительно отрезков, записанных в S:

$$\overrightarrow{A_4A_1}(2,2),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_4B_4}, \overrightarrow{A_4A_1}\right) = \operatorname{sgn}\begin{vmatrix} 8 & -2 \\ 2 & 2 \end{vmatrix} = +1,$$

т.е. 1-й отрезок следует размещать после 4-ого. Значит, нужно ещё проверить положение A_1 относительно отрезка A_2B_2:

$$\overrightarrow{A_2A_1}(2,0),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2}, \overrightarrow{A_2A_1}\right) = \operatorname{sgn}\begin{vmatrix} 8 & -7 \\ 2 & 0 \end{vmatrix} = +1.$$

Таким образом, $S := \{4, 2, \boxed{1}\}$. В структуре S 1-й отрезок соседствует со 2-м, значит, необходимо найти пересечения этих отрезков:

$$A_2B_2 : 7x + 8y - 72 = 0,$$
$$\overrightarrow{A_1B_1}(8,1) \perp (1,-8),$$
$$A_1B_1 : x - 8y + 70 = 0,$$
$$\begin{cases} 7x + 8y - 72 = 0, \\ x - 8y + 70 = 0 \end{cases} \Rightarrow x = \frac{1}{4}, y = \frac{281}{32}.$$

Найденная точка $C_{12}\left(\frac{1}{4}, \frac{281}{32}\right)$ пересечения двух прямых, однако, не является точкой пересечения заданных отрезков, так как её абсцисса меньше абсциссы точки $A_1(2,9)$, т.е. C_{12} не лежит на отрезке A_1B_1. Значит, никакие точки пересечения на этой итерации добавляться в P не будут:

$$P := \{\cancel{A_1}, C_{24}, A_3, B_3, B_2, B_4, B_1\}.$$

Далее в структуре P следует уже точка пересечения C_{24}. Поэтому необходимо в S найти отрезки 2 и 4 и поменять их местами:

$$S := \{\underline{2}, 4, 1\}.$$

Теперь у отрезка 4 появился новый сосед – отрезок 1, поэтому нужно искать их пересечение:

$$\begin{gathered} A_4B_4 : x + 4y - 28 = 0, \\ A_1B_1 : x - 8y + 70 = 0, \\ \begin{cases} x + 4y - 28 = 0, \\ x - 8y + 70 = 0 \end{cases} \Rightarrow x = -\frac{14}{3}, y = \frac{49}{6}. \end{gathered} \tag{2.6}$$

Несложно видеть, что пересечение этих прямых не является пересечением отрезков. Значит, новых пересечений снова не будет добавлено:

$$P := \{\cancel{C_{24}}, A_3, B_3, B_2, B_4, B_1\}.$$

На следующей итерации из P извлекается точка начала нового отрезка A_3. В структуре S содержится уже три отрезка, поэтому целесообразно вначале проверить расположение точки A_3 относительно среднего из них:

$$\begin{gathered} \overrightarrow{A_4A_3}(5, -5), \\ \operatorname{sgn} \angle\left(\overrightarrow{A_4B_4}, \overrightarrow{A_4A_3}\right) = \operatorname{sgn} \begin{vmatrix} 8 & -2 \\ 5 & -5 \end{vmatrix} = \operatorname{sgn}(-40 + 10) = -1. \end{gathered}$$

Значит, 3-й отрезок нужно разместить в S перед 4-м. Остаётся рассмотреть положение A_3 относительно спереди стоящего отрезка A_2B_2:

$$\begin{gathered} \overrightarrow{A_2A_3}(5, -7), \\ \operatorname{sgn} \angle\left(\overrightarrow{A_2B_2}, \overrightarrow{A_2A_3}\right) = \operatorname{sgn} \begin{vmatrix} 8 & -7 \\ 5 & -7 \end{vmatrix} = \operatorname{sgn}(-7 \cdot (8 - 5)) = -1. \end{gathered}$$

Таким образом, структура S станет равной $\{\boxed{3}, 2, 4, 1\}$. Значит, нужно вычислять пересечение отрезков A_3B_3 и A_2B_2:

$$\begin{gathered} A_2B_2 : 7x + 8y - 72 = 0, \\ \overrightarrow{A_3B_3}(1, 4) \perp (4, -1), \\ A_3B_3 : 4x - y - 18 = 0, \end{gathered}$$

$$\begin{cases} 7x+8y-72=0, \\ 4x-y-18=0 \end{cases} \Rightarrow x=\frac{72}{13}, y=\frac{54}{13}.$$

Точка пересечения двух прямых $C_{23}\left(\frac{72}{13},\frac{54}{13}\right)$ является также точкой пересечения двух отрезков. Значит, нужно эту точку занести в P:

$$P := \{\cancel{A_3}, \boxed{C_{23}}, B_3, B_2, B_4, B_1\}.$$

Эта же точка сразу извлекается на следующей итерации. Поэтому нужно два отрезка поменять местами в S:

$$S := \{\underline{2,3}, 4, 1\},$$

после чего искать пересечение отрезков A_3B_3 и A_4B_4, ставших соседями в S:

$$A_4B_4 : x+4y-28=0,$$
$$A_3B_3 : 4x-y-18=0,$$
$$\begin{cases} x+4y-28=0, \\ 4x-y-18=0 \end{cases} \Rightarrow x=\frac{100}{17}, y=\frac{94}{17}.$$

Получили точку пересечения прямых $C_{34}\left(\frac{100}{17},\frac{94}{17}\right)$, которая также является точкой пересечения двух отрезков. Значит, её тоже добавляем в P:

$$P := \{\cancel{C_{23}}, \boxed{C_{34}}, B_3, B_2, B_4, B_1\}.$$

После этого из P извлекаем эту же точку, что приводит к перестановке соответствующих отрезков местами в S:

$$S := \{2, \underline{4,3}, 1\},$$

Тут, вообще говоря, надо теперь проверять пересечения двух пар отрезков, ставшими соседями в S: A_2B_2 и A_4B_4, A_3B_3 и A_1B_1. Для первой пары уже найдено пересечение $C_{24}\left(\frac{16}{5},\frac{31}{5}\right)$ (см. (2.5)). Значит, остаётся вторая пара:

$$A_1B_1 : x - 8y + 70 = 0,$$
$$A_3B_3 : 4x - y - 18 = 0,$$
$$\begin{cases} x - 8y + 70 = 0, \\ 4x - y - 18 = 0 \end{cases} \Rightarrow x = \frac{214}{31}, y = \frac{298}{31}.$$

Точка $C_{13}\left(\frac{214}{31}, \frac{298}{31}\right)$ однако, не лежит на отрезке A_3B_3, а значит, не является точкой пересечения двух отрезков. После этой итерации, таким образом, имеем структуру P следующего вида:

$$P := \{\cancel{C_{34}}, B_3, B_2, B_4, B_1\}.$$

Следующая точка B_3 представляет уже конец отрезка 3. Значит, из S удаляем 3-й отрезок:

$$S := \{2, 4, \cancel{3}, 1\}.$$

Для отрезков A_1B_1 и A_4B_4, ставших теперь соседями в S, вычисления уже проведены ранее (см. (2.6)) – два отрезка не пересекаются. После этой итерации получаем

$$P := \{\cancel{B_3}, B_2, B_4, B_1\}.$$

Дальнейшие все итерации тривиальны: все точки, оставшиеся в P, по порядку извлекаются оттуда, при этом соответствующие отрезки удаляются из S, причём в каждой итерации удаляемый отрезок оказывается крайним, и после этого удаления ни для каких двух отрезков не нужно вычислять пересечения. Таким образом, нашли следующий список точек пересечения отрезков:

$$C_{24}\left(\frac{16}{5}, \frac{31}{5}\right),\ C_{23}\left(\frac{72}{13}, \frac{54}{13}\right),\ C_{34}\left(\frac{100}{17}, \frac{94}{17}\right).$$

Процесс сканирования заметающей прямой при выполнении алгоритма Бентли-Оттманна продемонстрирован на рисунке 2.18, где синим и красным цветами помечены соответственно точки начал и концов отрезков, хранящихся в структуре P на данной итерации, пунктиром отмечены отрезки, хранящиеся в S, чёрным жирным шрифтом подписана точка, извлекаемая из P на данной итерации, а фиолетовые точки – это найденные пересечения.

Замечание 1. Вместо вычисления пересечений прямых и последующего определения, лежат ли они на заданных отрезках, можно вначале проверять

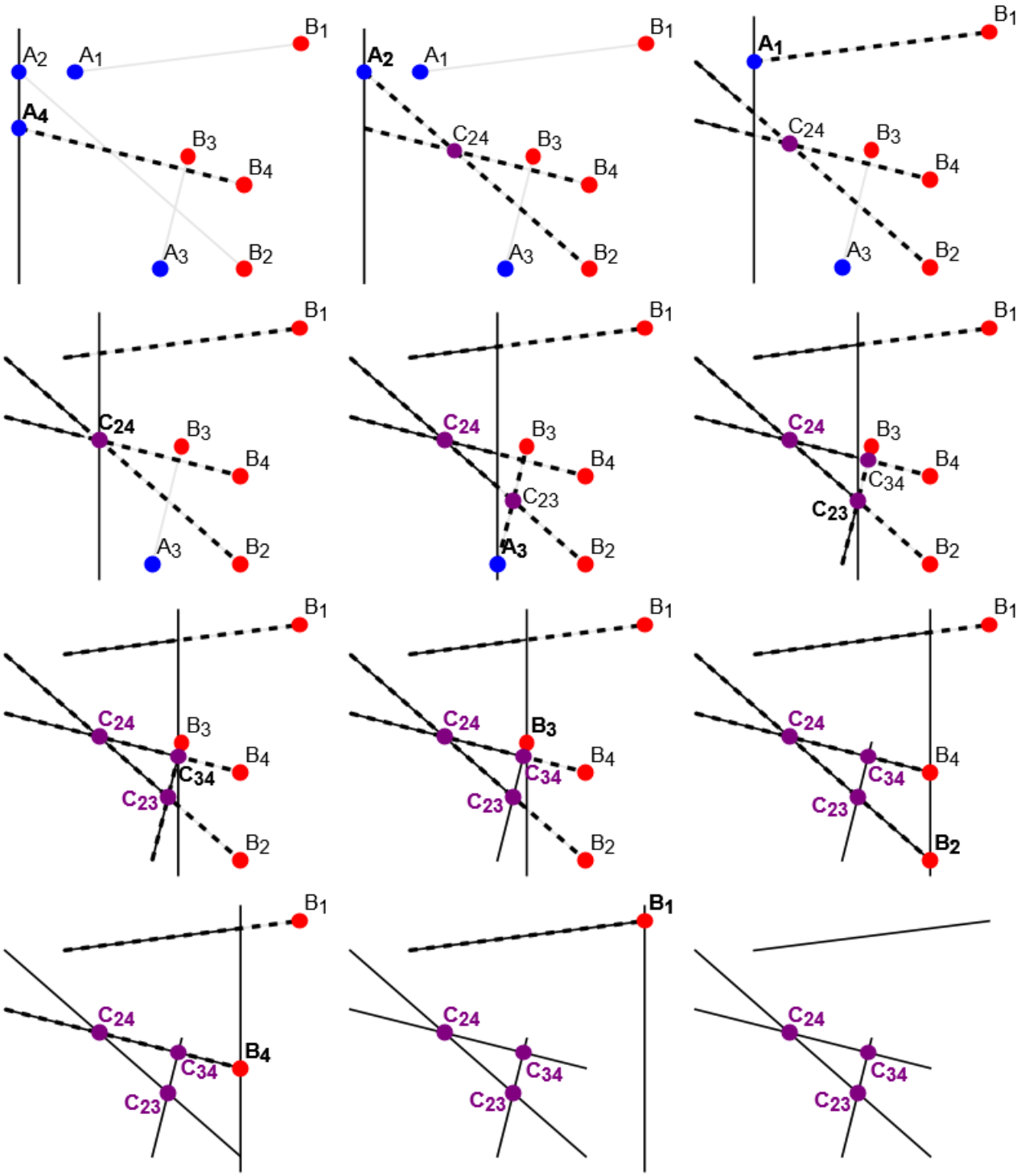


Рисунок 2.18 – Пример работы алгоритма Бентли-Оттманна для четырёх отрезков

условия пересекаемости двух отрезков (например, приведённые в первой части ЭУМК [1, п. 1.3.7]), само же пересечение вычислять только в том случае, когда эти условия выполнены.

Замечание 2. Для запоминания результатов вычисления пересекаемости отрезков можно определить дополнительную структуру данных D, в которой по двум заданным индексам отрезков i и j доступно логическое значение, указывающее на пересекаемость отрезков: $D(i, j) \in \{истина, ложь, неизвестно\}$. При этом операции доступа к элементу $D(i, j)$ и его редактирования должны

выполняться за $O(1)$. Однако для хранения данных в такой структуре необходимо $O(m^2)$ памяти.

Пример 1.10. Найти все пересечения отрезков A_iB_i, $i=\overline{1,4}$:

$$\begin{aligned} &A_1(7,0), \quad &&B_1(10,1),\\ &A_2(0,7), \quad &&B_2(7,0),\\ &A_3(6,0), \quad &&B_3(10,6),\\ &A_4(2,7), \quad &&B_4(8,10). \end{aligned}$$

Перед первой итерацией составим структуру P. Следует обратить внимание, что начало одного из отрезков совпадает с концом другого: $A_1 = B_2$. В этом случае начало имеет больший приоритет по сравнению с концом:

$$P := \{A_2, A_4, A_3, A_1, B_2, B_4, B_1, B_3\}.$$

Далее следует обработка P:
1-я итерация: извлекаем точку начала A_2, помещаем второй отрезок в S:

$$P := \{\cancel{A_2}, A_4, A_3, A_1, B_2, B_4, B_1, B_3\},$$
$$S := \{2\}.$$

2-я итерация: извлекаем точку начала A_4. Определяем, как разместить 4-й отрезок в S:

$$\overrightarrow{A_2B_2}(7,-7),$$
$$\overrightarrow{A_2A_4}(2,0),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2}, \overrightarrow{A_2A_4}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -7 \\ 2 & 0 \end{vmatrix} = +1,$$

значит, 4-й отрезок размещается после 2-го:

$$P := \{\cancel{A_4}, A_3, A_1, B_2, B_4, B_1, B_3\},$$
$$S := \{2, \boxed{4}\}.$$

Выясним, пересекаются ли два отрезка. Для этого, уже зная, что A_4 лежит по правую сторону от оси A_2B_2, вычислим положение B_4 относительно неё же:

$$\overrightarrow{A_2B_4}(8,3),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2},\overrightarrow{A_2A_4}\right)=\operatorname{sgn}\begin{vmatrix}7 & -7\\ 8 & 3\end{vmatrix}=+1.$$

Это означает, что раз конец четвёртого отрезка, как и его начало, расположен по правую сторону от A_2B_2, то два отрезка не пересекаются.

3-я итерация: извлекаем точку начала A_3. Определяем положение третьего отрезка в структуре *S*:

$$\overrightarrow{A_2A_3}(6,-7),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2},\overrightarrow{A_2A_3}\right)=\operatorname{sgn}\begin{vmatrix}7 & -7\\ 6 & -7\end{vmatrix}=-1,$$

значит, третий отрезок нужно поместить в *S* перед вторым:

$$P:=\{\not{A}_3,A_1,B_2,B_4,B_1,B_3\},$$

$$S:=\{\boxed{3},2,4\}.$$

Пересекаемость отрезков 2 и 3:

$$\overrightarrow{A_2B_3}(10,-1),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2},\overrightarrow{A_2B_3}\right)=\operatorname{sgn}\begin{vmatrix}7 & -7\\ 10 & -1\end{vmatrix}=+1,$$

т.е. концы отрезка A_3B_3 лежат по разные стороны от прямой A_2B_2. Значит, нужно ещё проверить расположение концов A_2B_2 относительно прямой A_3B_3:

$$\overrightarrow{A_3B_3}(4,6),$$

$$\overrightarrow{A_3A_2}=-\overrightarrow{A_2A_3},$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3B_3},\overrightarrow{A_3A_2}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ -6 & 7\end{vmatrix}=+1,$$

$$\overrightarrow{A_3B_2}(1,0),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3B_3},\overrightarrow{A_3B_2}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ 1 & 0\end{vmatrix}=-1.$$

Таким образом, два конца A_2 и B_2 тоже лежат по разные стороны от прямой A_3B_3, что означает пересекаемость двух отрезков. Вычислим это пересечение:

$$\begin{gathered} A_2B_2 : x + y - 7 = 0, \\ A_3B_3 : 3x - 2y - 18 = 0, \\ \begin{cases} x + y - 7 = 0, \\ 3x - 2y - 18 = 0 \end{cases} \Rightarrow x = \frac{32}{5}, y = \frac{3}{5}. \end{gathered} \tag{2.7}$$

Получили точку $C_{23}\left(\frac{32}{5}, \frac{3}{5}\right) = A_2B_2 \cap A_3B_3$, которая с учётом её положения относительно точек из P (располагается левее их всех) размещается в этой очереди следующим образом:

$$P := \{\boxed{C_{23}}, A_1, B_2, B_4, B_1, B_3\}.$$

4-я итерация: извлекается только что добавленная точка пересечения C_{23}, значит, в S нужно найти и поменять местами 2-й и 3-й отрезки:

$$\begin{gathered} P := \{\cancel{C_{23}}, A_1, B_2, B_4, B_1, B_3\}, \\ S := \{\underline{2,3}, 4\}. \end{gathered}$$

Третий и четвёртый отрезки оказались соседними, значит, необходимо выяснить, пересекаются ли они. Это легко сделать, зная исходя из предыдущих вычислений (2.7) уравнение прямой A_3B_3 и подставив в его левую часть координаты A_4 и B_4:

$$\begin{gathered} (3x - 2y - 18)\big|_{\substack{x=2 \\ y=7}} = 6 - 14 - 18 < 0, \\ (3x - 2y - 18)\big|_{\substack{x=8 \\ y=10}} = 24 - 20 - 18 < 0. \end{gathered}$$

Значит, два отрезка не пересекаются.

5-я итерация: из P извлекается точка начала A_1:

$$P := \{\cancel{A_1}, B_2, B_4, B_1, B_3\}.$$

Вычисляем положение отрезка A_1B_1 относительно отрезков из S, начиная с середины этой структуры, т.е. третьего отрезка:

$$\overrightarrow{A_3A_1}(1,0),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_3B_3},\overrightarrow{A_3A_1}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ 1 & 0\end{vmatrix}=-1.$$

Значит, начало A_1 расположено ниже A_3B_3, и далее надо рассматривать отрезок 2:

$$\overrightarrow{A_2A_1}(7,-7),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2},\overrightarrow{A_2A_1}\right)=\operatorname{sgn}\begin{vmatrix}7 & -7\\ 7 & -7\end{vmatrix}=0.$$

Точка A_1 оказалась лежащей на отрезке A_2B_2! В этом случае нужно проверить положение относительно этого отрезка конца B_1. Если он окажется **выше/ниже** отрезка A_2B_2, то отрезок 1 необходимо будет разместить **ниже/выше** в структуре S, ведь потом при обработке пересечения (коим является A_1) два отрезка поменяются местами.

$$\overrightarrow{A_2B_1}(10,-6),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_2B_2},\overrightarrow{A_2A_1}\right)=\operatorname{sgn}\begin{vmatrix}7 & -7\\ 10 & -6\end{vmatrix}=+1.$$

Таким образом, получаем структуру

$$S:=\{\boxed{1},2,3,4\}.$$

Пересечение $A_1B_1\cap A_2B_2=A_1$ обозначим через $C_{12}(7,0)$ и запишем в P. Его нужно записать перед точкой конца $B_2(7,0)$, имеющей равную абсциссу:

$$P:=\{\boxed{C_{12}},B_2,B_4,B_1,B_3\}.$$

6-я итерация: извлекаем точку C_{12}, при этом в S отрезки 1 и 2 меняются местами:

$$P:=\{\cancel{C_{12}},B_2,B_4,B_1,B_3\},$$
$$S:=\{\underline{2,1},3,4\}.$$

Отрезки 1 и 3 стали соседними, значит, нужно выяснить их пересекаемость. Опять же, можно использовать уравнение прямой A_3B_3:

$$(3x-2y-18)\big|_{\substack{x=7\\y=0}} = 21-18>0,$$
$$(3x-2y-18)\big|_{\substack{x=10\\y=1}} = 30-2-18>0,$$

что означает, что два отрезка не пересекаются.

Все дальнейшие итерации тривиальны: из P извлекаем точки концов разных отрезков, что приводит к их удалению из S. При этом каждый раз будут удаляться крайние отрезки, а значит, новых пар соседей, для которых необходимо вычислять пересечения, не возникает:

$$P := \{\cancel{B_2}, B_4, B_1, B_3\}, S := \{\cancel{2}, 1, 3, 4\},$$
$$P := \{\cancel{B_4}, B_1, B_3\}, S := \{1, 3, \cancel{4}\},$$
$$P := \{\cancel{B_1}, B_3\}, S := \{\cancel{1}, 3\},$$
$$P := \varnothing, S := \varnothing.$$

Таким образом, у четырёх отрезков всего два пересечения: $C_{23}\left(\frac{32}{5}, \frac{3}{5}\right)$ и $C_{12}(7,0) = A_1 = B_2$. Шаги алгоритма Бентли-Оттманна для данного примера представлены на рисунке 2.19.

2.1.6. Алгоритм Вейлера-Азертона

Алгоритм Вейлера-Азертона (стр. 43) отсечения произвольного полигона отсекателем в виде невыпуклого полигона подразумевает вычисление пересечений рёбер полигона с рёбрами отсекателя, а также последующий обход вершин полигона, отсекателя и полученных пересечений для составления искомых полигонов, получаемых в результате отсечения. При реализации этого алгоритма понадобится несколько видоизменённая реализация алгоритма Бентли-Оттманна для нахождения всех нужных пересечений. Также для хранения данных понадобятся специальные структуры. Всё изложенное ниже справедливо в случае отсутствия особых случаев, описанных в замечаниях к обоим алгоритмам.

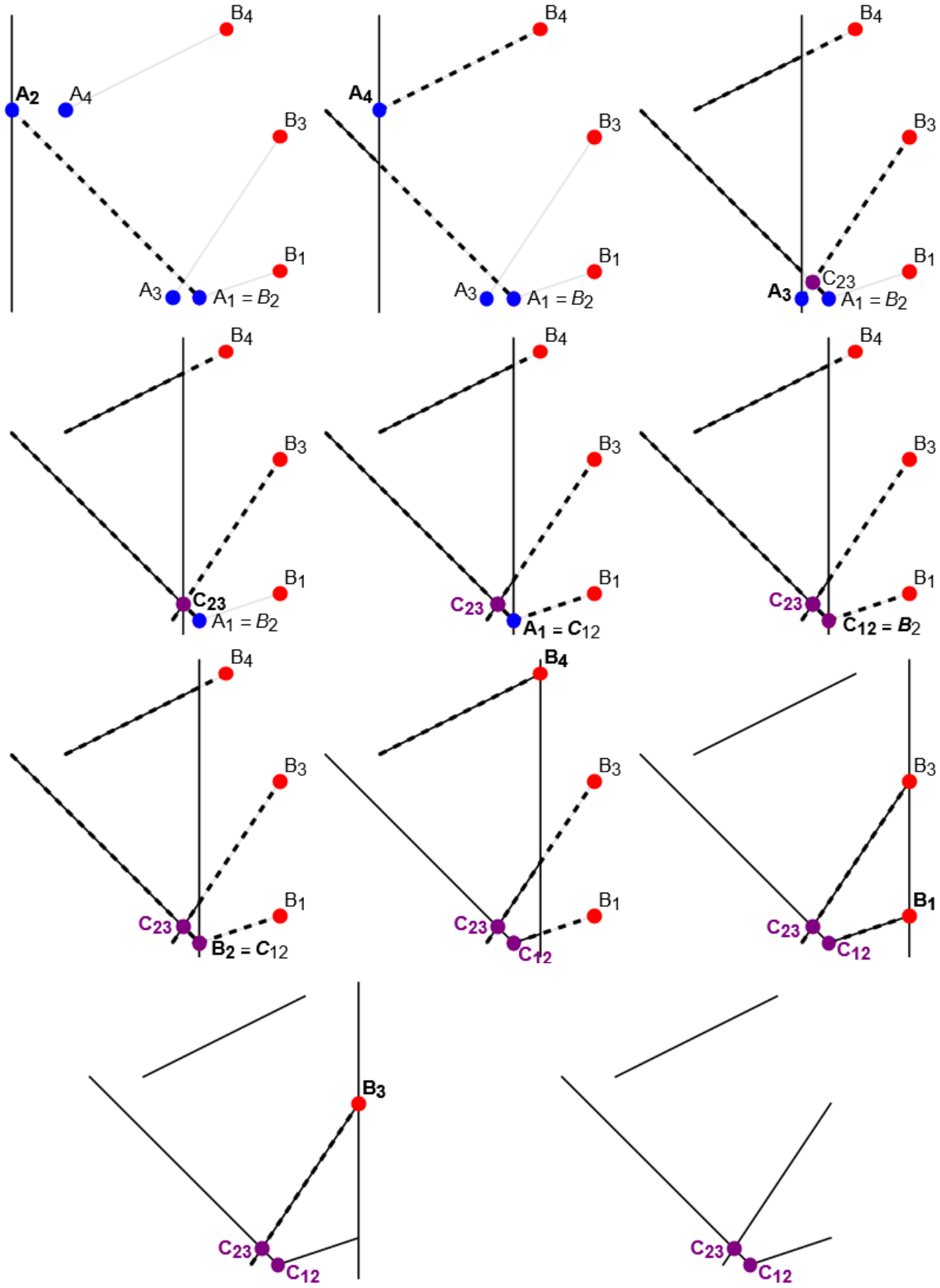


Рисунок 2.19 – Пример работы алгоритма Бентли-Оттманна при наличии смежных отрезков

Пример 1.11. Осуществить отсечение полигона $A_1A_2A_3A_1$ нерегулярным отсекателем $C_1C_2C_3C_4C_1$, представляющим собой невыпуклый полигон:

$$A_1(1,9), A_2(10,9), A_3(8,2),$$
$$C_1(0,1), C_2(4,7), C_3(3,3), C_4(8,5).$$

Вначале инициализируем список вершин полигона и отсекателя:

$$\mathcal{A} := \{A_1, A_2, A_3\},$$
$$\mathcal{C} := \{C_1, C_2, C_3, C_4\}.$$

В эти списки будут вставлены точки пересечений рёбер полигона и отсекателя, которые, как известно, можно найти при помощи алгоритма Бентли-Оттманна. С учётом того, что нужно искать пересечения не всех отрезков, а только тех, что относятся к разным полигонам, будем использовать следующую модификацию этого алгоритма:

- Вместо точек начал и концов отрезков будем рассматривать три категории: вершины полигонов, которые являются началами для обоих инцидентных им рёбер («начальные»); вершины, которые являются концами для обоих рёбер («конечные»); и вершины, которые для одного из ребра являются конечным, для другого – начальным («промежуточные»). Так, в данном примере вершины A_1, C_1 и C_3 являются «начальными», вершина A_3 – «промежуточная», а A_2, C_2 и C_4 – «конечные» (Рисунок 2.20).

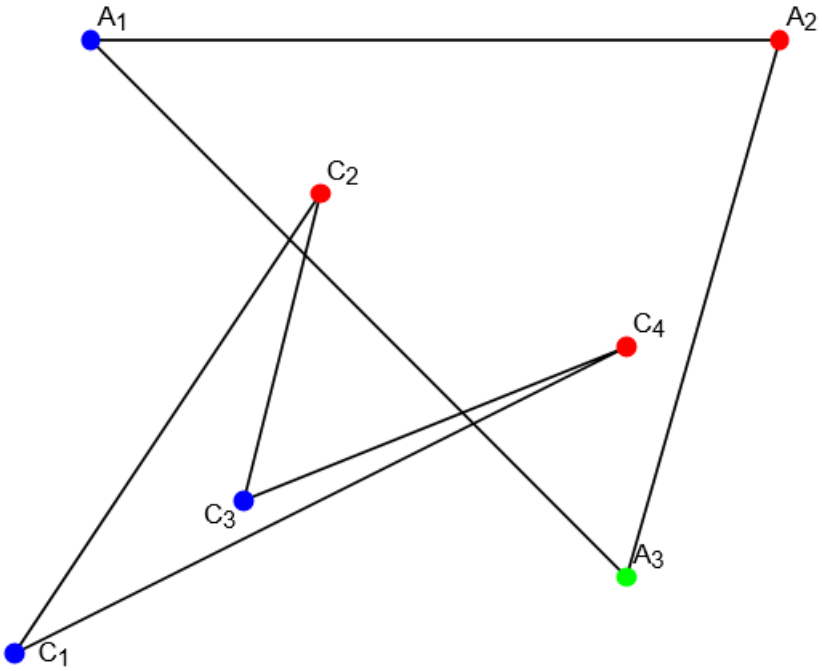


Рисунок 2.20 – Классификация вершин в модификации алгоритма Бентли-Оттманна: синим отмечены «начальные» вершины, зелёным – «промежуточные», красным – «конечные»

- Приоритетная очередь P работает так же, как и в классической реализации, т.е. все точки извлекаются из неё в порядке возрастания абсциссы, а новые пересечения добавляются с учётом этой же сортировки.

- При извлечении из P «начальной» вершины в структуру S нужно добавлять два смежных ребра в заданном порядке (например, начиная с нижнего отрезка). После этого между отрезками, ставшими соседями в S, пересечения вычисляются только в случае, когда эти отрезки являются рёбрами разных полигонов.

- При извлечении из P «промежуточной» вершины необходимо в S заменить отрезок, для которого текущая вершина является концом, на отрезок, для которого эта же вершина является началом, после чего при необходимости вычислить новые пересечения.

- При извлечении из P «конечной» вершины два ребра, инцидентные этой вершине, подлежат удалению из S. Пересечения вычисляются точно так же и при тех же условиях, что описаны выше.

- При извлечении из P точки пересечения, как и в классической реализации, два пересекающихся отрезка меняются местами в S, после чего при необходимости вычисляются новые пересечения.

Применим эту реализацию для вычисления пересечений рёбер двух заданных полигонов. Очередь P в начале алгоритма Бентли-Оттманна выглядит следующим образом:

$$P := \{C_1, A_1, C_3, C_2, A_3, C_4, A_2\}.$$

1-я итерация: извлекаем из P вершину C_1 – начало рёбер отсекателя C_1C_4 и C_1C_2 (здесь и далее при обозначении ребра вначале будем указывать левую вершину, а затем – правую). Из них ребро C_1C_2 расположено выше:

$$\overrightarrow{C_1C_4}(8,4),$$
$$\overrightarrow{C_1C_2}(4,6),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_4}, \overrightarrow{C_1C_2}\right) = \operatorname{sgn}\begin{vmatrix} 8 & 4 \\ 4 & 6 \end{vmatrix} = +1.$$

Следовательно, в структуру S добавляются эти два ребра в следующем порядке:

$$P := \{\cancel{C_1}, A_1, C_3, C_2, A_3, C_4, A_2\},$$
$$S := \{C_1C_4, C_1C_2\}.$$

2-я итерация: из P извлекаем A_1 – «начальную» вершину исходного полигона. Ей инцидентны рёбра A_1A_2 и A_1A_3. Выясним их взаимное расположение:

$$\overrightarrow{A_1A_2}(9,0),$$
$$\overrightarrow{A_1A_3}(7,-7),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2},\overrightarrow{A_1A_3}\right)=\operatorname{sgn}\begin{vmatrix}9 & 0\\ 7 & -7\end{vmatrix}=-1.$$

Значит, ребро A_1A_3 будет следовать в S перед A_1A_2, причём они обязательно будут соседями. Ещё нужно выяснить расположение точки A_1 относительно отрезков, уже записанных в S:

$$\overrightarrow{C_1A_1}(1,8),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_4},\overrightarrow{C_1A_1}\right)=\operatorname{sgn}\begin{vmatrix}8 & 4\\ 1 & 8\end{vmatrix}=+1, \qquad (2.8)$$

т.е. точка A_1 расположена выше, и нужно рассмотреть также отрезок C_1C_2:

$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_2},\overrightarrow{C_1A_1}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ 1 & 8\end{vmatrix}=+1.$$

Таким образом, в S новые отрезки будут добавлены следующим образом:

$$S:=\{C_1C_4,C_1C_2,\boxed{A_1A_3,A_1A_2}\}.$$

Теперь нужно выяснить, пересекаются ли соседние отрезки C_1C_2 и A_1A_3. Вначале определим наличие их пересечения, для чего рассмотрим положение точки A_3 относительно отрезка C_1C_2:

$$\overrightarrow{C_1A_3}(8,1),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_2},\overrightarrow{C_1A_3}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ 8 & 1\end{vmatrix}=-1.$$

Учитывая (2.8), получаем, что точки A_1 и A_3 расположены по разные стороны от прямой C_1C_2. Точно так же приходим к выводу, что C_1 и C_2 расположены по разные стороны от A_1A_3:

$$\overrightarrow{A_1C_1} = -\overrightarrow{C_1A_1}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_3}, \overrightarrow{A_1C_1}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -7 \\ -1 & -8 \end{vmatrix} = -1,$$

$$\overrightarrow{A_1C_2}(3,-2),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_3}, \overrightarrow{A_1C_2}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -7 \\ 3 & -2 \end{vmatrix} = +1.$$

Теперь вычислим пересечение двух отрезков:

$$\begin{gathered} A_1A_3 : x + y - 10 = 0, \\ C_1C_2 : 3x - 2y + 2 = 0, \\ \begin{cases} x + y - 10 = 0, \\ 3x - 2y + 2 = 0 \end{cases} \Rightarrow x = \frac{18}{5}, y = \frac{32}{5}. \end{gathered} \tag{2.9}$$

Таким образом, получили точку $A_1A_3 \cap C_1C_2 = B_{31}\left(\frac{18}{5}, \frac{32}{5}\right)$, которая подлежит размещению в P:

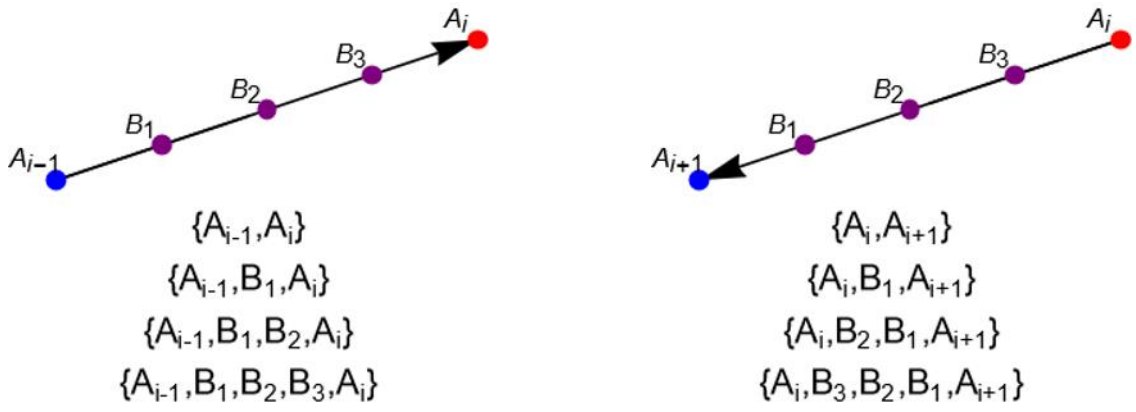
$$P := \{\not{A}_1, C_3, \boxed{B_{31}}, C_2, A_3, C_4, A_2\}.$$

Кроме того, найденная точка пересечения подлежит размещению в списках $\mathcal{A}$ и $\mathcal{C}$ по следующему правилу: если ребро полигона направлено **слева направо**, то точка пересечения добавляется в соответствующий список **перед** точкой правого конца этого ребра (Рисунок 2.21а), иначе – **после** точки этого же конца (Рисунок 2.21б). На рисунке 2.21 также указан порядок добавления точек B_1, B_2, B_3 по мере их нахождения по алгоритму Бентли-Оттманна.

Так, в данном примере ребро A_1A_3 при заданном направлении обхода полигона $A_1A_2A_3A_1$ направлено влево, поэтому точка B_{31} вставляется в $\mathcal{A}$ после A_3, а ребро C_1C_2 направлено вправо, значит, в $\mathcal{C}$ эта же точка вставляется перед C_2:

$$\mathcal{A} := \{A_1, A_2, A_3, \boxed{B_{31}}\},$$

$$\mathcal{C} := \{C_1, \boxed{B_{31}}, C_2, C_3, C_4\}.$$



а б

Рисунок 2.21 – Варианты размещения точек пересечений в списках вершин полигонов: а – между концами ребрами, направленного вправо, б – для ребра, направленного влево

3-я итерация: из P извлекаем точку C_3 – это ещё одна «начальная» вершина полигона, в которой сходятся смежные рёбра отсекателя C_3C_2 и C_3C_4. Выясним их взаимное расположение:

$$\overrightarrow{C_3C_2}(1,4),$$
$$\overrightarrow{C_3C_4}(5,2),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{C_3C_2},\overrightarrow{C_3C_4}\right)=\operatorname{sgn}\begin{vmatrix}1 & 4\\ 5 & 2\end{vmatrix}=-1.$$

Значит, C_3C_4 будет идти перед C_3C_2. Теперь найдём позицию в S для размещения этих отрезков, для чего вычислим позицию C_3 относительно среднего отрезка из S, т.е. C_1C_2 :

$$\overrightarrow{C_1C_3}(3,2),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_2},\overrightarrow{C_1C_3}\right)=\operatorname{sgn}\begin{vmatrix}4 & 6\\ 3 & 2\end{vmatrix}=-1.$$

Значит, точка C_3 расположена ниже C_1C_2, и осталось рассмотреть её положение относительно первого отрезка в S:

$$\operatorname{sgn}\angle\left(\overrightarrow{C_1C_4},\overrightarrow{C_1C_3}\right)=\operatorname{sgn}\begin{vmatrix}8 & 4\\ 3 & 2\end{vmatrix}=+1.$$

Таким образом, нашли позицию в S для добавления новых отрезков:

$$S := \{C_1C_4, \boxed{C_3C_4, C_3C_2}, C_1C_2, A_1A_3, A_1A_2\}.$$

Отметим, что в обеих парах новых соседей оба отрезка являются рёбрами отсекателя, поэтому на данной итерации не нужно вычислять никакие пересечения. Структура P примет следующий вид:

$$P := \{\cancel{C_3}, B_{31}, C_2, A_3, C_4, A_2\}.$$

4-я итерация: извлекается точка пересечения B_{31}. Соответственно, в S меняются местами два отрезка, пересечением которых эта точка является:

$$S := \{C_1C_4, C_3C_4, C_3C_2, \underline{A_1A_3, C_1C_2}, A_1A_2\}.$$

Здесь соседями стали две пары рёбер, принадлежащих двум разным полигонам: C_3C_2 и A_1A_3, C_1C_2 и A_1A_2. Для первой пары отрезков воспользуемся уравнением прямой A_1A_3, полученным ранее в (2.9):

$$(x + y - 10)\Big|_{\substack{x=3\\y=3}} = 3 + 3 - 10 < 0,$$

$$(x + y - 10)\Big|_{\substack{x=4\\y=7}} = 4 + 7 - 10 > 0,$$

$$C_3C_2 : 4x - y - 9 = 0,$$

$$\begin{cases} x + y - 10 = 0, \\ 4x - y - 9 = 0 \end{cases} \Rightarrow x = \frac{19}{5}, y = \frac{31}{5}.$$

Таким образом, имеем новую точку пересечения $A_1A_3 \cap C_3C_2 = B_{32}\left(\frac{19}{5}, \frac{31}{5}\right)$. Два других отрезка, несложно видеть, не пересекаются, так как ординаты обоих концов отрезка C_1C_2 меньше, чем у A_1A_2. Значит, на данной итерации в P добавляется одна точка:

$$P := \{\cancel{B_{31}}, \boxed{B_{32}}, C_2, A_3, C_4, A_2\}.$$

Эта же точка добавляется в списки вершин обоих полигонов:

$$\mathcal{A} := \{A_1, A_2, A_3, \boxed{B_{32}}, B_{31}\},$$

$$\mathcal{C} := \{C_1, B_{31}, C_2, \boxed{B_{32}}, C_3, C_4\}.$$

5-я итерация: из P извлекается точка пересечения B_{32}:

$$P := \{\cancel{B_{32}}, C_2, A_3, C_4, A_2\},$$
$$S := \{C_1C_4, C_3C_4, \underline{A_1A_3}, \underline{C_3C_2}, C_1C_2, A_1A_2\}.$$

Проверка пересекаемости отрезков C_3C_4 и A_1A_3:

$$(x+y-10)\big|_{\substack{x=8\\y=5}} = 8+5-10 > 0,$$
$$(x+y-10)\big|_{\substack{x=3\\y=3}} = 3+3-10 < 0,$$
$$C_3C_4 : 2x-5y+9=0,$$
$$\begin{cases} x+y-10=0, \\ 2x-5y+9=0 \end{cases} \Rightarrow x=\frac{41}{7}, y=\frac{29}{7}.$$

Значит, ещё одной точкой пересечения является $A_1A_3 \cap C_3C_4 = B_{33}\left(\frac{41}{7}, \frac{29}{7}\right)$.

Добавим её в очередь P, а также в списки $\mathcal{A}$ и $\mathcal{C}$:

$$P := \{C_2, \boxed{B_{33}}, A_3, C_4, A_2\},$$
$$\mathcal{A} := \{A_1, A_2, A_3, \boxed{B_{33}}, B_{32}, B_{31}\},$$
$$\mathcal{C} := \{C_1, B_{31}, C_2, B_{32}, C_3, \boxed{B_{33}}, C_4\}.$$

6-я итерация: из P извлекаем точку C_2 – «конечную» вершину отсекателя. В структуре S необходимо найти и удалить оба ребра с концом C_2:

$$P := \{\cancel{C_2}, B_{33}, A_3, C_4, A_2\},$$
$$S := \{C_1C_4, C_3C_4, A_1A_3, \cancel{C_3C_2}, \cancel{C_1C_2}, A_1A_2\}.$$

При этом стали соседними отрезки A_1A_3 и A_1A_2, но они являются рёбрами одного полигона, значит, никакие пересечения не вычисляются.

7-я итерация: извлекается точка пересечения B_{33}:

$$P := \{\cancel{B_{33}}, A_3, C_4, A_2\},$$
$$S := \{C_1C_4, \underline{A_1A_3}, \underline{C_3C_4}, A_1A_2\}.$$

Отрезки C_1C_4 и A_1A_3 пересекаются, что следует из следующих вычислений:

$$(x+y-10)\big|_{\substack{x=0\\y=1}} = 0+1-10<0,$$

$$(x+y-10)\big|_{\substack{x=8\\y=5}} = 8+5-10>0,$$

$$C_1C_4 : x-2y+2=0,$$

$$\begin{cases} x+y-10=0, \\ x-2y+2=0 \end{cases} \Rightarrow x=6, y=4.$$

Их точка пересечения $B_{34}(6,4)$ добавляется в очередь *P*, списки $\mathcal{A}$ и $\mathcal{C}$:

$$P := \{\boxed{B_{34}}, A_3, C_4, A_2\},$$

$$\mathcal{A} := \{A_1, A_2, A_3, \boxed{B_{34}}, B_{33}, B_{32}, B_{31}\},$$

$$\mathcal{C} := \{C_1, B_{31}, C_2, B_{32}, C_3, B_{33}, C_4, \boxed{B_{34}}\}.$$

Два других отрезка, C_3C_4 и A_1A_2, как нетрудно видеть, общих точек не имеют, так как обе вершины C_3 и C_4 расположены ниже вершин A_1 и A_2.

8-я итерация: из *P* извлекается точка пересечения B_{34} :

$$P := \{\cancel{B_{34}}, A_3, C_4, A_2\},$$

$$S := \{\underline{A_1A_3, C_1C_4}, C_3C_4, A_1A_2\}.$$

Как видим из полученной структуры *S*, новые пересечения на данной итерации не вычисляются.

9-я итерация: из *P* извлекается «промежуточная» вершина A_3. Значит, в *S* нужно найти отрезок с концом A_3 и заменить его на другое ребро:

$$P := \{\cancel{A_3}, C_4, A_2\},$$

$$S := \{\cancel{A_1A_3}, C_1C_4, C_3C_4, A_1A_2\},$$

$$S := \{\boxed{A_3A_2}, C_1C_4, C_3C_4, A_1A_2\}.$$

Значит, необходимо определить, пересекаются ли отрезки A_3A_2 и C_1C_4. Однако исходя из сравнения координат их концов несложно видеть, что пересечения они не имеют.

Наконец, на двух последних итерациях происходит извлечение «конечных» вершин, что приводит к удалению соответствующих отрезков из S:

$$P := \{\cancel{C_4}, A_2\}, S := \{A_3A_2, \cancel{C_1C_4}, \cancel{C_3C_4}, A_1A_2\},$$
$$P := \varnothing, S := \varnothing.$$

Итак, найдены все пересечения рёбер двух полигонов, которые записаны в список их вершин в правильном порядке:

$$\mathcal{A} = \{A_1, A_2, A_3, B_{34}, B_{33}, B_{32}, B_{31}\},$$
$$\mathcal{C} = \{C_1, B_{31}, C_2, B_{32}, C_3, B_{33}, C_4, B_{34}\}.$$

Теперь нужно определить положение вершин исходного полигона относительно отсекателя и наоборот для того, чтобы определить результаты отсечения. Для этого достаточно определить положение вершины A_1 относительно отсекателя $C_1C_2C_3C_4C_1$ и вершины C_1 относительно полигона $A_1A_2A_3A_1$. Вычисляется это, например, через октантный критерий, описанный в первой части ЭУМК [1, с. 66–70 и разд. 2.3.3]:

$$\overrightarrow{A_1C_1}(-1,-8),\ I_1 = 6,$$
$$\overrightarrow{A_1C_2}(3,-2),\ I_2 = 8,$$
$$\overrightarrow{A_1C_3}(2,-6),\ I_3 = 7,$$
$$\overrightarrow{A_1C_4}(7,-4),\ I_4 = 8,$$
$$\Delta_2 = I_2 - I_1 = 2, \Delta_3 = I_3 - I_2 = -1, \Delta_4 = I_4 - I_3 = 1, \Delta_1 = I_1 - I_4 = -2.$$

Ни одна из этих разностей не подлежит корректировке, их сумма даёт ноль, значит, точка A_1 лежит снаружи отсекателя. Аналогично с точкой C_1 и полигоном $A_1A_2A_3A_1$:

$$\overrightarrow{C_1A_1}(1,8),\quad I_1 = 2,$$
$$\overrightarrow{C_1A_2}(10,8),\ I_2 = 1,$$
$$\overrightarrow{C_1A_3}(8,1),\quad I_3 = 1,$$
$$\Delta_2 = I_2 - I_1 = -1, \Delta_3 = I_3 - I_2 = 0, \Delta_1 = I_1 - I_3 = 1.$$

Точно так делаем вывод, что точка C_1 лежит снаружи исходного полигона.

Зная расположение относительно отсекателя хотя бы одной из вершин полигона, можно определить расположение всех остальных точек списка $\mathcal{A}$.

Так, при обходе по периметру полигона $A_1A_2A_3A_1$, начиная от A_1, движемся снаружи отсекателя до тех пор, пока не достигнем одну из найденных ранее точек пересечения с границей отсекателя. В данном примере получаем, что вершины A_1, A_2, A_3 являются **внешними**. Далее достигаем точку пересечения B_{34} – она является **точкой входа** внутрь отсекателя. Все вершины исходного полигона между этой и следующей точкой пересечения являются уже **внутренними**. Однако после B_{34} сразу же следует точка пересечения B_{33}, которая является **точкой выхода** из отсекателя. Аналогично определяется, что B_{32} – это точка входа, B_{31} – точка выхода. Итог такого обхода схематично можно представить, выделив внутренние участки контура исходного полигона в списке $\mathcal{A}$:

$$\mathcal{A} = \{A_1, A_2, A_3, \boxed{B_{34}, B_{33}}, \boxed{B_{32}, B_{31}}\}.$$

Аналогично отмечаем части границы отсекателя в списке $\mathcal{C}$, попавшие внутрь исходного полигона:

$$\mathcal{C} = \{C_1, \boxed{B_{31}, C_2, B_{32}}, C_3, \boxed{B_{33}, C_4, B_{34}}\}.$$

Соединяя затем отмеченные участки и замыкая эти ломаные, окончательно получаем два полигона: $B_{34}B_{33}C_4B_{34}$ и $B_{32}B_{31}C_2B_{32}$ (Рисунок 2.22).

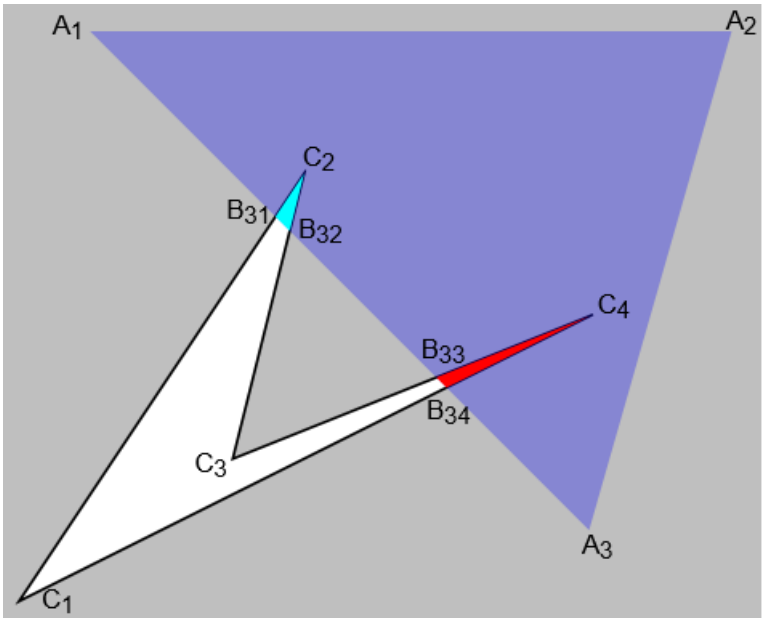


Рисунок 2.22 – Результат отсечения невыпуклым отсекателем

2.1.7. Удаление невидимых рёбер выпуклого полигона на плоскости

В случае, когда наблюдатель находится на той же плоскости, что и наблюдаемые объекты, используются особые алгоритмы для определения их видимости, которые основаны на определении взаимного положения наблюдателя и рассматриваемого объекта. Некоторые из таких алгоритмов (стр. 50 и стр. 51) позволяют определить часть границы выпуклого полигона, видимую для заданного дальнего и ближнего наблюдателя.

Пример 1.12. Определить часть границы выпуклого полигона $A_1A_2A_3A_4A_5A_6A_1$, видимую для дальнего наблюдателя, направление на которого характеризуется вектором $\vec{s}(1,-5)$:

$$A_1(8,1),\ A_2(10,5),\ A_3(9,10),\ A_4(0,10),\ A_5(0,9),\ A_6(1,2).$$

Ко всему полигону применим линейное преобразование с матрицей $\begin{pmatrix} s_x & s_y \\ -s_y & s_x \end{pmatrix} = \begin{pmatrix} 1 & -5 \\ 5 & 1 \end{pmatrix}$:

$$\begin{pmatrix} s_x & s_y \\ -s_y & s_x \end{pmatrix}(A_1,A_2,A_3,A_4,A_5,A_6) = \begin{pmatrix} 1 & -5 \\ 5 & 1 \end{pmatrix}\begin{pmatrix} 8 & 10 & 9 & 0 & 0 & 1 \\ 1 & 5 & 10 & 10 & 9 & 2 \end{pmatrix} =$$

$$= \begin{pmatrix} 3 & -15 & -41 & -50 & -45 & -9 \\ 41 & 55 & 55 & 10 & 9 & 7 \end{pmatrix}.$$

Столбцы полученной матрицы являются координатными столбцами точек A_i', полученных в результате такого поворота (и масштабирования), что направление на наблюдателя стало сонаправленным вектору с координатами (1, 0). Значит, крайними видимыми вершинами полученного полигона являются вершины с наименьшей и наибольшей ординатой, т.е. $A_6'(-9,7)$ и какая-то одна из вершин $A_2'(-15,55)$ либо $A_3'(-41,55)$. Из них выбирается вершина A_2', так как она расположена правее, т.е. ближе к наблюдателю.

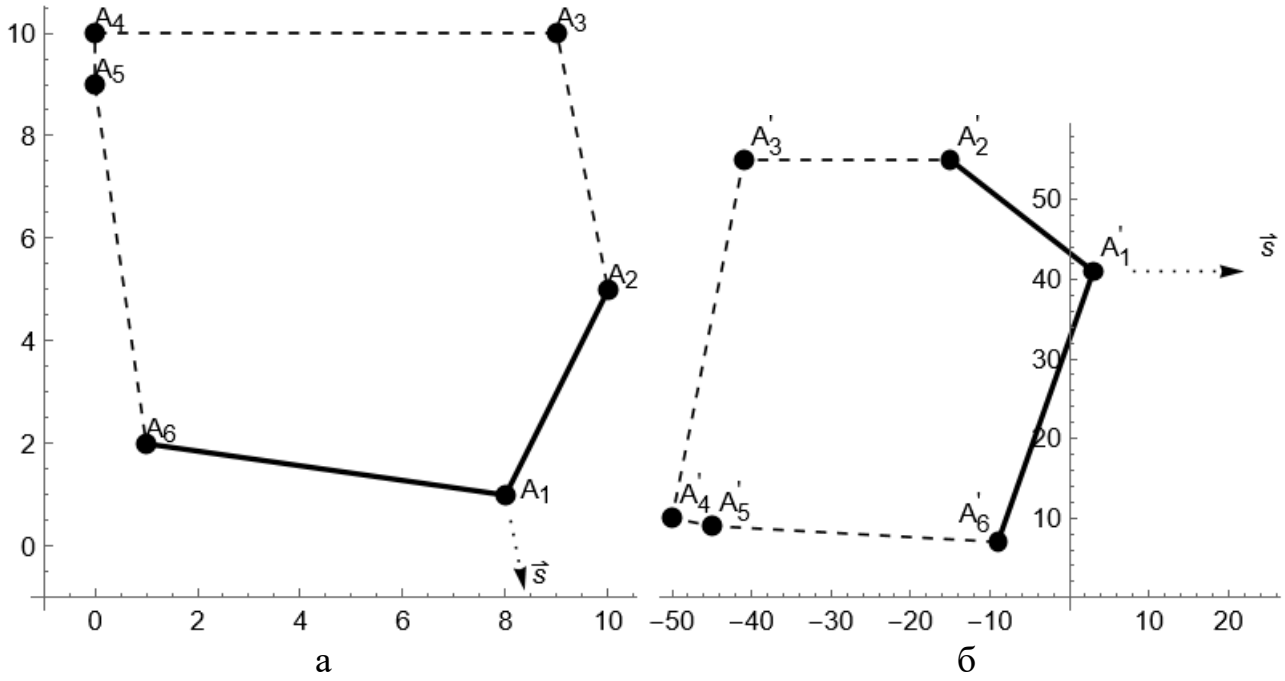
Две крайние вершины A_6' и A_2' разбивают границу полигона $A_1'A_2'A_3'A_4'A_5'A_6'A_1'$ на два участка. Определить, который из них является видимым, можно одним из двух способов:

- Вычислить вершину с наибольшей абсциссой, которая заведомо является видимой. В данном примере таковой является точка $A_1'(3,41)$. Значит, участок, содержащий эту вершину, т.е. $A_6'A_1'A_2'$, является видимым для наблюдателя с направлением (1, 0).

- Вычислить направление обхода исходного полигона:

$$\overrightarrow{A_1A_2}(2,4),$$
$$\overrightarrow{A_2A_3}(-1,5),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2},\overrightarrow{A_2A_3}\right)=\operatorname{sgn}\begin{vmatrix}2 & 4\\ -1 & 5\end{vmatrix}=+1.$$

Значит, в обоих полигонах имеет место левосторонний обход, при котором для получения видимого участка границы нужно сделать обход от самой нижней вершины полигона $A_1'A_2'A_3'A_4'A_5'A_6'A_1'$ к самой верхней, и снова получим $A_6'A_1'A_2'$. На рисунке 2.23 изображены два полигона, видимые и невидимые участки их границ изображены соответственно сплошными и пунктирными линиями.



а б

Рисунок 2.23 – Исходный (а) и повёрнутый (б) полигоны

Пример 1.13. Определить часть границы выпуклого полигона $A_1A_2A_3A_4A_5A_6A_1$, видимую для ближнего наблюдателя, расположенного в точке $S(12, 13)$:

$$A_1(4,3),\ A_2(7,0),\ A_3(10,4),\ A_4(10,10),\ A_5(2,10),\ A_6(2,9).$$

Вначале необходимо определить позицию наблюдателя относительно полигона. Нетрудно видеть, что S лежит снаружи полигона, т.к. абсцисса этой точки больше абсцисс каждой из вершин полигона $A_1A_2A_3A_4A_5A_6A_1$. Это означает, что, как и в предыдущем примере, граница полигона делится на два участка: видимый и невидимый.

Проведём от наблюдателя ко всем вершинам векторы $\overrightarrow{SA_i}$, $i=\overline{1,6}$:

$\overrightarrow{SA_1}(-8,-10),\ \overrightarrow{SA_2}(-5,-13),\ \overrightarrow{SA_3}(-2,-9),\ \overrightarrow{SA_4}(-2,-3),\ \overrightarrow{SA_5}(-10,-3),\ \overrightarrow{SA_6}(-10,-4).$

Среди них найдём крайние векторы $\vec{l}$ и $\vec{r}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{SA_1},\overrightarrow{SA_2}\right)=\operatorname{sgn}\begin{vmatrix}-8 & -10\\ -5 & -13\end{vmatrix}=+1\Rightarrow \vec{l}:=\overrightarrow{SA_2},\vec{r}:=\overrightarrow{SA_1}.$$

Значит, с точки зрения наблюдателя S вершина A_2 расположена левее A_1. Отсюда следует, что для нахождения крайней левой вершины нужно двигаться от A_2 далее по направлению обхода исходного полигона до тех пор, пока не получим нарушение равенства $\operatorname{sgn}\angle\left(\vec{l},\overrightarrow{SA_i}\right)=+1$, где A_i – очередная вершина:

$$\begin{gathered}\operatorname{sgn}\angle\left(\vec{l},\overrightarrow{SA_3}\right)=\operatorname{sgn}\begin{vmatrix}-5 & -13\\ -2 & -9\end{vmatrix}=+1\Rightarrow \vec{l}:=\overrightarrow{SA_3},\\ \operatorname{sgn}\angle\left(\vec{l},\overrightarrow{SA_4}\right)=\operatorname{sgn}\begin{vmatrix}-2 & -9\\ -2 & -3\end{vmatrix}=-1.\end{gathered}\qquad(2.10)$$

Итак, самой левой вершиной является A_3, так как согласно вычислениям (2.10) она расположена левее обеих своих соседних вершин A_2 и A_4. Аналогично проводится поиск самой правой вершины: от текущей правой вершины A_1 осуществляется обход в противоположном направлении, который заканчивается при нарушении равенства $\operatorname{sgn}\angle\left(\vec{r},\overrightarrow{SA_i}\right)=-1$:

$$\begin{gathered}\operatorname{sgn}\angle\left(\vec{r},\overrightarrow{SA_6}\right)=\operatorname{sgn}\begin{vmatrix}-8 & -10\\ -10 & -4\end{vmatrix}=-1\Rightarrow \vec{r}:=\overrightarrow{SA_6},\\ \operatorname{sgn}\angle\left(\vec{r},\overrightarrow{SA_5}\right)=\operatorname{sgn}\begin{vmatrix}-10 & -4\\ -10 & -3\end{vmatrix}=-1\Rightarrow \vec{r}:=\overrightarrow{SA_5},\\ \operatorname{sgn}\angle\left(\vec{r},\overrightarrow{SA_4}\right)=\operatorname{sgn}\begin{vmatrix}-10 & -3\\ -2 & -3\end{vmatrix}=+1.\end{gathered}$$

Значит, самой правой вершиной является A_5. Для определения, которая из двух ломаных $A_3A_4A_5$ и $A_5A_6A_1A_2A_3$ является видимой, необходимо определить направление обхода исходного полигона:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2},\overrightarrow{A_2A_3}\right)=\operatorname{sgn}\begin{vmatrix}3 & 3\\ 3 & 4\end{vmatrix}=+1.$$

Для левостороннего полигона видимым является участок границы от самой левой до самой правой вершины. Таким образом, из точки S видимой является часть границы $A_3A_4A_5$ (Рисунок 2.24).

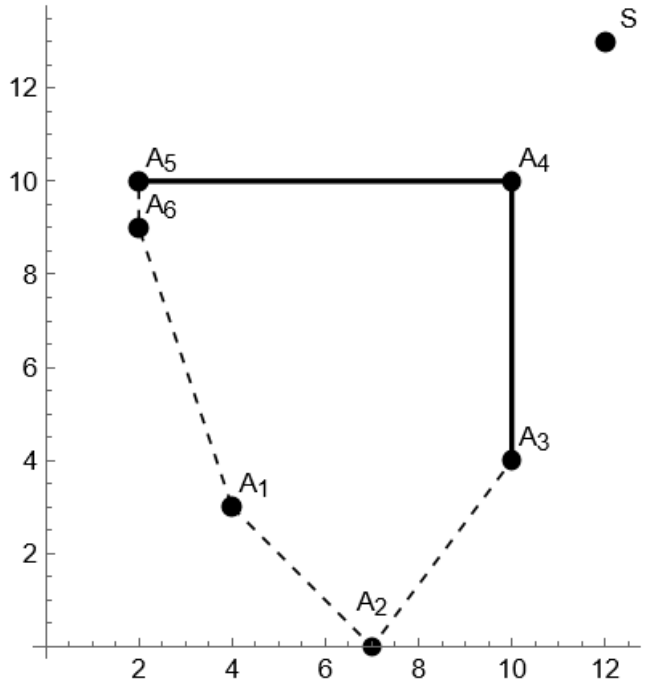


Рисунок 2.24 – Определение сторон, видимых для ближнего наблюдателя

2.1.8. Задачи

1. Осуществить на плоскости отсечение отрезка AB регулярным внешним окном, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$.

1.1. $A(2, -1)$, $B(5, 5)$, $x_{\min} = 0$, $x_{\max} = 5$, $y_{\min} = 0$, $y_{\max} = 3$
(перейти к ответу),

1.2. $A(-5, 10)$, $B(1, 1)$, $x_{\min} = 0$, $x_{\max} = 8$, $y_{\min} = 0$, $y_{\max} = 6$
(перейти к ответу),

1.3. $A(7, 4)$, $B(-1, 9)$, $x_{\min} = 0$, $x_{\max} = 5$, $y_{\min} = 0$, $y_{\max} = 7$
(перейти к ответу),

1.4. $A(-6, 1)$, $B(1, 12)$, $x_{\min} = 0$, $x_{\max} = 5$, $y_{\min} = 0$, $y_{\max} = 3$
(перейти к ответу),

1.5. $A(10, -1)$, $B(0, 2)$, $x_{\min} = 0$, $x_{\max} = 10$, $y_{\min} = 0$, $y_{\max} = 7$
(перейти к ответу),

1.6. $A(10, 2)$, $B(3, 15)$, $x_{\min} = 0$, $x_{\max} = 1$, $y_{\min} = 0$, $y_{\max} = 9$
(перейти к ответу),

1.7. $A(-14, 8)$, $B(8, -4)$, $x_{\min} = 0$, $x_{\max} = 8$, $y_{\min} = 0$, $y_{\max} = 3$
(перейти к ответу),

1.8. $A(-8, -2)$, $B(8, -2)$, $x_{\min} = 0$, $x_{\max} = 2$, $y_{\min} = 0$, $y_{\max} = 7$
(перейти к ответу),

1.9. $A(2, 11)$, $B(2, -1)$, $x_{\min} = 0$, $x_{\max} = 8$, $y_{\min} = 0$, $y_{\max} = 8$
(перейти к ответу),

1.10. $A(-14, -6)$, $B(8, 12)$, $x_{\min} = 0$, $x_{\max} = 9$, $y_{\min} = 0$, $y_{\max} = 10$
(перейти к ответу).

2. Осуществить на плоскости отсечение отрезка AB нерегулярным внешним отсекателем, представляющим собой выпуклый полигон $C_1C_2...C_nC_1$.

2.1. $A(-2, -1)$, $B(3, 1)$, $C_1(0,0)$, $C_2(1,0)$, $C_3(2,1)$, $C_4(1,2)$, $C_5(0,1)$
(перейти к ответу),

2.2. $A(-1, -2)$, $B(3, 3)$, $C_1(0,0)$, $C_2(1,0)$, $C_3(2,1)$, $C_4(1,2)$, $C_5(0,1)$
(перейти к ответу),

2.3. $A(1, 1)$, $B(2, -1)$, $C_1(0,0)$, $C_2(1,0)$, $C_3(2,1)$, $C_4(1,2)$, $C_5(0,1)$
(перейти к ответу),

2.4. $A(0, -1)$, $B(0, 5)$, $C_1(1,2)$, $C_2(-2,3)$, $C_3(-4,0)$, $C_4(-1,-2)$, $C_5(1,-1)$
(перейти к ответу),

2.5. $A(-6, 3)$, $B(0, -3)$, $C_1(1,2)$, $C_2(-2,3)$, $C_3(-4,0)$, $C_4(-1,-2)$, $C_5(1,-1)$
(перейти к ответу),

2.6. $A(-4, -3)$, $B(0, 1)$, $C_1(1,2)$, $C_2(-2,3)$, $C_3(-4,0)$, $C_4(-1,-2)$, $C_5(1,-1)$
(перейти к ответу),

2.7. $A(3, -2)$, $B(4, 2)$, $C_1(3,3)$, $C_2(2,4)$, $C_3(-3,3)$, $C_4(-4,-2)$, $C_5(2,-1)$
(перейти к ответу),

2.8. $A(-5, -4)$, $B(5, 6)$, $C_1(1,4)$, $C_2(-4,2)$, $C_3(-2,-5)$, $C_4(4,-3)$, $C_5(5,1)$
(перейти к ответу),

2.9. $A(4, 1)$, $B(-5, -2)$, $C_1(-3,-1)$, $C_2(1,-3)$, $C_3(4,-1)$, $C_4(2,3)$, $C_5(-2,2)$
(перейти к ответу),

2.10. $A(3, 5)$, $B(-5, -3)$, $C_1(-4,-1)$, $C_2(7,-1)$, $C_3(4,3)$, $C_4(-1,3)$, $C_5(-3,2)$
(перейти к ответу),

2.11. $A(4, -2)$, $B(1, 7)$, $C_1(3,3)$, $C_2(4,7)$, $C_3(7,5)$, $C_4(6,1)$, $C_5(3,2)$
(перейти к ответу).

3. Осуществить отсечение на плоскости полигона $A_1A_2...A_iA_1$ нерегулярным внешним отсекателем, представляющим собой выпуклый полигон $C_1C_2...C_nC_1$.

3.1. $A_1(14,8)$, $A_2(8,3)$, $A_3(3,2)$, $C_1(9,5)$, $C_2(14,10)$, $C_3(3,2)$
(перейти к ответу),

3.2. $A_1(1,-2)$, $A_2(3,-4)$, $A_3(0,13)$, $C_1(5,2)$, $C_2(1,-2)$, $C_3(-1,8)$
(перейти к ответу),

3.3. $A_1(2,8)$, $A_2(2,11)$, $A_3(4,7)$, $A_4(-2,4)$, $C_1(0,6)$, $C_2(5,11)$, $C_3(1,4)$
(перейти к ответу),

3.4. $A_1(-1,7)$, $A_2(0,1)$, $A_3(1,2)$, $A_4(4,0)$, $C_1(2,-3)$, $C_2(9,5)$, $C_3(11,-2)$
(перейти к ответу),

3.5. $A_1(13,2)$, $A_2(-3,9)$, $A_3(4,-4)$, $A_4(9,11)$, $C_1(9,4)$, $C_2(9,12)$, $C_3(3,14)$
(перейти к ответу),

3.6. $A_1(4,-4)$, $A_2(-3,-2)$, $A_3(6,4)$, $C_1(13,8)$, $C_2(9,12)$, $C_3(0,2)$, $C_4(5,-3)$
(перейти к ответу).

4. Среди отрезков A_iB_i, $i=\overline{1,m}$, найти все пересекающиеся пары и их пересечения при помощи алгоритма Бентли-Оттманна.

4.1. $A_1(4,3)$, $B_1(6,7)$, $A_2(1,0)$, $B_2(7,9)$, $A_3(9,5)$, $B_3(10,2)$
(перейти к ответу),

4.2. $A_1(1,8)$, $B_1(6,3)$, $A_2(4,8)$, $B_2(6,2)$, $A_3(3,5)$, $B_3(7,0)$
(перейти к ответу),

4.3. $A_1(0,7)$, $B_1(6,3)$, $A_2(6,2)$, $B_2(9,0)$, $A_3(0,10)$, $B_3(3,5)$
(перейти к ответу),

4.4. $A_1(1,4)$, $B_1(8,7)$, $A_2(2,5)$, $B_2(4,3)$, $A_3(5,4)$, $B_3(7,7)$
(перейти к ответу),

4.5. $A_1(6,2)$, $B_1(7,9)$, $A_2(6,2)$, $B_2(9,3)$, $A_3(1,4)$, $B_3(8,1)$, $A_4(0,10)$, $B_4(4,7)$
(перейти к ответу),

4.6. $A_1(3,1)$, $B_1(6,5)$, $A_2(5,5)$, $B_2(6,4)$, $A_3(0,4)$, $B_3(3,7)$, $A_4(4,2)$, $B_4(5,6)$
(перейти к ответу),

4.7. $A_1(6,3)$, $B_1(8,1)$, $A_2(1,8)$, $B_2(10,3)$, $A_3(0,0)$, $B_3(6,6)$, $A_4(3,6)$, $B_4(5,3)$
(перейти к ответу),

4.8. $A_1(1,1)$, $B_1(5,4)$, $A_2(4,7)$, $B_2(9,2)$, $A_3(1,5)$, $B_3(3,2)$, $A_4(6,1)$, $B_4(8,6)$
(перейти к ответу),

4.9. $A_1(2,1)$, $B_1(6,2)$, $A_2(1,1)$, $B_2(2,2)$, $A_3(2,8)$, $B_3(9,2)$, $A_4(7,8)$, $B_4(10,0)$
(перейти к ответу),

4.10. $A_1(0,1)$, $B_1(6,7)$, $A_2(4,2)$, $B_2(5,7)$, $A_3(5,2)$, $B_3(9,5)$, $A_4(1,1)$, $B_4(3,3)$
(перейти к ответу),

4.11. $A_1(0,1)$, $B_1(3,2)$, $A_2(2,2)$, $B_2(5,1)$, $A_3(4,1)$, $B_3(7,2)$, $A_4(6,2)$, $B_4(9,1)$, $A_5(8,0)$, $B_5(10,2)$, $A_6(9,3)$, $B_6(11,0)$

(перейти к ответу).

5. Осуществить отсечение на плоскости полигона $A_1A_2...A_1A_1$ нерегулярным внешним отсекателем, представляющим собой самонепересекающийся полигон $C_1C_2...C_nC_1$.

5.1. $A_1(2,6)$, $A_2(6,3)$, $A_3(7,1)$, $A_4(5,3)$, $A_5(1,2)$, $B_1(3,3)$, $B_2(5,2)$, $B_3(7,2)$, $B_4(2,0)$

(перейти к ответу),

5.2. $A_1(0,1)$, $A_2(1,8)$, $A_3(6,7)$, $C_1(4,4)$, $C_2(3,1)$, $C_3(1,4)$, $C_4(3,6)$, $C_5(2,8)$, $C_6(6,6)$

(перейти к ответу),

5.3. $A_1(-1,2)$, $A_2(6,4)$, $A_3(7,3)$, $A_4(0,1)$, $C_1(3,4)$, $C_2(5,2)$, $C_3(3,0)$, $C_4(0,0)$, $C_5(1,2)$, $C_6(0,4)$

(перейти к ответу),

5.4. $A_1(3,5)$, $A_2(6,4)$, $A_3(2,2)$, $A_4(4,4)$, $C_1(5,6)$, $C_2(3,0)$, $C_3(2,4)$, $C_4(3,4)$

(перейти к ответу),

5.5. $A_1(2,4)$, $A_2(4,2)$, $A_3(2,0)$, $A_4(0,2)$, $B_1(0,0)$, $B_2(3,2)$, $B_3(4,5)$, $B_4(5,2)$, $B_5(8,0)$, $B_6(4,1)$

(перейти к ответу)

5.6. $A_1(0,5)$, $A_2(3,3)$, $A_3(2,2)$, $A_4(-2,2)$, $A_5(-3,3)$, $C_1(0,3)$, $C_2(1,1)$, $C_3(2,3)$, $C_4(1,0)$, $C_5(-1,0)$, $C_6(-2,3)$, $C_7(-1,1)$

(перейти к ответу).

6. Определить часть границы выпуклого полигона $A_1A_2...A_nA_1$, видимую для дальнего наблюдателя, направление на которого характеризуется заданным вектором $\vec{s}$.

6.1. $A_1(0,10)$, $A_2(0,2)$, $A_3(3,0)$, $A_4(10,2)$, $A_5(4,9)$, $\vec{s}(2,-1)$

(перейти к ответу),

6.2. $A_1(7,9)$, $A_2(3,8)$, $A_3(1,2)$, $A_4(1,1)$, $A_5(4,0)$, $A_6(9,0)$, $A_7(10,1)$, $A_8(10,7)$, $\vec{s}(0,-1)$

(перейти к ответу),

6.3. $A_1(7,10)$, $A_2(0,10)$, $A_3(0,3)$, $A_4(8,0)$, $A_5(10,2)$, $A_6(9,5)$, $\vec{s}(-2,-1)$

(перейти к ответу),

6.4. $A_1(2,9)$, $A_2(0,2)$, $A_3(2,1)$, $A_4(9,1)$, $A_5(10,5)$, $A_6(10,8)$, $\vec{s}(1,-1)$

(перейти к ответу),

6.5. $A_1(5,9)$, $A_2(0,8)$, $A_3(1,3)$, $A_4(9,0)$, $A_5(10,2)$, $A_6(10,9)$, $\vec{s}(-1,5)$

(перейти к ответу),

6.6. $A_1(10,4)$, $A_2(7,9)$, $A_3(3,9)$, $A_4(2,7)$, $A_5(0,2)$, $A_6(5,2)$, $\vec{s}(-2,-1)$

(перейти к ответу),

6.7. $A_1(8,9)$, $A_2(0,10)$, $A_3(2,3)$, $A_4(7,2)$, $A_5(10,2)$, $\vec{s}(4,-3)$
(перейти к ответу),
6.8. $A_1(10,8)$, $A_2(6,10)$, $A_3(1,9)$, $A_4(0,8)$, $A_5(0,0)$, $A_6(9,0)$, $A_7(10,1)$, $\vec{s}(5,3)$
(перейти к ответу).

7. Определить часть границы выпуклого полигона $A_1A_2...A_nA_1$, видимую для ближнего наблюдателя, расположенного в заданной точке S.
7.1. $A_1(3,10)$, $A_2(1,3)$, $A_3(3,1)$, $A_4(8,0)$, $A_5(10,0)$, $A_6(9,8)$, $S(12,14)$
(перейти к ответу),
7.2. $A_1(10,4)$, $A_2(9,9)$, $A_3(2,8)$, $A_4(0,1)$, $A_5(5,0)$, $A_6(10,0)$, $S(11, -4)$
(перейти к ответу),
7.3. $A_1(1,2)$, $A_2(10,1)$, $A_3(10,9)$, $A_4(4,10)$, $A_5(1,8)$, $S(-3, 12)$
(перейти к ответу),
7.4. $A_1(9,10)$, $A_2(0,7)$, $A_3(0,3)$, $A_4(4,1)$, $A_5(8,3)$, $A_6(9,4)$, $S(-2, 2)$
(перейти к ответу),
7.5. $A_1(2,1)$, $A_2(6,0)$, $A_3(9,7)$, $A_4(9,10)$, $A_5(7,10)$, $A_6(3,8)$, $S(8, 2)$
(перейти к ответу),
7.6. $A_1(10,0)$, $A_2(10,6)$, $A_3(7,10)$, $A_4(5,10)$, $A_5(3,8)$, $A_6(0,3)$, $A_7(1,2)$, $A_8(9,0)$, $S(8, 2)$
(перейти к ответу),
7.7. $A_1(10,8)$, $A_2(7,10)$, $A_3(4,8)$, $A_4(1,5)$, $A_5(0,2)$, $A_6(6,3)$, $A_7(9,5)$, $S(8, 2)$
(перейти к ответу),
7.8. $A_1(9,9)$, $A_2(1,1)$, $A_3(4,2)$, $A_4(6,3)$, $A_5(8,6)$, $S(8, 1)$
(перейти к ответу).

2.2. Алгоритмы отсечения в трёхмерном пространстве

2.2.1. Отсечение отрезков трёхмерными выпуклыми отсекателями

При отсечении в трёхмерном пространстве в первую очередь необходимо задать отсекатель в каком-нибудь удобном виде, например найти его **матрицу тела**. При этом возможно проведение матричных вычислений, позволяющих одновременно проводить отсекание сразу нескольких геометрических объектов, либо алгоритмов, последовательно рассматривающих грани (стр. 57).

Пример 2.1. Осуществить отсечение отрезков A_1B_1 и A_2B_2 выпуклой пирамидой видимости, ограниченной лучами SC_i, $i=\overline{1,4}$:

$$A_1(2,1,3),\ B_1(0,1,2),\ A_2(2,3,1),\ B_2(-3,-2,1),\ S(0,0,5),$$

$$C_1(-3,2,0),\ C_2(3,2,0),\ C_3(3,-2,0),\ C_4(-3,-2,0).$$

Вначале найдём матрицу тела заданного отсекателя. Он ограничен четырьмя плоскостями: SC_1C_2, SC_2C_3, SC_3C_4, SC_4C_1 (Рисунок 2.25). Для этих плоскостей найдём их общие уравнения, причём необходимо их вывести исходя из векторов внутренней нормали:

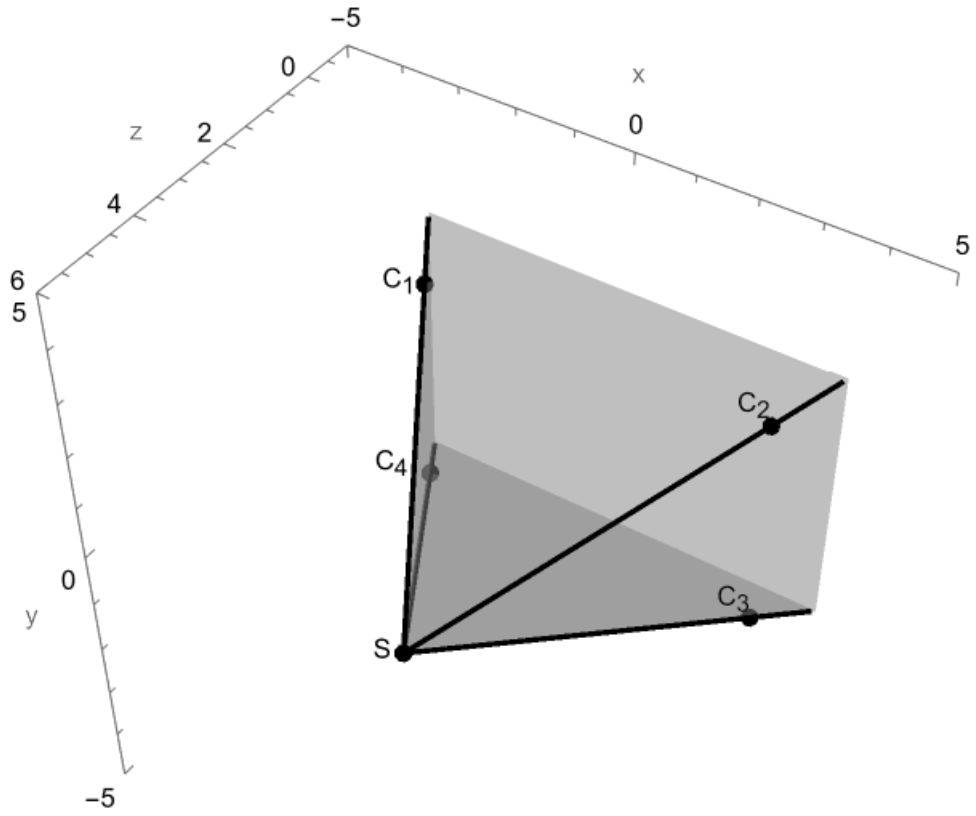


Рисунок 2.25 – Выпуклый внешний бесконечный отсекатель

$$SC_1C_2 : \overrightarrow{C_1C_2}(6,0,0), \overrightarrow{SC_1}(-3,2,-5), \overrightarrow{SC_3}(3,-2,-5),$$

$$\overrightarrow{C_1C_2} \times \overrightarrow{SC_1} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 6 & 0 & 0 \\ -3 & 2 & -5 \end{vmatrix} = 6 \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & 0 & 0 \\ -3 & 2 & -5 \end{vmatrix} = 6 \cdot (0,5,2),$$

$$\frac{1}{6}\overrightarrow{C_1C_2}\overrightarrow{SC_1}\overrightarrow{SC_3} = (0,5,2) \cdot (3,-2,-5) = -20 < 0,$$

$$\vec{n}_1 = \frac{1}{6} \cdot \operatorname{sgn}\left(\overrightarrow{C_1C_2}\overrightarrow{SC_1}\overrightarrow{SC_3}\right)\left(\overrightarrow{C_1C_2} \times \overrightarrow{SC_1}\right) = \vec{n}_1(0,-5,-2).$$

Итак, нашли вектор внутренней нормали для грани SC_1C_2. С учётом того, что на ней лежит точка S, получаем следующее уравнение:

$$SC_1C_2 : 0(x-0) - 5(y-0) - 2(z-5) = 0 \Rightarrow -5y - 2z + 10 = 0.$$

Таким образом, первой строкой матрицы тела будет (0, –5, –2, 10). Аналогично вычисляются остальные строки (учитываем то, что вершина *S* лежит также на всех остальных гранях):

$$SC_2C_3 : \overrightarrow{C_2C_3}(0,-4,0),$$

$$\overrightarrow{C_2C_3} \times \overrightarrow{SC_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & -4 & 0 \\ 3 & -2 & -5 \end{vmatrix} = -4 \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 1 & 0 \\ 3 & -2 & -5 \end{vmatrix} = -4 \cdot (-5,0,-3),$$

$$-\frac{1}{4}\overrightarrow{C_2C_3}\overrightarrow{SC_3}\overrightarrow{SC_1} = (-5,0,-3) \cdot (-3,2,-5) = 30 > 0,$$

$$\vec{n}_2 = -\frac{1}{4} \cdot \operatorname{sgn}\left(\overrightarrow{C_2C_3}\overrightarrow{SC_3}\overrightarrow{SC_1}\right)\left(\overrightarrow{C_2C_3} \times \overrightarrow{SC_3}\right) = \vec{n}_2(-5,0,-3),$$

$$-5(x-0) + 0(y-0) - 3(z-5) = 0,$$

$$-5x - 3z + 15 = 0;$$

$$SC_3C_4 : \overrightarrow{C_3C_4}(-6,0,0),$$

$$\overrightarrow{C_3C_4} \times \overrightarrow{SC_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -6 & 0 & 0 \\ 3 & -2 & -5 \end{vmatrix} = -6 \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & 0 & 0 \\ 3 & -2 & -5 \end{vmatrix} = -6 \cdot (0,5,-2),$$

$$-\frac{1}{6}\overrightarrow{C_3C_4}\overrightarrow{SC_3}\overrightarrow{SC_1} = (0,5,-2) \cdot (-3,2,-5) = 20 > 0,$$

$$\vec{n}_3 = -\frac{1}{6} \cdot \operatorname{sgn}\left(\overrightarrow{C_3C_4}\overrightarrow{SC_3}\overrightarrow{SC_1}\right)\left(\overrightarrow{C_3C_4} \times \overrightarrow{SC_3}\right) = \vec{n}_3(0,5,-2),$$

$$0(x-0) + 5(y-0) - 2(z-5) = 0,$$

$$5y - 2z + 10 = 0;$$

$$SC_4C_1 : \overrightarrow{C_4C_1}(0,4,0),$$

$$\overrightarrow{C_4C_1} \times \overrightarrow{SC_1} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 4 & 0 \\ -3 & 2 & -5 \end{vmatrix} = 4 \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 1 & 0 \\ -3 & 2 & -5 \end{vmatrix} = 4 \cdot (-5,0,3),$$

$$\frac{1}{4}\overrightarrow{C_4C_1}\overrightarrow{SC_1}\overrightarrow{SC_3} = (-5,0,3) \cdot (3,-2,-5) = -30 < 0,$$

$$\vec{n}_4 = \frac{1}{4} \cdot \operatorname{sgn}\left(\overrightarrow{C_4C_1}\overrightarrow{SC_1}\overrightarrow{SC_3}\right)\left(\overrightarrow{C_4C_1} \times \overrightarrow{SC_1}\right) = \vec{n}_4(5,0,-3),$$

$$5(x-0) + 0(y-0) - 3(z-5) = 0,$$

$$5x - 3z + 15 = 0.$$

Таким образом, получим следующую матрицу для отсекателя:

$$\mathcal{C} = \begin{pmatrix} 0 & -5 & -2 & 10 \\ -5 & 0 & -3 & 15 \\ 0 & 5 & -2 & 10 \\ 5 & 0 & -3 & 15 \end{pmatrix}.$$

Заданные же отрезки также можно представить в матричном виде:

$$\mathcal{S}_1 = (2-2t_1, 1, 3-t_1, 1)^T, \mathcal{S}_2 = (2-5t_2, 3-5t_2, 1, 1)^T. \qquad (2.11)$$

Из столбцов (2.11) составим матрицу и домножим её слева на $\mathcal{C}$:

$$\begin{pmatrix} 0 & -5 & -2 & 10 \\ -5 & 0 & -3 & 15 \\ 0 & 5 & -2 & 10 \\ 5 & 0 & -3 & 15 \end{pmatrix} \begin{pmatrix} 2-2t_1 & 2-5t_2 \\ 1 & 3-5t_2 \\ 3-t_1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} -1+2t_1 & -7+25t_2 \\ -4+13t_1 & 2+25t_2 \\ 9+2t_1 & 23-25t_2 \\ 16-7t_1 & 22-25t_2 \end{pmatrix}. \qquad (2.12)$$

Для того, чтобы некоторая точка, лежащая на отрезке A_1B_1 либо A_2B_2 и имеющая координаты из соответствующего столбца (2.11), лежала внутри отсекателя, необходимо и достаточно, чтобы все значения из соответствующего столбца в правой части (2.12) являлись положительными. Отсюда получаем системы неравенств:

$$A_1B_1: \begin{cases} 0 \le t_1 \le 1, \\ -1+2t_1 > 0, \\ -4+13t_1 > 0, \\ 9+2t_1 > 0, \\ 16-7t_1 > 0, \end{cases} \quad A_2B_2: \begin{cases} 0 \le t_2 \le 1, \\ -7+25t_2 > 0, \\ 2+25t_2 > 0, \\ 23-25t_2 > 0, \\ 22-25t_2 > 0. \end{cases}$$

В первой системе из второго, третьего и четвёртого неравенств получим $t_1 > 1/2$, из последнего – $t_1 < 16/7$. В итоге с учётом первого неравенства имеем $t_1 \in (1/2, 1]$. Аналогично из второй системы имеем $t_2 \in (7/25, 22/25)$. Подставляя концы этих промежутков в (2.11), окончательно получим, что результат отсечения первого отрезка – отрезок $A_1'B_1$, а второго – $A_2'B_2'$, где координаты новых точек равны $A_1'(1, 1, 5/2)$, $A_2'(3/5, 8/5, 1)$, $B_2'(-12/5, -7/5, 1)$. На рисунке 2.26 два исходных отрезка обозначены синим и красным цветами, их части, расположенные внутри отсекателя – фиолетовым.

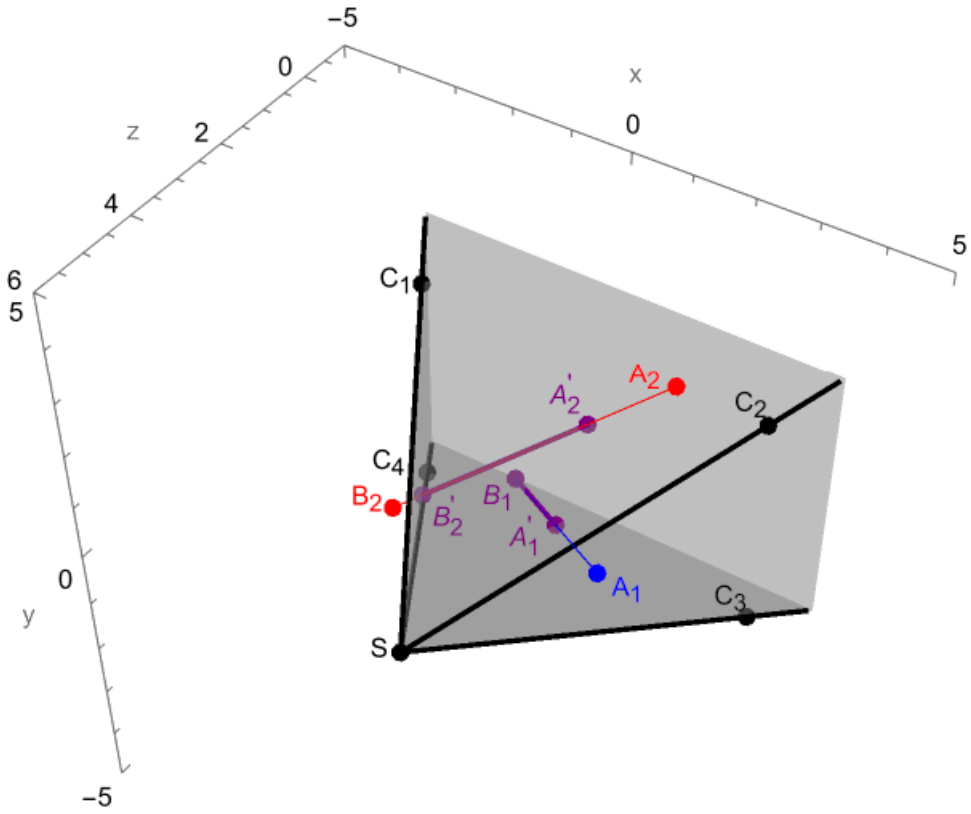


Рисунок 2.26 – Результат отсечения отрезков

Пример 2.2. Осуществить отсечение отрезков A_1B_1 и A_2B_2 внешним выпуклым отсекателем в форме полиэдра с вершинами в точках C_i, $i=\overline{1,5}$ (Рисунок 2.27):

$$A_1(1,1,1),\ B_1(0,1,-2),\ A_2(-1,0,1),\ B_2(2,1,-1),$$
$$C_1(0,0,0),\ C_2(2,0,0),\ C_3(0,2,0),\ C_4(0,0,2),\ C_5(2,2,0).$$

Обозначим координаты точек C_i через (x_i, y_i, z_i). Как и в предыдущем примере, вычислим матрицу тела для заданного отсекателя:

$$C_1C_2C_5C_3 : \overrightarrow{C_1C_2}(2,0,0), \overrightarrow{C_2C_5}(0,2,0), \overrightarrow{C_1C_4}(0,0,2),$$

$$\overrightarrow{C_1C_2} \times \overrightarrow{C_2C_5} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{vmatrix} = (0,0,4) = 4 \cdot (0,0,1),$$

$$\frac{1}{4}\overrightarrow{C_1C_2}\overrightarrow{C_2C_5}\overrightarrow{C_1C_4} = (0,0,1) \cdot (0,0,2) = 2 > 0,$$

$$\vec{n}_1 = \frac{1}{4}\operatorname{sgn}\left(\overrightarrow{C_1C_2}\overrightarrow{C_2C_5}\overrightarrow{C_1C_4}\right)\left(\overrightarrow{C_1C_2} \times \overrightarrow{C_2C_5}\right) = \vec{n}_1(0,0,1),$$

$$z - z_1 = 0 \Rightarrow z = 0;$$

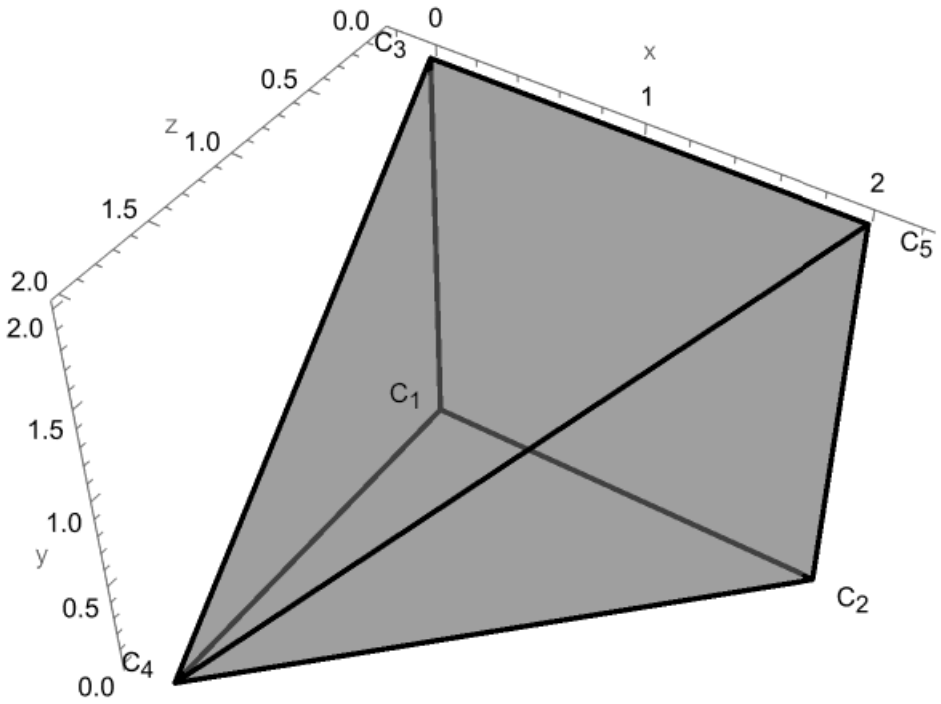


Рисунок 2.27 – Выпуклый полиэдр

$$C_4C_1C_3 : \overrightarrow{C_4C_1}(0,0,-2), \overrightarrow{C_4C_3}(0,2,-2), \overrightarrow{C_4C_2}(2,0,-2),$$

$$\overrightarrow{C_4C_1} \times \overrightarrow{C_4C_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 0 & -2 \\ 0 & 2 & -2 \end{vmatrix} = (4,0,0) = 4 \cdot (1,0,0),$$

$$\frac{1}{4}\overrightarrow{C_4C_1}\overrightarrow{C_4C_3}\overrightarrow{C_4C_2} = (1,0,0) \cdot (2,0,-2) = 2 > 0,$$

$$\vec{n}_2 = \frac{1}{4}\operatorname{sgn}\left(\overrightarrow{C_4C_1}\overrightarrow{C_4C_3}\overrightarrow{C_4C_2}\right)\left(\overrightarrow{C_4C_1} \times \overrightarrow{C_4C_3}\right) = \vec{n}_2(1,0,0),$$

$$x - x_1 = 0,$$

$$x = 0;$$

$$C_4C_3C_5 : \overrightarrow{C_4C_5}(2,2,-2),$$

$$\overrightarrow{C_4C_5} \times \overrightarrow{C_4C_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 2 & 2 & -2 \\ 0 & 2 & -2 \end{vmatrix} = (0,4,4) = 4 \cdot (0,1,1),$$

$$\frac{1}{4}\overrightarrow{C_4C_5}\overrightarrow{C_4C_3}\overrightarrow{C_4C_2} = (0,1,1) \cdot (2,0,-2) = -2 < 0,$$

$$\vec{n}_3 = \frac{1}{4}\operatorname{sgn}\left(\overrightarrow{C_4C_5}\overrightarrow{C_4C_3}\overrightarrow{C_4C_2}\right)\left(\overrightarrow{C_4C_5} \times \overrightarrow{C_4C_3}\right) = \vec{n}_3(0,-1,-1),$$

$$-(y - y_3) - (z - z_3) = 0,$$

$$-y - z + 2 = 0;$$

$$C_4C_5C_2 : \overrightarrow{C_4C_5} \times \overrightarrow{C_4C_2} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 2 & 2 & -2 \\ 2 & 0 & -2 \end{vmatrix} = (-4,0,-4) = -4 \cdot (1,0,1),$$

$$-\frac{1}{4}\overrightarrow{C_4C_5}\overrightarrow{C_4C_2}\overrightarrow{C_4C_1} = (1,0,1) \cdot (0,0,-2) = -2 < 0,$$

$$\vec{n}_4 = -\frac{1}{4}\operatorname{sgn}\left(\overrightarrow{C_4C_5}\overrightarrow{C_4C_2}\overrightarrow{C_4C_1}\right)\left(\overrightarrow{C_4C_5} \times \overrightarrow{C_4C_2}\right) = \vec{n}_4(-1,0,-1),$$

$$-(x - x_4) - (z - z_4) = 0,$$

$$-x - z + 2 = 0;$$

$$C_4C_2C_1 : \overrightarrow{C_4C_2} \times \overrightarrow{C_4C_1} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 2 & 0 & -2 \\ 0 & 0 & -2 \end{vmatrix} = (0,4,0) = 4 \cdot (0,1,0),$$

$$\frac{1}{4}\overrightarrow{C_4C_2}\overrightarrow{C_4C_1}\overrightarrow{C_4C_5} = (0,1,0) \cdot (2,2,-2) = 2 > 0,$$

$$\vec{n}_5 = \frac{1}{4}\operatorname{sgn}\left(\overrightarrow{C_4C_2}\overrightarrow{C_4C_1}\overrightarrow{C_4C_5}\right)\left(\overrightarrow{C_4C_2} \times \overrightarrow{C_4C_1}\right) = \vec{n}_5(0,1,0),$$

$$y - y_4 = 0,$$

$$y = 0.$$

Таким образом, для отсекателя получаем следующую матрицу тела:

$$\mathcal{C} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 2 \\ -1 & 0 & -1 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Отрезки представим в виде следующих столбцов:

$$\mathcal{S}_1 = (1 - t_1, 1, 1 - 3t_1, 1)^T, \ \mathcal{S}_2 = (-1 + 3t_2, t_2, 1 - 2t_2, 1)^T. \qquad (2.13)$$

Перемножим матрицы $\mathcal{C}$ и $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 2 \\ -1 & 0 & -1 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1-t_1 & -1+3t_2 \\ 1 & t_2 \\ 1-3t_1 & 1-2t_2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1-3t_1 & 1-2t_2 \\ 1-t_1 & -1+3t_2 \\ 3t_1 & 1+t_2 \\ 4t_1 & 2-t_2 \\ 1 & t_2 \end{pmatrix}.$$

Сравнивая все элементы первого столбца полученной матрицы с нулём, получим, что они все больше ноля тогда и только тогда, когда $t_1 \in (0,1/3)$. Аналогично для второго столбца имеем $t_2 \in (1/3,1/2)$. Подставляя концы этих промежутков в (2.13), окончательно получим отрезки A_1B_1' и $A_2'B_2'$, где найденные точки имеют координаты $B_1'(2/3,1,0)$, $A_2'(0,1/3,1/3)$, $B_2'(1/2,1/2,0)$ (Рисунок 2.28).

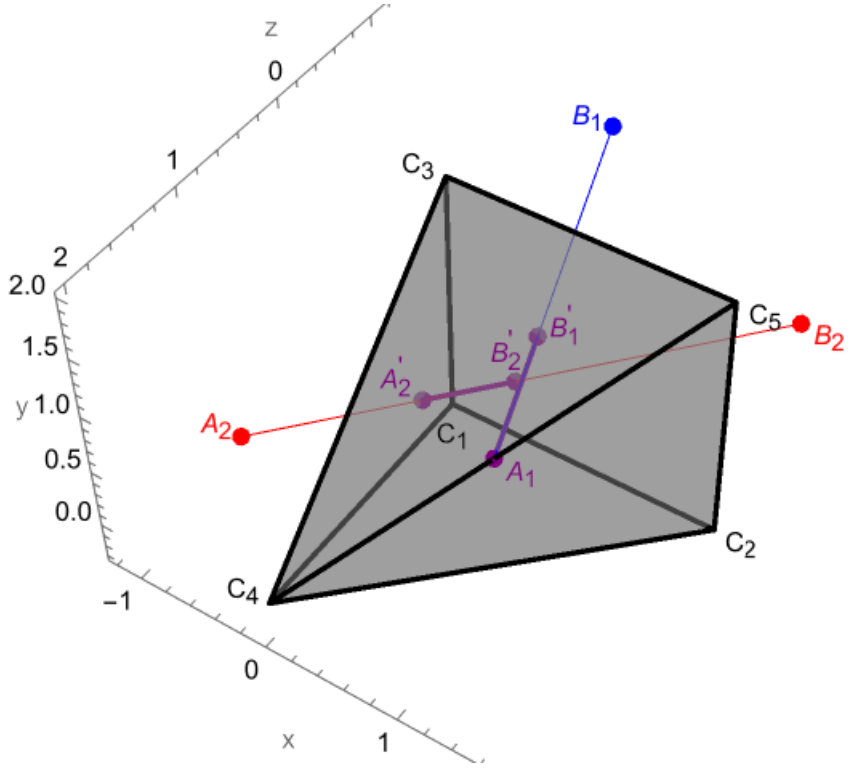


Рисунок 2.28 – Отсечение отрезков выпуклым полиэдром

2.2.2. Алгоритм Робертса удаления нелицевых граней выпуклого полиэдра

Для решения различных задач, связанных с визуализацией трёхмерной сцены, а также для упрощения многих алгоритмов часто необходимо определять лицевые и невидимые грани и поверхности тел. Алгоритм Робертса

(стр. 60) позволяет быстро и эффективно решить эту задачу для выпуклого полиэдра при заданном наблюдателе.

Пример 2.3. Для полиэдра с вершинами в точках A_i, $i=\overline{1,5}$, и гранями $\triangle A_3A_5A_1$, $\triangle A_5A_2A_1$, $\triangle A_2A_3A_1$, $\triangle A_3A_4A_5$, $\triangle A_4A_2A_5$, $\triangle A_2A_4A_3$ определить лицевые и невидимые грани при дальнем наблюдателе, расположенном в направлении $\vec{s}(1,2,1)$:

$$A_1(2,0,2),\ A_2(4,7,8),\ A_3(3,9,4),\ A_4(5,9,3),\ A_5(6,7,1).$$

Для всех граней вначале необходимо вычислить векторы внешних нормалей:

$$\overrightarrow{A_3A_5}\times\overrightarrow{A_5A_1}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\3&-2&-3\\-4&-7&1\end{vmatrix}=\vec{n}_{351}(-23,9,-29),\ \overrightarrow{A_5A_2}\times\overrightarrow{A_2A_1}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\-2&0&7\\-2&-7&-6\end{vmatrix}=\vec{n}_{521}(49,-26,14),$$

$$\overrightarrow{A_2A_3}\times\overrightarrow{A_3A_1}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\-1&2&-4\\-1&-9&-2\end{vmatrix}=\vec{n}_{231}(-40,2,11),\ \overrightarrow{A_3A_4}\times\overrightarrow{A_4A_5}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\2&0&-1\\1&-2&-2\end{vmatrix}=\vec{n}_{345}(-2,3,-4),$$

$$\overrightarrow{A_4A_2}\times\overrightarrow{A_2A_5}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\-1&-2&5\\2&0&-7\end{vmatrix}=\vec{n}_{425}(14,3,4),\ \overrightarrow{A_2A_4}\times\overrightarrow{A_4A_3}=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\1&2&-5\\-2&0&1\end{vmatrix}=\vec{n}_{243}(2,9,4).$$

Все эти векторы являются внешними (а не внутренними) нормалями, что можно проверить, вычислив знак скалярного произведения каждого из этих векторов с вектором, проведённым из некоторой точки соответствующей грани к любой внутренней точке полиэдра или его вершине, не лежащей на той же грани. Например, от грани $\triangle A_3A_5A_1$ можно провести вектор $\overrightarrow{A_5A_2}$, направленный внутрь полиэдра:

$$\vec{n}_{351}\cdot\overrightarrow{A_5A_2}=-23\cdot(-2)+9\cdot0-29\cdot7<0.$$

Далее нужно вычислить знаки скалярных произведений:

$$\vec{n}_{351}\cdot\vec{s}=-23\cdot1+9\cdot2-29\cdot1<0,$$
$$\vec{n}_{521}\cdot\vec{s}=49\cdot1-26\cdot2+14\cdot1=49-52+14>0,$$
$$\vec{n}_{231}\cdot\vec{s}=-40\cdot1+2\cdot2+11\cdot1<0,$$
$$\vec{n}_{345}\cdot\vec{s}=-2\cdot1+3\cdot2-4\cdot1=0,$$

$$\vec{n}_{425} \cdot \vec{s} = 14 \cdot 1 + 3 \cdot 2 + 4 \cdot 1 > 0,$$
$$\vec{n}_{243} \cdot \vec{s} = 2 \cdot 1 + 9 \cdot 2 + 4 \cdot 1 > 0.$$

Только те грани, для которых эти скалярные произведения оказались положительными, являются лицевыми. Грань $\triangle A_3A_4A_5$, для которой получилось $\vec{n}_{345} \cdot \vec{s} = 0,$ с точки зрения заданного наблюдателя будет вырождаться в отрезок. Чертёж на рисунке 2.29 можно получить, построив по вектору $\vec{s}$ аксонометрическую проекцию полиэдра и координатных осей. Здесь рёбра лицевых граней изображены жирными линиями, остальные – пунктирными. Проекции точек A_3, A_4 и A_5 оказываются лежащими на одной прямой.

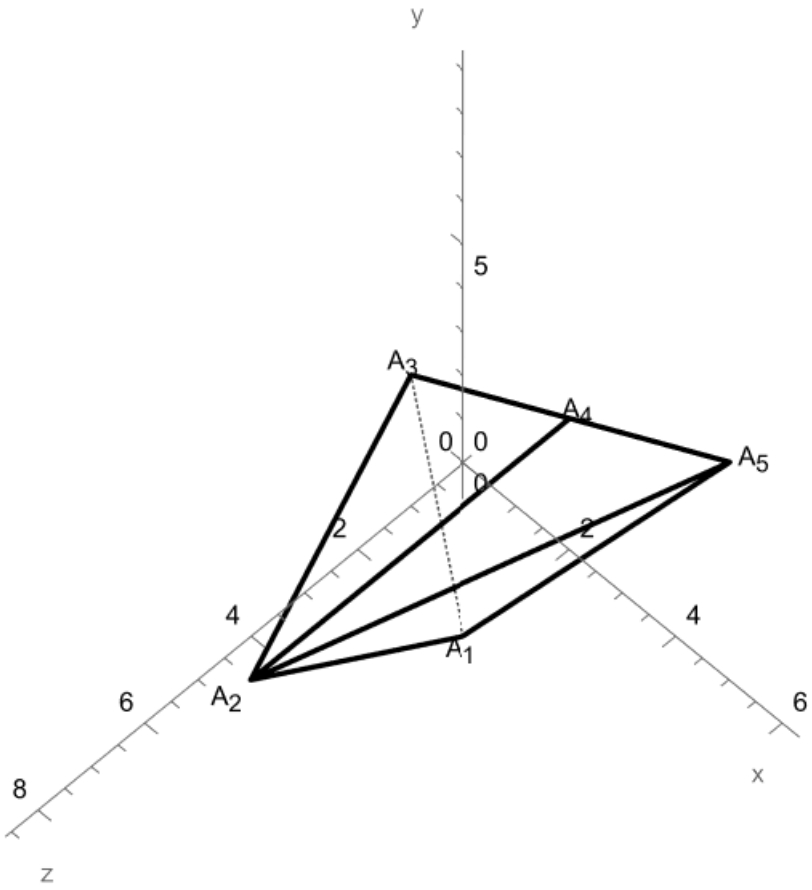


Рисунок 2.29 – Аксонометрическая проекция полиэдра –
вид с точки зрения дальнего наблюдателя, заданного вектором $\vec{s}$

Пример 2.4. Для выпуклого тела из примера 2.3 определить лицевые грани при ближнем наблюдателе, расположенном в точке *S*(7, 5, 4).

Здесь для вычисления скалярных произведений вместо вектора $\vec{s}$ уже следует использовать векторы, проведённые от каждой грани к точке *S*. Это может быть любая точка грани, например, её центр или одна из вершин:

$$\vec{n}_{351} \cdot \overrightarrow{A_3S} = -23 \cdot 4 + 9 \cdot (-4) - 29 \cdot 0 < 0,$$
$$\vec{n}_{521} \cdot \overrightarrow{A_5S} = 49 \cdot 1 - 26 \cdot 2 + 14 \cdot 3 = 49 - 52 + 14 > 0,$$
$$\vec{n}_{231} \cdot \overrightarrow{A_2S} = -40 \cdot 3 + 2 \cdot (-2) + 11 \cdot (-4) < 0,$$
$$\vec{n}_{345} \cdot \overrightarrow{A_3S} = -2 \cdot 4 + 3 \cdot (-4) - 4 \cdot 0 < 0,$$
$$\vec{n}_{425} \cdot \overrightarrow{A_4S} = 14 \cdot 2 + 3 \cdot (-4) + 4 \cdot 1 > 0,$$
$$\vec{n}_{243} \cdot \overrightarrow{A_2S} = 2 \cdot 3 + 9 \cdot (-2) + 4 \cdot (-4) < 0.$$

Значит, для такого наблюдателя лицевыми являются уже только две грани $\triangle A_5A_2A_1$ и $\triangle A_4A_2A_5$ (Рисунок 2.30).

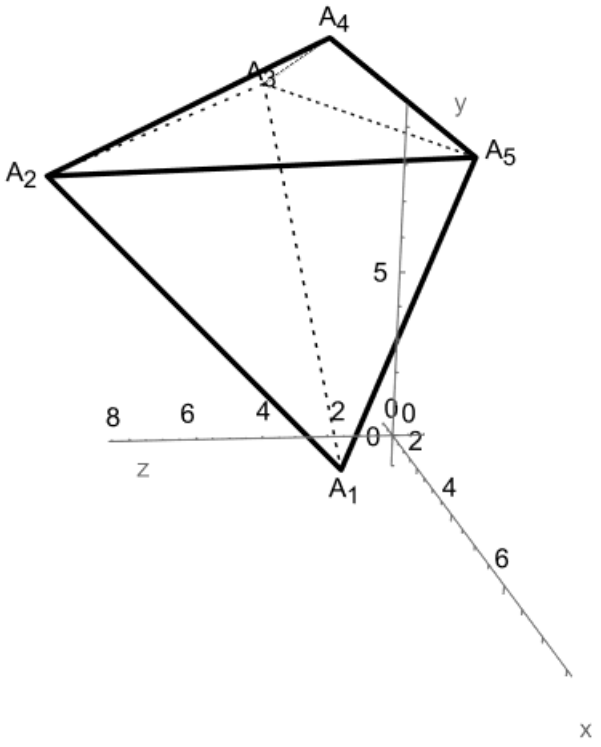


Рисунок 2.30 – Вид полиэдра из точки *S*

2.2.3. Удаление отрезка при экранировании выпуклым полиэдром

При визуализации сцены, содержащей несколько объектов, важно уметь определять, какие из объектов или их частей загораживаются другими, а следовательно, подлежат удалению. Алгоритм, описанный на стр. 70, позволяет осуществить удаление части отрезка с заданными концами выпуклым телом, заданным в виде матрицы тела.

Пример 2.5. Осуществить экранирование отрезка AB выпуклым полиэдром с вершинами $A_i(x_i, y_i, z_i)$, $i = \overline{1,5}$, и гранями $\triangle A_2A_1A_3$, $\triangle A_1A_5A_3$, $\triangle A_2A_5A_1$, $\triangle A_4A_2A_3$, $\triangle A_5A_4A_3$, $\triangle A_4A_5A_2$, при заданном направлении на дальнего наблюдателя $\vec{s}$:

$$A(1,1,5),\ B(1,2,1),$$
$$A_1(4,1,4),\ A_2(1,0,4),\ A_3(0,4,4),\ A_4(0,1,3),\ A_5(2,1,0),$$
$$\vec{s}(1,1,1).$$

Выпишем формулу (1.26) для данного примера. В случае дальнего наблюдателя нужно положить значение индикатора близости $h_s = 0$:

$$\tilde{L}(t,\alpha) = \tilde{A} + (1 - h_s\alpha)t\left(\tilde{B} - \tilde{A}\right) + \alpha\left(\tilde{S} - h_s\tilde{A}\right) =$$
$$= \begin{pmatrix}1\\1\\5\\1\end{pmatrix} + t\begin{pmatrix}0\\1\\-4\\0\end{pmatrix} + \alpha\begin{pmatrix}1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}1+\alpha\\1+t+\alpha\\5-4t+\alpha\\1\end{pmatrix}. \qquad (2.14)$$

Заметим, что эту формулу можно представить немного иначе:

$$\tilde{L}(t,\alpha) = \tilde{A} + (1 - h_s\alpha)t\left(\tilde{B} - \tilde{A}\right) + \alpha\left(\tilde{S} - h_s\tilde{A}\right) =$$
$$= \left(\tilde{A}, \tilde{B} - \tilde{A}, \tilde{S} - h_s\tilde{A}\right)\begin{pmatrix}1\\(1-h_s\alpha)t\\\alpha\end{pmatrix} = \begin{pmatrix}1&0&1\\1&1&1\\5&-4&1\\1&0&0\end{pmatrix}\begin{pmatrix}1\\t\\\alpha\end{pmatrix} = \begin{pmatrix}1+\alpha\\1+t+\alpha\\5-4t+\alpha\\1\end{pmatrix}. \qquad (2.15)$$

Этот столбец нужно домножить слева на матрицу тела заданного полиэдра. Вычислим её:

$$\triangle A_2A_1A_3 : \overrightarrow{A_2A_1} \times \overrightarrow{A_1A_3} = \begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\3&1&0\\-4&3&0\end{vmatrix} = 13\vec{k} \,||\, \vec{k},$$
$$\vec{k} \cdot \overrightarrow{A_2A_4} = 1 \cdot (-1) < 0 \Rightarrow \vec{n}_{213}(0,0,-1),$$
$$-(z - z_2) = 0 \Rightarrow -z + 4 = 0;$$

$$\triangle A_1A_5A_3 : \overrightarrow{A_1A_3} \times \overrightarrow{A_1A_5} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -4 & 3 & 0 \\ -2 & 0 & -4 \end{vmatrix} = -12\vec{i} - 16\vec{j} + 6\vec{k} \,||\, -6\vec{i} - 8\vec{j} + 3\vec{k},$$

$$(-6\vec{i} - 8\vec{j} + 3\vec{k}) \cdot \overrightarrow{A_1A_2} = -6 \cdot (-3) - 8 \cdot (-1) + 3 \cdot 0 > 0 \Rightarrow \vec{n}_{153}(-6,-8,3),$$

$$-6(x - x_5) - 8(y - y_5) + 3(z - z_5) = 0 \Rightarrow -6x - 8y + 3z + 20 = 0;$$

$$\triangle A_2A_5A_1 : \overrightarrow{A_2A_1} \times \overrightarrow{A_1A_5} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 3 & 1 & 0 \\ -2 & 0 & -4 \end{vmatrix} = -4\vec{i} + 12\vec{j} + 2\vec{k} \,||\, -2\vec{i} + 6\vec{j} + \vec{k},$$

$$(-2\vec{i} + 6\vec{j} + \vec{k}) \cdot \overrightarrow{A_1A_3} = -2 \cdot (-4) + 6 \cdot 3 + 1 \cdot 0 > 0 \Rightarrow \vec{n}_{251}(-2,6,1),$$

$$-2(x - x_5) + 6(y - y_5) + (z - z_5) = 0 \Rightarrow -2x + 6y + z - 2 = 0;$$

$$\triangle A_4A_2A_3 : \overrightarrow{A_4A_2} \times \overrightarrow{A_2A_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & -1 & 1 \\ -1 & 4 & 0 \end{vmatrix} = -4\vec{i} - \vec{j} + 3\vec{k},$$

$$(-4\vec{i} - \vec{j} + 3\vec{k}) \cdot \overrightarrow{A_2A_1} = -4 \cdot 3 - 1 + 3 \Rightarrow \vec{n}_{423}(4,1,-3),$$

$$4(x - x_4) + (y - y_4) - 3(z - z_4) = 0 \Rightarrow 4x + y - 3z + 8 = 0;$$

$$\triangle A_5A_4A_3 : \overrightarrow{A_5A_4} \times \overrightarrow{A_4A_3} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -2 & 0 & 3 \\ 0 & 3 & 1 \end{vmatrix} = -9\vec{i} + 2\vec{j} - 6\vec{k},$$

$$(-9\vec{i} + 2\vec{j} - 6\vec{k}) \cdot \overrightarrow{A_3A_1} = -9 \cdot 4 + 2 \cdot (-3) - 6 < 0 \Rightarrow \vec{n}_{543}(9,-2,6),$$

$$9(x - x_5) - 2(y - y_5) + 6(z - z_5) = 0 \Rightarrow 9x - 2y + 6z - 16 = 0;$$

$$\triangle A_4A_5A_2 : \overrightarrow{A_4A_2} \times \overrightarrow{A_5A_4} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & -1 & 1 \\ -2 & 0 & 3 \end{vmatrix} = -3\vec{i} - 5\vec{j} - 2\vec{k},$$

$$(-3\vec{i} - 5\vec{j} - 2\vec{k}) \cdot \overrightarrow{A_4A_3} = -3 \cdot 0 - 5 \cdot 3 - 2 \cdot 1 < 0 \Rightarrow \vec{n}_{452}(3,5,2),$$

$$3(x - x_5) + 5(y - y_5) + 2(z - z_5) = 0 \Rightarrow 3x + 5y + 2z - 11 = 0.$$

Собирая коэффициенты выписанных уравнений, получим следующую матрицу для заданного полиэдра:

$$\mathcal{C} = \begin{pmatrix} 0 & 0 & -1 & 4 \\ -6 & -8 & 3 & 20 \\ -2 & 6 & 1 & -2 \\ 4 & 1 & -3 & 8 \\ 9 & -2 & 6 & -16 \\ 3 & 5 & 2 & -11 \end{pmatrix}. \tag{2.16}$$

Таким образом, перемножая матрицы (2.16) и (2.14), получим матрицу из левых частей неравенств (1.27):

$$\mathcal{C} \cdot \tilde{L}(t,\alpha) = \begin{pmatrix} -1+4t-\alpha \\ 21-20t-11\alpha \\ 7+2t+5\alpha \\ -2+13t+2\alpha \\ 21-26t+13\alpha \\ 7-3t+10\alpha \end{pmatrix} \succ 0. \tag{2.17}$$

Для этой системы нужно найти все значения $t \in [0,1]$, для каждого из которых найдётся некоторое значение $\alpha \in [0,+\infty)$, при котором все неравенства окажутся верными. Но вначале для упрощения дальнейших вычислений проверим, есть ли среди неравенств в (2.17) те, что выполняются (или, наоборот, нарушаются) для всех $t \in [0,1]$ и $\alpha \in [0,+\infty)$. Так, 3-е и 6-е неравенства (т.е. $7+2t+5\alpha > 0$ и $7-3t+10\alpha > 0$) всегда выполняются для указанных значений t и α, так как сумма первых двух слагаемых в обоих неравенствах больше нуля для всякого $t \in [0,1]$, и последние слагаемые также больше нуля для всех $\alpha \in [0,+\infty)$. Для всех остальных неравенств найдутся наборы значений, при которых они будут выполняться или нарушаться. Значит, можно обойтись системой из четырёх неравенств:

$$\begin{cases} -1+4t-\alpha > 0, \\ 21-20t-11\alpha > 0, \\ -2+13t+2\alpha > 0, \\ 21-26t+13\alpha > 0. \end{cases} \tag{2.18}$$

Далее систему (2.18) можно исследовать следующим образом:
- При $t = 0$ имеем систему

$$\begin{cases} -1-\alpha>0, \\ 21-11\alpha>0, \\ -2+2\alpha>0, \\ 21+13\alpha>0. \end{cases}$$

Она является несовместной при $\alpha\in[0,+\infty)$, значит, точка A исходного отрезка экранирована не будет.

- При t = 1 получаются следующие неравенства:

$$\begin{cases} 3-\alpha>0, \\ 1-11\alpha>0, \\ -11+2\alpha>0, \\ -5+13\alpha>0. \end{cases}$$

Из второго неравенства этой системы имеем $\alpha<1/11$, а из третьего – $\alpha>11/2$. Таким образом, снова получается несовместная система – конец B исходного отрезка экранирован не будет.

- $\alpha=0$. При подстановке этого значения в (2.18) имеем следующую систему:

$$\begin{cases} -1+4t>0, \\ 21-20t>0, \\ -2+13t>0, \\ 21-26t>0. \end{cases}$$

Отсюда можно найти часть отрезка, лежащая внутри заданного полиэдра. Решая её, получим множество соответствующих параметров t: $t\in(1/4,21/26)$. Значит, пока экранироваться будет часть исходного отрезка с параметрами $t_{\text{нач}}:=\frac{1}{4},\ t_{\text{кон}}:=\frac{21}{26}$.

После этого на основе всех пар неравенств системы (2.18) составляем системы уравнений, которые дадут решения (t_{ij},α_{ij}). Для этих решений затем нужно проверить, будут ли при них выполняться все оставшиеся неравенства.

Так, возьмём вначале первые два неравенства и составим из них систему уравнений:

$$\begin{cases} -1+4t-\alpha=0, \\ 21-20t-11\alpha=0. \end{cases}$$

Её решение: $t_{12}=1/2,\ \alpha_{12}=1$. Однако значение t_{12} уже находится внутри отрезка $[t_{\text{нач}},t_{\text{кон}}]$, значит, проверка полученного решения не даст никакой новой информации: точка с параметром t_{12} и так экранируется полиэдром.

Аналогично решим остальные пять систем. При этом вначале будут получаться отрицательные значения t_{ij} или α_{ij}, что не даёт новых экранируемых точек отрезка:

$$\begin{cases}-1+4t-\alpha=0,\\ -2+13t+2\alpha=0\end{cases}\Rightarrow t_{13}=\frac{4}{21},\alpha_{13}=-\frac{5}{21}<0;$$

$$\begin{cases}-1+4t-\alpha=0,\\ 21-26t+13\alpha=0\end{cases}\Rightarrow t_{14}=-\frac{4}{13}<0,\alpha_{14}=-\frac{29}{13}<0;$$

$$\begin{cases}21-20t-11\alpha=0,\\ -2+13t+2\alpha=0\end{cases}\Rightarrow t_{23}=-\frac{20}{103}<0,\alpha_{23}=\frac{233}{103}.$$

Рассматривая очередную систему, получим следующее решение:

$$\begin{cases}21-20t-11\alpha=0,\\ 21-26t+13\alpha=0\end{cases}\Rightarrow t_{24}=\frac{12}{13},\alpha_{24}=\frac{3}{13}.$$

Здесь уже имеем $t_{24}\in[0,1],\ \alpha_{24}>0$, а также $t_{24}>t_{\text{кон}}$. Таким образом, нужно подставить эти решения в остальные неравенства системы (2.18):

$$-1+4t_{24}-\alpha_{24}=-1+4\cdot\frac{12}{13}-\frac{3}{13}=\frac{-13+48-3}{13}>0,$$

$$-2+13t_{24}+2\alpha_{24}=-2+13\cdot\frac{12}{13}+2\cdot\frac{3}{13}>0.$$

Они выполняются. Значит, точка с параметром t_{24} действительно экранируется полиэдром, а следовательно, нужно переопределить одно из значений параметра для искомого отрезка: $t_{\text{кон}}:=t_{24}=12/13$.

Наконец, шестая система

$$\begin{cases}-2+13t+2\alpha=0,\\ 21-26t+13\alpha=0\end{cases}$$

имеет решение $t_{34}=4/13,\alpha_{34}=-1$, которое также пропускается, потому что, во-первых, $t_{34}\in[t_{\text{нач}},t_{\text{кон}}]$, во-вторых, $\alpha_{34}<0$. Итак, окончательно имеем, что часть отрезка *AB*, соответствующая числовому отрезку $t\in[t_{\text{нач}},t_{\text{кон}}]=[1/4,12/13]$,

экранируется полиэдром, а вся оставшаяся будет видна для заданного наблюдателя. Несложно найти координаты точек конца экранируемого отрезка:

$$AB:\begin{cases} x=1, \\ y=1+t, \\ z=5-4t, \end{cases} \quad A'=A+t_{\text{нач}}\overrightarrow{AB}=A'\left(1,\frac{5}{4},4\right),\ B'=A+t_{\text{кон}}\overrightarrow{AB}=B'\left(1,\frac{25}{13},\frac{17}{13}\right).$$

На рисунке 2.31 искомый отрезок обозначен красным цветом (его часть, расположенная внутри полиэдра, нарисована пунктиром), видимая часть исходного отрезка – синим.

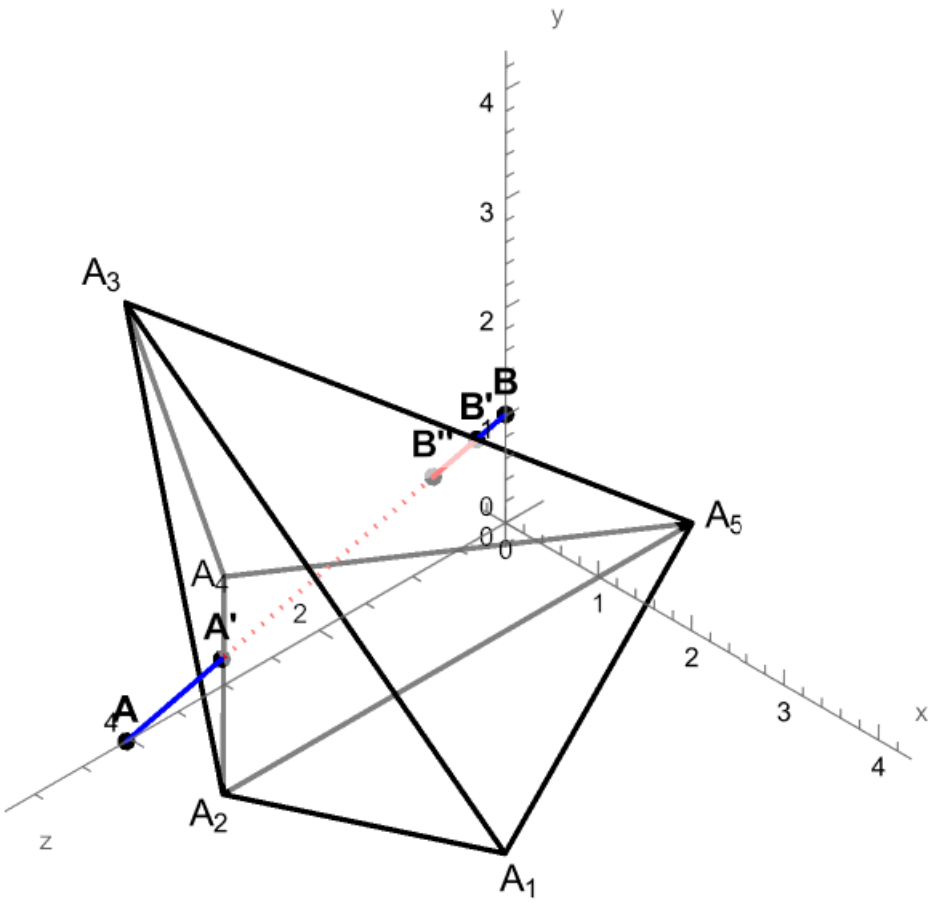


Рисунок 2.31 – Результат экранирования отрезка AB полиэдром $A_1A_2A_3A_4A_5A_6$ при дальнем наблюдателе

Пример 2.6. Осуществить экранирование отрезка с концами в точках $A(2, 2, 5)$ и $B(-2, 1, 2)$ полиэдром из предыдущего примера при ближнем наблюдателе в точке $S(0, 3, 1)$.

Используем формулу (2.15). Здесь получается уже другой столбец:

$$\tilde{L}(t,\alpha)=\left(\tilde{A},\tilde{B}-\tilde{A},\tilde{S}-h_s\tilde{A}\right)\begin{pmatrix}1\\(1-h_s\alpha)t\\\alpha\end{pmatrix}=[h_s=1,\ \text{так}$$

как здесь уже имеет место ближнее наблюдение] =

$$=\left(\tilde{A},\tilde{B}-\tilde{A},\tilde{S}-\tilde{A}\right)\begin{pmatrix}1\\(1-\alpha)t\\\alpha\end{pmatrix}=\begin{pmatrix}2&-4&-2\\2&-1&1\\5&-3&-4\\1&0&0\end{pmatrix}\begin{pmatrix}1\\(1-\alpha)t\\\alpha\end{pmatrix}.$$

Домножая этот столбец слева на матрицу тела (2.16), получим следующие коэффициенты для левых частей неравенств:

$$\mathcal{C}\cdot\tilde{L}(t,\alpha)=\begin{pmatrix}0&0&-1&4\\-6&-8&3&20\\-2&6&1&-2\\4&1&-3&8\\9&-2&6&-16\\3&5&2&-11\end{pmatrix}\begin{pmatrix}2&-4&-2\\2&-1&1\\5&-3&-4\\1&0&0\end{pmatrix}\begin{pmatrix}1\\(1-\alpha)t\\\alpha\end{pmatrix}=\begin{pmatrix}-1&3&4\\7&23&-8\\11&-1&6\\3&-8&5\\28&-52&-44\\15&-23&-9\end{pmatrix}\begin{pmatrix}1\\(1-\alpha)t\\\alpha\end{pmatrix}.$$

Таким образом, имеем следующую систему (коэффициенты в пятом неравенстве сокращены на 4):

$$\begin{cases}-1+3(1-\alpha)t+4\alpha>0,\\7+23(1-\alpha)t-8\alpha>0,\\11-(1-\alpha)t+6\alpha>0,\\3-8(1-\alpha)t+5\alpha>0,\\7-13(1-\alpha)t-11\alpha>0,\\15-23(1-\alpha)t-9\alpha>0.\end{cases}$$

Аналогично предыдущему примеру найдём неравенства, что выполняются или нарушаются при всех $t\in[0,1]$, $\alpha\in[0,1)$. Так, третье неравенство выполняется при всех таких значениях, так как при $\alpha=0$ его левая часть станет равной $11-t$, что больше нуля при всяком $t\in[0,1]$, а $\alpha=1$ даст тождественное неравенство $17>0$. Итак, остаётся система из пяти неравенств:

$$\begin{cases} -1+3(1-\alpha)t+4\alpha>0, \\ 7+23(1-\alpha)t-8\alpha>0, \\ 3-8(1-\alpha)t+5\alpha>0, \\ 7-13(1-\alpha)t-11\alpha>0, \\ 15-23(1-\alpha)t-9\alpha>0. \end{cases} \quad (2.19)$$

Далее проверяются следующие значения t и α:

- Подстановка $\alpha = 1$ позволяет проверить, действительно ли находится наблюдатель снаружи заданного полиэдра[1]. Если это вдруг окажется не так, то задача сведётся к отсечению отрезка внешним отсекателем (см. подраздел 2.2.1). Но в данном примере имеем нарушение четвёртого неравенства при такой подстановке, значит, всё-таки имеет место наблюдение снаружи.

- При $t = 0$ получается система

$$\begin{cases} -1+4\alpha>0, \\ 7-8\alpha>0, \\ 3+5\alpha>0, \\ 7-11\alpha>0, \\ 15-9\alpha>0, \end{cases}$$

которая имеет решение $\alpha \in (1/4, 7/11)$. Таким образом, система имеет решение, а значит, конец A экранируется полиэдром, и нужно положить $t_{\text{нач}} := 0$.

- При $t = 1$ получается система

$$\begin{cases} 2+\alpha>0, \\ 30-31\alpha>0, \\ -5+13\alpha>0, \\ -6+2\alpha>0, \\ -8+14\alpha>0. \end{cases}$$

Четвёртое неравенство не выполняется ни при каких $\alpha \in [0,1)$. Значит, вся система является несовместной, и конец B экранироваться заданным полиэдром не будет.

[1] Вообще говоря, подобную проверку лучше проводить перед применением алгоритма, просто домножив матрицу тела справа на координатный столбец точки S в однородных координатах

- Подстановка $\alpha = 0$ даёт следующую систему:

$$\begin{cases} -1+3t>0, \\ 7+23t>0, \\ 3-8t>0, \\ 7-13t>0, \\ 15-23t>0. \end{cases}$$

Её решение – $t \in (1/3, 3/8)$. С учётом ранее вычисленного значения для $t_{\text{нач}}$ получается, что нужно учитывать только правый конец этого решения: $t_{\text{кон}} := 3/8$.

После этого составляются системы уравнений на основе неравенств (2.19). Возьмём вначале два первых неравенства:

$$\begin{cases} -1+3(1-\alpha)t+4\alpha=0, \\ 7+23(1-\alpha)t-8\alpha=0. \end{cases}$$

Выражая t через α из обоих уравнений, получим уравнение относительно α:

$$t_{12} = \frac{1-4\alpha}{3(1-\alpha)} = \frac{-7+8\alpha}{23(1-\alpha)} \Rightarrow \frac{1-4\alpha}{3} = \frac{-7+8\alpha}{23} \Rightarrow \alpha_{12} = \frac{11}{29}.$$

Подставляя полученное значение α_{12} обратно в выражение для t_{12}, получим решение $t_{12} = \left(\frac{1-4\alpha}{3(1-\alpha)} \right)\Big|_{\alpha=\alpha_{12}} = \frac{1-4\cdot 11/29}{3(1-11/29)} = -\frac{5}{18}$. Это число меньше нуля, значит, полученное решение не изменяет отрезок $[t_{\text{нач}}, t_{\text{кон}}]$.

Аналогично решаются другие системы. В следующих системах получаются решения (t_{ij}, α_{ij}), которые также не учитываются в силу того, что по крайней мере одно из найденных неизвестных отрицательно:

$$\begin{cases} -1+3(1-\alpha)t+4\alpha=0, \\ 3-8(1-\alpha)t+5\alpha=0 \end{cases} \Rightarrow t_{13} = \frac{17}{48}, \alpha_{13} = -\frac{1}{47};$$

$$\begin{cases} -1+3(1-\alpha)t+4\alpha=0, \\ 7-13(1-\alpha)t-11\alpha=0 \end{cases} \Rightarrow t_{14} = \frac{17}{27}, \alpha_{14} = -\frac{8}{19};$$

$$\begin{cases} -1+3(1-\alpha)t+4\alpha=0, \\ 15-23(1-\alpha)t-9\alpha=0 \end{cases} \Rightarrow t_{15} = \frac{17}{29}, \alpha_{15} = -\frac{22}{65};$$

$$\begin{cases}7+23(1-\alpha)t-8\alpha=0,\\3-8(1-\alpha)t+5\alpha=0\end{cases}\Rightarrow t_{23}=-\frac{59}{176},\alpha_{23}=-\frac{125}{51};$$

$$\begin{cases}7+23(1-\alpha)t-8\alpha=0,\\7-13(1-\alpha)t-11\alpha=0\end{cases}\Rightarrow t_{24}=-\frac{1}{5},\alpha_{24}=\frac{12}{17};$$

$$\begin{cases}7+23(1-\alpha)t-8\alpha=0,\\15-23(1-\alpha)t-9\alpha=0\end{cases}\Rightarrow t_{25}=-\frac{57}{115},\alpha_{25}=\frac{22}{17}.$$

В очередной системе получаются следующие решения:

$$\begin{cases}3-8(1-\alpha)t+5\alpha=0,\\7-13(1-\alpha)t-11\alpha=0\end{cases}\Rightarrow t_{34}=\frac{1}{2},\alpha_{34}=\frac{1}{9}.$$

Для них уже, во-первых, выполняются неравенства $t_{34}\in[0,1]$, $\alpha\in[0,1)$, во-вторых, значение t_{34} находится за пределами отрезка $[t_{\text{нач}},t_{\text{кон}}]$. Подставляя найденные значения в систему (2.19), имеем выполнение всех неравенств (или их обращение в равенства). Поэтому нужно переопределить один из концов: $t_{\text{кон}}:=1/2$.

В следующей системе также получится подходящее решение:

$$\begin{cases}3-8(1-\alpha)t+5\alpha=0,\\15-23(1-\alpha)t-9\alpha=0\end{cases}\Rightarrow t_{35}=\frac{3}{4},\alpha_{35}=\frac{3}{11}.$$

При этом значение t_{35} снова лежит за пределами отрезка $[t_{\text{нач}},t_{\text{кон}}]$. Однако система (2.19) при этих значениях не выполняется.

Наконец, последняя система не даёт решения в нужных пределах:

$$\begin{cases}7-13(1-\alpha)t-11\alpha=0,\\15-23(1-\alpha)t-9\alpha=0\end{cases}\Rightarrow t_{45}=\frac{3}{5},\alpha_{45}=-\frac{1}{4}.$$

Таким образом, имеем окончательный числовой отрезок $[t_{\text{нач}},t_{\text{кон}}]=[0,1/2]$, что соответствует геометрическому отрезку AB', где новый конец имеет координаты $B'=A+\frac{1}{2}\overrightarrow{AB}=B'\left(0,\frac{3}{2},\frac{7}{2}\right)$. На рисунке 2.32 сечение полиэдра $A_1A_2A_3A_4A_5A_6$ треугольником SAB закрашено в светло-серый цвет. Это сечение является полигоном, среди вершин которого точки C_1 и C_2, ограничивающие часть отрезка AB внутри полиэдра, а также пересечение отрезков SB' и A_3A_4.

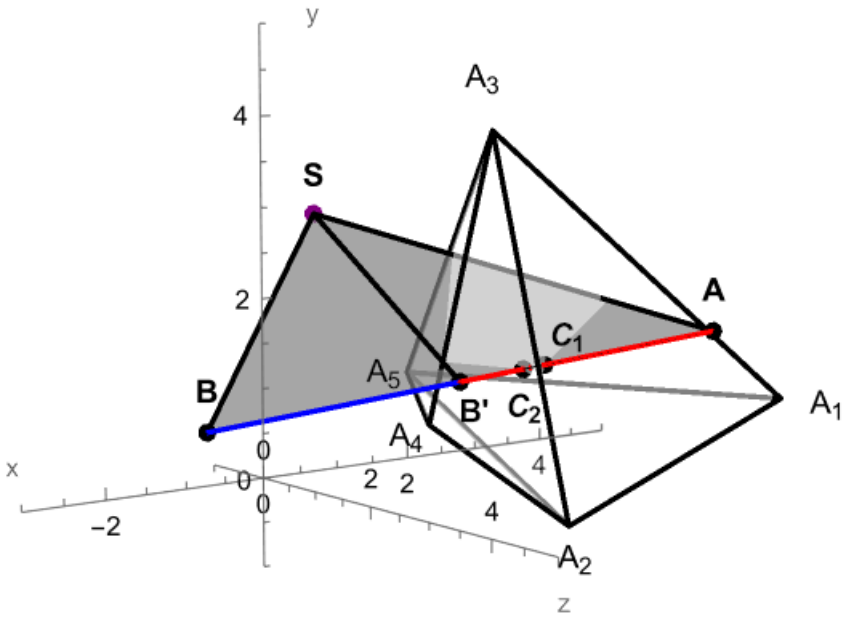


Рисунок 2.32 – Результат экранирования отрезка *AB* полиэдром $A_1A_2A_3A_4A_5A_6$ при ближнем наблюдателе

2.2.4. Алгоритм художника

Алгоритм художника определения порядка отрисовки плоских полигонов и прочих поверхностей (см. подраздел 1.2.4) позволяет отсортировать грани одного или нескольких полигонов по убыванию глубины, которая вычисляется для некоторой точки каждой грани в зависимости от типа и расположения наблюдателя.

Пример 2.7. Пользуясь алгоритмом художника, определить порядок вывода граней невыпуклого полиэдра $A_1A_2A_3A_4A_5A_6A_7A_8A_9$ с приведёнными ниже вершинами и гранями при дальнем наблюдателе $\vec{s}(1,0,1)$:

$$A_1(0,0,0),\ A_2(4,0,0),\ A_3(8,0,-4),\ A_4(8,0,-8),\ A_5(2,10,-4),$$
$$A_6(4,10,-6),\ A_7(1,10,-4),\ A_8(4,10,-7),\ A_9(0,20,-8),$$
$$G_1 = A_1A_2A_5A_7A_9A_1,\ G_2 = A_3A_4A_9A_8A_6A_3,\ G_3 = A_2A_3A_6A_5A_2,\ G_4 = A_5A_6A_8A_7A_5,$$
$$G_5 = {}_{\triangle}A_7A_8A_9,\ G_6 = {}_{\triangle}A_1A_4A_9,\ G_7 = A_1A_2A_3A_4A_1.$$

Вначале вычислим барицентры C_i для каждой из граней G_i, $i = \overline{1,7}$:

$$\overrightarrow{OC_1} = \frac{1}{5}\sum_{i\in\{1,2,5,7,9\}}\overrightarrow{OA_i} = \left[\frac{1}{5}\right.\big(0+4+2+1+0,0+0+10+10+20,$$

$$0+0-4-4-8)\Big] = \overrightarrow{OC_1}\left(\frac{7}{5}, 8, -\frac{16}{5}\right),$$

$$\overrightarrow{OC_2} = \frac{1}{5}\sum_{i\in\{3,4,9,8,6\}} \overrightarrow{OA_i} = \overrightarrow{OC_2}\left(\frac{24}{5}, 8, -\frac{33}{5}\right),$$

$$\overrightarrow{OC_3} = \frac{1}{4}\sum_{i\in\{2,3,6,5\}} \overrightarrow{OA_i} = \overrightarrow{OC_3}\left(\frac{9}{2}, 5, -\frac{7}{2}\right),$$

$$\overrightarrow{OC_4} = \frac{1}{4}\sum_{i\in\{5,6,8,7\}} \overrightarrow{OA_i} = \overrightarrow{OC_4}\left(\frac{11}{4}, 10, -\frac{21}{4}\right), \qquad (2.20)$$

$$\overrightarrow{OC_5} = \frac{1}{3}\sum_{i\in\{7,8,9\}} \overrightarrow{OA_i} = \overrightarrow{OC_5}\left(\frac{5}{3}, \frac{40}{3}, -\frac{19}{3}\right),$$

$$\overrightarrow{OC_6} = \frac{1}{3}\sum_{i\in\{1,4,9\}} \overrightarrow{OA_i} = \overrightarrow{OC_6}\left(\frac{8}{3}, \frac{20}{3}, -\frac{16}{3}\right),$$

$$\overrightarrow{OC_7} = \frac{1}{4}\sum_{i\in\{1,2,3,4\}} \overrightarrow{OA_i} = \overrightarrow{OC_7}(5, 0, -3).$$

Для каждой из этих точек затем вычисляется глубина, которую также будем считать глубиной соответствующей грани. При дальнем наблюдателе это просто скалярное произведение вектора $\vec{s}$ и радиус-вектор точки, взятое со знаком «минус»:

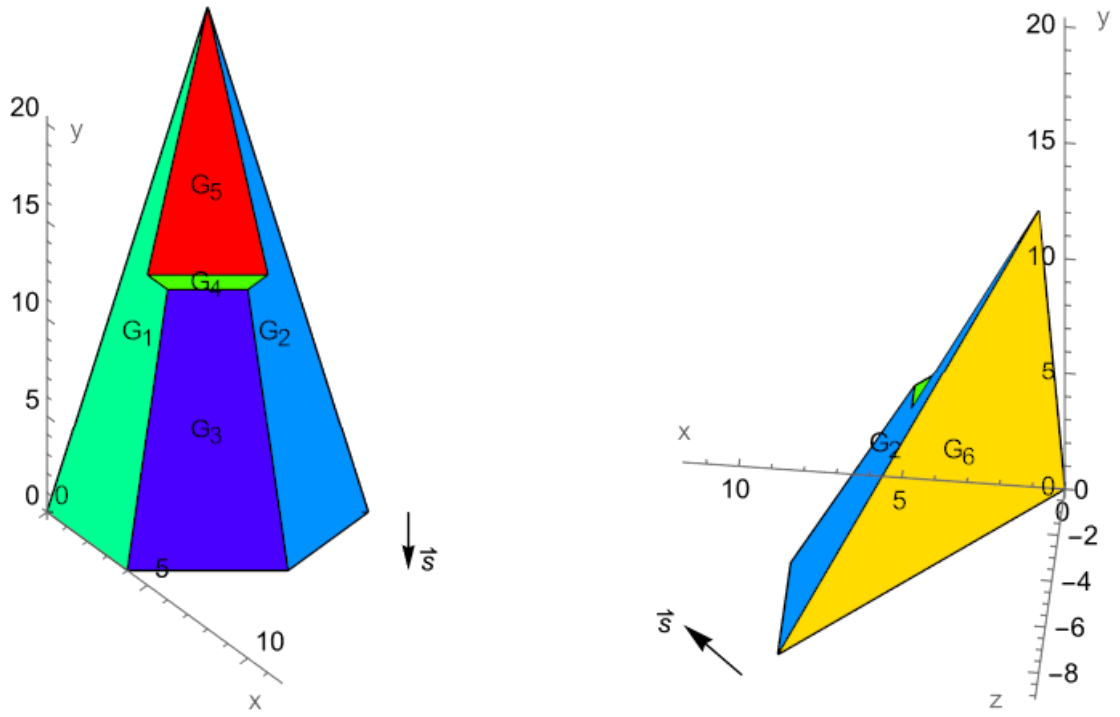
$$d(C_1) = -\vec{s}\cdot\overrightarrow{OC_1} = -\frac{7}{5}+\frac{16}{5} = \frac{9}{5}, d(C_2) = -\vec{s}\cdot\overrightarrow{OC_2} = -\frac{24}{5}+\frac{33}{5} = \frac{9}{5},$$

$$d(C_3) = -\vec{s}\cdot\overrightarrow{OC_3} = -\frac{9}{2}+\frac{7}{2} = -1, d(C_4) = -\vec{s}\cdot\overrightarrow{OC_4} = -\frac{11}{4}+\frac{21}{4} = \frac{5}{2},$$

$$d(C_5) = -\vec{s}\cdot\overrightarrow{OC_5} = -\frac{5}{3}+\frac{19}{3} = \frac{14}{3}, d(C_6) = -\vec{s}\cdot\overrightarrow{OC_6} = -\frac{8}{3}+\frac{16}{3} = \frac{8}{3},$$

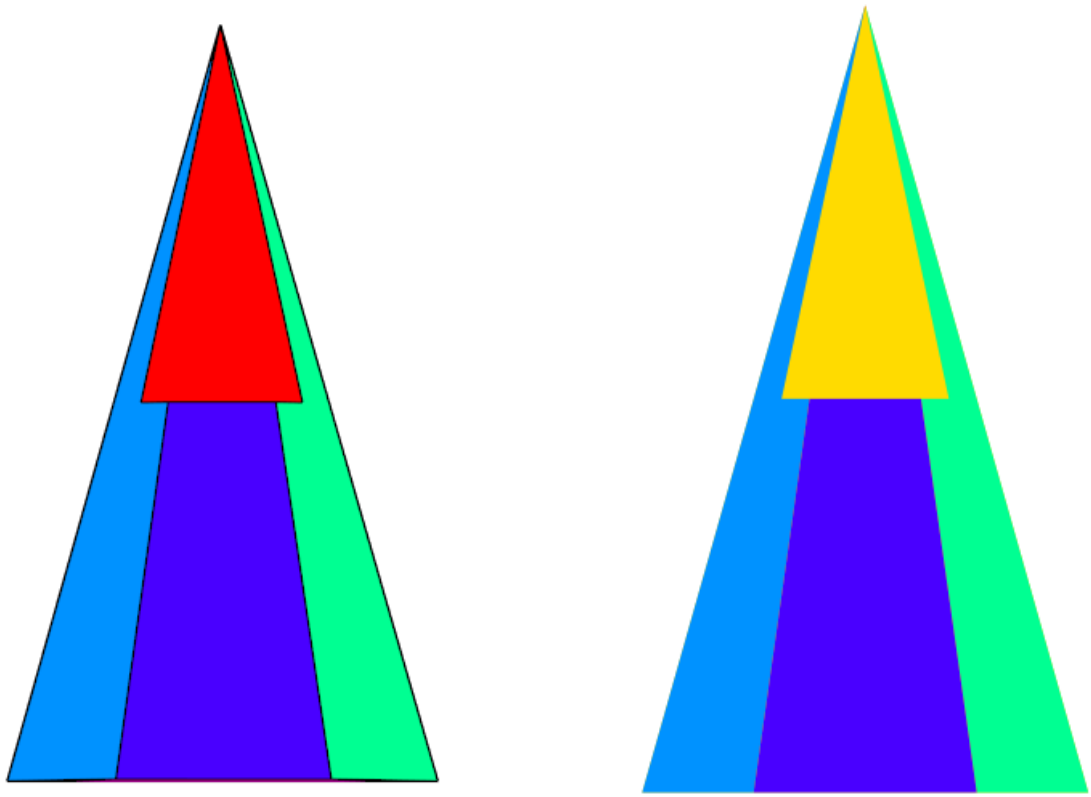
$$d(C_7) = -\vec{s}\cdot\overrightarrow{OC_7} = -5+3 = -2.$$

Располагая точки и соответствующие грани в порядке убывания глубины, приходим к выводу о том, что грани следует отрисовывать в следующем порядке: G_5, G_6, G_4, G_1 вместе с G_2, G_3, G_7. На рисунках 2.33 и 2.34 грани окрашены цветами радуги от красного к синему в порядке убывания их глубин. При этом на рисунке 2.34б задняя грань G_6 (жёлтая) отрисовалась поверх лицевой грани G_5 (красной), что привело к некорректной отрисовке полиэдра. **Поэтому часто в первую очередь определяют лицевые грани, после чего алгоритм художника применяется только к ним!**



а б

Рисунок 2.33 – Передний (а) и заднебоковой (б) виды полиэдра. Основание G_7 является невидимым на обоих чертежах



а б

Рисунок 2.34 – Порядок отрисовки граней по алгоритму художника: а – реально видимые грани, б – изображение проекций граней в порядке убывания их глубин (проекции менее глубоких граней накладываются на проекции более глубоких)

Пример 2.8. Определить порядок отрисовки граней этого же полиэдра при ближнем наблюдателе в точке $S(6, 16, -10)$.

Для сравнения глубин вычислим квадраты длин векторов $\overrightarrow{SC_i}$, $i=\overline{1,7}$, зная координаты точек C_i исходя из вычислений (2.20):

$$\overrightarrow{SC_1}^2=\left(\frac{7}{5}-6\right)^2+(8-16)^2+\left(-\frac{16}{5}+10\right)^2=\frac{657}{5},$$

$$\overrightarrow{SC_2}^2=\left(\frac{24}{5}-6\right)^2+(8-16)^2+\left(-\frac{33}{5}+10\right)^2=77,$$

$$\overrightarrow{SC_3}^2=\left(\frac{9}{2}-6\right)^2+(5-16)^2+\left(-\frac{7}{2}+10\right)^2=\frac{331}{2},$$

$$\overrightarrow{SC_4}^2=\left(\frac{11}{4}-6\right)^2+(10-16)^2+\left(-\frac{21}{4}+10\right)^2=\frac{553}{8},$$

$$\overrightarrow{SC_5}^2=\left(\frac{5}{3}-6\right)^2+\left(\frac{40}{3}-16\right)^2+\left(-\frac{19}{3}+10\right)^2=\frac{118}{3},$$

$$\overrightarrow{SC_6}^2=\left(\frac{8}{3}-6\right)^2+\left(\frac{20}{3}-16\right)^2+\left(-\frac{16}{3}+10\right)^2=120,$$

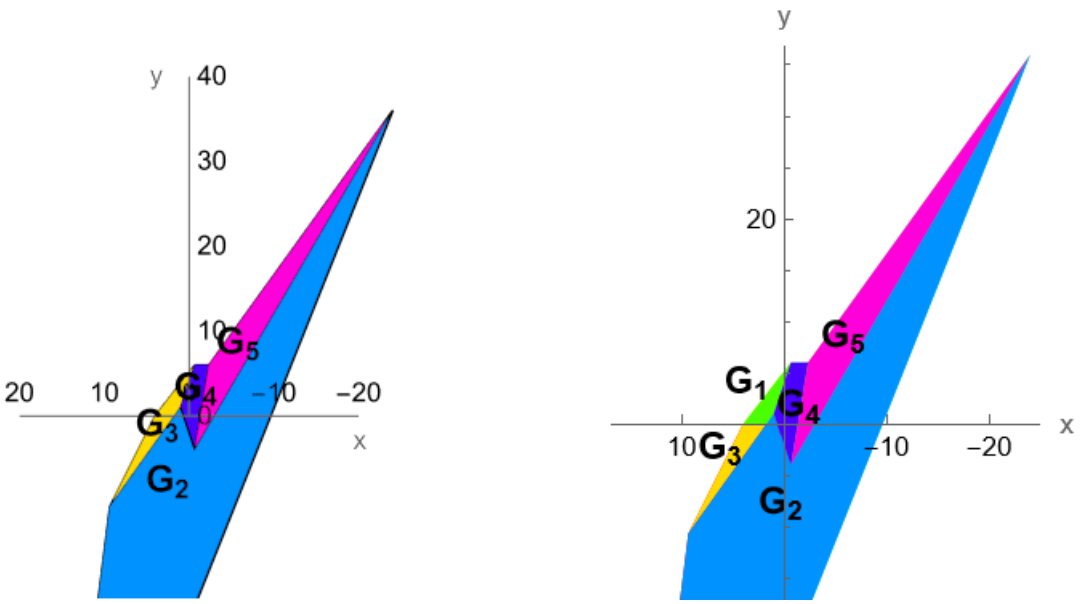
$$\overrightarrow{SC_7}^2=(5-6)^2+(0-16)^2+(-3+10)^2=306.$$

Сортируя эти значения по убыванию, получим следующую последовательность граней: G_7, G_3, G_1, G_6, G_2, G_4, G_5. Реальный вид полиэдра с точки зрения ближнего наблюдателя S и результат отрисовки всех граней (полученный в результате фронтального центрального проектирования всех граней в заданном порядке) представлены на рисунке 2.35. Снова получается, что одна из нелицевых граней (G_1, обозначена зелёным цветом) рисуется поверх одной из лицевых граней (G_3, жёлтая грань).

Пример 2.9. При помощи алгоритма Ньюэлла (стр. 79) определить порядок рисования треугольников $G_i=\triangle A_{i1}A_{i2}A_{i3}$, $i=\overline{1,3}$, при дальнем наблюдателе с вектором $\vec{s}(0,0,1)$, если заданы координаты $A_{ij}(x_{ij}, y_{ij}, z_{ij})$:

$$A_{11}(3,4,0), A_{12}(1,0,3), A_{13}(5,5,0),$$
$$A_{21}(4,2,2), A_{22}(4,4,5), A_{23}(2,2,0),$$
$$A_{31}(5,3,2), A_{32}(0,4,3), A_{33}(1,3,4).$$

Три треугольника обладают следующими габаритными глубинами: $[d_{1\min}, d_{1\max}]=[-3,0]$, $[d_{2\min}, d_{2\max}]=[-5,0]$, $[d_{3\min}, d_{3\max}]=[-4,-2]$. Значит, их



а б

Рисунок 2.35 – Порядок отрисовки граней: а – реально видимые грани, б – отрисовка граней в порядке убывания глубин

сортировка по максимальным глубинам даёт следующий первоначальный список: $D := \{G_1, G_2, G_3\}$ (при равенстве максимальных глубин у нескольких полигонов они сортируются в порядке убывания минимальных глубин).

Теперь для первой грани этого списка, G_1, нужно сформировать список других граней, которые, возможно, частично экранируются ей. Сравнивая габаритные глубины G_1 с глубинами других граней, имеем, что минимальная глубина $d_{1\min} = -3$ меньше, чем обе максимальные глубины $d_{2\max} = 0$ и $d_{3\max} = -2$. Поэтому составляется ещё один список: $D' := \{G_2, G_3\}$.

Обрабатываем теперь список D'. Рассматривая в нём вначале полигон G_2, вычислим габаритные прямоугольники ортографических фронтальных проекций треугольников G_1 и G_2:

$$[x_{1\min}, x_{1\max}] = [1,5], \qquad [x_{2\min}, x_{2\max}] = [2,4],$$
$$[y_{1\min}, y_{1\max}] = [0,5], \qquad [y_{2\min}, y_{2\max}] = [2,4].$$

По этим значениям легко видеть, что габаритные прямоугольники пересекаются. Поэтому проверяются дальнейшие условия.

Следующее условие – расположение треугольника G_2 целиком перед плоскостью G_1. Несложно выписать уравнение плоскости для G_1:

$$\vec{n}_1 = \overrightarrow{A_{11}A_{12}} \times \overrightarrow{A_{12}A_{13}} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -2 & -4 & 3 \\ 4 & 5 & -3 \end{vmatrix} = \vec{n}_1(-3,6,6),$$

$$\Pi_1 : -3(x - x_{11}) + 6(y - y_{11}) + 6(z - z_{11}) = 0 \Rightarrow -x + 2y + 2z - 5 = 0. \qquad (2.21)$$

Аппликата нормального вектора положительна, значит, если при подстановке координат некоторой точки в левую часть уравнения (2.21) получится положительное число, то это будет означать, что эта точка лежит спереди плоскости Π_1; а если результат равен отрицательному числу – точка лежит за этой плоскостью. Проверяем расположение вершин треугольника G_2:

$$\begin{gathered} A_{21} : -4 + 2 \cdot 2 + 2 \cdot 2 - 5 = 8 - 9 = -1 < 0, \\ A_{22} : -4 + 2 \cdot 4 + 2 \cdot 5 - 5 = 9 > 0, \\ A_{23} : -2 + 2 \cdot 2 + 2 \cdot 0 - 5 = -3 < 0. \end{gathered} \qquad (2.22)$$

Таким образом, треугольник G_2 не лежит целиком перед плоскостью треугольника G_1: вершины A_{21} и A_{23} расположены позади плоскости Π_1, а A_{22} – спереди неё. Поэтому дальше следует проверка, лежит ли целиком треугольник G_1 позади плоскости, проведённой через G_2:

$$\vec{n}_2 = \overrightarrow{A_{21}A_{22}} \times \overrightarrow{A_{22}A_{23}} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 2 & 3 \\ -2 & -2 & -5 \end{vmatrix} = \vec{n}_2(-4, -6, 4),$$

$$\Pi_2 : -4(x - x_{21}) - 6(y - y_{21}) + 4(z - z_{21}) = 0 \Rightarrow -2x - 3y + 2z + 10 = 0;$$

$$\begin{gathered} A_{11} : -2 \cdot 3 - 3 \cdot 4 + 2 \cdot 0 + 10 = -6 - 12 + 10 < 0, \\ A_{12} : -2 \cdot 1 - 3 \cdot 0 + 2 \cdot 3 + 10 > 0, \\ A_{13} : -2 \cdot 5 - 3 \cdot 5 + 2 \cdot 0 + 10 < 0. \end{gathered}$$

Отсюда следует, что треугольник G_1 не лежит целиком позади плоскости Π_2.

Последним из проверяемых условий является пересекаемость проекций G_1 и G_2. Для проверки наличия пересечения может быть использована модификация алгоритма Вейлера-Азертона:

- Если хотя бы два ребра разных полигонов пересекаются, то это сразу означает, что два полигона пересекаются. Значит, при нахождении хотя бы одной из пар таких рёбер алгоритм может быть завершён с положительным результатом.

- Если ни одна пара рёбер не пересекается, то либо два полигона не пересекаются, либо один расположен целиком внутри другого. Поэтому затем нужно найти вершины одного полигона, расположенные внутри другого, и наоборот. Если найдена хотя бы одна такая вершина, то снова получаем положительный результат. В противном же случае имеем отсутствие пересечения.

Итак, для полигонов $G_1' = \triangle A_{11}' A_{12}' A_{13}'$ и $G_2' = \triangle A_{21}' A_{22}' A_{23}'$ (где $A_{ij}'(x_{ij}, y_{ij})$ – это ортографическая фронтальная проекция точки A_{ij}) найдём какую-либо пару пересекающихся рёбер. По алгоритму Бентли-Оттманна (см. пример 1.11) имеем следующую последовательность вершин:

$$P := \{A_{12}', A_{23}', A_{11}', A_{21}', A_{22}', A_{13}'\}.$$

Первая вершина является начальной для двух рёбер $A_{11}'A_{12}'$ и $A_{12}'A_{13}'$ первого полигона. Из них ребро $A_{11}'A_{12}'$ выше:

$$S := \{A_{12}'A_{13}', A_{11}'A_{12}'\}.$$

Далее в списке P идёт вершина A_{23}' – начало двух рёбер второго полигона, $A_{22}'A_{23}'$ и $A_{23}'A_{21}'$. Эти два ребра помещаются в структуру S следующим образом:

$$S := \{A_{12}'A_{13}', A_{23}'A_{21}', A_{22}'A_{23}', A_{11}'A_{12}'\}.$$

Отсюда сразу получается, что пересечение двух полигонов непустое, ведь угол $\angle A_{22}'A_{23}'A_{21}'$ находится внутри угла $\angle A_{11}'A_{12}'A_{13}'$. На самом деле точка A_{23}' лежит на ребре $A_{11}'A_{12}'$ (Рисунок 2.36). Таким образом, треугольник G_1, возможно, хотя бы частично экранирует треугольник G_2, и происходит определение ещё одного списка: $D'' := \{G_2\}$.

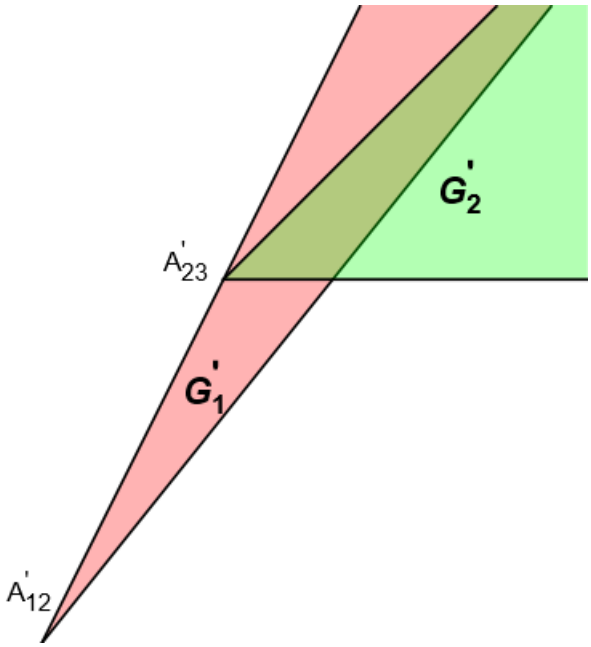


Рисунок 2.36 – Взаимное расположение
двух полигонов на картинной плоскости

При аналогичном рассмотрении полигонов G_1 и G_3 оказывается, что полигон G_3 лежит перед плоскостью Π_1 (точнее, есть вершина, лежащая на этой плоскости, а все остальные расположены перед ней). Значит, G_1 не может экранировать G_3, и помещения этого полигона в список D'' не происходит.

Исходя из всех полученных результатов, имеем, что список D нужно изменить следующим образом: все полигоны из D'' (т.е. треугольник G_2) переносится в начало списка, а полигон G_1 помечается:

$$D := \{G_2, \boxed{G_1}, G_3\}.$$

Теперь первым полигоном в этом списке стал G_2. Согласно алгоритму Ньюэлла, происходит добавление всех помеченных полигонов в список D': $D' := \{G_1\}$. Рассматривая полигоны G_2 и G_1 аналогично представленным выше шагам, получаем следующее:

- габаритные прямоугольники их проекций пересекаются,
- G_1 не находится целиком перед плоскостью Π_2,
- G_2 не находится целиком перед плоскостью Π_1,
- проекции двух полигонов пересекаются.

Таким образом, G_2, возможно, хотя бы частично экранирует G_1. Но полигон G_1 уже помечен, а значит, нужно осуществить разрезание текущего полигона G_2 плоскостью Π_1. Такое разрезание может быть осуществлено, например, по алгоритму со стр. 79.

С учётом вычислений (2.22) имеем, что рёбра $A_{21}A_{22}$ и $A_{22}A_{23}$ пересекают плоскость Π_1. Кроме того, точки этих пересечений $B_{21} = A_{21}A_{22} \cap \Pi_1$ и $B_{22} = A_{22}A_{23} \cap \Pi_1$ легко вычисляются по формуле (1.44), куда подставляются вычисленные значения:

$$\overrightarrow{OB_{21}} = \left|\frac{9}{9+1}\right|\overrightarrow{OA_{21}} + \left|\frac{-1}{9+1}\right|\overrightarrow{OA_{22}} = \frac{9}{10}\overrightarrow{OA_{21}} + \frac{1}{10}\overrightarrow{OA_{22}} = \overrightarrow{OB_{21}}\left(4, \frac{11}{5}, \frac{23}{10}\right),$$

$$\overrightarrow{OB_{22}} = \left|\frac{-3}{-3-9}\right|\overrightarrow{OA_{22}} + \left|\frac{9}{-3-9}\right|\overrightarrow{OA_{23}} = \frac{1}{4}\overrightarrow{OA_{22}} + \frac{3}{4}\overrightarrow{OA_{23}} = \overrightarrow{OB_{22}}\left(\frac{5}{2}, \frac{5}{2}, \frac{5}{4}\right).$$

Значит, имеем новые два полигона: $G_{21} = A_{21}B_{21}B_{22}A_{23}A_{21}$ – часть полигона G_2, расположенную позади плоскости Π_1, и $G_{22} = \triangle A_{22}B_{22}B_{21}$ – часть полигона G_2, что расположена спереди этой же плоскости. Задний полигон G_{21} записывается в D вместо исходного полигона G_2. Передний полигон G_{22} нужно записать где-то после помеченного полигона G_1 таким образом, чтобы все полигоны после G_1 (а в данном примере таковых только два: G_{22} и G_3)

следовали в порядке убывания своих максимальных глубин. Для G_{22} имеем $d_{22\max} = -5/4$, значит, в списке D этот полигон должен идти перед G_3. Таким образом, после разрезания полигона G_2 получается следующий список:

$$D := \{G_{21}, \boxed{G_1}, G_{22}, G_3\}.$$

Здесь первым полигоном уже является G_{21}. Среди полигонов, которые потенциально могут быть экранированы им, только G_1. Но мы уже получили, что G_{21} расположен целиком позади этого полигона, а значит, для G_{21} вообще не остаётся полигонов, которые он мог бы экранировать. Итак, G_{21} – первый из полигонов, подлежащий рисованию. Далее идёт помеченный полигон G_1, который также может быть немедленно отрисован. После этого два полигона удаляются из D:

$$D := \{G_{22}, G_3\}.$$

Рассматривая далее следующий полигон G_{22}, имеем, что его минимальная глубина $d_{22\min} = -5$ меньше, чем максимальная глубина полигона G_3, равная –2, значит, G_3 помещается в новый список D': $D' := \{G_3\}$.

Проверка всех условий даёт следующие результаты:

- Габаритные прямоугольники проекций обоих полигонов пересекаются.
- G_3 не расположен целиком спереди плоскости Π_2:

$$\begin{gathered} \Pi_2 : -2x - 3y + 2z + 10 = 0; \\ A_{31} : -2 \cdot 5 - 3 \cdot 3 + 2 \cdot 2 + 10 = -5 < 0, \\ A_{32} : -2 \cdot 0 - 3 \cdot 4 + 2 \cdot 3 + 10 = 4 > 0, \\ A_{33} : -2 \cdot 1 - 3 \cdot 3 + 2 \cdot 4 + 10 = 7 > 0. \end{gathered} \tag{2.23}$$

- G_{22} не расположен целиком позади плоскости Π_3, проходящей через G_3:

$$\vec{n}_3 = \overrightarrow{A_{31}A_{32}} \times \overrightarrow{A_{32}A_{33}} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -5 & 1 & 1 \\ 1 & -1 & 1 \end{vmatrix} = \vec{n}_3(2,6,4),$$

$$\Pi_3 : 2(x - x_{31}) + 6(y - y_{31}) + 4(z - z_{31}) = 0 \Rightarrow x + 3y + 2z - 18 = 0;$$

$$\begin{gathered} A_{21} : 4 + 3 \cdot 2 + 2 \cdot 2 - 18 < 0, \\ A_{22} : 4 + 3 \cdot 4 + 2 \cdot 5 - 18 > 0, \\ A_{23} : 2 + 3 \cdot 2 + 2 \cdot 0 - 18 < 0. \end{gathered}$$

- Проекции обоих полигонов пересекаются.

Это всё означает, что G_3 нужно перенести в начало D, а G_{22} помечается:

$$D := \{G_3, \boxed{G_{22}}\}.$$

Аналогично выясняется, что все указанные выше свойства (только уже проверяется, находится ли G_3 позади Π_2, а G_{22} – спереди Π_3) имеют место при рассмотрении полигона G_3, ставшего первым в списке D, и полигона G_{22}, попавшим в список $D' := \{G_{22}\}$. А с учётом того, что G_{22} теперь помеченный, нужно делать разрезание полигона G_3 плоскостью Π_2. Используя результаты из (2.23), имеем $A_{31}A_{32} \cap \Pi_2 = B_{31}\left(\frac{20}{9}, \frac{32}{9}, \frac{23}{9}\right)$, $A_{33}A_{31} \cap \Pi_2 = B_{32}\left(\frac{10}{3}, 3, \frac{17}{6}\right)$, и полигон G_3 разбивается на две части: $G_{31} = {}_{\triangle}A_{31}B_{31}B_{32}$ – заднюю – и $G_{32} = A_{32}A_{33}B_{32}B_{31}A_{32}$ – переднюю. Список D обретёт следующий вид:

$$D := \{G_{31}, \boxed{G_{22}}, G_{32}\}.$$

С учётом того, что после разбиения полигон G_{31} лежит целиком позади G_{22}, а G_{32} – целиком спереди него, получается, что все полигоны из D могут быть отрисованы в этом порядке без искажений. Итак, окончательно получили следующую последовательность полигонов для отрисовки:

$$\mathcal{G} = \{G_{21}, G_1, G_{31}, G_{22}, G_{32}\}.$$

Порядок отрисовки полигонов и их частей представлен на рисунке 2.37.

2.2.5. Задачи

1. По алгоритму Робертса определить видимость граней выпуклого полиэдра при заданном наблюдателе.

1.1. Тетраэдр с вершинами в точках $A_1(7,3,7)$, $A_2(3,0,6)$, $A_3(4,2,5)$, $A_4(1,9,7)$ при дальнем наблюдателе $\vec{s}(-4,3,-5)$ (перейти к ответу).

1.2. Тетраэдр с вершинами в точках $A_1(4,6,7)$, $A_2(6,6,4)$, $A_3(0,0,4)$, $A_4(1,9,10)$ при дальнем наблюдателе $\vec{s}(-4,4,3)$ (перейти к ответу).

1.3. Тетраэдр с вершинами в точках $A_1(5,4,2)$, $A_2(4,5,2)$, $A_3(5,5,3)$, $A_4(4,4,5)$ при дальнем наблюдателе $\vec{s}(-1,2,5)$ (перейти к ответу).

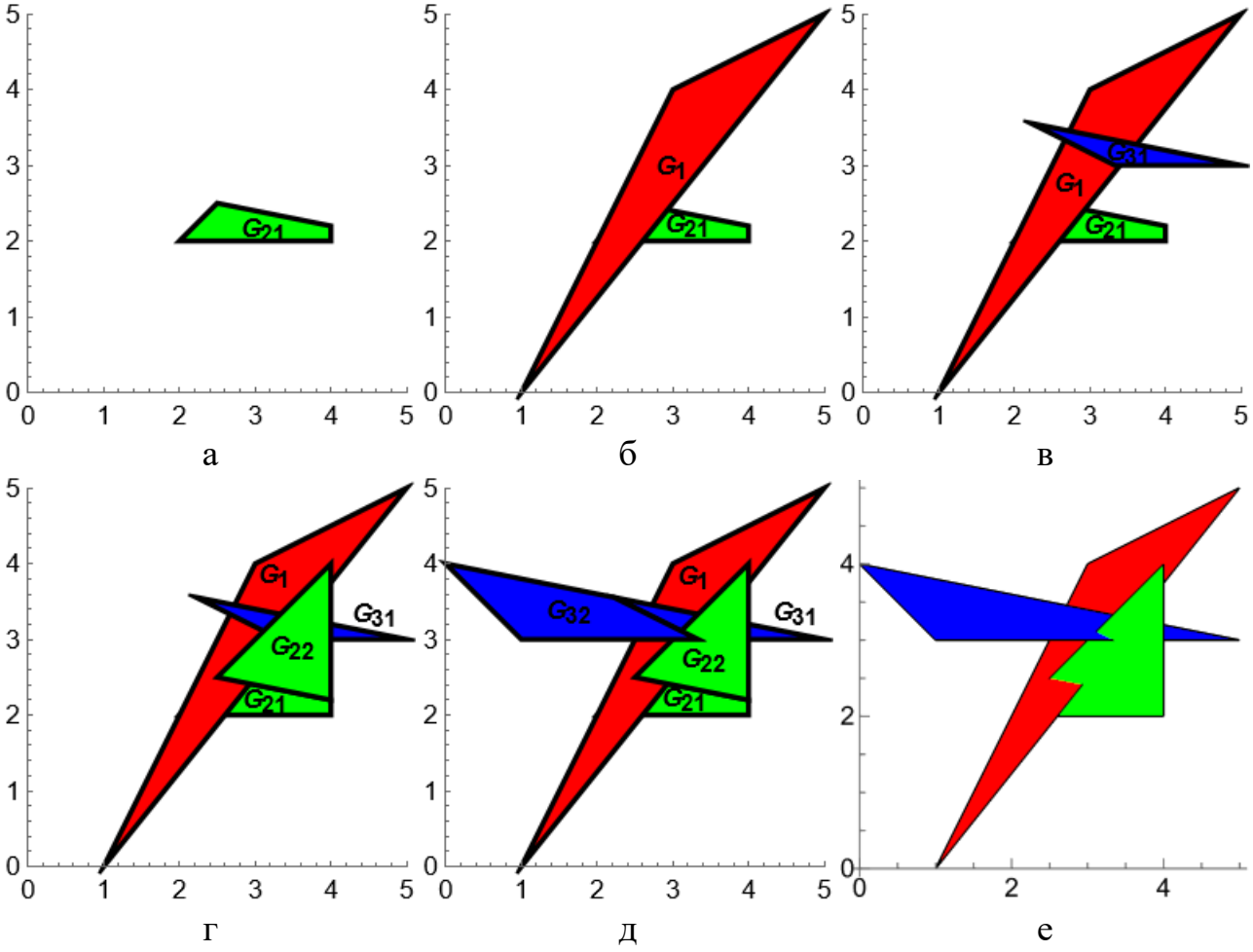


Рисунок 2.37 – Порядок отрисовки полигонов и их частей (а-д)
и окончательное изображение сцены (е)

1.4. Тетраэдр с вершинами в точках $A_1(7,9,8)$, $A_2(9,0,4)$, $A_3(3,7,10)$, $A_4(0,6,5)$ при дальнем наблюдателе $\vec{s}(2,-1,-5)$ (перейти к ответу).

1.5. Тетраэдр с вершинами в точках $A_1(0,5,0)$, $A_2(2,0,9)$, $A_3(2,6,8)$, $A_4(5,4,8)$ при дальнем наблюдателе $\vec{s}(-4,3,3)$ (перейти к ответу).

1.6. Тетраэдр с вершинами в точках $A_1(3,0,6)$, $A_2(5,0,1)$, $A_3(1,1,1)$, $A_4(4,5,5)$ при дальнем наблюдателе $\vec{s}(1,5,3)$ (перейти к ответу).

1.7. Тетраэдр с вершинами в точках $A_1(6,10,6)$, $A_2(8,5,5)$, $A_3(8,7,7)$, $A_4(0,2,3)$ при ближнем наблюдателе $S(3, 0, -3)$ (перейти к ответу).

1.8. Тетраэдр с вершинами в точках $A_1(2,2,3)$, $A_2(1,2,2)$, $A_3(6,8,9)$, $A_4(10,7,12)$ при ближнем наблюдателе $S(-5, -2, -3)$ (перейти к ответу).

1.9. Тетраэдр с вершинами в точках $A_1(9,4,2)$, $A_2(3,1,8)$, $A_3(5,9,2)$, $A_4(9,5,10)$ при ближнем наблюдателе $S(-4, -4, -5)$ (перейти к ответу).

1.10. Тетраэдр с вершинами в точках $A_1(0,4,2)$, $A_2(7,7,5)$, $A_3(9,3,8)$, $A_4(6,0,6)$ при ближнем наблюдателе $S(-3, 2, -4)$ (перейти к ответу).

1.11. Тетраэдр с вершинами в точках $A_1(6,0,8)$, $A_2(1,4,5)$, $A_3(6,2,0)$, $A_4(1,3,3)$ при ближнем наблюдателе $S(-5, -5, -1)$ (перейти к ответу).

1.12. Тетраэдр с вершинами в точках $A_1(2,10,5)$, $A_2(5,9,4)$, $A_3(6,1,0)$, $A_4(7,4,8)$ при ближнем наблюдателе $S(1, -5, 2)$ (перейти к ответу).

1.13. Полиэдр с вершинами в точках $A_1(0,0,0)$, $A_2(3,0,0)$, $A_3(3,0,3)$, $A_4(0,0,3)$, $A_5(0,2,0)$, $A_6(2,1,0)$, $A_7(0,2,2)$ и гранями $A_1A_2A_3A_4A_1$, $A_1A_5A_7A_4A_1$, $A_1A_5A_6A_2A_1$, $\triangle A_2A_6A_3$, $\triangle A_7A_6A_3$, $\triangle A_7A_3A_4$, $\triangle A_7A_5A_6$ при дальнем наблюдателе $\vec{s}(2,-5,-5)$ (перейти к ответу).

1.14. Тот же самый полиэдр при ближнем наблюдателе $S(1, 2, 5)$ (перейти к ответу).

1.15. Полиэдр с вершинами в точках $A_1(2,0,7)$, $A_2(4,5,8)$, $A_3(7,2,7)$, $A_4(0,10,9)$, $A_5(0,4,3)$, $A_6(9,7,8)$ и гранями $\triangle A_6A_5A_4$, $\triangle A_5A_1A_4$, $\triangle A_3A_5A_6$, $\triangle A_5A_3A_1$, $\triangle A_2A_6A_4$, $\triangle A_1A_2A_4$, $A_1A_3A_6A_2A_1$ при дальнем наблюдателе $\vec{s}(2,2,-3)$ (перейти к ответу).

1.16. Полиэдр с вершинами в точках $A_1(0,5,2)$, $A_2(2,4,6)$, $A_3(6,0,2)$, $A_4(2,3,8)$, $A_5(9,4,5)$, $A_6(-4,-5,2)$ с гранями $\triangle A_4A_1A_6$, $\triangle A_1A_3A_6$, $\triangle A_3A_1A_5$, $A_5A_4A_6A_3$, $\triangle A_2A_4A_5$, $\triangle A_1A_2A_5$, $\triangle A_4A_2A_1$ при ближнем наблюдателе $S(-4, 1, -4)$ (перейти к ответу).

2. Вычислить матрицу тел для выпуклых полиэдров (перейти к ответам).

2.1. Тетраэдр с вершинами в точках $A_1(2,3,1)$, $A_2(0,5,5)$, $A_3(4,0,0)$, $A_4(1,3,4)$.

2.2. Тетраэдр с вершинами в точках $A_1(0,5,3)$, $A_2(3,-4,5)$, $A_3(-2,1,0)$, $A_4(3,-4,1)$.

2.3. Тетраэдр с вершинами в точках $A_1(-2,4,-4)$, $A_2(5,4,3)$, $A_3(2,-1,-5)$, $A_4(5,-5,-3)$.

2.4. Тетраэдр с вершинами в точках $A_1(1,-1,-3)$, $A_2(3,-1,0)$, $A_3(-5,-3,-4)$, $A_4(-2,0,-3)$.

2.5. Тетраэдр с вершинами в точках $A_1(1,1,2)$, $A_2(-3,4,1)$, $A_3(3,-5,-5)$, $A_4(5,1,3)$.

2.6. Тетраэдр с вершинами в точках $A_1(0,1,1)$, $A_2(5,-4,3)$, $A_3(0,3,4)$, $A_4(1,2,4)$.

2.7. Полиэдр с вершинами в точках $A_1(0,5,1)$, $A_2(0,1,1)$, $A_3(5,5,0)$, $A_4(3,0,2)$, $A_5(1,1,2)$ и гранями $\triangle A_4A_3A_1$, $\triangle A_3A_2A_1$, $\triangle A_4A_2A_3$, $\triangle A_5A_4A_1$, $\triangle A_2A_5A_1$, $\triangle A_5A_2A_4$.

2.8. Полиэдр с вершинами в точках $A_1(0,2,3)$, $A_2(1,5,4)$, $A_3(0,3,4)$, $A_4(2,5,5)$, $A_5(2,5,3)$ и гранями $\triangle A_4A_3A_1$, $\triangle A_3A_2A_1$, $\triangle A_4A_2A_3$, $\triangle A_5A_4A_1$, $\triangle A_2A_5A_1$, $\triangle A_5A_2A_4$.

2.9. Полиэдр с вершинами в точках $A_1(0,2,2)$, $A_2(3,4,4)$, $A_3(0,5,2)$, $A_4(1,0,5)$, $A_5(5,3,5)$, $A_6(-1,1,4)$ и гранями $\triangle A_3A_1A_6$, $\triangle A_1A_3A_5$, $\triangle A_1A_4A_6$, $\triangle A_4A_1A_5$, $\triangle A_2A_3A_6$, $\triangle A_3A_2A_5$, $A_4A_5A_2A_6A_4$.

3. Осуществить отсечение отрезка AB выпуклым телом с заданной матрицей $\mathcal{C}$.

3.1. $A(3, 1, 4)$, $B(-10, 9, 9)$, $\mathcal{C} = \begin{pmatrix} 0 & 1 & 2 & -3 \\ 1 & -1 & -2 & 2 \\ -4 & 2 & 7 & 1 \\ 1 & -2 & -7 & 11 \end{pmatrix}$ (перейти к ответу),

3.2. $A(3, 4, 1)$, $B(4, 0, 2)$, $\mathcal{C} = \begin{pmatrix} -1 & 2 & 0 & -3 \\ -1 & -4 & -2 & 23 \\ 1 & -4 & 2 & 9 \\ 1 & 2 & 0 & -9 \end{pmatrix}$ (перейти к ответу),

3.3. $A(-4, 1, 1)$, $B(0, 3, -1)$, $\mathcal{C} = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 1 & 0 & 2 & 3 \\ -7 & -6 & -8 & -3 \\ -11 & 12 & -4 & -9 \end{pmatrix}$ (перейти к ответу),

3.4. $A(2, 1, -1)$, $B(-1, 0, -2)$, $\mathcal{C} = \begin{pmatrix} 0 & 1 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 2 & -5 & -1 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}$ (перейти к ответу),

3.5. $A(-2, 2, 1)$, $B(2, -3, 1)$, $\mathcal{C} = \begin{pmatrix} -1 & -2 & -2 & 0 \\ 5 & -2 & 4 & -6 \\ -4 & 1 & -2 & 6 \\ 2 & 1 & 1 & 0 \end{pmatrix}$ (перейти к ответу),

3.6. $A(0, 3, 0)$, $B(-3, 2, 2)$, $\mathcal{C} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ -3 & -2 & 5 & -5 \\ 2 & -1 & -1 & 8 \\ -3 & -2 & -2 & 2 \end{pmatrix}$ (перейти к ответу).

4. Осуществить удаление части отрезка AB, экранируемой выпуклым телом с заданной матрицей $\mathcal{C}$.

4.1. $A(-4, -2, -3)$, $B(1, 4, -3)$, $\mathcal{C} = \begin{pmatrix} -5 & 2 & 4 & -2 \\ 4 & -2 & -3 & 3 \\ 4 & -1 & -2 & -2 \\ -10 & 4 & 5 & 5 \end{pmatrix}$ при дальнем наблюдателе $\vec{s}(5, -2, 2)$ (перейти к ответу),

4.2. $A(0, 4, 0)$, $B(2, 0, 4)$, $C = \begin{pmatrix} 1 & -2 & 2 & 0 \\ -1 & 1 & -2 & 3 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 0 & -3 \end{pmatrix}$ при дальнем наблюдателе $\vec{s}(5,0,2)$ (перейти к ответу),

4.3. $A(1, 2, 3)$, $B(-2, 1, 0)$, $C = \begin{pmatrix} 0 & -1 & -4 & 20 \\ 7 & -1 & -4 & -1 \\ -7 & -2 & 6 & 5 \\ 0 & 2 & 1 & -5 \end{pmatrix}$ при дальнем наблюдателе $\vec{s}(5,-3,4)$ (перейти к ответу),

4.4. $A(-3, 5, 0)$, $B(4, -1, 4)$, $C = \begin{pmatrix} 0 & 2 & -1 & 1 \\ -8 & 6 & 7 & -7 \\ 4 & -6 & 1 & -1 \\ 0 & -2 & -1 & 9 \end{pmatrix}$ при дальнем наблюдателе $\vec{s}(-4,-5,-3)$ (перейти к ответу),

4.5. $A(-1, 3, 2)$, $B(2, 0, 2)$, $C = \begin{pmatrix} -4 & 16 & -3 & 3 \\ -4 & -4 & -3 & 23 \\ 1 & -4 & 2 & -2 \\ 2 & 2 & -1 & 1 \end{pmatrix}$ при ближнем наблюдателе $S(-1, -3, -1)$ (перейти к ответу),

4.6. $A(1, 5, 2)$, $B(2, 0, 2)$, $C = \begin{pmatrix} 1 & 3 & 3 & -18 \\ 0 & -1 & -1 & 6 \\ 1 & 3 & 1 & -14 \\ -1 & -2 & -1 & 11 \end{pmatrix}$ при ближнем наблюдателе $S(-1, -4, -1)$ (перейти к ответу).

5. Используя алгоритм художника, определить порядок отрисовки **лицевых** граней невыпуклого полиэдра с вершинами A_i при заданном наблюдателе, если для каждой грани G_j известна её внешняя нормаль $\vec{n}_j$.

5.1. $A_1(0,1,1)$, $A_2(0,4,2)$, $A_3(0,2,5)$, $A_4(0,3,3)$, $A_5(4,3,4)$; $G_1 = A_1A_2A_3A_4A_1$, $G_2 = \triangle A_1A_2A_5$, $G_3 = \triangle A_2A_3A_5$, $G_4 = \triangle A_3A_4A_5$, $G_5 = \triangle A_1A_4A_5$; $\vec{n}_1(-1,0,0)$, $\vec{n}_2(7,4,-12)$, $\vec{n}_3(-1,12,8)$, $\vec{n}_4(1,-8,-4)$, $\vec{n}_5(-1,-4,4)$ при дальнем наблюдателе $\vec{s}(-3,-1,5)$ (перейти к ответу).

5.2. $A_1(2,-2,2)$, $A_2(10,1,7)$, $A_3(7,5,6)$, $A_4(7,2,3)$, $A_5(9,8,1)$; $G_1 = \triangle A_2A_4A_5$, $G_2 = \triangle A_2A_3A_5$, $G_3 = \triangle A_3A_4A_5$, $G_4 = \triangle A_1A_2A_4$, $G_5 = \triangle A_1A_3A_4$, $G_6 = \triangle A_1A_2A_3$;

$\vec{n}_1(11,-7,-10)$, $\vec{n}_2(1,1,1)$, $\vec{n}_3(-4,1,-1)$, $\vec{n}_4(1,-1,-1)$, $\vec{n}_5(-3,5,-5)$, $\vec{n}_6(-23,-7,41)$ при дальнем наблюдателе $\vec{s}(5,-1,-10)$ (перейти к ответу).

5.3. $A_1(6,7,1)$, $A_2(1,6,9)$, $A_3(4,-2,5)$, $A_4(5,6,2)$, $A_5(4,4,9)$, $A_6(3,4,8)$, $A_7(9,5,4)$; $G_1 = \triangle A_3A_5A_7$, $G_2 = \triangle A_2A_3A_5$, $G_3 = \triangle A_2A_5A_6$, $G_4 = \triangle A_5A_6A_7$, $G_5 = \triangle A_1A_3A_7$, $G_6 = \triangle A_1A_6A_7$, $G_7 = \triangle A_1A_4A_6$, $G_8 = \triangle A_1A_3A_4$, $G_9 = \triangle A_3A_4A_6$, $G_{10} = \triangle A_2A_3A_6$; $\vec{n}_1(17,-10,15)$, $\vec{n}_2(-4,-6,9)$, $\vec{n}_3(2,3,-2)$, $\vec{n}_4(-1,10,1)$, $\vec{n}_5(19,-18,-31)$, $\vec{n}_6(1,6,3)$, $\vec{n}_7(-1,1,0)$, $\vec{n}_8(-5,-2,-7)$, $\vec{n}_9(-3,0,-1)$, $\vec{n}_{10}(0,1,-2)$ при дальнем наблюдателе $\vec{s}(1,0,-3)$ (перейти к ответу).

5.4. $A_1(2,3,4)$, $A_2(7,3,8)$, $A_3(2,1,9)$, $A_4(9,7,3)$, $A_5(9,4,4)$, $A_6(6,1,5)$, $A_7(7,10,2)$; $G_1 = \triangle A_1A_2A_3$, $G_2 = \triangle A_2A_3A_6$, $G_3 = \triangle A_2A_5A_6$, $G_4 = \triangle A_2A_4A_5$, $G_5 = \triangle A_2A_4A_7$, $G_6 = \triangle A_1A_2A_7$, $G_7 = A_4A_5A_6A_7A_4$, $G_8 = \triangle A_1A_6A_7$, $G_9 = \triangle A_1A_3A_6$; $\vec{n}_1(-8,25,10)$, $\vec{n}_2(1,-2,1)$, $\vec{n}_3(11,-10,3)$, $\vec{n}_4(11,2,6)$, $\vec{n}_5(11,12,14)$, $\vec{n}_6(-28,30,35)$, $\vec{n}_7(0,-1,-3)$, $\vec{n}_8(3,-13,-38)$, $\vec{n}_9(-2,-5,-2)$ при дальнем наблюдателе $\vec{s}(-1,-5,5)$ (перейти к ответу).

5.5. $A_1(1,0,2)$, $A_2(1,0,5)$, $A_3(2,0,4)$, $A_4(2,0,3)$, $A_5(4,2,1)$, $A_6(3,-1,3)$; $G_1 = \triangle A_1A_2A_5$, $G_2 = \triangle A_1A_4A_5$, $G_3 = \triangle A_3A_4A_5$, $G_4 = \triangle A_2A_3A_5$, $G_5 = \triangle A_1A_2A_6$, $G_6 = \triangle A_2A_3A_6$, $G_7 = \triangle A_3A_4A_6$, $G_8 = \triangle A_1A_4A_6$; $\vec{n}_1(-2,3,0)$, $\vec{n}_2(1,-2,-1)$, $\vec{n}_3(1,-1,0)$, $\vec{n}_4(2,1,2)$, $\vec{n}_5(-1,-2,0)$, $\vec{n}_6(1,0,1)$, $\vec{n}_7(1,1,0)$, $\vec{n}_8(1,1,-1)$ при ближнем наблюдателе $S(1, -3, -3)$ (перейти к ответу).

5.6. $A_1(4,1,1)$, $A_2(9,1,9)$, $A_3(0,8,3)$, $A_4(6,1,10)$, $A_5(0,1,9)$, $A_6(6,1,7)$; $G_1 = \triangle A_1A_2A_3$, $G_2 = \triangle A_2A_3A_6$, $G_3 = \triangle A_3A_4A_6$, $G_4 = \triangle A_3A_4A_5$, $G_5 = A_2A_1A_5A_4A_6A_2$, $G_6 = \triangle A_1A_3A_5$; $\vec{n}_1(8,6,-5)$, $\vec{n}_2(-2,0,3)$, $\vec{n}_3(7,6,0)$, $\vec{n}_4(-7,36,42)$, $\vec{n}_5(0,-1,0)$, $\vec{n}_6(-14,-6,-7)$ при ближнем наблюдателе $S(2, -3, -3)$ (перейти к ответу).

6. Используя алгоритм Ньюэлла, определить порядок рисования проекций заданных полигонов трёхмерного пространства и их частей при дальнем наблюдателе $\vec{s}(0,0,1)$.

6.1. Три треугольника: $G_i = \triangle A_{i1}A_{i2}A_{i3}$, $i = \overline{1,3}$;
$A_{11}(2,5,0)$, $A_{12}(1,2,2)$, $A_{13}(1,1,0)$,
$A_{21}(4,5,3)$, $A_{22}(3,3,3)$, $A_{23}(0,0,4)$,
$A_{31}(5,2,1)$, $A_{32}(2,1,0)$, $A_{33}(4,4,2)$
(перейти к ответу),

6.2. Три треугольника: $G_i = \triangle A_{i1}A_{i2}A_{i3}$, $i = \overline{1,3}$;
$A_{11}(2,2,2)$, $A_{12}(1,1,5)$, $A_{13}(0,1,2)$,
$A_{21}(4,1,2)$, $A_{22}(4,5,2)$, $A_{23}(3,4,3)$,
$A_{31}(0,4,5)$, $A_{32}(5,2,0)$, $A_{33}(3,5,5)$
(перейти к ответу),

6.3. Три треугольника: $G_i = \triangle A_{i1}A_{i2}A_{i3},\ i = \overline{1,3}$;
$A_{11}(2,3,1)$, $A_{12}(4,1,4)$, $A_{13}(4,2,2)$,
$A_{21}(0,3,5)$, $A_{22}(2,4,5)$, $A_{23}(0,2,3)$,
$A_{31}(0,1,3)$, $A_{32}(4,1,0)$, $A_{33}(4,5,5)$
(перейти к ответу),

6.4. Три треугольника: $G_i = \triangle A_{i1}A_{i2}A_{i3},\ i = \overline{1,3}$;
$A_{11}(3,1,1)$, $A_{12}(0,3,4)$, $A_{13}(3,4,1)$,
$A_{21}(1,1,4)$, $A_{22}(0,4,0)$, $A_{23}(3,3,2)$,
$A_{31}(5,5,5)$, $A_{32}(2,2,5)$, $A_{33}(0,3,3)$
(перейти к ответу),

6.5. Четыре треугольника: $G_i = \triangle A_{i1}A_{i2}A_{i3},\ i = \overline{1,4}$;
$A_{11}(4,1,1)$, $A_{12}(3,4,2)$, $A_{13}(1,1,0)$,
$A_{21}(0,2,3)$, $A_{22}(2,2,0)$, $A_{23}(0,4,5)$,
$A_{31}(5,0,2)$, $A_{32}(4,3,2)$, $A_{33}(4,2,4)$,
$A_{41}(4,3,0)$, $A_{42}(1,2,2)$, $A_{43}(2,0,0)$
(перейти к ответу).

2.3. Базовые растровые алгоритмы

2.3.1. Алгоритм DDA-линии

Интуитивно понятный алгоритм DDA-линии (стр. 106) позволяет осуществить растеризацию отрезка с концами в точках с действительными координатами.

Пример 3.1. Осуществить растеризацию отрезка AB, концы которого имеют координаты $A(9.24, 0.64)$ и $B(0.16, 4.97)$.

Способ 1. Концы искомого отрезка легко получить, округляя координаты точек A и B: $(x_{\text{нач}}, y_{\text{нач}}) = (9,1)$, $(x_{\text{кон}}, y_{\text{кон}}) = (0,5)$. Далее воспользуемся **симметричной** реализацией алгоритма и формулами (1.53) и (1.55):

$$L = \max\left\{\left|0-9\right|, \left|5-1\right|\right\} = 9.$$

Значит, итоговая растеризация состоит из $L + 1 = 10$ пикселей. Полагая в качестве начальных значений $x := x_A = 9.24$, $y := y_A = 0.64$, будем вычислять координаты $L - 1$ промежуточного узла исходного отрезка, подлежащие последующему округлению:

$$\frac{x_B - x_A}{L} = \frac{0.16 - 9.24}{9} \approx -1.009, \frac{y_B - y_A}{L} = \frac{4.97 - 0.64}{9} \approx 0.481,$$

$$x := x + \frac{x_B - x_A}{L} = 9.24 - 1.009 = 8.231, y := y + \frac{y_B - y_A}{L} = 0.64 + 0.481 = 1.121,$$

$$(x_1, y_1) = (\text{round}(8.231), \text{round}(1.121)) = (8,1),$$

$$x := x + \frac{x_B - x_A}{L} = 8.231 - 1.009 = 7.222, y := y + \frac{y_B - y_A}{L} = 1.121 + 0.481 = 1.602,$$

$$(x_2, y_2) = (\text{round}(7.222), \text{round}(1.602)) = (7,2),$$

$$x := x + \frac{x_B - x_A}{L} = 7.222 - 1.009 = 6.213, y := y + \frac{y_B - y_A}{L} = 1.602 + 0.481 = 2.083,$$

$$(x_3, y_3) = (\text{round}(6.213), \text{round}(2.083)) = (6,2),$$

$$x := x + \frac{x_B - x_A}{L} = 6.213 - 1.009 = 5.204, y := y + \frac{y_B - y_A}{L} = 2.083 + 0.481 = 2.564,$$

$$(x_4, y_4) = (\text{round}(5.204), \text{round}(2.564)) = (5,3),$$

$$x := x + \frac{x_B - x_A}{L} = 5.204 - 1.009 = 4.195, y := y + \frac{y_B - y_A}{L} = 2.564 + 0.481 = 3.045,$$

$$(x_5, y_5) = (\text{round}(4.195), \text{round}(3.045)) = (4,3),$$

$$x := x + \frac{x_B - x_A}{L} = 4.195 - 1.009 = 3.186, y := y + \frac{y_B - y_A}{L} = 3.045 + 0.481 = 3.526,$$

$$(x_6, y_6) = (\text{round}(3.186), \text{round}(3.526)) = (3,4),$$

$$x := x + \frac{x_B - x_A}{L} = 3.186 - 1.009 = 2.177, y := y + \frac{y_B - y_A}{L} = 3.526 + 0.481 = 4.007,$$

$$(x_7, y_7) = (\text{round}(2.177), \text{round}(4.007)) = (2,4),$$

$$x := x + \frac{x_B - x_A}{L} = 2.177 - 1.009 = 1.168, y := y + \frac{y_B - y_A}{L} = 4.007 + 0.481 = 4.488,$$

$$(x_8, y_8) = (\text{round}(1.168), \text{round}(4.488)) = (1,4),$$

а координаты последнего пикселя равны $(x_9, y_9) = (x_{\text{кон}}, y_{\text{кон}}) = (0,5)$.

По **несимметричной** же реализации, в которой используются формулы (1.57), вычисляющие промежуточные узлы отрезка с концами в точках с целочисленными координатами, имеют место следующие вычисления:

$$x := x_{\text{нач}} = 9, y := y_{\text{нач}} = 1,$$

$$\frac{x_{\text{кон}} - x_{\text{нач}}}{L} = \frac{0 - 9}{9} = -1, \frac{y_{\text{кон}} - y_{\text{нач}}}{L} = \frac{5 - 1}{9} \approx 0.444,$$

$$x := x + \frac{x_{\text{кон}} - x_{\text{нач}}}{L} = 9 - 1 = 8, y := y + \frac{y_{\text{кон}} - y_{\text{нач}}}{L} = 1 + 0.444 = 1.444,$$

$$(x_1, y_1) = (8, \text{round}(1.444)) = (8,1),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 8 - 1 = 7, y := y + \frac{y_{кон} - y_{нач}}{L} = 1.444 + 0.444 = 1.888,$$

$$(x_2, y_2) = (7, \text{round}(1.888)) = (7, 2),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 7 - 1 = 6, y := y + \frac{y_{кон} - y_{нач}}{L} = 1.888 + 0.444 = 2.332,$$

$$(x_3, y_3) = (6, \text{round}(2.332)) = (6, 2),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 6 - 1 = 5, y := y + \frac{y_{кон} - y_{нач}}{L} = 2.332 + 0.444 = 2.776,$$

$$(x_4, y_4) = (5, \text{round}(2.776)) = (5, 3),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 5 - 1 = 4, y := y + \frac{y_{кон} - y_{нач}}{L} = 2.776 + 0.444 = 3.22,$$

$$(x_5, y_5) = (4, \text{round}(3.22)) = (4, 3),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 4 - 1 = 3, y := y + \frac{y_{кон} - y_{нач}}{L} = 3.22 + 0.444 = 3.664,$$

$$(x_6, y_6) = (3, \text{round}(3.664)) = (3, 4),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 3 - 1 = 2, y := y + \frac{y_{кон} - y_{нач}}{L} = 3.664 + 0.444 = 4.108,$$

$$(x_7, y_7) = (2, \text{round}(4.108)) = (2, 4),$$

$$x := x + \frac{x_{кон} - x_{нач}}{L} = 2 - 1 = 1, y := y + \frac{y_{кон} - y_{нач}}{L} = 4.108 + 0.444 = 4.552,$$

$$(x_8, y_8) = (1, \text{round}(4.552)) = (1, 5),$$

$$(x_9, y_9) = (x_{кон}, y_{кон}) = (0, 5).$$

Полученные два множества пикселей (синие – в ходе симметричной реализации и красные – в ходе несимметричной) отмечены на рисунке 2.38. На нём заметно, что получились разные предпоследние пиксели.

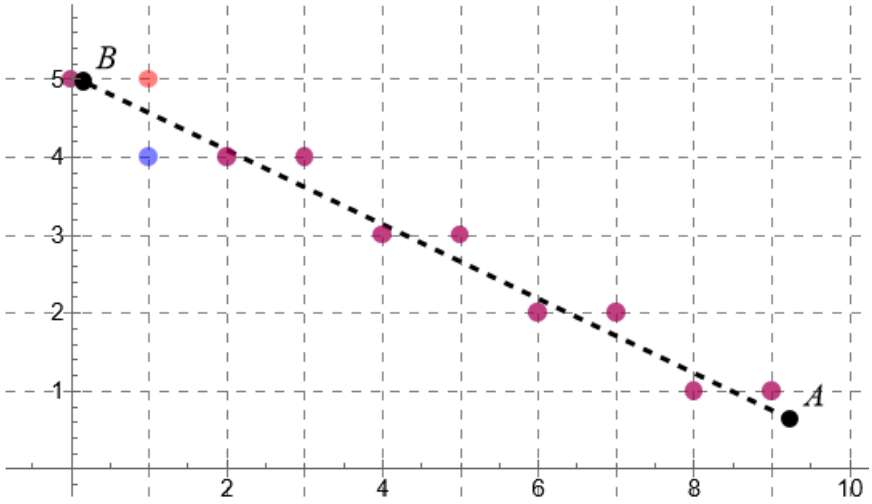


Рисунок 2.38 – Результаты растеризации отрезка алгоритмом DDA-линии

2.3.2. Алгоритм Брезенхема

Целочисленный алгоритм Брезенхема (стр. 113) находит пиксели растрированного отрезка посредством вычисления приращений их координат, которые могут при этом кодироваться кодом Ротштейна. В примерах ниже наряду с вычислением самой растеризации будем также вычислять коды Ротштейна для растрированных отрезков.

Пример 3.2. Осуществить растеризацию отрезка *AB*, концы которого имеют координаты *A*(0, 0) и *B*(7, 2).

Координаты вектора $\overrightarrow{AB}$, равные $x = 7$ и $y = 2$, удовлетворяют неравенствам $x > y > 0$. Следовательно, отрезок *AB* лежит в I октантном угле, и можно пользоваться формулами (1.62) и (1.61) без каких-либо преобразований либо изменений. При этом будем также строить код Ротштейна, записываемый в строку *s*:

$$d_1 = 2y - x = 2 \cdot 2 - 7 = -3 < 0 \Rightarrow \Delta y_1 = 0, s := \mathrm{S}.$$

Для дальнейших итераций можно сразу вычислить значения $2y - 2x = -10$, $2y = 4$. Вычислим приращения координат всех остальных пикселей на основе значений d_i, $i = \overline{2,7}$:

$$\begin{aligned}
&d_2 = d_1 + 2y = \quad -3 + 4 = 1 > 0 \quad \Rightarrow \Delta y_2 = 1, s := \mathrm{SD},\\
&d_3 = d_2 + 2y - 2x = 1 - 10 = -9 < 0 \Rightarrow \Delta y_3 = 0, s := \mathrm{SDS},\\
&d_4 = d_3 + 2y = \quad -9 + 4 = -5 < 0 \Rightarrow \Delta y_4 = 0, s := \mathrm{SDSS},\\
&d_5 = d_4 + 2y = \quad -5 + 4 = -1 < 0 \Rightarrow \Delta y_5 = 0, s := \mathrm{SDSSS},\\
&d_6 = d_5 + 2y = \quad -1 + 4 = 3 > 0 \quad \Rightarrow \Delta y_6 = 1, s := \mathrm{SDSSSD},\\
&d_7 = d_6 + 2y - 2x = 3 - 10 = -7 < 0 \Rightarrow \Delta y_7 = 0, s := \mathrm{SDSSSDS}.
\end{aligned}$$

Значит, координаты искомых пикселей равны

$$\begin{aligned}
&(x_1, y_1) = (x_A, y_A) = (0,0), && (x_2, y_2) = (x_1 + 1, y_1 + \Delta y_1) = (1,0),\\
&(x_3, y_3) = (x_2 + 1, y_2 + \Delta y_2) = (2,1), && (x_4, y_4) = (x_3 + 1, y_3 + \Delta y_3) = (3,1),\\
&(x_5, y_5) = (x_4 + 1, y_4 + \Delta y_4) = (4,1), && (x_6, y_6) = (x_5 + 1, y_5 + \Delta y_5) = (5,1),\\
&(x_7, y_7) = (x_6 + 1, y_6 + \Delta y_6) = (6,2), && (x_8, y_8) = (x_7 + 1, y_7 + \Delta y_7) = (7,2).
\end{aligned}$$

Итоговая растеризация представлена на рисунке 2.39. Ломаную линию, соединяющую эти пиксели, можно получить по коду Ротштейна. Для отрезка из I октантного угла код S означает смещение вправо, D – вправо и вверх.

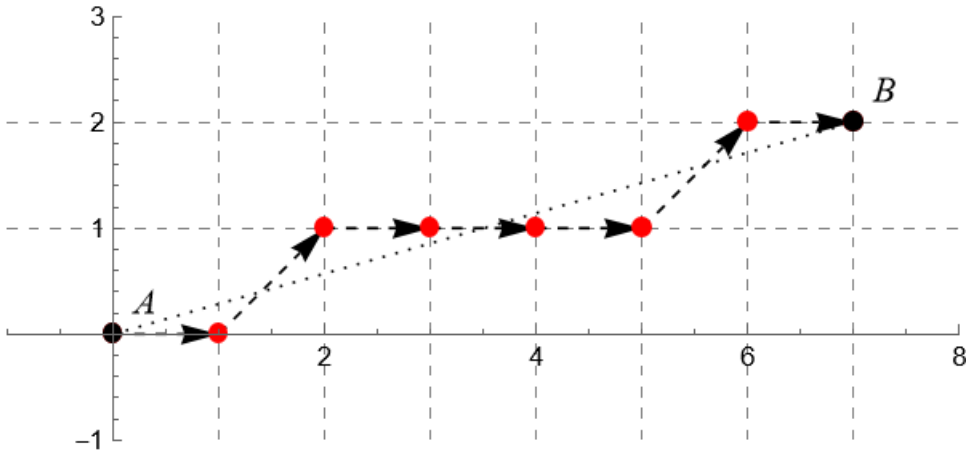


Рисунок 2.39 – Растеризация отрезка алгоритмом Брезенхема

Растеризуем теперь по Брезенхему отрезок из примера 3.1.

Пример 3.3. Осуществить растеризацию отрезка AB, концы которого имеют координаты $A(9.24, 0.64)$ и $B(0.16, 4.97)$.

При растеризации алгоритмом Брезенхема отрезка с концами, имеющими действительные координаты, снова можно положить концы растрированного отрезка в точках с округлёнными координатами: $A'(x_{\text{нач}}, y_{\text{нач}}) = A'(9,1)$, $B'(x_{\text{кон}}, y_{\text{кон}}) = B'(0,5)$.

С учётом того, что координаты вектора $\overrightarrow{A'B'}(-9,4)$ удовлетворяют неравенствам $x < 0$, $y > 0$, $|x| > |y|$, имеем, что отрезок $A'B'$ лежит в IV октантном угле. При помощи аффинных преобразований (в данном примере это параллельный перенос на вектор $\overrightarrow{A'O}(-9,-1)$, а затем симметрическое отражение относительно оси Oy, меняющее знак абсциссы всех точек плоскости) можно перевести отрезок $A'B'$ в отрезок с концами в начале координат и точке (9, 4). К этому новому отрезку применим алгоритм Брезенхема. При этом необходимо учитывать, что для исходного отрезка из IV октанта имеют место приращения координат $\Delta x_i = -1$, $\Delta y_i \in \{0,1\}$ (см. рисунок 1.64).

$$
\begin{aligned}
&2y - 2x = 2\cdot 4 - 2\cdot 9 = -10, 2y = 8,\\
&d_1 = 2y - x = \quad 2\cdot 4 - 9 = -1 < 0 \Rightarrow \Delta y_1 = 0, s := \text{S},\\
&d_2 = d_1 + 2y = \quad -1 + 8 = 7 > 0 \Rightarrow \Delta y_2 = 1, s := \text{SD},\\
&d_3 = d_2 + 2y - 2x = \quad 7 - 10 = -3 < 0 \Rightarrow \Delta y_3 = 0, s := \text{SDS},\\
&d_4 = d_3 + 2y = \quad -3 + 8 = 5 > 0 \Rightarrow \Delta y_4 = 1, s := \text{SDSD},\\
&d_5 = d_4 + 2y - 2x = \quad 5 - 10 = -5 < 0 \Rightarrow \Delta y_5 = 0, s := \text{SDSDS},\\
&d_6 = d_5 + 2y = \quad -5 + 8 = 3 > 0 \Rightarrow \Delta y_6 = 1, s := \text{SDSDSD},\\
&d_7 = d_6 + 2y - 2x = \quad 3 - 10 = -7 < 0 \Rightarrow \Delta y_7 = 0, s := \text{SDSDSDS},
\end{aligned}
$$

$$d_8 = d_7 + 2y = \quad -7+8=1>0 \quad \Rightarrow \Delta y_8 = 1, s := \text{SDSDSDSD},$$
$$d_9 = d_8 + 2y - 2x = \quad 1-10=-9<0 \quad \Rightarrow \Delta y_9 = 0, s := \text{SDSDSDSDS}.$$

Значит, координаты искомых пикселей равны

$$(x_1, y_1) = A'(9,1), \qquad (x_2, y_2) = (x_1 - 1, y_1 + \Delta y_1) = (8,1),$$
$$(x_3, y_3) = (x_2 - 1, y_2 + \Delta y_2) = (7,2), \qquad (x_4, y_4) = (x_3 - 1, y_3 + \Delta y_3) = (6,2),$$
$$(x_5, y_5) = (x_4 - 1, y_4 + \Delta y_4) = (5,3), \qquad (x_6, y_6) = (x_5 - 1, y_5 + \Delta y_5) = (4,3),$$
$$(x_7, y_7) = (x_6 - 1, y_6 + \Delta y_6) = (3,4), \qquad (x_8, y_8) = (x_7 - 1, y_7 + \Delta y_7) = (2,4),$$
$$(x_9, y_9) = (x_8 - 1, y_8 + \Delta y_8) = (1,5), \qquad (x_{10}, y_{10}) = (x_9 - 1, y_9 + \Delta y_9) = (0,5).$$

Результат работы алгоритма Брезенхема для этого отрезка представлен на рисунке 2.40. Растрированный отрезок можно также получить, используя код Ротштейна. При этом для отрезка из IV октантного угла код S соответствует движению влево, D – влево и вверх. Сравнивая рисунки 2.38 и 2.40, можно сделать вывод, что алгоритм Брезенхема даёт ту же растеризацию, что и несимметричная реализация алгоритма DDA-линии.

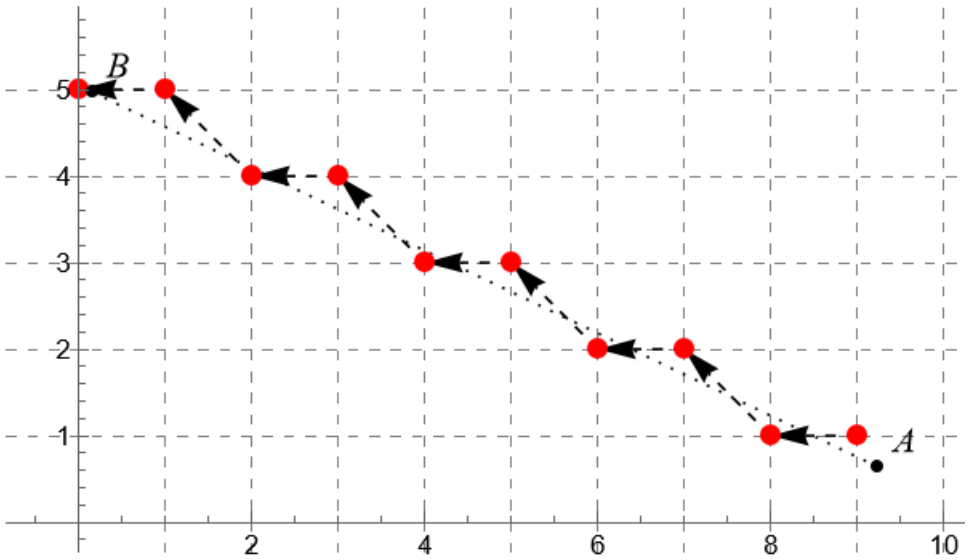


Рисунок 2.40 – Растеризация алгоритмом Брезенхема отрезка из IV октантного угла

2.3.3. Алгоритм Кастла-Питвея

Алгоритм Кастла-Питвея (стр. 115) позволяет вычислить код Ротштейна по заданным значениям $x > y > 0$, которые соответствуют некоторому отрезку из I октантного угла с началом в начале координат и концом в точке (x, y). Код Ротштейна (как уже было рассмотрено в подразделе 2.3.2) позволяет вычислить

пиксели растрированного отрезка, и этот результат совпадает с растеризацией, получаемой в результате применения алгоритма Брезенхема.

Пример 3.4. Осуществить растеризацию отрезка *AB*, концы которого имеют координаты *A*(–1, 8) и *B*(7, –2).

Координаты вектора $\overrightarrow{AB}(8,-10)$ удовлетворяют неравенствам $x > 0$, $y < 0$, $|x| < |y|$, т.е. отрезок *AB* лежит в VII октантном угле. При помощи симметрических отражений относительно прямой $y = -x$, а затем оси *Ox* этот отрезок переводится в I октантный угол, и его направляющий вектор становится равным $\overrightarrow{A'B'}(10,8)$. Поэтому в данном примере полагаем $x_B = 10$, $y_B = 8$.

Изначально в алгоритме Кастла-Питвея полагаем $m_1 := \mathrm{S}$, $m_2 := \mathrm{D}$, $x := x_B - y_B = 2$, $y := y_B = 8$. Итерации алгоритма выглядят следующим образом:

1. Так как $x < y$, то переопределяется приближение растеризации снизу: $m_1 := m_2 \oplus \sim m_1 = \mathrm{DS}$, $y := y - x = 6$.

2. Снова $x < y$, значит, $m_1 := m_2 \oplus \sim m_1 = \mathrm{D} \oplus \sim \mathrm{DS} = \mathrm{DSD}$, $y := y - x = 4$.

3. $x < y \Rightarrow m_1 := m_2 \oplus \sim m_1 = \mathrm{D} \oplus \sim \mathrm{DSD} = \mathrm{DDSD}$, $y := y - x = 2$.

4. Теперь выполняется $x = y$, следовательно, строка $m_2 \oplus \sim m_1 = \mathrm{DDSDD}$ повторяется $x = y = 2$ раза, и итоговый код равен DD S DDDD S DD.

Для отрезка из VII октантного угла код S соответствует движению вниз ($\Delta y_i = -1, \Delta x_i = 0$), D – движению вправо и вниз ($\Delta y_i = -1, \Delta x_i = 1$). Итоговая растеризация представлена на рисунке 2.41.

Пример 3.5. Осуществить растеризацию отрезка *AB*, концы которого имеют координаты *A*(–1, 2) и *B*(10, 7).

По вектору $\overrightarrow{AB}(11,5)$ легко видеть, что отрезок *AB* лежит в первом октантном угле. Значит, имеем изначально $x_B = 11$, $y_B = 5$, $x := x_B - y_B = 6$, $y := y_B = 5$, $m_1 := \mathrm{S}$, $m_2 := \mathrm{D}$.

1. $x > y$, следовательно, переопределяется приближение сверху: $m_2 := m_1 \oplus \oplus \sim m_2 = \mathrm{SD}$, $x := x - y = 1$.

2. $x < y \Rightarrow m_1 := m_2 \oplus \sim m_1 = \mathrm{SD} \oplus \sim \mathrm{S} = \mathrm{SDS}$, $y := y - x = 4$.

3. $x < y \Rightarrow m_1 := m_2 \oplus \sim m_1 = \mathrm{SD} \oplus \sim \mathrm{SDS} = \mathrm{SDSDS}$, $y := y - x = 3$.

4. $x < y \Rightarrow m_1 := m_2 \oplus \sim m_1 = \mathrm{SD} \oplus \sim \mathrm{SDSDS} = \mathrm{SDSDSDS}$, $y := y - x = 2$.

5. $x < y \Rightarrow m_1 := m_2 \oplus \sim m_1 = \mathrm{SD} \oplus \sim \mathrm{SDSDSDS} = \mathrm{SDSDSDSDS}$, $y := y - x = 1$.

6. Теперь уже $x = y = 1$, значит, итоговый код имеет вид $m_2 \oplus \sim m_1 = \mathrm{SD} \oplus \oplus \sim \mathrm{SDSDSDSDS} = \mathrm{SDSDSDSDSDS}$.

Результат выполнения алгоритма Кастла-Питвея для этого отрезка проиллюстрирован на рисунке 2.42.

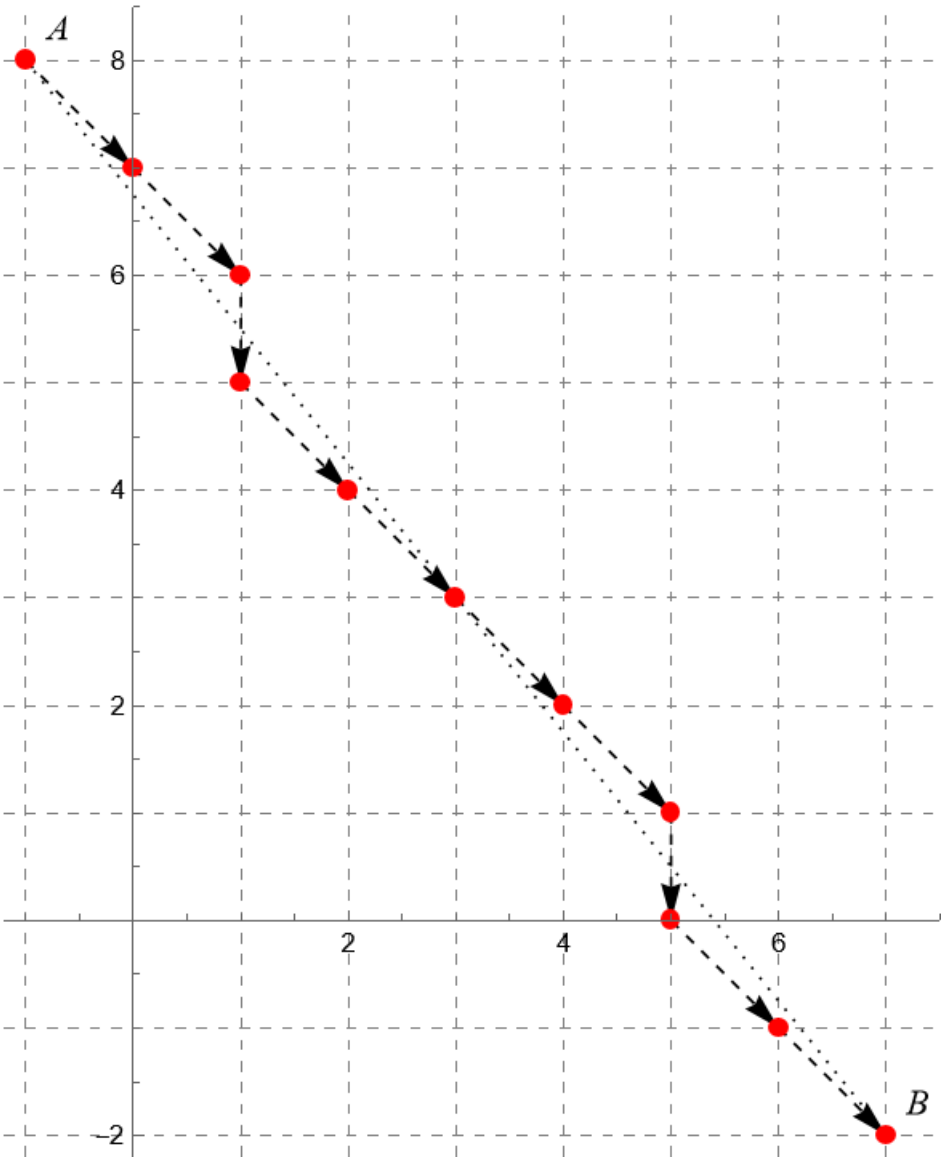


Рисунок 2.41 – Растеризация алгоритмом Кастла-Питвея отрезка из VII октантного угла

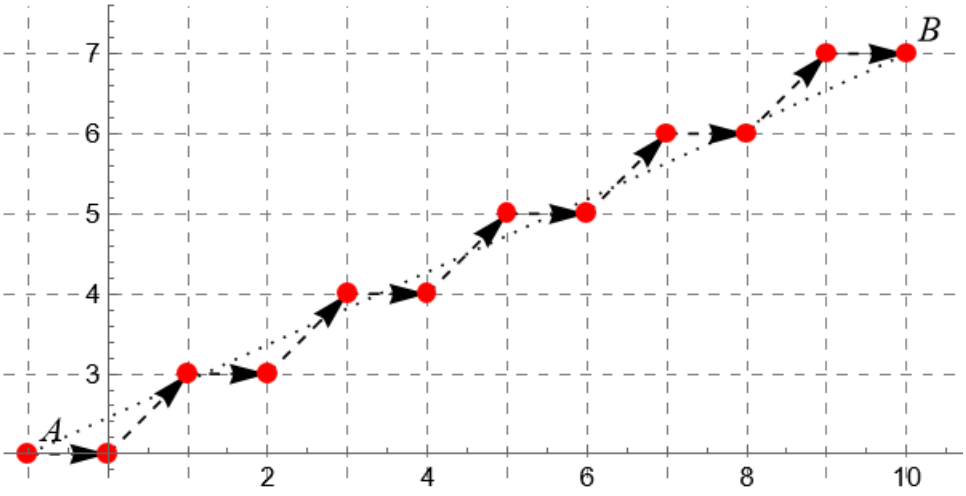


Рисунок 2.42 – Растеризация отрезка из I октантного угла

2.3.4. Алгоритм Ву

Алгоритм Ву (стр. 121) позволяет получить растеризацию отрезка с использованием различных цветовых оттенков, что даёт сглаженное изображение без «лесенок». Рассмотрим работу алгоритма Ву на примере сглаживания чёрных отрезков на белом фоне.

Пример 3.6. Построить растеризацию отрезка AB с антиалиасингом, концы которого имеют координаты $A(0, 0)$ и $B(12, 5)$. Использовать для этого $M = 2^m = 16$ оттенков серого.

Вначале необходимо выбрать параметр N, которым ограничивается значение итерационной переменной D, определяющий цвет текущих пикселей. Сам автор алгоритма утверждает, что погрешность при вычислении оттенка серого для того или иного пикселя можно оценить сверху следующим выражением [24]:

$$L2^{-n-1}(2^m - 1) + 1, \tag{2.24}$$

где L – длина растеризуемого отрезка, $n = \log_2 N$. Кроме того, согласно этому же источнику, «относительная погрешность в 10% … не приводит к заметному ухудшению качества изображения». Значит, для подбора подходящего значения параметра $N = 2^n$ достаточно выражение (2.24) ограничить сверху числом 0.1×2^m, и найти минимальное значение n, при котором полученное неравенство заведомо выполняется. Для $m = 4$ получим оценку $n > \log_2 L + 3$. Следовательно, можно положить $n = 8$, откуда $N = 256$, $N/M = 16$. Концевые пиксели окрашиваются в чёрный цвет: $g(0, 0) = g(12, 5) = 0$. По формуле (1.66) получается $d = \text{round}\left(\frac{5 \cdot 256}{12}\right) = 107$. Именно на это число каждую итерацию будет возрастать переменная $D := 0$, отвечающая за окрашивание пикселей искомого растрированного отрезка.

Итерации алгоритма Ву при заданных параметрах выглядят следующим образом:

1. Происходит сближение пикселей по горизонтали: $(x_1, y_1) := (1, 0)$, $(x_2, y_2) := (11, 5)$, а также увеличение итерационной переменной: $D := D + d = 0 + 107 = 107$. Неравенство $D < N$ остаётся справедливым, поэтому сближения по вертикали пока не происходит. Пиксели окрашиваются в следующие цвета:

$$g(x_1, y_1) = g(1, 0) = \left\lfloor \frac{D}{N/M} \right\rfloor = \left\lfloor \frac{107}{16} \right\rfloor = \left\lfloor 6\frac{11}{16} \right\rfloor = 6,$$

$$g(x_1, y_1 + 1) = g(1, 1) = M - 1 - g(1, 0) = 15 - 6 = 9,$$

$$g(x_2, y_2) = g(11,5) = 6,$$
$$g(x_2, y_2 - 1) = g(11,4) = 9.$$

2. Теперь рассматриваются пиксели $(x_1, y_1) := (2,0)$, $(x_2, y_2) := (10,5)$. Для них имеем $D := D + d = 214 < N = 256$, значит, сближение по вертикали снова не происходит. Вычисляем цвета:

$$g(2,0) = g(10,5) = \left\lfloor \frac{214}{16} \right\rfloor = \left\lfloor 13\frac{6}{16} \right\rfloor = 13, \quad g(2,1) = g(10,4) = 15 - 13 = 2.$$

3. Значения очередных пикселей: $(x_1, y_1) := (3,0)$, $(x_2, y_2) := (9,5)$. Тут уже необходимо сдвигать их по вертикали, так как $D := D + d = 321$ становится уже больше N. Поэтому получается $(x_1, y_1) := (3,1)$, $(x_2, y_2) := (9,4)$, $D := D - N = 321 - 256 = 65$. Вычислим цвета:

$$g(3,1) = g(9,4) = \left\lfloor \frac{65}{16} \right\rfloor = \left\lfloor 4\frac{1}{16} \right\rfloor = 4, \quad g(3,2) = g(9,3) = 15 - 4 = 11.$$

4. Следующие пиксели имеют координаты $(x_1, y_1) := (4,1)$, $(x_2, y_2) := (8,4)$. Так как $D := D + d = 65 + 107 = 172 < N$, то сближения по вертикали не происходит. Цвета текущих и соседних пикселей равны

$$g(4,1) = g(8,4) = \left\lfloor \frac{172}{16} \right\rfloor = \left\lfloor 10\frac{12}{16} \right\rfloor = 10, \quad g(4,2) = g(8,3) = 15 - 10 = 5.$$

5. Далее идут пиксели с координатами $(x_1, y_1) := (5,1)$, $(x_2, y_2) := (7,4)$. Для них получается $D := D + d = 172 + 107 = 279 > N$ – происходит сближение пикселей по вертикали: $(x_1, y_1) := (5,2)$, $(x_2, y_2) := (7,3)$, $D := D - N = 23$. Тогда имеет место окрашивание текущих пикселей в следующие цвета:

$$g(5,2) = g(7,3) = \left\lfloor \frac{23}{16} \right\rfloor = \left\lfloor 1\frac{7}{16} \right\rfloor = 1, \quad g(5,3) = g(7,2) = 15 - 1 = 14.$$

6. Текущая итерация является последней, так как на ней рассматриваются соседние по вертикали пиксели: $(x_1, y_1) := (6,2)$, $(x_2, y_2) := (6,3)$. Их можно раскрасить, например, в такие цвета:

$$D := D + d = 23 + 107 = 130,$$
$$g(6,2) = \left\lfloor \frac{130}{16} \right\rfloor = \left\lfloor 8\frac{2}{16} \right\rfloor = 8,$$
$$g(6,3) = 15 - 8 = 7.$$

Итоговая растеризация представлена на рисунке 2.43.

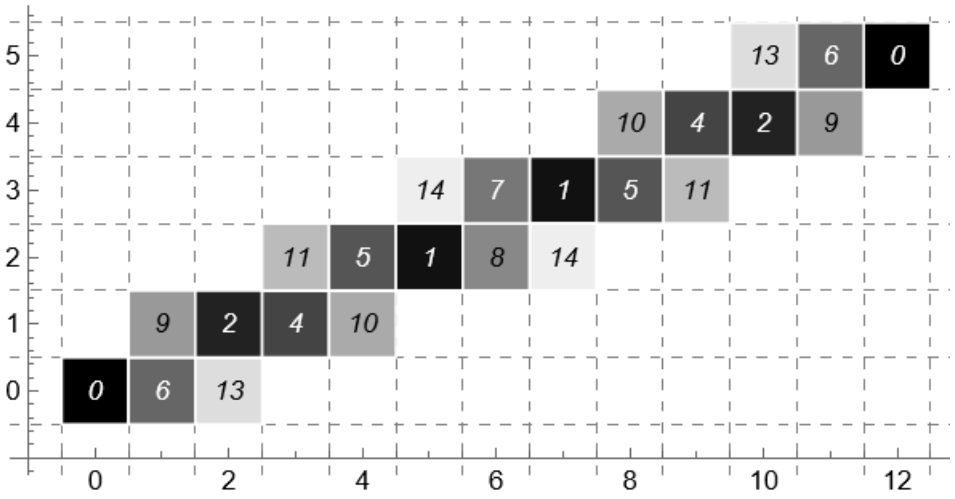


Рисунок 2.43 – Растеризация отрезка с использованием 16 оттенков серого

2.3.5. Растеризация окружностей и эллипсов

Растеризация окружностей и эллипсов может быть осуществлена по алгоритмам Брезенхема, представленным на стр. 127, либо их модификациям.

Пример 3.7. Осуществить растеризацию окружности с центром в точке $A(2, 4)$ и радиусом $R = 7$.

Осуществим вначале растеризацию окружности радиуса R с центром в начале координат. Найдём все пиксели **из II октантного угла**:

1. Начинается алгоритм с пикселя $(x_1, y_1) = (0, 7),\ \Delta_1 = 2 - 2R = -12.$
2. Так как $\Delta_1 < 0,$ то необходимо вычислить значение δ_-, для чего воспользуемся формулой (1.71):

$$\delta_- = 2\Delta_1 + 2y_1 - 1 = 2(-12) + 2 \cdot 7 - 1 < 0.$$

Значит, происходит сдвиг вправо: $(x_2, y_2) = (1, 7).$ Для следующей итерации будет использоваться значение

$$\Delta_2 = \Delta_1 + 2x_2 + 1 = -12 + 2 \cdot 1 + 1 = -9.$$

3. Аналогично первой итерации, имеем $\Delta_2 < 0,$ значит, получаются следующие результаты:

$$\delta_- = 2\Delta_2 + 2y_2 - 1 = 2(-9) + 2 \cdot 7 - 1 < 0,$$
$$\Delta_2 < 0 \,\&\, \delta_- < 0 \Rightarrow (x_3, y_3) = (2,7),$$
$$\Delta_3 = \Delta_2 + 2x_3 + 1 = -9 + 2 \cdot 2 + 1 = -4.$$

4. На следующей итерации по-прежнему $\Delta_3 < 0$, так что снова вычисляется значение δ_-:

$$\delta_- = 2\Delta_3 + 2y_3 - 1 = 2(-4) + 2 \cdot 7 - 1 > 0.$$

Следовательно, здесь уже происходит смещение по диагонали:

$$(x_4, y_4) = (3,6),$$
$$\Delta_4 = \Delta_3 + 2x_4 - 2y_4 + 2 = -4 + 2 \cdot 3 - 2 \cdot 6 + 2 = -8.$$

5. Снова $\Delta_4 < 0$:

$$\delta_- = 2\Delta_4 + 2y_4 - 1 = 2(-8) + 2 \cdot 6 - 1 < 0$$

– происходит движение вправо:

$$(x_5, y_5) = (4,6),$$
$$\Delta_5 = \Delta_4 + 2x_5 + 1 = -8 + 2 \cdot 4 + 1 = 1.$$

6. Теперь уже $\Delta_5 > 0$, значит, уже вычисляется число δ_+ по формуле (1.74):

$$\delta_+ = 2\Delta_5 - 2x_5 - 1 = 2 \cdot 1 - 2 \cdot 4 - 1 = -7 < 0.$$

Согласно алгоритму, происходит смещение по диагонали:

$$(x_6, y_6) = (5,5),$$
$$\Delta_6 = \Delta_5 + 2x_6 - 2y_6 + 2 = 1 + 2 \cdot 5 - 2 \cdot 5 + 2 = 3.$$

7. Получили пиксель, лежащий на границе II и I октантных углов: $x_6 = y_6 = 5$. Следовательно, дуга окружности с центром в начале координат радиуса $R = 7$, расположенная во II октанте, имеет следующую растеризацию:

II: (0, 7), (1, 7), (2, 7), (3, 6), (4, 6), (5, 5).

Растеризация дуг из остальных октантных углов получается при помощи различных симметрических отражений:

I: (7, 0), (7, 1), (7, 2), (6, 3), (6, 4), (5, 5);
III: (0, 7), (–1, 7), (–2, 7), (–3, 6), (–4, 6), (–5, 5);
IV: (–7, 0), (–7, 1), (–7, 2), (–6, 3), (–6, 4), (–5, 5);
V: (–7, 0), (–7, –1), (–7, –2), (–6, –3), (–6, –4), (–5, –5);
VI: (0, –7), (–1, –7), (–2, –7), (–3, –6), (–4, –6), (–5, –5);
VII: (0, –7), (1, –7), (2, –7), (3, –6), (4, –6), (5, –5);
VIII: (7, 0), (7, –1), (7, –2), (6, –3), (6, –4), (5, –5).

Сместив их все на вектор $\overrightarrow{OA}(2,4)$, получим растеризацию исходной окружности, представленную на рисунке 2.44. При выполнении алгоритма Брезенхема вычислялись только координаты синих пикселей.

Замечание. При растеризации дуги окружности, расположенной во II октантном угле, следующий пиксель всегда смещён относительно предыдущего вправо или по диагонали, но не вниз. Поэтому при растеризации дуги из II октантного угла при $\Delta_i > 0$ можно не вычислять δ_+, а сразу осуществлять вычисление следующего пикселя и значения Δ_{i+1} по формулам (1.78) и (1.82).

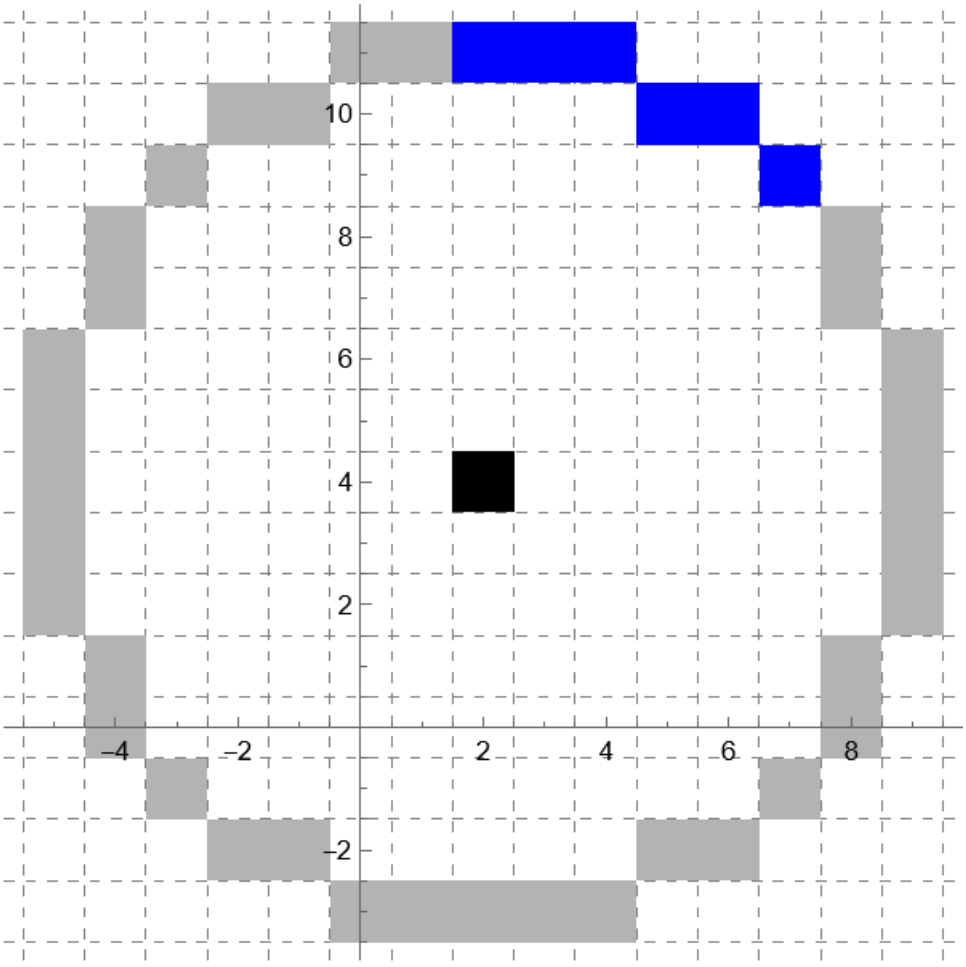


Рисунок 2.44 – Растеризация окружности по алгоритму Брезенхема

Пример 3.8. Осуществить растеризацию эллипса с центром в начале координат, полуоси которого параллельны координатным осям, причём длина горизонтальной полуоси равна $a = 5$, а вертикальной – $b = 3$.

Применим алгоритм растеризации к части эллипса, расположенной в I квадранте:

1. Начинается растеризация с пикселя $(x_1, y_1) = (0, b) = (0, 3)$, $\Delta_1 = a^2 + b^2 - 2a^2 b = 5^2 + 3^2 - 2 \cdot 5^2 \cdot 3 = -116$.
2. Так как $\Delta_1 < 0$, то на данной итерации вычисляется δ_- по формуле (1.83):

$$\delta_- = 2\Delta_1 + a^2(2y_1 - 1) = 2(-116) + 5^2(2 \cdot 3 - 1) = -232 + 25 \cdot 5 = -107.$$

Это значение меньше нуля, значит, далее применяются формулы (1.85):

$$(x_2, y_2) = (1, 3),$$
$$\Delta_2 = \Delta_1 + b^2(2x_2 + 1) = -116 + 3^2(2 \cdot 1 + 1) = -116 + 27 = -89.$$

3. На следующей итерации снова $\Delta_2 < 0$, значит, имеют место те же вычисления:

$$\delta_- = 2\Delta_2 + a^2(2y_2 - 1) = 2(-89) + 5^2(2 \cdot 3 - 1) = -178 + 25 \cdot 5 = -53,$$
$$\Delta_2 < 0 \,\&\, \delta_- < 0 \Rightarrow (x_3, y_3) = (2, 3),$$
$$\Delta_3 = \Delta_2 + b^2(2x_3 + 1) = -89 + 3^2(2 \cdot 2 + 1) = -89 + 45 = -44.$$

4. На следующей итерации происходят следующие вычисления:

$$\Delta_3 < 0 \Rightarrow \delta_- = 2\Delta_3 + a^2(2y_3 - 1) = 2(-44) + 5^2(2 \cdot 3 - 1) = -88 + 25 \cdot 5 = 37.$$

Значит, на этой итерации следующий пиксель расположен по диагонали от текущего пикселя. При этом значение Δ_4 вычисляется уже по формуле (1.86):

$$(x_4, y_4) = (3, 2),$$
$$\Delta_4 = \Delta_3 + b^2(2x_4 + 1) + a^2(-2y_4 + 1) = -44 + 3^2(2 \cdot 3 + 1) +$$
$$+5^2(-2 \cdot 2 + 1) = -44 + 9 \cdot 7 - 25 \cdot 3 = -56.$$

5. Снова $\Delta_4 < 0$:

$$\delta_- = 2\Delta_4 + a^2(2y_4 - 1) = 2(-56) + 5^2(2\cdot 2 - 1) = -112 + 25\cdot 3 = -37,$$
$$\Delta_4 < 0 \,\&\, \delta_- < 0 \Rightarrow (x_5, y_5) = (4, 2),$$
$$\Delta_5 = \Delta_4 + b^2(2x_5 + 1) = -56 + 3^2(2\cdot 4 + 1) = -56 + 81 = 25.$$

6. Теперь уже $\Delta_5 > 0$, значит, вычислению по формуле (1.84) подлежит значение δ_+:

$$\delta_+ = 2\Delta_5 - b^2(2x_5 + 1) = 2\cdot 25 - 3^2(2\cdot 4 + 1) = 50 - 81 = -31 < 0.$$

Значит, следующий пиксель расположен по диагонали от текущего:

$$(x_6, y_6) = (5,1),$$
$$\Delta_6 = \Delta_5 + b^2(2x_6 + 1) + a^2(-2y_6 + 1) = 25 +$$
$$+3^2(2\cdot 5 + 1) + 5^2(-2\cdot 1 + 1) = 25 + 99 - 25 = 99.$$

7. Пиксель (x_6, y_6) имеет абсциссу $x_6 = a = 5$, следовательно, все остальные пиксели будут расположены внизу от него: $(x_7, y_7) = (5,0)$. В итоге получаются следующие координаты искомых пикселей:

(0, 3), (1, 3), (2, 3), (3, 2), (4, 2), (5, 1), (5, 0).

Находя при помощи симметрических отражений пиксели для растеризации частей исходного эллипса во II, III и IV квадрантах, получим итоговую растеризацию всего эллипса, представленную на рисунке 2.45.

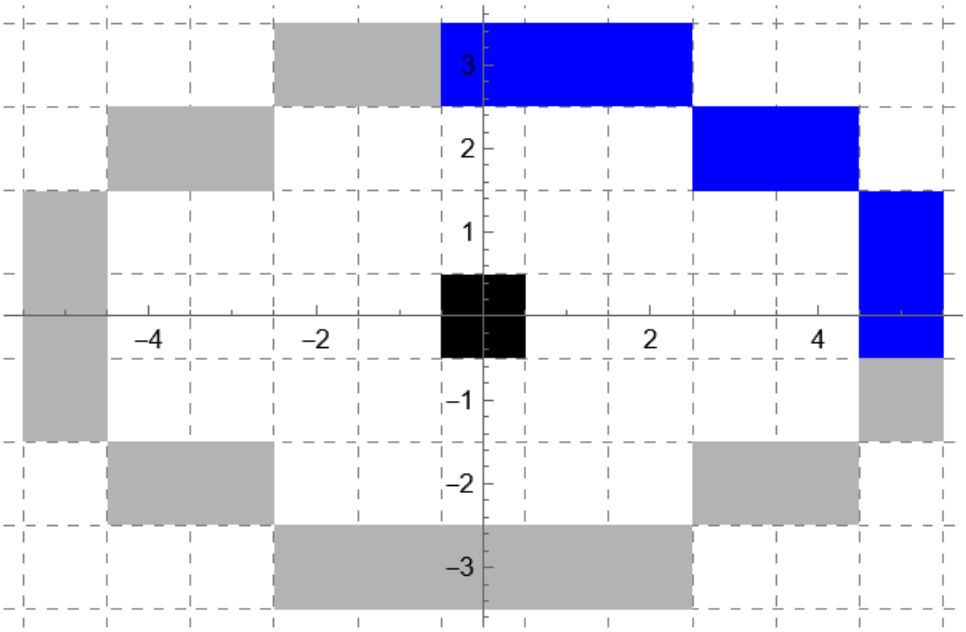


Рисунок 2.45 – Растеризация эллипса

2.3.6. Задачи

1. Построить код Ротштейна и растеризацию отрезка AB с концами в точках с заданными координатами.

1.1. $A(0, 0)$, $B(13, 5)$ (перейти к ответу),
1.2. $A(0, 0)$, $B(5, 8)$ (перейти к ответу),
1.3. $A(4, 2.6)$, $B(10.4, 6.6)$ (перейти к ответу),
1.4. $A(–8.8, –7.6)$, $B(2.2, 1.9)$ (перейти к ответу),
1.5. $A(2.4, 6.3)$, $B(6.6, 9.7)$ (перейти к ответу),
1.6. $A(9.8, 2.7)$, $B(3.3, 6.4)$ (перейти к ответу),
1.7. $A(7.8, 7)$, $B(3.9, 4.4)$ (перейти к ответу),
1.8. $A(2.4, 2.1)$, $B(8.1, 7.4)$ (перейти к ответу),
1.9. $A(5.3, 3)$, $B(1.6, 8.2)$ (перейти к ответу),
1.10. $A(0.7, 6.8)$, $B(1.1, 2.4)$ (перейти к ответу),
1.11. $A(1.3, 9)$, $B(2.6, 0.1)$ (перейти к ответу),
1.12. $A(1.4, 2.2)$, $B(9.4, 9.7)$ (перейти к ответу).

2. Построить растеризацию отрезка AB с антиалиасингом при $M \in \{4,8, 16\}$ оттенках серого (положить $N = 64$).

2.1. $A(0, 0)$, $B(6, 5)$ (перейти к ответу),
2.2. $A(0, 0)$, $B(7, 6)$ (перейти к ответу),
2.3. $A(0, 0)$, $B(6, 3)$ (перейти к ответу),
2.4. $A(0, 0)$, $B(7, 3)$ (перейти к ответу),
2.5. $A(0, 0)$, $B(9, 4)$ (перейти к ответу),
2.6. $A(0, 0)$, $B(10, 7)$ (перейти к ответу),
2.7. $A(0, 0)$, $B(12, 7)$ (перейти к ответу).

3. Осуществить растеризацию окружностей и эллипсов, оси которых параллельны координатным осям, если заданы координаты их центров и радиусы / длины полуосей.

3.1. $A(5, –5)$, $R = 11$ (перейти к ответу),
3.2. $A(–4, –10)$, $R = 15$ (перейти к ответу),
3.3. $A(–3, –9)$, $R = 9$ (перейти к ответу),
3.4. $A(–2, –4)$, $R = 10$ (перейти к ответу),
3.5. $A(–7, 8)$, $R = 4$ (перейти к ответу),
3.6. $A(2, 6)$, $R = 5$ (перейти к ответу),
3.7. $A(0, 0)$, $a = 5$, $b = 2$ (перейти к ответу),
3.8. $A(0, 0)$, $a = 8$, $b = 3$ (перейти к ответу),
3.9. $A(0, 0)$, $a = 10$, $b = 4$ (перейти к ответу),
3.10. $A(0, 0)$, $a = 4$, $b = 8$ (перейти к ответу),
3.11. $A(0, 0)$, $a = 5$, $b = 9$ (перейти к ответу).

2.4. Алгоритмы построения выпуклой оболочки множества точек на плоскости

2.4.1. Алгоритм Джарвиса

Алгоритм Джарвиса (алгоритм заворачивания подарка, подраздел 1.4.1) строит выпуклую оболочку множества точек, проводя на каждой итерации ребро, в дальнейшем не подлежащее изменению либо какому-либо ещё рассмотрению.

Пример 4.1. Построить выпуклую оболочку множества точек $\{A_i\}_{i=1}^{7}$ с координатами $A_1(-4,5)$, $A_2(-6,9)$, $A_3(7,-5)$, $A_4(10,3)$, $A_5(-7,-6)$, $A_6(10,6)$ $A_7(-9,\ -4)$.

Перед выполнениями итераций алгоритма Джарвиса нужно выбрать какую-нибудь крайнюю точку, например, самую нижнюю (если таких несколько, то обычно выбирают самую правую). В нашем случае такой является точка A_5 (Рисунок 2.46). Эта точка является вершиной искомой минимальной выпуклой оболочки, потому что всякую выпуклую оболочку, содержащие точки ниже проведённой прямой, можно минимизировать, отсекая этой прямой нижнюю часть. В случае, если бы на этой прямой лежало бы несколько точек, то сразу получили бы одно из рёбер искомой выпуклой оболочки.

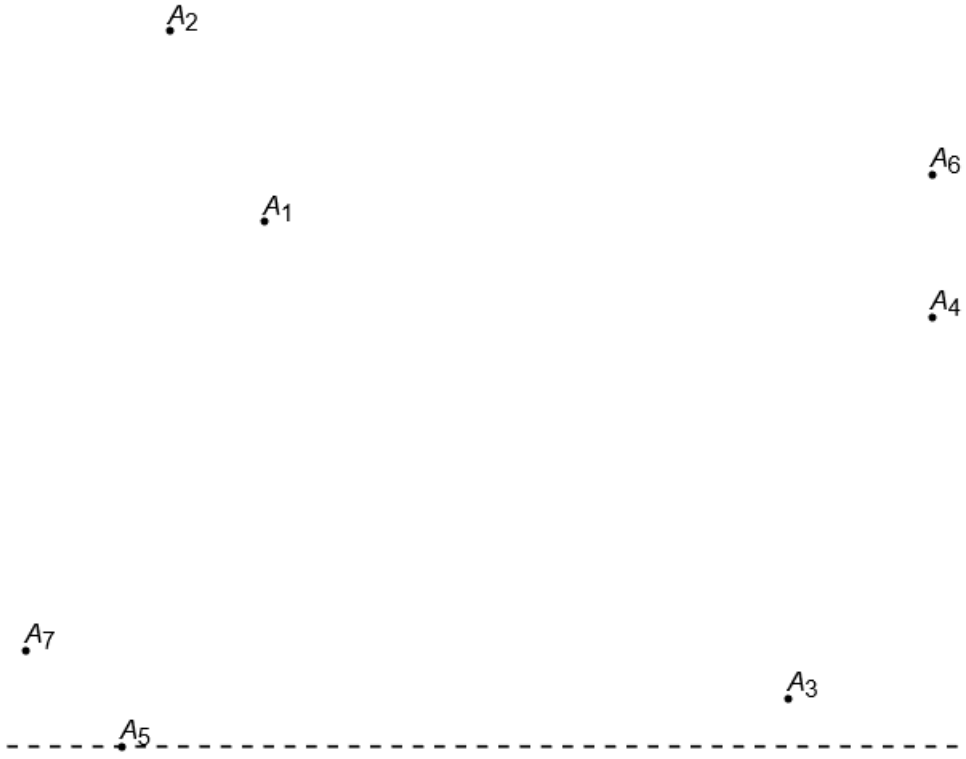


Рисунок 2.46 – Нахождение первой вершины выпуклой оболочки

Далее проводятся все векторы вида $\overrightarrow{A_5A_i}$, $i \neq 5$, из них выбирается тот, который находится ближе всего к проведённой пунктирной прямой. Это можно сделать, если найти вектор, угол между которым и каждым из остальных **больше нуля (если нужно обеспечить левосторонний обход)** либо **меньше нуля (правосторонний обход)**. Перебираем все векторы:

$$\overrightarrow{A_5A_1}(3,11), \overrightarrow{A_5A_2}(1,15), \overrightarrow{A_5A_3}(14,1), \overrightarrow{A_5A_4}(17,9), \overrightarrow{A_5A_6}(17,12), \overrightarrow{A_5A_7}(-2,2);$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_1}, \overrightarrow{A_5A_2}\right) = \operatorname{sgn}\begin{vmatrix} 3 & 11 \\ 1 & 15 \end{vmatrix} = 1.$$

Это означает, что от вектора $\overrightarrow{A_5A_1}$ поворот к вектору $\overrightarrow{A_5A_2}$ происходит против часовой стрелки, значит, вектор $\overrightarrow{A_5A_2}$ отбрасывается из рассмотрения (отбросили бы $\overrightarrow{A_5A_1}$, если нужно было строить последовательность точек, обеспечивающую правосторонний обход выпуклой оболочки).

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_1}, \overrightarrow{A_5A_3}\right) = \operatorname{sgn}\begin{vmatrix} 3 & 11 \\ 14 & 1 \end{vmatrix} = -1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_3}, \overrightarrow{A_5A_4}\right) = \operatorname{sgn}\begin{vmatrix} 14 & 1 \\ 17 & 9 \end{vmatrix} = 1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_3}, \overrightarrow{A_5A_6}\right) = \operatorname{sgn}\begin{vmatrix} 14 & 1 \\ 17 & 12 \end{vmatrix} = 1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_3}, \overrightarrow{A_5A_7}\right) = \operatorname{sgn}\begin{vmatrix} 14 & 1 \\ -2 & 2 \end{vmatrix} = 1.$$

В конце отбрасывается вектор $\overrightarrow{A_5A_7}$, и остаётся вектор $\overrightarrow{A_5A_3}$. Таким образом, проводится ребро A_5A_3. Это ребро является окончательным и уже не будет отброшено на последующих итерациях (Рисунок 2.47).

Теперь от точки A_3 можно провести следующее ребро ко всем остальным точкам, кроме A_5. По тому же алгоритму перебираем векторы $\overrightarrow{A_3A_i}$, $i \in \{1,2,4,6,7\}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_1}, \overrightarrow{A_3A_2}\right) = \operatorname{sgn}\begin{vmatrix} -11 & 10 \\ -13 & 14 \end{vmatrix} = -1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_2}, \overrightarrow{A_3A_4}\right) = \operatorname{sgn}\begin{vmatrix} -13 & 14 \\ 3 & 8 \end{vmatrix} = -1,$$

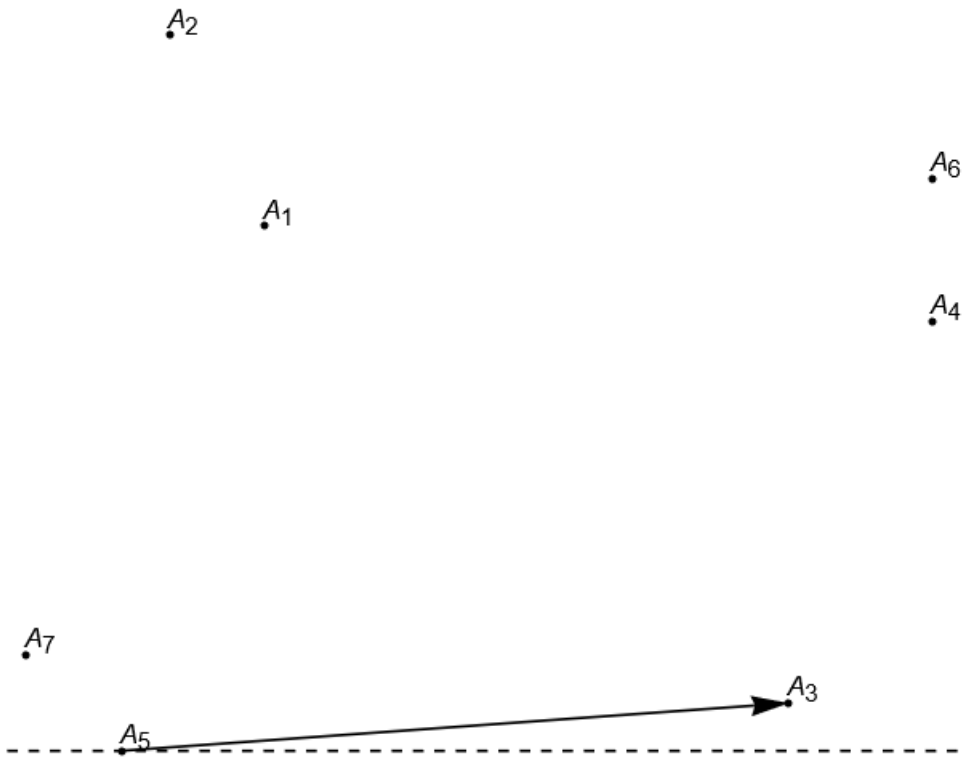


Рисунок 2.47 – Проведение первого ребра искомой выпуклой оболочки по алгоритму Джарвиса

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_4},\overrightarrow{A_3A_6}\right)=\operatorname{sgn}\begin{vmatrix}3 & 8\\ 3 & 11\end{vmatrix}=1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_4},\overrightarrow{A_3A_7}\right)=\operatorname{sgn}\begin{vmatrix}3 & 8\\ -16 & 1\end{vmatrix}=1.$$

Проводим ребро A_3A_4 (Рисунок 2.48).

Следующее ребро можно провести от точки A_4 ко всем остальным, за исключением A_3 (так как для ней уже найдены две смежные вершины), **в том числе и к самой первой точке** A_5. Если нужным ребром окажется A_4A_5, то тогда обход замкнётся, и выпуклая оболочка будет построена – алгоритм завершится. В противном случае нужно будет продолжить обход, проводя рёбра от последней полученной точки ко всем ещё не достигнутым, а также к первоначально выбранной точке обхода.

Проведём векторы $\overrightarrow{A_4A_i}, i\in\{1,2,5,6,7\}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_1},\overrightarrow{A_4A_2}\right)=\operatorname{sgn}\begin{vmatrix}-14 & 2\\ -16 & 6\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_2},\overrightarrow{A_4A_5}\right)=\operatorname{sgn}\begin{vmatrix}-16 & 6\\ -17 & -9\end{vmatrix}=1,$$

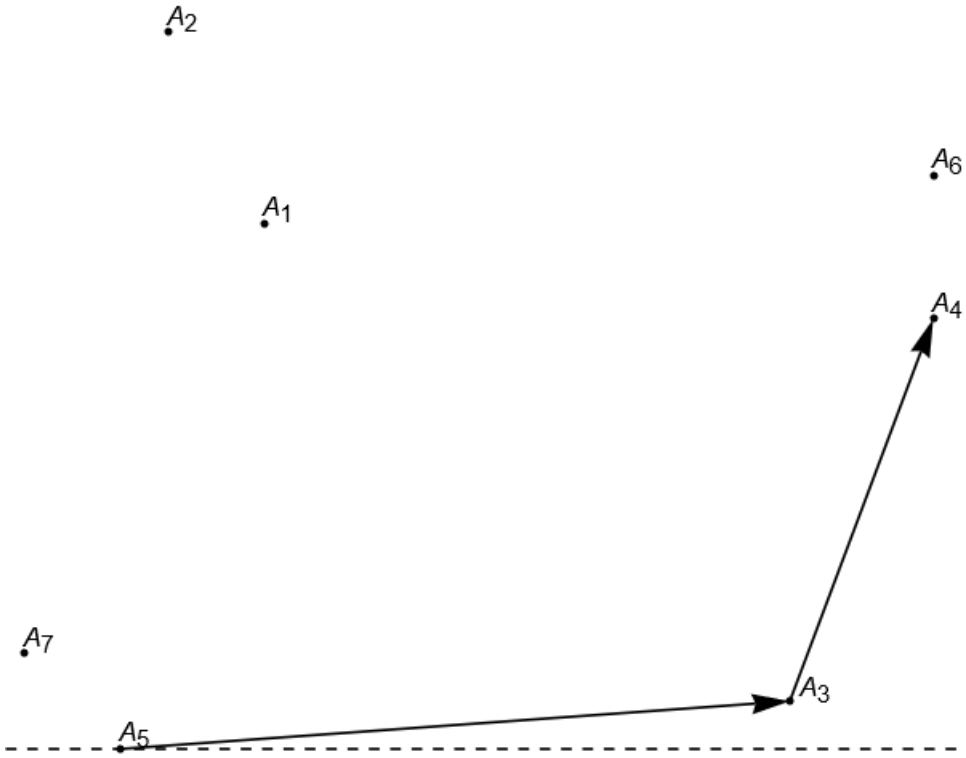


Рисунок 2.48 – Второе ребро

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_2},\overrightarrow{A_4A_6}\right)=\operatorname{sgn}\begin{vmatrix}-16 & 6\\ 0 & 3\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_6},\overrightarrow{A_4A_7}\right)=\operatorname{sgn}\begin{vmatrix}0 & 3\\ -19 & -7\end{vmatrix}=1.$$

Значит, на данной итерации необходимо добавить ребро A_4A_6 (Рисунок 2.49). От точки A_6 проводятся векторы к точкам A_1, A_2, A_5, A_7:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_6A_1},\overrightarrow{A_6A_2}\right)=\operatorname{sgn}\begin{vmatrix}-14 & -1\\ -16 & 3\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_6A_2},\overrightarrow{A_6A_5}\right)=\operatorname{sgn}\begin{vmatrix}-16 & 3\\ -17 & -12\end{vmatrix}=1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_6A_2},\overrightarrow{A_6A_7}\right)=\operatorname{sgn}\begin{vmatrix}-16 & 3\\ -19 & -10\end{vmatrix}=1.$$

Проводится ребро A_6A_2 (Рисунок 2.50).

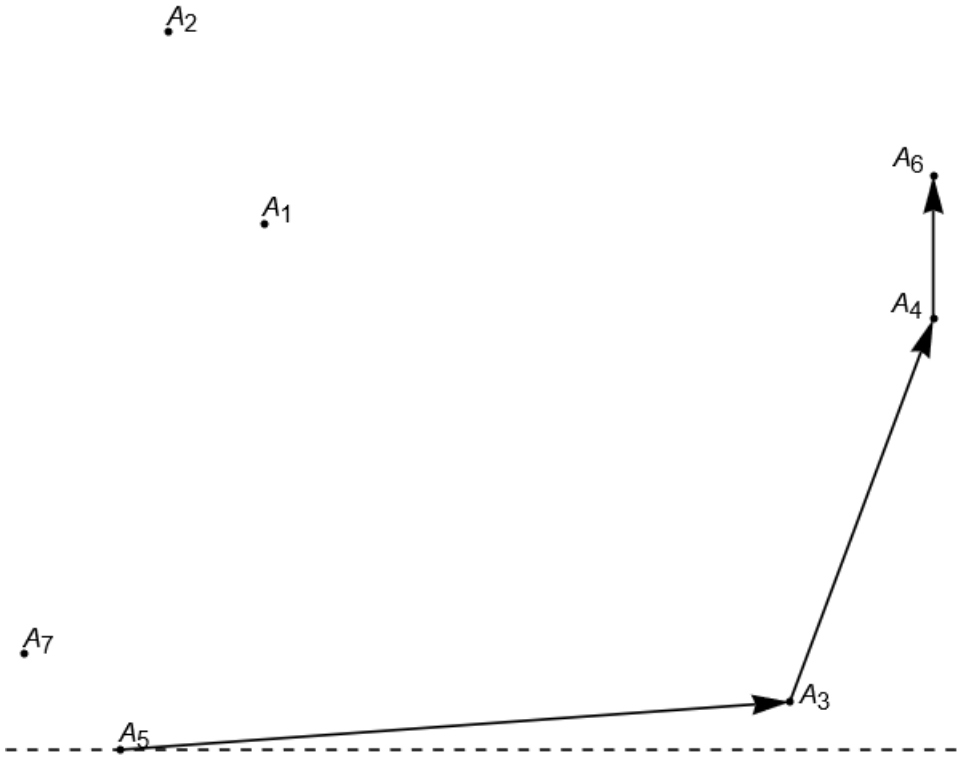


Рисунок 2.49 – Третье ребро

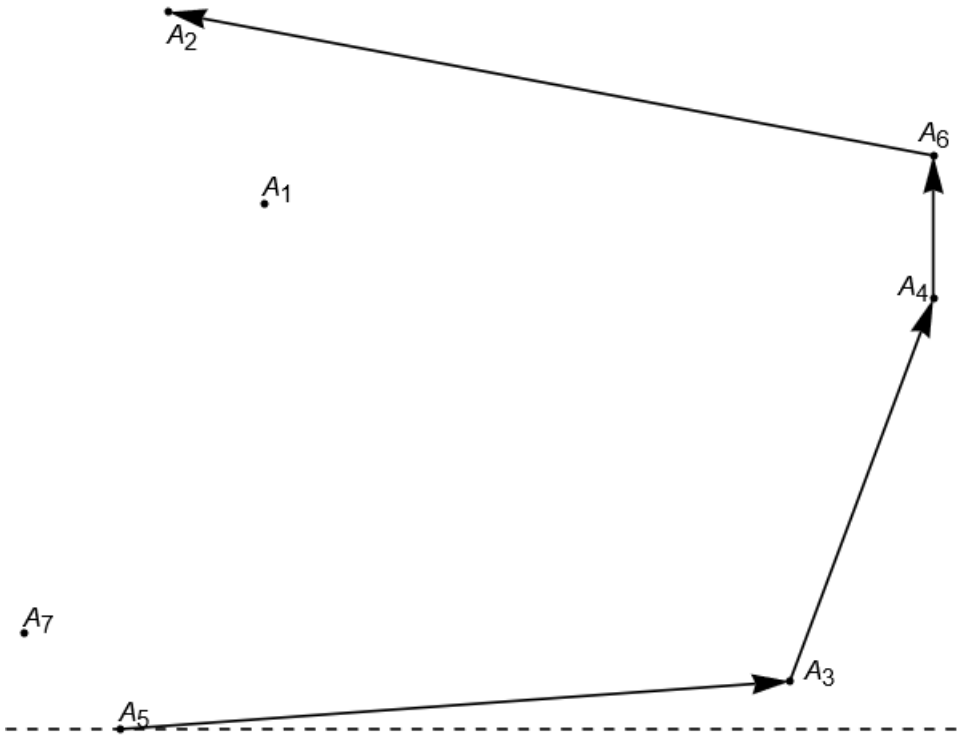


Рисунок 2.50 – Четвёртое ребро

На следующей итерации проводится одно из трёх рёбер: A_2A_1, A_2A_5 либо A_2A_7:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_2A_1}, \overrightarrow{A_2A_5}\right) = \operatorname{sgn}\begin{vmatrix} 2 & -4 \\ -1 & -15 \end{vmatrix} = -1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_2A_5}, \overrightarrow{A_2A_7}\right) = \operatorname{sgn}\begin{vmatrix} -1 & -15 \\ -3 & -13 \end{vmatrix} = -1.$$

Значит, следующим ребром будет A_2A_7 (Рисунок 2.51).

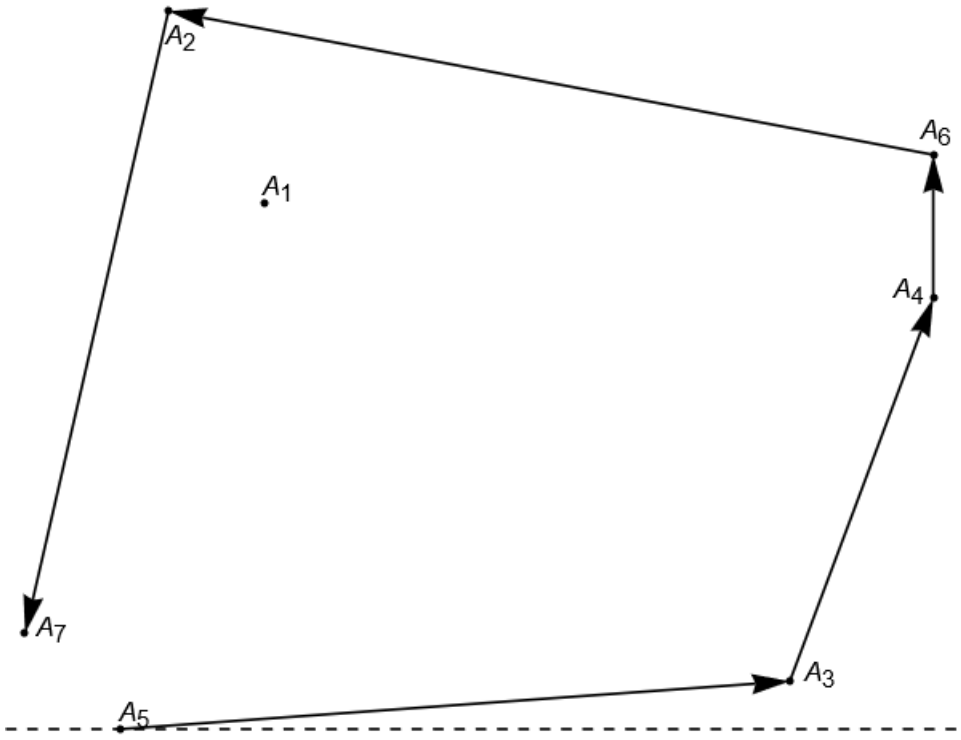


Рисунок 2.51 – Пятое ребро

Наконец, из двух рёбер A_7A_1 и A_7A_5 проводится последнее, так как $\operatorname{sgn}\angle\left(\overrightarrow{A_7A_1}, \overrightarrow{A_7A_5}\right) = \begin{vmatrix} 5 & 9 \\ 2 & -2 \end{vmatrix} = -1$, т.е. от вектора $\overrightarrow{A_7A_1}$ к вектору $\overrightarrow{A_7A_5}$ происходит поворот по часовой стрелке. Обход замкнулся, и получилась выпуклая оболочка $A_5A_3A_4A_6A_2A_7A_5$ с левосторонним обходом (Рисунок 2.52).

Замечание. Если при нахождении знака угла между двумя очередными векторами получается ноль, то из них отбрасывается вектор меньшей длины. Отсюда также следует, что нашлась некоторая точка исходного множества, находящаяся на границе выпуклой оболочки.

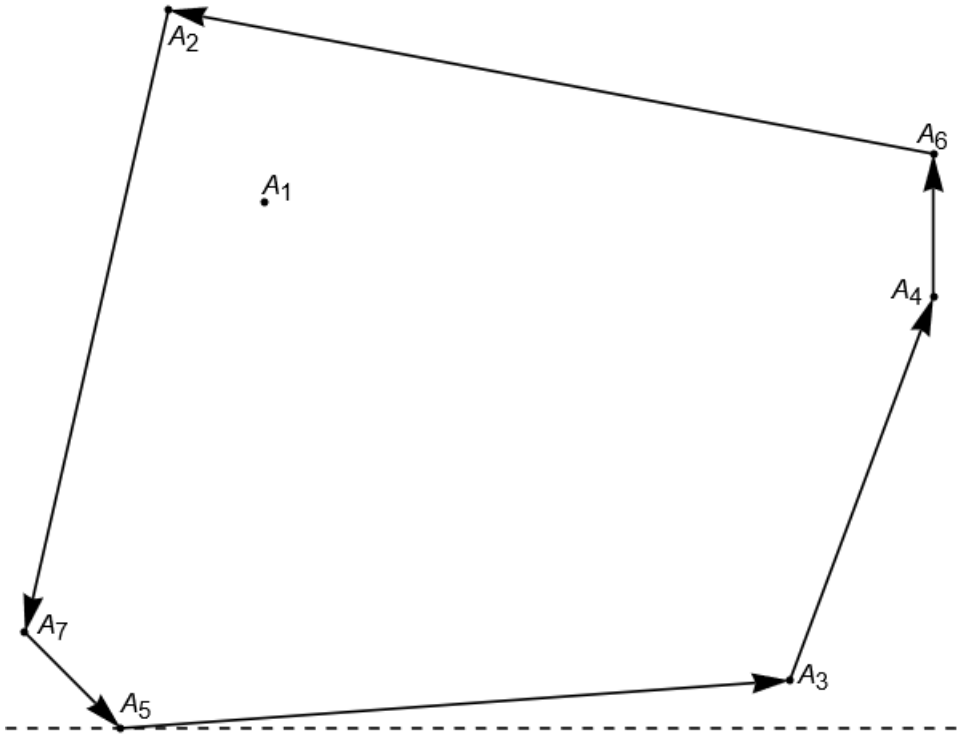


Рисунок 2.52 – Проведение последнего ребра и нахождение искомой выпуклой оболочки

2.4.2. Алгоритм Грэхема

Алгоритм Грэхема (подраздел 1.4.2) основан на сортировке векторов, проведённых от некоторой крайней точки исходного множества точек ко всем остальным. Благодаря использованию оптимальных алгоритмов сортировки, в том числе и тех, которые используются при сортировке обычных числовых массивов, возможно построение выпуклой оболочки за линейно-логарифмическое время.

Пример 4.2. Построить выпуклую оболочку множества точек $\{A_i\}_{i=1}^{7}$ с координатами $A_1(6,9)$, $A_2(-8,0)$, $A_3(6,5)$, $A_4(-5,8)$, $A_5(-3,2)$, $A_6(3,4)$, $A_7(6,-5)$.

Вначале, как и в алгоритме Джарвиса, выбирается точка с наименьшей ординатой. В данном примере это точка A_7 (Рисунок 2.53).

Далее следует отсортировать все остальные точки следующим образом: сортируется последовательность векторов $\overrightarrow{A_7A_i}$, $i=\overline{1,6}$, в порядке поворота **против часовой стрелки (если нужно получить левосторонний обход)** либо **по часовой стрелке (для правостороннего обхода)**. Выберем левосторонний обход. Сортировку векторов можно осуществить двумя способами.

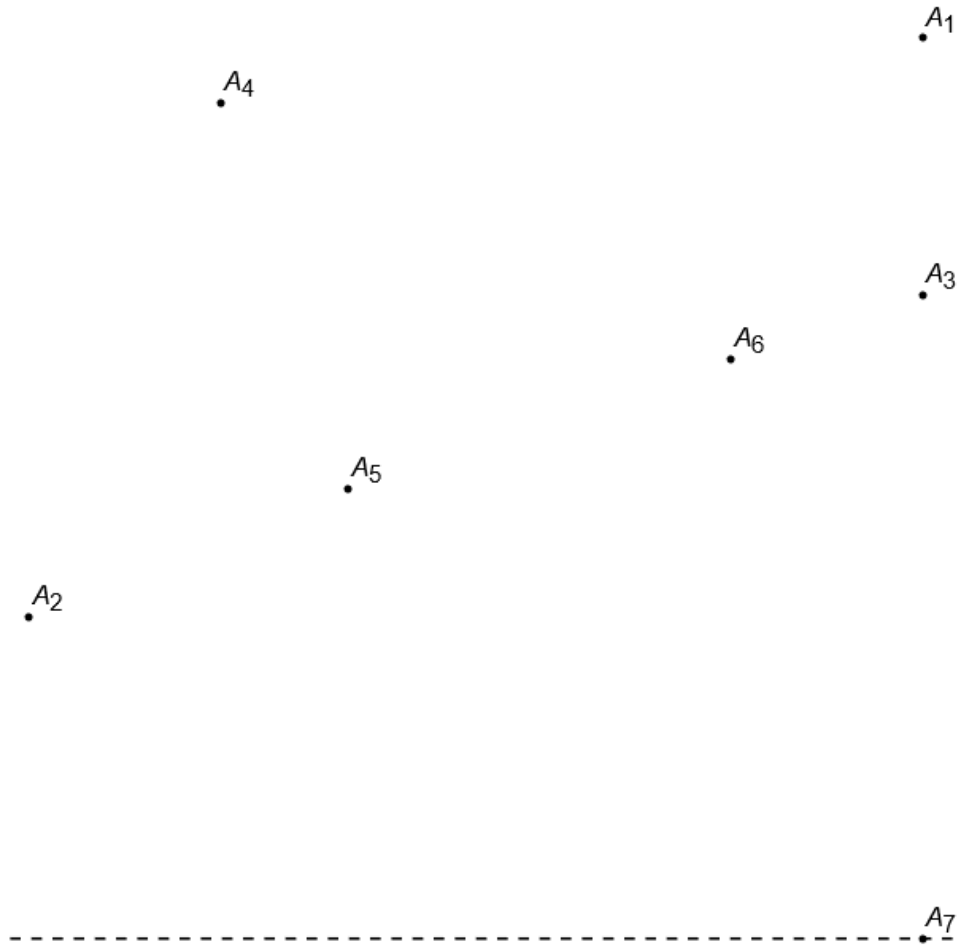


Рисунок 2.53 – Нахождение первой вершины выпуклой оболочки по алгоритму Грэхема

I. **Алгоритм вставки сортировки массива.** Как и числовые массивы, массивы векторов можно сортировать при помощи алгоритма вставки [1]. Заключается он в том, что при формировании отсортированного массива очередной элемент сравнивается с элементом посередине, на основании чего определяется, должен ли новый элемент быть вставлен до либо после этого среднего элемента. Этот подход повторяется для выбранной половинки массива, пока не будет определено точное положение для нового элемента. Благодаря такому принципу (**принцип дихотомии**) можно избежать попарных сравнений всех векторов, что при надлежащей программной реализации даёт сортировку за линейно-логарифмическое время. Рисунок 2.54 демонстрирует этот процесс для списка из семи векторов. Как нетрудно видеть, для вставки вектора в него достаточно трёх сравнений (знаками «больше» и «меньше» на рисунках 2.54, а-в, отмечается тот факт, что угол между новым и текущим вектором больше либо меньше нуля соответственно).

Применим этот подход в данном примере. При этом будем строить последовательность векторов, в которой слева направо идут векторы,

[1] Но только если их все, отложенные от одной точки, можно ограничить углом меньше развёрнутого, что, впрочем, и происходит на первом этапе алгоритма Грэхема

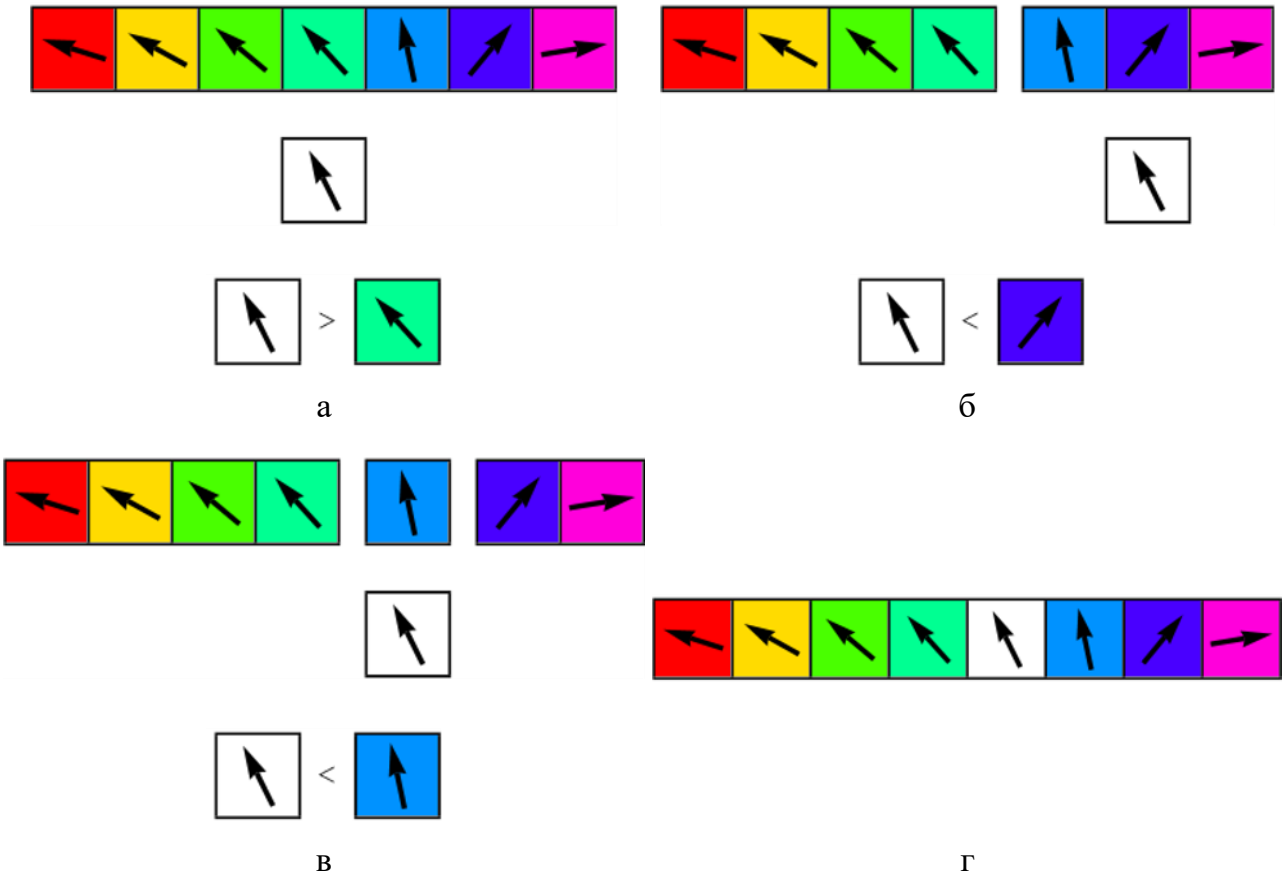


Рисунок 2.54 – Вставка нового вектора в отсортированный список векторов

поворачивающиеся друг к другу по направлению часовой стрелки (как на рисунке 2.54). Вначале вычислим знак угла между двумя первыми векторами:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_1},\overrightarrow{A_7A_2}\right)=\operatorname{sgn}\begin{vmatrix}0 & 14\\ -14 & 5\end{vmatrix}=1.$$

Значит, вектор $\overrightarrow{A_7A_1}$ по направлению часовой стрелки идёт после вектора $\overrightarrow{A_7A_2}$, и на этом этапе получается последовательность из двух векторов, изображённая на рисунке 2.55.

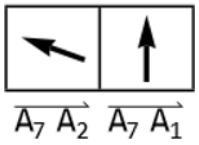


Рисунок 2.55 – Отсортированный список из двух векторов

Далее определяется, в какое место этой последовательности нужно вставить вектор $\overrightarrow{A_7A_3}$. Для этого достаточно провести не более двух операций вычисления знака углов между векторами:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_3},\overrightarrow{A_7A_1}\right)=\operatorname{sgn}\begin{vmatrix}0 & 10\\ 0 & 14\end{vmatrix}=0.$$

Получился ноль, т.е. три точки A_7, A_1, A_3 лежат на одной прямой. В таком случае один из векторов можно вообще не вносить в отсортированную последовательность. Из двух векторов $\overrightarrow{A_7A_1}$ и $\overrightarrow{A_7A_3}$ большую длину имеет первый (это легко проверить, сравнивая ординаты трёх точек), значит, $\overrightarrow{A_7A_1}$ остаётся, а $\overrightarrow{A_7A_3}$ исключается из рассмотрения. Таким образом, последовательность векторов не изменяется. Если бы $\overrightarrow{A_7A_3}$ оказался длиннее $\overrightarrow{A_7A_1}$, то в последовательности заменили бы вектор $\overrightarrow{A_7A_1}$ на $\overrightarrow{A_7A_3}$.

Для вектора $\overrightarrow{A_7A_4}$ производятся следующие вычисления:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_4},\overrightarrow{A_7A_1}\right)=\operatorname{sgn}\begin{vmatrix}-11 & 13\\ 0 & 14\end{vmatrix}=-1.$$

Значит, вектор $\overrightarrow{A_7A_4}$ будет располагаться перед вектором $\overrightarrow{A_7A_1}$ в отсортированной последовательности. Ещё надо сравнить взаимное расположение векторов $\overrightarrow{A_7A_4}$ и $\overrightarrow{A_7A_2}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_4},\overrightarrow{A_7A_2}\right)=\operatorname{sgn}\begin{vmatrix}-11 & 13\\ -14 & 5\end{vmatrix}=1.$$

Таким образом, вектор $\overrightarrow{A_7A_4}$ располагается после вектора $\overrightarrow{A_7A_2}$, и вставка происходит, как показано на рисунке 2.56.

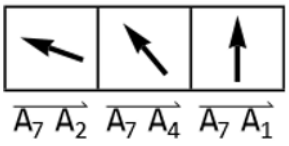


Рисунок 2.56 – Дальнейшее построение отсортированного списка

Переходя к вектору $\overrightarrow{A_7A_5}$, найдём вначале знак угла между векторами $\overrightarrow{A_7A_5}$ и $\overrightarrow{A_7A_4}$. Такой выбор вектора обеспечивает деление всей последовательности точек пополам, и вне зависимости от вычисленного знака дальше нужно будет вычислять знак только одного угла:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_5},\overrightarrow{A_7A_4}\right)=\operatorname{sgn}\begin{vmatrix}-9 & 7\\ -11 & 13\end{vmatrix}=-1.$$

Значит, вектор $\overrightarrow{A_7A_5}$ будет расположен перед вектором $\overrightarrow{A_7A_4}$. Дальше нужно определить взаимное расположение векторов $\overrightarrow{A_7A_5}$ и $\overrightarrow{A_7A_2}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_5},\overrightarrow{A_7A_2}\right)=\operatorname{sgn}\begin{vmatrix}-9 & 7\\ -14 & 5\end{vmatrix}=1.$$

Значит, вектор $\overrightarrow{A_7A_5}$ располагается после $\overrightarrow{A_7A_2}$, и получается список, представленный на рисунке 2.57а.

Рассматривая далее вектор $\overrightarrow{A_7A_6}$, выбираем для сравнения один из векторов, находящихся примерно посередине уже построенной последовательности, например $\overrightarrow{A_7A_4}$:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_6},\overrightarrow{A_7A_4}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 9\\ -11 & 13\end{vmatrix}=1.$$

Значит, $\overrightarrow{A_7A_6}$ находится после $\overrightarrow{A_7A_4}$ в последовательности. Осталось вычислить знак ещё одного угла:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_7A_6},\overrightarrow{A_7A_1}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 9\\ 0 & 14\end{vmatrix}=-1.$$

Таким образом, окончательно получается отсортированная последовательность, представленная на рисунке 2.57б.

а б

Рисунок 2.57 – Заключительные шаги построения отсортированного списка векторов

II. Отсортировать векторы можно также по их **угловым коэффициентам**:

$$\overrightarrow{A_7A_1}(0,14):k_1=14/0=+\infty,$$
$$\overrightarrow{A_7A_2}(-14,5):k_2=-5/14,$$
$$\overrightarrow{A_7A_3}(0,10):k_3=10/0=+\infty,$$
$$\overrightarrow{A_7A_4}(-11,13):k_4=-13/11,$$

$$\overrightarrow{A_7A_5}(-9,7): k_5 = -7/9,$$
$$\overrightarrow{A_7A_6}(-3,9): k_6 = -9/3 = -3.$$

Согласно выбору точки A_7, каждый из этих векторов должны лежать в I либо во II квадранте. I квадранту соответствует положительный угловой коэффициент, II квадранту – отрицательный. При этом при повороте луча с началом в точке A_7 против часовой стрелки в пределах одного квадранта угловой коэффициент луча будет всегда увеличиваться. В этом нетрудно убедиться, рассматривая график тангенса на отрезке [0, π] (Рисунок 2.58).

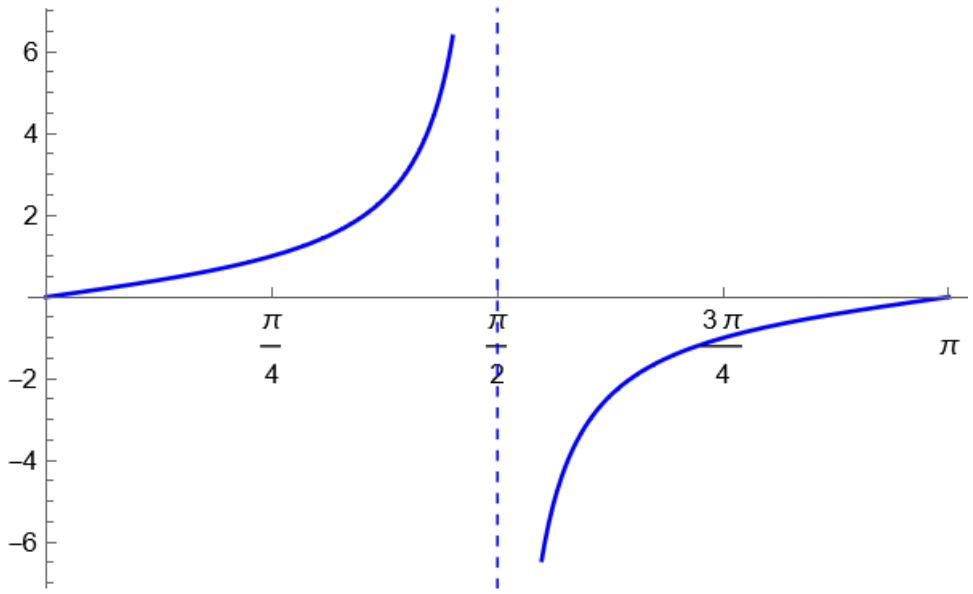


Рисунок 2.58 – Зависимость углового коэффициента от угла поворота от положительного направления оси абсцисс к заданному вектору

Итак, векторы следуют в таком порядке: $\overrightarrow{A_7A_1}$ вместе с $\overrightarrow{A_7A_3}$ (у них равные угловые коэффициенты), затем все векторы из II квадранта: $\overrightarrow{A_7A_6}$, $\overrightarrow{A_7A_4}$, $\overrightarrow{A_7A_5}$, $\overrightarrow{A_7A_2}$. Из векторов, имеющих равные угловые коэффициенты, остаётся вектор максимальной длины, а остальные можно отбросить – соответствующие точки обязательно окажутся внутренними или граничными для искомой выпуклой оболочки. В нашем примере с учётом всех этих соображений в итоге получается последовательность, обратная списку с рисунка 2.57б: $\overrightarrow{A_7A_1}$, $\overrightarrow{A_7A_6}$, $\overrightarrow{A_7A_4}$, $\overrightarrow{A_7A_5}$, $\overrightarrow{A_7A_2}$.

По полученному отсортированному списку векторов легко восстанавливается список точек, которые можно соединить самонепересекающейся ломаной линией: от конца первого вектора A_1 проводится ребро к точке A_6, затем от этой точки – к точке A_5 и так далее ко всем точкам – концам векторов построенной последовательности (по способу I – справа налево, по способу II – слева направо, при построении выпуклой

оболочки с правосторонним обходом – всё наоборот) (Рисунок 2.59). Получается последовательность вершин для искомой выпуклой оболочки. При этом, в отличие от алгоритма Джарвиса, вершины могут уже быть отброшены из новой последовательности. Редактирование списка вершин происходит по следующему алгоритму:

1. Начинается рассмотрение списка со второй вершины.
2. Вычисляется направление поворота в текущей вершине, для чего необходимо вычислить знак угла между двумя векторами: проведённым от предыдущей вершины к текущей и от текущей вершины к следующей. Благоприятным является случай, когда **угол оказывается положительным**, что соответствует повороту налево – **при левостороннем обходе**; либо когда **угол отрицательный**, или происходит поворот направо – **при правостороннем обходе**.
3. Если имеет место поворот не в ту сторону (или же угол оказался развёрнутым, т.е. знак угла равен нулю, и в вершине никакого поворота не происходит), то текущая вершина удаляется из списка, происходит переход к предыдущей вершине этого списка. В противном случае происходит переход к следующей вершине.
4. При переходе к первой вершине автоматически происходит переход к следующей, второй вершине.

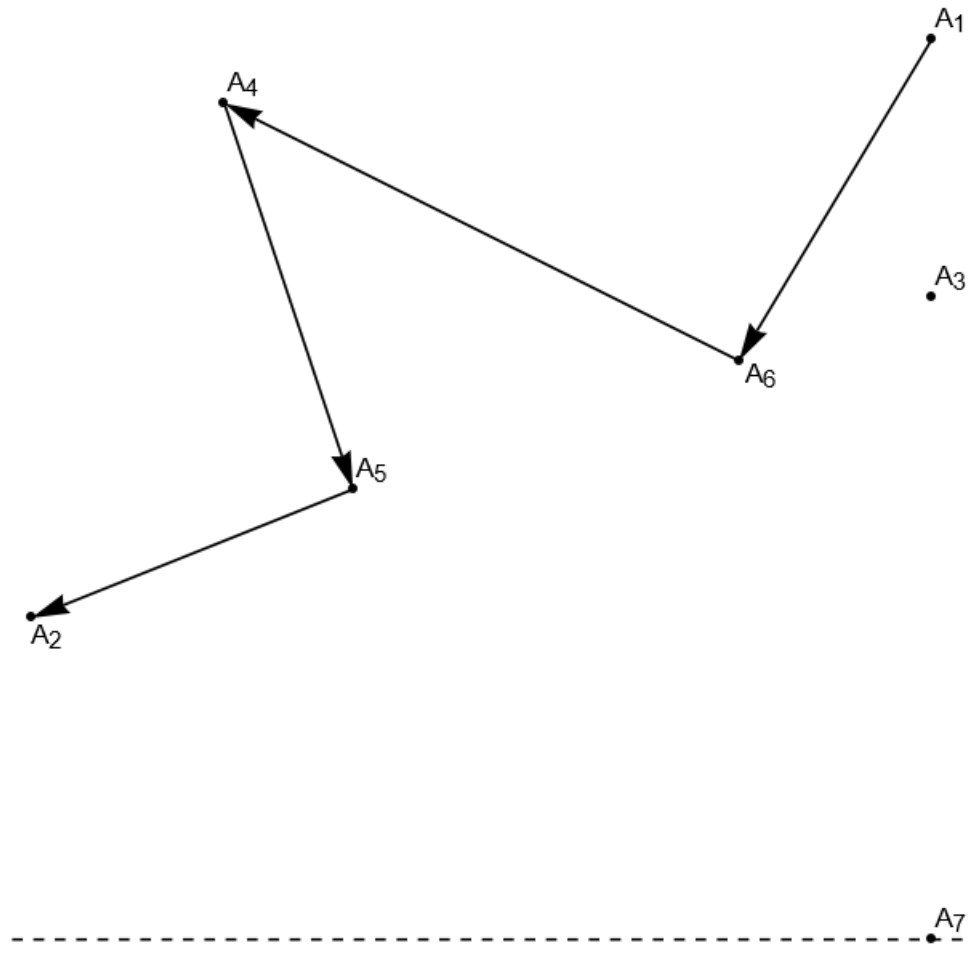


Рисунок 2.59 – Текущие вершины и рёбра выпуклой оболочки, полученные в результате сортировки векторов, проведённых из крайней вершины

5. Шаги 2–4 повторяются до тех пор, пока не произошёл переход к последней вершине. Тогда в последовательности останутся окончательные вершины выпуклой оболочки. Сюда в конец необходимо также добавить крайнюю точку, выбранную на самом первом шаге алгоритма Грэхема.

Итак, в последовательности $A_1A_6A_4A_5A_2$ первоначально проверяется вершина A_6:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_6}, \overrightarrow{A_6A_4}\right) = \operatorname{sgn}\begin{vmatrix} -3 & -5 \\ -8 & 4 \end{vmatrix} = \operatorname{sgn}(-12-40) = -1.$$

Значит, в вершине A_6 происходит поворот направо. Так как нужно построить выпуклую оболочку с левосторонним обходом, эту вершину нужно удалить и перейти к предыдущей (Рисунок 2.60). С учётом того, что первая вершина заведомо окажется среди вершин окончательной выпуклой оболочки, на следующей итерации рассматривается вторая вершина нового списка $A_1A_4A_5A_2$.

Для второй вершины получаются следующие вычисления:

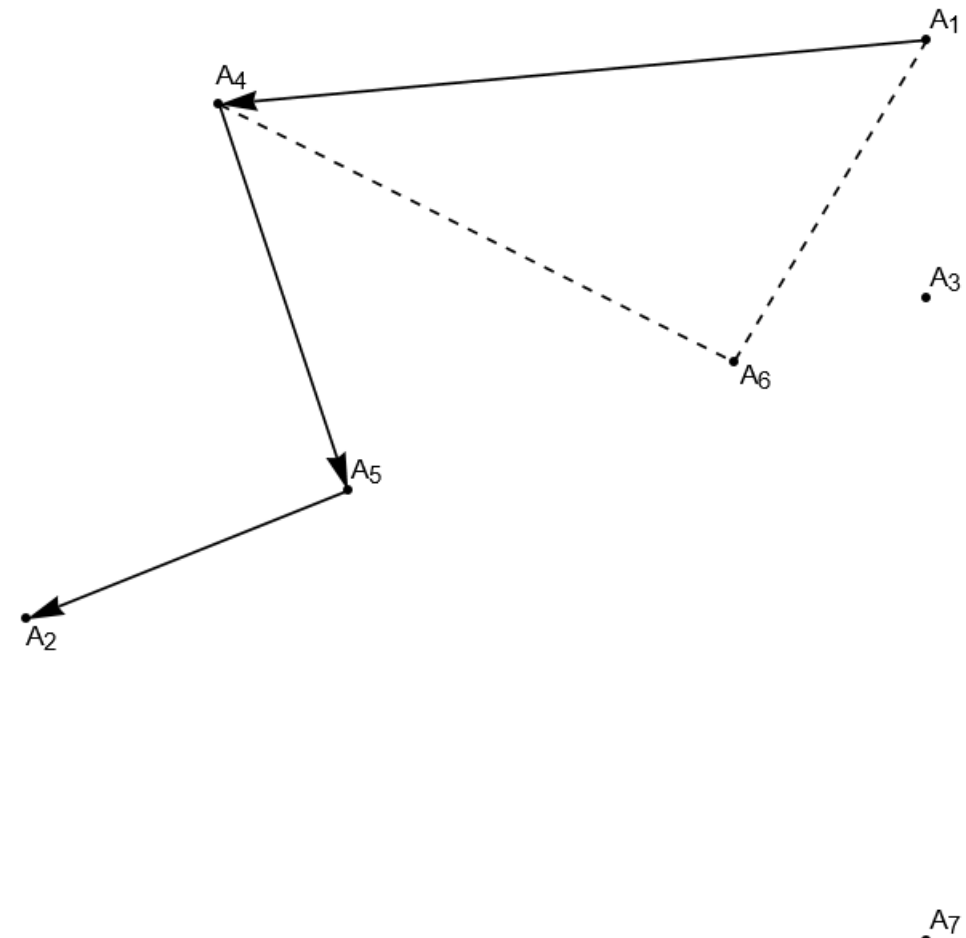


Рисунок 2.60 – Удаление точки из списка вершин выпуклой оболочки

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_4},\overrightarrow{A_4A_5}\right)=\operatorname{sgn}\begin{vmatrix}-11 & -1\\ 2 & -6\end{vmatrix}=\operatorname{sgn}(66+2)=+1.$$

Таким образом, вершина A_4 пока остаётся, и происходит переход к вершине A_5. Для этой вершины проверка даёт вот такой результат:

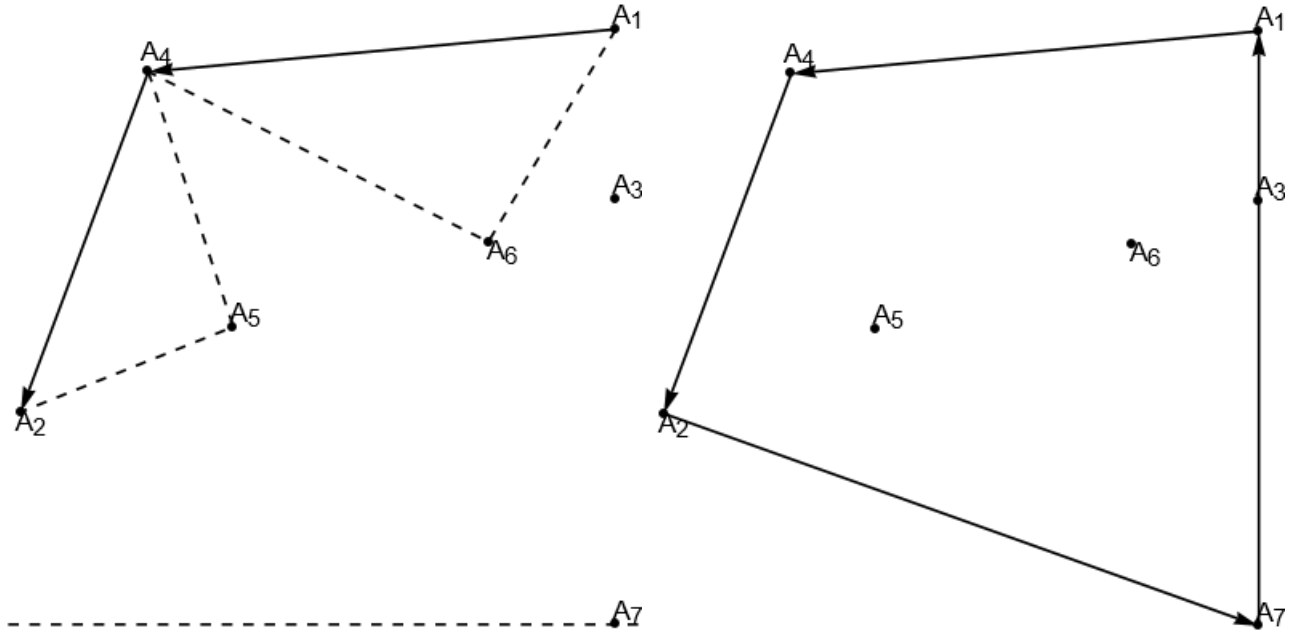
$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_5},\overrightarrow{A_5A_2}\right)=\operatorname{sgn}\begin{vmatrix}2 & -6\\ -5 & -2\end{vmatrix}=\operatorname{sgn}(-4-30)=-1.$$

Значит, точка A_5 окажется внутри выпуклой оболочки, и её необходимо удалить из списка вершин (Рисунок 2.61а). После этого переходим обратно к вершине A_4, у которой в новом списке $A_1A_4A_2$ оказывается новый сосед спереди:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_4},\overrightarrow{A_4A_2}\right)=\operatorname{sgn}\begin{vmatrix}-11 & -1\\ -3 & -8\end{vmatrix}=\operatorname{sgn}(88-3)=+1.$$

После поворота влево в точке A_4 попадаем в A_2 – конечную точку полученного списка вершин искомой выпуклой оболочки. Осталось только добавить точку A_7, и получится полигон $A_7A_1A_4A_2A_7$ (Рисунок 2.61б).

Замечание. На втором этапе вместо последовательности вершин можно рассматривать последовательность рёбер, представленных в виде векторов. На



а б

Рисунок 2.61 – Заключительные шаги алгоритма Грэхема

каждом шаге этого этапа проверяется знак угла между двумя соседними векторами, и если этот знак оказался неудовлетворительным, то два вектора заменяются на их сумму. При этом первый и последний векторы в этом списке (соответствующие рёбрам, проводимым к крайней вершине, выбранной в самом начале) редактированию не подлежат, так как они заведомо окажутся рёбрами окончательной выпуклой оболочки.

2.4.3. Алгоритм Эндрю

Алгоритм Эндрю (подраздел 1.4.3), как и алгоритм Грэхема, опирается на сортировку точек, но благодаря некоторым дополнительным действиям позволяет это делать при помощи простых и интуитивно понятных приёмов.

Пример 4.3. Построить выпуклую оболочку множества точек $\{A_i\}_{i=1}^{7}$ с координатами $A_1(-8,-4)$, $A_2(-9,4)$, $A_3(7,9)$, $A_4(10,2)$, $A_5(5,6)$, $A_6(1,-5)$, $A_7(6,6)$.

В этом алгоритме всё множество точек делится прямой, проведённой через две точки этого множества: одна имеет минимальную абсциссу (если таких точек несколько, то обычно из них выбирают точку с наименьшей ординатой), другая – максимальную абсциссу (и максимальную ординату, если таких точек несколько). В данном примере проводится прямая A_2A_4, которая задаётся уравнением $\frac{x-x_2}{x_4-x_2}=\frac{y-y_2}{y_4-y_2}\Rightarrow\frac{x+9}{19}=\frac{y-4}{-2}\Rightarrow -2x-19y+58=0.$ Для остальных точек проверяется, находятся ли они выше либо ниже этой прямой, путём подстановки их координат в левую часть этого уравнения: если получилось число, большее нуля, значит, текущая точка находится в той полуплоскости, куда проводится нормальный вектор (он восстанавливается по коэффициентам полученного уравнения прямой, в данном случае это $\vec{n}(-2,-19)$) при его откладывании от некоторой точки прямой. В нашем примере ордината вектора $\vec{n}$ меньше нуля, значит, если для точки с координатами (x_0,y_0) получится $-2x_0-19y_0+58>0,$ то эта точка находится ниже прямой A_2A_4, если же $-2x_0-19y_0+58<0$ – выше. Если точка окажется лежащей на проведённой прямой, то такая точка является внутренней для искомой выпуклой оболочки, и её можно дальше не рассматривать.

$$A_1:-2(-8)-19(-4)+58=16+56+58>0,$$
$$A_3:-2\cdot 7-19\cdot 9+58=-14-171+58<0,$$
$$A_5:-2\cdot 5-19\cdot 6+58=-10-114+58<0,$$
$$A_6:-2\cdot 1-19(-5)+58=-2+95+58>0,$$
$$A_7:-2\cdot 6-19\cdot 6+58=-12-114+58<0.$$

Таким образом, имеем, что точки A_1 и A_6 лежат ниже прямой A_2A_4, а точки A_3, A_5, A_7 – выше (Рисунок 2.62).

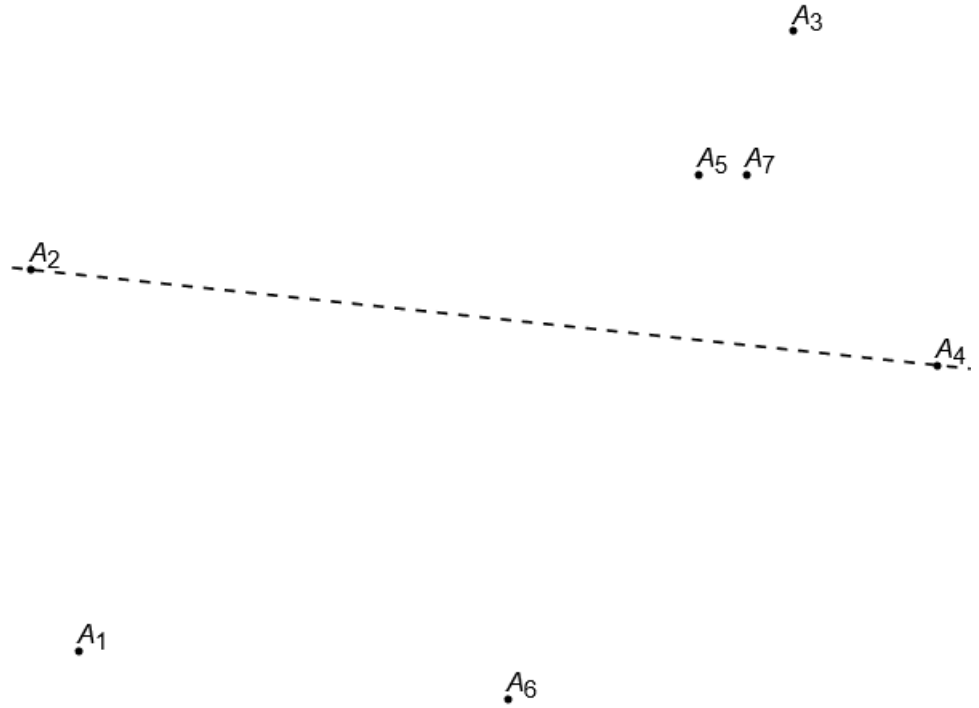


Рисунок 2.62 – Первоначальное определение расположения точек относительно друг друга в ходе алгоритма Эндрю

Далее строятся отдельно две выпуклые оболочки для точек, расположенных не выше разделяющей прямой, и точек, расположенных не ниже неё. Вначале построим выпуклую оболочку точек из верхней полуплоскости, для чего рассмотрим множество $\{A_3, A_5, A_7, A_2, A_4\}$.

Ясно, что выпуклая оболочка множества точек из верхней полуплоскости содержит ребро A_2A_4. Остальные получатся при некотором обходе всех точек рассматриваемого множества. Порядок обхода можно определить аналогично тому, как это было проделано в алгоритме Грэхема, ***только с некоторыми оговорками:***

1. Если нужно построить левосторонний обход, то в верхней полуплоскости в качестве начальной точки выбирается правая точка на разделяющей прямой, если нужен правосторонний обход – то левая.
2. Сортировка точек происходит **по их абсциссам**. В верхней полуплоскости при выборе левостороннего обхода точки сортируются в порядке убывания их абсцисс, при правостороннем обходе – в порядке их возрастания.
3. Если некоторые точки имеют равные абсциссы, то из них можно отбросить все, кроме точки с максимальной ординатой (кроме исходных двух точек, через которые проведена разделяющая прямая).
4. Для нижней полуплоскости пункты 1–3 выполняются наоборот (подчёркнутые слова заменяются на противоположные по смыслу).

В данном примере при выборе левостороннего обхода получится последовательность точек $A_4A_3A_7A_5A_2$ (Рисунок 2.63). Далее осуществляется обход, как в алгоритме Грэхема.

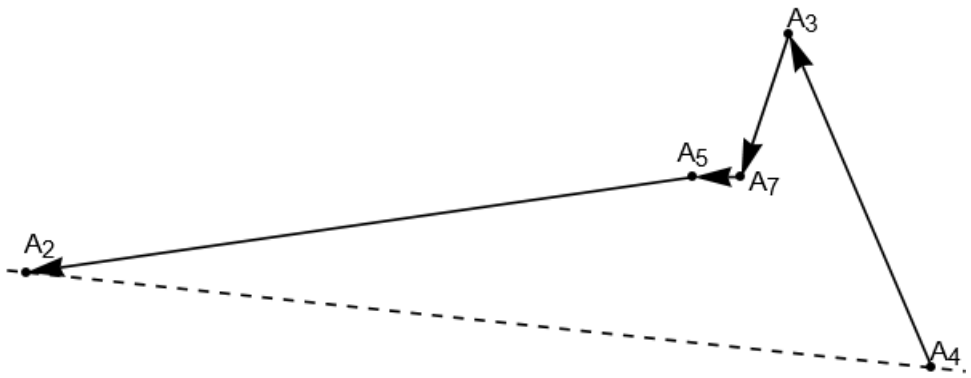


Рисунок 2.63 – Начало построения выпуклой оболочки точек в верхней полуплоскости

В вершине A_3 имеет место поворот налево, как и полагается при левостороннем обходе:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_3},\overrightarrow{A_3A_7}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 7\\ -1 & -3\end{vmatrix}=1.$$

Значит, затем переходим к следующей вершине A_7:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_7},\overrightarrow{A_7A_5}\right)=\operatorname{sgn}\begin{vmatrix}-1 & -3\\ -1 & 0\end{vmatrix}=-1.$$

Здесь уже происходит поворот направо, что означает, что вершина A_7 подлежит удалению из списка (Рисунок 2.64). После этого возвращаемся к вершине A_3:

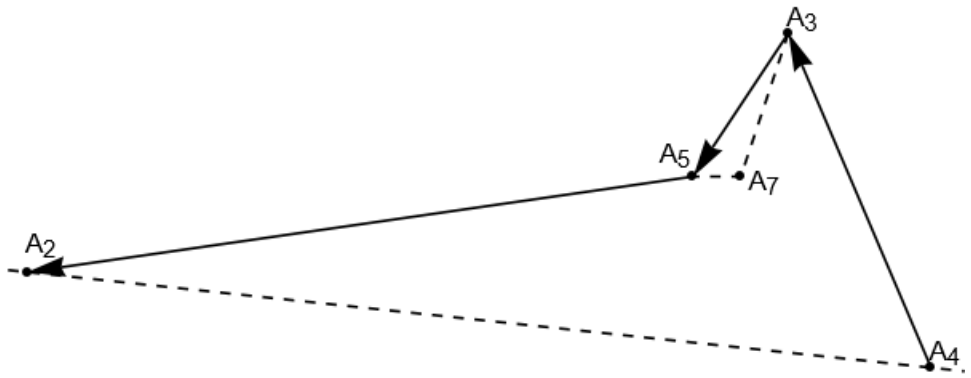


Рисунок 2.64 – Удаление вершины в ходе алгоритма Эндрю

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_3},\overrightarrow{A_3A_5}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 7\\ -2 & -3\end{vmatrix}=1.$$

Значит, в вершине A_3 после удаления одной из вершин по-прежнему происходит поворот влево, и можно переходить к следующей точке последовательности, которой уже является A_5:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_5},\overrightarrow{A_5A_2}\right)=\operatorname{sgn}\begin{vmatrix}-2 & -3\\ -14 & -2\end{vmatrix}=-1.$$

Значит, точку A_5 нужно удалить, и останется список из трёх вершин $A_4A_3A_2$ (Рисунок 2.65). Для трёх вершин необязательно проверять знак угла[1], так как три точки всегда образуют выпуклый полигон, поэтому для верхней полуплоскости получаем список $A_4A_3A_2$.

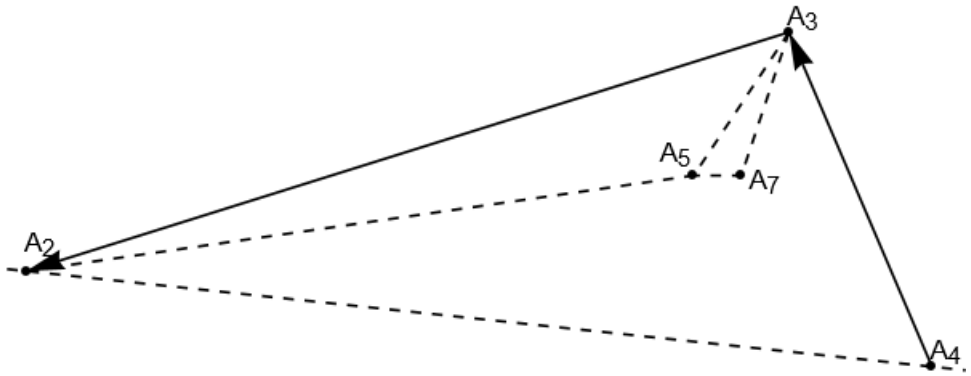


Рисунок 2.65 – Итоговая выпуклая оболочка точек из верхней полуплоскости

Теперь рассмотрим нижнюю полуплоскость, в которой лежат точки A_1 и A_6. Здесь нужно сортировать точки уже в порядке возрастания их абсцисс: $A_2A_1A_6A_4$. Здесь поступаем аналогично рассмотрению верхней полуплоскости:

$$A_1:\overrightarrow{A_2A_1}(1,-8),\overrightarrow{A_1A_6}(9,-1),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_2A_1},\overrightarrow{A_1A_6}\right)=\begin{vmatrix}1 & -8\\ 9 & -1\end{vmatrix}=1,$$
$$A_6:\overrightarrow{A_1A_6}(9,-1),\overrightarrow{A_6A_4}(9,7),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_6},\overrightarrow{A_6A_4}\right)=\begin{vmatrix}9 & -1\\ 9 & 7\end{vmatrix}=1.$$

[1] Здесь проявляется отличие от алгоритма Грэхема, где возможна ситуация, когда в списке вершин остаются только две точки, которые дополняются третьей точкой на самом последнем шаге

Исходя из этих вычислений легко видеть, что при обходе этого списка не нужно удалять никакую вершину. Значит, получится обход $A_2A_1A_6A_4$, дающий левосторонний обход выпуклой оболочки множества точек нижней полуплоскости. Объединяя обе выпуклые оболочки $A_4A_3A_2$ и $A_2A_1A_6A_4$, окончательно получим выпуклую оболочку исходного множества всех точек $A_4A_3A_2A_1A_6A_4$ (Рисунок 2.66).

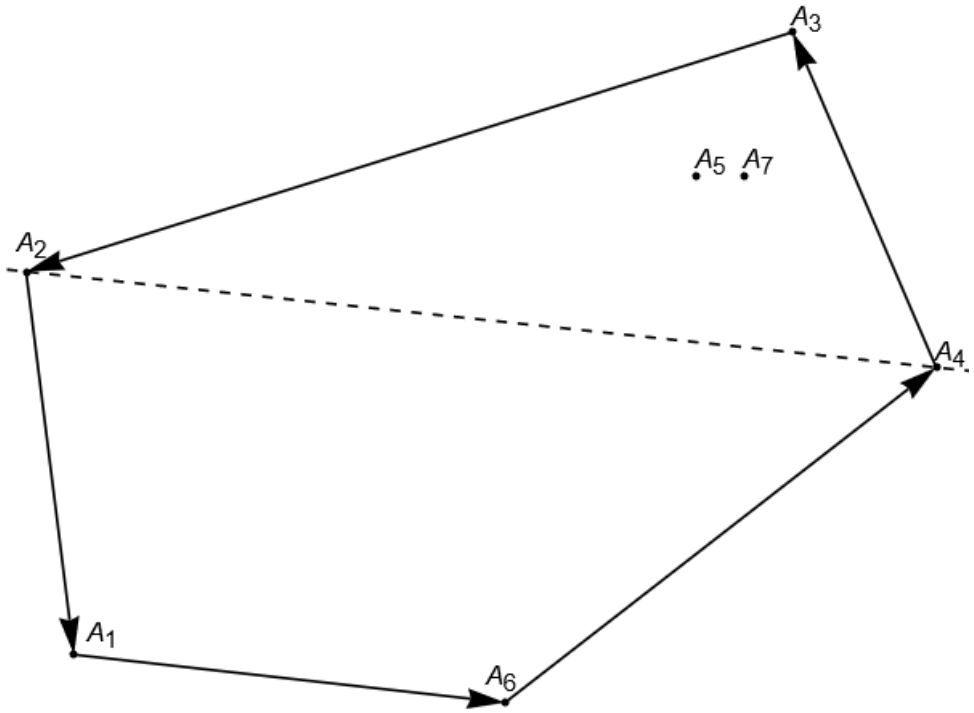


Рисунок 2.66 – Объединение двух выпуклых оболочек

2.4.4. Алгоритм быстрой оболочки

Алгоритм быстрой оболочки (подраздел 1.4.4), как и алгоритм Эндрю, рассматривает отдельно точки из верхней и нижней полуплоскости относительно некоторой секущей прямой. Преимуществом данного алгоритма является то, что уже после первых пару итераций зачастую получается довольно точное приближение выпуклой оболочки, что может оказаться решающим при обработке большого числа точек.

Пример 4.4. Построить выпуклую оболочку множества точек $\{A_i\}_{i=1}^{7}$ с координатами $A_1(2,-4)$, $A_2(-3,-5)$, $A_3(-10,-7)$, $A_4(-3,-8)$, $A_5(3,-3)$, $A_6(-4,0)$, $A_7(10,8)$.

Согласно алгоритму быстрой оболочки, вначале следует разделить всё множество точек прямой, как при алгоритме Эндрю. В данном примере получится прямая A_3A_7, которая описывается уравнением $3x-4y+2=0$. Определим положение остальных точек относительно неё:

$$
\begin{aligned}
A_1 &: 3\cdot 2 - 4(-4) + 2 = 6 + 16 + 2 = 24 > 0,\\
A_2 &: 3(-3) - 4(-5) + 2 = -9 + 20 + 2 = 13 > 0,\\
A_4 &: 3(-3) - 4(-8) + 2 = -9 + 32 + 2 = 25 > 0,\\
A_5 &: 3\cdot 3 - 4(-3) + 2 = 9 + 12 + 2 = 23 > 0,\\
A_6 &: 3(-4) - 4\cdot 0 + 2 = -12 + 2 = -10 < 0.
\end{aligned}
\qquad (2.25)
$$

Таким образом, точка A_6 находится выше прямой A_3A_7, остальные – ниже. В верхней полуплоскости наиболее удалённой точкой от прямой A_3A_7 является A_6, а в нижней, согласно вычислениям (2.25), – точка A_4. Поэтому необходимо провести по паре рёбер от каждой из этих точек к крайним точкам, через которые провели разделяющую прямую: в верхней полуплоскости это A_7A_6 и A_6A_3, в нижней – A_3A_4 и A_4A_7 (при выборе левостороннего обхода для искомой выпуклой оболочки, рисунок 2.67).

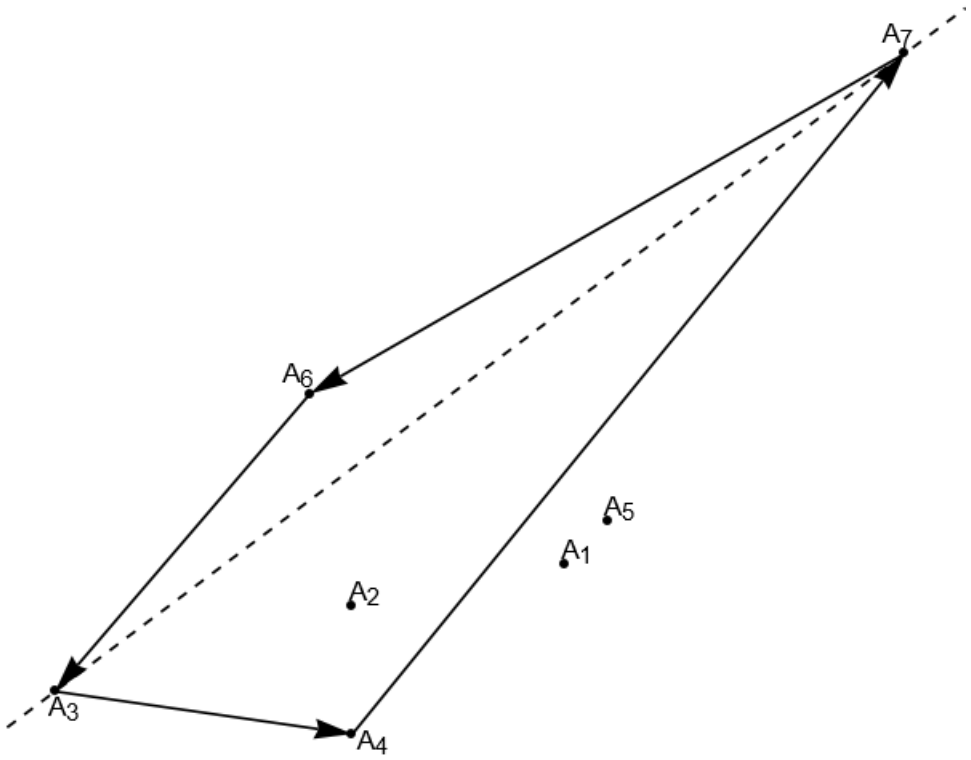


Рисунок 2.67 – Проведение начальных рёбер по алгоритму быстрой оболочки

Дальнейшие этапы алгоритма выглядят следующим образом:

1. Для каждой из пар рёбер и соответствующих им оставшихся точек определяется, какие из этих точек расположены по левую сторону от этих рёбер. Такие точки являются внутренними и могут дальше не рассматриваться. Остальные лежат по правую сторону **ровно от одного** из пары рёбер. При вычислении выпуклой оболочки с правосторонним обходом подчёркнутые слова следует поменять на противоположные по смыслу.

2. Если для ребра не оказалось точек, лежащих от него по правую сторону, то это ребро являются окончательным и далее не рассматривается.

3. Если же у ребра оказалась хотя бы одна точка, лежащая от него по правую сторону, то среди таких точек выбирается наиболее удалённая, и от неё к концам текущих рёбер проводятся два новых ребра, а текущее ребро удаляется.

4. Таким образом, вместо одного ребра получается пара рёбер со множеством точек, лежащим от исходного ребра по правую сторону. К этой паре рёбер и этим точкам повторно применяются шаги 1–3.

5. Алгоритм заканчивается, когда больше не остаётся внешних точек.

Так, для пары рёбер в верхней полуплоскости не остаётся точек, которые потенциально могли бы лежать справа от них. Значит, рёбра A_7A_6 и A_6A_3 – окончательные (Рисунок 2.68).

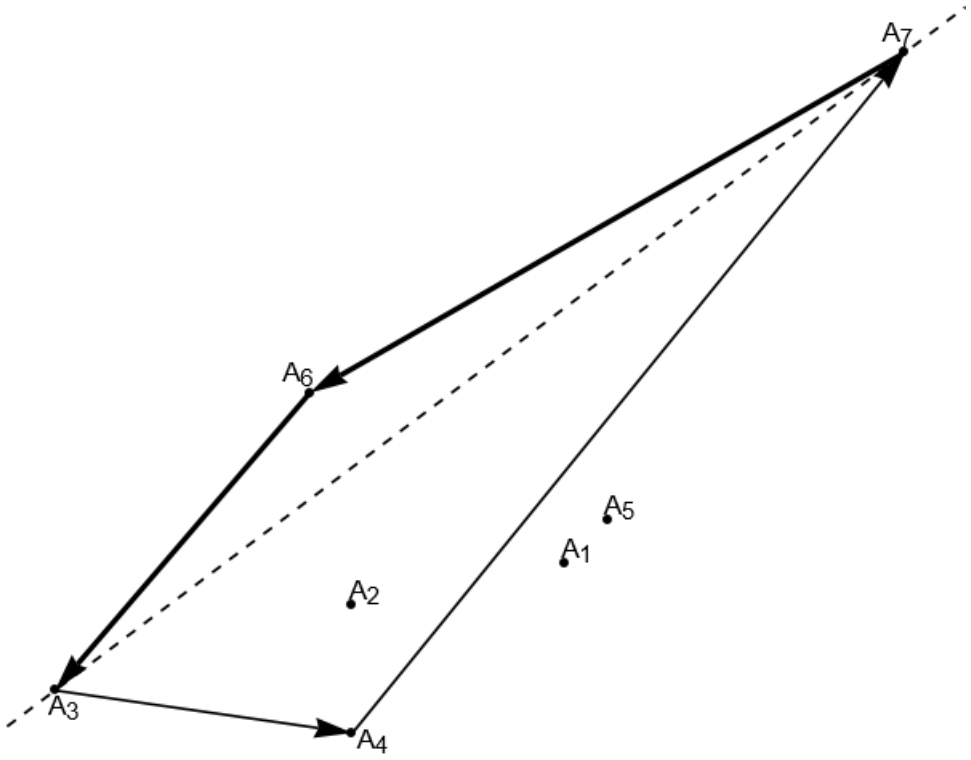


Рисунок 2.68 – Выделение рёбер, не подлежащих редактированию

Для пары рёбер из нижней же полуплоскости A_3A_4 и A_4A_7 остаются точки A_1, A_2 и A_5. Их расположение относительно рёбер следующее:

$$\overrightarrow{A_3A_4}(7,-1),$$

$$A_1 : \operatorname{sgn}\angle\left(\overrightarrow{A_3A_4}, \overrightarrow{A_3A_1}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -1 \\ 12 & 3 \end{vmatrix} = \operatorname{sgn}(33) = +1 \Rightarrow \text{внутри},$$

$$A_2 : \operatorname{sgn} \angle\left(\overrightarrow{A_3A_4}, \overrightarrow{A_3A_2}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -1 \\ 7 & 2 \end{vmatrix} = \operatorname{sgn}(21) = +1 \Rightarrow \text{внутри},$$

$$A_5 : \operatorname{sgn} \angle\left(\overrightarrow{A_3A_4}, \overrightarrow{A_3A_5}\right) = \operatorname{sgn}\begin{vmatrix} 7 & -1 \\ 13 & 4 \end{vmatrix} = \operatorname{sgn}(41) = +1 \Rightarrow \text{внутри},$$

$$\overrightarrow{A_4A_7}(13,16),$$

$$A_1 : \operatorname{sgn} \angle\left(\overrightarrow{A_4A_7}, \overrightarrow{A_4A_1}\right) = \operatorname{sgn}\begin{vmatrix} 13 & 16 \\ 5 & 4 \end{vmatrix} = \operatorname{sgn}(-28) = -1 \Rightarrow \text{снаружи}, \qquad (2.26)$$

$$A_2 : \operatorname{sgn} \angle\left(\overrightarrow{A_4A_7}, \overrightarrow{A_4A_2}\right) = \operatorname{sgn}\begin{vmatrix} 13 & 16 \\ 0 & 3 \end{vmatrix} = \operatorname{sgn}(39) = +1 \Rightarrow \text{внутри},$$

$$A_5 : \operatorname{sgn} \angle\left(\overrightarrow{A_4A_7}, \overrightarrow{A_4A_5}\right) = \operatorname{sgn}\begin{vmatrix} 13 & 16 \\ 6 & 5 \end{vmatrix} = \operatorname{sgn}(-31) = -1 \Rightarrow \text{снаружи}. \qquad (2.27)$$

Итак, получили, что для ребра A_3A_4 нет точек, находящихся по наружную от него сторону – это ребро окончательное; а у ребра A_4A_7 есть точки A_1 и A_5, лежащие снаружи него. Из них для проведения новых рёбер выбирается A_5, так как она расположена дальше (значение определителя для точки A_5, полученное при подсчётах (2.27), превышает по модулю аналогичное значение из вычислений (2.26) для точки A_1), и получается новая пара рёбер A_4A_5 и A_5A_7, которым соответствует точка A_1, лежащая снаружи от исходного ребра A_4A_7 (Рисунок 2.69).

Остаётся только одна пара рёбер A_4A_5, A_5A_7 и точка A_1. Определим положение этой точки относительно них:

$$\overrightarrow{A_4A_5}(6,5),$$

$$A_1 : \operatorname{sgn} \angle\left(\overrightarrow{A_4A_5}, \overrightarrow{A_4A_1}\right) = \operatorname{sgn}\begin{vmatrix} 6 & 5 \\ 5 & 4 \end{vmatrix} = -1 \Rightarrow \text{снаружи}.$$

Пользуясь тем, что точка A_1 может лежать снаружи только относительно одного из рёбер, можем ребро A_4A_5 сразу заменить на два новых ребра A_4A_1 и A_1A_5 (Рисунок 2.70). После этого точек, лежащих снаружи от какого-то ребра, больше не остаётся, поэтому представленная на рисунке 2.70 оболочка является окончательной.

Замечание. Если для некоторого ребра оказалось больше одной максимально удалённой точки от него, то рекомендуется выбирать какую-нибудь крайнюю из них (например, имеющую наименьшую абсциссу) во избежание получения рёбер, лежащих на одной прямой. На рисунке 2.71 номерами указан порядок рисования рёбер в зависимости от выбора одной из наиболее удалённых точек на самой первой итерации.

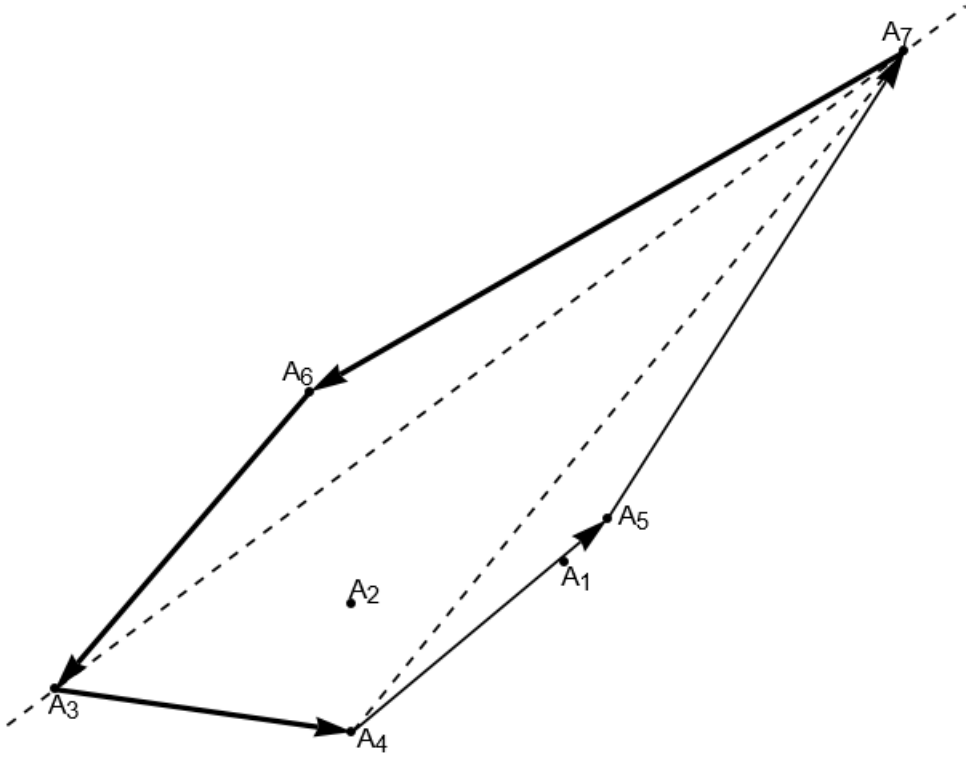


Рисунок 2.69 – Редактирование рёбер в ходе алгоритма быстрой оболочки

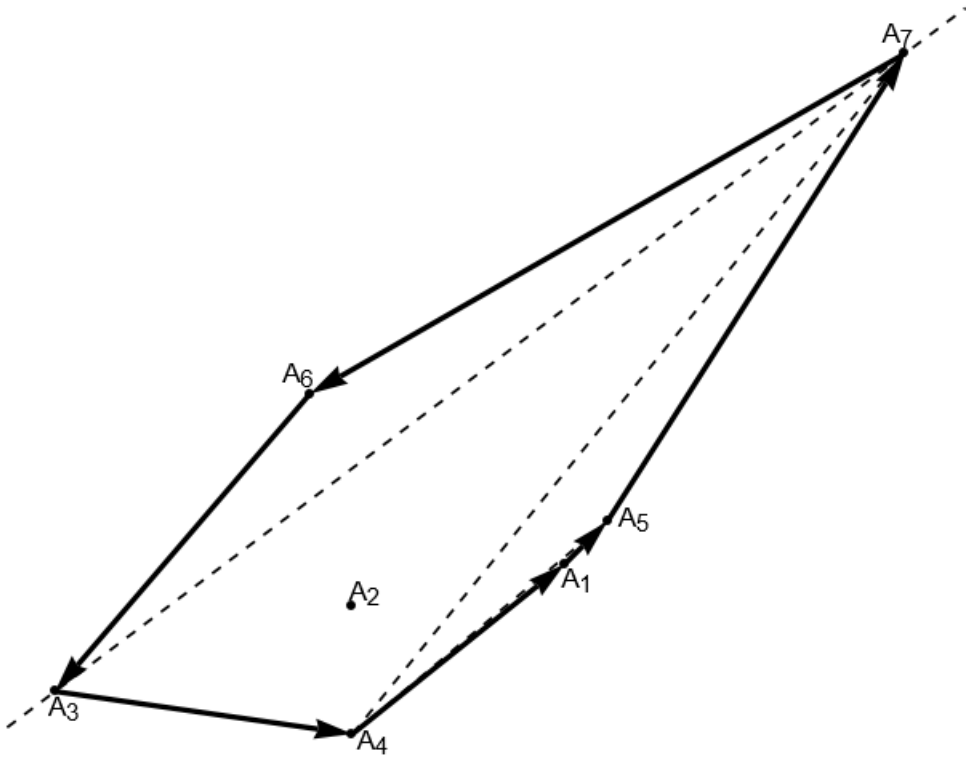


Рисунок 2.70 – Итоговая выпуклая оболочка, вычисленная
по алгоритму быстрой оболочки

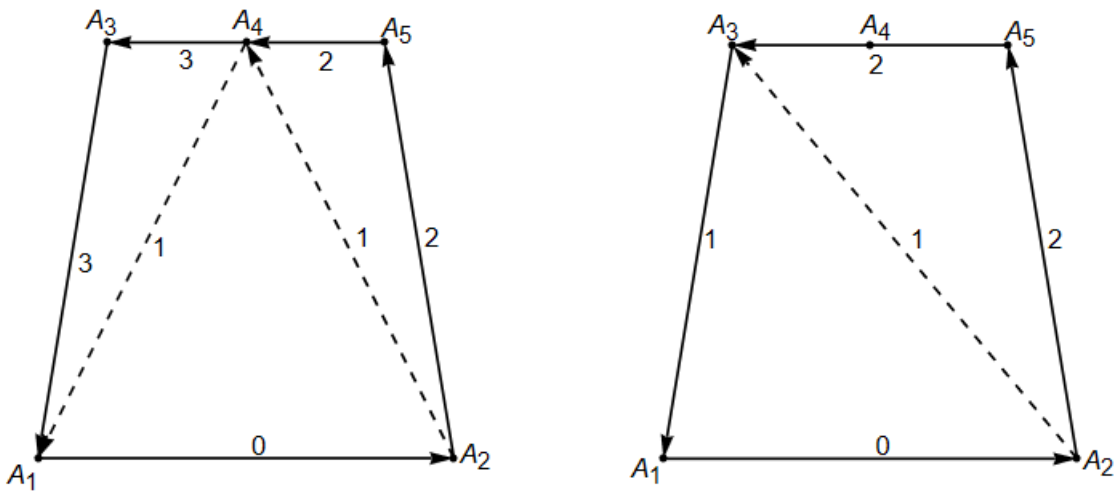


Рисунок 2.71 – Варианты выполнения алгоритма выпуклой оболочки при наличии равноудалённых от некоторого ребра точек

2.4.5. Задачи

Построить выпуклую оболочку заданного множества точек.

1. $A_1(7,6)$, $A_2(8,4)$, $A_3(3,-4)$, $A_4(-3,0)$, $A_5(-2,3)$, $A_6(2,0)$, $A_7(-6,-4)$ (перейти к ответу),
2. $A_1(9,-8)$, $A_2(10,-4)$, $A_3(10,2)$, $A_4(-2,-1)$, $A_5(9,2)$, $A_6(-2,-2)$, $A_7(-5,-10)$ (перейти к ответу),
3. $A_1(6,10)$, $A_2(10,2)$, $A_3(8,-10)$, $A_4(5,-1)$, $A_5(-4,0)$, $A_6(1,-8)$, $A_7(-10,-5)$ (перейти к ответу),
4. $A_1(9,-9)$, $A_2(8,-4)$, $A_3(9,5)$, $A_4(9,10)$, $A_5(6,5)$, $A_6(-4,0)$, $A_7(1,0)$ (перейти к ответу),
5. $A_1(6,0)$, $A_2(0,0)$, $A_3(9,6)$, $A_4(-9,-9)$, $A_5(6,8)$, $A_6(-6,-9)$, $A_7(-6,0)$ (перейти к ответу),
6. $A_1(4,-1)$, $A_2(-3,1)$, $A_3(2,-1)$, $A_4(7,-9)$, $A_5(6,-8)$, $A_6(10,-3)$, $A_7(-6,-3)$ (перейти к ответу),
7. $A_1(9,-8)$, $A_2(-7,1)$, $A_3(-10,7)$, $A_4(-7,0)$, $A_5(-9,7)$, $A_6(2,2)$, $A_7(6,-4)$ (перейти к ответу),
8. $A_1(-10,9)$, $A_2(-5,-3)$, $A_3(0,-10)$, $A_4(6,1)$, $A_5(-7,2)$, $A_6(-2,2)$, $A_7(-8,-1)$ (перейти к ответу),
9. $A_1(-10,4)$, $A_2(-6,8)$, $A_3(-6,10)$, $A_4(-5,-1)$, $A_5(-5,1)$, $A_6(-4,5)$, $A_7(-2,7)$, $A_8(0,3)$, $A_9(1,2)$, $A_{10}(3,4)$ (перейти к ответу),
10. $A_1(9,8)$, $A_2(10,3)$, $A_3(-1,9)$, $A_4(-10,10)$, $A_5(8,-1)$, $A_6(-9,-4)$, $A_7(-10,-8)$, $A_8(-8,-8)$, $A_9(10,-8)$, $A_{10}(10,10)$ (перейти к ответу).

2.5. Триангуляция полигонов

2.5.1. Триангуляция диагоналями

Для построения минимальной триангуляции полигона с n вершинами, т.е. его триангуляции без вычисления дополнительных вершин, достаточно провести некоторые $n-3$ диагонали. Алгоритм триангуляции самонепересекающегося полигона диагоналями (стр. 150) опирается на тот факт, что изо всякой вогнутой его вершины можно провести по крайней мере одну диагональ.

Пример 5.1. Осуществить триангуляцию диагоналями полигона с вершинами в точках с координатами $A_1(8,-8)$, $A_2(10,0)$, $A_3(7,3)$, $A_4(-5,10)$, $A_5(-5,1)$, $A_6(-4,0)$, $A_7(-7,-1)$, $A_8(-3,-5)$, $A_9(-4,-2)$.

Для того, чтобы выяснить, какие из вершин являются вогнутыми, а какие – выпуклыми, воспользуемся методом октантных углов, описанным на стр. 150. Определим октантные углы для направляющих векторов всех рёбер:

$$
\begin{aligned}
&\overrightarrow{A_1A_2}(2,8), && x>0, y>0, |x|<|y| && \Rightarrow I_1=2,\\
&\overrightarrow{A_2A_3}(-3,3), && x<0, y>0, |x|=|y| && \Rightarrow I_2=4,\\
&\overrightarrow{A_3A_4}(-12,7), && x<0, y>0, |x|>|y| && \Rightarrow I_3=4,\\
&\overrightarrow{A_4A_5}(0,-9), && x=0, y<0 && \Rightarrow I_4=7,\\
&\overrightarrow{A_5A_6}(1,-1), && x>0, y<0, |x|=|y| && \Rightarrow I_5=8,\\
&\overrightarrow{A_6A_7}(-3,-1), && x<0, y<0, |x|>|y| && \Rightarrow I_6=5,\\
&\overrightarrow{A_7A_8}(4,-4), && x>0, y<0, |x|=|y| && \Rightarrow I_7=8,\\
&\overrightarrow{A_8A_9}(-1,3), && x<0, y>0, |x|<|y| && \Rightarrow I_8=3,\\
&\overrightarrow{A_9A_1}(12,-6), && x>0, y<0, |x|>|y| && \Rightarrow I_9=8.
\end{aligned}
$$

Разности между октантными углами этих векторов указывают на величину угла поворота в той или иной вершине при обходе полигона: равные октантные углы означают, что в заданной вершине произошёл поворот на угол менее $\pi/4$, разность, равная ±1, свидетельствует о повороте на угол между $\pi/4$ и $\pi/2$ и т.д. Сумма этих углов равна $+2\pi$ при левостороннем обходе и -2π при правостороннем обходе. Следовательно, скорректированные разности в сумме могут дать либо +8, что эквивалентно левостороннему обходу, либо –8, что будет означать, что последовательность вершин обеспечивает правосторонний обход. В данном примере имеют место следующие вычисления:

$$\Delta_2 = I_2 - I_1 = 2, \Delta_3 = I_3 - I_2 = 0, \Delta_4 = I_4 - I_3 = 3, \Delta_5 = I_5 - I_4 = 1, \Delta_6 = I_6 - I_5 = -3,$$
$$\Delta_7 = I_7 - I_6 = 3, \Delta_8 = I_8 - I_7 = -5, \Delta_9 = I_9 - I_8 = 5, \Delta_1 = I_1 - I_9 = -6.$$

Корректировке подлежат разности, по модулю не меньшие 4, а также нулевые разности (т.к. необходимо определить знак угла при соответствующей вершине). Нулевая разность Δ_3 корректируется следующим образом:

$$\operatorname{sgn} \angle\left(\overrightarrow{A_2A_3}, \overrightarrow{A_3A_4}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 3 \\ -12 & 7 \end{vmatrix} = +1,$$

что означает поворот налево в вершине A_3. Этот факт можно условно обозначить $\Delta_3' = +0$. Корректируются также следующие разности: $\Delta_1' = 2$, $\Delta_8' = 3$, $\Delta_9' = -3$, для всех остальных разностей имеем $\Delta_i' = \Delta_i$. Сумма скорректированных разностей даёт

$$\Delta = \sum_{i=1}^{9} \Delta_i' = 2 + 2 + 0 + 3 + 1 - 3 + 3 + 3 - 3 = +8,$$

значит, полигон $A_1A_2A_3A_4A_5A_6A_7A_8A_9A_1$ имеет левосторонний обход, вершины, где происходит поворот налево, т.е. A_1, A_2, A_3, A_4, A_5, A_7 и A_8, являются выпуклыми, остальные, т.е. A_6 и A_9, – вогнутыми. На рисунке 2.72 выпуклые вершины отмечены синим цветом, вогнутые – красным.

Для проведения диагонали выберем какую-нибудь вогнутую вершину, например A_6. В общем случае диагональ A_iA_j должна удовлетворять двум условиям:

- при левостороннем обходе угол между направляющими векторами одного из смежных рёбер и диагонали должен быть положительным: $\operatorname{sgn} \angle\left(\overrightarrow{A_{i-1}A_i}, \overrightarrow{A_iA_j}\right) = +1$ или $\operatorname{sgn} \angle\left(\overrightarrow{A_iA_{i+1}}, \overrightarrow{A_iA_j}\right) = +1$ (а при правостороннем – отрицательным),

- диагональ не пересекает рёбра полигона, не инцидентные концам этой диагонали: $A_iA_j \cap A_kA_{k+1} = \varnothing,\ k \neq i-1,\ k \neq i,\ k \neq j-1,\ k \neq j.$

Вначале проведём отрезок A_6A_8 и проверим эти условия. Проверка первого условия даёт следующие вычисления:

$$\operatorname{sgn} \angle\left(\overrightarrow{A_6A_7}, \overrightarrow{A_6A_8}\right) = \operatorname{sgn}\begin{vmatrix} -3 & -1 \\ 1 & -5 \end{vmatrix} = +1.$$

Для проверки второго условия, например, рассматривая первоначально отрезки A_6A_8 и A_9A_1, проверим вначале необходимое **габаритное** условие

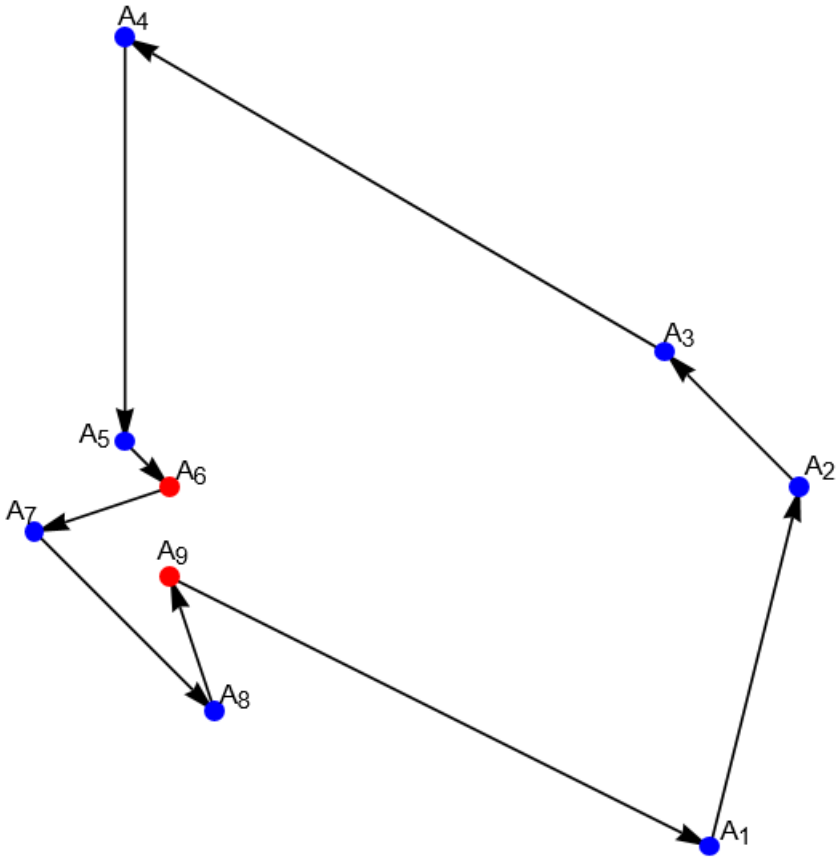


Рисунок 2.72 – Самонепересекающийся полигон и его выпуклые и вогнутые вершины

пересечения двух геометрических объектов: если пересечение двух геометрических множеств содержит по крайней мере одну точку, то их **габаритные прямоугольники** (т.е. минимальные прямоугольники, чьи стороны параллельны координатным осям и содержат эти множества) также должны пересекаться. Это условие позволяет сразу же отбросить рёбра полигона, заведомо не пересекающие проведённый отрезок. Для двух отрезков имеем следующие габаритные прямоугольники:

$$A_6A_8 : x^1_{\min} = \min\{x_6, x_8\} = -4, x^1_{\max} = \max\{x_6, x_8\} = -3,$$
$$y^1_{\min} = \min\{y_6, y_8\} = -5, y^1_{\max} = \max\{y_6, y_8\} = 0,$$
$$A_9A_1 : x^2_{\min} = \min\{x_9, x_1\} = -4, x^2_{\max} = \max\{x_9, x_1\} = 8,$$
$$y^2_{\min} = \min\{y_9, y_1\} = -8, y^2_{\max} = \max\{y_9, y_1\} = -2.$$

Числовые отрезки $[x^1_{\min}, x^1_{\max}]$ и $[x^2_{\min}, x^2_{\max}]$, $[y^1_{\min}, y^1_{\max}]$ и $[y^2_{\min}, y^2_{\max}]$ пересекаются, значит, пересекаются и габаритные прямоугольники. Проверим, пересекаются ли два отрезка:

$$\overrightarrow{A_6A_8}(1,-5)\perp(5,1),\qquad \overrightarrow{A_9A_1}(12,-6)\perp(1,2),$$
$$A_6A_8: 5x+y+20=0,\qquad A_9A_1: x+2y+8=0,$$
$$A_9: 5\cdot(-4)+(-2)+20<0,\qquad A_6: -4+2\cdot 0+8>0,$$
$$A_1: 5\cdot 8+(-8)+20>0,\qquad A_8: -3+2\cdot(-5)+8<0.$$

То есть точки A_9 и A_1 расположены по разные стороны от прямой A_6A_8, а точки A_6 и A_8 – по разные стороны от прямой A_9A_1. Это и означает пересечение двух отрезков. Отсюда следует, что отрезок A_6A_8 не образует диагонали исходного полигона.

Попробуем провести другой отрезок, например A_6A_9. Он расположен по левую сторону от ребра A_6A_7:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_6A_7},\overrightarrow{A_6A_9}\right)=\operatorname{sgn}\begin{vmatrix}-3 & -1\\ 0 & -2\end{vmatrix}=+1.$$

Его габаритный прямоугольник характеризуется значениями $x_{\min}=-4$, $x_{\max}=-4$, $y_{\min}=-2$, $y_{\max}=0$, или системой уравнения $x=-4$ и двойного неравенства $-2\le y\le 0$. Габаритный прямоугольник, скажем, ребра A_1A_2 равен $\{(x,y): 8\le x\le 10, -8\le y\le 0\}$. Нетрудно видеть, что два габаритных прямоугольника не пересекаются, так как ни для какой точки плоскости не могут выполняться одновременно равенство $x=-4$ и неравенство $8\le x\le 10$. Отсюда сразу же следует, что отрезки A_6A_9 и A_1A_2 не пересекаются. При помощи похожих рассуждений доказывается, что отрезок A_6A_9 не имеет общих точек с отрезками A_iA_{i+1}, $i=\overline{1,4}$, а вот с ребром A_7A_8 габаритное условие выполняется (Рисунок 2.73). Тем не менее, сами отрезки A_6A_9 и A_7A_8 не пересекаются:

$$\overrightarrow{A_7A_8}(4,-4)\perp(1,1),$$
$$A_7A_8: x+y+8=0,$$
$$A_6: -4+8<0,$$
$$A_9: -4-2+8<0.$$

Итак, отрезок A_6A_9 не пересекает никакие несмежные с ним рёбра исходного полигона, значит, образует его диагональ. Этой диагональю полигон делится на полигоны $A_6A_7A_8A_9A_6$ и $A_9A_1A_2A_3A_4A_5A_6A_9$ (Рисунок 2.74). В новых полигонах сохраняется направление обхода, и все выпуклые вершины исходного полигона остаются выпуклыми в них. Рассматривая четырёхугольник $A_6A_7A_8A_9A_6$, проверим, являются ли в нём вогнутыми вершины A_6 и A_9:

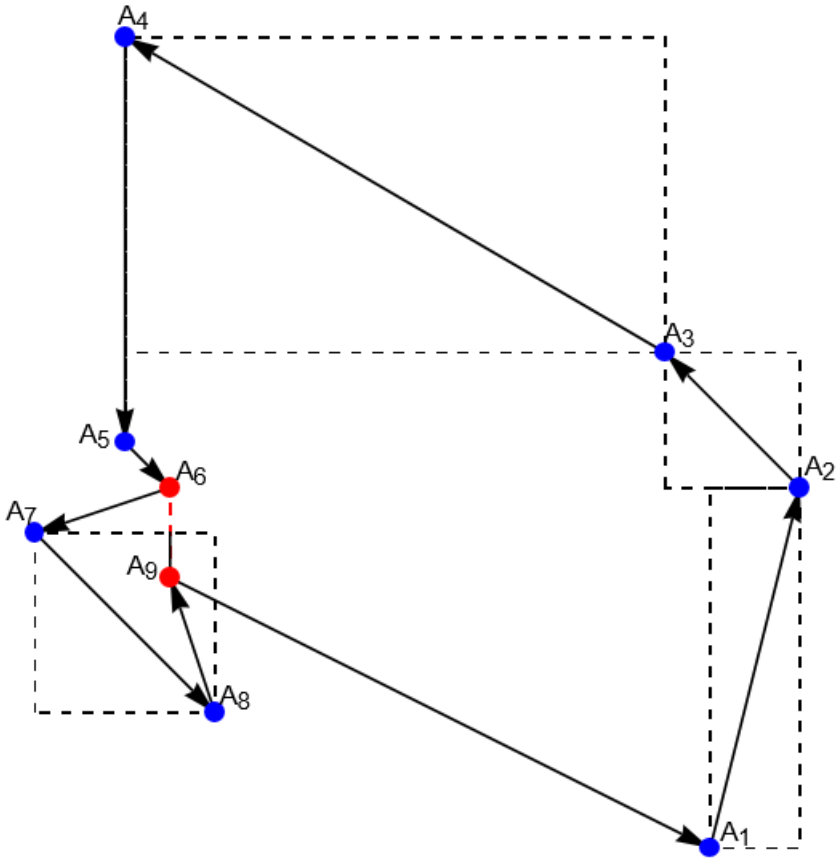


Рисунок 2.73 – Проверка габаритных условий пересечения проведённого отрезка с рёбрами исходного полигона

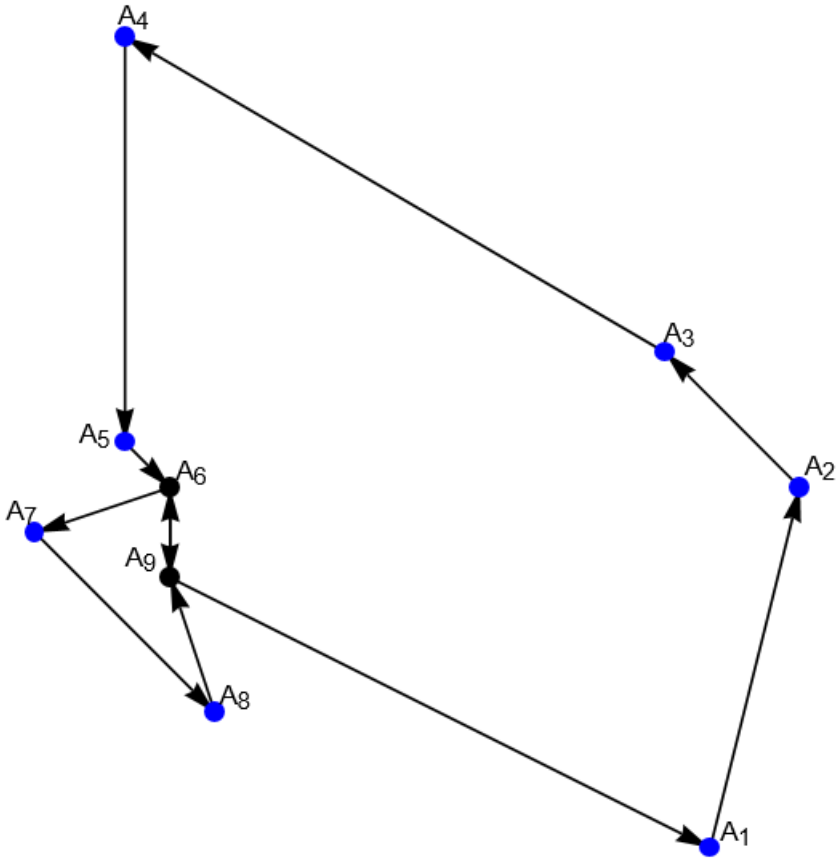


Рисунок 2.74 – Проведение диагонали в самонепересекающемся полигоне

$$\operatorname{sgn}\angle\left(\overrightarrow{A_9A_6},\overrightarrow{A_6A_7}\right)=\operatorname{sgn}\begin{vmatrix}0 & 2\\ -3 & -1\end{vmatrix}=+1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_8A_9},\overrightarrow{A_9A_6}\right)=\operatorname{sgn}\begin{vmatrix}-1 & 3\\ 0 & 2\end{vmatrix}=-1.$$

Значит, вершина A_6 становится выпуклой, а вершина A_9 остаётся вогнутой. В четырёхугольнике из вогнутой вершины диагональ проводится единственным образом и разбивает его на два треугольника $\triangle A_7A_8A_9$ и $\triangle A_9A_6A_7$ (Рисунок 2.75).

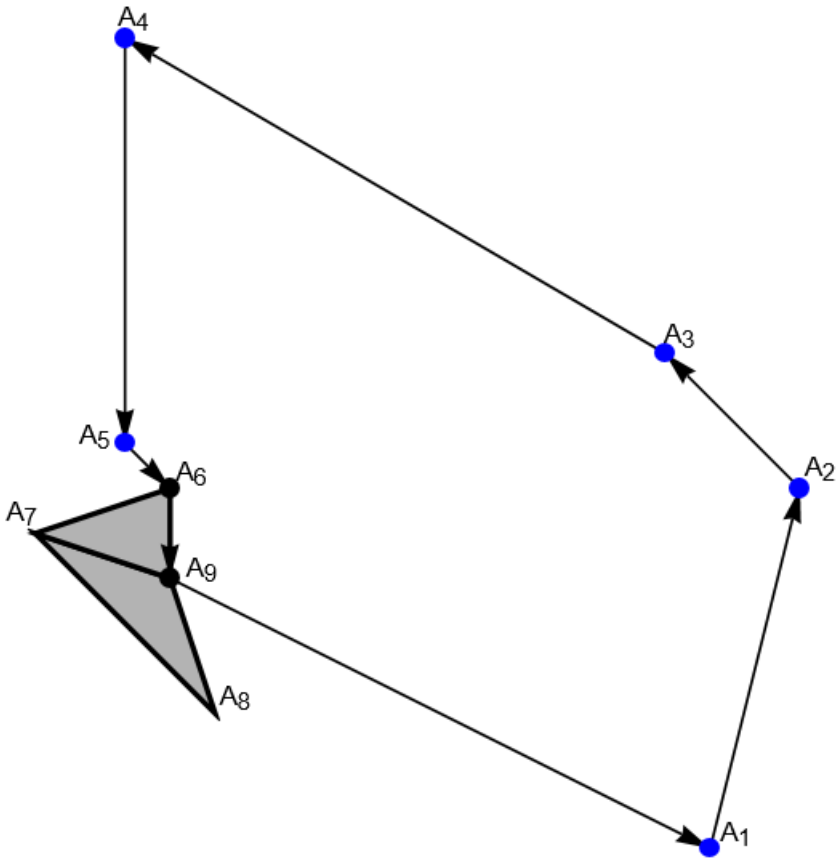


Рисунок 2.75 – Проведение диагонали в четырёхугольнике

Переходя теперь к полигону $A_9A_1A_2A_3A_4A_5A_6A_9$, имеем, что в нём вершина A_6 остаётся вогнутой, а A_9 оказывается выпуклой:

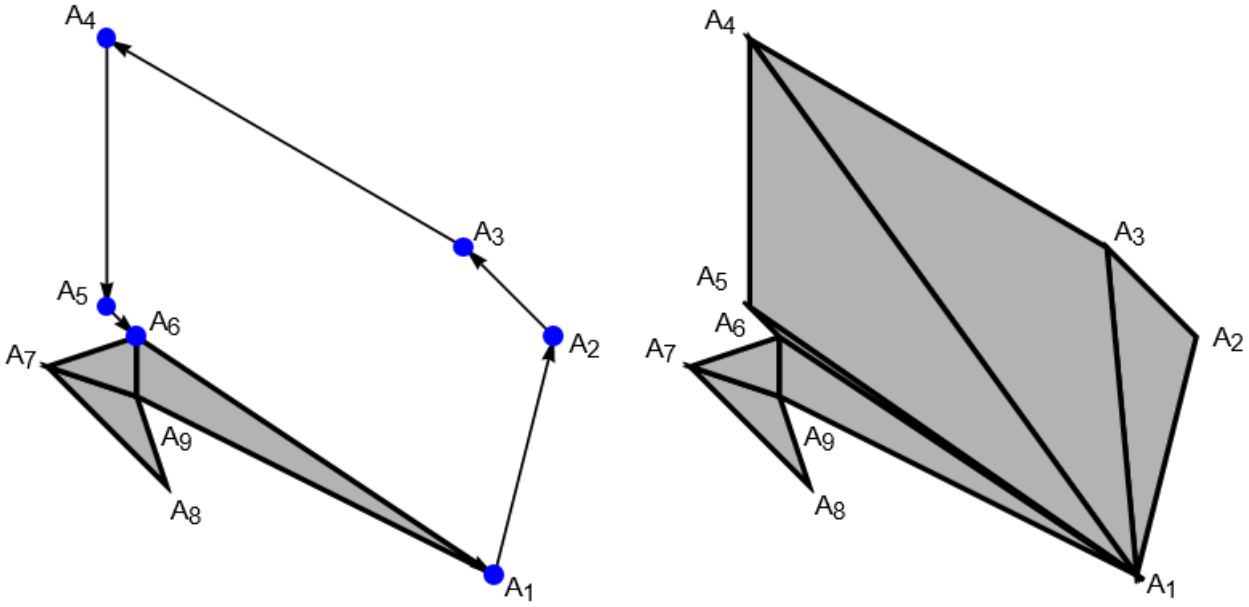
$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_6},\overrightarrow{A_6A_9}\right)=\operatorname{sgn}\begin{vmatrix}1 & -1\\ 0 & -2\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_6A_9},\overrightarrow{A_9A_1}\right)=\operatorname{sgn}\begin{vmatrix}0 & -2\\ 12 & -6\end{vmatrix}=+1.$$

Снова прибегая к габаритному условию, можно быстро проверить, что из вершины A_6 можно провести диагональ A_6A_1 (Рисунок 2.76а). При этом после проведения диагонали A_6A_1 оказывается, что вершина A_6 стала выпуклой:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_6},\overrightarrow{A_6A_1}\right)=\operatorname{sgn}\begin{vmatrix}1 & -1\\ 12 & -8\end{vmatrix}=+1,$$

поэтому оставшийся полигон $A_6A_1A_2A_3A_4A_5A_6$ является выпуклым, и все остальные диагонали можно провести из одной вершины, например, A_1 (Рисунок 2.76б). После этого получается искомая триангуляция исходного полигона.



а б

Рисунок 2.76 – Проведение заключительных диагоналей

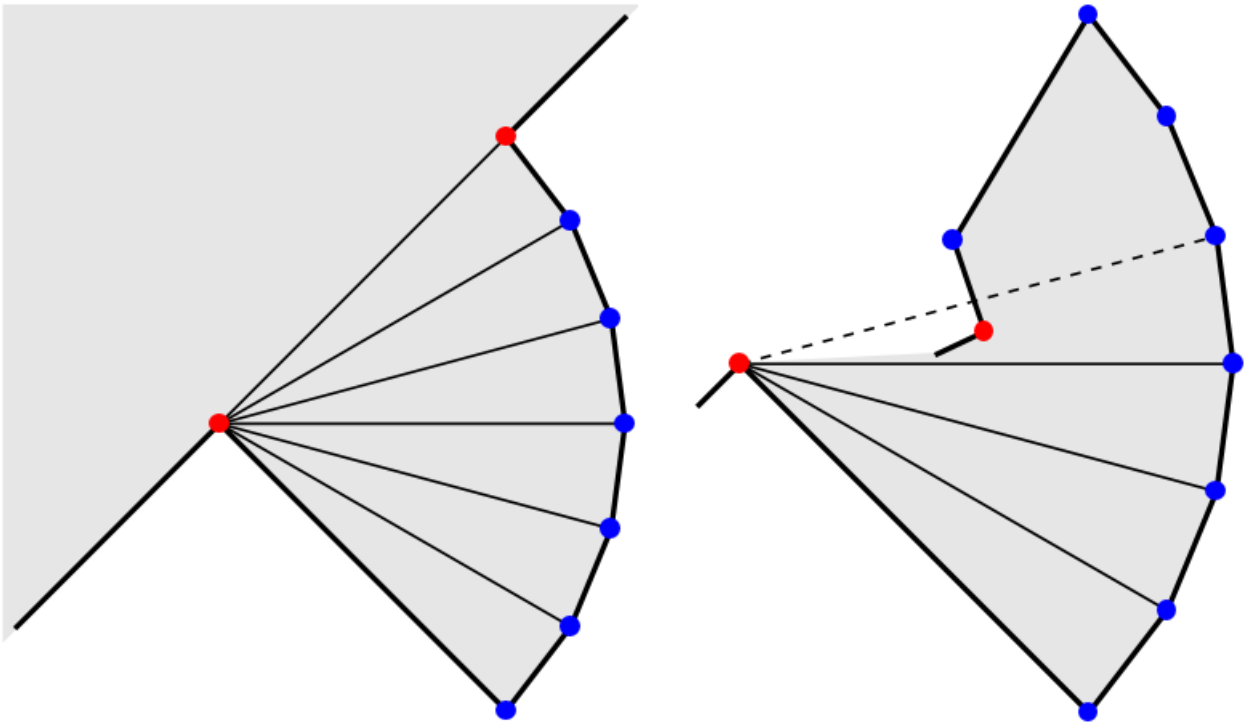
Замечание. Решение этой задачи можно значительно упростить, пользуясь следующими двумя свойствами:

- Если самонепересекающийся полигон $A_1A_2...A_nA_1$ содержит ровно одну вогнутую вершину A_1, то из неё можно провести все диагонали A_1A_i, $i=\overline{3,n-1}$.

- Если самонепересекающийся полигон содержит ровно две вогнутые вершины, и они не являются смежными, то через них можно провести диагональ.

Доказательство этих фактов опирается на свойство ломаной линии, у которой в каждой вершине происходят повороты в одну сторону, согласно которому всякий отрезок, проведённый от начала такой ломаной к произвольной её вершине, не пересекает её. Это свойство более подробно рассмотрено в первой части ЭУМК в ходе доказательства критерия выпуклости

самонепересекающегося полигона [1, с. 54–57, рис. 1.24–1.26]. Оно означает, что от вогнутой вершины A_1, у которой соседняя вершина A_2 является выпуклой, можно проводить диагонали A_1A_3, A_1A_4, …, A_1A_i до тех пор, пока либо очередная вершина не окажется вогнутой (Рисунок 2.77а), либо когда в очередном треугольнике $\triangle A_1A_iA_{i+1}$ не окажется внутренних вершин, среди которых по крайней мере одна является вогнутой (Рисунок 2.77б).



а б

Рисунок 2.77 – Проведение последовательных диагоналей из вогнутой вершины

2.5.2. Триангуляция «отрезанием ушей»

При «отрезании ушей» происходит отделение от заданного полигона некоторого треугольника искомой минимальной триангуляции при помощи проведения одной-единственной диагонали. Алгоритм «отрезания ушей» (стр. 154) гарантирует, что одна из двух частей, на которую разбивается текущий полигон на конкретной итерации, всегда является треугольником, и дальнейшей триангуляции подлежит только другая часть.

Пример 5.2. Осуществить триангуляцию полигона с вершинами в точках с координатами $A_1(13,-10)$, $A_2(10,-10)$, $A_3(11,-15)$, $A_4(9,-16)$, $A_5(7,-12)$, $A_6(4,-9)$, $A_7(2,\ -7)$, $A_8(3,-11)$, $A_9(1,-7)$, $A_{10}(-3,-4)$.

Вначале, как и в предыдущем примере, выясним направление обхода полигона, а также определим, какие вершины являются выпуклыми:

$$\overrightarrow{A_1A_2}(-3,0),\overrightarrow{A_2A_3}(1,-5),\overrightarrow{A_3A_4}(-2,-1),\overrightarrow{A_4A_5}(-2,4),\overrightarrow{A_5A_6}(-3,3),$$
$$\overrightarrow{A_6A_7}(-2,2),\overrightarrow{A_7A_8}(1,-4),\overrightarrow{A_8A_9}(-2,4),\overrightarrow{A_9A_{10}}(-4,3),\overrightarrow{A_{10}A_1}(16,-6),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2},\overrightarrow{A_2A_3}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 0\\ 1 & -5\end{vmatrix}=+1,\operatorname{sgn}\angle\left(\overrightarrow{A_2A_3},\overrightarrow{A_3A_4}\right)=\operatorname{sgn}\begin{vmatrix}1 & -5\\ -2 & -1\end{vmatrix}=-1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_3A_4},\overrightarrow{A_4A_5}\right)=\operatorname{sgn}\begin{vmatrix}-2 & -1\\ -2 & 4\end{vmatrix}=-1,\operatorname{sgn}\angle\left(\overrightarrow{A_4A_5},\overrightarrow{A_5A_6}\right)=\operatorname{sgn}\begin{vmatrix}-2 & 4\\ -3 & 3\end{vmatrix}=+1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_6},\overrightarrow{A_6A_7}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 3\\ -2 & 2\end{vmatrix}=0.$$

Получили, что очередные векторы являются коллинеарными. Несложной проверкой можно убедиться, что они сонаправлены, откуда следует принадлежность точки A_6 отрезку A_5A_7. Таким образом, рёбра A_5A_6 и A_6A_7 сливаются в одно, и точку A_6 можно больше не рассматривать в качестве вершины исходного полигона.

$$\overrightarrow{A_5A_7}(-5,5),$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_7},\overrightarrow{A_7A_8}\right)=\operatorname{sgn}\begin{vmatrix}-5 & 5\\ 1 & -4\end{vmatrix}=+1,\operatorname{sgn}\angle\left(\overrightarrow{A_7A_8},\overrightarrow{A_8A_9}\right)=\operatorname{sgn}\begin{vmatrix}1 & -4\\ -2 & 4\end{vmatrix}=-1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{A_8A_9},\overrightarrow{A_9A_{10}}\right)=\operatorname{sgn}\begin{vmatrix}-2 & 4\\ -4 & 3\end{vmatrix}=+1,\operatorname{sgn}\angle\left(\overrightarrow{A_9A_{10}},\overrightarrow{A_{10}A_1}\right)=\operatorname{sgn}\begin{vmatrix}-4 & 3\\ 16 & -6\end{vmatrix}=-1.$$

Далее вычисляются октантные углы:

$$\overrightarrow{A_1A_2}:I_1=5,\overrightarrow{A_2A_3}:I_2=7,\overrightarrow{A_3A_4}:I_3=5,\overrightarrow{A_4A_5}:I_4=3,\overrightarrow{A_5A_7}:I_5=4,$$
$$\overrightarrow{A_7A_8}:I_7=7,\overrightarrow{A_8A_9}:I_8=3,\overrightarrow{A_9A_{10}}:I_9=4,\overrightarrow{A_{10}A_1}:I_{10}=8,$$
$$\Delta_2=I_2-I_1=2,\Delta_3=I_3-I_2=-2,\Delta_4=I_4-I_3=-2,\Delta_5=I_5-I_4=1,\Delta_7=I_7-I_5=3,$$
$$\Delta_8=I_8-I_7=-4\equiv-4,\Delta_9=I_9-I_8=1,\Delta_{10}=I_{10}-I_9=4\equiv-4,\Delta_1=I_1-I_{10}=-3.$$

Здесь подлежат корректировке только значения $\Delta_8=-4$ и $\Delta_{10}=4$, по модулю равные 4. У первой разности знак «минус» остаётся, так как соответствующий угол $\angle\left(\overrightarrow{A_7A_8},\overrightarrow{A_8A_9}\right)$ отрицательный, у второй знак меняется на «минус», так как угол $\angle\left(\overrightarrow{A_9A_{10}},\overrightarrow{A_{10}A_1}\right)$ тоже отрицательный. В итоге имеем сумму $\Delta=\sum_{i\in\{1,2,3,4,5,7,8,910\}}\Delta_i'=-8$, что соответствует правостороннему обходу (Рисунок 2.78).

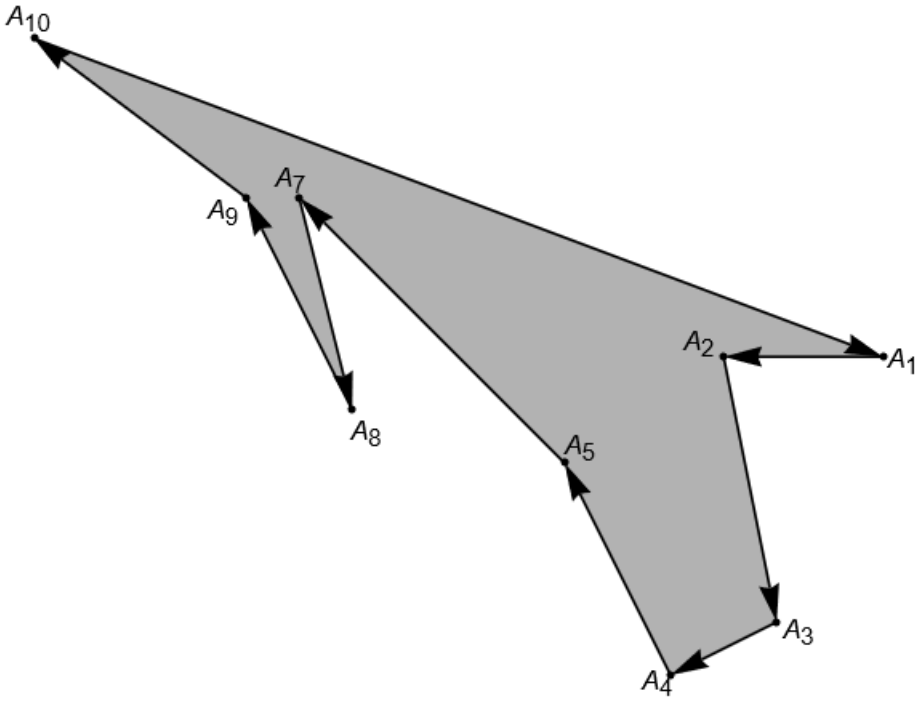


Рисунок 2.78 – Самонепересекающийся полигон после удаления фиктивной вершины

На каждой итерации будем выбирать одну из выпуклых вершин и проверять, можно ли провести внутреннюю диагональ между двумя смежными ей вершинами. У данного полигона согласно значениям скорректированных разностей Δ_i' выпуклыми являются вершины $A_1(13,-10)$, $A_3(11,-15)$, $A_4(9,-16)$, $A_8(3,-11)$, $A_{10}(-3,-4)$. Среди них выберем какую-нибудь крайнюю, например, крайнюю справа вершину A_1. Такой выбор выпуклой вершины увеличивает вероятность того, что в треугольник при этой вершине не попадут другие вершины, а следовательно, треугольник образует ухо.

Проверим, попадёт ли в треугольник $\triangle A_{10}A_1A_2$ какая-нибудь из остальных вершин полигона. Для этого снова прибегнем к габаритному условию. Так, треугольник $\triangle A_{10}A_1A_2$ ограничивается габаритным прямоугольником $\{(x,y)|\ -3\le x\le 13\,\&\,-10\le y\le -4\}$. Несложно проверить, что все остальные вершины, кроме A_7 и A_9, находятся за его пределами (Рисунок 2.79).

Для этих двух точек теперь проверяется принадлежность самому треугольнику через критерий принадлежности точки треугольнику:

$$\overrightarrow{A_1A_2}(-3,0),\overrightarrow{A_1A_{10}}(-16,6),\overrightarrow{A_1A_7}(-11,3),\overrightarrow{A_1A_9}(-12,3),$$

$$\left(\begin{array}{cc|cc} -3 & -16 & -11 & -12 \\ 0 & 6 & 3 & 3 \end{array}\right)\sim\left(\begin{array}{cc|cc} -3 & 0 & -3 & -4 \\ 0 & 2 & 1 & 1 \end{array}\right)\sim\left(\begin{array}{cc|cc} 1 & 0 & 1 & 4/3 \\ 0 & 1 & 1/2 & 1/2 \end{array}\right).$$

В обоих столбцах в правой части матрицы получились положительные значения, сумма которых, однако, превышает 1, следовательно, обе точки

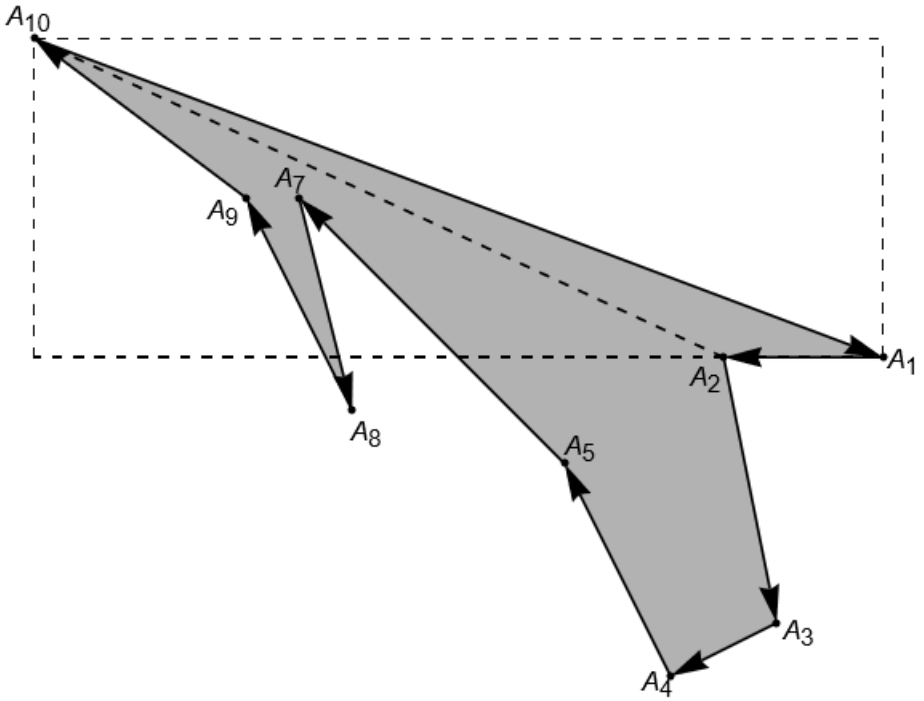


Рисунок 2.79 – Проверка габаритного условия принадлежности точки треугольнику

расположены за пределом треугольника. Отсюда следует, что $\triangle A_{10}A_1A_2$ образует ухо исходного полигона, и его можно отсечь (Рисунок 2.80).

После отрезания уха проверяется, окажутся ли выпуклыми вершины A_{10} и A_2 в новом полигоне. Вершина A_{10} была выпуклой в исходном полигоне, значит, она выпуклой и останется. Вершина A_2 исходного полигона была вогнутой, значит, необходима проверка:

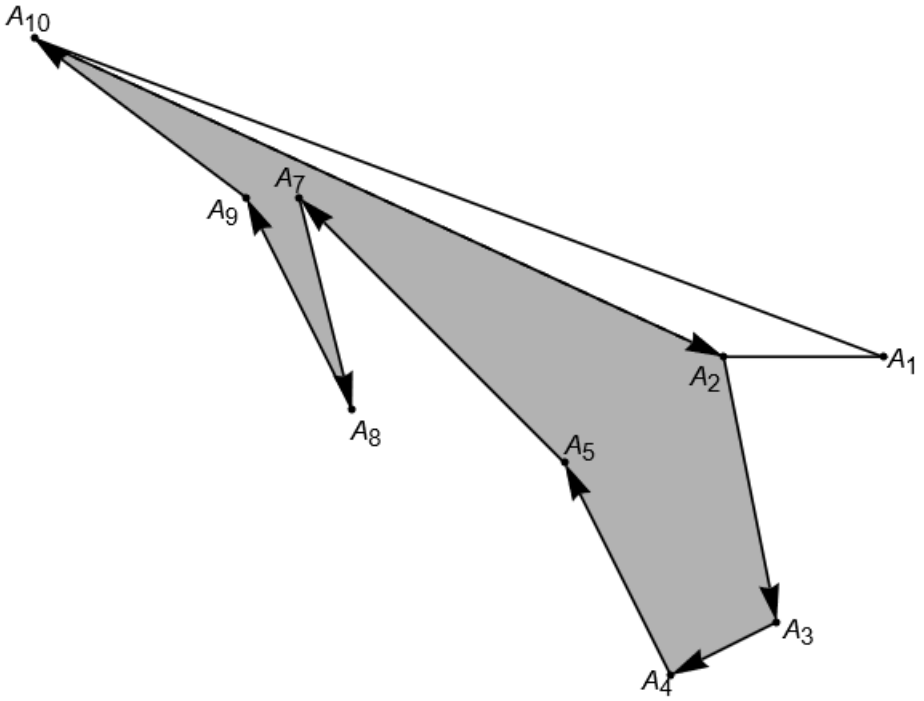
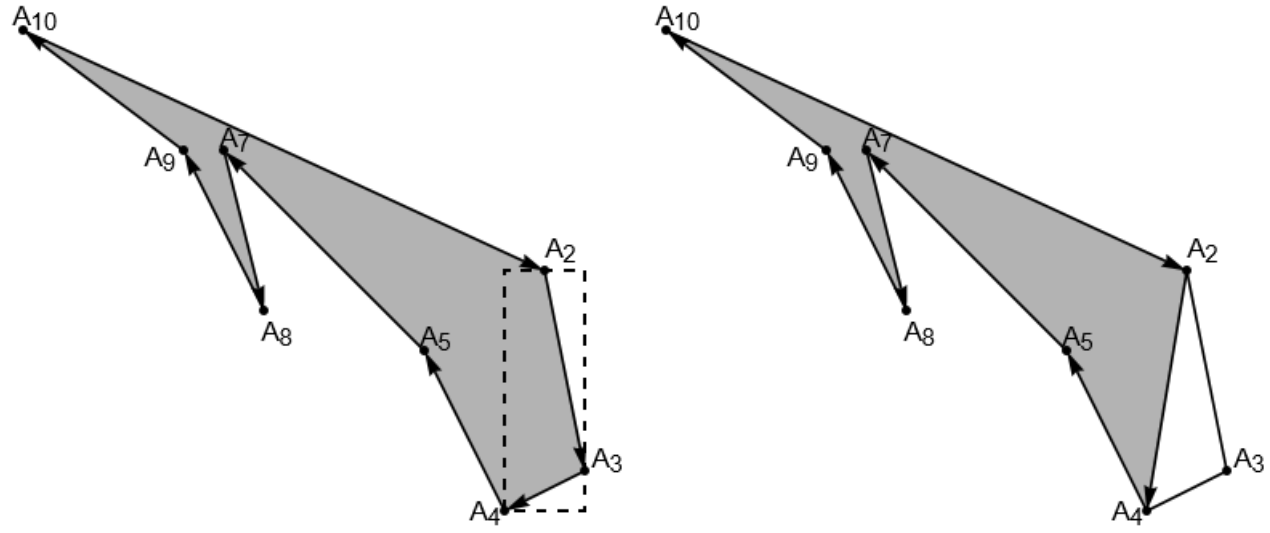


Рисунок 2.80 – Отсечение уха

$$\operatorname{sgn}\angle\left(\overrightarrow{A_{10}A_2},\overrightarrow{A_2A_3}\right)=\operatorname{sgn}\begin{vmatrix}13 & -6\\ 1 & -5\end{vmatrix}=-1.$$

При правостороннем обходе знак «минус» соответствует выпуклой вершине. Значит, теперь список выпуклых вершин выглядит следующим образом: $A_2(10,-10)$, $A_3(11,-15)$, $A_4(9,-16)$, $A_8(3,-11)$, $A_{10}(-3,-4)$.

Теперь крайней справа является вершина A_3. В прямоугольнике, ограничивающем треугольник $\triangle A_2A_3A_4$, нет никакой другой вершины нового полигона, значит, в сам треугольник они тоже не попадут (Рисунок 2.81а). Получили второе ухо, подлежащее отсечению (Рисунок 2.81б).



а б

Рисунок 2.81 – Отсечение второго уха

Вершины A_2 и A_4 как были выпуклыми в только что рассмотренном полигоне, так и останутся выпуклыми в новом. Значит, из списка выпуклых вершин просто исключается точка A_3, и остаются $A_2(10,-10)$, $A_4(9,-16)$, $A_8(3,-11)$, $A_{10}(-3,-4)$.

Для следующей крайней справа выпуклой вершины A_2 имеем треугольник $\triangle A_{10}A_2A_4$. При помощи прямоугольника уже не получается отсеять внешние точки (Рисунок 2.82). Значит, необходимо использовать критерий принадлежности точек A_5, A_7, A_8, A_9 этому треугольнику:

$$\overrightarrow{A_2A_4}(-1,-6),\overrightarrow{A_2A_{10}}(-13,6),\overrightarrow{A_2A_5}(-3,-2),\overrightarrow{A_2A_7}(-8,3),\overrightarrow{A_2A_8}(-7,-1),\overrightarrow{A_2A_9}(-9,3),$$

$$\left(\begin{array}{cc|cccc}-1 & -13 & -3 & -8 & -7 & -9\\ -6 & 6 & -2 & 3 & -1 & 3\end{array}\right)\sim\left(\begin{array}{cc|cccc}-1 & -13 & -3 & -8 & -7 & -9\\ 0 & 84 & 16 & 51 & 41 & 57\end{array}\right)\sim$$

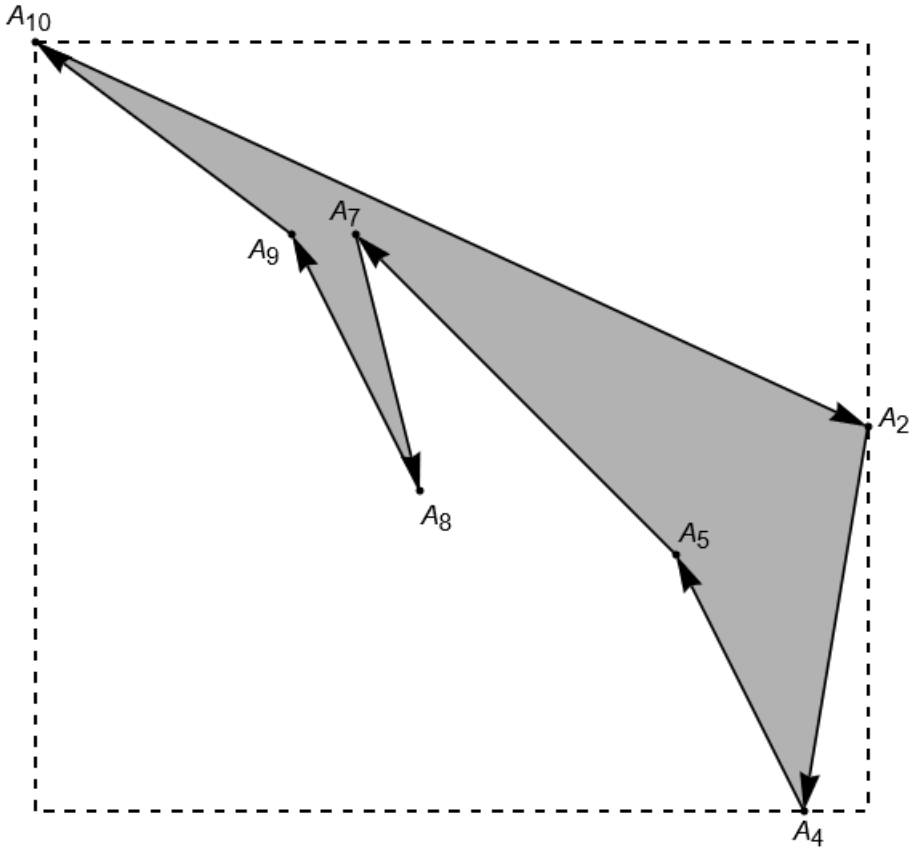


Рисунок 2.82 – Вершины попали в габаритный прямоугольник,
но отсюда ещё не следует их принадлежность треугольнику

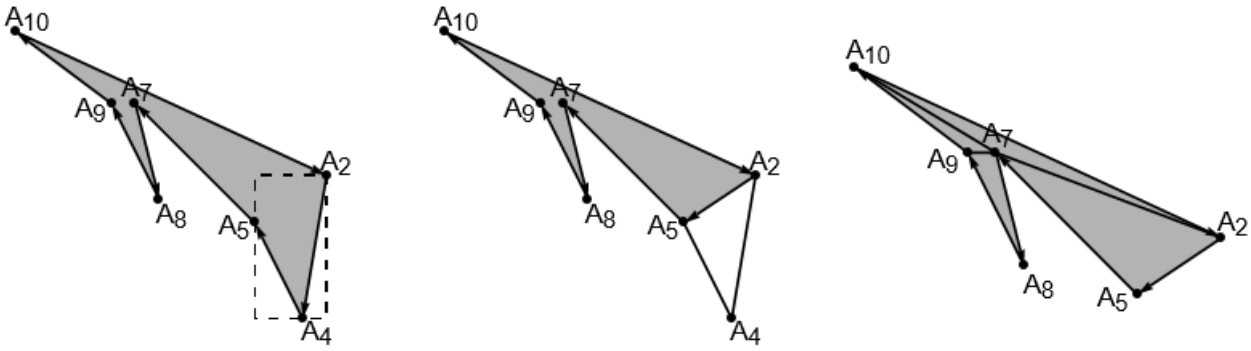
$$\sim\left(\begin{array}{cc|cccc} -1 & -13 & -3 & -8 & -7 & -9 \\ 0 & 1 & 4/21 & 17/28 & 41/84 & 19/28 \end{array}\right)\sim$$

$$\sim\left(\begin{array}{cc|cccc} -1 & 0 & -3+4\cdot 13/21 & -8+17\cdot 13/28 & -7+41\cdot 13/84 & -9+19\cdot 13/28 \\ 0 & 1 & 4/21 & 17/28 & 41/84 & 19/28 \end{array}\right)\sim$$

$$\sim\left(\begin{array}{cc|cccc} 1 & 0 & \frac{11}{21} & \frac{3}{28} & \frac{55}{84} & \frac{5}{28} \\ 0 & 1 & \frac{4}{21} & \frac{17}{28} & \frac{41}{84} & \frac{19}{28} \end{array}\right).$$

Отсюда получается, что вершины A_5, A_7 и A_9 лежат внутри $\triangle A_{10}A_2A_4$. Таким образом, этот треугольник не образует уха текущего полигона – нужно брать другую выпуклую вершину.

Рассматривая теперь выпуклую вершину A_4, нетрудно убедиться, что треугольник $\triangle A_2A_4A_5$ не содержит других вершин полигона (Рисунок 2.83а). Значит, можно отрезать ухо $\triangle A_2A_4A_5$, при этом появится новая выпуклая вершина $A_5(7,-12)$ (Рисунок 2.83б). Дальнейшее отрезание ушей может быть осуществлено, например, как показано на рисунке 2.83в. Окончательная триангуляция исходного полигона приведена на рисунке 2.84.



а б в

Рисунок 2.83 – Заключительные шаги алгоритма «отрезания ушей»

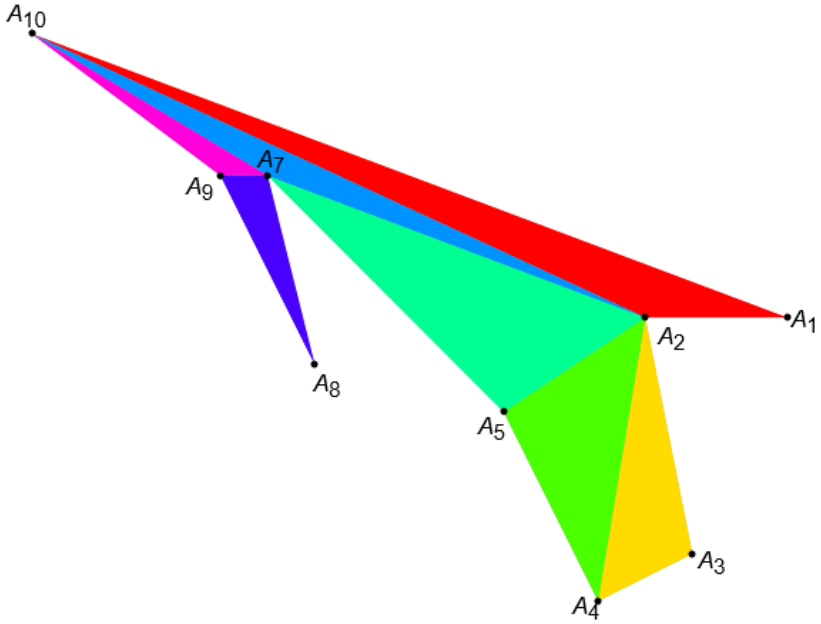


Рисунок 2.84 – Результат триангуляции при помощи «отрезания ушей»

Замечание 1. В ситуациях, когда некоторые точки попали в очередной треугольник, вместо перехода к другой выпуклой вершине и проверки того, образует ли она ухо, можно провести диагональ. Так, в данном примере на шаге, представленном на рисунке 2.82, можно проверить, какие точки попадают в треугольник $\triangle A_{10}A_2A_4$, после чего из всех внутренних точек выбрать наиболее удалённую от прямой $A_{10}A_4$ и провести от этой точки диагональ к вершине A_2. Недостатками такого подхода является, во-первых, необходимость определять принадлежность всех оставшихся вершин текущему треугольнику, во-вторых, возможность разбиения полигона на две части, каждая из которых подлежит дальнейшей триангуляции, что может усложнить реализацию алгоритма.

Замечание 2. С учётом замечания из подраздела 2.5.1 можно несколько изменить этот алгоритм:

1. Первоначально выбирается **вогнутая** вершина A_i таким образом, чтобы следующая вершина A_{i+1} оказалась выпуклой.

2. Для треугольника $\triangle A_i A_{i+1} A_{i+2}$ проверяется, образует ли он ухо. Для этого достаточно проверить, не находится ли хотя бы одна **вогнутая** вершина внутри него. Если ни одной вогнутой вершины не оказалось внутри треугольника, то не окажется внутри него и ни одной выпуклой вершины, и треугольник $\triangle A_i A_{i+1} A_{i+2}$ образует ухо.

3. После отсечения уха проверяется, осталась ли вершина A_i вогнутой. Если да, то также проверяется, является ли очередная вершина, A_{i+2}, выпуклой. Если да, то шаги 2–3 повторяются.

4. Если на шаге 2 оказалось, что $\triangle A_i A_{i+1} A_{i+2}$ не образует уха, а также при нарушении одного из условий, описанных на шаге 3, происходит переход к следующей вогнутой вершине, у которой соседняя вершина является выпуклой.

5. В ходе выполнения предыдущих шагов алгоритма вогнутых вершин рано или поздно может остаться не более одной. Это означает, что получили полигон, в котором все диагонали проводятся из единственной вогнутой вершины (или вообще из любой, если получили выпуклый полигон).

Этот алгоритм позволяет несколько ускорить триангуляцию в ситуациях, когда полигон состоит из большого числа выпуклых вершин, эффективно уменьшая число вогнутых вершин. Тем не менее, в целом временную асимптотику он не улучшает, так как всего вогнутых вершин может оказаться $n-3$, где n – количество вершин полигона (Рисунок 2.85), и на каждой из итераций происходит проверка принадлежности $O(n)$ точек текущему треугольнику.

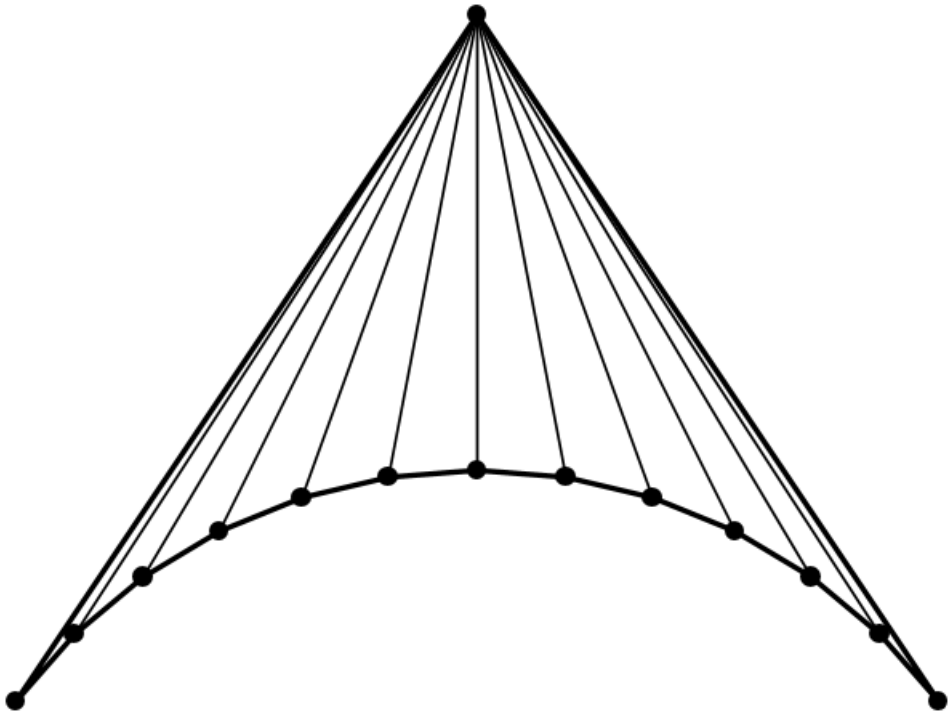


Рисунок 2.85 – Триангуляция полигона с максимально возможным числом вогнутых вершин

2.5.3. Разрезание полигона хордами

В некоторых приложениях компьютерной графики иногда бывает достаточно разбивать полигоны не на треугольные, а на произвольные выпуклые части. Минимального числа выпуклых частей можно добиться, проводя через вогнутые вершины хорды – продолжения рёбер внутрь полигона. Любая хорда делит угол, откуда она проведена, на развёрнутый и острый, тем самым уменьшая на один число всех вогнутых вершин. Может этот подход использоваться и для триангуляции (все выпуклые области легко делятся на треугольники проведением диагоналей из какой-то одной вершины), однако число треугольников не является минимальным из-за возникновения дополнительных узлов.

Пример 5.3. Разделить хордами на выпуклые части полигон из примера 5.2.

Как было вычислено ранее, вогнутыми в данном полигоне являются вершины A_2, A_5, A_7 и A_9. Для проведения первой хорды можно взять любую из них, скажем, A_2. Проводится хорда – продолжение ребра A_1A_2 (Рисунок 2.86).

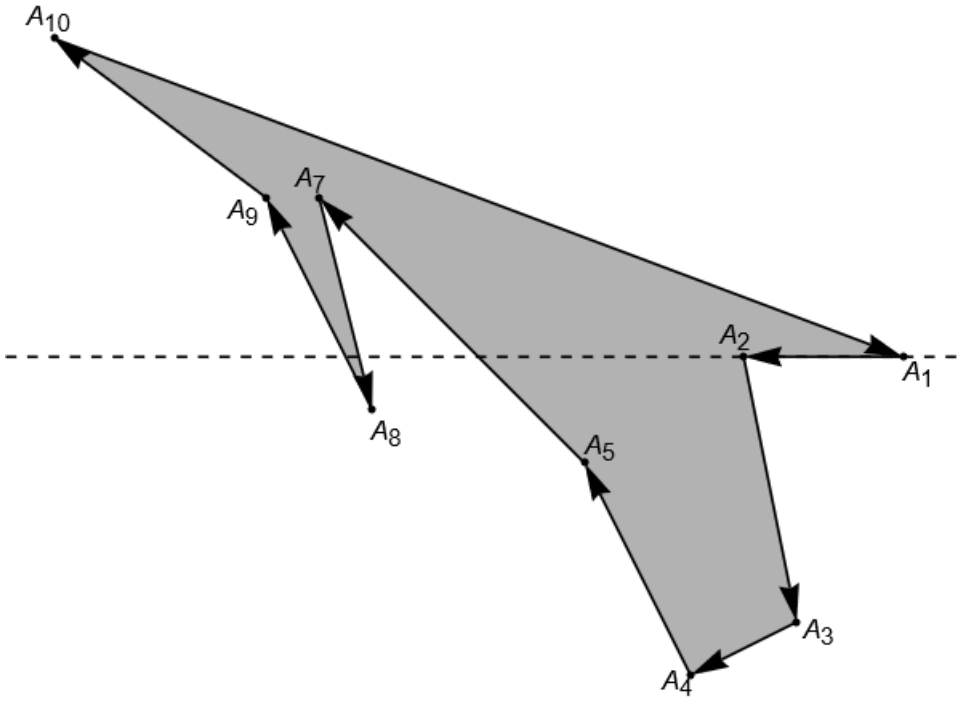


Рисунок 2.86 – Проведение хорды из вогнутой вершины полигона

Запишем параметрические уравнения луча A_1A_2:

$$A_1A_2: \begin{cases} x = 13 - 3t, \\ y = -10, \\ t \geq 0. \end{cases} \qquad (2.28)$$

При таком задании луча его началу соответствует параметр $t = 0$, а вершине A_2 – значение $t = 1$. Теперь необходимо проверить, с какими рёбрами полигона пересекается этот луч, и из них выбрать тот, точке пересечения луча с которым отвечает наименьший параметр $t > 1$. Для определения рёбер, имеющих пересечение с прямой A_1A_2, проверим расположение вершин A_3, A_4, …, A_9, A_{10}:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_3}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -2 & -5 \end{vmatrix} = +1 \Rightarrow A_3 \text{ лежит слева от луча,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_4}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -4 & -6 \end{vmatrix} = +1 \Rightarrow A_4 \text{ лежит слева,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_5}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -6 & -2 \end{vmatrix} = +1 \Rightarrow A_5 \text{ лежит слева,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_7}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -11 & 3 \end{vmatrix} = -1 \Rightarrow A_7 \text{ лежит справа,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_8}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -10 & -1 \end{vmatrix} = +1 \Rightarrow A_8 \text{ лежит слева,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_9}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -12 & 3 \end{vmatrix} = -1 \Rightarrow A_9 \text{ лежит справа,}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_{10}}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 0 \\ -16 & 6 \end{vmatrix} = -1 \Rightarrow A_{10} \text{ лежит справа.}$$

Прямая A_1A_2 пересекает те и только те отрезки, концы которых лежат по разные стороны от неё. Если бы оказались вершины, лежащие на самой прямой, то рёбра с концами в этих точках также следовало бы рассматривать. В данном примере получается, что прямая A_1A_2 пересекает рёбра A_5A_7, A_7A_8 и A_8A_9. Вычисляем пересечения:

$$\overrightarrow{A_5A_7}(-5,5) \perp (1,1),$$
$$A_5A_7 : x + y + 5 = 0.$$

В это уравнение подставим значения x и y из уравнений (2.28):

$$A_5A_7 \cap A_1A_2 : 13 - 3t_5 - 10 + 5 = 0 \Rightarrow t_5 = 8/3.$$

Получилось значение $t_5 > 1$, значит, хорда (или её продолжение) действительно пересекает ребро A_5A_7. Аналогичные действия применяются к остальным рёбрам:

$$\overrightarrow{A_7A_8}(1,-4) \perp (4,1),$$
$$A_7A_8 : 4x + y - 1 = 0,$$
$$A_7A_8 \cap A_1A_2 : 4(13 - 3t_7) - 10 - 1 = 0 \Rightarrow t_7 = \frac{11-52}{-12} = \frac{41}{12}.$$

Этот параметр также превышает единицу, однако он больше, чем t_5. Это означает, что луч A_1A_2 пересекает ребро A_7A_8 позже, чем ребро A_5A_7, а следовательно, конец хорды на ребре A_7A_8 не окажется.

$$\overrightarrow{A_8A_9}(-2,4) \perp (2,1),$$
$$A_8A_9 : 2x + y + 5 = 0,$$
$$A_8A_9 \cap A_1A_2 : 2(13 - 3t_8) - 10 + 5 = 0 \Rightarrow t_8 = \frac{5-26}{-6} = \frac{7}{2}.$$

Этот параметр также больше, чем t_5. Таким образом, конец хорды, проведённой как продолжение ребра A_1A_2, лежит на ребре A_5A_7. Его координаты получаются при подстановке параметра t_5 в систему (2.28):

$$A_5^1(13 - 3t, -10)\Big|_{t=t_5=8/3} = A_5^1(5, -10).$$

После проведения хорды $A_2A_5^1$ исходный полигон делится на две части: одна получается при обходе вершин от A_2 до A_5^1, вторая – от новой вершины A_5^1 до вершины, предшествующей A_2, т.е. A_1 (Рисунок 2.87).

При проведении хорды, порождающей новую вершину, эта вершина в обоих полигонах является выпуклой (ведь сумма углов при них даёт развёрнутый угол). Вогнутая вершина A_2, откуда провели первую хорду, в полигоне $A_2A_3A_4A_5A_5^1A_2$ также является выпуклой, так как угол при ней в сумме с углом $180°$, получившимся в другом полигоне, должен давать угол при вершине A_2 исходного полигона, который, конечно же, должен быть меньше $360°$. Таким образом, в полигоне $A_2A_3A_4A_5A_5^1A_2$ остаётся ровно одна вогнутая вершина A_5, откуда проводим хорду – продолжение ребра A_4A_5 :

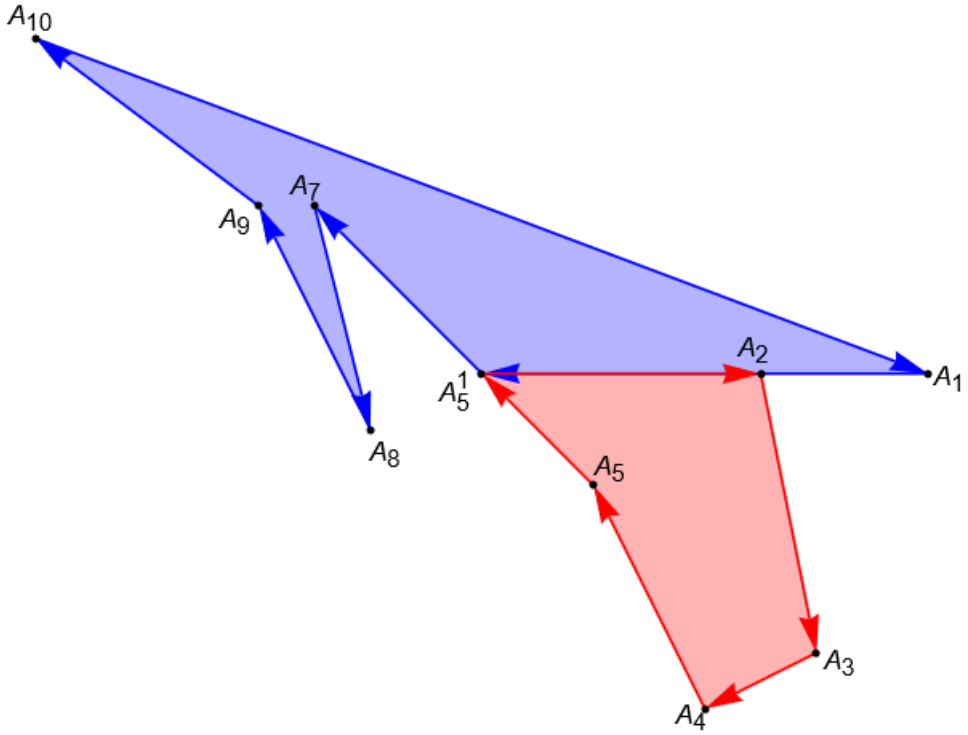


Рисунок 2.87 – Разбиение хордой самонепересекающегося полигона

$$A_4A_5: \begin{cases} x = 9 - 2t, \\ y = -16 + 4t, \\ t \geq 0, \end{cases}$$

$$\operatorname{sgn} \angle\left(\overrightarrow{A_4A_5}, \overrightarrow{A_4A_5^1}\right) = \operatorname{sgn} \begin{vmatrix} -2 & 4 \\ -4 & 6 \end{vmatrix} = +1,$$

$$\operatorname{sgn} \angle\left(\overrightarrow{A_4A_5}, \overrightarrow{A_4A_2}\right) = \operatorname{sgn} \begin{vmatrix} -2 & 4 \\ 1 & 6 \end{vmatrix} = -1,$$

$$\operatorname{sgn} \angle\left(\overrightarrow{A_4A_5}, \overrightarrow{A_4A_3}\right) = \operatorname{sgn} \begin{vmatrix} -2 & 4 \\ 2 & 1 \end{vmatrix} = -1.$$

Значит, прямая A_4A_5 пересекает только одно ребро $A_5^1A_2$:

$$\overrightarrow{A_5^1A_2}(-5,0) \perp (0,1),$$

$$A_5^1A_2: y = -10,$$

$$A_5^1A_2 \cap A_4A_5: -16 + 4t = -10 \Rightarrow t = 3/2,$$

$$A_5^2(9 - 2t, -16 + 4t)\Big|_{t=3/2} = A_5^2(6, -10).$$

Хорда $A_5A_5^2$ разделяет полигон $A_2A_3A_4A_5A_5^1A_2$ на треугольник $\triangle A_5A_5^1A_5^2$ и четырёхугольник $A_5^2A_2A_3A_4A_5^2$. Внутри обоих полигонов вогнутых вершин больше не остаётся, и дальнейшему разрезанию они не подлежат (Рисунок 2.88).

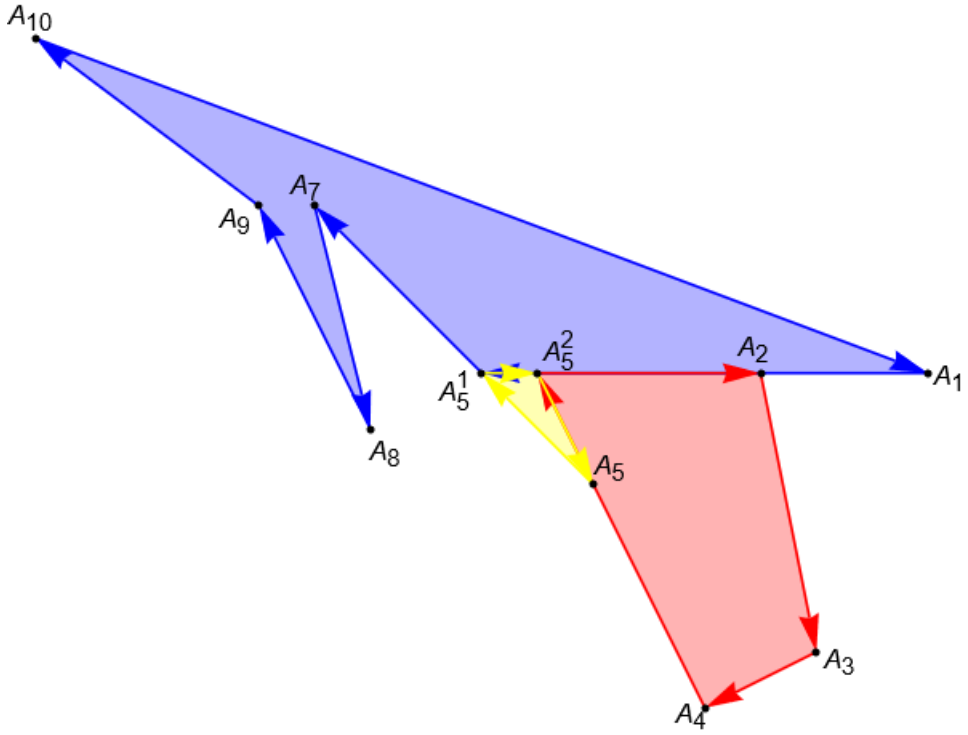


Рисунок 2.88 – Разрезание одной из получившихся частей исходного полигона

В полигоне $A_5^1A_7A_8A_9A_{10}A_1A_5^1$ остаются две вогнутые вершины. Проведём очередную хорду из одной из них, например из A_7:

$$A_5^1A_7: \begin{cases} x = 5 - 3t, \\ y = -10 + 3t, \\ t \geq 0. \end{cases}$$

Остальные вершины данного полигона относительно него лежат следующим образом:

$$\operatorname{sgn} \angle\left(\overrightarrow{A_5^1A_7}, \overrightarrow{A_5^1A_8}\right) = \operatorname{sgn} \begin{vmatrix} -3 & 3 \\ -2 & -1 \end{vmatrix} = +1,$$

$$\operatorname{sgn} \angle\left(\overrightarrow{A_5^1A_7}, \overrightarrow{A_5^1A_9}\right) = \operatorname{sgn} \begin{vmatrix} -3 & 3 \\ -4 & 3 \end{vmatrix} = +1,$$

$$\operatorname{sgn} \angle\left(\overrightarrow{A_5^1A_7}, \overrightarrow{A_5^1A_{10}}\right) = \operatorname{sgn} \begin{vmatrix} -3 & 3 \\ -8 & 6 \end{vmatrix} = +1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5^1A_7},\overrightarrow{A_5^1A_1}\right)=\operatorname{sgn}\begin{vmatrix}-3 & 3\\ 8 & 0\end{vmatrix}=-1.$$

Значит, хорда получится в результате пересечения луча $A_5^1A_7$ и ребра $A_{10}A_1$:

$$\overrightarrow{A_{10}A_1}(16,-6)\perp(3,8),$$
$$A_{10}A_1:3x+8y+41=0,$$
$$A_{10}A_1\cap A_5^1A_7:3(5-3t)+8(-10+3t)+41=0\Rightarrow t=\frac{-15+80-41}{-9+24}=\frac{24}{15}=\frac{8}{5},$$
$$A_{10}^1(5-3t,-10+3t)\Big|_{t=8/5}=A_{10}^1\left(\frac{1}{5},-\frac{26}{5}\right).$$

Хордой $A_7A_{10}^1$ полигон $A_5^1A_7A_8A_9A_{10}A_1A_5^1$ делится на полигоны $A_7A_8A_9A_{10}A_{10}^1A_7$ и $\triangle A_{10}^1A_1A_5^1$ (Рисунок 2.89).

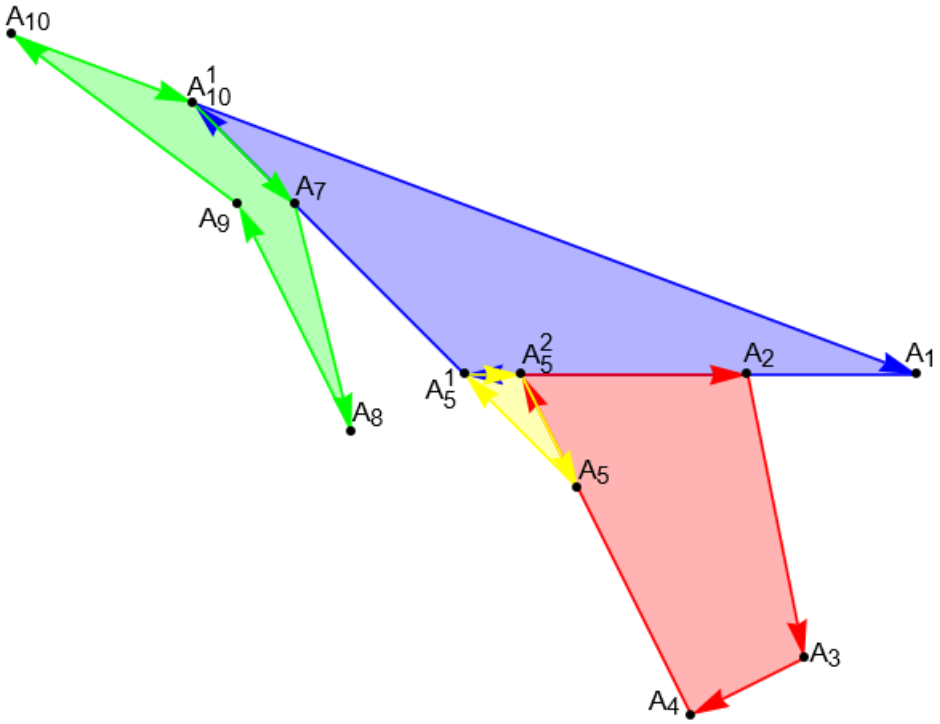


Рисунок 2.89 – Проведение хорды в другой части

Заключительная хорда проводится из вершины A_9 в полигоне $A_7A_8A_9A_{10}A_{10}^1A_7$:

$$A_8A_9:\begin{cases}x=3-2t,\\ y=-11+4t,\\ t\ge 0,\end{cases}$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_8A_9},\overrightarrow{A_8A_{10}}\right)=\operatorname{sgn}\begin{vmatrix}-2 & 4\\ -6 & 7\end{vmatrix}=+1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_8A_9},\overrightarrow{A_8A_{10}^1}\right)=\operatorname{sgn}\begin{vmatrix}-2 & 4\\ -14/5 & 29/5\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_8A_9},\overrightarrow{A_8A_7}\right)=\operatorname{sgn}\begin{vmatrix}-2 & 4\\ -1 & 4\end{vmatrix}=-1,$$

$$\overrightarrow{A_{10}A_{10}^1}\left(\frac{16}{5},-\frac{6}{5}\right)\perp(3,8),$$

$$A_{10}A_{10}^1:3x+8y+41=0,$$

$$A_{10}A_{10}^1\cap A_8A_9:3(3-2t)+8(-11+4t)+41=0\Rightarrow t=\frac{-9+88-41}{-6+32}=\frac{38}{26}=\frac{19}{13},$$

$$A_{10}'(3-2t,-11+4t)\big|_{t=19/13}=A_{10}'\left(\frac{1}{13},-\frac{67}{13}\right).$$

После проведения хорды A_9A_{10}' получается окончательное разрезание исходного полигона: $\triangle A_5A_5^1A_5^2$, $A_5^2A_2A_3A_4A_5^2$, $\triangle A_{10}^1A_1A_5^1$, $\triangle A_9A_{10}A_{10}'$, $A_{10}'A_{10}^1A_7A_8A_{10}'$ (Рисунок 2.90).

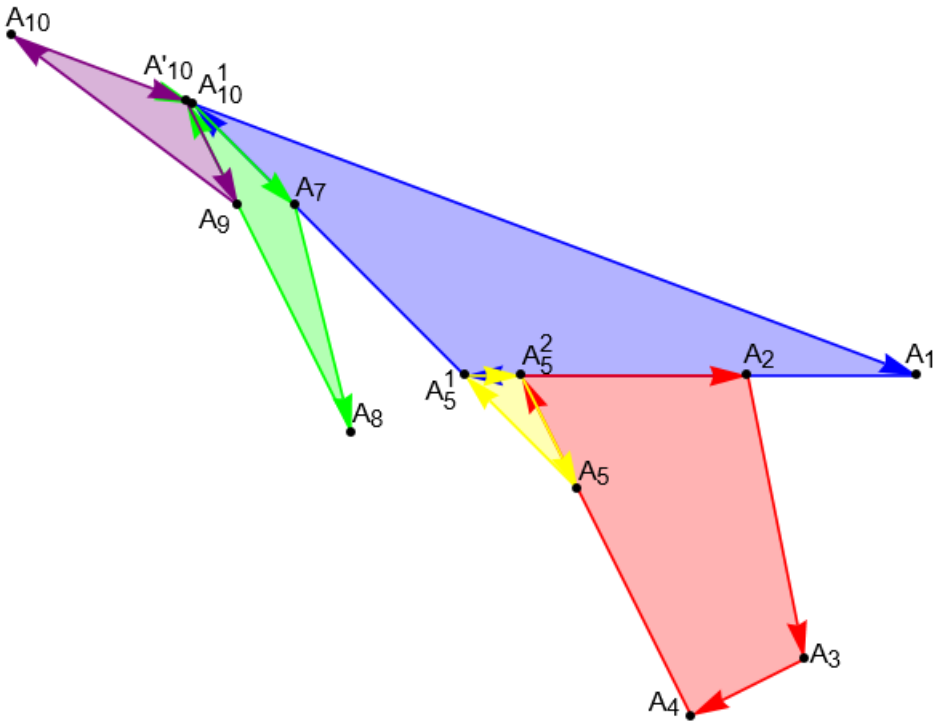


Рисунок 2.90 – Разбиение самонепересекающегося полигона на выпуклые части

Замечание. При попадании хорды на другую вершину полигона необходимо осуществлять проверку, окажется ли эта вершина в новых полигонах вогнутой, выпуклой или даже фиктивной (угол при ней окажется равным развёрнутому). Таким образом, количество хорд, достаточных для деления полигона на выпуклые части, может оказаться меньше числа вогнутых вершин.

2.5.4. Задачи

1. Осуществить триангуляцию полигонов, заданных последовательностью своих вершин $\{A_i\}_{i=1}^{n}$, при помощи диагоналей.

1.1. $A_1(3,1)$, $A_2(-2,1)$, $A_3(-2,2)$, $A_4(-3,3)$, $A_5(-2,-1)$, $A_6(-2,-3)$, $A_7(-3,-5)$, $A_8(5,-4)$, $A_9(5,-1)$ (перейти к ответу),

1.2. $A_1(-5,3)$, $A_2(-2,5)$, $A_3(2,-1)$, $A_4(1,-4)$, $A_5(-1,-2)$, $A_6(-2,-2)$, $A_7(-1,-5)$, $A_8(-2,-5)$, $A_9(-4,0)$ (перейти к ответу),

1.3. $A_1(-1,-1)$, $A_2(1,-5)$, $A_3(-5,-3)$, $A_4(-3,0)$, $A_5(-1,-3)$, $A_6(-1,-2)$, $A_7(-3,1)$, $A_8(-3,4)$, $A_9(5,-1)$ (перейти к ответу),

1.4. $A_1(-3,-1)$, $A_2(-3,0)$, $A_3(0,4)$, $A_4(4,4)$, $A_5(5,-5)$, $A_6(-2,-5)$, $A_7(1,1)$, $A_8(1,2)$, $A_9(-2,-4)$ (перейти к ответу),

1.5. $A_1(4,7)$, $A_2(10,9)$, $A_3(2,-1)$, $A_4(0,-1)$, $A_5(-6,-6)$, $A_6(-7,3)$, $A_7(-9,0)$, $A_8(-8,10)$, $A_9(-6,10)$ (перейти к ответу),

1.6. $A_1(-6,7)$, $A_2(-10,-10)$, $A_3(10,-10)$, $A_4(1,-3)$, $A_5(8,6)$, $A_6(4,2)$, $A_7(-1,5)$, $A_8(-4,9)$, $A_9(-3,-2)$ (перейти к ответу),

1.7. $A_1(9,6)$, $A_2(8,-4)$, $A_3(8,-9)$, $A_4(-5,4)$, $A_5(-3,-9)$, $A_6(-9,-9)$, $A_7(-4,10)$, $A_8(2,10)$, $A_9(-1,7)$ (перейти к ответу),

1.8. $A_1(5,-6)$, $A_2(-4,-7)$, $A_3(-3,1)$, $A_4(-10,2)$, $A_5(-9,9)$, $A_6(-7,5)$, $A_7(-5,8)$, $A_8(1,-2)$, $A_9(2,7)$, $A_{10}(8,-8)$ (перейти к ответу).

2. Разрезать хордами полигоны, заданные последовательностью своих вершин $\{A_i\}_{i=1}^{n}$.

2.1. $A_1(0,0)$, $A_2(3,-1)$, $A_3(7,1)$, $A_4(10,-2)$, $A_5(10,-4)$, $A_6(12,-7)$ (перейти к ответу),

2.2. $A_1(0,0)$, $A_2(0,3)$, $A_3(-3,5)$, $A_4(-5,4)$, $A_5(-2,2)$, $A_6(-3,4)$ (перейти к ответу),

2.3. $A_1(0,0)$, $A_2(-3,2)$, $A_3(-1,4)$, $A_4(0,2)$, $A_5(3,3)$, $A_6(1,2)$ (перейти к ответу),

2.4. $A_1(0,0)$, $A_2(2,1)$, $A_3(1,1)$, $A_4(-2,2)$, $A_5(-1,1)$, $A_6(-1,-1)$ (перейти к ответу),

2.5. $A_1(0,2)$, $A_2(1,4)$, $A_3(2,3)$, $A_4(3,8)$, $A_5(5,6)$, $A_6(4,11)$ (перейти к ответу),

2.6. $A_1(0,0)$, $A_2(0,1)$, $A_3(3,3)$, $A_4(-2,4)$, $A_5(1,2)$, $A_6(-1,1)$ (перейти к ответу).

2.6. Триангуляция Делоне

Для построения триангуляции Делоне в основном используются итеративные алгоритмы и алгоритмы слияния, которые опираются на построение триангуляции для некоторых подмножеств и последующем их редактировании. В итоге получается единственная возможная триангуляция (не считая ситуаций, когда среди исходных точек имеются по крайней мере четыре, лежащие на одной окружности).

2.6.1. Итеративный алгоритм

Пример 6.1. Построить итеративным алгоритмом триангуляцию Делоне множества точек $A_i(x_i, y_i),\ i = \overline{1,7}$:

$$A_1(10,2), A_2(9,4), A_3(-3,6), A_4(-3,8), A_5(0,-8), A_6(0,-7), A_7(1,-2).$$

Воспользуемся алгоритмом, приведённым на стр. 169. Каждая итерация состоит из двух шагов: локализация новой точки, т.е. определение, которому треугольнику текущей триангуляции она принадлежит (либо же она является внешней), и редактирование текущей триангуляции.

Для данного примера первая итерация даёт треугольник $\triangle A_1A_2A_3$, причём левосторонний обход даёт последовательность $A_1A_2A_3A_1$, ибо

$$\operatorname{sgn} \angle(\overrightarrow{A_1A_2}, \overrightarrow{A_1A_3}) = \operatorname{sgn} \begin{vmatrix} -1 & 2 \\ -13 & 4 \end{vmatrix} > 0.$$

Значит, после первой итерации имеет место следующий промежуточный результат (Рисунок 2.91):

$$G : \triangle A_1A_2A_3,$$
$$H : A_1A_2A_3A_1.$$

На следующей итерации добавляется точка A_4. Определим её принадлежность треугольнику $\triangle A_1A_2A_3$:

$$\overrightarrow{A_1A_2}(-1,2), \overrightarrow{A_1A_3}(-13,4), \overrightarrow{A_1A_4}(-13,6),$$

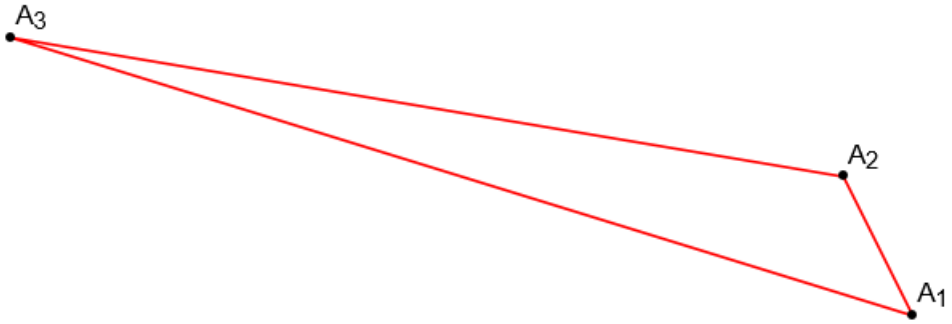


Рисунок 2.91 – Триангуляция Делоне трёх точек – это сам треугольник с вершинами в этих точках

$$\alpha\overrightarrow{A_1A_2}+\beta\overrightarrow{A_1A_3}=\overrightarrow{A_1A_4}:$$

$$\left(\begin{array}{cc|c}-1 & -13 & -13\\ 2 & 4 & 6\end{array}\right)\sim\left(\begin{array}{cc|c}-1 & -13 & -13\\ 0 & -22 & -20\end{array}\right)\sim\left(\begin{array}{cc|c}-11 & -143 & -143\\ 0 & 11 & 10\end{array}\right)\sim$$

$$\sim\left(\begin{array}{cc|c}-11 & 0 & -13\\ 0 & 11 & 10\end{array}\right)\sim\left(\begin{array}{cc|c}1 & 0 & 13/11\\ 0 & 1 & 10/11\end{array}\right).$$

Сумма полученных значений превышает единицу, следовательно, $A_4\notin\triangle A_1A_2A_3$. То есть новая точка A_4 находится за пределами текущей триангуляции, и нужно редактировать выпуклую оболочку. Фактически это означает нахождение рёбер выпуклой оболочки, видимых из новой точки, с последующих их удалением из неё, а также добавлением двух новых рёбер. Выполняется это по алгоритму нахождения видимых рёбер при ближнем наблюдателе (стр. 51).

Проведём векторы от точки A_4 ко всем вершинам из H:

$$\overrightarrow{A_4A_1}(13,-6),\ \overrightarrow{A_4A_2}(12,-4),\ \overrightarrow{A_4A_3}(0,-2).$$

Среди этих векторов необходимо выбрать два вектора $\vec{a}_{\text{ccw}}$ и $\vec{a}_{\text{cw}}$ – крайние векторы при движении против направления и по направлению часовой стрелки соответственно. Если несколько векторов подходят в качестве одного из этих векторов, то необходимо выбирать вектор наименьшей длины. Положение векторов относительно друг друга, как обычно, определяется посредством вычисления знака угла между ними:

$$\operatorname{sgn}\angle\left(\overrightarrow{A_4A_1},\overrightarrow{A_4A_2}\right)=\operatorname{sgn}\begin{vmatrix}13 & -6\\ 12 & -4\end{vmatrix}=+1\Rightarrow\vec{a}_{\text{ccw}}:=\overrightarrow{A_4A_2},\vec{a}_{\text{cw}}:=\overrightarrow{A_4A_1},$$

$$\operatorname{sgn}\angle\left(\vec{a}_{\text{ccw}},\overrightarrow{A_4A_3}\right)=\operatorname{sgn}\angle\left(\overrightarrow{A_4A_2},\overrightarrow{A_4A_3}\right)=\operatorname{sgn}\begin{vmatrix}12 & -4\\ 0 & -2\end{vmatrix}=-1\Rightarrow\vec{a}_{\text{ccw}}\text{ остаётся прежним,}$$

$$\operatorname{sgn}\angle\left(\vec{a}_{\text{cw}}, \overrightarrow{A_4A_3}\right) = \operatorname{sgn}\angle\left(\overrightarrow{A_4A_1}, \overrightarrow{A_4A_3}\right) = \operatorname{sgn}\begin{vmatrix} 13 & -6 \\ 0 & -2 \end{vmatrix} = -1 \Rightarrow \vec{a}_{\text{cw}} := \overrightarrow{A_4A_3}.$$

Значит, крайними вершинами выпуклой оболочки $H = A_1A_2A_3A_1$, видимыми из точки A_4, являются $A_4 + \vec{a}_{\text{ccw}} = A_2$ (слева) и $A_4 + \vec{a}_{\text{cw}} = A_3$ (справа). В список видимых рёбер записываются все рёбра из выпуклой оболочки при проходе от самой левой до самой правой точки: $V := \{A_2A_3\}$ (на рисунке 2.92 единственное ребро этого списка выделено синим), а новая выпуклая оболочка получается, напротив, при обходе от самой правой к самой левой вершине (при этом добавляется также новая вершина, от которой ко всем видимым точкам проводятся новые рёбра, отмеченные на рисунке 2.92 зелёным): $H := A_3A_1A_2A_4A_3$.

Граф G, состоящий из одной вершины, очевидным образом дополняется второй вершиной, отвечающей новому треугольнику (на рисунке 2.93 зелёным цветом обозначены новая вершина и ребро).

Теперь можно приступить ко второй фазе текущей итерации. Она заключается в рассмотрении каждого ребра из V. Текущее ребро может являться граничным для одного либо двух треугольников триангуляции G. Оно может являться граничным ровно для одного треугольника, только если это ребро выпуклой оболочки H. Если же оно не оказалось среди рёбер выпуклой оболочки, значит, ему соответствует некоторое ребро графа G, по которому определяются инцидентные ему треугольники. Для этих треугольников проверяется условие Делоне (1.97), и в случае его нарушения происходит флип

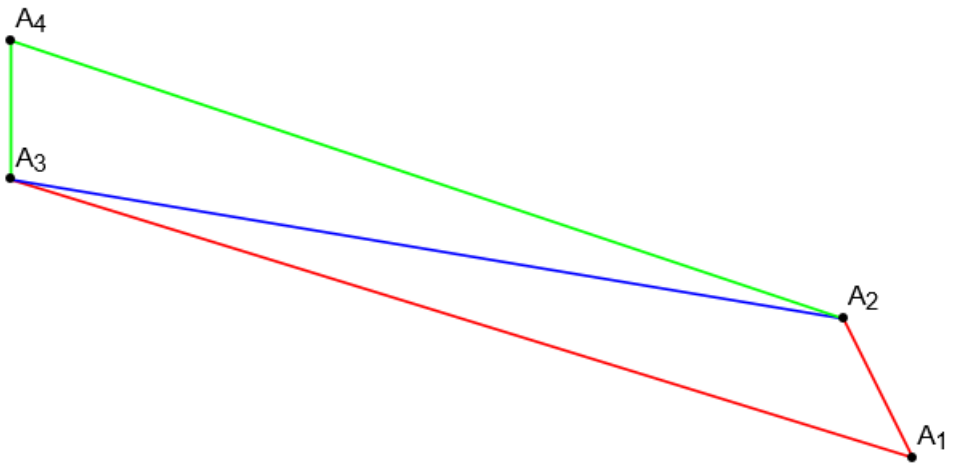


Рисунок 2.92 – Добавление новой точки в триангуляцию Делоне

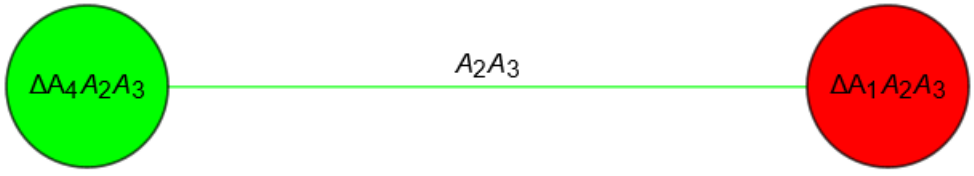


Рисунок 2.93 – Добавление нового треугольника (зелёная вершина) в граф G

текущего ребра, что приводит к замене этих треугольников на два других, а также к добавлению двух других рёбер в список V вместо текущего.

Итак, в данном примере видимое ребро A_2A_3 не является ребром выпуклой оболочки, а в графе G инцидентными для этого ребра являются треугольники $\triangle A_4A_2A_3$ и $\triangle A_1A_2A_3$. Прежде чем подставлять их вершины в формулу (1.97), проверим достаточное условие выполнения этого неравенства – остроту двух углов, опирающихся на общее ребро:

$$\overrightarrow{A_4A_2}\cdot\overrightarrow{A_4A_3}=12\cdot 0+(-4)\cdot(-2)=8>0,$$
$$\overrightarrow{A_1A_2}\cdot\overrightarrow{A_1A_3}=(-1)\cdot(-13)+2\cdot 4=21>0.$$

Скалярные произведения векторов, образующих два противолежащих угла, положительны, значит, они являются острыми, и их сумма не может превышать π, что означает выполнение условия Делоне. Ребро A_2A_3 после рассмотрения удаляется из списка: $V:=\{\}$. Итак, видимых рёбер больше не осталось, следовательно, триангуляцию, чертёж которой представлен на рисунке 2.92, а её граф G – на рисунке 2.93, не нужно редактировать – это триангуляция Делоне для четырёх точек.

На следующей итерации перейдём к точке A_5. Проверка принадлежности этой точки треугольникам из графа G даёт следующие результаты:

$$A_5\in\triangle A_4A_2A_3-?$$
$$\overrightarrow{A_2A_3}(-12,2),\overrightarrow{A_2A_4}(-12,4),\overrightarrow{A_2A_5}(-9,-12),$$
$$\alpha\overrightarrow{A_2A_3}+\beta\overrightarrow{A_2A_4}=\overrightarrow{A_2A_5}:$$
$$\left(\begin{array}{cc|c}-12&-12&-9\\2&4&-12\end{array}\right)\sim\left(\begin{array}{cc|c}-4&-4&-3\\1&2&-6\end{array}\right)\sim\left(\begin{array}{cc|c}-2&0&-15\\1&2&-6\end{array}\right)\sim$$
$$\sim\left(\begin{array}{cc|c}-2&0&-15\\2&4&-12\end{array}\right)\sim\left(\begin{array}{cc|c}-2&0&-15\\0&4&-27\end{array}\right)\sim\left(\begin{array}{cc|c}1&0&15/2\\0&1&-27/4\end{array}\right),$$
$$\beta<0\Rightarrow A_5\notin\triangle A_4A_2A_3,$$
$$A_5\in\triangle A_1A_2A_3-?$$
$$\alpha\overrightarrow{A_2A_3}+\beta\overrightarrow{A_2A_1}=\overrightarrow{A_2A_5}:$$
$$\overrightarrow{A_2A_3}(-12,2),\overrightarrow{A_2A_1}(1,-2),\overrightarrow{A_2A_5}(-9,-12),$$
$$\left(\begin{array}{cc|c}-12&1&-9\\2&-2&-12\end{array}\right)\sim\left(\begin{array}{cc|c}-12&1&-9\\1&-1&-6\end{array}\right)\sim\left(\begin{array}{cc|c}-11&0&-15\\1&-1&-6\end{array}\right)\sim$$
$$\sim\left(\begin{array}{cc|c}-11&0&-15\\11&-11&-66\end{array}\right)\sim\left(\begin{array}{cc|c}-11&0&-15\\0&-11&-81\end{array}\right)\sim\left(\begin{array}{cc|c}1&0&15/11\\0&1&81/11\end{array}\right),$$
$$\alpha+\beta>1\Rightarrow A_5\notin\triangle A_1A_2A_3.$$

Таким образом, точка A_5 является внешней по отношению к полигону H, следовательно, выпуклая оболочка подлежит редактированию:

$$\overrightarrow{A_5A_1}(10,10), \overrightarrow{A_5A_2}(9,12), \overrightarrow{A_5A_3}(-3,14), \overrightarrow{A_5A_4}(-3,16),$$

$$\operatorname{sgn}\angle\left(\overrightarrow{A_5A_1}, \overrightarrow{A_5A_2}\right) = \operatorname{sgn}\begin{vmatrix} 10 & 10 \\ 9 & 12 \end{vmatrix} = +1 \Rightarrow \vec{a}_{\text{ccw}} := \overrightarrow{A_5A_2}, \vec{a}_{\text{cw}} := \overrightarrow{A_5A_1},$$

$$\operatorname{sgn}\angle\left(\vec{a}_{\text{ccw}}, \overrightarrow{A_5A_3}\right) = \operatorname{sgn}\angle\left(\overrightarrow{A_5A_2}, \overrightarrow{A_5A_3}\right) = \operatorname{sgn}\begin{vmatrix} 9 & 12 \\ -3 & 14 \end{vmatrix} = +1 \Rightarrow \vec{a}_{\text{ccw}} = \overrightarrow{A_5A_3},$$

$$\operatorname{sgn}\angle\left(\vec{a}_{\text{ccw}}, \overrightarrow{A_5A_4}\right) = \operatorname{sgn}\angle\left(\overrightarrow{A_5A_3}, \overrightarrow{A_5A_4}\right) = \operatorname{sgn}\begin{vmatrix} -3 & 14 \\ -3 & 16 \end{vmatrix} = -1 \Rightarrow \vec{a}_{\text{ccw}} \text{ остаётся прежним,}$$

$$\operatorname{sgn}\angle\left(\vec{a}_{\text{cw}}, \overrightarrow{A_5A_4}\right) = \operatorname{sgn}\angle\left(\overrightarrow{A_5A_1}, \overrightarrow{A_5A_4}\right) = \operatorname{sgn}\begin{vmatrix} 10 & 10 \\ -3 & 16 \end{vmatrix} = +1 \Rightarrow \vec{a}_{\text{cw}} \text{ остаётся прежним.}$$

Значит, $V := \{A_3A_1\}$, $H := A_1A_2A_4A_3A_5A_1$ (Рисунок 2.94), а граф G преобразуется к виду, представленному на рисунке 2.95.

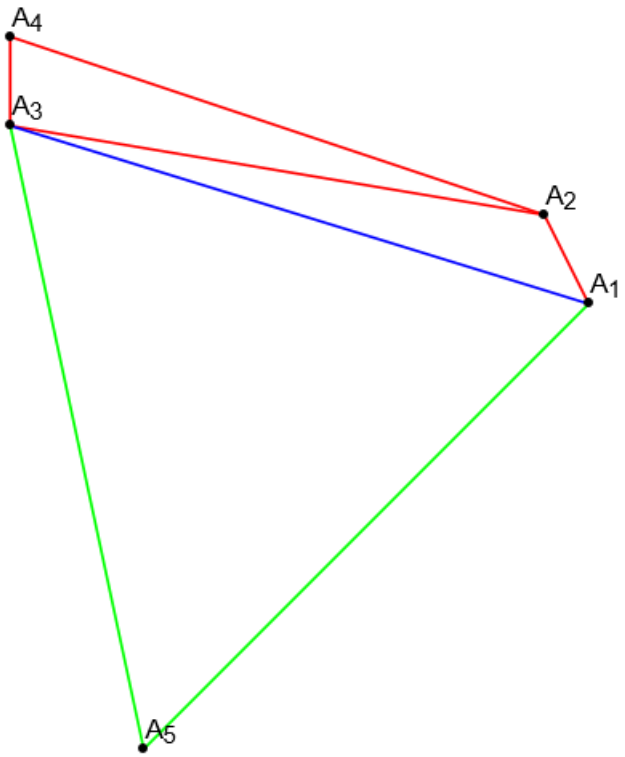


Рисунок 2.94 – Триангуляция после добавления новой вершины

Рисунок 2.95 – Граф триангуляции после добавления новой вершины

Редактирование полученной триангуляции начинается с видимого ребра A_3A_1. Оно не является ребром выпуклой оболочки, следовательно, найдутся два треугольника, граничащих друг с другом по этому ребру. Это треугольники $\triangle A_5A_3A_1$ и $\triangle A_1A_2A_3$. В них вначале проверяется острота двух противолежащих углов:

$$\overrightarrow{A_5A_3}\cdot\overrightarrow{A_5A_1}=-3\cdot10+14\cdot10=110>0,$$
$$\overrightarrow{A_2A_3}\cdot\overrightarrow{A_2A_1}=-12\cdot1+2\cdot(-2)=-16<0,$$

т.е. один угол острый, другой – тупой. Придётся проверять условие (1.97):

$$\left|\overrightarrow{A_5B_3}\times\overrightarrow{A_5B_1}\right|\left(\overrightarrow{A_2B_3}\cdot\overrightarrow{A_2B_1}\right)+\left(\overrightarrow{A_5B_3}\cdot\overrightarrow{A_5B_1}\right)\left|\overrightarrow{A_2B_3}\times\overrightarrow{A_2B_1}\right|=$$
$$=170\cdot(-16)+110\cdot22=-1700-1020+2200+220<0.$$

Неравенство (1.97) нарушается, что означает необходимость флипа видимого ребра A_3A_1. При этом граф G редактируется следующим образом:

- Вершины, отвечающие двум старым треугольникам, объединяются в одну, которая соответствует объединённому четырёхугольнику. Новая вершина смежна с теми и только теми вершинами, с которыми были смежны вершины, отвечающие старым треугольникам (Рисунок 2.96). При этом все остальные вершины и рёбра графа (в т.ч. их метки) остаются неизменными.

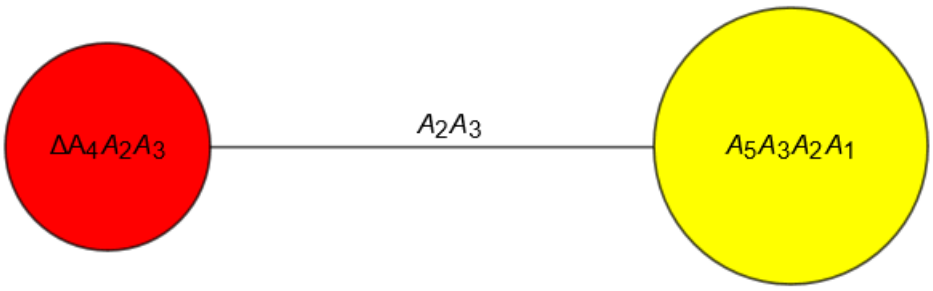


Рисунок 2.96 – Граф G после удаления видимого ребра

- Вместо объединённой вершины вставляются две вершины, отвечающие двум новым треугольникам. Две новые вершины смежны между собой, кроме того, каждая из них инцидентна тем и только тем из старых рёбер, которые соответствуют сторонам нужного треугольника (Рисунок 2.97).

Рисунок 2.97 – Граф G после проведения нового ребра

Также изменяется список видимых рёбер: $V := \{A_3A_2, A_2A_1\}$. После флипа ребра A_3A_1 на A_5A_2 триангуляция примет вид, изображённый на рисунке 2.98.

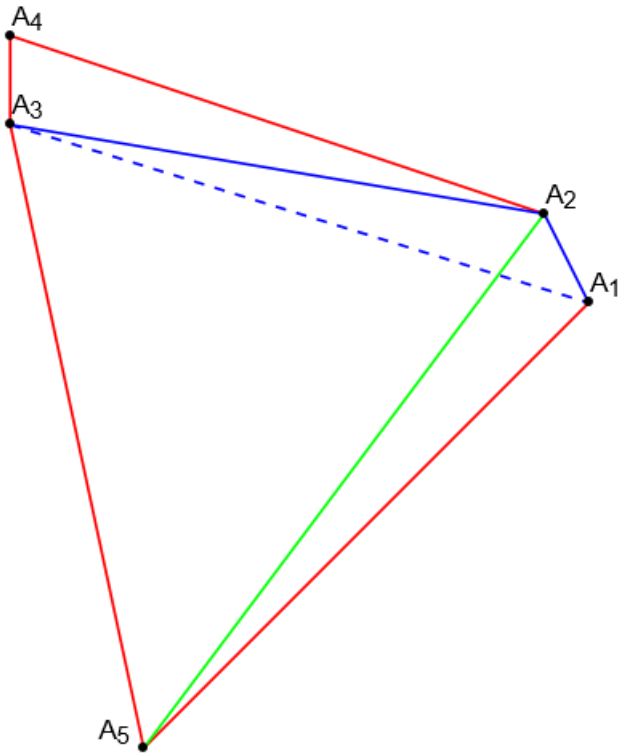


Рисунок 2.98 – Флип ребра

Далее рассмотрим ребро A_3A_2. В графе G вершины, инцидентные соответствующему ребру, представляют треугольники $\triangle A_3A_5A_2$ и $\triangle A_4A_2A_3$. Проверим для них условие Делоне:

$$\overrightarrow{A_5A_2} \cdot \overrightarrow{A_5A_3} = 9 \cdot (-3) + 12 \cdot 14 > 0,$$
$$\overrightarrow{A_4A_2} \cdot \overrightarrow{A_4A_3} = 8 > 0.$$

Это означает выполнение условия Делоне, и треугольники $\triangle A_3A_5A_2$ и $\triangle A_4A_2A_3$ остаются неизменными. Ребро A_3A_2 удаляется из списка V.

Наконец, ребро A_2A_1 оказывается ребром выпуклой оболочки и не влечёт изменений триангуляции. Таким образом, после третьей итерации имеем выпуклую оболочку $H = A_1A_2A_4A_3A_5A_1$ и триангуляцию с рисунков 2.97 и 2.98.

На следующей итерации добавляется точка A_6. Несложно видеть, что точка $A_6(0,-7)$ не может принадлежать треугольнику $\triangle A_4A_2A_3$, ограниченного габаритным прямоугольником $\{(x,y): -3 \le x \le 9, 4 \le y \le 8\}$. По этой же причине точка A_6 не лежит внутри треугольника $\triangle A_1A_5A_2$, габаритный прямоугольник которого задаётся формулой $\{(x,y): 0 \le x \le 10, -8 \le y \le 4\}$. Проверим отношение $A_6 \in \triangle A_3A_5A_2$:

$$\overrightarrow{A_2A_5}(-9,-12), \overrightarrow{A_2A_3}(-12,2), \overrightarrow{A_2A_6}(-9,-11),$$

$$\alpha\overrightarrow{A_2A_5} + \beta\overrightarrow{A_2A_3} = \overrightarrow{A_2A_6}:$$

$$\left(\begin{array}{cc|c} -9 & -12 & -9 \\ -12 & 2 & -11 \end{array}\right) \sim \left(\begin{array}{cc|c} 3 & 4 & 3 \\ -12 & 2 & -11 \end{array}\right) \sim \left(\begin{array}{cc|c} 3 & 4 & 3 \\ 0 & 18 & 1 \end{array}\right) \sim$$

$$\sim \left(\begin{array}{cc|c} 27 & 36 & 27 \\ 0 & 18 & 1 \end{array}\right) \sim \left(\begin{array}{cc|c} 27 & 0 & 25 \\ 0 & 18 & 1 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & 25/27 \\ 0 & 1 & 1/18 \end{array}\right),$$

$$\alpha > 0, \beta > 0, \alpha + \beta < 1 \Rightarrow A_6 \in \triangle A_3A_5A_2.$$

При добавлении точки внутрь некоторого треугольника граф G редактируется следующим образом:

- Удаляется вершина, отвечающая этому треугольнику. При этом все остальные вершины и рёбра графа остаются неизменными.
- Вместо неё вставляются три новые, попарно связные вершины, соответствующие новым треугольникам. Они инцидентны тем и только тем из старых рёбер, которые обозначают границы старого треугольника. В данном примере получается граф, представленный на рисунке 2.99.

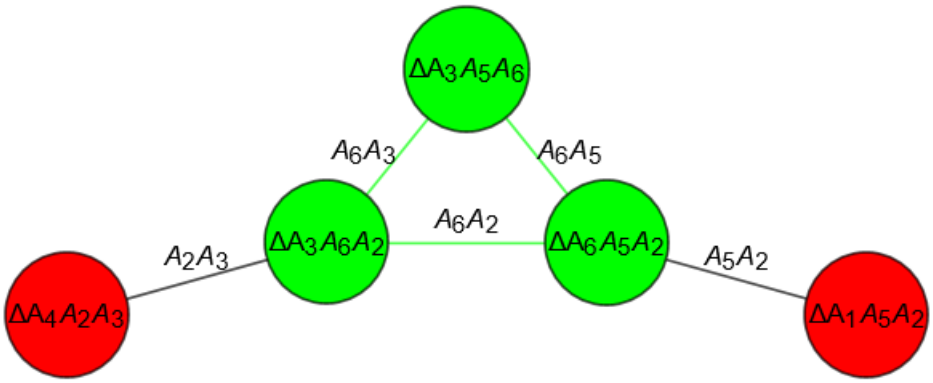


Рисунок 2.99 – Граф G после добавления новой точки

Список V пополняется рёбрами, которые образовывали границу старого треугольника: $V := \{A_3A_5, A_5A_2, A_2A_3\}$, выпуклая оболочка остаётся неизменной: $H = A_1A_2A_4A_3A_5A_1$. Сама триангуляция выглядит теперь, как на рисунке 2.100.

Ребро A_3A_5 оказывается ребром выпуклой оболочки и просто подлежит удалению: $V := \{A_5A_2, A_2A_3\}$.

Ребро A_5A_2 не является ребром выпуклой оболочки. При поиске в графе G соответствующего ребра находятся инцидентные ему вершины, соответствующие треугольникам $\triangle A_6A_5A_2$ и $\triangle A_1A_5A_2$. Проверка условия Делоне:

$$\overrightarrow{A_6A_5}(0,-1), \overrightarrow{A_6A_2}(9,11),$$

$$\overrightarrow{A_6A_5} \cdot \overrightarrow{A_6A_2} = -11 < 0,$$

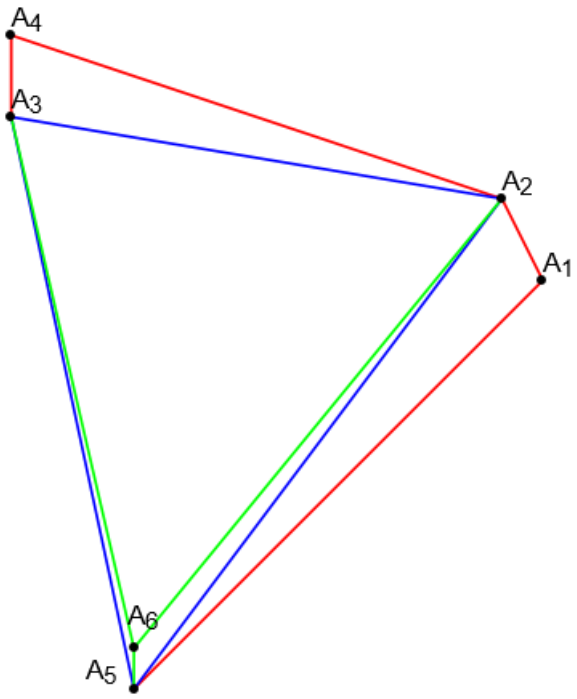
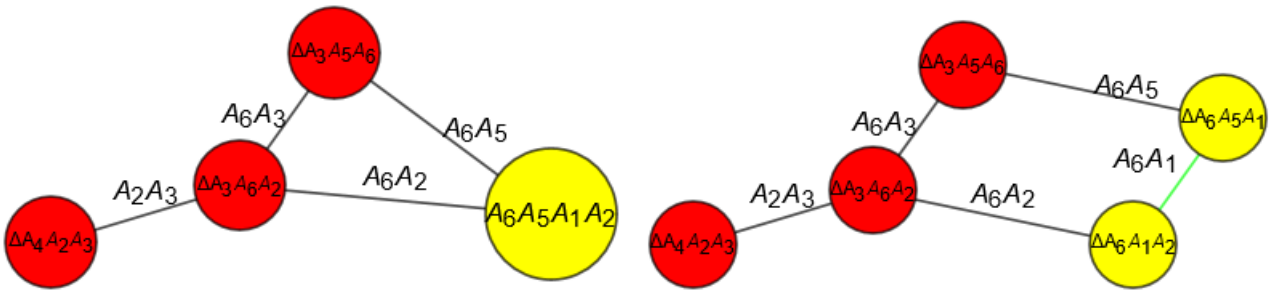


Рисунок 2.100 – Добавление новой точки внутрь треугольника триангуляции

$$\overrightarrow{A_1A_5} \cdot \overrightarrow{A_1A_2} = -10 \cdot (-1) - 10 \cdot 2 = -10 < 0.$$

Значит, имеем два тупых противолежащих угла, что сразу означает нарушение условия Делоне – необходимо редактирование триангуляции (Рисунок 2.101). Из списка V удаляется ребро A_5A_2, а затем добавляются A_5A_1 и A_1A_2: $V := \{A_2A_3, A_5A_1, A_1A_2\}$. Новая триангуляция представлена на рисунке 2.102.



а б

Рисунок 2.101 – Редактирование графа G (а – удаление старого ребра и слияние вершин графа, б – добавление нового ребра)

Переходя к ребру A_2A_3, обнаруживаем, во-первых, что оно является границей двух треугольников, во-вторых, что оба опирающихся на него угла являются острыми:

$$\overrightarrow{A_4A_2} \cdot \overrightarrow{A_4A_3} = 8 > 0,$$
$$\overrightarrow{A_6A_2}(9,11), \overrightarrow{A_6A_3}(-3,13),$$

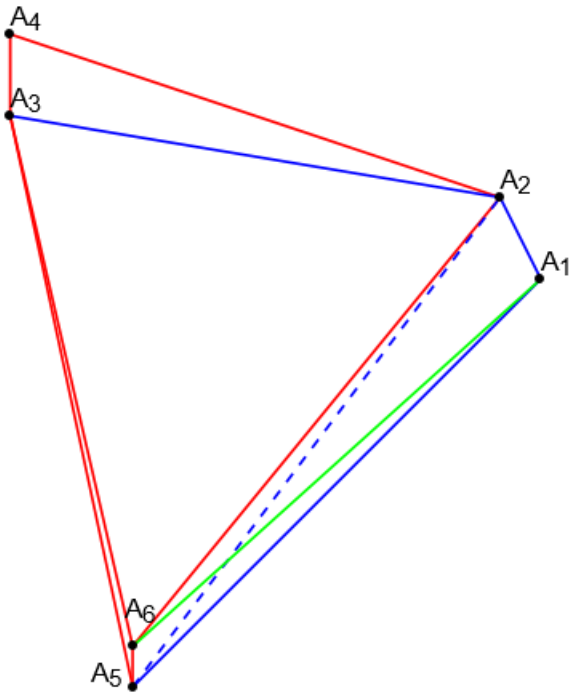


Рисунок 2.102 – Триангуляция после флипа ребра

$$\overrightarrow{A_6A_2}\cdot\overrightarrow{A_6A_3}=9\cdot(-3)+11\cdot 13>0.$$

Значит, ребро A_2A_3 просто удаляется из списка V. Все остальные рёбра также удаляются, поскольку они оказываются рёбрами выпуклой оболочки H. Таким образом, после четвёртой итерации имеем триангуляцию, представленную на рисунках 2.101б и 2.102.

На заключительной итерации добавляется точка $A_7(1,-2)$. При помощи, например, обычного перебора треугольников можно выяснить, что $A_7\in\triangle A_3A_6A_2$. Действительно,

$$\overrightarrow{A_6A_2}(9,11),\overrightarrow{A_6A_3}(-3,13),\overrightarrow{A_6A_7}(1,5),$$

$$\alpha\overrightarrow{A_6A_2}+\beta\overrightarrow{A_6A_3}=\overrightarrow{A_6A_7}:$$

$$\left(\begin{array}{cc|c}9&-3&1\\11&13&5\end{array}\right)\sim\left(\begin{array}{cc|c}9&-3&1\\33&39&15\end{array}\right)\sim\left(\begin{array}{cc|c}9&-3&1\\150&0&28\end{array}\right)\sim\left(\begin{array}{cc|c}9&-3&1\\150&0&28\end{array}\right)\sim$$

$$\sim\left(\begin{array}{cc|c}225&-75&25\\75&0&14\end{array}\right)\sim\left(\begin{array}{cc|c}0&-75&-17\\75&0&14\end{array}\right)\sim\left(\begin{array}{cc|c}1&0&14/75\\0&1&17/75\end{array}\right),$$

$$\alpha>0,\beta>0,\alpha+\beta<1.$$

Значит, проводятся новые рёбра и редактируется граф G (Рисунок 2.103). Список видимых рёбер: $V:=\{A_3A_2,A_2A_6,A_6A_3\}$, выпуклая оболочка: $H=A_1A_2A_4A_3A_5A_1$. Триангуляция после добавления новой точки представлена на рисунке 2.104.

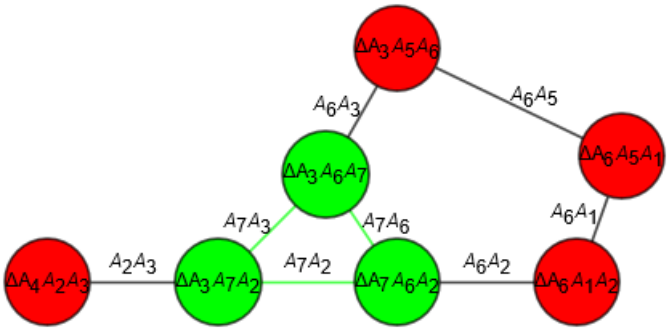


Рисунок 2.103 – Редактирование графа *G*
при добавлении вершины внутри треугольника

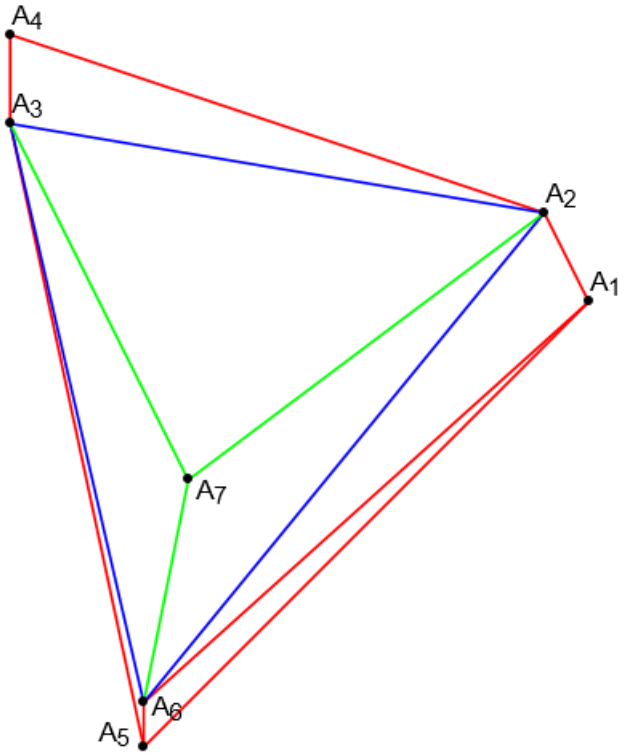


Рисунок 2.104 – Триангуляция после добавления вершины

Видимое ребро A_3A_2 является внутренним ребром триангуляции, которое служит границей для треугольников $\triangle A_4A_2A_3$ и $\triangle A_3A_7A_2$. У них противолежащие углы являются острыми:

$$\overrightarrow{A_4A_2}\cdot\overrightarrow{A_4A_3}=8>0,$$
$$\overrightarrow{A_7A_2}(8,6),\overrightarrow{A_7A_3}(-4,8),$$
$$\overrightarrow{A_7A_2}\cdot\overrightarrow{A_7A_3}=8\cdot(-4)+6\cdot8>0.$$

После этой проверки ребро A_3A_2 удаляется из списка V: $V:=\{A_2A_6,A_6A_3\}$. Для следующего ребра A_2A_6 имеем, что оно является граничным для треугольников $\triangle A_7A_6A_2$ и $\triangle A_6A_1A_2$. Проверка условия Делоне:

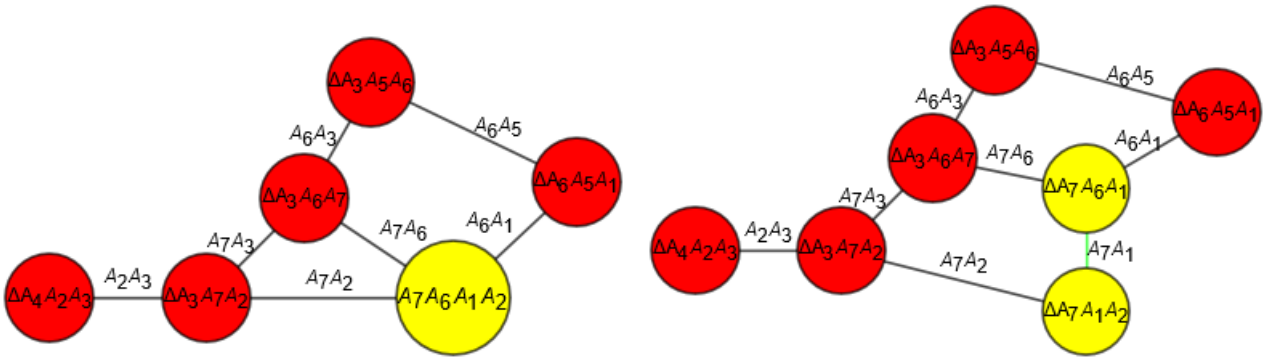
$$\overrightarrow{A_7A_2}(8,6), \overrightarrow{A_7A_6}(-1,-5) \Rightarrow \overrightarrow{A_7A_2} \cdot \overrightarrow{A_7A_6} = 8 \cdot (-1) + 6 \cdot (-5) = -38 < 0,$$

$$\overrightarrow{A_1A_2}(-1,2), \overrightarrow{A_1A_6}(-13,4) \Rightarrow \overrightarrow{A_1A_2} \cdot \overrightarrow{A_1A_6} = (-1) \cdot (-13) + 2 \cdot 4 = 21 > 0,$$

$$\left|\overrightarrow{A_7A_2} \times \overrightarrow{A_7A_6}\right| \left(\overrightarrow{A_1A_2} \cdot \overrightarrow{A_1A_6}\right) + \left(\overrightarrow{A_7A_2} \cdot \overrightarrow{A_7A_6}\right) \left|\overrightarrow{A_1A_2} \times \overrightarrow{A_1A_6}\right| =$$

$$= 34 \cdot 21 - 38 \cdot 22 = 21 \cdot (34 - 38) - 38 < 0.$$

Следовательно, нужно делать флип ребра. После него граф G имеет структуру, представленную на рисунке 2.105, а сама триангуляция выглядит как на рисунке 2.106. Список видимых рёбер становится равным $V := \{A_6A_3, A_2A_1, A_1A_6\}$.



а б

Рисунок 2.105 – Редактирование графа G при флипе ребра

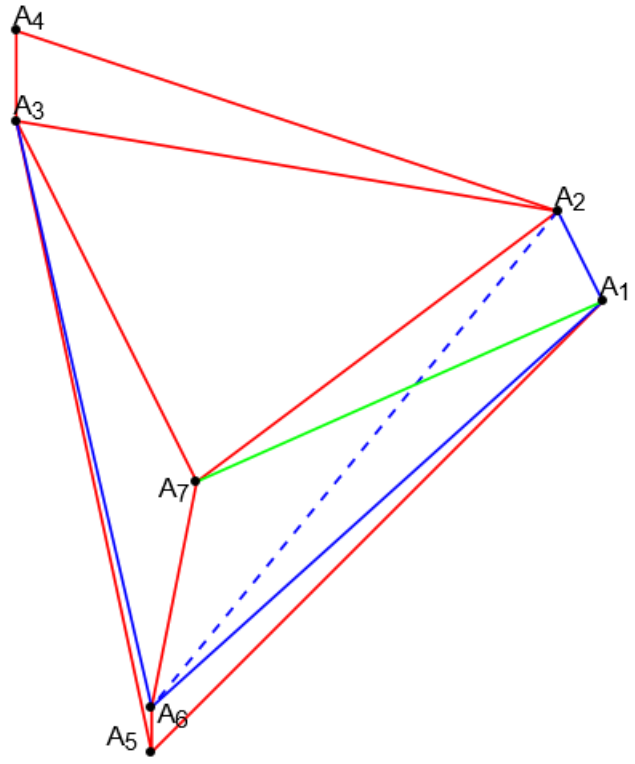


Рисунок 2.106 – Редактирование триангуляции при флипе ребра

Следующее видимое ребро A_6A_3 оказывается граничным для треугольников $\triangle A_3A_6A_7$ и $\triangle A_3A_5A_6$. Они удовлетворяют условию Делоне:

$$\overrightarrow{A_7A_6}(-1,-5), \overrightarrow{A_7A_3}(-4,8),$$
$$\overrightarrow{A_7A_6} \cdot \overrightarrow{A_7A_3} = (-1)\cdot(-4) - 5\cdot 8 = -36 < 0,$$
$$\overrightarrow{A_5A_6}(0,1), \overrightarrow{A_5A_3}(-3,14),$$
$$\overrightarrow{A_5A_6} \cdot \overrightarrow{A_5A_3} = 0\cdot(-3) + 1\cdot 14 = 14 > 0,$$
$$\left|\overrightarrow{A_7A_6} \times \overrightarrow{A_7A_3}\right|\left(\overrightarrow{A_5A_6} \cdot \overrightarrow{A_5A_3}\right) + \left(\overrightarrow{A_7A_6} \cdot \overrightarrow{A_7A_3}\right)\left|\overrightarrow{A_5A_6} \times \overrightarrow{A_5A_3}\right| = 28\cdot 14 - 36\cdot 3 > 0.$$

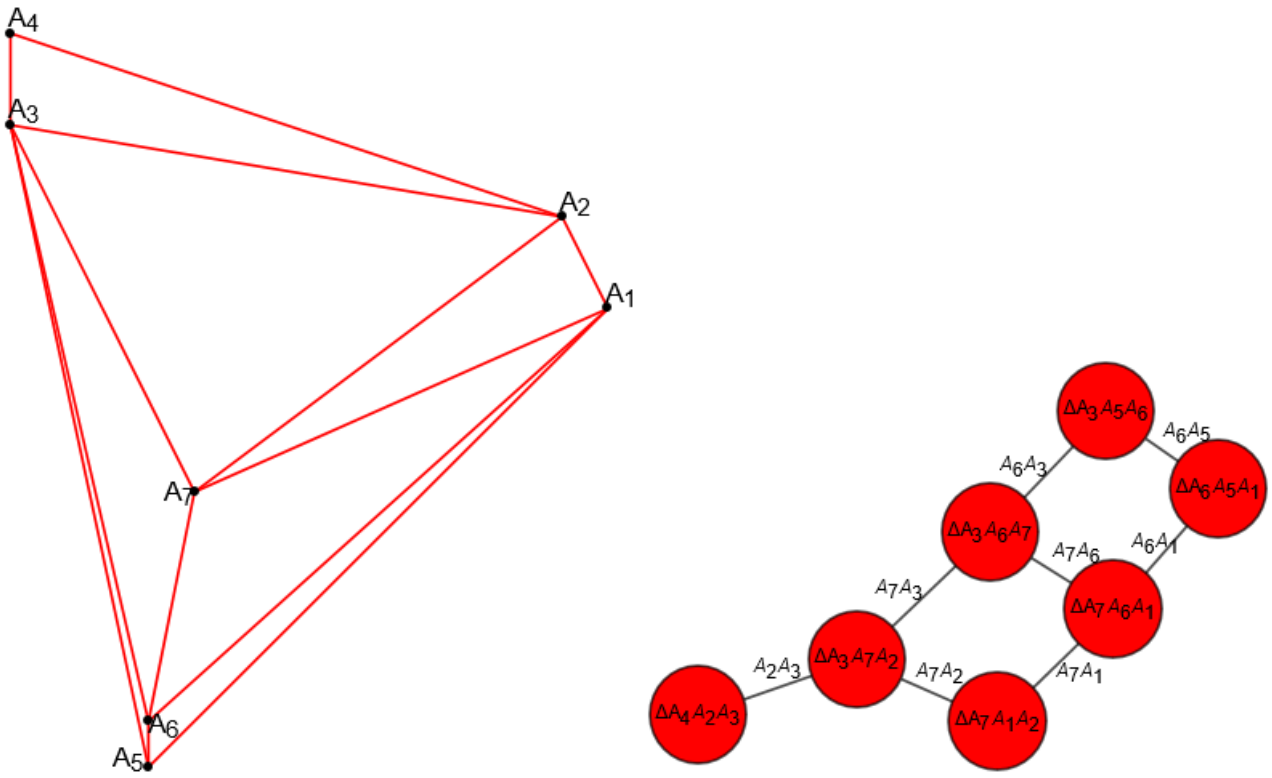
Текущее ребро удаляется: $V := \{A_2A_1, A_1A_6\}$. Следующее ребро, A_2A_1, является ребром выпуклой оболочки $H = A_1A_2A_4A_3A_5A_1$, оно также удаляется: $V := \{A_1A_6\}$. Наконец, ребро A_1A_6 является внутренним ребром триангуляции при смежных треугольниках $\triangle A_7A_6A_1$ и $\triangle A_6A_5A_1$, которые удовлетворяют условию Делоне:

$$\overrightarrow{A_7A_1}(9,4), \overrightarrow{A_7A_6}(-1,-5),$$
$$\overrightarrow{A_7A_1} \cdot \overrightarrow{A_7A_6} = 9\cdot(-1) + 4\cdot(-5) = -29 < 0,$$
$$\overrightarrow{A_5A_1}(10,10), \overrightarrow{A_5A_6}(0,1),$$
$$\overrightarrow{A_5A_1} \cdot \overrightarrow{A_5A_6} = 10 > 0,$$
$$\left|\overrightarrow{A_7A_1} \times \overrightarrow{A_7A_6}\right|\left(\overrightarrow{A_5A_1} \cdot \overrightarrow{A_5A_6}\right) + \left(\overrightarrow{A_7A_1} \cdot \overrightarrow{A_7A_6}\right)\left|\overrightarrow{A_5A_1} \times \overrightarrow{A_5A_6}\right| = 41\cdot 10 - 29\cdot 10 > 0.$$

Таким образом, окончательно получим триангуляцию, изображённую на рисунке 2.107а. Она представима графом, изображённым на рисунке 2.107б.

2.6.2. Алгоритм слияния

Один из алгоритмов построения триангуляции Делоне состоят из таких этапов как разделение множества точек (стр. 173), объединение их выпуклых оболочек (стр. 173) и объединение триангуляций двух подмножеств (стр. 175). В примере ниже эти алгоритмы, а также быстрый алгоритм построения триангуляции Делоне в элементарных случаях, рассмотрены более подробно.



а б

Рисунок 2.107 – Итоговая триангуляция (а) и её графовое представление (б)

Пример 6.2. Построить триангуляцию точек из примера 6.1 алгоритмом слияния.

Для эффективного разбиения исходного множества точек, которое должно давать две непересекающиеся триангуляции, вначале отсортируем всё множество точек, например, в порядке возрастания их абсцисс:

$$A_1(10,2), A_2(9,4), A_3(-3,6), A_4(-3,8), A_5(0,-8), A_6(0,-7), A_7(1,-2)$$

$$\Downarrow$$

$$B_1(-3,6), B_2(-3,8), B_3(0,-8), B_4(0,-7), B_5(1,-2), B_6(9,4), B_7(10,2).$$

Множество точек $\{B_i\}_{i=1}^{7}$ разбивается на левое $\{B_1, B_2, B_3\}$ и правое $\{B_4, B_5, B_6, B_7\}$ подмножества. Для левого подмножества триангуляция Делоне описывается совокупностью рёбер $T_L := \{B_1B_2, B_2B_3, B_1B_3\}$, а выпуклая оболочка с левосторонним обходом представляется последовательностью вершин $H_L := B_3B_2B_1B_3$, что определяется через знак угла между направляющими векторами рёбер треугольника $\triangle B_1B_2B_3$:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_2}, \overrightarrow{B_2B_3}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 2 \\ 3 & -16 \end{vmatrix} = -1.$$

Для другого подмножества, состоящего из четырёх точек, необходимо вначале найти выпуклую оболочку. Это может быть проделано с использованием следующего алгоритма:

1. Среди четырёх точек определяются крайние точки (верхняя, нижняя, правая, левая). Их может оказаться от 2 до 4.

2. Если три точки одновременно оказались крайними верхними / нижними / правыми / левыми, то выпуклая оболочка является треугольной (Рисунок 2.108а). В противном случае рассматриваются шаги 3–5, приведённые ниже.

3. Если крайних точек оказалось всего 4, то они все образуют выпуклую оболочку (Рисунок 2.108б).

4. Если оказались 3 крайние точки, то определяется, принадлежит ли четвёртая точка треугольнику с вершинами в них. В зависимости от этого получается выпуклая оболочка из трёх либо четырёх вершин (рисунки 2.108, в-г).

5. В случае 2 крайних точек определяется, по какую сторону от прямой, проведённой через них, лежат две другие точки. Если они лежат по одну сторону, и к тому же более близко расположенная точка оказывается внутри треугольника из трёх других точек, то выпуклая оболочка имеет треугольную форму (Рисунок 2.108д). В остальных же случаях выпуклая оболочка образуется всеми четырьмя точками (рисунки 2.108, е-ж).

6. Из полученных вершин выпуклой оболочки составляется последовательность, обеспечивающая левосторонний обход (например, для выпуклой

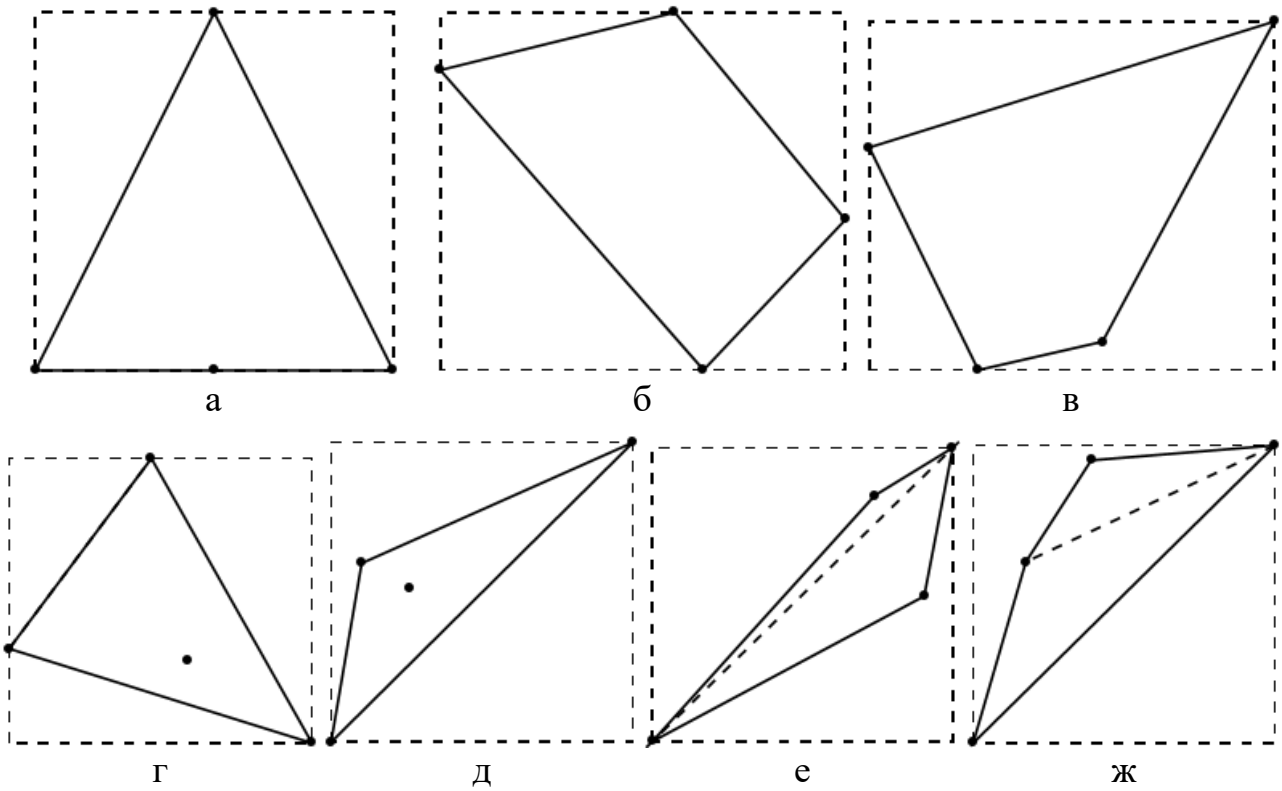


Рисунок 2.108 – Варианты построения выпуклой оболочки четырёх точек
(а – три точки расположены на одной границе габаритного прямоугольника,
б – все точки являются крайними, в-г – три точки являются крайними,
д-ж – две точки являются крайними)

оболочки на рисунке 2.108б обход может начинаться от верхней вершины и проходить через левую, нижнюю и правую вершину).

В данном примере получаются следующие крайние вершины (верхняя, нижняя, правая и левая соответственно):

$$T = B_6(9,4),\ B = B_4(0,-7),\ R = B_7(10,2),\ L = B_4(0,-7).$$

Значит, имеем три крайние вершины, и нужно проверить, принадлежит ли точка B_5 треугольнику $\triangle B_4B_6B_7$:

$$\overrightarrow{B_4B_5}(1,5), \overrightarrow{B_4B_6}(9,11), \overrightarrow{B_4B_7}(10,9),$$
$$\alpha\overrightarrow{B_4B_6} + \beta\overrightarrow{B_4B_7} = \overrightarrow{B_4B_5}:$$
$$\left(\begin{array}{cc|c} 9 & 10 & 1 \\ 11 & 9 & 5 \end{array}\right) \sim \left(\begin{array}{cc|c} 9 & 10 & 1 \\ 2 & -1 & 4 \end{array}\right) \sim \left(\begin{array}{cc|c} 29 & 0 & 41 \\ 2 & -1 & 4 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & 41/29 \\ 2 & -1 & 4 \end{array}\right) \Rightarrow \alpha = \frac{41}{29} \Rightarrow$$
$$\Rightarrow \alpha + \beta > 1 \vee \beta < 0 \Rightarrow B_5 \notin \triangle B_4B_6B_7.$$

Это означает, что выпуклая оболочка состоит из всех вершин $B_i,\ i = \overline{4,7}$. Ясно, что в треугольнике $\triangle B_4B_6B_7$ последовательность вершин $B_6B_4B_7$ обеспечивает левосторонний обход (от верхней точки через точку в левом нижнем углу к правой точке). В эту последовательность нужно вставить точку B_5 таким образом, чтобы в ней происходил поворот налево. Перебираем все варианты, пока не найдём благоприятный:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_6B_5}, \overrightarrow{B_5B_4}\right) = \operatorname{sgn}\begin{vmatrix} -8 & -6 \\ -1 & -5 \end{vmatrix} = +1.$$

Таким образом, выпуклой оболочкой правого подмножества исходного множества точек является четырёхугольник $H_R := B_6B_5B_4B_7B_6$. Проверим условие Делоне, например, для треугольников $\triangle B_6B_5B_4$ и $\triangle B_6B_7B_4$:

$$\overrightarrow{B_5B_6}(8,6), \overrightarrow{B_5B_4}(-1,-5),$$
$$\overrightarrow{B_5B_6} \cdot \overrightarrow{B_5B_4} = -38 < 0,$$
$$\overrightarrow{B_7B_6}(-1,2), \overrightarrow{B_7B_4}(-10,-9),$$
$$\overrightarrow{B_7B_6} \cdot \overrightarrow{B_7B_4} = -8 < 0.$$

Значит, для этих двух треугольников условие Делоне не выполняется – нужно в четырёхугольнике провести другую диагональ. Таким образом, в правом

подмножестве получается триангуляция с рёбрами $T_R := \{B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7\}$. Обе триангуляции представлены на рисунке 2.109.

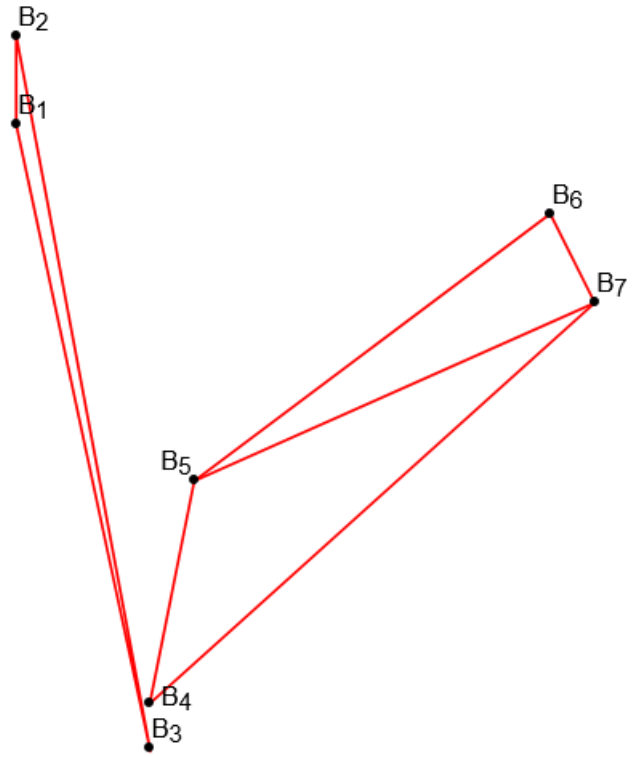


Рисунок 2.109 – Триангуляции двух подмножеств

Теперь необходимо объединить эти триангуляции. Для начала нужно объединить их выпуклые оболочки, в частности, найти их **нижнюю и верхнюю касательные**. В данном примере для выпуклых оболочек $H_L = B_3B_2B_1B_3$ и $H_R = B_6B_5B_4B_7B_6$ получаются следующие вычисления:

$$L'R' := B_3B_4,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_4}, \overrightarrow{B_3B_7}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 1 \\ 10 & 10 \end{vmatrix} = -1 \Rightarrow L'R' := B_3B_7,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_7}, \overrightarrow{B_3B_6}\right) = \operatorname{sgn}\begin{vmatrix} 10 & 10 \\ 9 & 12 \end{vmatrix} = +1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_7}, \overrightarrow{B_3B_1}\right) = \operatorname{sgn}\begin{vmatrix} 10 & 10 \\ -3 & 14 \end{vmatrix} = +1 \Rightarrow L_BR_B = B_3B_7.$$

Аналогичным образом находится верхняя касательная:

$$L'R' := B_3B_4,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_4}, \overrightarrow{B_3B_5}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 1 \\ 1 & 6 \end{vmatrix} = -1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_4},\overrightarrow{B_3B_2}\right)=\operatorname{sgn}\begin{vmatrix}0 & 1\\ -3 & 16\end{vmatrix}=+1\Rightarrow L'R':=B_2B_4,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_2B_4},\overrightarrow{B_2B_5}\right)=\operatorname{sgn}\begin{vmatrix}3 & -15\\ 4 & -10\end{vmatrix}=+1\Rightarrow L'R':=B_2B_5,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_2B_5},\overrightarrow{B_2B_6}\right)=\operatorname{sgn}\begin{vmatrix}4 & -10\\ 12 & -4\end{vmatrix}=+1\Rightarrow L'R':=B_2B_6,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_2B_6},\overrightarrow{B_2B_7}\right)=\operatorname{sgn}\begin{vmatrix}12 & -4\\ 13 & -6\end{vmatrix}=-1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_2B_6},\overrightarrow{B_2B_1}\right)=\operatorname{sgn}\begin{vmatrix}12 & -4\\ 0 & -2\end{vmatrix}=-1\Rightarrow L_TR_T=B_2B_6.$$

Все сдвиги в ходе вычисления обеих касательных проиллюстрированы на рисунке 2.110. Итоговая выпуклая оболочка равна $H_L\cup H_R=B_7B_6B_2B_1B_3B_7$.

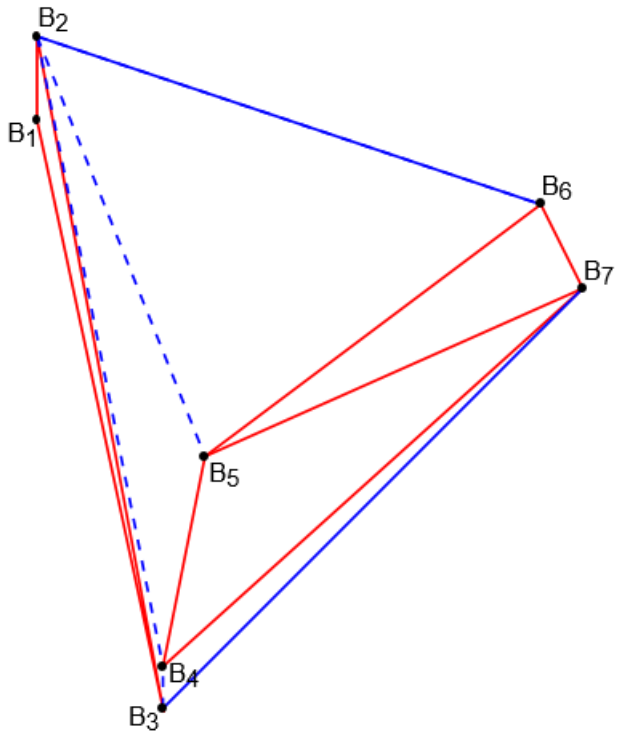


Рисунок 2.110 – Нахождение нижней и верхней касательных между двумя выпуклыми оболочками

Теперь приступим к объединению двух триангуляций T_L и T_R. В данном примере первоначальное объединение рёбер обеих триангуляций даёт множество $T=T_L\cup T_R=\{B_1B_2, B_2B_3, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7\}$. Сюда добавляется текущая нижняя касательная B_3B_7. Затем проверяется, не нужно ли удалять какие-либо рёбра, инцидентные вершинам B_7 и B_3.

Для вершины B_7 имеем смежные вершины B_4, B_6 и B_5. Отсортируем их таким образом, чтобы три вектора, проведённые к этим точкам от точки B_7, по порядку давали отрицательные углы. Это делается при помощи сортировки векторов $\overrightarrow{B_7B_3}$, $\overrightarrow{B_7B_4}$, $\overrightarrow{B_7B_5}$, $\overrightarrow{B_7B_6}$ через их квадранты и угловые коэффициенты (см. стр. 304):

$$\overrightarrow{B_7B_3}(-10,-10) \Rightarrow \text{III квадрант, } k = 1,$$
$$\overrightarrow{B_7B_4}(-10,-9) \Rightarrow \text{III квадрант, } k = 9/10,$$
$$\overrightarrow{B_7B_5}(-9,-4) \Rightarrow \text{III квадрант, } k = 4/9,$$
$$\overrightarrow{B_7B_6}(-1,2) \Rightarrow \text{II квадрант, } k = -2.$$

При сортировке векторов по направлению движения часовой стрелки они упорядочиваются в порядке убывания квадрантов, а векторы в пределах одного квадранта – в порядке убывания углового коэффициента. Отсюда получаем последовательность точек B_3, B_4, B_5, B_6. Если бы левый конец касательной B_3 не оказался бы на первом месте, то все точки перед ним следовало бы переместить в конец списка.

Затем проверяется, следует ли удалить ребро B_7B_4. В случае его удаления перейдём к ребру B_7B_5, а если и его нужно будет удалить, то останется крайнее ребро B_7B_6, оно удалению не подлежит.

Рассматривая ребро B_7B_4, отметим, во-первых, что все точки B_4, B_5 и B_6 расположены левее ребра B_3B_7 в силу того, что это ребро – исходная нижняя касательная между двумя выпуклыми оболочками. Во-вторых, проверим условие Делоне для треугольников $\triangle B_3B_7B_4$ и $\triangle B_7B_4B_5$:

$$\overrightarrow{B_3B_7}(10,10), \overrightarrow{B_3B_4}(0,1),$$
$$\overrightarrow{B_3B_7} \cdot \overrightarrow{B_3B_4} = 10 > 0,$$
$$\overrightarrow{B_5B_7}(9,4), \overrightarrow{B_5B_4}(-1,-5),$$
$$\overrightarrow{B_5B_7} \cdot \overrightarrow{B_5B_4} = -29 < 0,$$
$$\left|\overrightarrow{B_3B_7} \times \overrightarrow{B_3B_4}\right|\left(\overrightarrow{B_5B_7} \cdot \overrightarrow{B_5B_4}\right) + \left(\overrightarrow{B_3B_7} \cdot \overrightarrow{B_3B_4}\right)\left|\overrightarrow{B_5B_7} \times \overrightarrow{B_5B_4}\right| = 10 \cdot (-29) + 10 \cdot 41 > 0.$$

Таким образом, нарушения условия Делоне не происходит, а значит, рёбра B_7B_4, B_7B_5 и B_7B_6 пока остаются.

Аналогично рассмотрим рёбра из триангуляции левого подмножества. Всего три вершины, смежные с B_3: B_7, B_2 и B_1. Отсортируем этот список так, чтобы векторы, проведённые из B_3 к этим точкам, поворачивались друг к другу на положительные углы:

$$\overrightarrow{B_3B_7}(10,10) \Rightarrow \text{I квадрант}, k = 1,$$
$$\overrightarrow{B_3B_2}(-3,16) \Rightarrow \text{II квадрант}, k = -16/3,$$
$$\overrightarrow{B_3B_1}(-3,14) \Rightarrow \text{II квадрант}, k = -14/3.$$

При сортировке векторов против направления движения часовой стрелки они упорядочиваются в порядке возрастания квадрантов, а векторы в пределах одного квадранта – в порядке возрастания углового коэффициента. Значит, получается последовательность точек B_7, B_2, B_1. Опять же, обе вершины B_2 и B_1 расположены слева от ребра B_3B_7. Проверка условия Делоне для треугольников $\triangle B_3B_7B_2$ и $\triangle B_3B_2B_1$ даёт следующий результат:

$$\overrightarrow{B_7B_3}(-10,-10), \overrightarrow{B_7B_2}(-13,6),$$
$$\overrightarrow{B_7B_3} \cdot \overrightarrow{B_7B_2} = 70 > 0,$$
$$\overrightarrow{B_1B_3}(3,-14), \overrightarrow{B_1B_2}(0,2),$$
$$\overrightarrow{B_1B_3} \cdot \overrightarrow{B_1B_2} = -28 < 0,$$
$$\left|\overrightarrow{B_7B_3} \times \overrightarrow{B_7B_2}\right|\left(\overrightarrow{B_1B_3} \cdot \overrightarrow{B_1B_2}\right) + \left(\overrightarrow{B_7B_3} \cdot \overrightarrow{B_7B_2}\right)\left|\overrightarrow{B_1B_3} \times \overrightarrow{B_1B_2}\right| = 190 \cdot (-28) + 70 \cdot 6 < 0.$$

Так как условие Делоне нарушено, то необходимо удалить ребро B_3B_2. Далее следует ребро B_3B_1, но оно остаётся, так как является последним.

Теперь нужно выяснить, как изменится нижняя касательная: сдвинется либо левый конец от точки B_3 к B_1, либо правый конец от точки B_7 к B_4. По порядку проверим условия из пунктов 11а–11г алгоритма со стр. 175:

$$B_4 \in \triangle B_3B_7B_1 - ?$$
$$\overrightarrow{B_3B_7}(10,10), \overrightarrow{B_3B_1}(-3,14), \overrightarrow{B_3B_4}(0,1),$$
$$\alpha\overrightarrow{B_3B_7} + \beta\overrightarrow{B_3B_1} = \overrightarrow{B_3B_4},$$
$$\left(\begin{array}{cc|c} 10 & -3 & 0 \\ 10 & 14 & 1 \end{array}\right) \sim \left(\begin{array}{cc|c} 10 & -3 & 0 \\ 0 & 17 & 1 \end{array}\right) \sim \left(\begin{array}{cc|c} 10 & -3 & 0 \\ 0 & 1 & 1/17 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & 3/170 \\ 0 & 1 & 1/17 \end{array}\right),$$
$$\alpha = 3/170, \beta = 1/17 \Rightarrow \alpha > 0, \beta > 0, \alpha + \beta < 1 \Rightarrow B_4 \in \triangle B_3B_7B_1.$$

Таким образом, получаем сразу, что сдвигается правый конец нижней касательной, и на следующей итерации будем рассматривать касательную B_3B_4. После проведения первой итерации имеем список рёбер $T = \{B_1B_2, \cancel{B_2B_3}, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7, \underline{B_3B_7}\}$. На рисунке 2.111 представлен результат редактирования списка T: красным отмечены все старые рёбра, красным

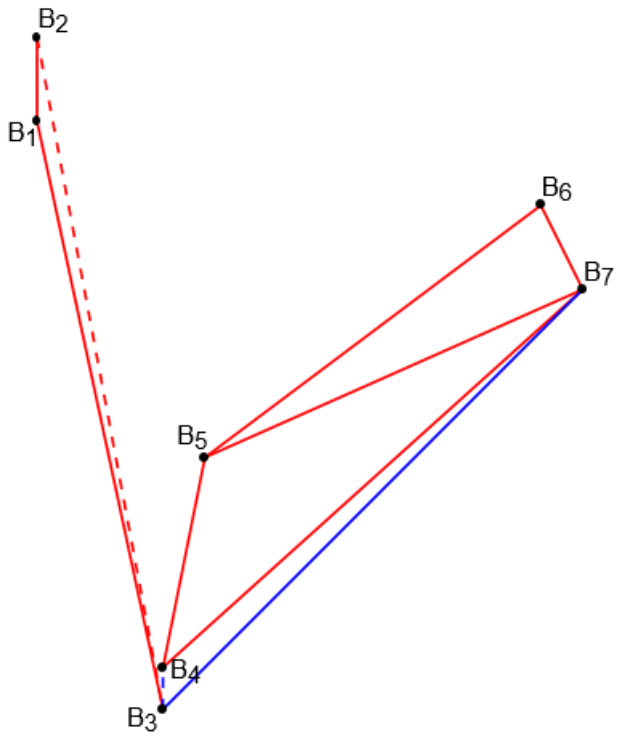


Рисунок 2.111 – Рёбра триангуляции после первой итерации

пунктиром – удалённое ребро, синим – новое ребро, синим пунктиром – новая нижняя касательная.

На следующей итерации добавляем касательную B_3B_4 в список T. Её правый конец является смежным с вершинами B_3, B_5, B_7. Отсортируем эти точки:

$$\overrightarrow{B_4B_3}(0,-1) \Rightarrow \text{лежит на границе III и IV квадрантов}, k=-\infty,$$
$$\overrightarrow{B_4B_5}(1,5) \Rightarrow \text{I квадрант}, k=5,$$
$$\overrightarrow{B_4B_7}(10,9) \Rightarrow \text{I квадрант}, k=9/10.$$

Значит, получается последовательность из точек B_3, B_5, B_7. Проверим положение вершины B_5 относительно ребра B_3B_4:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_3B_4}, \overrightarrow{B_3B_5}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 1 \\ 1 & 6 \end{vmatrix} = -1.$$

Отсюда сразу следует, что, во-первых, никакое ребро, проведённое из вершины B_4, не подлежит удалению на этой итерации; во-вторых, у касательной будет изменён левый конец. Осталось только определить, какие рёбра нужно удалить из тех, что проведены из левого конца B_3. В списке T уже записаны рёбра B_3B_1, B_3B_7 и B_3B_4:

$$\overrightarrow{B_3B_1}(-3,14) \Rightarrow \text{II квадрант}, k = -14/3,$$
$$\overrightarrow{B_3B_7}(10,10) \Rightarrow \text{I квадрант}, k = 1,$$
$$\overrightarrow{B_3B_4}(0,1) \Rightarrow \text{лежит на границе I и II квадрантов}, k = +\infty.$$

Отсюда имеем последовательность точек B_7, B_4, B_1. Первой точкой должен стать правый конец текущей касательной, значит, последовательность изменяется следующим образом: B_4, B_1, B_7. Так как

$$\operatorname{sgn} \angle\left(\overrightarrow{B_3B_4}, \overrightarrow{B_3B_1}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 1 \\ -3 & 14 \end{vmatrix} = +1,$$

то точка B_1 расположена слева от ребра B_3B_4. Следующая вершина, однако, располагается уже справа от него:

$$\operatorname{sgn} \angle\left(\overrightarrow{B_3B_4}, \overrightarrow{B_3B_7}\right) = \operatorname{sgn}\begin{vmatrix} 0 & 1 \\ 10 & 10 \end{vmatrix} = -1,$$

что означает, что ни ребро B_3B_1, ни все следующие за ним не подлежат удалению. С учётом всего сказанного выше, получаем, что в результате данной итерации имеем рёбра $T = \{B_1B_2, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7, B_3B_7, \underline{B_3B_4}\}$ и новую нижнюю касательную B_1B_4 (Рисунок 2.112).

На следующей итерации после добавления ребра B_1B_4 рассмотрим все рёбра, смежные правому концу B_4, т.е. B_4B_5, B_4B_7, B_4B_3, B_4B_1:

$$\overrightarrow{B_4B_5}(1,5) \Rightarrow \text{I квадрант}, k = 5,$$
$$\overrightarrow{B_4B_7}(10,9) \Rightarrow \text{I квадрант}, k = 9/10,$$
$$\overrightarrow{B_4B_3}(0,-1) \Rightarrow \text{лежит на границе III и IV квадрантов}, k = -\infty,$$
$$\overrightarrow{B_4B_1}(-3,13) \Rightarrow \text{II квадрант}, k = -13/3.$$

Значит, получается список из точек B_3, B_1, B_5, B_7, а после смещения элементов списка после левого конца касательной – B_1, B_5, B_7, B_3. Проверим, какие из вершин этого списка находятся левее касательной B_1B_4:

$$\operatorname{sgn} \angle\left(\overrightarrow{B_1B_4}, \overrightarrow{B_1B_5}\right) = \operatorname{sgn}\begin{vmatrix} 3 & -13 \\ 4 & -8 \end{vmatrix} = +1,$$

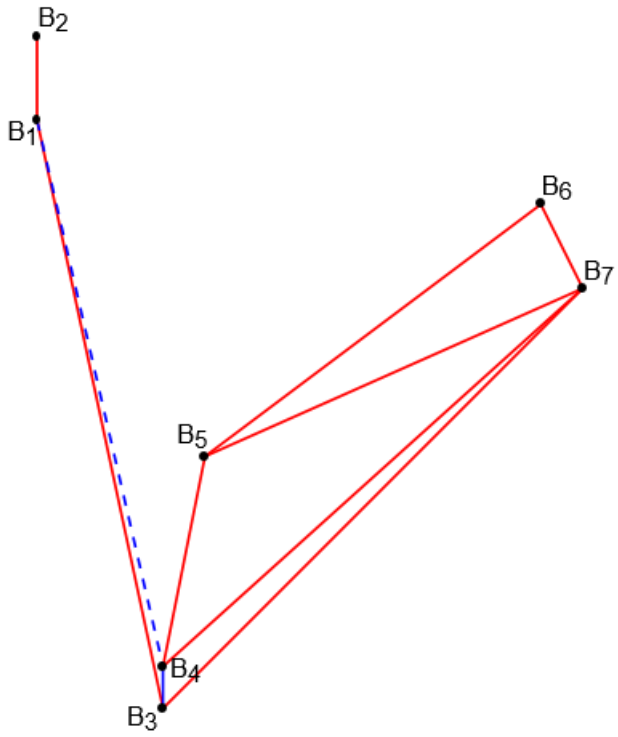


Рисунок 2.112 – Рёбра текущей триангуляции после второй итерации

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_4},\overrightarrow{B_1B_7}\right)=\operatorname{sgn}\begin{vmatrix}3 & -13\\ 13 & -4\end{vmatrix}=+1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_4},\overrightarrow{B_1B_3}\right)=\operatorname{sgn}\begin{vmatrix}3 & -13\\ 3 & -14\end{vmatrix}=-1.$$

Значит, все рёбра, начиная от B_4B_7, не подлежат удалению на данной итерации (т.к. после точки B_7 в списке идёт точка, лежащая по правую сторону от касательной B_1B_4). Остаётся проверить ребро B_4B_5, для чего нужно проверить условие Делоне для треугольников $\triangle B_1B_4B_5$ и $\triangle B_4B_5B_7$:

$$\overrightarrow{B_1B_4}(3,-13),\overrightarrow{B_1B_5}(4,-8),$$
$$\overrightarrow{B_1B_4}\cdot\overrightarrow{B_1B_5}=116>0,$$
$$\overrightarrow{B_7B_4}(-10,-9),\overrightarrow{B_7B_5}(-9,-4),$$
$$\overrightarrow{B_7B_4}\cdot\overrightarrow{B_7B_5}=126>0.$$

Отсюда сразу следует выполнение условия Делоне, и ребро B_4B_5 удалению не подлежит.

Аналогично рассматривая левый конец касательной B_1, имеем, что он инцидентен рёбрам B_1B_2, B_1B_3, B_1B_4:

$$\overrightarrow{B_1B_2}(0,2) \Rightarrow \text{лежит на границе I и II квадрантов, } k = +\infty,$$
$$\overrightarrow{B_1B_3}(3,-14) \Rightarrow \text{IV квадрант, } k = -14/3,$$
$$\overrightarrow{B_1B_4}(3,-13) \Rightarrow \text{IV квадрант, } k = -13/3.$$

Имеет место сортировка точек в следующем порядке: B_2, B_3, B_4, а после переноса всех вершин перед вершиной B_4 в конец списка – B_4, B_2, B_3. Вершины B_2 и B_3 расположены относительно касательной следующим образом:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_4}, \overrightarrow{B_1B_2}\right) = \operatorname{sgn}\begin{vmatrix} 3 & -13 \\ 0 & 2 \end{vmatrix} = +1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_4}, \overrightarrow{B_1B_3}\right) = \operatorname{sgn}\begin{vmatrix} 3 & -13 \\ 3 & -14 \end{vmatrix} = -1,$$

т.е. B_2 – крайняя вершина в левой полуплоскости от ребра B_1B_4, а значит, никакое из рёбер B_1B_2 и B_1B_3 удалению не подлежит. Итак, никакое ребро на этой итерации не оказалось удалено.

Теперь необходимо выяснить, как сдвинется касательная: от точки B_4 к B_5 или от точки B_1 к B_2. Для этого проверяем все возможные условия:

$$B_5 \in_{\triangle} B_1B_4B_2 - ?$$
$$\overrightarrow{B_1B_4}(3,-13), \overrightarrow{B_1B_2}(0,2), \overrightarrow{B_1B_5}(4,-8),$$
$$\alpha\overrightarrow{B_1B_4} + \beta\overrightarrow{B_1B_2} = \overrightarrow{B_1B_5}:$$
$$\left(\begin{array}{cc|c} 3 & 0 & 4 \\ -13 & 2 & -8 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & 4/3 \\ -13 & 2 & -8 \end{array}\right) \Rightarrow \alpha = \frac{4}{3} \Rightarrow \alpha + \beta > 1 \vee \beta < 0 \Rightarrow B_5 \notin_{\triangle} B_1B_4B_2,$$
$$B_2 \in_{\triangle} B_1B_4B_5 - ?,$$
$$\alpha\overrightarrow{B_1B_4} + \beta\overrightarrow{B_1B_5} = \overrightarrow{B_1B_2}:$$
$$\left(\begin{array}{cc|c} 3 & 4 & 0 \\ -13 & -8 & 2 \end{array}\right) \sim \left(\begin{array}{cc|c} 3 & 4 & 0 \\ -7 & 0 & 2 \end{array}\right) \sim \left(\begin{array}{cc|c} 3 & 4 & 0 \\ 1 & 0 & -2/7 \end{array}\right) \Rightarrow \alpha = -\frac{2}{7} < 0 \Rightarrow B_2 \notin_{\triangle} B_1B_4B_5,$$
$$\triangle B_1B_4B_2, \triangle B_4B_2B_5:$$
$$\overrightarrow{B_1B_4} \cdot \overrightarrow{B_1B_2} = -26 < 0,$$
$$\overrightarrow{B_5B_4}(-1,-5) \cdot \overrightarrow{B_5B_2}(-4,10) = -46 < 0.$$

Все эти условия означают, что так как ни одна из точек не попала в треугольник из трёх других, а также что треугольники $\triangle B_1B_4B_2$ и $\triangle B_4B_2B_5$ с общей стороной B_4B_2 не удовлетворяют условию Делоне, то новой касательной

становится ребро B_1B_5. В результате этой итерации имеет место список рёбер $T=\{B_1B_2, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7, B_3B_7, B_3B_4, \underline{B_1B_4}\}$ и новая касательная B_1B_5 (Рисунок 2.113).

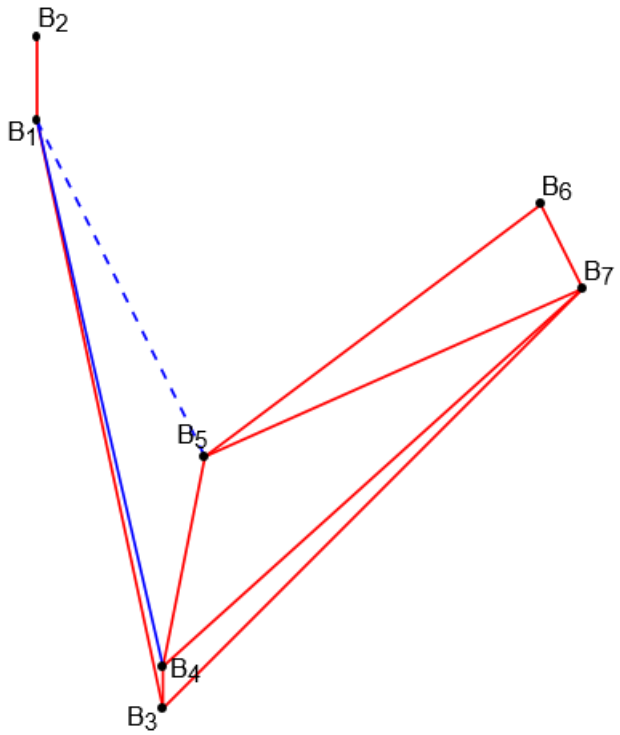


Рисунок 2.113 – Рёбра триангуляции после третьей итерации

На следующей итерации добавляется новое ребро B_1B_5. Его правый конец инцидентен рёбрам B_5B_6, B_5B_4, B_5B_7, B_5B_1. Их сортировка даёт следующий результат:

$$\overrightarrow{B_5B_6}(8,6) \Rightarrow \text{I квадрант}, k=3/4,$$
$$\overrightarrow{B_5B_4}(-1,-5) \Rightarrow \text{III квадрант}, k=5,$$
$$\overrightarrow{B_5B_7}(9,4) \Rightarrow \text{I квадрант}, k=4/9,$$
$$\overrightarrow{B_5B_1}(-4,8) \Rightarrow \text{II квадрант}, k=-2,$$

т.е. последовательность точек B_4, B_1, B_6, B_7, которая преобразуется к виду B_1, B_6, B_7, B_4. Проверяем расположение этих точек относительно касательной:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_5}, \overrightarrow{B_1B_6}\right) = \operatorname{sgn}\begin{vmatrix} 4 & -8 \\ 12 & -2 \end{vmatrix} = +1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_5}, \overrightarrow{B_1B_7}\right) = \operatorname{sgn}\begin{vmatrix} 4 & -8 \\ 13 & -4 \end{vmatrix} = +1,$$

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_5},\overrightarrow{B_1B_4}\right)=\operatorname{sgn}\begin{vmatrix}4 & -8\\ 3 & -13\end{vmatrix}-1,$$

откуда сразу следует, что все рёбра после B_5B_6 на этой итерации остаются в списке. Для проверки, подлежит ли удалению ребро B_5B_6, снова используется условие Делоне:

$$\triangle B_1B_5B_6, \triangle B_5B_6B_7:$$
$$\overrightarrow{B_1B_5}(4,-8), \overrightarrow{B_1B_6}(12,-2),$$
$$\overrightarrow{B_1B_5}\cdot\overrightarrow{B_1B_6}=64>0,$$
$$\overrightarrow{B_7B_5}(-9,-4), \overrightarrow{B_7B_6}(-1,2),$$
$$\overrightarrow{B_7B_5}\cdot\overrightarrow{B_7B_6}=1>0.$$

Условие Делоне выполняется, а значит, ребро B_5B_6 также не подлежит удалению.

Для левого конца касательной B_1 имеем рёбра B_1B_2, B_1B_3, B_1B_4, B_1B_5:

$$\overrightarrow{B_1B_2}(0,2)\Rightarrow \text{лежит на границе I и II квадрантов, } k=+\infty,$$
$$\overrightarrow{B_1B_3}(3,-14)\Rightarrow \text{IV квадрант, } k=-14/3,$$
$$\overrightarrow{B_1B_4}(3,-13)\Rightarrow \text{IV квадрант, } k=-13/3,$$
$$\overrightarrow{B_1B_5}(4,-8)\Rightarrow \text{IV квадрант, } k=-2.$$

Значит, точки располагаются в следующем порядке: B_2, B_3, B_4, B_5, а после смещения – B_5, B_2, B_3, B_4. Расположение этих точек относительно касательной таково:

$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_5},\overrightarrow{B_1B_2}\right)=\operatorname{sgn}\begin{vmatrix}4 & -8\\ 0 & 2\end{vmatrix}=+1,$$
$$\operatorname{sgn}\angle\left(\overrightarrow{B_1B_5},\overrightarrow{B_1B_3}\right)=\operatorname{sgn}\begin{vmatrix}4 & -8\\ 3 & -14\end{vmatrix}=-1,$$

откуда сразу получается, что никакое ребро из инцидентных вершине B_1 не подлежит удалению.

У касательной B_1B_5 может сдвинуться либо левый конец в вершину B_2, либо правый в вершину B_6. Проверим все необходимые условия:

$$B_6 \in_{\triangle} B_1B_5B_2 - ?$$

$$\overrightarrow{B_1B_5}(4,-8), \overrightarrow{B_1B_2}(0,2), \overrightarrow{B_1B_6}(12,-2),$$

$$\alpha\overrightarrow{B_1B_5} + \beta\overrightarrow{B_1B_2} = \overrightarrow{B_1B_6}:$$

$$\left(\begin{array}{cc|c} 4 & 0 & 12 \\ -8 & 2 & -2 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & 3 \\ -8 & 2 & -2 \end{array}\right) \Rightarrow \alpha = 3 \Rightarrow \alpha + \beta > 1 \vee \beta < 0 \Rightarrow B_6 \notin_{\triangle} B_1B_5B_2,$$

$$B_2 \in_{\triangle} B_1B_5B_6 - ?$$

$$\alpha\overrightarrow{B_1B_5} + \beta\overrightarrow{B_1B_6} = \overrightarrow{B_1B_2}:$$

$$\left(\begin{array}{cc|c} 4 & 12 & 0 \\ -8 & -2 & 2 \end{array}\right) \sim \left(\begin{array}{cc|c} 4 & 12 & 0 \\ 0 & 22 & 2 \end{array}\right) \sim \left(\begin{array}{cc|c} 4 & 12 & 0 \\ 0 & 1 & 1/11 \end{array}\right) \sim \left(\begin{array}{cc|c} 4 & 0 & -12/11 \\ 0 & 1 & 1/11 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 0 & -3/11 \\ 0 & 1 & 1/11 \end{array}\right),$$

$$\alpha = -3/11, \beta = 1/11, \alpha < 0 \Rightarrow B_2 \notin_{\triangle} B_1B_5B_6,$$

$$_{\triangle}B_1B_5B_2, {}_{\triangle}B_5B_2B_6:$$

$$\overrightarrow{B_1B_5} \cdot \overrightarrow{B_1B_2} = -16 < 0,$$

$$\overrightarrow{B_6B_5}(-8,-6) \cdot \overrightarrow{B_6B_2}(-12,4) = 72 > 0,$$

$$\left|\overrightarrow{B_1B_5} \times \overrightarrow{B_1B_2}\right|\left(\overrightarrow{B_6B_5} \cdot \overrightarrow{B_6B_2}\right) + \left(\overrightarrow{B_1B_5} \cdot \overrightarrow{B_1B_2}\right)\left|\overrightarrow{B_6B_5} \times \overrightarrow{B_6B_2}\right| = 8 \cdot 72 - 16 \cdot 104 < 0.$$

Значит, новой касательной станет ребро B_1B_6. Список всех рёбер после этой итерации становится равен $T = \{B_1B_2, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7, B_3B_7, B_3B_4, B_1B_4, \underline{B_1B_5}\}$ (Рисунок 2.114).

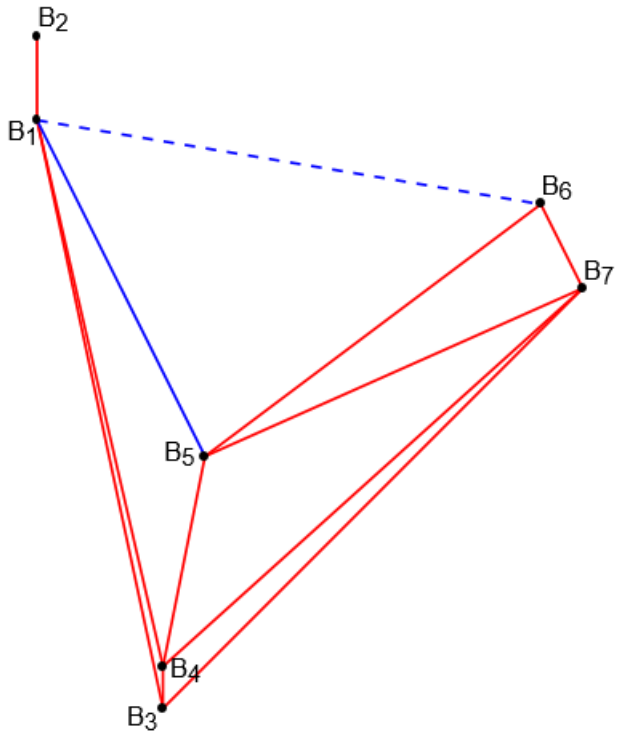


Рисунок 2.114 – Рёбра триангуляции после четвёртой итерации

На новой итерации после добавления ребра B_1B_6 получается следующий список рёбер, выходящих из правого конца касательной: B_6B_5, B_6B_7, B_6B_1. При их сортировке имеют место следующие вычисления:

$$\overrightarrow{B_6B_5}(-8,-6) \Rightarrow \text{III квадрант, } k = 3/4,$$
$$\overrightarrow{B_6B_7}(1,-2) \Rightarrow \text{IV квадрант, } k = -2,$$
$$\overrightarrow{B_6B_1}(-12,2) \Rightarrow \text{II квадрант, } k = -1/6.$$

Значит, получается последовательность точек B_7, B_5, B_1, а после смещения – B_1, B_7, B_5. После этого получаем, что первая же точка после B_1 из этого списка расположена по правую сторону от ребра B_1B_6:

$$\operatorname{sgn} \angle\left(\overrightarrow{B_1B_6}, \overrightarrow{B_1B_7}\right) = \operatorname{sgn} \begin{vmatrix} 12 & -2 \\ 13 & -4 \end{vmatrix} = -1.$$

Значит, у касательной будет изменена левая вершина. Слева, у вершины B_1, имеем рёбра B_1B_2, B_1B_3, B_1B_4, B_1B_5, B_1B_6, которые сортируются исходя из следующих вычислений:

$$\overrightarrow{B_1B_2}(0,2) \Rightarrow \text{лежит на границе I и II квадрантов, } k = +\infty,$$
$$\overrightarrow{B_1B_3}(3,-14) \Rightarrow \text{IV квадрант, } k = -14/3,$$
$$\overrightarrow{B_1B_4}(3,-13) \Rightarrow \text{IV квадрант, } k = -13/3,$$
$$\overrightarrow{B_1B_5}(4,-8) \Rightarrow \text{IV квадрант, } k = -2,$$
$$\overrightarrow{B_1B_6}(12,-2) \Rightarrow \text{IV квадрант, } k = -1/6.$$

То есть получается последовательность точек B_2, B_3, B_4, B_5, B_6, а после смещения $-B_6$, B_2, B_3, B_4, B_5. Положение этих точек относительно ребра B_1B_6:

$$\operatorname{sgn} \angle\left(\overrightarrow{B_1B_6}, \overrightarrow{B_1B_2}\right) = \operatorname{sgn} \begin{vmatrix} 12 & -2 \\ 0 & 2 \end{vmatrix} = +1,$$
$$\operatorname{sgn} \angle\left(\overrightarrow{B_1B_6}, \overrightarrow{B_1B_3}\right) = \operatorname{sgn} \begin{vmatrix} 12 & -2 \\ 3 & -14 \end{vmatrix} = -1,$$

т.е. ни одно ребро не удаляется в ходе этой итерации, так как только одно из них лежит слева от касательной. Итак, с учётом всего сказанного выше получаем, что новой касательной является ребро B_2B_6. Оно совпадает с ранее

вычисленной верхней касательной, поэтому после добавления этого ребра получим окончательную триангуляцию $T = \{B_1B_2, B_1B_3, B_6B_5, B_5B_4, B_4B_7, B_7B_6, B_5B_7, B_3B_7, B_3B_4, B_1B_4, B_1B_5, \underline{B_1B_6}, \underline{B_2B_6}\}$ (Рисунок 2.115).

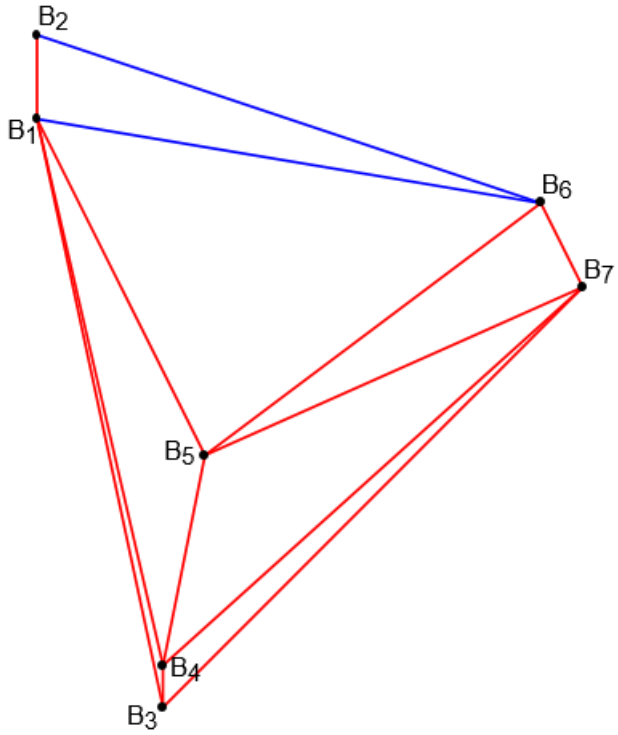


Рисунок 2.115 – Триангуляция Делоне, полученная в результате слияния триангуляций Делоне левого и правого подмножеств

Замечание. Для хранения графа текущей триангуляции (впрочем, едва ли можно назвать совокупность рёбер на рисунках 2.111 – 2.114 триангуляцией) при использовании данного алгоритма весьма подходящей является такая структура как **список смежности**, где каждой вершине графа ставится в соответствие список всех вершин, смежных с ней, причём вершины хранятся в отсортированном виде. Это несколько усложняет добавление нового ребра, так как в списках для каждого из концов нового ребра нужно найти место, куда следует вставить вершину другого конца, чтобы не нарушить сортировку уже имеющихся вершин.

2.6.3. Задачи

1. Построить триангуляцию Делоне для заданного множества точек $\{A_i\}_{i=1}^{n}$.

1.1. $A_1(1,-5)$, $A_2(4,1)$, $A_3(-5,1)$, $A_4(3,-2)$, $A_5(-4,2)$ (перейти к ответу),

1.2. $A_1(1,0)$, $A_2(-5,-4)$, $A_3(2,2)$, $A_4(5,-5)$, $A_5(-2,-5)$ (перейти к ответу),

1.3. $A_1(3,-4)$, $A_2(1,4)$, $A_3(-1,2)$, $A_4(3,2)$, $A_5(5,5)$ (перейти к ответу),

1.4. $A_1(3,5)$, $A_2(4,-5)$, $A_3(-2,-2)$, $A_4(-4,2)$, $A_5(-4,-1)$ (перейти к ответу),

1.5. $A_1(1,2)$, $A_2(3,-5)$, $A_3(3,2)$, $A_4(-5,1)$, $A_5(-1,-1)$ (перейти к ответу),

1.6. $A_1(-4,5)$, $A_2(-2,4)$, $A_3(3,1)$, $A_4(4,2)$, $A_5(1,-4)$ (перейти к ответу),

1.7. $A_1(-4,4)$, $A_2(-1,2)$, $A_3(-5,5)$, $A_4(5,5)$, $A_5(0,1)$ (перейти к ответу),

1.8. $A_1(0,4)$, $A_2(5,1)$, $A_3(0,1)$, $A_4(0,2)$, $A_5(-3,-2)$ (перейти к ответу),

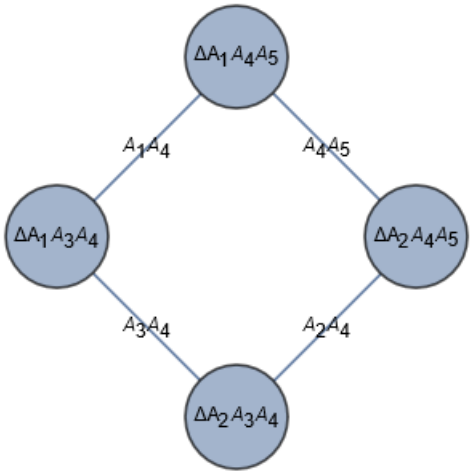
1.9. $A_1(7,-6)$, $A_2(-4,-8)$, $A_3(10,-1)$, $A_4(-5,1)$, $A_5(2,-4)$, $A_6(9,2)$, $A_7(8,7)$ (перейти к ответу),

1.10. $A_1(0,5)$, $A_2(-9,2)$, $A_3(-4,-1)$, $A_4(1,4)$, $A_5(1,-2)$, $A_6(8,6)$, $A_7(4,3)$ (перейти к ответу),

1.11. $A_1(5,-4)$, $A_2(-3,-6)$, $A_3(9,9)$, $A_4(7,10)$, $A_5(-4,-2)$, $A_6(-10,-1)$, $A_7(-5,-5)$ (перейти к ответу).

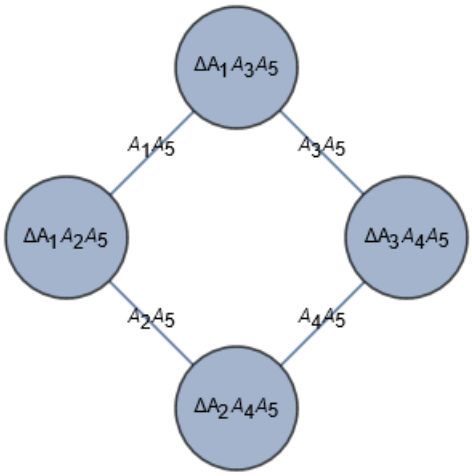
2. Достроить известную триангуляцию Делоне множества точек $\{A_i\}_{i=1}^n$, заданную в виде графа, после добавления точек $\{B_j\}_{j=1}^m$. Известна также выпуклая оболочка H_A точек $\{A_i\}_{i=1}^n$, заданная в виде последовательности её вершин, обеспечивающей её левосторонний обход.

2.1. $A_1(-2,-3)$, $A_2(2,3)$, $A_3(-2,0)$, $A_4(-1,-2)$, $A_5(4,0)$, $B_1(-4,-2)$, $B_2(-4,0)$, $H_A = A_1A_5A_2A_3A_1$ (перейти к ответу),



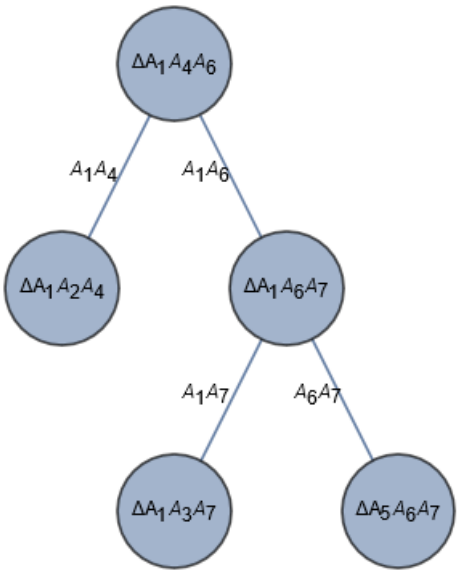
2.2. $A_1(-4,-3)$, $A_2(4,-1)$, $A_3(-2,4)$, $A_4(0,5)$, $A_5(-3,-5)$, $B_1(-5,-4)$, $B_2(1,0)$, $H_A = A_1A_5A_2A_4A_3A_1$ (перейти к ответу),

2.3. $A_1(1,5)$, $A_2(-5,4)$, $A_3(3,-4)$, $A_4(-4,2)$, $A_5(-1,2)$, $B_1(5,-4)$, $B_2(-5,5)$, $H_A = A_1A_2A_4A_3A_1$ (перейти к ответу),

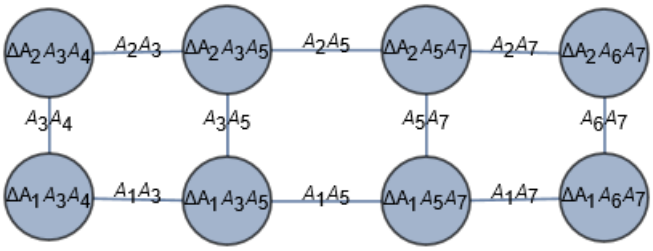


2.4. $A_1(5,-1)$, $A_2(5,2)$, $A_3(-4,-1)$, $A_4(5,4)$, $A_5(1,-4)$, $B_1(-5,-1)$, $B_2(0,3)$, $H_A = A_1A_2A_4A_3A_5A_1$ (перейти к ответу),

2.5. $A_1(-5,-4)$, $A_2(3,5)$, $A_3(-5,-5)$, $A_4(3,1)$, $A_5(0,-5)$, $A_6(1,-3)$, $A_7(-1,-5)$, $B_1(-1,-3)$, $H_A = A_1A_3A_7A_5A_6A_4A_2A_1$ (перейти к ответу),



2.6. $A_1(-2,4)$, $A_2(-2,-5)$, $A_3(-4,2)$, $A_4(-5,3)$, $A_5(-3,1)$, $A_6(4,-1)$, $A_7(-2,0)$, $B_1(-4,3)$, $H_A = A_1A_4A_2A_6A_1$ (перейти к ответу),



2.7. $A_1(4,5)$, $A_2(5,4)$, $A_3(3,2)$, $A_4(1,1)$, $A_5(-2,1)$, $A_6(-4,5)$, $A_7(-1,1)$, $B_1(2,-3)$, $H_A = A_1A_6A_5A_7A_4A_3A_2A_1$ (перейти к ответу).

3. Осуществить слияние триангуляций Делоне левого $\{A_i\}_{i=1}^{n}$ и правого $\{B_j\}_{j=1}^{m}$ множества точек при известных выпуклых оболочках H_A и H_B, которые заданы как последовательности вершин, обеспечивающие левосторонний обход. Обе триангуляции заданы в виде списков смежности, где каждой вершине C_i поставлен в соответствие **список смежных с ней вершин** C'_1, C'_2, … C'_j, **отсортированный** таким образом, что векторы $\overrightarrow{C_iC'_1}$, $\overrightarrow{C_iC'_2}$, …, $\overrightarrow{C_iC'_j}$ поворачиваются друг к другу **по часовой стрелке**.

3.1. $A_1(-9,-4)$, $A_2(-2,-8)$, $A_3(3,6)$, $A_4(4,-1)$, $B_1(5,-8)$, $B_2(8,-4)$, $B_3(9,8)$, $B_4(10,-2)$, $H_A = A_1A_2A_4A_3A_1$, $H_B = B_1B_4B_3B_1$ (перейти к ответу);

$A_1 : \{A_3, A_4, A_2\}$,
$A_2 : \{A_1, A_4\}$,
$A_3 : \{A_4, A_1\}$,
$A_4 : \{A_2, A_1, A_3\}$;

$B_1 : \{B_3, B_2, B_4\}$,
$B_2 : \{B_1, B_3, B_4\}$,
$B_3 : \{B_4, B_2, B_1\}$,
$B_4 : \{B_1, B_2, B_3\}$;

3.2. $A_1(-10,-5)$, $A_2(-5,-2)$, $A_3(-3,0)$, $A_4(-1,-6)$, $B_1(1,-4)$, $B_2(2,6)$, $B_3(6,7)$, $B_4(9,-6)$, $H_A = A_1A_4A_3A_1$, $H_B = B_1B_4B_3B_2B_1$ (перейти к ответу);

$A_1 : \{A_3, A_2, A_4\}$,
$A_2 : \{A_1, A_3, A_4\}$,
$A_3 : \{A_4, A_2, A_1\}$,
$A_4 : \{A_1, A_2, A_3\}$;

$B_1 : \{B_2, B_3, B_4\}$,
$B_2 : \{B_3, B_1\}$,
$B_3 : \{B_4, B_1, B_2\}$,
$B_4 : \{B_1, B_3\}$;

3.3. $A_1(-8,10)$, $A_2(-7,-5)$, $A_3(-6,10)$, $A_4(-4,-4)$, $A_5(-1,-9)$, $B_1(3,8)$, $B_2(6,-5)$, $B_3(6,5)$, $B_4(7,1)$, $B_5(10,-7)$, $H_A = A_1A_2A_5A_3A_1$, $H_B = B_1B_2B_5B_3B_1$ (перейти к ответу);

$A_1 : \{A_3, A_4, A_2\}$, $B_1 : \{B_3, B_4, B_2\}$,
$A_2 : \{A_1, A_4, A_5\}$, $B_2 : \{B_1, B_4, B_5\}$,
$A_3 : \{A_5, A_4, A_1\}$, $B_3 : \{B_5, B_4, B_1\}$,
$A_4 : \{A_1, A_3, A_5, A_2\}$, $B_4 : \{B_1, B_3, B_5, B_2\}$,
$A_5 : \{A_2, A_4, A_3\}$; $B_5 : \{B_2, B_4, B_3\}$.

2.7. Основные алгоритмы геометрической оптики

2.7.1. Моделирование световых лучей

Различные процессы и явления, происходящие в результате распространения световых лучей и их попадания на различные поверхности сцены, моделируются исходя из законов геометрической оптики. Во-первых, имеют место основные законы геометрической оптики, единые для различных оптических явлений и эффектов:

- Прямолинейность распространения света (**принцип Ферма**). По этому закону свет, испускаемый некоторым источником, можно задать как семейство лучей, имеющих единое начало (при ближнем источнике света) либо один и тот же направляющий вектор (при дальнем источнике света).
- Обратимость светового луча, т.е. при отражении светового луча вспять (например, в результате падения на зеркальную поверхность под прямым углом) свет пойдёт по той же траектории в обратном направлении. Этот принцип позволяет рассматривать направление наблюдения аналогично движению световых лучей и строить наблюдаемые результаты оптических эффектов (например, зеркальное отражение некоторого объекта от поверхности, наблюдаемое из конкретной точки).
- Независимость хода световых лучей, которая позволяет рассматривать различные световые лучи, порождаемые одним или разными источниками, независимо друг от друга при помощи средств параллельного программирования.

Во-вторых, имеют место специальные принципы, по которым вычисляются результаты отдельных эффектов и явлений, например отражение и преломление световых лучей.

Пример 7.1. Построить модель светового луча, который является отражением от плоской поверхности $\Pi: 3x + 2y - z = 3$ первичного светового луча, заданным источником $l(5,2,1)$ и направлением $\vec{v}(2,7,25)$.

Для начала стоит выяснить, будет ли вообще иметь место отражение заданного светового луча от заданной поверхности. Для этого необходимо найти точку их пересечения (при её наличии). Зададим луч в параметрическом виде:

$$\Delta: \begin{cases} x = 5 + 2t, \\ y = 2 + 7t, \\ z = 1 + 25t, \\ t \geq 0. \end{cases}$$

Координаты этого луча подставим в уравнение плоскости:

$$\Delta \cap \Pi: 3(5 + 2t) + 2(2 + 7t) - (1 + 25t) = 3,$$
$$-5t + 15 = 0,$$
$$t = 3.$$

Получили положительное значение параметра, т.е. луч Δ действительно пересекает поверхность П. Координаты точки пересечения равны $p(11, 23, 76)$.

Теперь остаётся лишь найти направление отражённого луча. Он моделируется по следующим принципам:

- отражённый луч лежит в одной плоскости с падающим лучом и вектором нормали поверхности, отложенный от точки падения падающего луча;
- угол падения равен углу отражения;
- проекции направляющих векторов падающего и отражённого лучей на ось, проведённую вдоль нормали, противонаправлены.

Исходя из этих принципов выводится формула (1.99). Одним из её свойств является то, что получается направляющий вектор отражённого луча той же длины, что и у первичного: $\left|\vec{v}_{\text{отр}}\right| = \left|\vec{v}\right|$. Однако в ситуациях, когда не нужно учитывать расстояние, пройденное световым лучом, можно упростить вычисления, домножив правую часть формулы (1.99) на $\vec{n}^2$:

$$\vec{v}_{\text{отр}} = \vec{n}^2\vec{v} - 2(\vec{v} \cdot \vec{n})\vec{n}. \qquad (2.29)$$

Для плоской отражающей поверхности вектор нормали не зависит от точки падения p и в данном примере равен $\vec{n}(3,2,-1)$. Подставляя его и

заданный вектор $\vec{v}$ в формулу (2.29), получим направляющий вектор отражённого луча:

$$\vec{v}_{\text{отр}} = (3^2 + 2^2 + (-1)^2)\vec{v} - 2(2\cdot 3 + 7\cdot 2 + 25\cdot(-1))\vec{n} = 14\vec{v} + 10\vec{n}.$$

Нетрудно видеть, что возможно сокращение на положительный коэффициент 2, что даст сонаправленный вектор $\vec{v}'_{\text{отр}} = \frac{1}{2}\vec{v}_{\text{отр}}$, также являющийся направляющим вектором искомого луча:

$$\vec{v}'_{\text{отр}}(7\cdot 2 + 5\cdot 3, 7\cdot 7 + 5\cdot 2, 7\cdot 25 + 5\cdot(-1)) = \vec{v}'_{\text{отр}}(29, 59, 170).$$

В итоге имеем, что искомый луч имеет начало в точке падения с координатами $p(11, 23, 76)$ и сонаправлен вектору с координатами $\vec{v}'_{\text{отр}}(29, 59, 170)$. Такой луч можно описать также системой параметрических уравнений:

$$\Delta_{\text{отр}}: \begin{cases} x = 11 + 29t, \\ y = 23 + 59t, \\ z = 76 + 170t, \\ t \geq 0. \end{cases}$$

Чертёж для этой задачи представлен на рисунке 2.116.

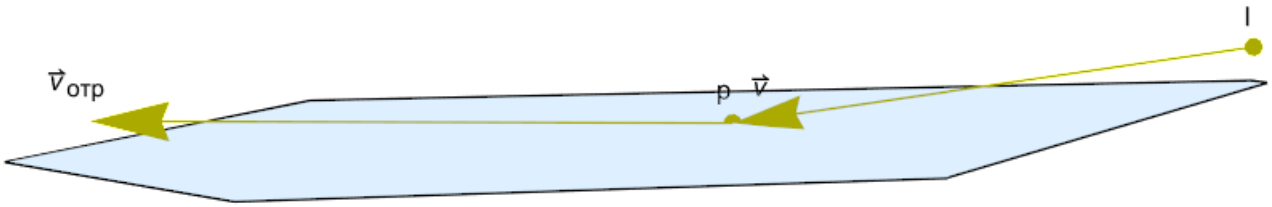


Рисунок 2.116 – Отражение светового луча от плоской отражающей поверхности

Пример 7.2. Для светового луча с началом в точке $l(-1, -10, 7)$, направленным вдоль вектора $\vec{v}(1, 5, -3)$, найти его отражение от поверхности $\Xi: -28x^2 + 3y^2 + 3z^2 = -25$ (двуполостный гиперболоид).

Найдём пересечение светового луча с поверхностью:

$$\Delta:\begin{cases} x=-1+t, \\ y=-10+5t, \\ z=7-3t, \\ t\ge 0, \end{cases}$$

$$\Delta\cap\Xi:-28(-1+t)^2+3(-10+5t)^2+3(7-3t)^2+25=0,$$

$$(-28+75+27)t^2+(56-300-126)t-28+300+147+25=0,$$

$$74t^2-370t+444=0,$$

$$t^2-5t+6=0,$$

$$t\in\{2,3\}.$$

Из полученных значений параметра t следует брать наименьший положительный – это параметр, соответствующий точке первого пересечения луча Δ с поверхностью Ξ. Если не оказалось ни одного положительного параметра, это, опять же, означает отсутствие пересечения лучом отражающей поверхности, т.е. отсутствие отражения. В данном примере имеем точку падения, которой отвечает значение параметра $t = 2$: $p(1, 0, 1)$.

Для криволинейной поверхности вектор нормали зависит от точки, откуда проводится этот вектор: $\vec{n}=\vec{n}(p)$. Для вычисления вектора нормали поверхности, заданной в виде неявной функции $F(x, y, z) = C$, нужно найти все частные производные функции F и подставить в них координаты точки падения:

$$\vec{n}(p)=\left(\frac{\partial F(p_x,p_y,p_z)}{\partial x},\frac{\partial F(p_x,p_y,p_z)}{\partial y},\frac{\partial F(p_x,p_y,p_z)}{\partial z}\right),$$

$$\frac{\partial F}{\partial x}=\left(-28x^2+3y^2+3z^2\right)_x'=-56x,\frac{\partial F}{\partial y}=6y,\frac{\partial F}{\partial z}=6z,$$

$$\vec{n}(p)=(-56x,6y,6z)\big|_{x=1,y=0,z=1}=(-56,0,6).$$

Положим вектор нормали, в два раза короче получившегося: $\vec{n}(-28,0,3)$. Теперь можно воспользоваться формулой (2.29):

$$\vec{n}^2=28^2+3^2=793,\vec{v}\cdot\vec{n}=1\cdot(-28)+5\cdot 0-3\cdot 3=-37$$

$$\vec{v}_{\text{отр}}=\vec{n}^2\vec{v}-2(\vec{v}\cdot\vec{n})\vec{n}=793\vec{v}+74\vec{n},$$

$$\vec{v}_{\text{отр}}(793\cdot 1+74\cdot(-28),793\cdot 5+74\cdot 0,793\cdot(-3)+74\cdot 3)=\vec{v}_{\text{отр}}(-1279,3965,-2157).$$

Таким образом, вторичный луч имеет начало в точке $p(1, 0, 1)$ и направление $\vec{v}_{\text{отр}}(-1279,3965,-2157)$ (Рисунок 2.117).

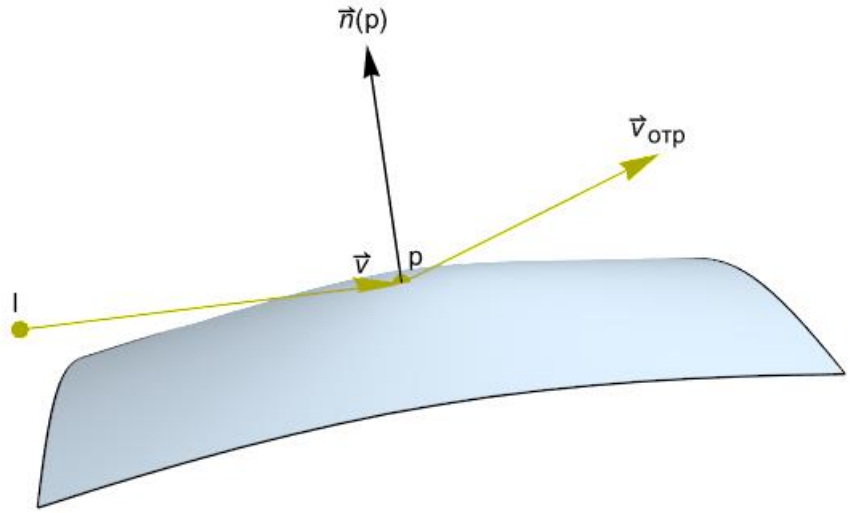


Рисунок 2.117 – Отражение светового луча
от криволинейной поверхности

Пример 7.3. Для светового луча с началом в точке $l(-1, -7, 14)$, направленным вдоль вектора $\vec{v}(1,2,-3)$, найти луч, преломлённый после прохождения через плоскую поверхность Π: $3x - 4z = 1$, если известен относительный показатель преломления $n_{прел} = 3/2$.

Преломление светового луча подчиняется следующим принципам:

- преломлённый луч лежит в одной плоскости с падающим лучом и вектором нормали, отложенным от точки падения;
- связь между углами падения α и преломления γ описывается законом Снеллиуса-Декарта (1.100).
- проекции направляющих векторов падающего и преломлённого лучей на ось, проведённую вдоль вектора нормали, сонаправлены.

Исходя из этих принципов выводится формула расчёта направляющего вектора преломлённого луча (1.101).

Снова ищем точку пересечения луча с поверхностью:

$$\Delta: \begin{cases} x = -1 + t, \\ y = -7 + 2t, \\ z = 14 - 3t, \\ t \geq 0, \end{cases}$$

$$\Delta \cap \Pi: 3(-1+t) - 4(14 - 3t) = 1,$$

$$15t - 60 = 0,$$

$$t = 4.$$

Значит, луч пересекается с поверхностью в точке $p(3, 1, 2)$. Вектор нормали равен $\vec{n}(3,0,-4)$. Перед нахождением направляющего вектора

преломлённого луча проверим условие наличия преломления светового луча (1.102):

$$\vec{v}^2 = 1^2 + 2^2 + (-3)^2 = 14,\ \vec{n}^2 = 3^2 + 0^2 + (-4)^2 = 25,$$

$$\vec{v} \times \vec{n} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & 2 & -3 \\ 3 & 0 & -4 \end{vmatrix} = -8\vec{i} - 5\vec{j} - 6\vec{k},\ \left|\vec{v} \times \vec{n}\right|^2 = 8^2 + 5^2 + 6^2 = 125,$$

$$\left|\vec{v}\right|^2 \left|\vec{n}\right|^2 - n_{\text{прел}}^2 \left|\vec{v} \times \vec{n}\right|^2 = 14 \cdot 25 - \frac{9}{4} \cdot 125 = \frac{275}{4} \geq 0.$$

Это неравенство означает, что действительно имеет место преломление. Найдём направляющий вектор по формуле (1.101):

$$\vec{v}_{\text{прел}} = n_{\text{прел}}\vec{v} + \frac{\text{sgn}(\vec{v} \cdot \vec{n})\sqrt{\left|\vec{v}\right|^2 \left|\vec{n}\right|^2 - n_{\text{прел}}^2 \left|\vec{v} \times \vec{n}\right|^2} - n_{\text{прел}}\vec{v} \cdot \vec{n}}{\vec{n}^2}\vec{n} =$$

$$= \left[\vec{v} \cdot \vec{n} = 1 \cdot 3 + 2 \cdot 0 - 3 \cdot (-4) = 15\right] = \frac{3}{2}\vec{v} + \frac{\sqrt{275/4} - 15 \cdot 3/2}{25}\vec{n} = \frac{3}{2}\vec{v} + \frac{\sqrt{11} - 9}{10}\vec{n},$$

$$\vec{v}_{\text{прел}}\left(\frac{3}{2} \cdot 1 + \frac{\sqrt{11} - 9}{10} \cdot 3, \frac{3}{2} \cdot 2, \frac{3}{2} \cdot (-3) + \frac{\sqrt{11} - 9}{10} \cdot (-4)\right) = \vec{v}_{\text{прел}}\left(\frac{3\sqrt{11} - 12}{10}, 3, \frac{-4\sqrt{11} - 9}{10}\right).$$

Значит, искомый луч задаётся началом $p(3, 1, 2)$ и направлением $\vec{v}_{\text{прел}}\left(\frac{3\sqrt{11} - 12}{10}, 3, \frac{-4\sqrt{11} - 9}{10}\right)$ (Рисунок 2.118).

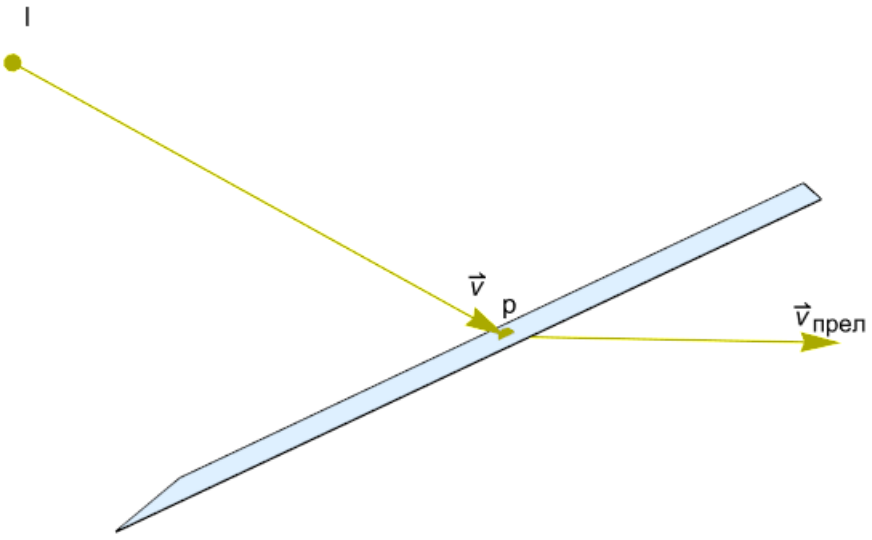


Рисунок 2.118 – Преломление светового луча

2.7.2. Лучевые методы моделирования оптических эффектов

При моделировании оптических эффектов, возникающих благодаря воздействию объектов сцены на траектории световых лучей, также используются основные законы оптики, в первую очередь законы прямолинейного распространения света и обратимости световых лучей. Оптические эффекты, например тень, зеркальное отражение и преломлённое изображение некоторого объекта, строятся на некоторой поверхности, на которой эти эффекты видны. Часто используются методы построения оптических эффектов для плоских поверхностей, ведь именно для них легко определить все необходимые формулы, в т.ч. условия видимости эффектов, без выполнения которых нет смысла в дальнейших вычислениях.

Пример 7.4. Построить тени для точек $a_i(x_i, y_i, z_i)$, $i = \overline{1,6}$, отбрасываемые ими от ближнего источника света $l(-2, -5, 4)$ на плоскую поверхность П: $2x - z - 2 = 0$, если имеет место дальнее наблюдение с вектором $\vec{s}(-2,-3,-1)$:

$$a_1(-3,-3,-3),\ a_2(-1,0,5),\ a_3(4,-2,-3),\ a_4(3,0,4),\ a_5(-3,1,1),\ a_6(-5,-2,3).$$

В случае плоской подложки можно (и нужно) вначале проверить условие видимости теней. Это можно выполнить по следующей формуле:

$$\operatorname{sgn}(N^T(h_l O - L)) = -\operatorname{sgn}(N^T(S - h_s O)), \qquad (2.30)$$

где N – координатный столбец нормального вектора подложки, O – координатный столбец произвольной точки $o \in \Pi$, L – координатный столбец источника света (точки l, если он является ближним, либо вектора $\vec{l}$, если он дальний), S – координатный столбец наблюдателя (точки s либо вектора $\vec{s}$) $h_l, h_s \in \{0,1\}$ – индикаторы близости источника света и наблюдателя соответственно. Если условие (2.30) нарушено, то это означает, что источник света и наблюдатель находятся по разные стороны от подложки, и заданный источник света не порождает видимых для этого наблюдателя теней.

Проверим условие (2.30) для данного примера:

$$L = (-2,-5,4)^T, h_l = 1,$$
$$S = (-2,-3,-1)^T, h_s = 0,$$
$$N = (2,0,-1)^T, O = (1,0,0)^T,$$
$$N^T(h_l O - L) = 2\cdot(1+2) + 0\cdot(0+5) - 1\cdot(0-4) = 6 + 4 > 0,$$
$$N^T(S - h_s O) = 2\cdot(-2) + 0\cdot(-3) - 1\cdot(-1) = -4 + 1 < 0.$$

Знаки различны, что означает, что при отбрасывании некоторым объектом тени на плоскость П она является видимой для заданного наблюдателя.

Далее происходит вычисление теней для всех точек. Для некоторой точки p с координатным столбцом P тенью является точка q, равная пересечению луча $Q = Q(P)$, задаваемого формулой (1.103), с плоскостью П. Это эквивалентно равенству нулю скалярного произведения $\vec{n} \cdot \overrightarrow{oq}$, которое в матричном виде записывается следующим образом:

$$N^T(Q-O)=0,$$
$$N^T(P+t(h_lP-L)-O)=0,$$
$$N^T(P-O)+tN^T(h_lP-L)=0,$$
$$t=-\frac{N^T(P-O)}{N^T(h_lP-L)}. \qquad (2.31)$$

Для пересечения подложки лучом (1.103) необходимо выполнение неравенства $t > 0$. Значит, при построении тени от точки p на плоскую подложку можно проделать следующую последовательность действий:

- Вычисляются выражения $N^T(P-O)$ и $N^T(h_lP-L)$. Для наличия тени необходимо, чтобы они были отличны от нуля и имели разные знаки.
- Если эти условия соблюдены, то вычисляется параметр t по формуле (2.31).
- Полученное значение подставляется в формулу (1.103).

Для упрощения вычислений заметим, что $N^T(P-O)$ – не что иное, как результат подстановки координат точки p в левую часть уравнения для П. Точно так же число $N^T(h_lP-L)$ при ближнем освещении ($h_l = 1$) может быть вычислено путём подстановки координат точек p и l в левую часть уравнения подложки и последующего вычитания этих двух чисел.

Итак, вычисляем тени для всех точек A_i:

$$L: 2\cdot(-2)-4-2=-10;$$
$$a_1: N^T(P-O)=2x_1-z_1-2=-5<0,\ N^T(h_lP-L)=-5+10=5>0,$$
$$t_1=-\frac{-5}{5}=1\Rightarrow Q_1=A_1+t_1(A_1-L)=2A_1-L=(-4,-1,-10)^T;$$
$$a_2: 2x_2-z_2-2=-9<0,-9+10=1>0,$$
$$t_2=-\frac{-9}{1}=9, Q_2=A_2+t_2(A_2-L)=10A_2-9L=(8,45,14)^T;$$
$$a_3: 2x_3-z_3-2=9>0, 9+10=19>0\Rightarrow t_3<0;$$
$$a_4: 2x_4-z_4-2=0\Rightarrow a_4\in\Pi;$$

$$a_5 : 2x_5 - z_5 - 2 = -9 < 0, -9 + 10 = 1 > 0,$$

$$t_5 = -\frac{-9}{1} = 9, Q_5 = A_5 + t_5(A_5 - L) = 10A_5 - 9L = (-12, 55, -26)^T;$$

$$a_6 : 2x_6 - z_6 - 2 = -15 < 0, -15 + 10 = -5 < 0 \Rightarrow t_6 < 0.$$

Таким образом, тень отбрасывают только точки a_1, a_2 и a_5. На рисунке 2.119 они обозначены тёмно-зелёным цветом, светло-зелёным – их тени, красным – все остальные исходные точки (), оливковым – положение источника света.

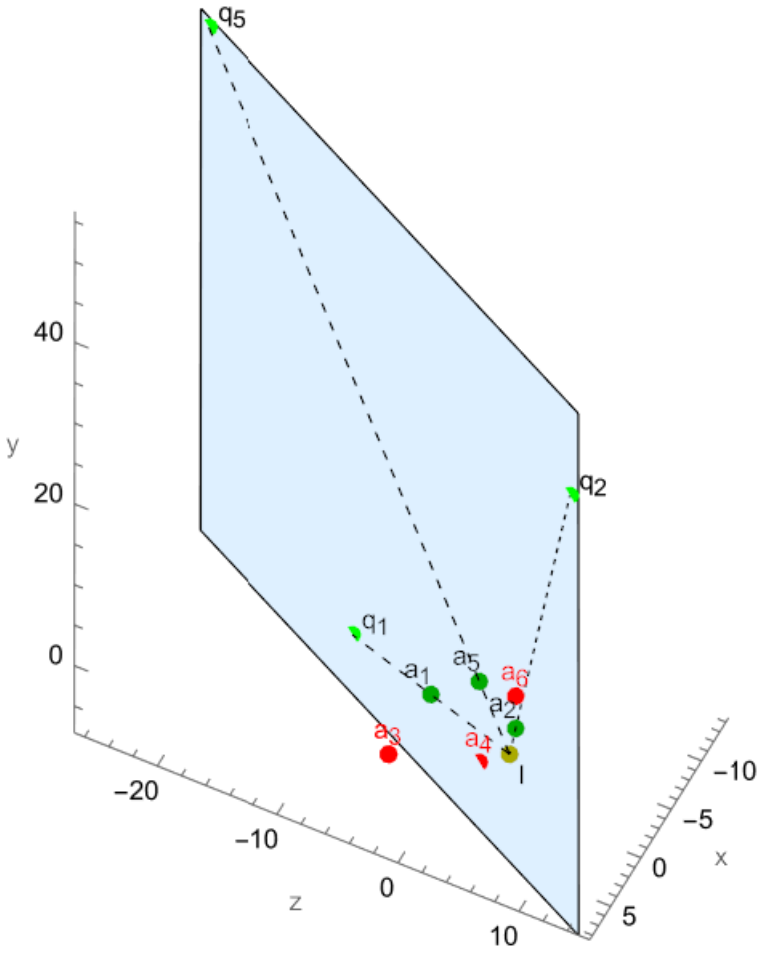


Рисунок 2.119 – Отбрасывание теней точек на плоскую подложку

Пример 7.5. Определить зеркальные отражения точки $p(1, -1, -4)$ на плоской отражающей поверхности П: $x - y + 3z - 4 = 0$, наблюдаемые разными наблюдателями:

$$\vec{s}_1(0,0,2),\ \vec{s}_2(-2,1,-1),\ s_3(2,-3,-3).$$

Первоначально следует определить видимость отражения точки для разных наблюдателей. Видимым отражение является тогда и только тогда,

когда по формуле (1.106) получается положительный коэффициент λ, что эквивалентно равным знакам чисел $N^T(P-O)$ и $N^T(S-h_sO)$, где N и O – координатные столбцы нормального вектора и некоторой точки плоскости П, P – координатный столбец точки p, S и h_s – координатный столбец и индикатор близости конкретного наблюдателя:

$$N=(1,-1,3)^T,\ O=(1,0,1)^T,\ P=(1,-1,-4)^T \Rightarrow N^T(P-O)=$$
$$=(x-y+3z-4)\big|_{(x,y,z)=p}=1+1-12-4=-14<0;$$
$$\vec{s}_1: h_{s_1}=0, S_1=(0,0,2)^T, N^T(S_1-h_{s_1}O)=N^TS_1=1\cdot 0-1\cdot 0+3\cdot 2=6>0\Rightarrow \lambda_1<0;$$
$$\vec{s}_2: h_{s_2}=0, S_2=(-2,1,-1)^T, N^T(S_2-h_{s_2}O)=N^TS_2=$$
$$=1\cdot(-2)-1\cdot 1+3\cdot(-1)=-6<0, \lambda_2=\frac{-14}{-6}=\frac{7}{3};$$
$$s_3: h_{s_3}=1, S_3=(2,-3,-3)^T, N^T(S_3-h_{s_3}O)=(x-y+3z-4)\big|_{(x,y,z)=s_3}=$$
$$=2+3-9-4=-8<0, \lambda_3=\frac{-14}{-8}=\frac{7}{4}.$$

Итак, наблюдатель $\vec{s}_1$ не видит никакого отражения точки p, а для $\vec{s}_2$ и s_3 получили значения коэффициентов $\lambda_2=7/3$ и $\lambda_3=7/4$. Подставим их в формулу (1.107) для вычисления искомых отражений:

$$Q_2=\frac{1}{h_{s_2}\lambda_2+1}\left(P+\lambda_2S_2+2\frac{N^T(O-P)}{N^TN}N\right)=(1,-1,-4)^T+$$
$$+\frac{7}{3}(-2,1,-1)^T+2\cdot\frac{14}{11}(1,-1,3)^T=\left(-\frac{37}{33},-\frac{40}{33},\frac{43}{33}\right)^T;$$
$$Q_3=\frac{1}{h_{s_3}\lambda_3+1}\left(P+\lambda_3S_3+2\frac{N^T(O-P)}{N^TN}N\right)=\frac{1}{7/4+1}\Big((1,-1,-4)^T+$$
$$+\frac{7}{4}(2,-3,-3)^T+2\cdot\frac{14}{11}(1,-1,3)^T\Big)=\left(\frac{310}{121},-\frac{387}{121},-\frac{71}{121}\right)^T.$$

На рисунке 2.120 обозначены исходная точка p и наблюдатели $\vec{s}_2$ и s_3, а также видимые им отражения q_2 и q_3 с вычисленными выше координатными столбцами Q_2 и Q_3.

Пример 7.6. Определить преломлённое изображение точки $p(-3,-2,2)$ на плоской поверхности П: $x+y+z+6=0$, наблюдаемое дальним наблюдателем $\vec{s}(1,-1,-2)$, если полуплоскость, в которой находится точка p, обладает

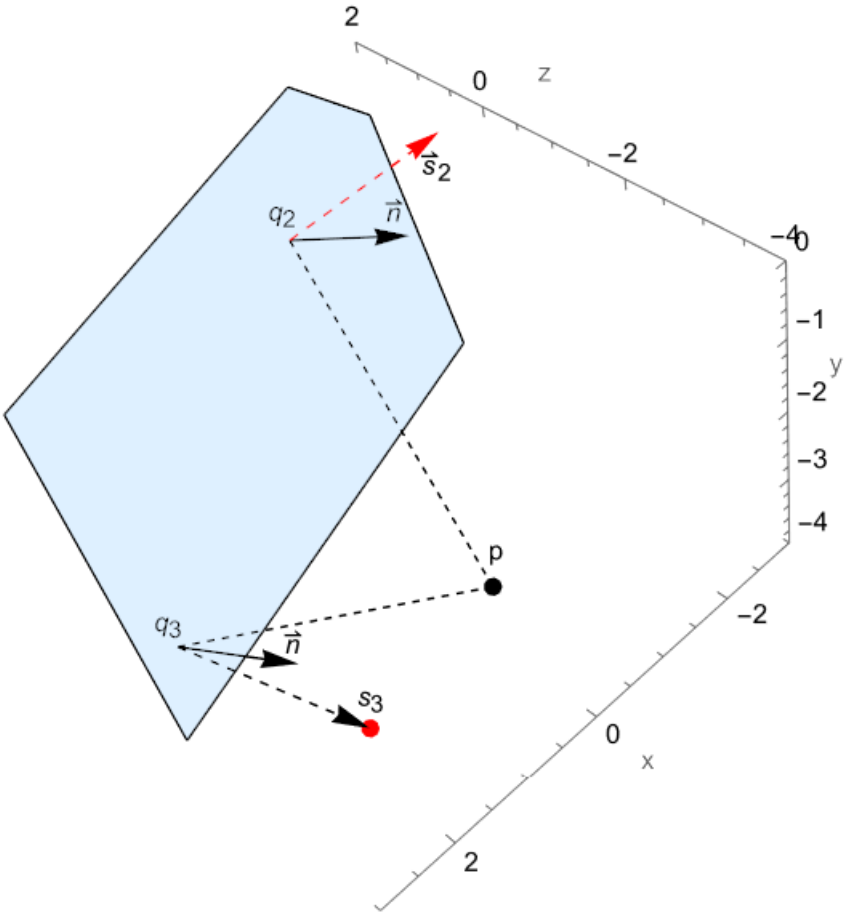


Рисунок 2.120 – Построение зеркальных отражений точки
для разных наблюдателей

абсолютным показателем преломления, равным $n_1 = 8$, а другая полуплоскость – $n_2 = 9$.

Снова вначале необходимо определить видимость оптического эффекта. Для наличия видимого преломления изображения точки необходимо расположение точки и наблюдателя по разные стороны от поверхности, что эквивалентно разным знакам выражений $N^T(P-O)$ и $N^T(S-h_sO)$:

$$N = (1,1,1)^T, O = (-2,-2,-2)^T, P = (-3,-2,2)^T, h_s = 0, S = (1,-1,-2)^T,$$

$$N^T(P-O) = (x+y+z+6)\big|_{(x,y,z)=p} = 3 > 0,$$

$$N^T(S-h_sO) = N^TS = 1\cdot 1+1\cdot(-1)+1\cdot(-2) = -2 < 0.$$

Также предварительно вычисляется знак подкоренного выражения в формуле (1.109):

$$n'_{\text{прел}} = n_2/n_1 = 9/8, \vec{s}\times\vec{n} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & -1 & -2 \\ 1 & 1 & 1 \end{vmatrix} = \vec{i} - 3\vec{j} + 2\vec{k},$$

$$|\vec{s}|^2 |\vec{n}|^2 - n'^2_{\text{прел}} |\vec{s}\times\vec{n}|^2 = (1^2+1^2+2^2)(1^2+1^2+1^2) -$$

$$-\frac{81}{64}(1^2+3^2+2^2) = 18 - \frac{81\cdot 14}{64} = \frac{9}{32} > 0.$$

Таким образом, действительно имеет место преломлённое изображение точки p, видимое дальнему наблюдателю $\vec{s}$. Оно вычисляется по формулам (1.109) и (1.110):

$$\lambda = \frac{-\operatorname{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2}}{\overrightarrow{op}\cdot\vec{n}} = \frac{+1\cdot\sqrt{9/32}}{3} = \frac{1}{4\sqrt{2}},$$

$$q = p + \frac{1}{\lambda}\left(n'_{\text{прел}}\vec{s} + \frac{\operatorname{sgn}(\vec{s}\cdot\vec{n})\sqrt{|\vec{s}|^2|\vec{n}|^2 - n'^2_{\text{прел}}|\vec{s}\times\vec{n}|^2} - n'_{\text{прел}}\vec{s}\cdot\vec{n}}{|\vec{n}|^2}\vec{n} \right) =$$

$$= p + 4\sqrt{2}\left(\frac{9}{8}\vec{s} + \frac{-1\cdot\sqrt{9/32} - 9/8\cdot(-2)}{3}\vec{n} \right) = p + \frac{9\sqrt{2}}{2}\vec{s} + \left(3\sqrt{2}-1\right)\vec{n},$$

$$q\left(-3 + \frac{9\sqrt{2}}{2}\cdot 1 + 3\sqrt{2} - 1, -2 + \frac{9\sqrt{2}}{2}\cdot(-1) + 3\sqrt{2} - 1, \right.$$

$$\left. 2 + \frac{9\sqrt{2}}{2}\cdot(-2) + 3\sqrt{2} - 1 \right) = q\left(\frac{15\sqrt{2}}{2} - 4, -\frac{3\sqrt{2}}{2} - 3, -6\sqrt{2} + 1 \right).$$

Чертёж для этой задачи представлен на рисунке 2.121.

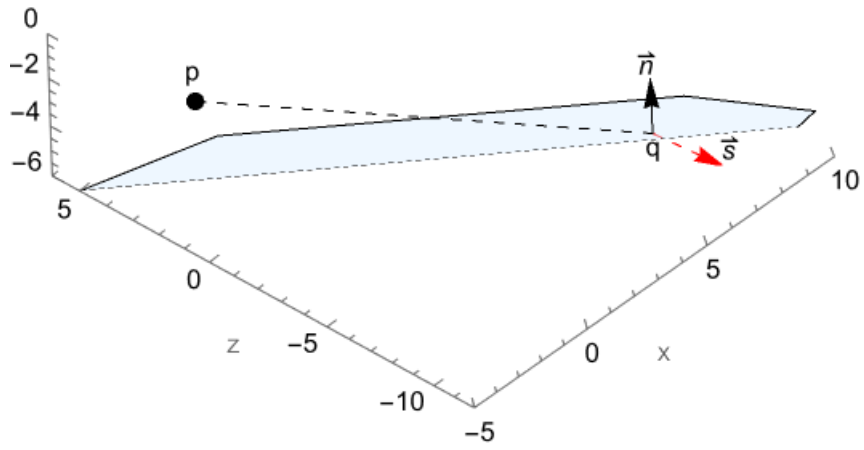


Рисунок 2.121 – Построение преломлённого изображения точки для дальнего наблюдателя

Пример 7.7. Определить преломлённое изображение точки $p(-4,-5,0)$ на плоской поверхности П: $2y - z + 7 = 0$, наблюдаемое ближним наблюдателем $s(2,-1,4)$, если известны абсолютные показатели преломления обеих сред: $n_1 = 6,\ n_2 = \sqrt{29}$.

Проверка условия видимости преломлённого изображения точки p даёт следующие результаты:

$$N = (0,2,-1)^T, O = (0,-3,1)^T, P = (-4,-5,0)^T, h_s = 1, S = (2,-1,4)^T,$$
$$N^T(P-O) = (2y - z + 7)\big|_{(x,y,z)=p} = -3 < 0,$$
$$N^T(S - h_s O) = N^T(S-O) = (2y - z + 7)\big|_{(x,y,z)=s} = 1 > 0.$$

Это означает, что точки p и s расположены по разные стороны от преломляющей поверхности, а следовательно, действительно имеет место видимое преломлённое изображение.

Вычислим ортогональные проекции точек p и s на плоскость П по формулам (1.112) и расстояния (1.113):

$$p_\perp = p - \frac{\overrightarrow{op}\cdot\vec{n}}{\vec{n}^2}\vec{n} = p + \frac{3}{5}\vec{n},\ p_\perp\left(-4+\frac{3}{5}\cdot 0, -5+\frac{3}{5}\cdot 2, 0+\frac{3}{5}\cdot(-1)\right) = p_\perp\left(-4,-\frac{19}{5},-\frac{3}{5}\right),$$
$$s_\perp = s - \frac{\overrightarrow{os}\cdot\vec{n}}{\vec{n}^2}\vec{n} = s - \frac{1}{5}\vec{n}, s_\perp\left(2-\frac{1}{5}\cdot 0, -1-\frac{1}{5}\cdot 2, 4-\frac{1}{5}\cdot(-1)\right) = s_\perp\left(2,-\frac{7}{5},\frac{21}{5}\right),$$
$$P = \frac{\left|\overrightarrow{op}\cdot\vec{n}\right|}{\left|\vec{n}\right|} = \frac{3}{\sqrt{5}}, S = \frac{\left|\overrightarrow{os}\cdot\vec{n}\right|}{\left|\vec{n}\right|} = \frac{1}{\sqrt{5}}, D = \left|\overrightarrow{ps} - \frac{\overrightarrow{ps}\cdot\vec{n}}{\vec{n}^2}\vec{n}\right| = \left|\overrightarrow{ps} - \frac{4}{5}\vec{n}\right| = \left|\left(2+4-\frac{4}{5}\cdot 0,\right.\right.$$
$$\left.\left.-1+5-\frac{4}{5}\cdot 2, 4-\frac{4}{5}\cdot(-1)\right)\right| = \left|\left(6,\frac{12}{5},\frac{24}{5}\right)\right| = \frac{6}{5}\left|(5,2,4)\right| = \frac{18\sqrt{5}}{5}.$$

Теперь можно составить уравнение:

$$(D - Pt)^2(n_2^2 + (n_2^2 - n_1^2)t^2) - S^2 n_1^2 t^2 = 0,$$
$$\left(\frac{18\sqrt{5}}{5} - \frac{3}{\sqrt{5}}t\right)^2 (29 + (29-36)t^2) - \frac{1}{5}\cdot 36t^2 = 0,$$
$$\frac{9}{5}(6-t)^2(29-7t^2) - \frac{36}{5}t^2 = 0,$$
$$(36 - 12t + t^2)(29 - 7t^2) - 4t^2 = 0,$$
$$-7t^4 + 84t^3 - 223t^2 - 348t + 1044 = 0.$$

Единственный корень этого уравнения $t \in [0, D/P] = [0,6]$ можно найти, перебирая делители свободного члена: $t = 2$. Осталось только подставить это значение в формулу (1.115):

$$q = p_\perp + \frac{P}{D} t \overrightarrow{p_\perp s_\perp} = p_\perp + \frac{3 \cdot 5}{\sqrt{5} \cdot 18\sqrt{5}} \cdot 2 \overrightarrow{p_\perp s_\perp} = p_\perp + \frac{1}{3} \overrightarrow{p_\perp s_\perp},$$

$$q\left(-4 + \frac{1}{3} \cdot (2+4), -\frac{19}{5} + \frac{1}{3} \cdot \left(-\frac{7}{5} + \frac{19}{5}\right), -\frac{3}{5} + \frac{1}{3} \cdot \left(\frac{21}{5} + \frac{3}{5}\right)\right) = q(-2, -3{,}1).$$

Чертёж для задачи приведён на рисунке 2.122.

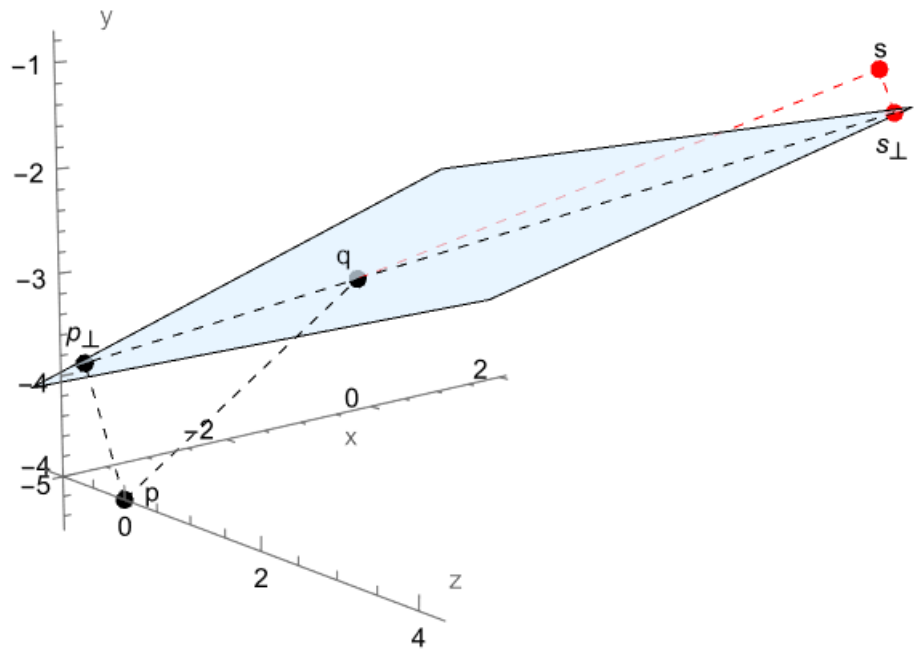


Рисунок 2.122 – Построение преломлённого изображения точки для ближнего наблюдателя

2.7.3. Задачи

1. Для первичного луча с заданным началом l и направляющим вектором $\vec{v}$ построить луч, отражённый от поверхности П (перейти к ответам).

1.1. $l(-8, -1, 1)$, $\vec{v}(2,1,0)$, П: $x + y + 3z - 6 = 0$,

1.2. $l(-2, -2, -2)$, $\vec{v}(1,1,1)$, П: $x + 4y - 3z = 0$,

1.3. $l(4, 4, 0)$, $\vec{v}(1,-1,-1)$, П: $-4x + z + 21 = 0$,

1.4. $l(5, -4, -4)$, $\vec{v}(-5,-5,1)$, П: $x - 4y + z - 5 = 0$,

1.5. $l(2, -5, -2)$, $\vec{v}(1,-3,4)$, П: $3x + 5y - z - 8 = 0$,

1.6. $l(2, 3, -3)$, $\vec{v}(1,-1,2)$, П: $2x + 3y - 2z - 4 = 0$,

1.7. $l(4, -1, 6)$, $\vec{v}(-1,2,-1)$, $\Pi: (2x-3)^2 + 2(y-3)^2 + 4z - 17 = 0$,

1.8. $l(0, 1, 3)$, $\vec{v}(3,2,-3)$, $\Pi: x^2 + (y-5)^2 + (z+5)^2 - 38 = 0$,

1.9. $l(1, -2, 0)$, $\vec{v}(1,0,1)$, $\Pi: z - 2^{x+y} = 0$.

2. Для первичного луча с заданным началом l и направляющим вектором $\vec{v}$ построить луч, преломлённый через поверхность П, если известен относительный показатель преломления $n_{\text{прел}}$ (перейти к ответам).

2.1. $l(0, 3, -4)$, $\vec{v}(-1,-2,0)$, П: $-x + 3y - z - 12 = 0$, $n_{\text{прел}} = \sqrt{3/2}$,

2.2. $l(4, -1, -1)$, $\vec{v}(-1,1,2)$, П: $x - y - 2z - 4 = 0$, $n_{\text{прел}} = 2$,

2.3. $l(-5, 4, 0)$, $\vec{v}(3,-3,1)$, П: $2x - 3y + z + 5 = 0$, $n_{\text{прел}} = \sqrt{7}$,

2.4. $l(-5, 4, 5)$, $\vec{v}(1,1,0)$, П: $x + y + z - 7 = 0$, $n_{\text{прел}} = \sqrt{5/2}$,

2.5. $l(-2, -2, -2)$, $\vec{v}(1,1,1)$, П: $x + 4y - 3z = 0$, $n_{\text{прел}} = 1/\sqrt{2}$,

2.6. $l(-1, -3, 5)$, $\vec{v}(-1,1,0)$, П: $-x + y - z - 4 = 0$, $n_{\text{прел}} = 1/\sqrt{2}$.

3. Построить видимые для заданного наблюдателя (дальнего $\vec{s}$ либо ближнего s) тени, отбрасываемые точками a_i при освещении заданным источником света (дальним $\vec{l}$ либо ближним l) на плоскую подложку П (перейти к ответам).

3.1. $a_1(-4,-3,-2)$, $a_2(4,1,-1)$, $a_3(4,5,5)$, $a_4(5,0,3)$, $a_5(-3,4,-5)$, $a_6(3,1,-4)$, $\vec{s}(-5,0,-1)$, $l(4, -4, -4)$, П: $5x + 5y - 3z - 9 = 0$,

3.2. $a_1(-3,3,-2)$, $a_2(0,-2,2)$, $a_3(2,2,3)$, $a_4(2,1,-3)$, $a_5(-3,-2,-1)$, $a_6(-1,-3,-3)$, $\vec{s}(1,-3,3)$, $l(-1, 1, -2)$, П: $2x + y + 3z + 18 = 0$,

3.3. $a_1(3,1,3)$, $a_2(1,-3,1)$, $a_3(3,3,0)$, $a_4(-1,1,-1)$, $a_5(0,-2,-2)$, $a_6(-1,1,1)$, $\vec{s}(-1,-1,2)$, $l(0, -1, -1)$, П: $x - 3y - 2z - 14 = 0$,

3.4. $a_1(-3,3,4)$, $a_2(0,0,3)$, $a_3(3,-4,2)$, $a_4(4,3,-4)$, $a_5(5,1,5)$, $a_6(0,-3,3)$, $\vec{s}(1,4,-2)$, $\vec{l}(0,2,-1)$, П: $-3x + 2y + 2z + 15 = 0$,

3.5. $a_1(2,1,0)$, $a_2(-2,1,2)$, $a_3(-3,3,-1)$, $a_4(-3,-3,-2)$, $a_5(1,1,1)$, $a_6(3,3,-2)$, $s(3, 0, 5)$, $\vec{l}(2,-3,0)$, П: $-2x + 2y + 3z - 12 = 0$,

3.6. $a_1(-2,3,3)$, $a_2(-1,-3,-2)$, $a_3(-3,-3,0)$, $a_4(-3,0,0)$, $a_5(0,0,1)$, $a_6(0,-1,-2)$, $s(-3, -5, -5)$, $l(3, -3, -1)$, П: $x - 3y - 2z - 9 = 0$.

4. Построить видимые для заданного наблюдателя (дальнего $\vec{s}$ либо ближнего s) зеркальные отражения точек a_i от плоской поверхности П (перейти к ответам).

4.1. $a_1(2,-2,-3)$, $a_2(0,-3,-3)$, $a_3(-3,-2,3)$, $s(2, 0, -3)$, П: $-3y + z - 7 = 0$,

4.2. $a_1(2,3,0)$, $a_2(0,3,0)$, $a_3(0,-1,2)$, $\vec{s}(1,0,-1)$, П: $-x - y + z + 1 = 0$,

4.3. $a_1(2,-3,-3)$, $a_2(0,1,-3)$, $a_3(2,-2,1)$, $\vec{s}(3,2,-2)$, П: $-x + 2y - 3z - 9 = 0$,

4.4. $a_1(2,0,2)$, $a_2(-1,2,1)$, $a_3(1,-1,3)$, $s(1, -3, 1)$, П: $-x + y - z = 0$,

4.5. $a_1(-2,0,-1)$, $a_2(1,-1,-2)$, $a_3(-1,3,2)$, $s(1, -1, 2)$, П: $-3x + 3y - z = 0$,

4.6. $a_1(1,2,-3)$, $a_2(-3,-2,-1)$, $a_3(-2,-2,3)$, $\vec{s}(0,3,2)$, П: $-x + 2y - z + 8 = 0$.

5. Построить видимые для заданного наблюдателя (дальнего $\vec{s}$ либо ближнего s) преломлённые изображения точек a_i на плоской поверхности П, если также заданы абсолютные показатели преломления n_1 (в полуплоскости, для точек которой существуют видимые изображения) и n_2 (в полуплоскости, где расположен наблюдатель) (перейти к ответам).

5.1. $a_1(0,1,0)$, $a_2(-3,-3,-1)$, $a_3(3,1,0)$, $\vec{s}(-3,1,0)$, П: $x = -4$, $n_1 = 1$, $n_2 = \sqrt{7}$,

5.2. $a_1(1,-2,3)$, $a_2(3,-3,-3)$, $a_3(2,-2,-3)$, $\vec{s}(1,-2,3)$, П: $-3y - 2z + 12 = 0$, $n_1 = 1$, $n_2 = 2$,

5.3. $a_1(-1,3,3)$, $a_2(2,0,-3)$, $a_3(1,-2,-1)$, $\vec{s}(0,-1,1)$, П: $2x + 3y + z + 16 = 0$, $n_1 = \sqrt{2}$, $n_2 = 1$,

5.4. $a_1(0,1,3)$, $a_2(3,3,1)$, $a_3(1,-3,-1)$, $\vec{s}(1,-1,-1)$, П: $z = 0$, $n_1 = \sqrt{8}$, $n_2 = \sqrt{3}$,

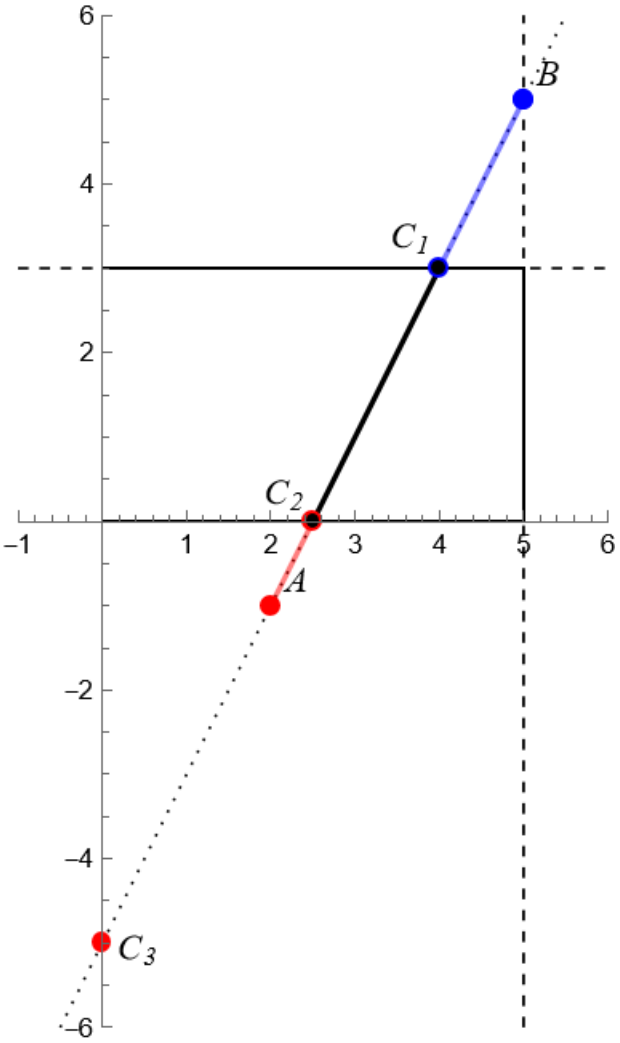
5.5. $a_1(2,2,3)$, $a_2(-1,1,3)$, $a_3(2,-2,0)$, $s(0, 0, 3)$, П: $x + 2y + 2z - 9 = 0$, $n_1 = 3\sqrt{5}$, $n_2 = \sqrt{13}$,

5.6. $a_1(1,-1,-2)$, $a_2(-2,-2,-3)$, $a_3(-9,-2,10)$, $s(-3, -2, -2)$, П: $3x + 4z - 4 = 0$, $n_1 = \sqrt{13}$, $n_2 = \sqrt{37}$.

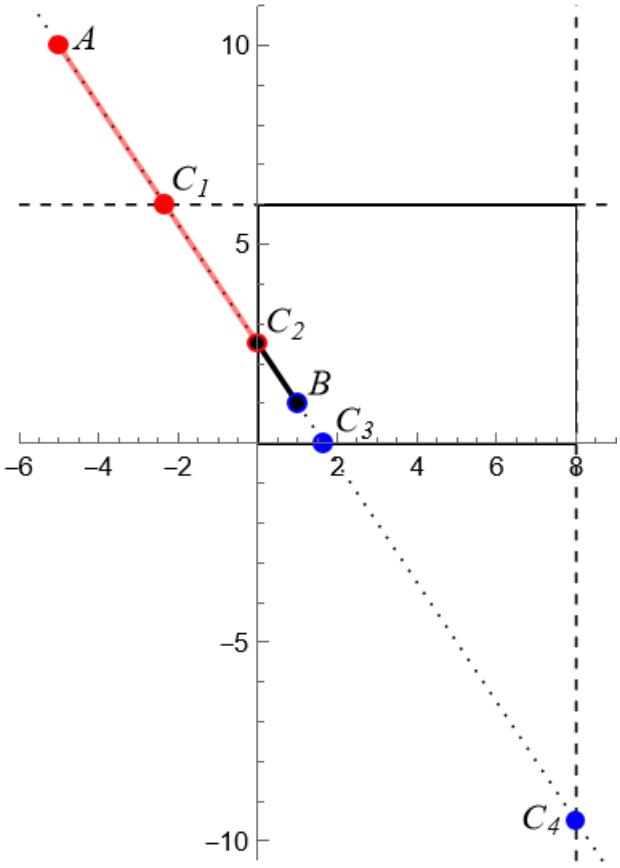
2.8. Ответы

Ответы к задачам подраздела 2.1.

1.1. (Перейти к условию).
Отрезок C_1C_2 с концами в точках $C_1(4,3)$ и $C_2\left(\frac{5}{2},0\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_3(0,-5)$ и $B(5,5)$. Параметры t, соответствующие этим точкам, равны $2/3$, $1/6$, $-2/3$, 1.

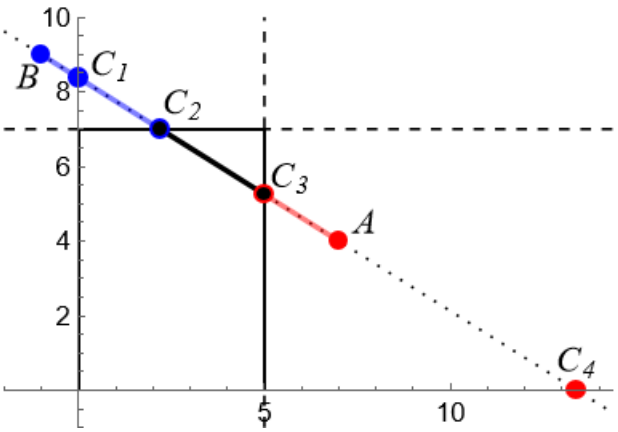


1.2. (Перейти к условию).
Отрезок BC_2 с концами в точках $B(1,1)$ и $C_2\left(0,\frac{5}{2}\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_1\left(-\frac{7}{3},6\right)$, $C_3\left(\frac{5}{3},0\right)$ и $C_4\left(8,-\frac{19}{2}\right)$. Параметры t, соответствующие этим точкам, равны 1, $5/6$, $4/9$, $10/9$, $13/6$.



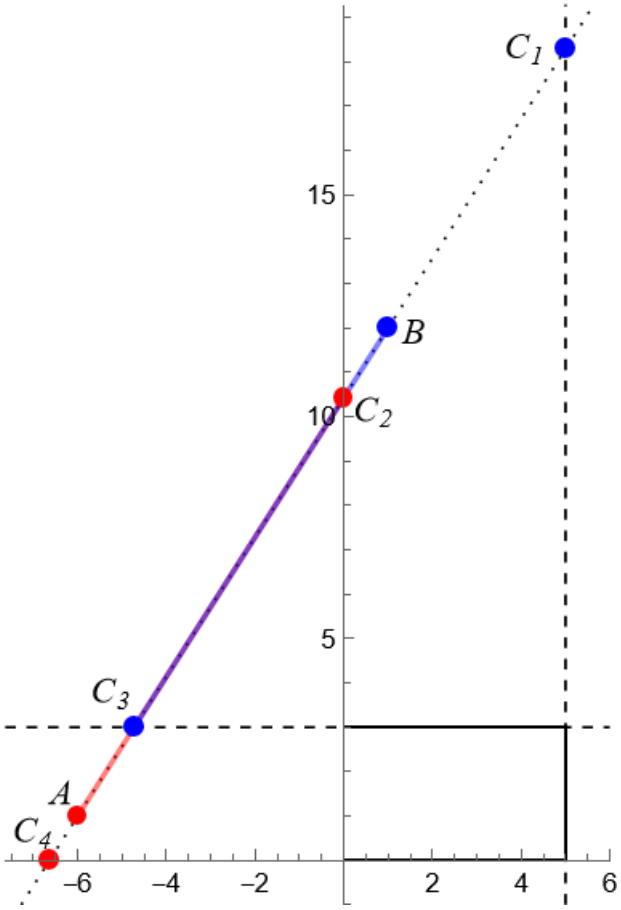
1.3. (Перейти к условию).

Отрезок C_2C_3 с концами в точках $C_2\left(\frac{11}{5},7\right)$ и $C_3\left(5,\frac{21}{4}\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_1\left(0,\frac{67}{8}\right)$ и $C_4\left(\frac{67}{5},0\right)$. Параметры t, соответствующие этим точкам, равны $3/5$, $1/4$, $7/8$, $-4/5$.



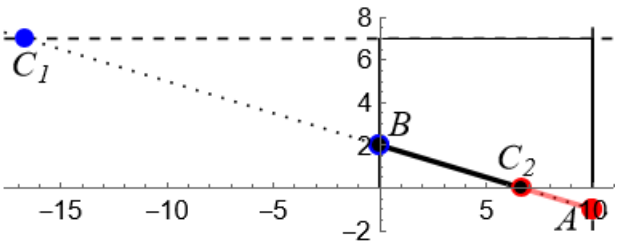
1.4. (Перейти к условию).

Происходит полное отсечение. Прямая AB пересекает продолжения границ окна в точках $C_1\left(5,\frac{128}{7}\right)$, $C_2\left(0,\frac{73}{7}\right)$, $C_3\left(-\frac{52}{11},3\right)$ и $C_4\left(-\frac{73}{11},0\right)$. Параметры t, соответствующие этим точкам, равны $11/7$, $6/7$, $2/11$, $-1/11$.



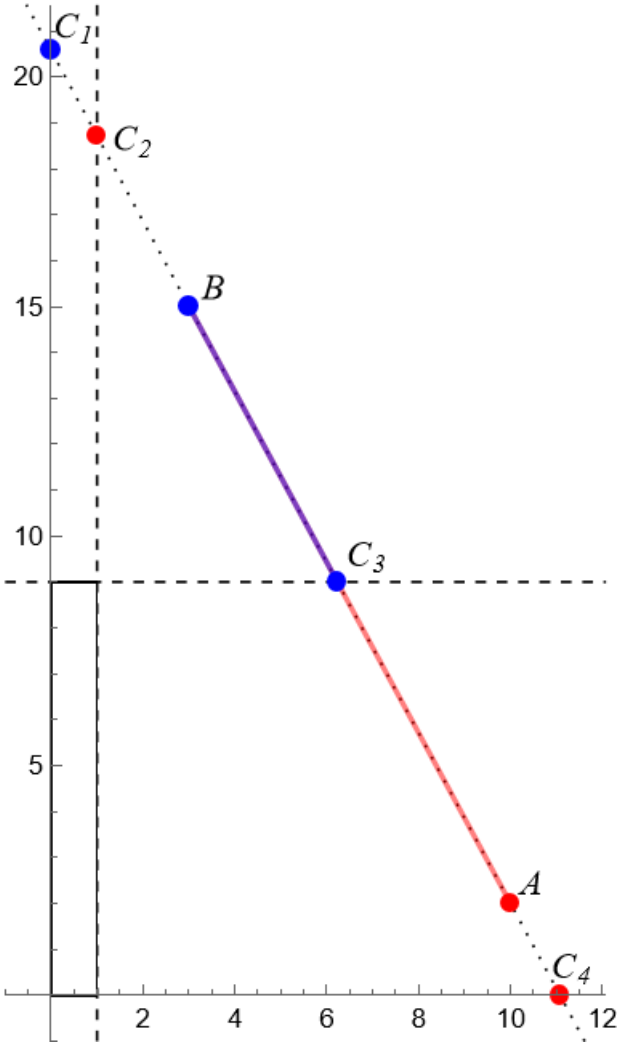
1.5. (Перейти к условию).

Отрезок BC_2 с концами в точках $B(0, 2)$ и $C_2\left(\frac{20}{3}, 0\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_1\left(-\frac{50}{3}, 7\right)$ и $A(10, -1)$. Параметры t, соответствующие этим точкам, равны $1,\ 1/3,\ 8/3,\ 0$.



1.6. (Перейти к условию).

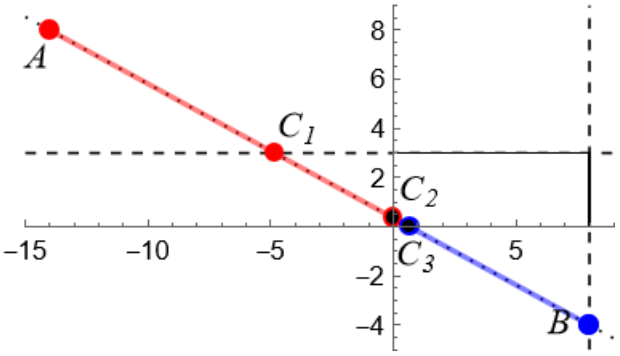
Происходит полное отсечение. Прямая AB пересекает продолжения границ окна в точках $C_1\left(0, \frac{144}{7}\right)$, $C_2\left(1, \frac{131}{7}\right)$, $C_3\left(\frac{81}{13}, 9\right)$ и $C_4\left(\frac{144}{13}, 0\right)$. Параметры t, соответствующие этим точкам, равны $10/7,\ 9/7,\ 7/13,\ -2/13$.



1.7. (Перейти к условию).

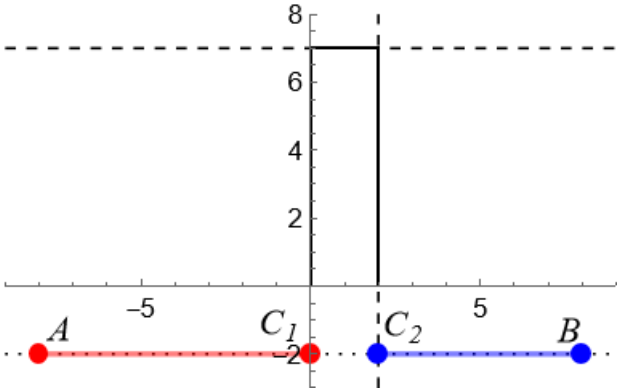
Отрезок C_2C_3 с концами в точках $C_2\left(0,\frac{4}{11}\right)$ и $C_3\left(\frac{2}{3},0\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_1\left(-\frac{29}{6},3\right)$ и $B(8,-4)$

Параметры t, соответствующие этим точкам, равны 7/11, 2/3, 5/12, 1.



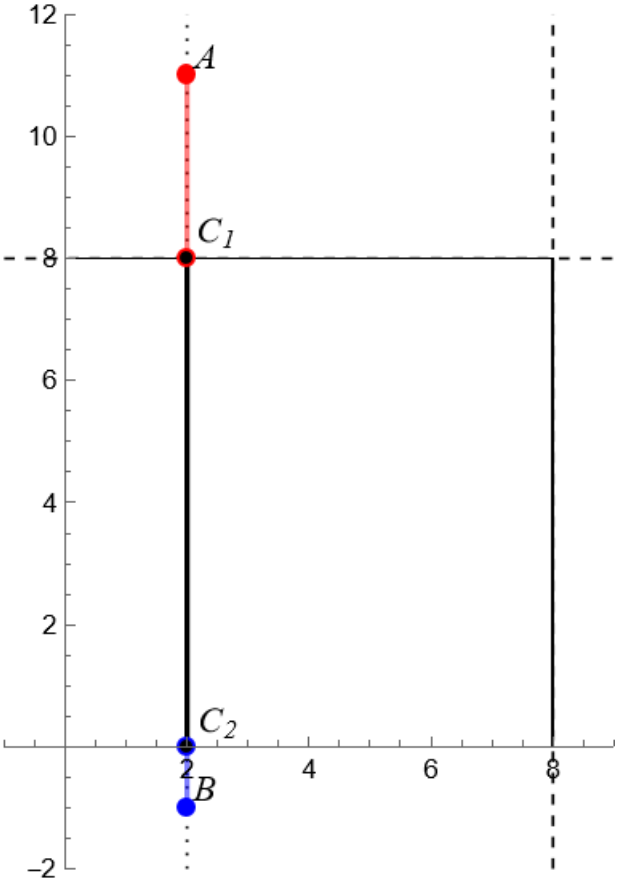
1.8. (Перейти к условию).

Происходит полное отсечение. Прямая AB пересекает продолжения границ окна в точках $C_1(0,-2)$ и $C_2(2,-2)$. Параметры t, соответствующие этим точкам, равны 1/2 и 5/8. Отсечения, изображённые на рисунке ниже, происходят в ходе алгоритма Лианга-Барски, если во время первых двух итераций происходит отсечение левой и правой границей окна. После этого происходит полное отсечение нижней границей.



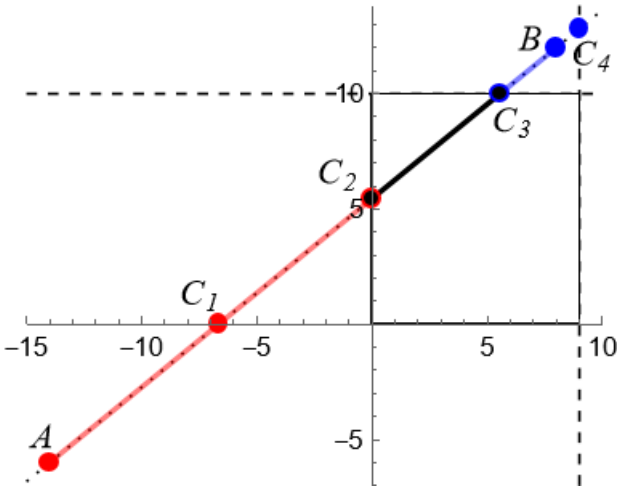
1.9. (Перейти к условию).

Отрезок C_1C_2 с концами в точках $C_1(2,8)$ и $C_2(2,0)$. Параметры t, соответствующие ним, равны 1/4 и 11/12.

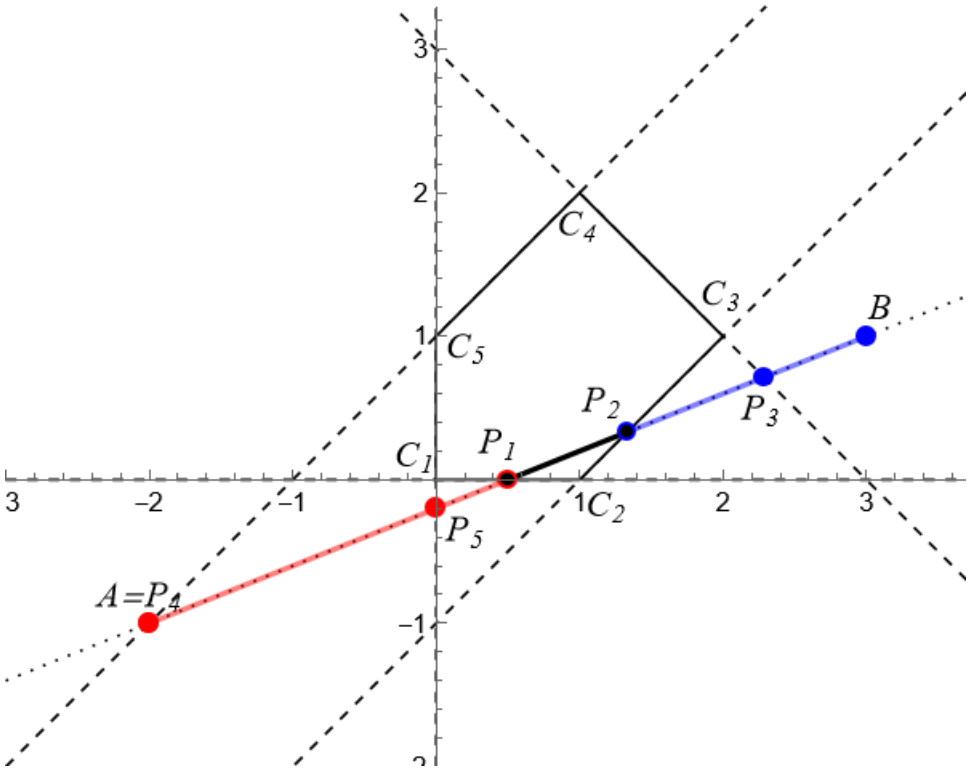


1.10. (Перейти к условию).

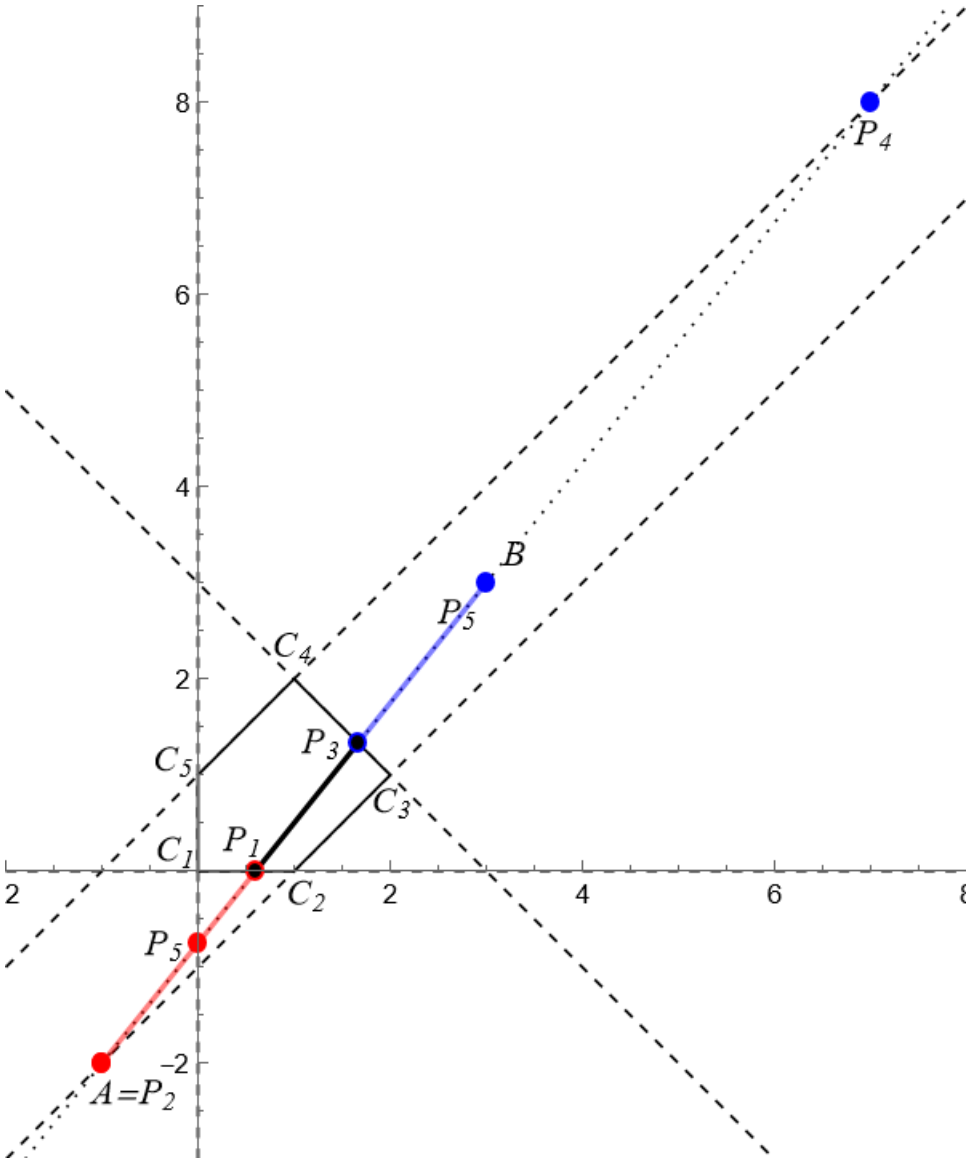
Отрезок C_2C_3 с концами в точках $C_2\left(0,\frac{60}{11}\right)$ и $C_3\left(\frac{50}{9},10\right)$. Прямая AB пересекает продолжения границ окна также в точках $C_1\left(-\frac{20}{3},0\right)$ и $C_4\left(9,\frac{141}{11}\right)$. Параметры t, соответствующие этим точкам, равны 7/11, 8/9, 1/3, 23/22.



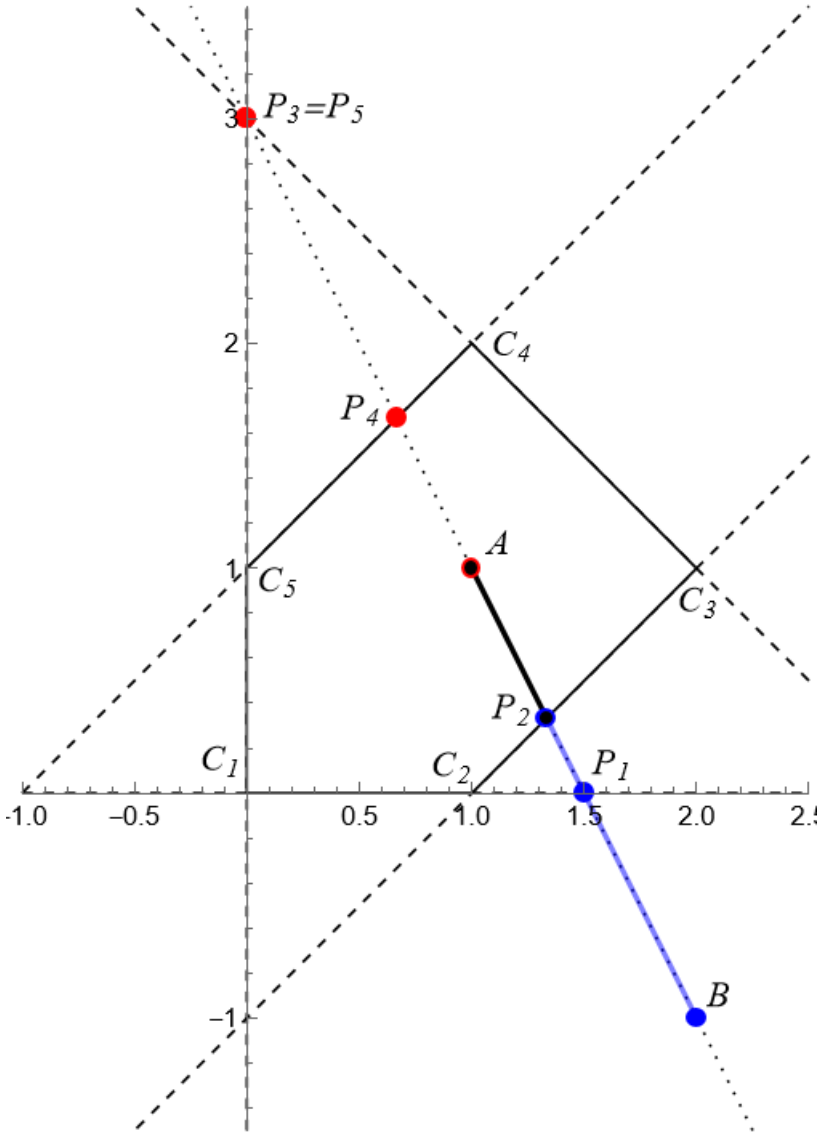
2.1. (Перейти к условию). Отрезок P_1P_2 с концами в точках $P_1\left(\frac{1}{2},0\right)$ и $P_2\left(\frac{4}{3},\frac{1}{3}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_3\left(\frac{16}{7},\frac{5}{7}\right)$, $P_4 = A(-2,-1)$ и $P_5\left(0,-\frac{1}{5}\right)$. Параметры t, соответствующие этим точкам, равны $1/2$, $2/3$, $6/7$, 0, $2/5$.



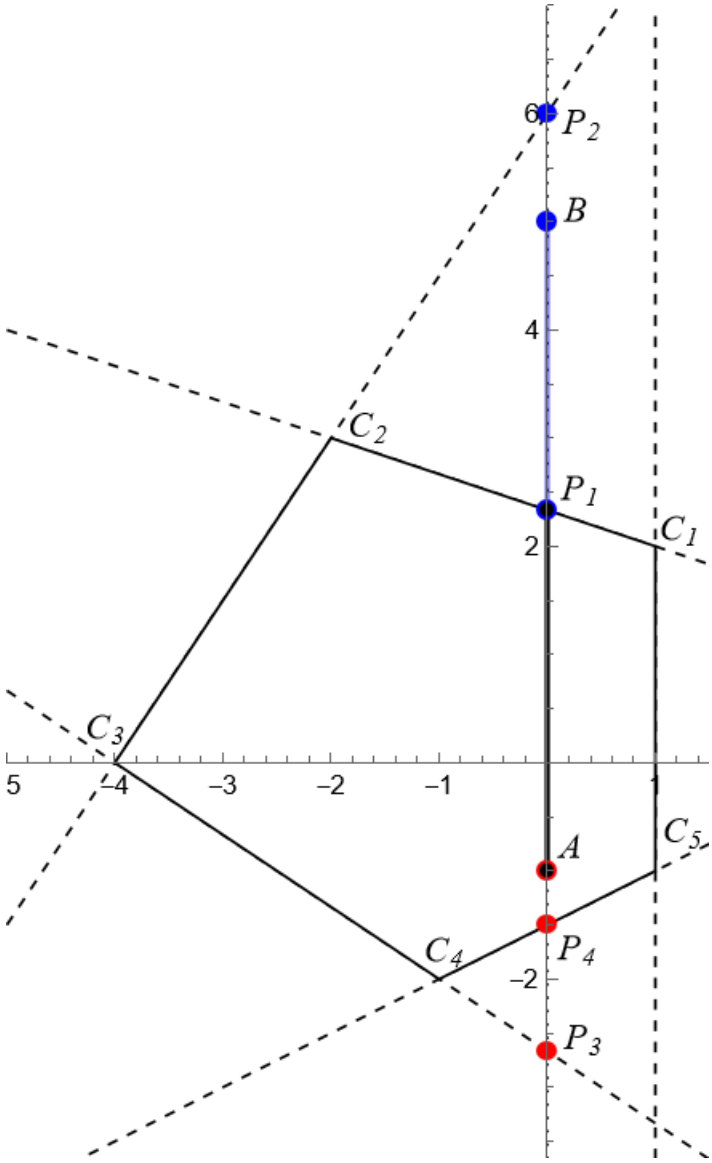
2.2. (Перейти к условию). Отрезок P_1P_3 с концами в точках $P_1\left(\frac{3}{5},0\right)$ и $P_3\left(\frac{5}{3},\frac{4}{3}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_2 = A(-1,-2)$, $P_4(7,8)$ и $P_5\left(0,-\frac{3}{4}\right)$. Параметры t, соответствующие этим точкам, равны $2/5,\ 2/3,\ 0,\ 2,\ 1/4$.



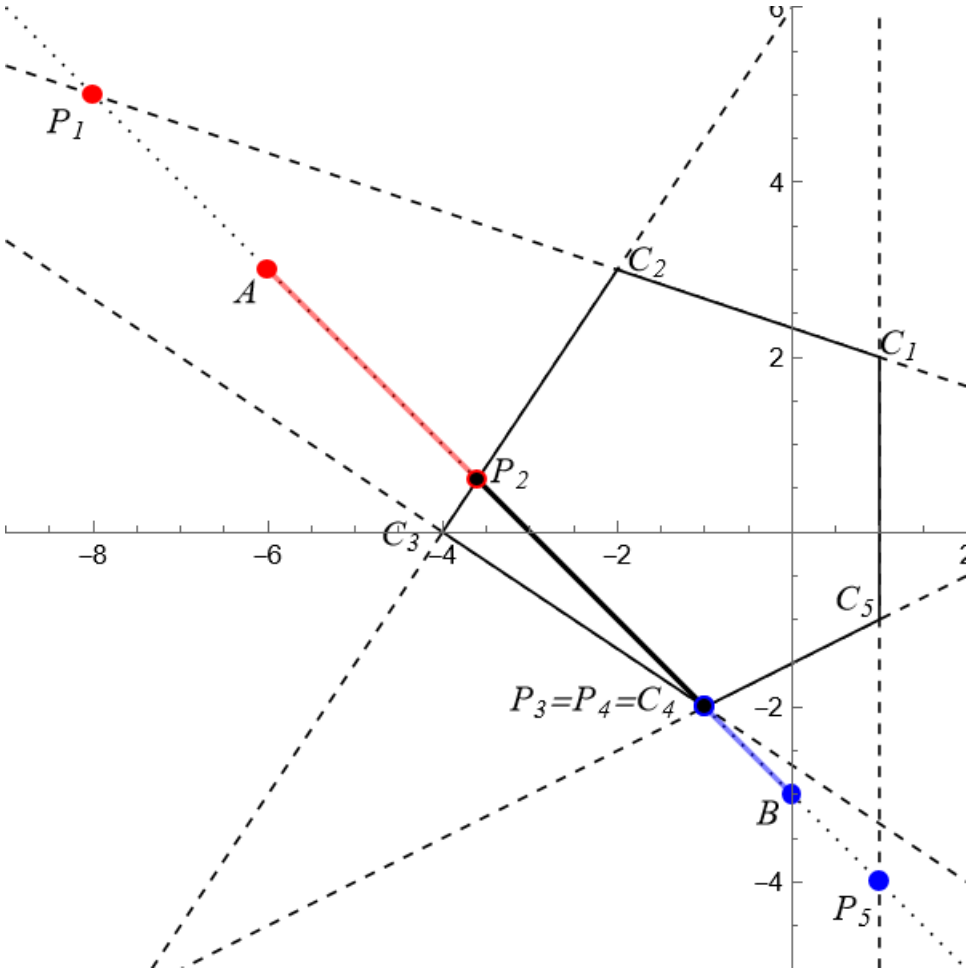
2.3. (Перейти к условию). Отрезок AP_2 с концами в точках $A(1, 1)$ и $P_2\left(\frac{4}{3},\frac{1}{3}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_1\left(\frac{3}{2},0\right)$, $P_3 = P_5(0,3)$ и $P_4\left(\frac{2}{3},\frac{5}{3}\right)$. Параметры t, соответствующие этим точкам, равны 0, 1/3, 1/2, –1, –1/3.



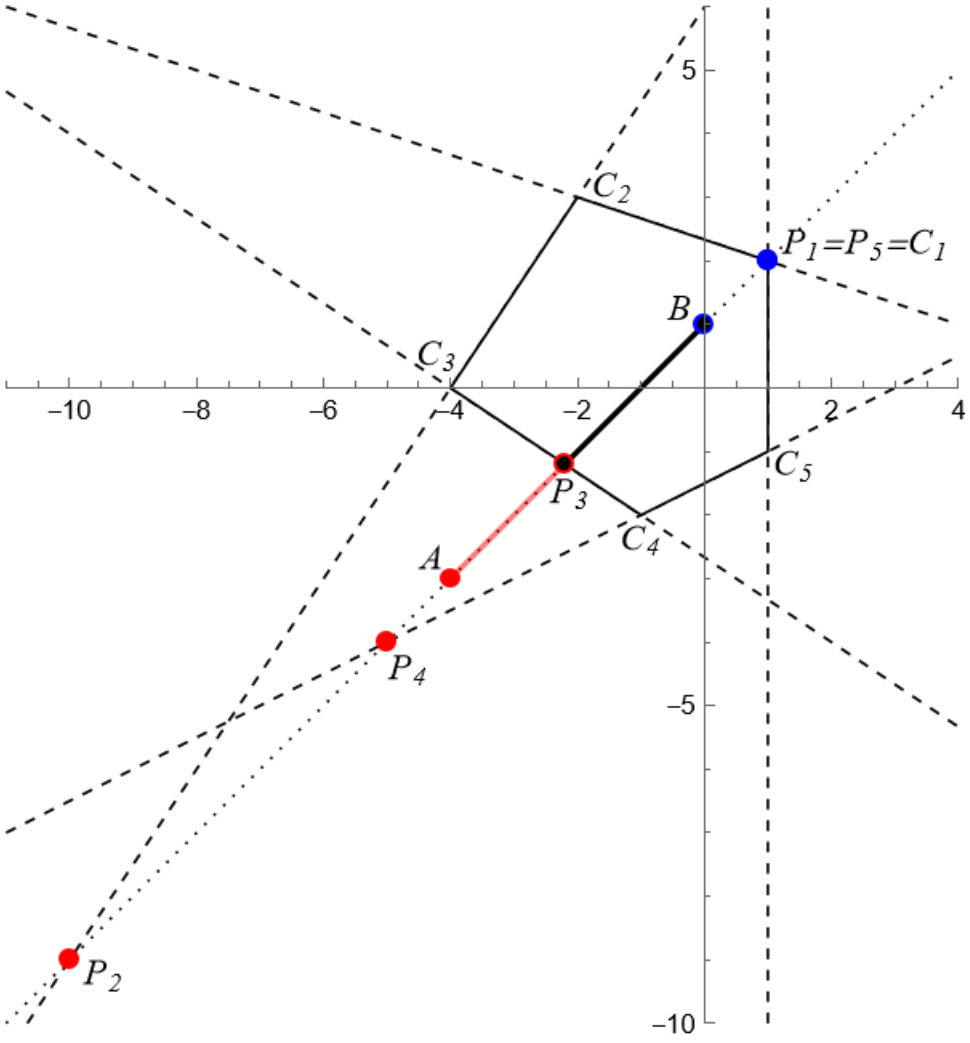
2.4. (Перейти к условию). Отрезок AP_1 с концами в точках $A(0,-1)$ и $P_1\left(0,\frac{7}{3}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_2(0,6)$, $P_3\left(0,-\frac{8}{3}\right)$ и $P_4\left(0,-\frac{3}{2}\right)$. Параметры t, соответствующие этим точкам, равны $0,\ 5/9,\ 7/6,\ -5/18,\ -1/12$.



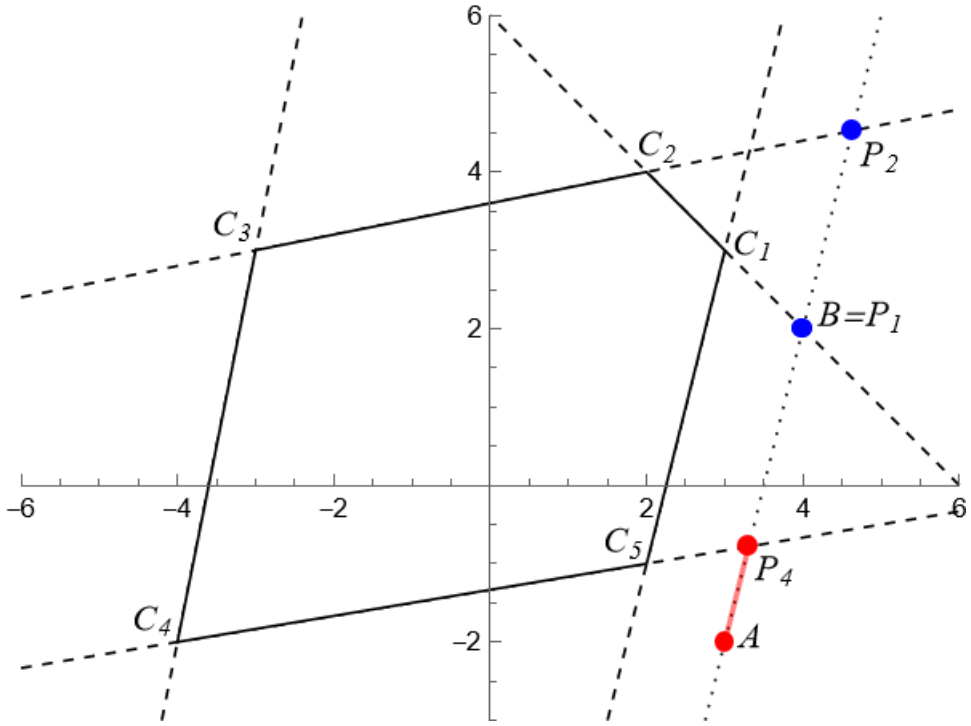
2.5. (Перейти к условию). Отрезок P_2C_4 с концами в точках $P_2\left(-\frac{18}{5},\frac{3}{5}\right)$ и $C_4 = P_3 = P_4(-1,-2)$. Прямая AB пересекает продолжения границ окна также в точках $P_1(-8,5)$ и $P_5(1,-4)$. Параметры t, соответствующие этим точкам, равны $2/5,\ 5/6,\ -1/3,\ 7/6$.



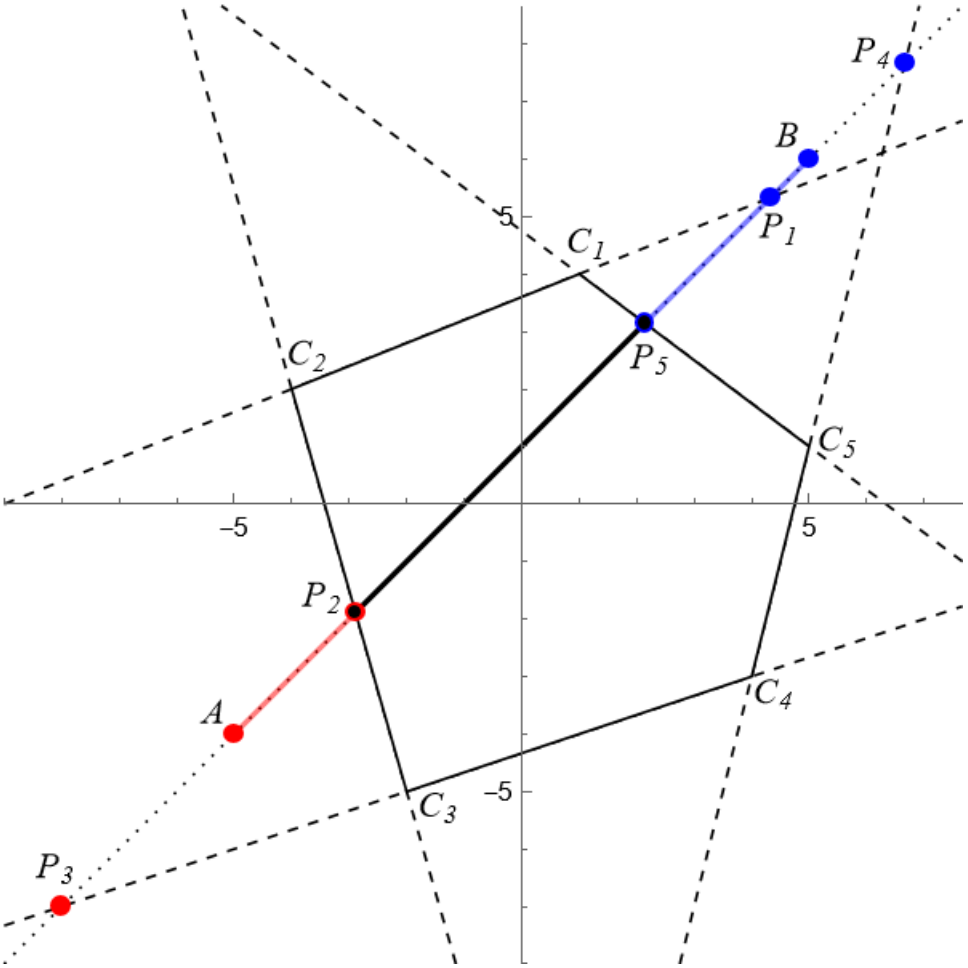
2.6. (Перейти к условию). Отрезок BP_3 с концами в точках $B(0,\ 1)$ и $P_3\left(-\frac{11}{5},-\frac{6}{5}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_1 = P_5 = C_1(1,2)$, $P_2(-10,-9)$ и $P_4(-5,-4)$. Параметры t, соответствующие этим точкам, равны 1, $9/20$, $5/4$, $-3/2$, $-1/4$.



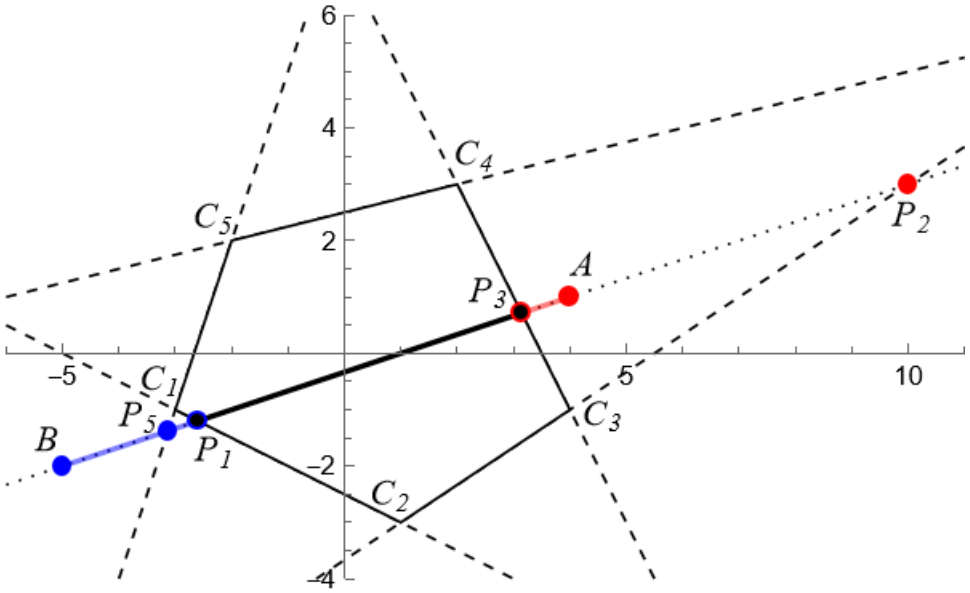
2.7. (Перейти к условию). Происходит полное отсечение. Прямая AB пересекает продолжения границ окна в точках $P_1 = B(4,2)$, $P_2\left(\frac{88}{19},\frac{86}{19}\right)$, $P_3(-32,-142)$ (не указана на чертеже) и $P_4\left(\frac{76}{23},-\frac{18}{23}\right)$. Параметры t, соответствующие этим точкам, равны 1, 31/19, –35, 7/23. Полное отсечение происходит ребром C_5C_1, параллельным отрезку AB.



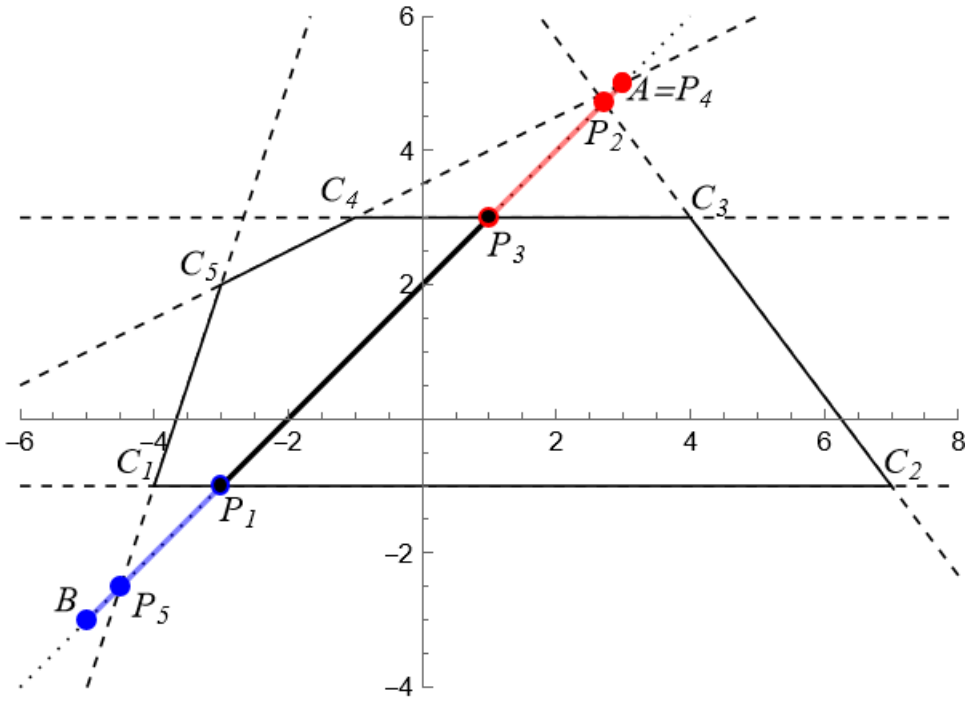
2.8. (Перейти к условию). Отрезок P_2P_5 с концами в точках $P_2\left(-\frac{26}{9},-\frac{17}{9}\right)$ и $P_5\left(\frac{15}{7},\frac{22}{7}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_1\left(\frac{13}{3},\frac{16}{3}\right)$, $P_3(-8,-7)$ и $P_4\left(\frac{20}{3},\frac{23}{3}\right)$. Параметры t, соответствующие этим точкам, равны $19/90$, $5/7$, $14/15$, $-3/10$, $7/6$.



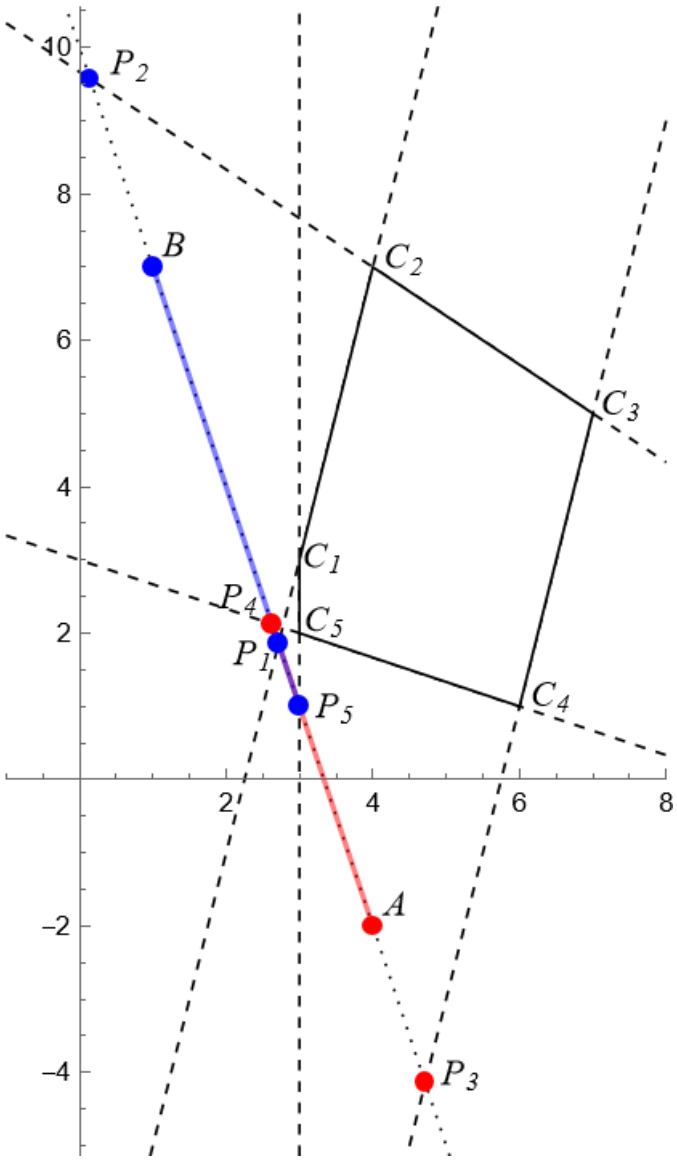
2.9. (Перейти к условию). Отрезок P_1P_3 с концами в точках $P_1\left(-\frac{13}{5},-\frac{6}{5}\right)$ и $P_3\left(\frac{22}{7},\frac{5}{7}\right)$. Прямая AB пересекает продолжения границ окна также в точках $P_2(10,3)$, $P_4(34,11)$ (не указана на чертеже) и $P_5\left(-\frac{25}{8},-\frac{11}{8}\right)$. Параметры t, соответствующие этим точкам, равны $11/15$, $2/21$, $-2/3$, $-10/3$, $19/24$.



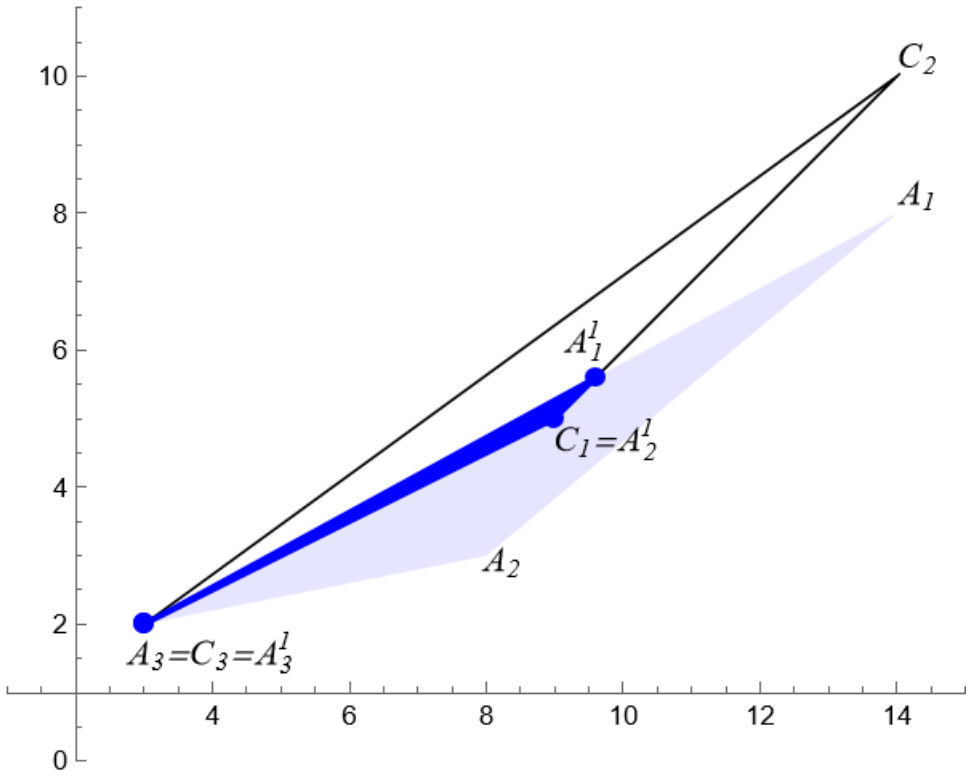
2.10. (Перейти к условию). Отрезок P_1P_3 с концами в точках $P_1(-3,-1)$ и $P_3(1,3)$. Прямая AB пересекает продолжения границ окна также в точках $P_2\left(\frac{19}{7},\frac{33}{7}\right)$, $P_4 = A(3,5)$ и $P_5\left(-\frac{9}{2},-\frac{5}{2}\right)$. Параметры t, соответствующие этим точкам, равны $3/4, 1/4, 1/28,\ 0,\ 15/16$.



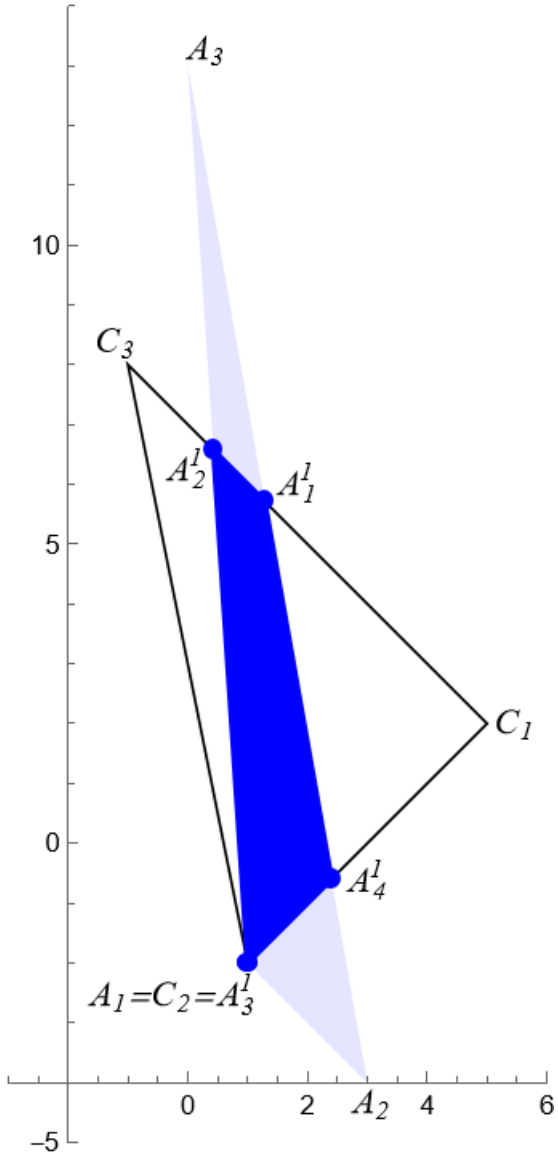
2.11. (Перейти к условию). Происходит полное отсечение. Прямая AB пересекает продолжения границ окна в точках $P_1\left(\frac{19}{7},\frac{13}{7}\right)$, $P_2\left(\frac{1}{7},\frac{67}{7}\right)$, $P_3\left(\frac{33}{7},-\frac{29}{7}\right)$, $P_4\left(\frac{21}{8},\frac{17}{8}\right)$ и $P_5(3,1)$. Параметры t, соответствующие этим точкам, равны $3/7$, $9/7$, $-5/21$, $11/24$, $1/3$.



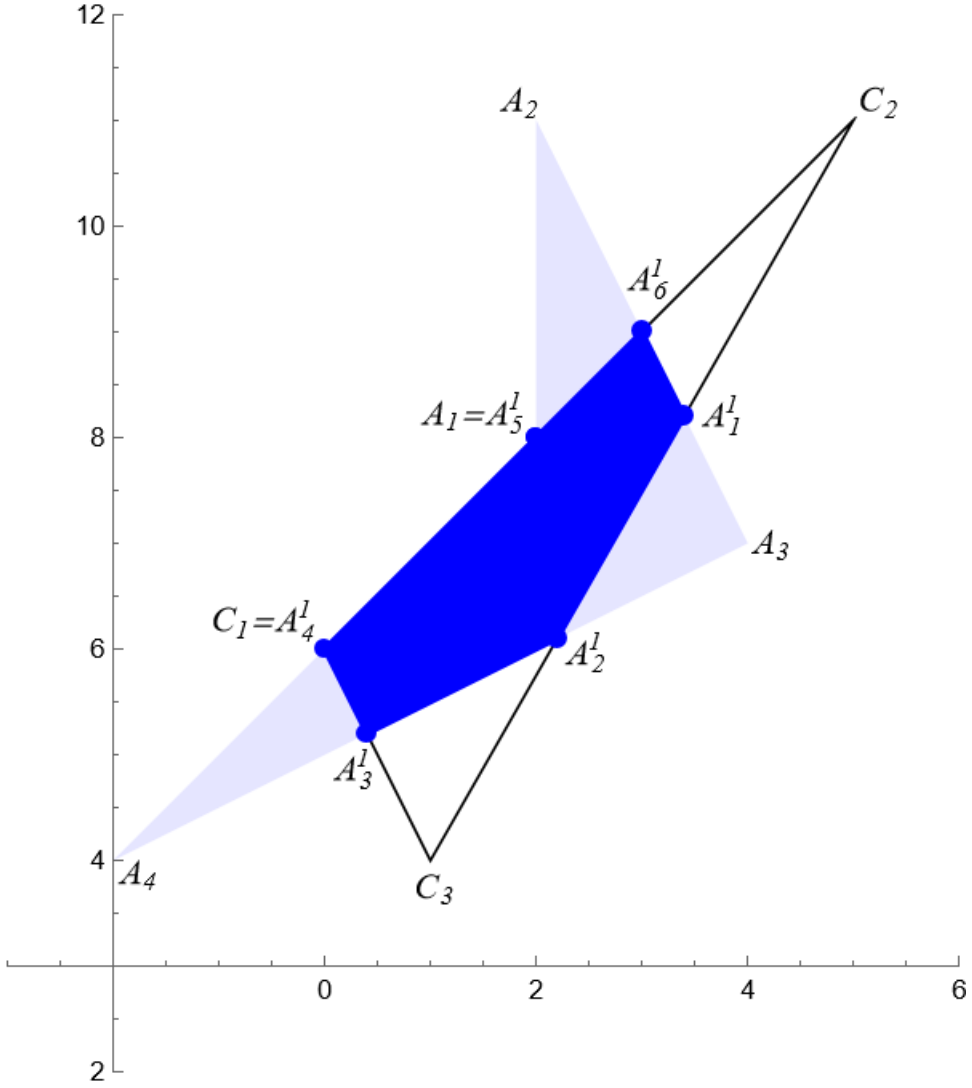
3.1. (Перейти к условию). Треугольник $\triangle A_1^1 A_2^1 A_3^1$ с вершинами $A_1^1\left(\frac{48}{5},\frac{28}{5}\right)$, $A_2^1 = C_1(9,5)$, $A_3^1 = = A_3 = C_3(3,2)$.



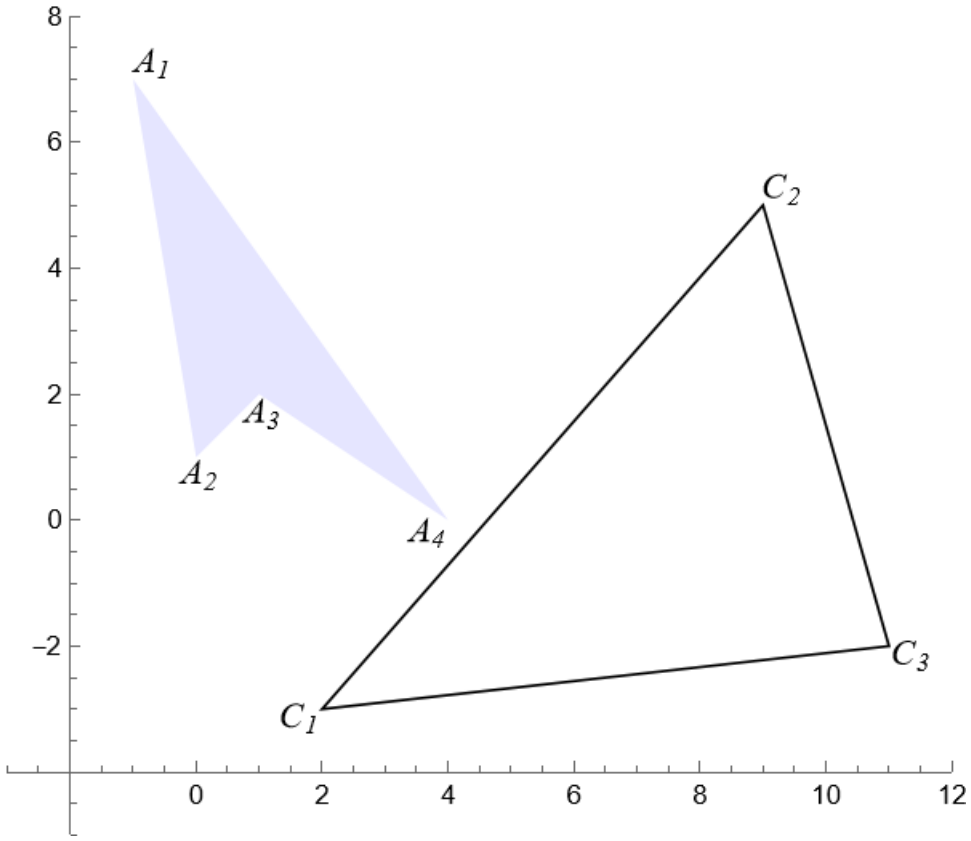
3.2. (Перейти к условию). Полигон $A_1^1A_2^1A_3^1A_4^1A_1^1$ с вершинами $A_1^1\left(\frac{9}{7},\frac{40}{7}\right)$, $A_2^1\left(\frac{3}{7},\frac{46}{7}\right)$, $A_3^1 = A_1 = = C_2(1,-2)$, $A_4^1\left(\frac{12}{5},-\frac{3}{5}\right)$.



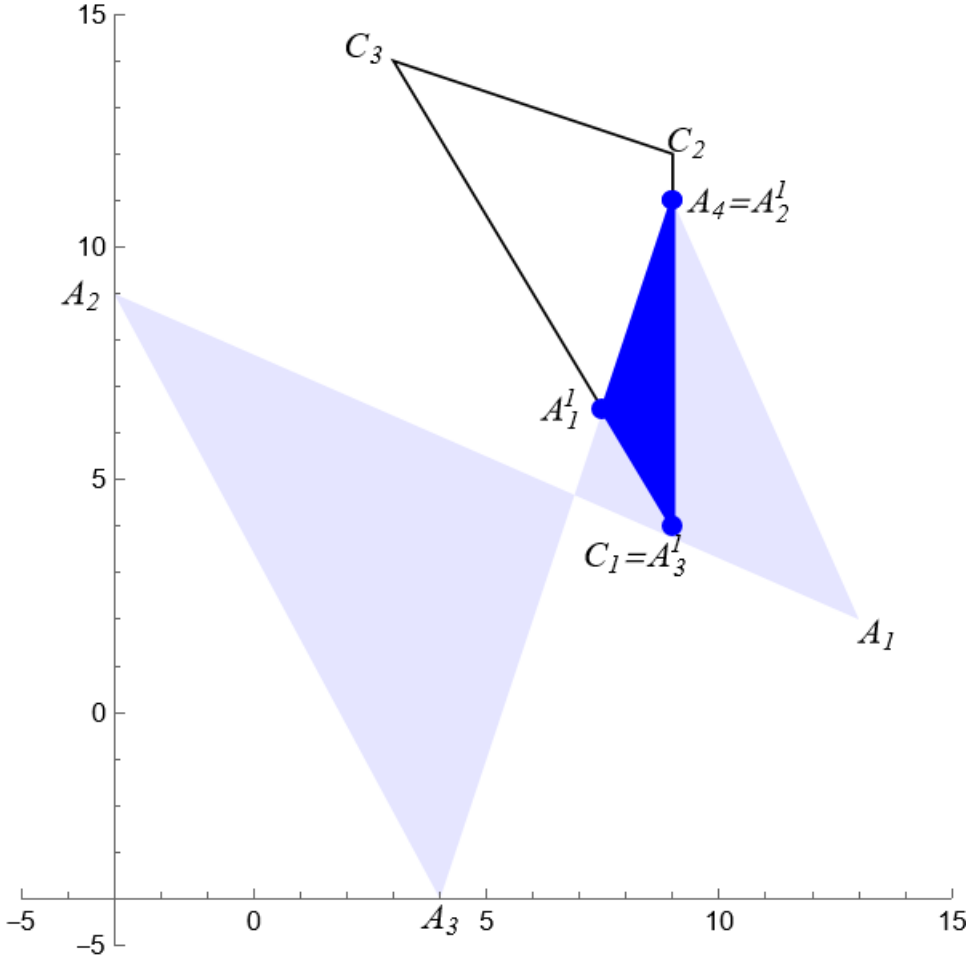
3.3. (Перейти к условию). Полигон $A_1^1A_2^1A_3^1A_4^1A_5^1A_6^1A_1^1$ с вершинами $A_1^1\left(\frac{17}{5},\frac{41}{5}\right)$, $A_2^1\left(\frac{11}{5},\frac{61}{10}\right)$, $A_3^1\left(\frac{2}{5},\frac{26}{5}\right)$, $A_4^1 = C_1(0,6)$, $A_5^1 = A_1(2,8)$, $A_6^1(3,9)$.



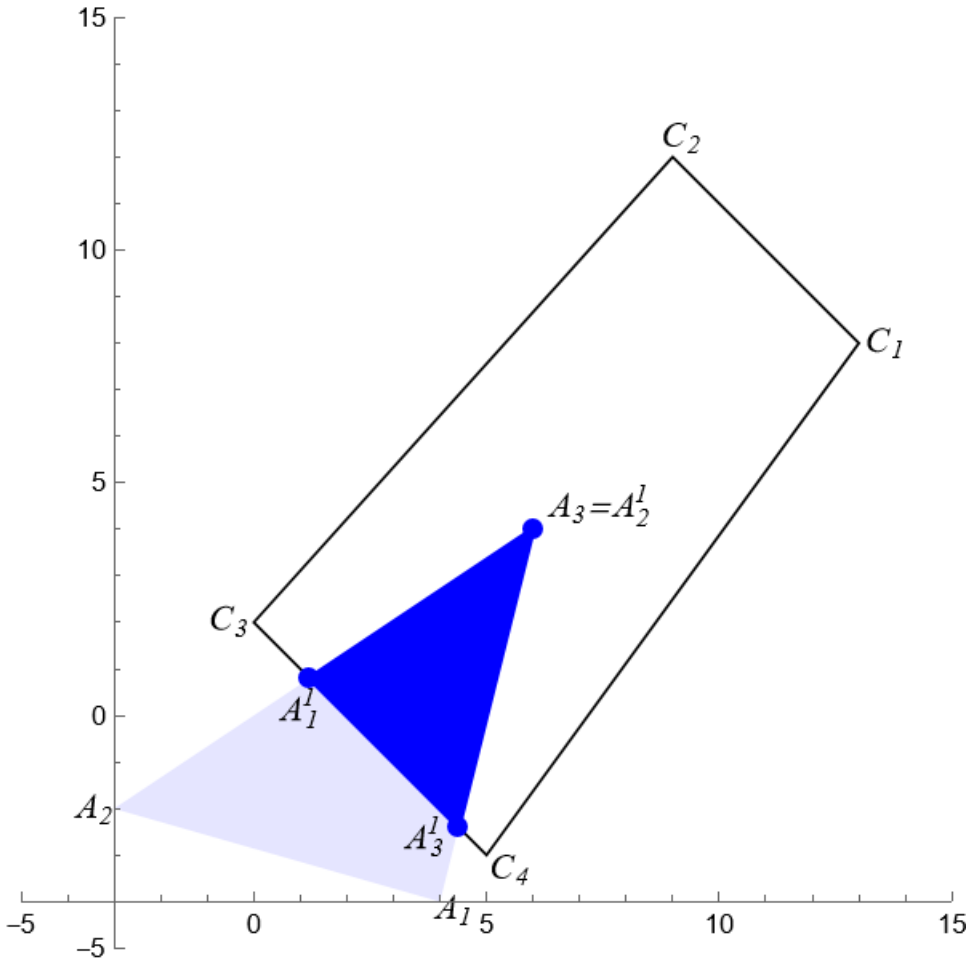
3.4. (Перейти к условию). Происходит полное отсечение.



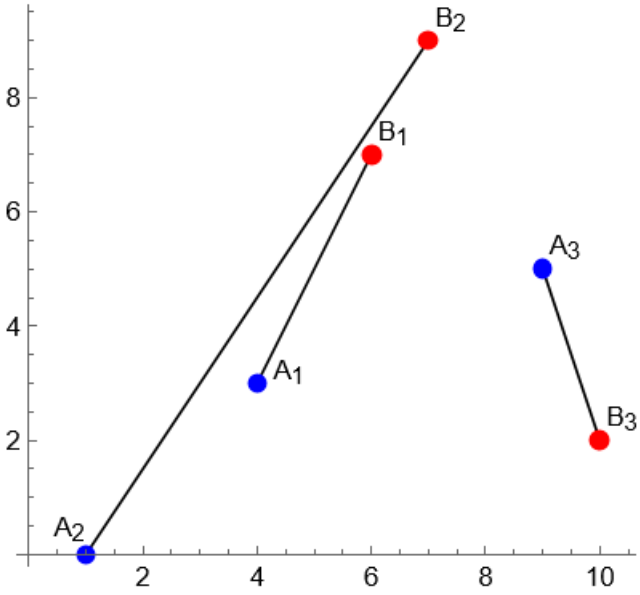
3.5. (Перейти к условию). Треугольник $\triangle A_1^1 A_2^1 A_3^1$ с вершинами $A_1^1\left(\frac{15}{2},\frac{13}{2}\right)$, $A_2^1 = A_4(9,11)$, $A_3^1 = C_1(9,4)$.



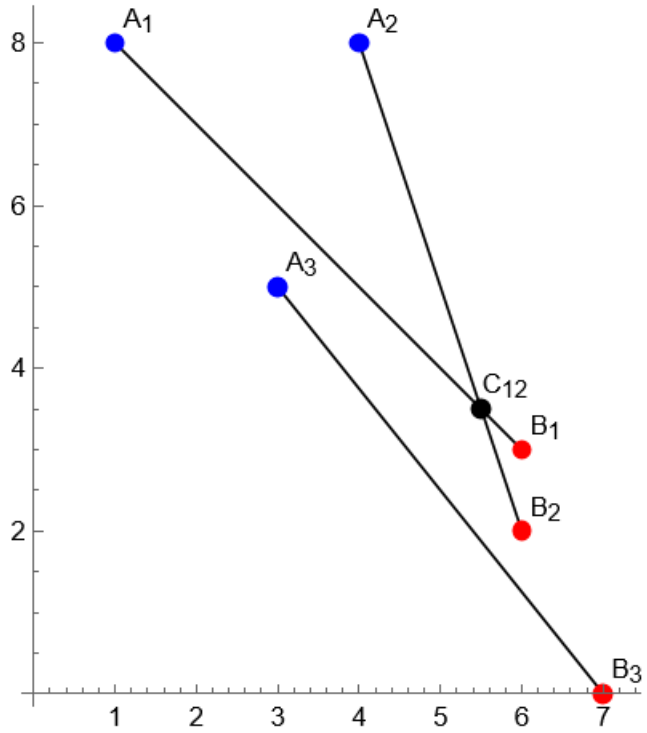
3.6. (Перейти к условию). Треугольник $\triangle A_1^1 A_2^1 A_3^1$ с вершинами $A_1^1\left(\frac{6}{5},\frac{4}{5}\right)$, $A_2^1 = A_3(6,4)$, $A_3^1\left(\frac{22}{5},-\frac{12}{5}\right)$.



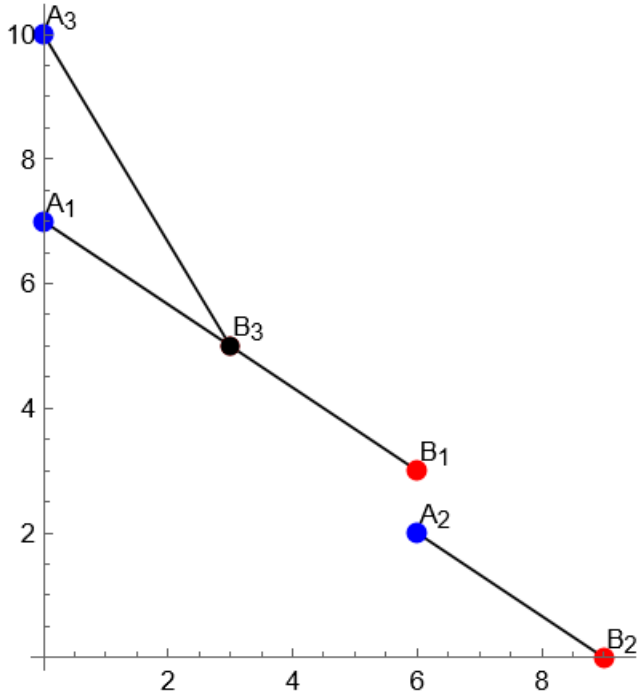
4.1. (Перейти к условию). Пересечений нет



4.2. (Перейти к условию). Отрезки A_1B_1 и A_2B_2 пересекаются в точке $C_{12}\left(\frac{11}{2},\frac{7}{2}\right)$



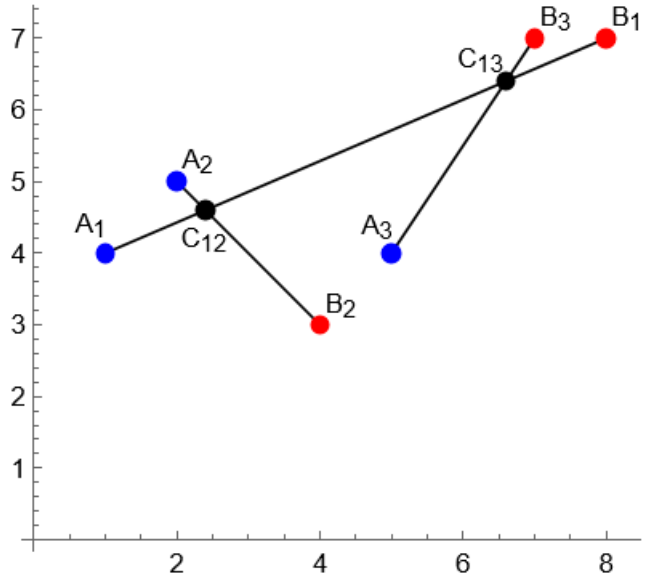
4.3. (Перейти к условию). Отрезки A_1B_1 и A_3B_3 пересекаются в точке B_3



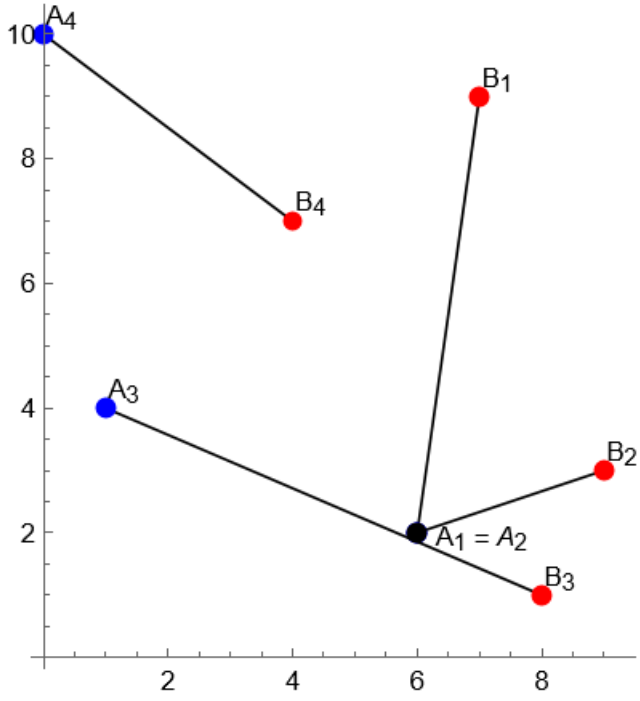
4.4. (Перейти к условию).
Пересекаются две пары отрезков:

$$A_1B_1 \cap A_2B_2 = C_{12}\left(\frac{12}{5},\frac{23}{5}\right),$$

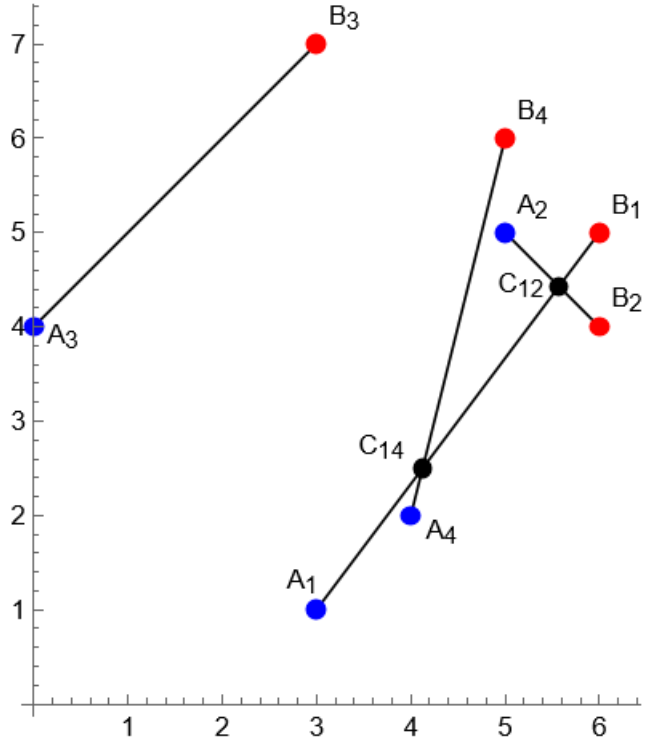
$$A_1B_1 \cap A_3B_3 = C_{13}\left(\frac{33}{5},\frac{32}{5}\right)$$



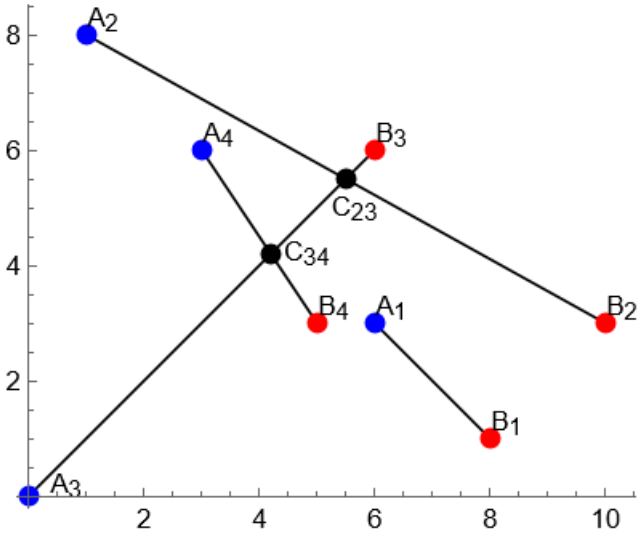
4.5. (Перейти к условию). Отрезки A_1B_1 и A_2B_2 являются смежными $(A_1 = A_2)$



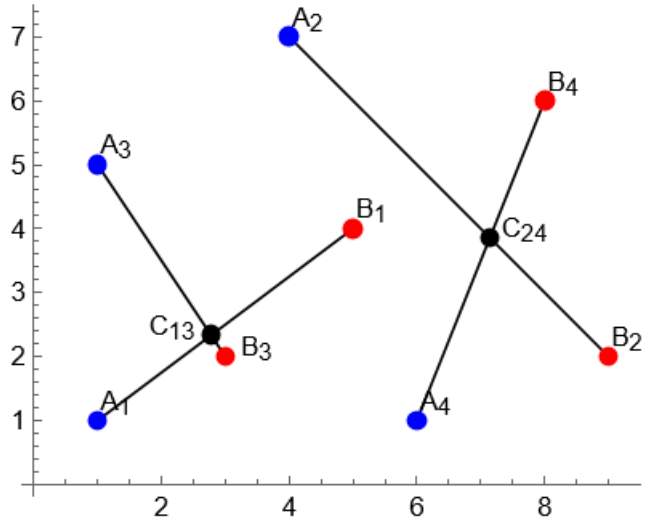
4.6. (Перейти к условию).
Пересекаются две пары отрезков:
$A_1B_1 \cap A_4B_4 = C_{14}(33/8, 5/2)$,
$A_1B_1 \cap A_2B_2 = C_{12}(39/7, 31/7)$



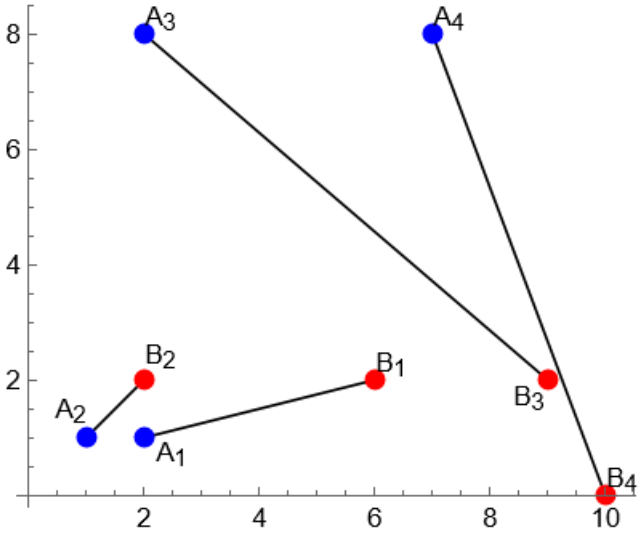
4.7. (Перейти к условию).
Пересекаются две пары отрезков:
$A_3B_3 \cap A_4B_4 = C_{34}(21/5, 21/5)$,
$A_3B_3 \cap A_2B_2 = C_{23}(11/2, 11/2)$



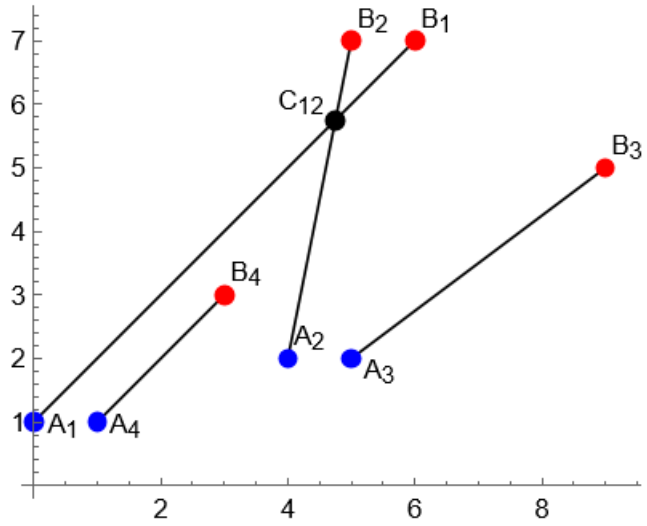
4.8. (Перейти к условию).
Пересекаются две пары отрезков:
$A_1B_1 \cap A_3B_3 = C_{13}(25/9, 7/3)$,
$A_2B_2 \cap A_4B_4 = C_{24}(50/7, 27/7)$



4.9. (Перейти к условию). Пересечений нет



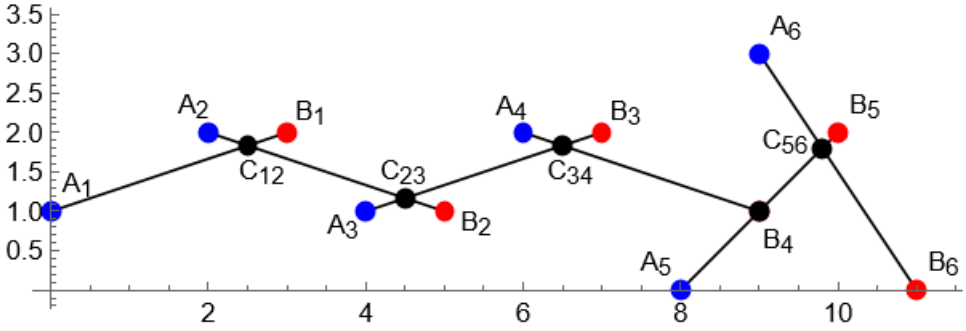
4.10. (Перейти к условию). Отрезки A_1B_1 и A_2B_2 пересекаются в точке $C_{12}\left(\frac{19}{4},\frac{23}{4}\right)$



4.11. (Перейти к условию). Имеют место следующие пересечения:

$A_1B_1 \cap A_2B_2 = C_{12}\left(\frac{5}{2},\frac{11}{6}\right)$, $A_2B_2 \cap A_3B_3 = C_{23}\left(\frac{9}{2},\frac{7}{6}\right)$, $A_3B_3 \cap A_4B_4 = C_{34}\left(\frac{13}{2},\frac{11}{6}\right)$,

$A_4B_4 \cap A_5B_5 = B_4$, $A_5B_5 \cap A_6B_6 = C_{56}\left(\frac{49}{5},\frac{9}{5}\right)$



5.1. (Перейти к условию).

Полигоны $B_{22}B_{23}B_{33}B_{32}B_{22}$ и $\triangle B_{41}B_{44}C_1$, где

$B_{22}\left(13/2,2\right)=A_2A_3\cap C_2C_3$,

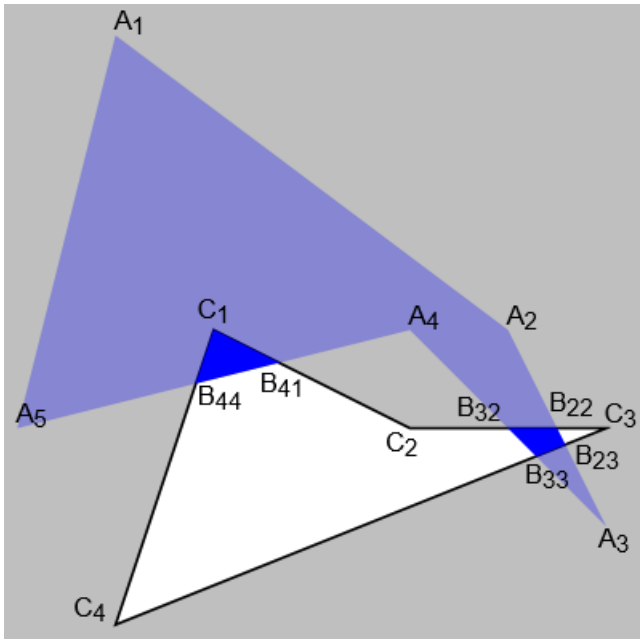
$B_{23}\left(79/12,11/6\right)=A_2A_3\cap C_3C_4$,

$B_{33}\left(44/7,12/7\right)=A_3A_4\cap C_3C_4$,

$B_{32}\left(6,2\right)=A_3A_4\cap C_2C_3$,

$B_{41}\left(11/3,8/3\right)=A_4A_5\cap C_1C_2$,

$B_{44}\left(31/11,27/11\right)=A_4A_5\cap C_4C_1$



5.2. (Перейти к условию).

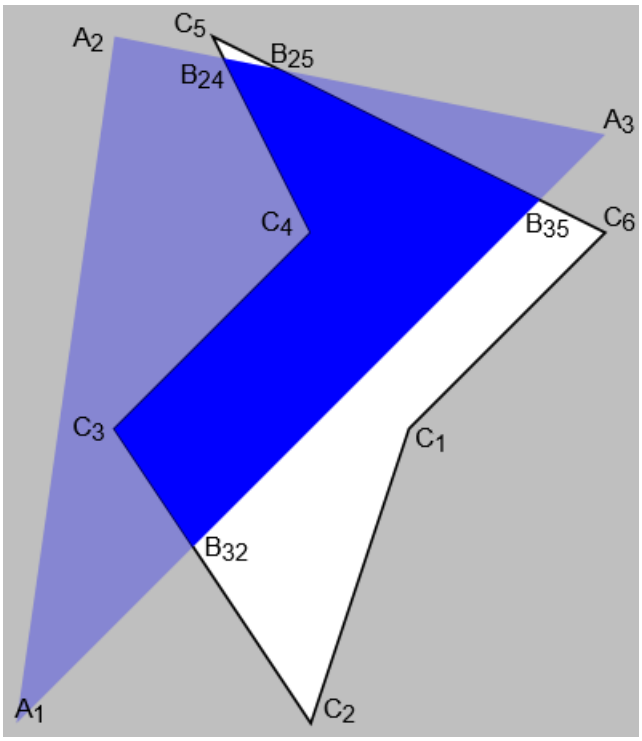
Полигон $B_{24}B_{25}B_{35}B_{32}C_3C_4B_{24}$, где

$B_{24}\left(19/9,70/9\right)=A_2A_3\cap C_4C_5$,

$B_{25}\left(8/3,23/3\right)=A_2A_3\cap C_5C_6$,

$B_{35}\left(16/3,19/3\right)=A_3A_1\cap C_5C_6$,

$B_{32}\left(9/5,14/5\right)=A_3A_1\cap C_2C_3$



5.3. (Перейти к условию).

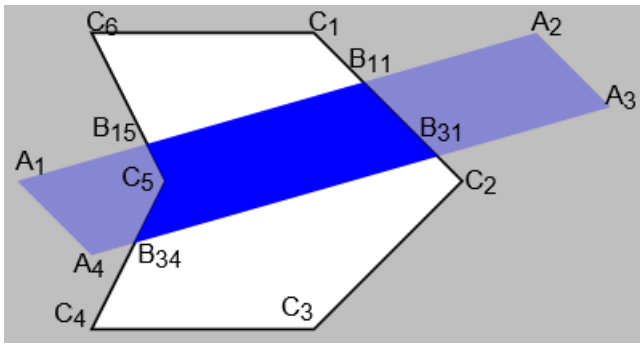
Полигон $B_{34}C_5B_{15}B_{11}B_{31}B_{34}$, где

$B_{34}\left(7/12,7/6\right)=A_3A_4\cap C_4C_5$,

$B_{15}\left(3/4,5/2\right)=A_1A_2\cap C_5C_6$,

$B_{11}\left(11/3,10/3\right)=A_1A_2\cap C_1C_2$,

$B_{31}\left(14/3,7/3\right)=A_3A_4\cap C_1C_2$



5.4. (Перейти к условию).

Полигон

$B_{21}B_{22}B_{32}A_4B_{44}B_{14}B_{11}B_{21}$, где

$B_{21}(4,3)=A_2A_3\cap C_1C_2$,

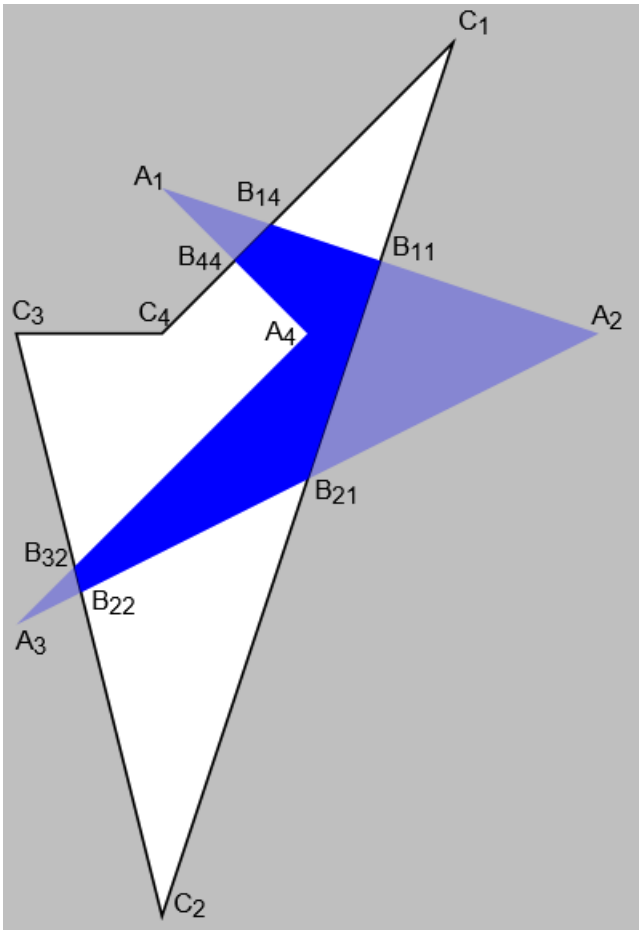
$B_{22}\left(22/9,20/9\right)=A_2A_3\cap C_2C_3$,

$B_{32}\left(12/5,12/5\right)=A_3A_4\cap C_2C_3$,

$B_{44}\left(7/2,9/2\right)=A_4A_5\cap C_4C_1$,

$B_{14}\left(15/4,19/4\right)=A_1A_2\cap C_4C_1$,

$B_{11}\left(9/2,9/2\right)=A_1A_2\cap C_1C_2$



5.5. (Перейти к условию).

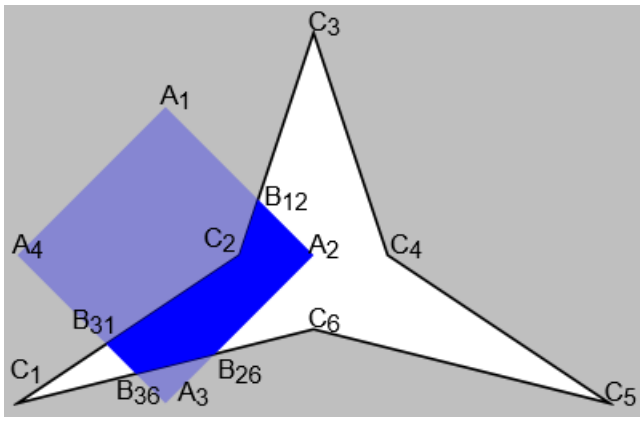
Полигон $B_{31}C_2B_{12}A_2B_{26}B_{36}B_{31}$, где

$B_{31}(6/5,4/5) = A_3A_4 \cap C_1C_2$,

$B_{12}(13/4,11/4) = A_1A_2 \cap C_2C_3$,

$B_{26}(8/3,2/3) = A_2A_3 \cap C_6C_1$,

$B_{36}(8/5,2/5) = A_3A_4 \cap C_6C_1$



5.6. (Перейти к условию).

Полигоны

$\triangle B_{35}C_6B_{36}$, $\triangle B_{37}C_1B_{31}$, $\triangle B_{32}C_3B_{33}$, где

$B_{35}(-5/3,2) = A_3A_4 \cap C_5C_6$,

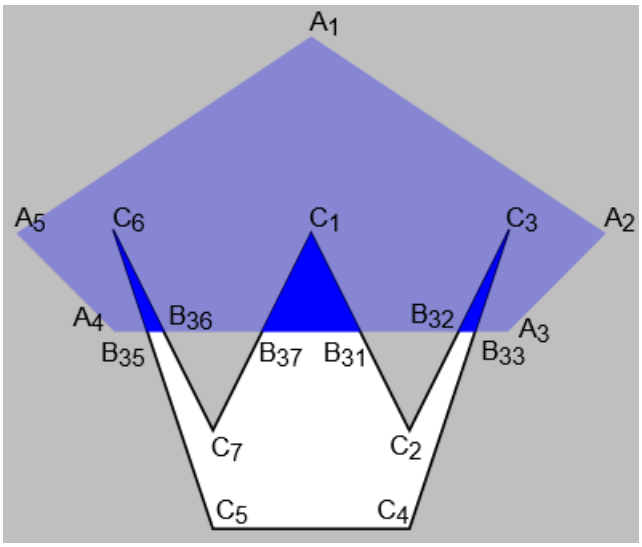
$B_{36}(-3/2,2) = A_3A_4 \cap C_6C_7$,

$B_{37}(-1/2,2) = A_3A_4 \cap C_7C_1$,

$B_{31}(1/2,2) = A_3A_4 \cap C_1C_2$,

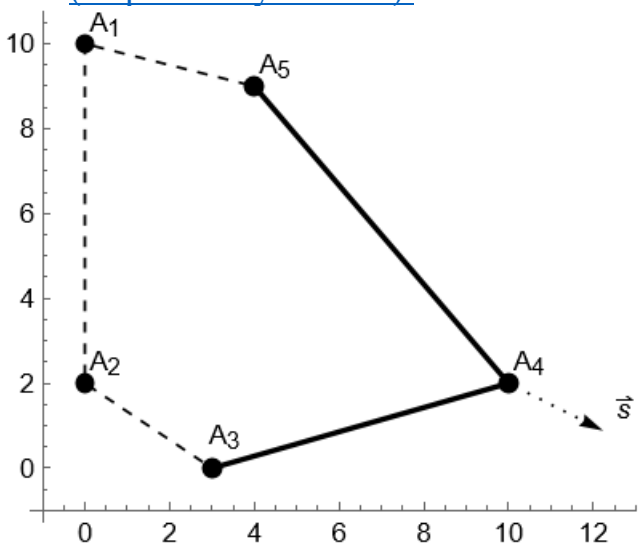
$B_{32}(3/2,2) = A_3A_4 \cap C_2C_3$,

$B_{33}(5/3,2) = A_3A_4 \cap C_3C_4$

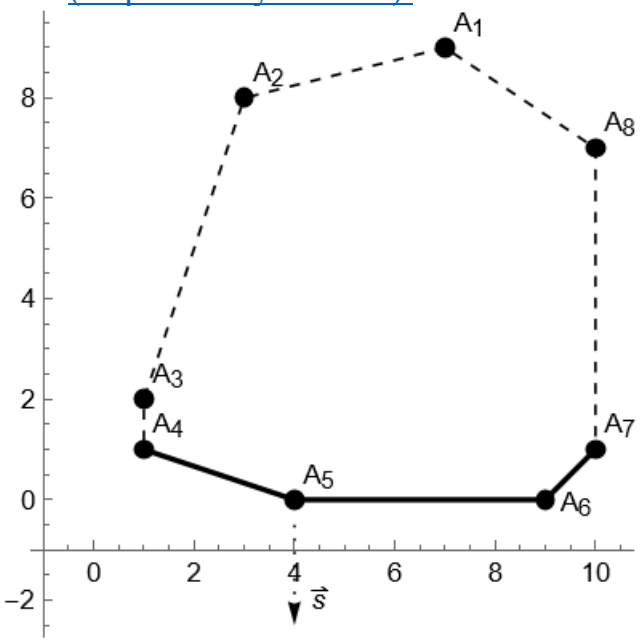


6. В ответах приведены рисунки, на которых сплошной ломаной линией изображена видимая часть границы исходного полигона, пунктирной – невидимая, а стрелкой указано направление на наблюдателя.

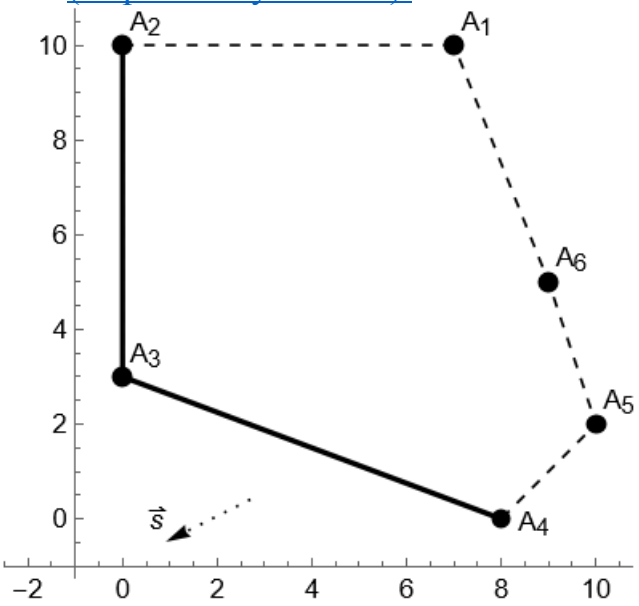
6.1. (Перейти к условию).



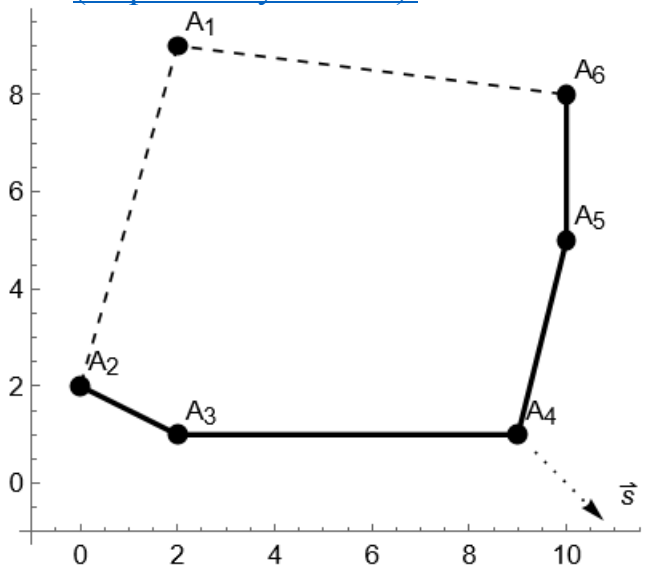
6.2. (Перейти к условию).



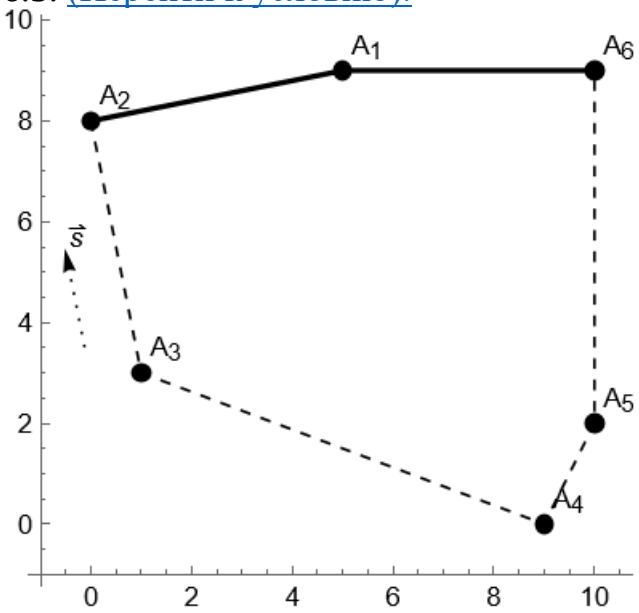
6.3. (Перейти к условию).



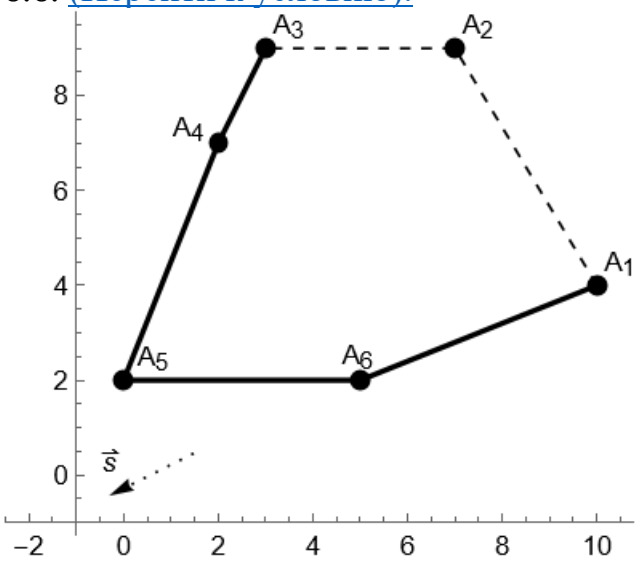
6.4. (Перейти к условию).



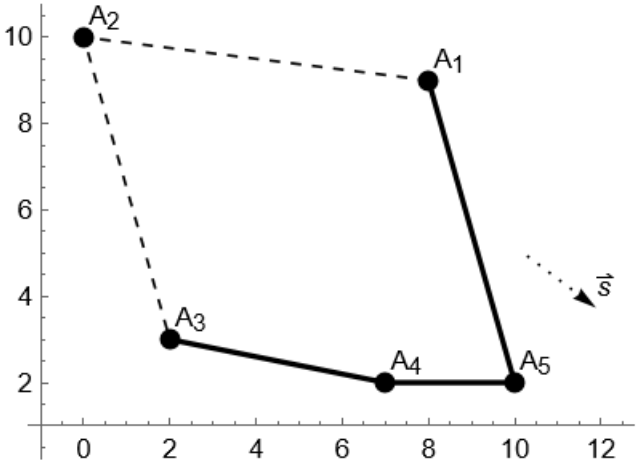
6.5. (Перейти к условию).



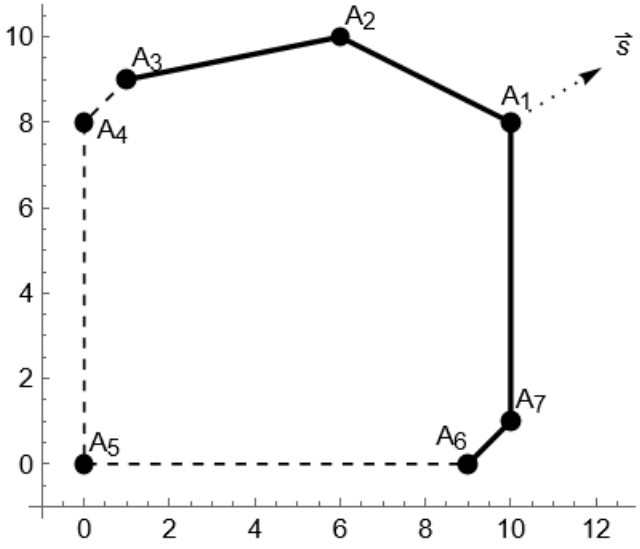
6.6. (Перейти к условию).



6.7. (Перейти к условию).

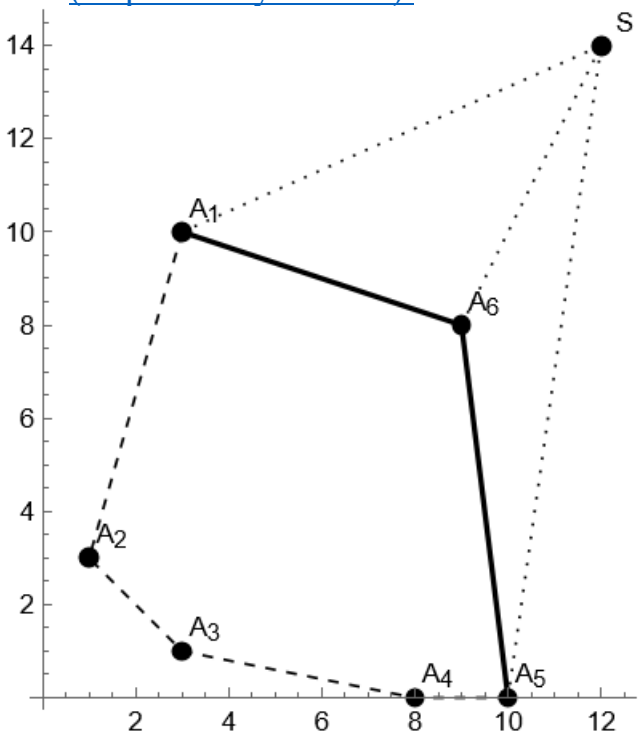


6.8. (Перейти к условию).

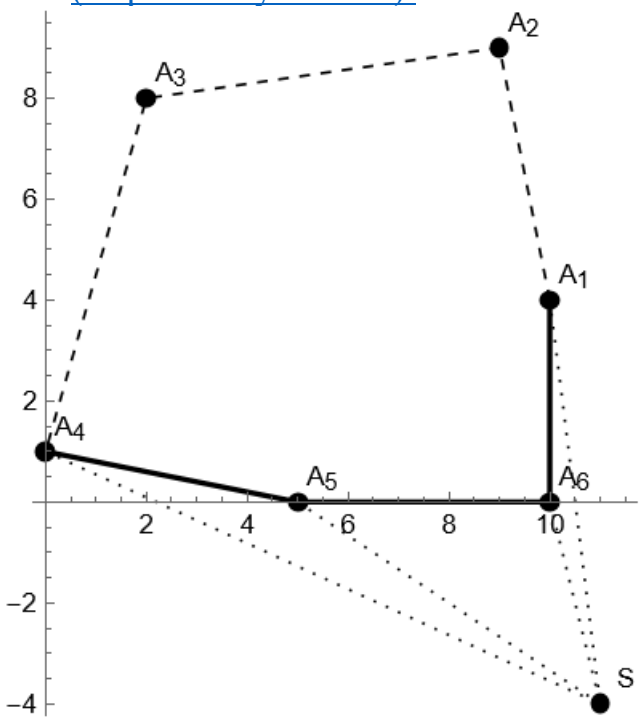


7. На каждом из рисунков ниже указаны видимая и невидимая части границ заданного полигона, положение наблюдателя, а также проведены отрезки от наблюдателя к видимым вершинам.

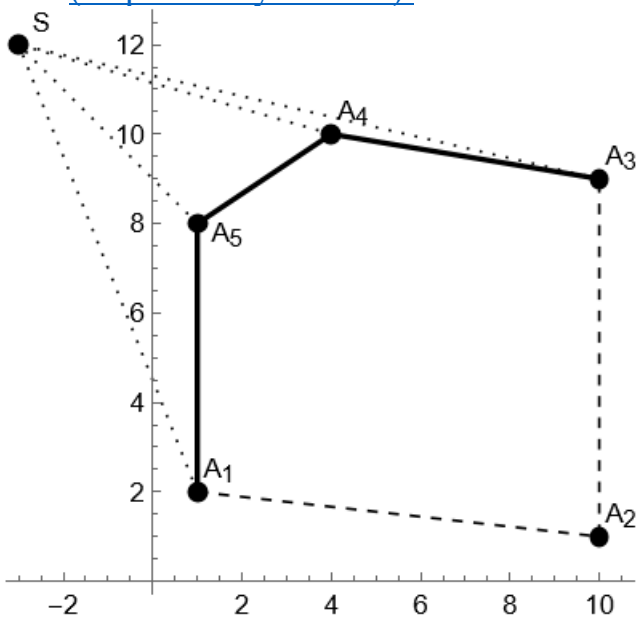
7.1. (Перейти к условию).



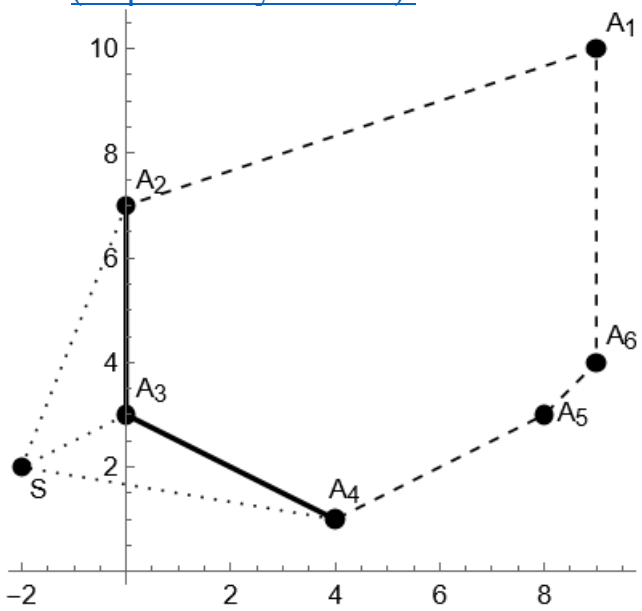
7.2. (Перейти к условию).



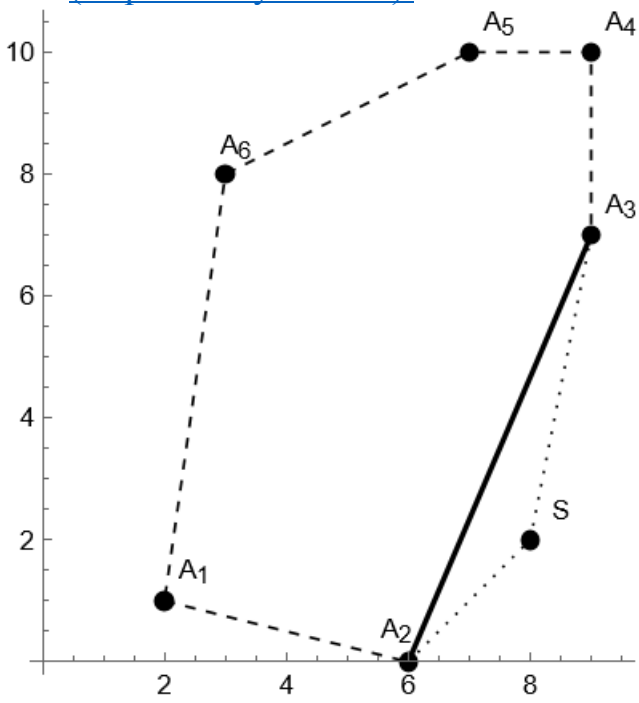
7.3. (Перейти к условию).



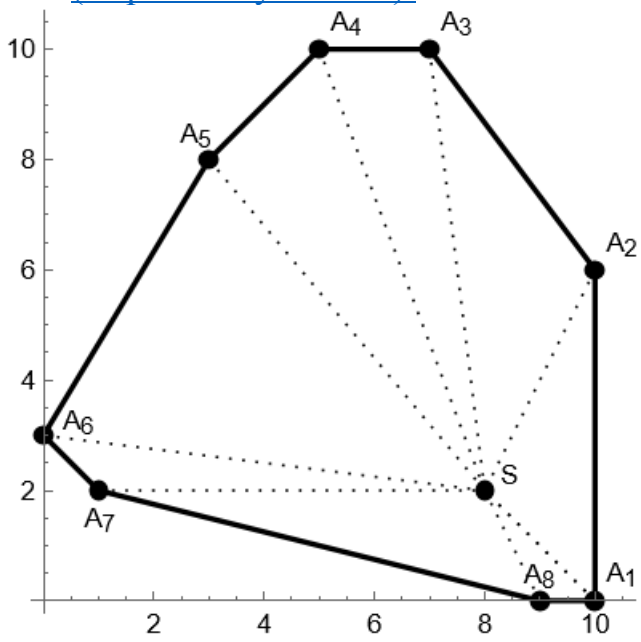
7.4. (Перейти к условию).



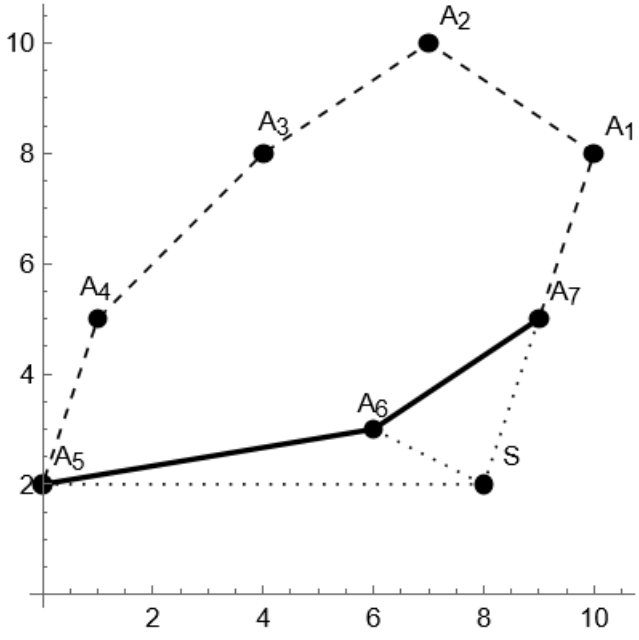
7.5. (Перейти к условию).



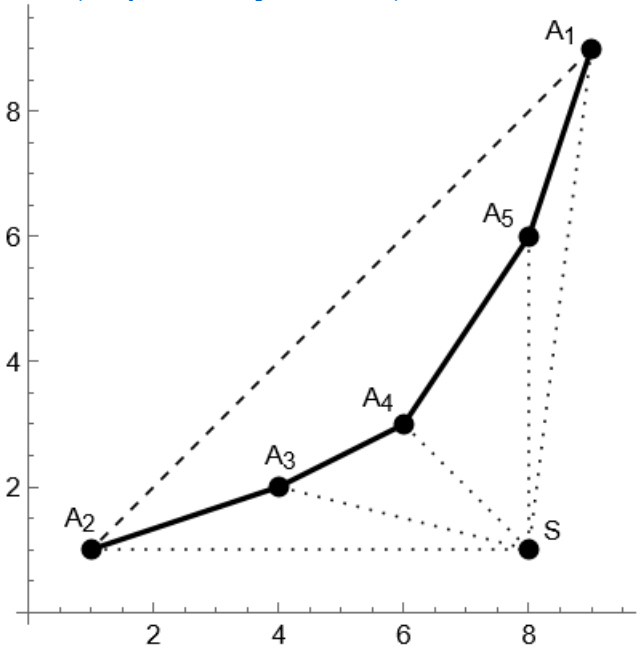
7.6. (Перейти к условию).



7.7. (Перейти к условию).



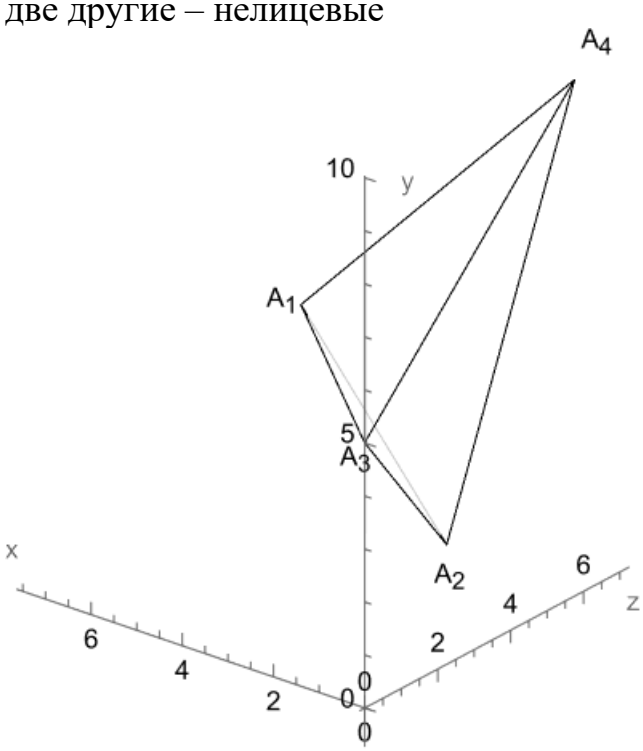
7.8. (Перейти к условию).



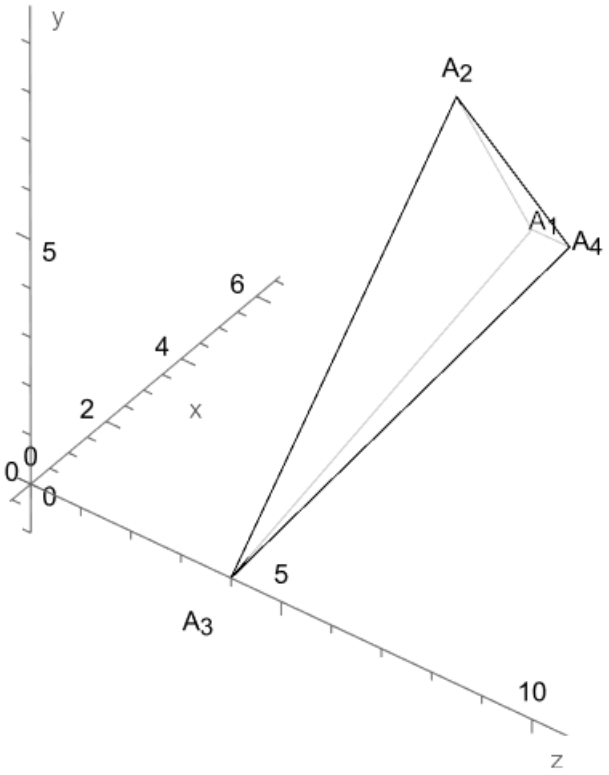
Ответы к задачам подраздела 2.2.

1. В ответах приводятся чертежи, построенные с учётом положения наблюдателя (показано, каким конкретно образом наблюдатель видит заданные полигоны).

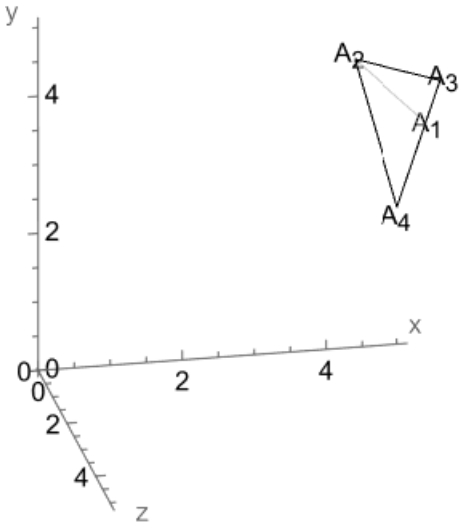
1.1. (Перейти к условию).
Грани $\triangle A_1A_3A_4$ и $\triangle A_2A_3A_4$ – лицевые, две другие – нелицевые



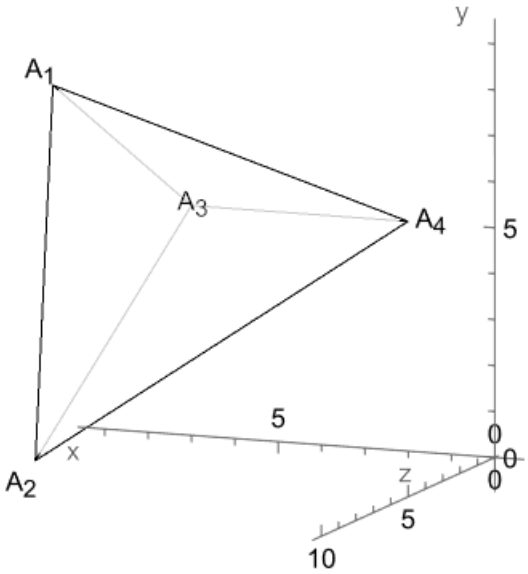
1.2. (Перейти к условию).
Грань $\triangle A_2A_3A_4$ – лицевая, остальные – нелицевые



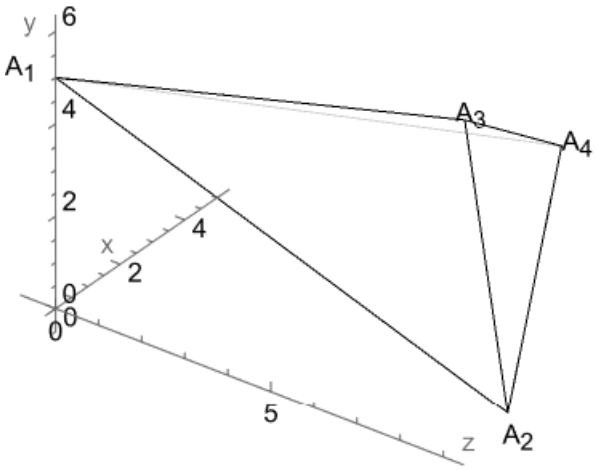
1.3. (Перейти к условию).
Грань $\triangle A_2A_3A_4$ – лицевая, остальные – нелицевые (к тому же грань $\triangle A_1A_3A_4$ вырождается)



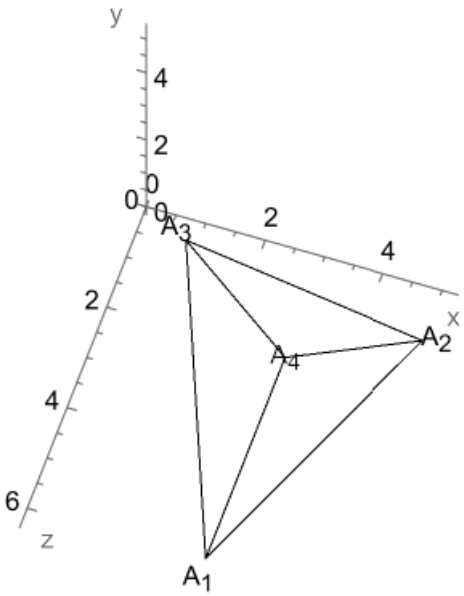
1.4. (Перейти к условию).
Грань $\triangle A_1A_2A_4$ является лицевой, остальные – нелицевые



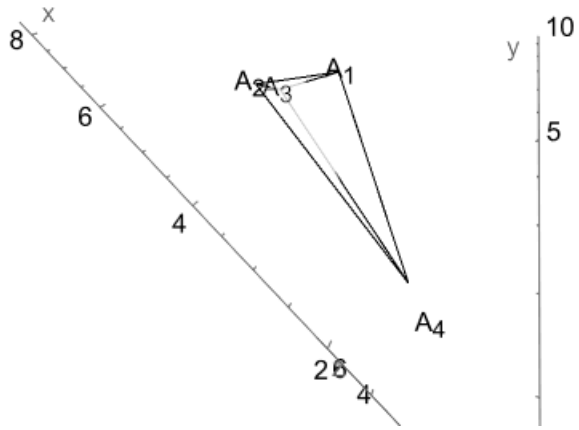
1.5. (Перейти к условию).
Грани $\triangle A_1A_2A_3$ и $\triangle A_2A_3A_4$ – лицевые, две другие – нелицевые



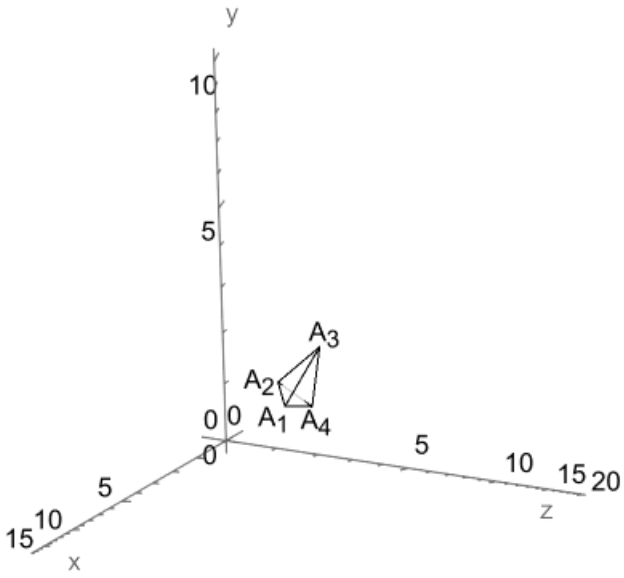
1.6. (Перейти к условию).
Грань $\triangle A_1A_2A_3$ – нелицевая, остальные – лицевые



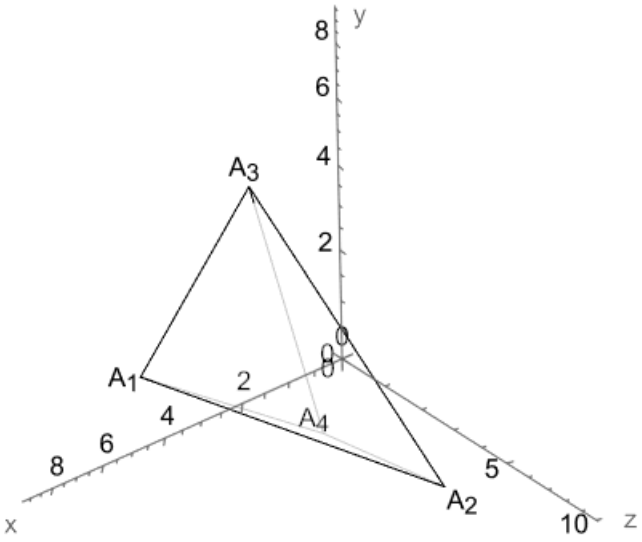
1.7. (Перейти к условию).
Грань $\triangle A_1A_2A_4$ – лицевая, остальные – нелицевые



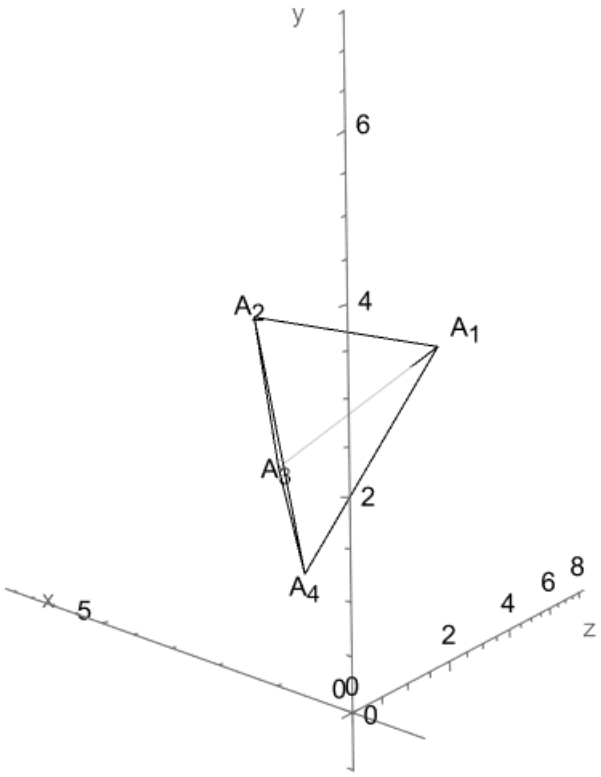
1.8. (Перейти к условию).
Грани $\triangle A_1A_2A_3$ и $\triangle A_1A_3A_4$ – лицевые, две другие – нелицевые



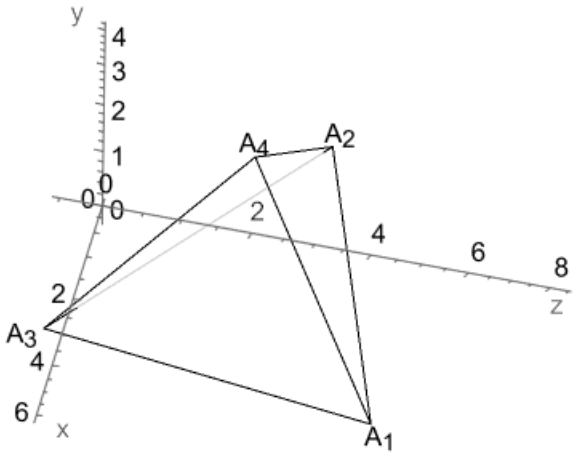
1.9. (Перейти к условию).
Грань $\triangle A_1A_2A_3$ – лицевая, остальные – нелицевые



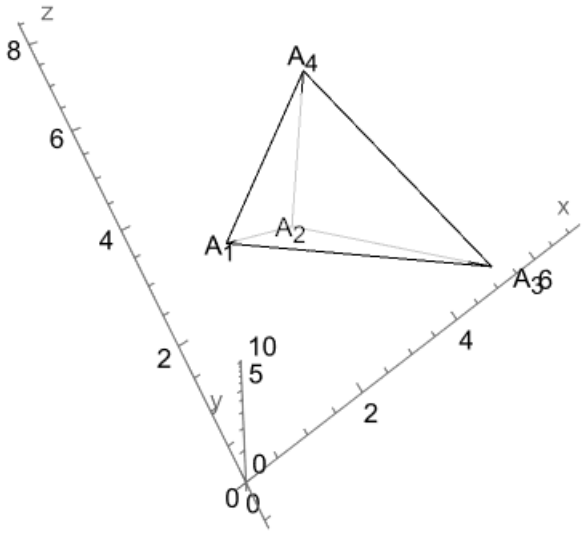
1.10. (Перейти к условию).
Грани $\triangle A_2A_3A_4$ и $\triangle A_1A_2A_4$ – лицевые, две другие – нелицевые



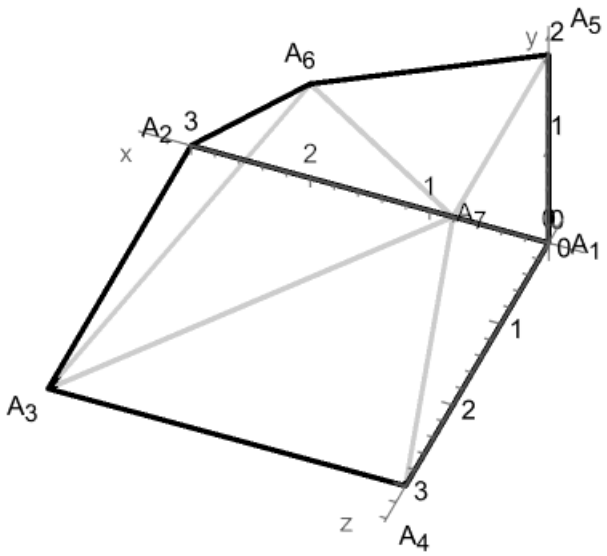
1.11. (Перейти к условию).
Грани $\triangle A_1A_2A_4$ и $\triangle A_1A_3A_4$ – лицевые, две другие – нелицевые



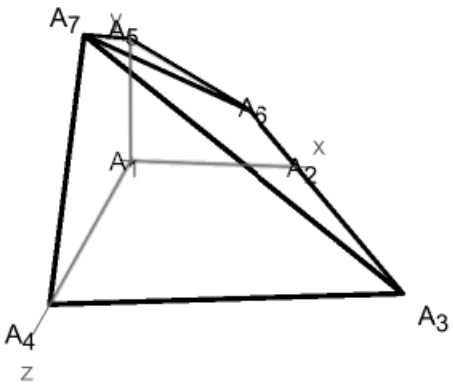
1.12. (Перейти к условию).
Грань $\triangle A_1A_3A_4$ – лицевая, остальные – нелицевые



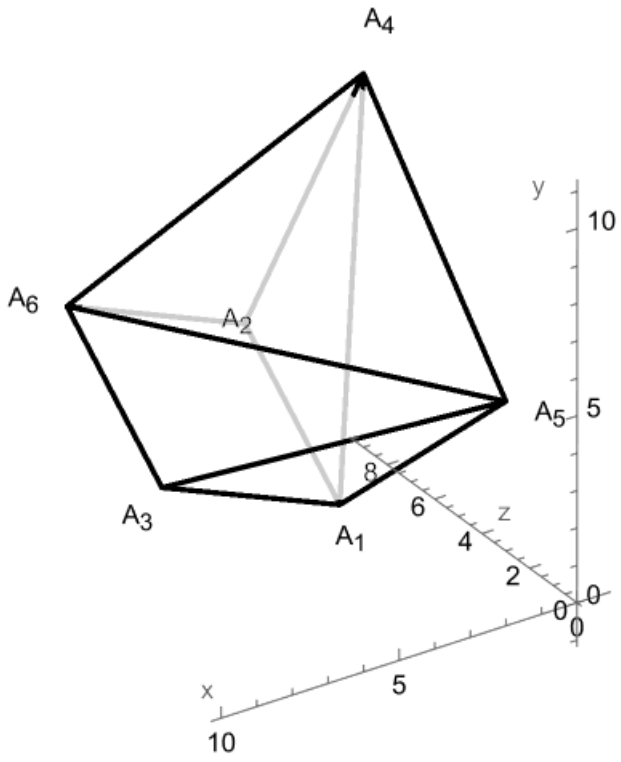
1.13. (Перейти к условию).
Грани $A_1A_5A_6A_2A_1$ и $A_1A_2A_3A_4A_1$ – лицевые, остальные – нелицевые



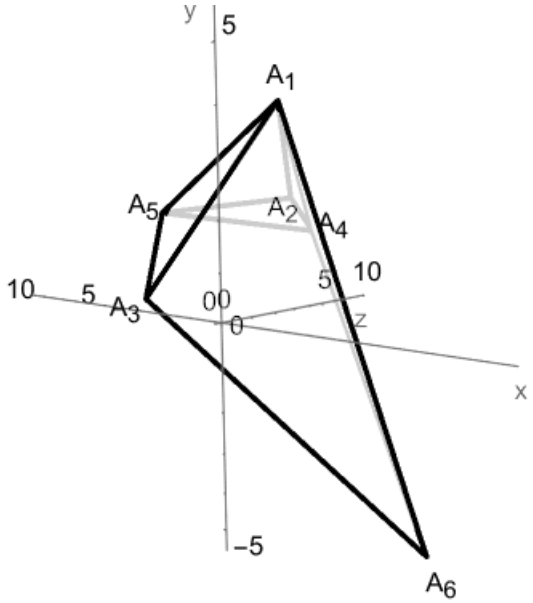
1.14. (Перейти к условию).
Грани $\triangle A_4A_7A_3$, $\triangle A_6A_7A_3$ и $\triangle A_5A_6A_7$ – лицевые, остальные – нелицевые (а грань $\triangle A_2A_6A_3$ к тому же вырождается)



1.15. (Перейти к условию).
Грани $\triangle A_6A_5A_4$, $\triangle A_3A_5A_6$, $\triangle A_5A_3A_1$ – лицевые, остальные – нелицевые



1.16. (Перейти к условию).
Грани $\triangle A_1A_3A_6$ и $\triangle A_3A_1A_5$ – лицевые, остальные – нелицевые



2. (Перейти к условию).

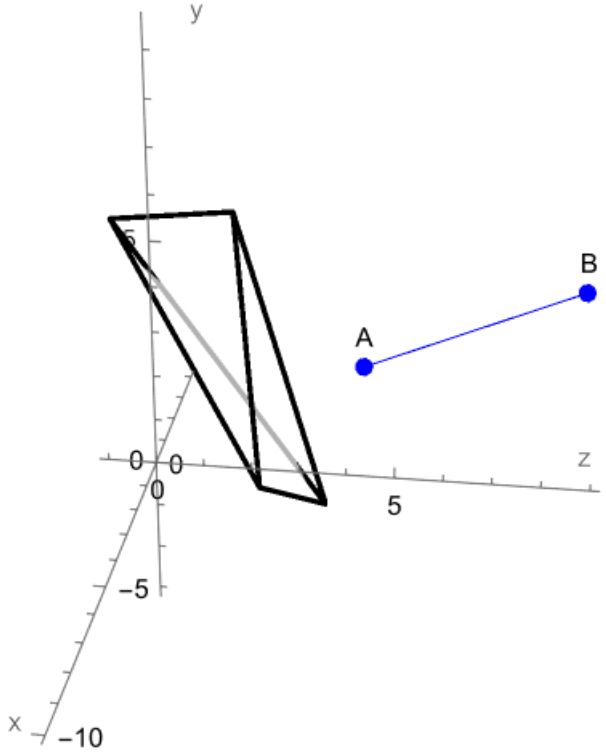
2.1. $\begin{pmatrix} -5 & -3 & -1 & 20 \\ 3 & 1 & 1 & -10 \\ 9 & 5 & 3 & -36 \\ -5 & -1 & -3 & 20 \end{pmatrix}$. 2.2. $\begin{pmatrix} 7 & 1 & -6 & 13 \\ -3 & -1 & 0 & 5 \\ -19 & -13 & 30 & -25 \\ 1 & 1 & 0 & 1 \end{pmatrix}$. 2.3. $\begin{pmatrix} -1 & -1 & 1 & 6 \\ 3 & 2 & -3 & -14 \\ 14 & 11 & 1 & -12 \\ -14 & -6 & 9 & 67 \end{pmatrix}$.

2.4. $\begin{pmatrix} -3 & 8 & 2 & 17 \\ -3 & -9 & 2 & 0 \\ -1 & -3 & 12 & 34 \\ 5 & -2 & -9 & -17 \end{pmatrix}$. 2.5. $\begin{pmatrix} 9 & 10 & -6 & -7 \\ 1 & 0 & -4 & 7 \\ 1 & 5 & -4 & 2 \\ -6 & -10 & 9 & 13 \end{pmatrix}$. 2.6. $\begin{pmatrix} 19 & 15 & -10 & -5 \\ -17 & -13 & 10 & 3 \\ -3 & -3 & 2 & 1 \\ 1 & 1 & -2 & 5 \end{pmatrix}$.

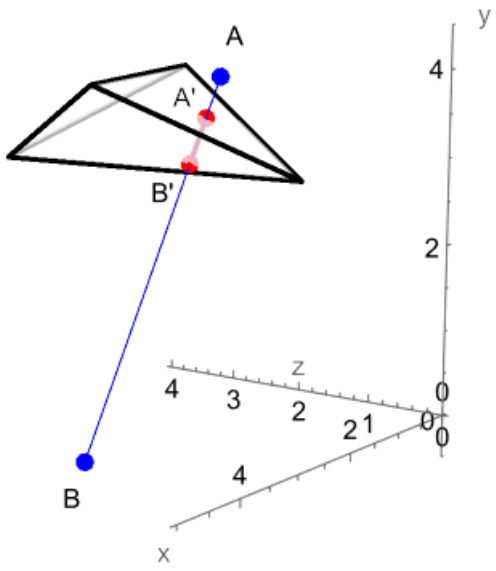
2.7. $\begin{pmatrix} -5 & -8 & -25 & 65 \\ 1 & 0 & 5 & -5 \\ -3 & 8 & 17 & -25 \\ -1 & -2 & -7 & 17 \\ 1 & 0 & -1 & 1 \\ 1 & 2 & -1 & -1 \end{pmatrix}$. 2.8. $\begin{pmatrix} -1 & 2 & -2 & 2 \\ 2 & -1 & 1 & -1 \\ 2 & -1 & -2 & 11 \\ -3 & 2 & 0 & -4 \\ 3 & -2 & 3 & -5 \\ 0 & -1 & 0 & 5 \end{pmatrix}$. 2.9. $\begin{pmatrix} 2 & 0 & 1 & -2 \\ -3 & 0 & 5 & -10 \\ 1 & 5 & 3 & -16 \\ -9 & 12 & 11 & -46 \\ 6 & -8 & -13 & 66 \\ -1 & -1 & 1 & 3 \\ 3 & -4 & -10 & 47 \end{pmatrix}$.

3. На рисунках ниже исходный отрезок AB окрашен в синий цвет, результат отсечения $A'B'$ – в красный.

3.1. (Перейти к условию).
Происходит полное отсечение

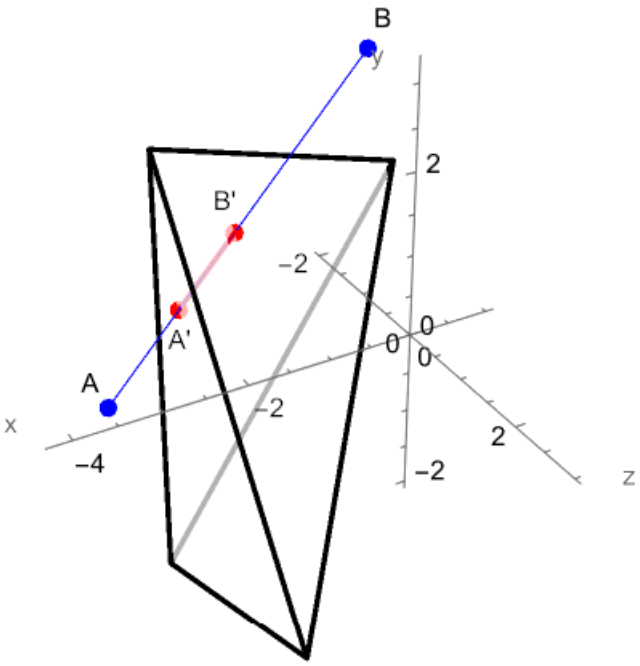


3.2. (Перейти к условию).
$A'(59/19, 68/19, 21/19)$, $B'(29/9, 28/9, 11/9)$. Этим точкам соответствуют значения параметра $2/19$ и $2/9$



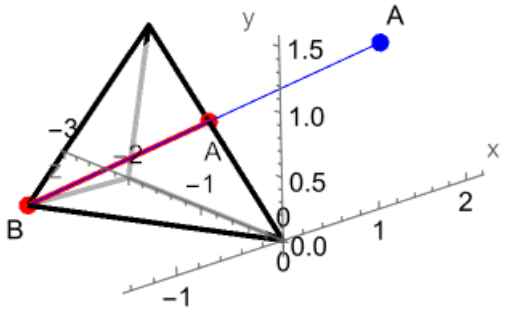
3.3. (Перейти к условию).

$A'(-3,3/2,1/2)$, $B'(-13/6,23/12,1/12)$. Этим точкам соответствуют значения параметра $1/4$ и $11/24$



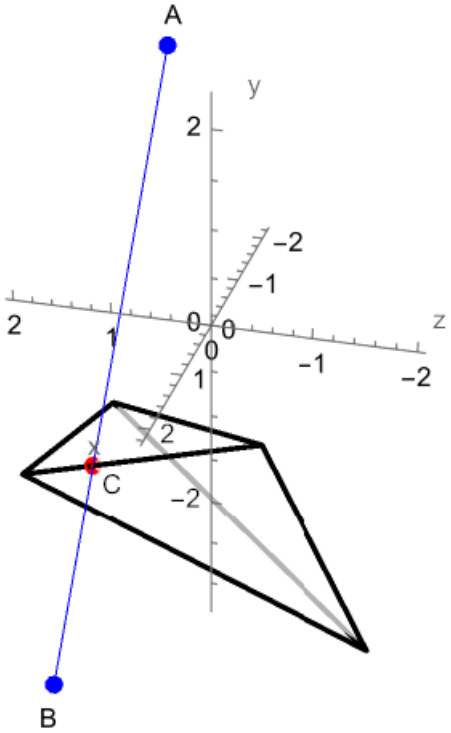
3.4. (Перейти к условию).

$A'(1/2,1/2,-3/2)$, $B' = B(-1,0,-2)$. Этим точкам соответствуют значения параметра $1/2$ и 1



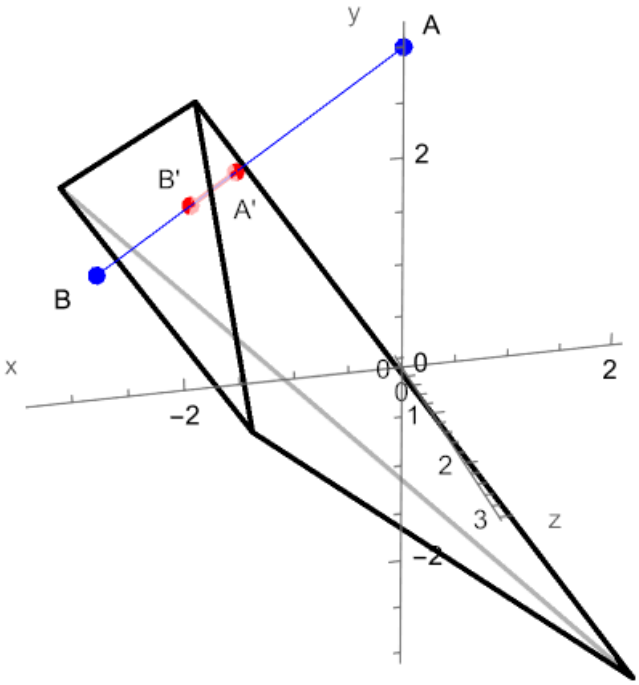
3.5. (Перейти к условию).

Точка $C(2/3,-4/3,1)$ с параметром $t = 2/3$



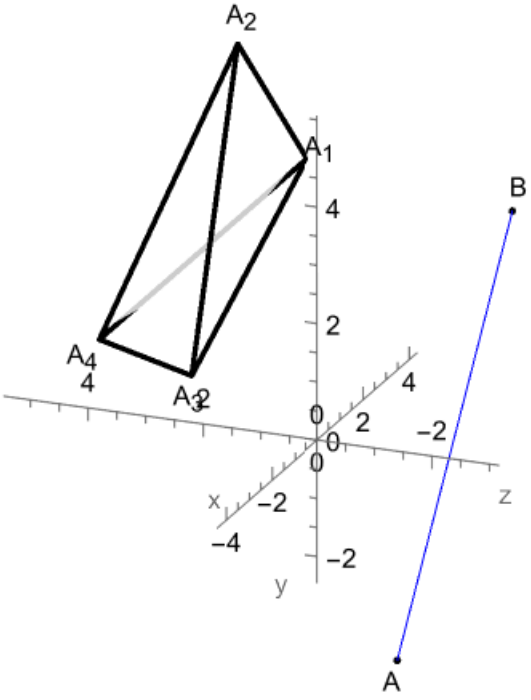
3.6. (Перейти к условию).

$A'(-12/7,17/7,8/7)$, $B'(-15/7,16/7,10/7)$. Этим точкам соответствуют значения параметра $4/7$ и $5/7$

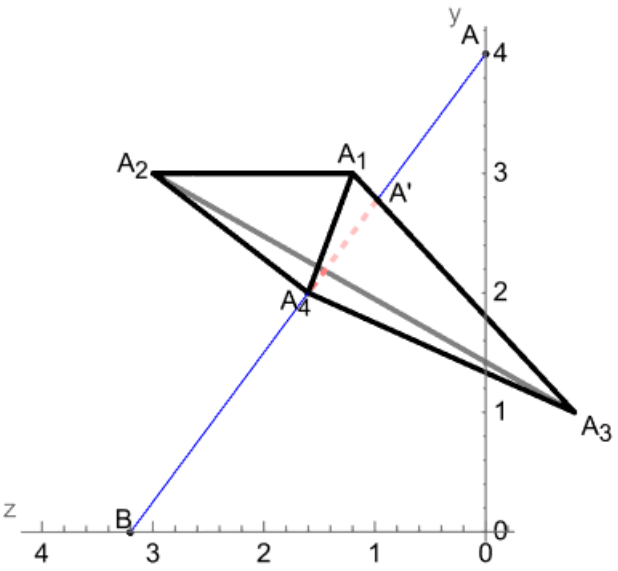


4. На рисунках ниже исходный отрезок окрашен в синий цвет, экранируемая его часть – в красный.

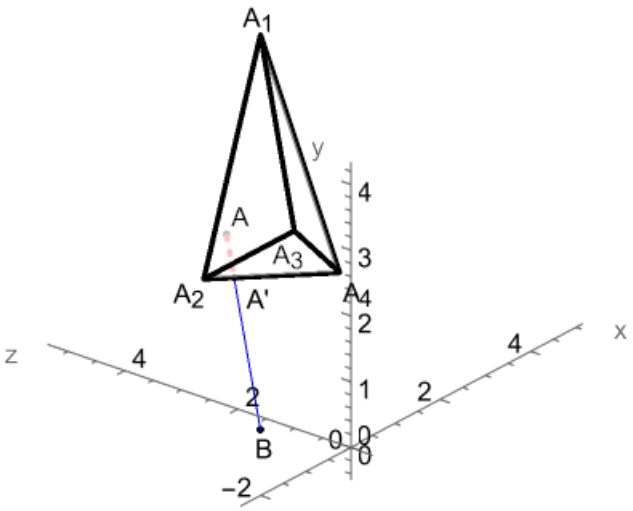
4.1. (Перейти к условию).
Экранирования не происходит



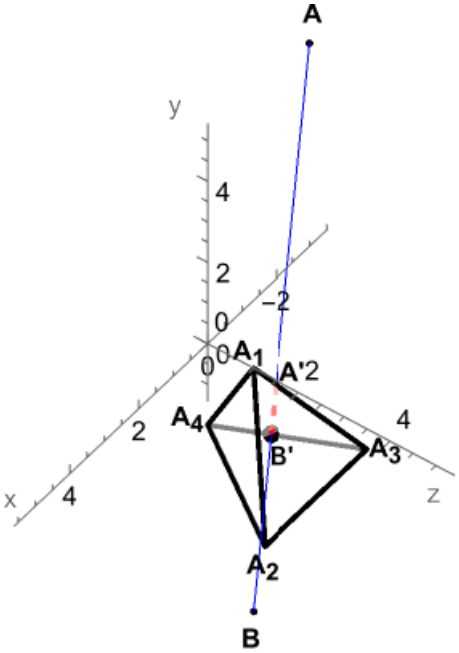
4.2. (Перейти к условию).
Отрезок с концами в точках $A'(11/18, 25/9, 11/9)$ и $A_4(1,2,2)$, которым соответствуют значения параметра $11/36$ и $1/2$



4.3. (Перейти к условию).
Отрезок с концами в точках A и $A'\left(\frac{1}{3}, \frac{16}{9}, \frac{7}{3}\right)$, которым соответствуют значения параметра 0 и $2/9$

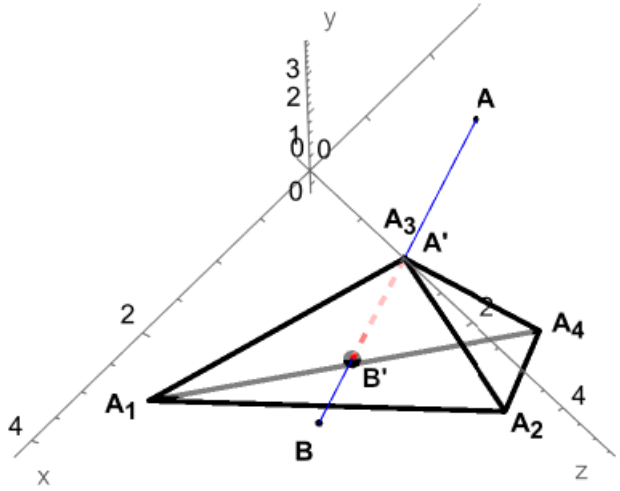


4.4. (Перейти к условию).
Отрезок с концами в точках $A'(6/5, 7/5, 12/5)$ и $B'(29/16, 7/8, 11/4)$, $B' \in_{\triangle} A_1A_2A_3$, которым соответствуют значения параметра $3/5$ и $11/16$



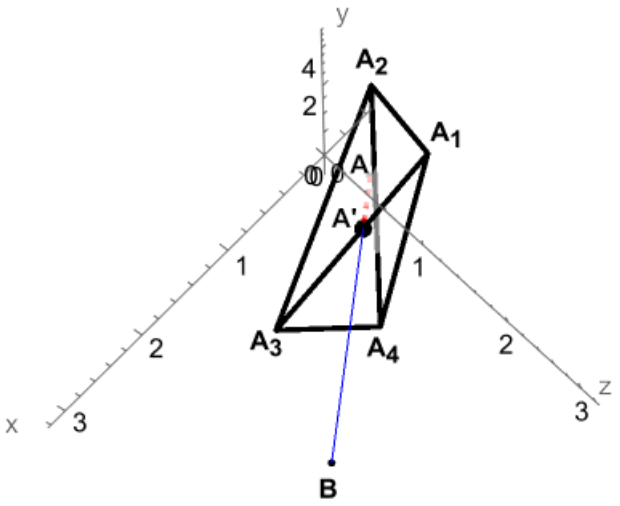
4.5. (Перейти к условию).

Отрезок с концами в точках $A'\left(\frac{1}{2},\frac{3}{2},2\right)$ и $B'\left(\frac{29}{20},\frac{11}{20},2\right)$, $B'\in\triangle A_1A_2A_3$, которым соответствуют значения параметра $\frac{1}{2}$ и $\frac{49}{60}$. На чертеже ниже точки A' и A_3 совмещаются в одну, так как лежат на одной прямой с точкой S.



4.6. (Перейти к условию).

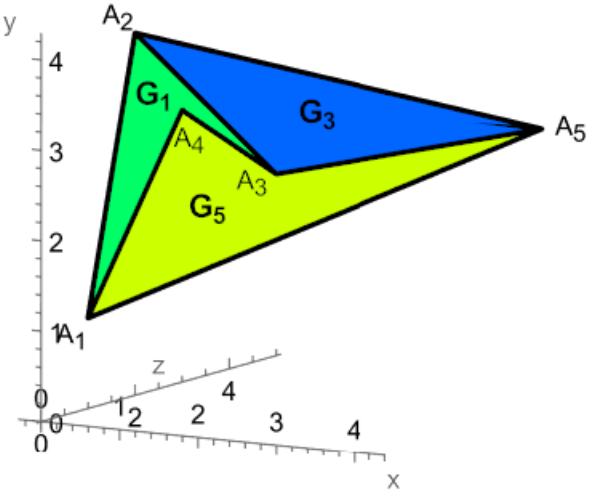
Отрезок с концами в точках A и $A'\left(\frac{9}{7},\frac{25}{7},2\right)$, $A'\in A_1A_3$, которым соответствуют значения параметра 0 и $\frac{2}{7}$



5. В ответах ниже указаны лицевые грани в порядке их отрисовки (по убыванию их глубин)

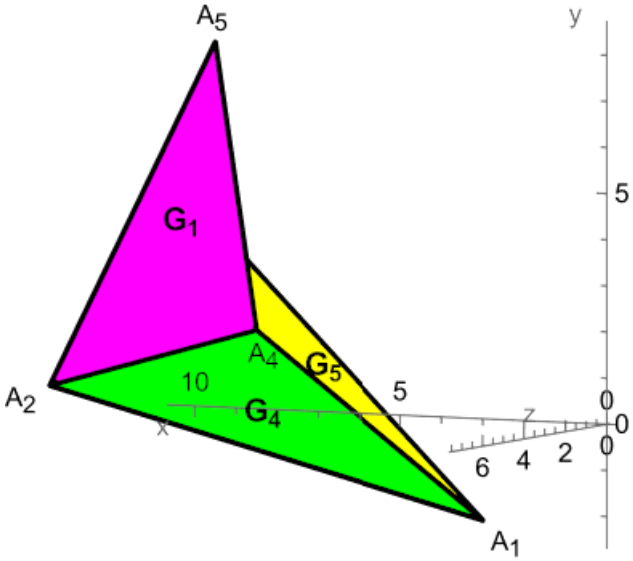
5.1. (Перейти к условию).

G_5, G_1, G_3



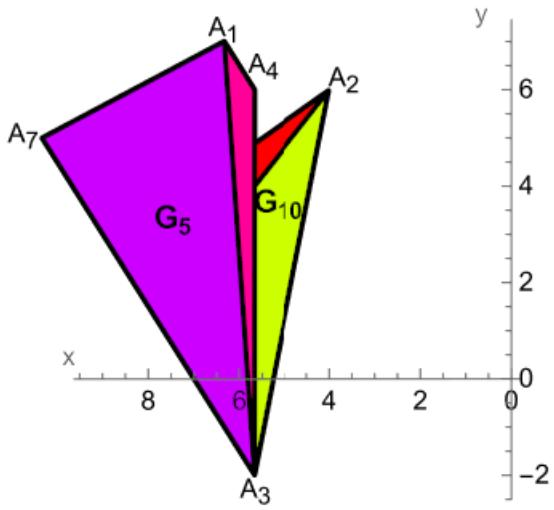
5.2. (Перейти к условию).

G_5, G_4, G_1



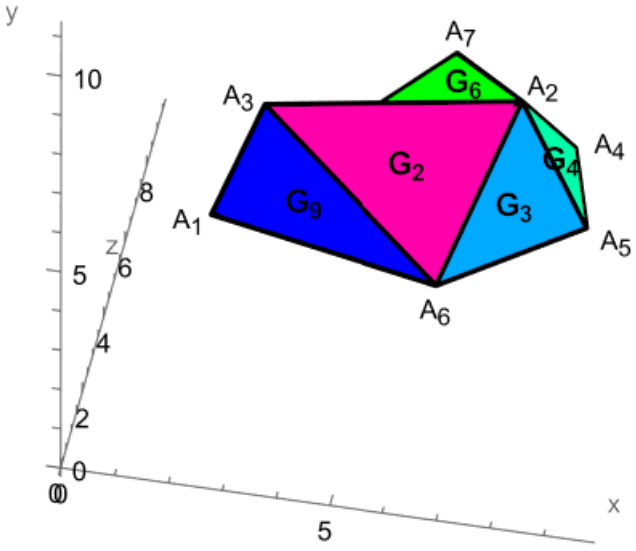
5.3. (Перейти к условию).

G_3 (на чертеже ниже – красная грань), G_{10}, G_5, G_8 (пурпурно-розовая грань)



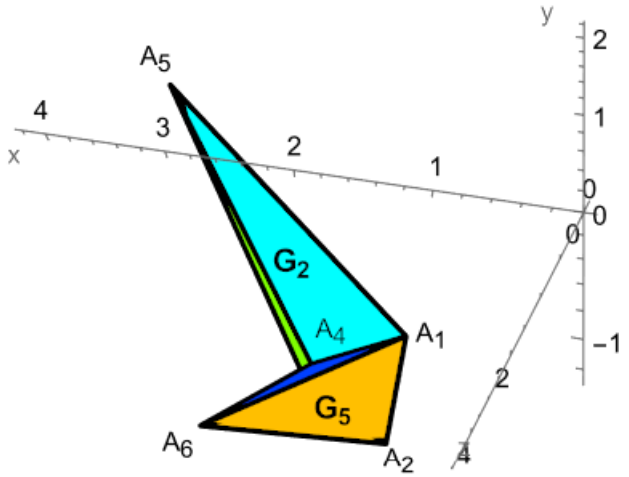
5.4. (Перейти к условию).

G_6, G_4, G_3, G_9, G_2



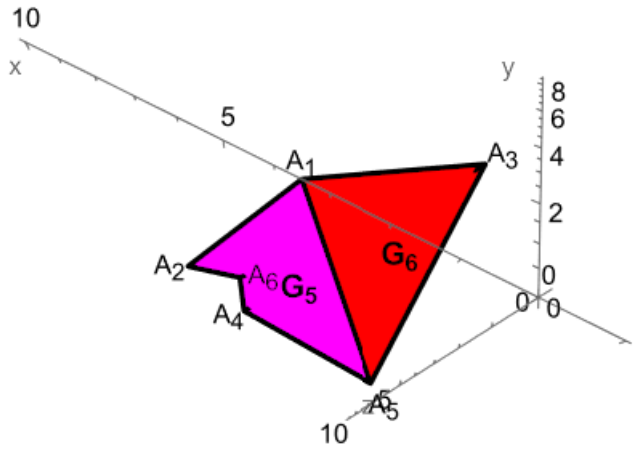
5.5. (Перейти к условию).

G_5, G_3 (зелёная грань), G_2, G_8 (синяя грань)



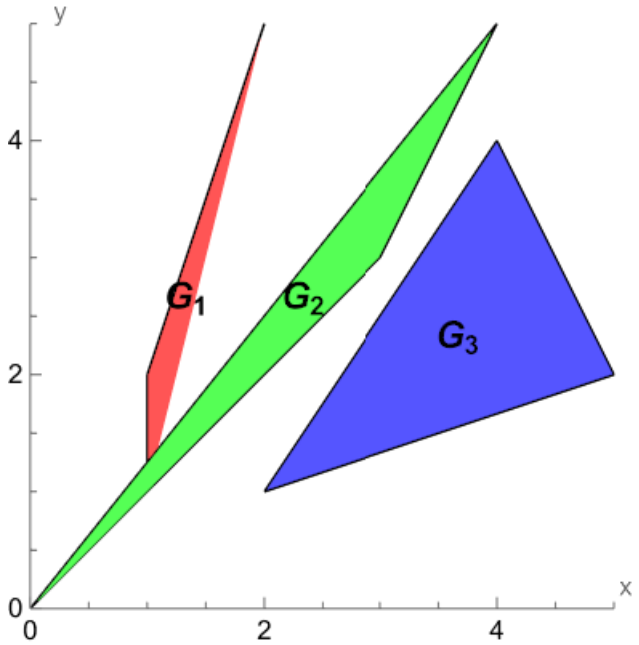
5.6. (Перейти к условию).

G_6, G_5

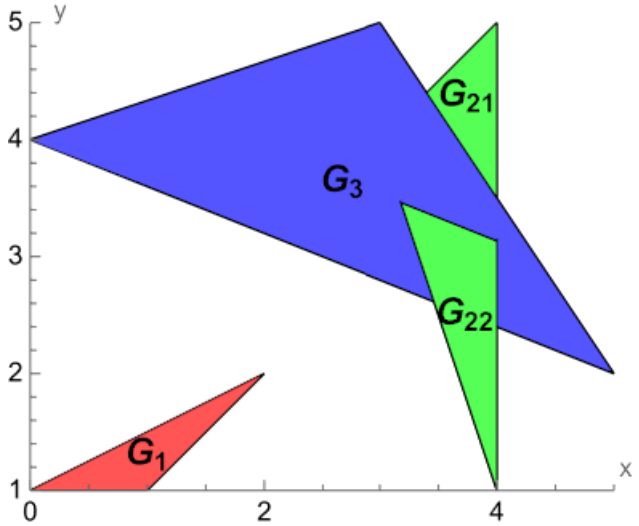


6. В ответах ниже дан порядок отрисовки полигонов и их частей. Части одного и того же полигона окрашены в один цвет.

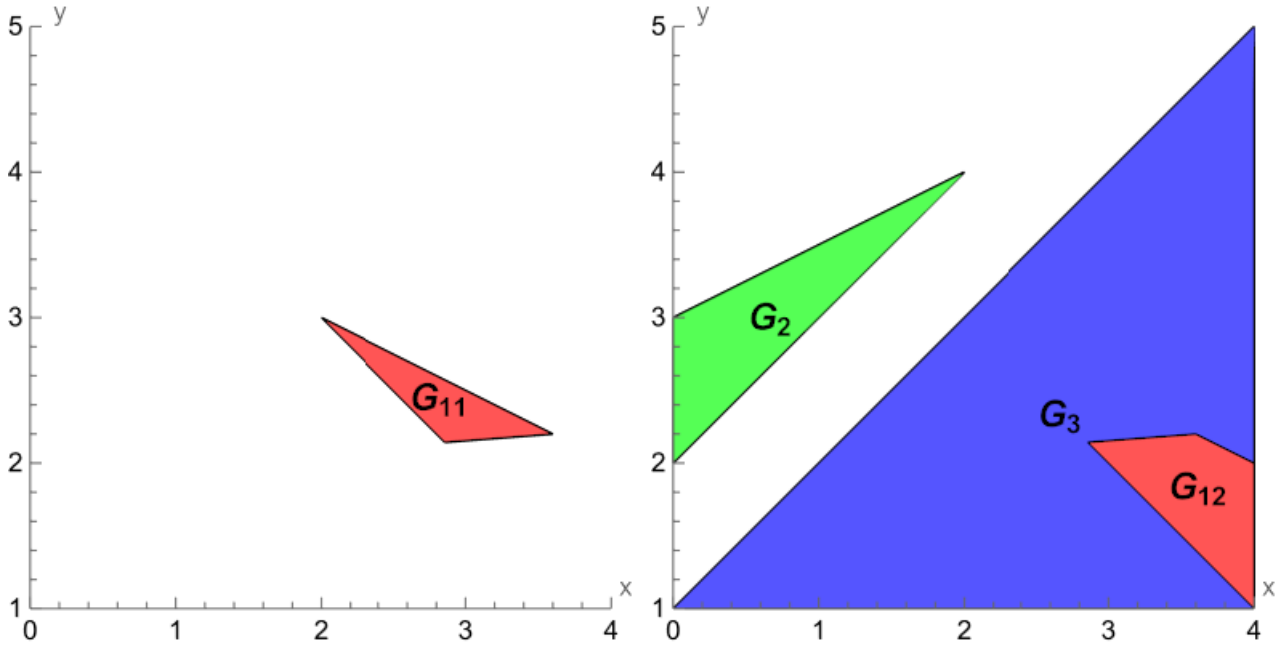
6.1. (Перейти к условию). Разбиений не происходит, и просто отрисовываются полигоны G_1, G_3, G_2.



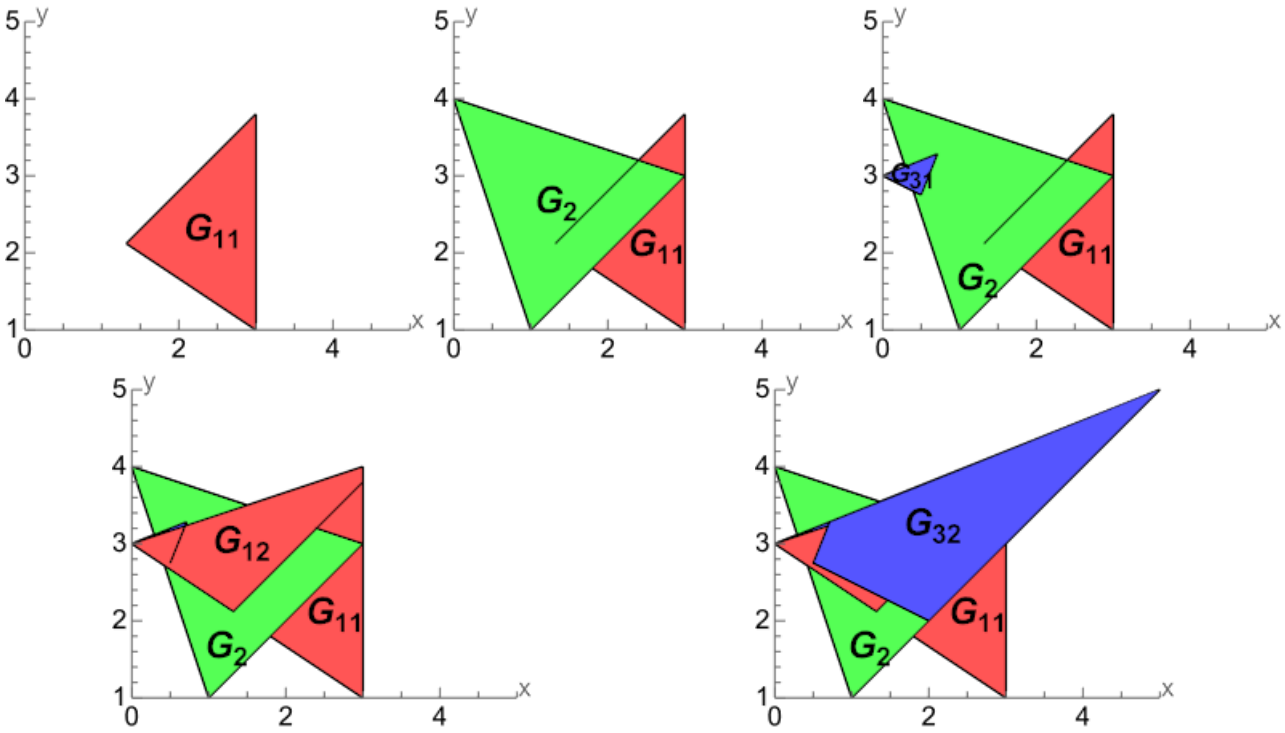
6.2. (Перейти к условию). Происходит разрезание полигона G_2 полигоном G_3 на части $G_{21} = A^1_{21}A_{22}A_{23}A^1_{23}A^1_{23}$, где новые вершины имеют координаты $A^1_{21}\left(4, \frac{47}{15}, 2\right)$ и $A^1_{23}\left(\frac{124}{39}, \frac{45}{13}, \frac{110}{39}\right)$, и $G_{22} = \triangle A_{21}A^1_{21}A^1_{23}$. Порядок рисования полигонов: G_{21}, G_3, G_{22}, G_1.



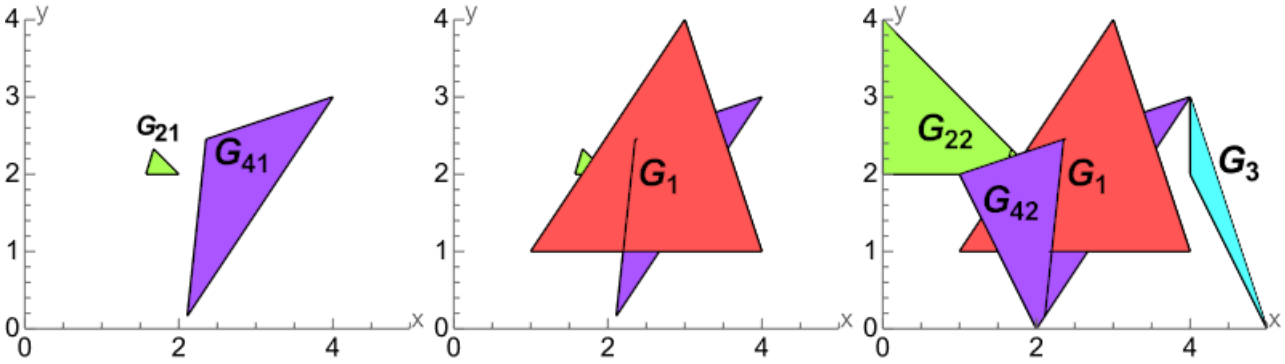
6.3. (Перейти к условию). Происходит разрезание полигона G_1 полигоном G_3 на полигоны $G_{11} = \triangle A_{11}A^1_{11}A^1_{13}$, где новые вершины имеют координаты $A^1_{11}\left(\frac{20}{7}, \frac{15}{7}, \frac{16}{7}\right)$, $A^1_{13}\left(\frac{18}{5}, \frac{11}{5}, \frac{9}{5}\right)$, и $G_{12} = A^1_{11}A_{12}A_{13}A^1_{13}A^1_{11}$. Вначале отрисовывается полигон G_{11}, затем поверх него – G_3, потом G_{12} и G_2.



6.4. (Перейти к условию). Происходит разрезание полигона G_1 полигоном G_2 на части $G_{11} = \triangle A_{11}A_{11}^1A_{13}^1$, где новые вершины имеют координаты $A_{11}^1\left(\frac{33}{25},\frac{53}{25},\frac{67}{25}\right)$, $A_{13}^1\left(3,\frac{19}{5},1\right)$, и $G_{12} = A_{11}^1A_{12}A_{13}A_{13}^1A_{11}^1$; а также полигона G_3 полигоном G_1 на части $G_{31} = \triangle A_{32}^1A_{33}A_{33}^1$ и $G_{32} = A_{31}A_{32}A_{32}^1A_{33}^1A_{31}$ с новыми вершинами $A_{32}^1\left(\frac{1}{2},\frac{11}{4},\frac{7}{2}\right)$ и $A_{33}^1\left(\frac{5}{7},\frac{23}{7},\frac{23}{7}\right)$. Порядок отрисовки: G_{11}, G_2, G_{31}, G_{12}, G_{32}.



6.5. (Перейти к условию). Полигон G_1 делает следующие разрезания: $G_2 = G_{21} \cup G_{22}$, $G_4 = G_{41} \cup G_{42}$, где $G_{21} = \triangle A_{21}^1 A_{22} A_{22}^1$, $G_{22} = A_{21} A_{21}^1 A_{22}^1 A_{23} A_{21}$, $G_{41} = \triangle A_{41} A_{41}^1 A_{43}^1$, $G_{42} = A_{41}^1 A_{42} A_{43} A_{43}^1 A_{41}^1$, а новые вершины имеют координаты $A_{21}^1\left(\frac{52}{33}, 2, \frac{7}{11}\right)$, $A_{22}^1\left(\frac{72}{43}, \frac{100}{43}, \frac{35}{43}\right)$, $A_{41}^1\left(\frac{73}{31}, \frac{76}{31}, \frac{34}{31}\right)$, $A_{43}^1\left(\frac{19}{9}, \frac{1}{6}, 0\right)$. Порядок рисования полигонов: $G_{41}, G_{21}, G_1, G_{22}, G_{42}, G_3$.

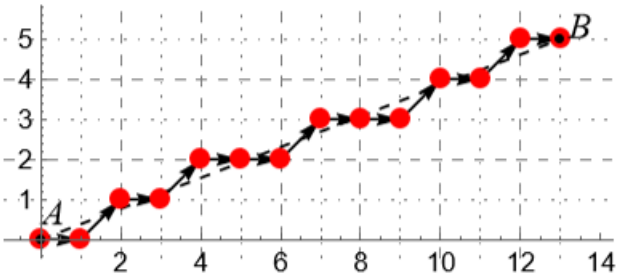


Ответы к задачам подраздела 2.3.

1. В ответах приводятся вначале координаты пикселей, затем код Ротштейна и график. При использовании симметричной реализации алгоритма DDA-линии могут получиться немного другие ответы.

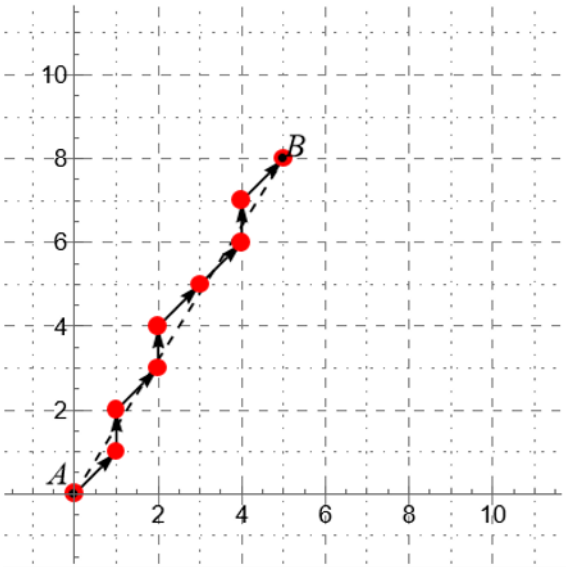
1.1. (Перейти к условию).

```
{{0, 0}, {1, 0}, {2, 1}, {3, 1},
 {4, 2}, {5, 2}, {6, 2}, {7, 3},
 {8, 3}, {9, 3}, {10, 4}, {11, 4},
 {12, 5}, {13, 5}}
"SDSDSSDSSDSDS"
```



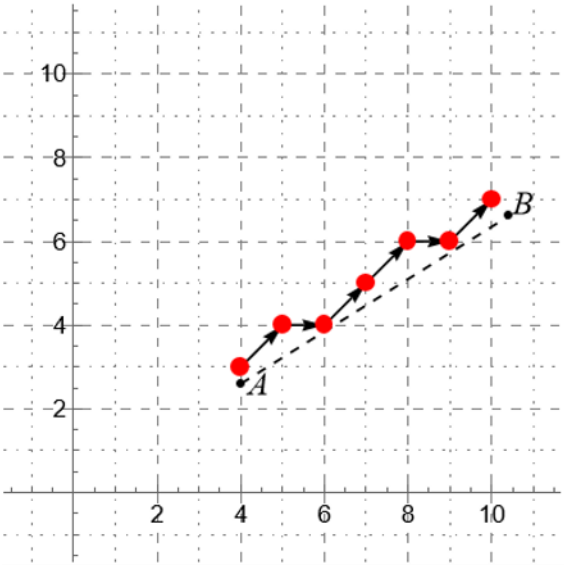
1.2. (Перейти к условию).

```
{{0, 0}, {1, 1}, {1, 2}, {2, 3},
 {2, 4}, {3, 5}, {4, 6}, {4, 7},
 {5, 8}}
"DSDSDDSD"
```



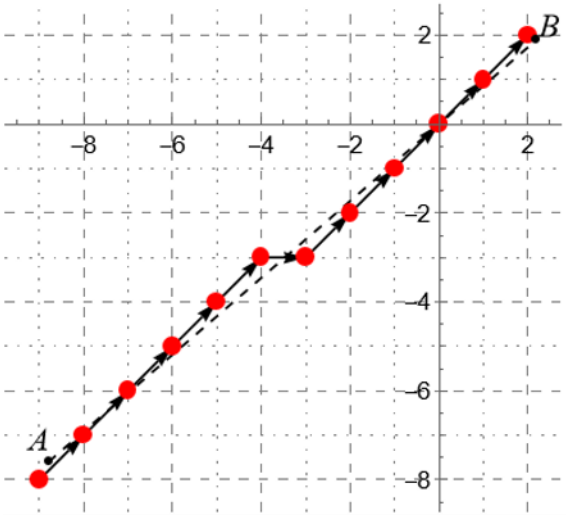
1.3. (Перейти к условию).

```
{{4, 3}, {5, 4}, {6, 4}, {7, 5},
 {8, 6}, {9, 6}, {10, 7}}
"DSDDSD"
```



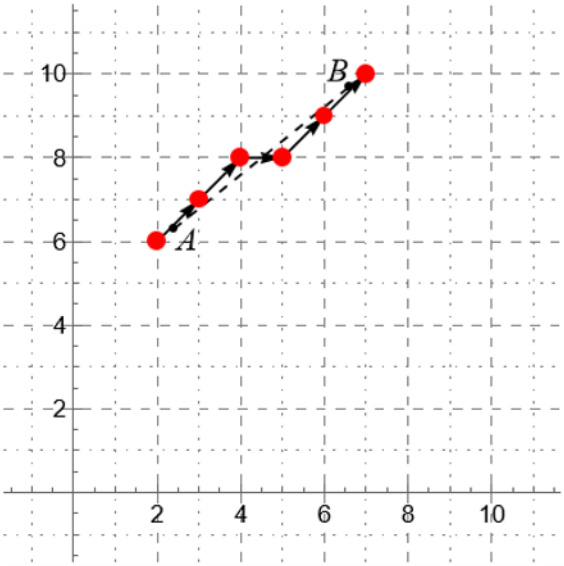
1.4. (Перейти к условию).

```
{{-9, -8}, {-8, -7}, {-7, -6},
 {-6, -5}, {-5, -4}, {-4, -3},
 {-3, -3}, {-2, -2}, {-1, -1},
 {0, 0}, {1, 1}, {2, 2}}
"DDDDDSDDDDD"
```



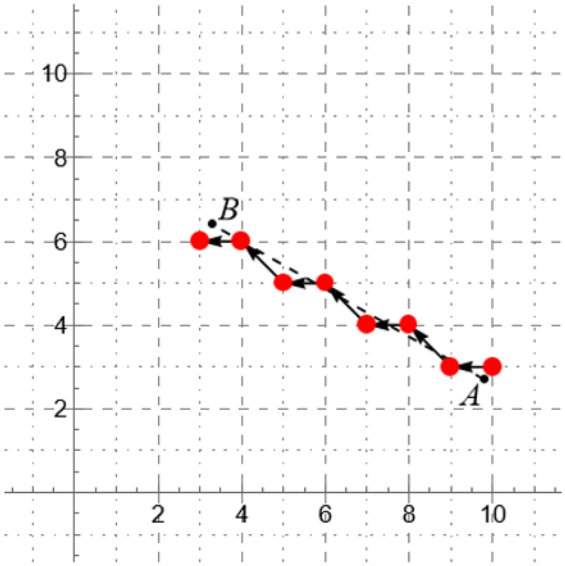
1.5. (Перейти к условию).

```
{{2, 6}, {3, 7}, {4, 8}, {5, 8},
 {6, 9}, {7, 10}}
"DDSDD"
```



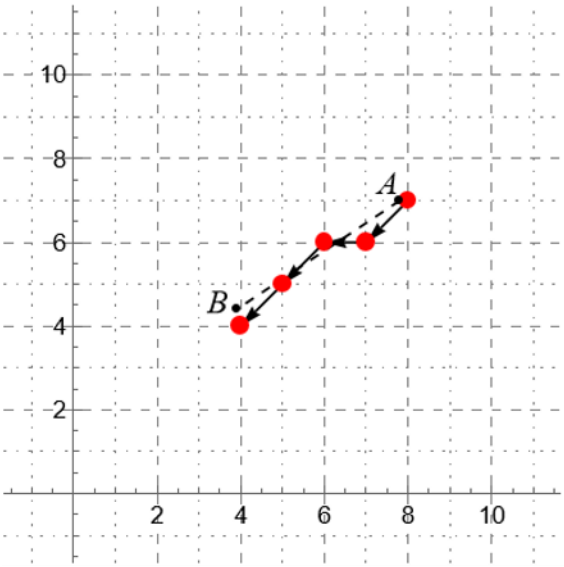
1.6. (Перейти к условию).

```
{{10, 3}, {9, 3}, {8, 4}, {7, 4},
 {6, 5}, {5, 5}, {4, 6}, {3, 6}}
"SDSDSDS"
```



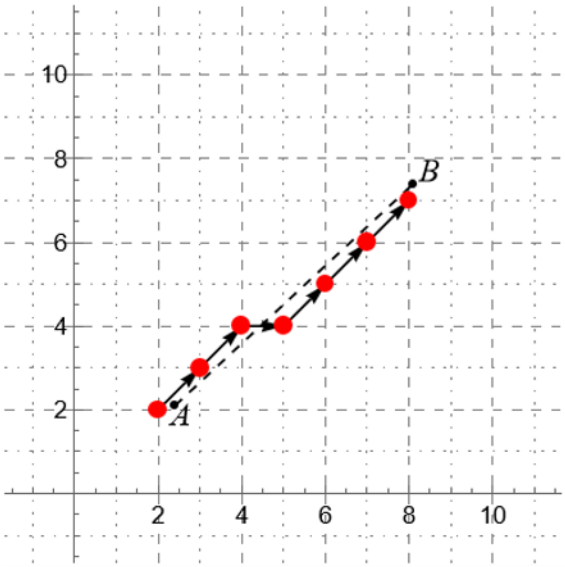
1.7. (Перейти к условию).

```
{{8, 7}, {7, 6}, {6, 6}, {5, 5},
 {4, 4}}
"DSDD"
```



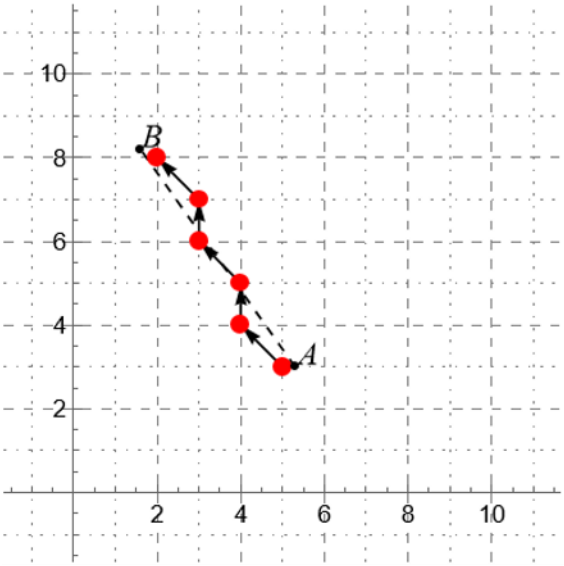
1.8. (Перейти к условию).

```
{{2, 2}, {3, 3}, {4, 4}, {5, 4},
 {6, 5}, {7, 6}, {8, 7}}
"DDSDDD"
```



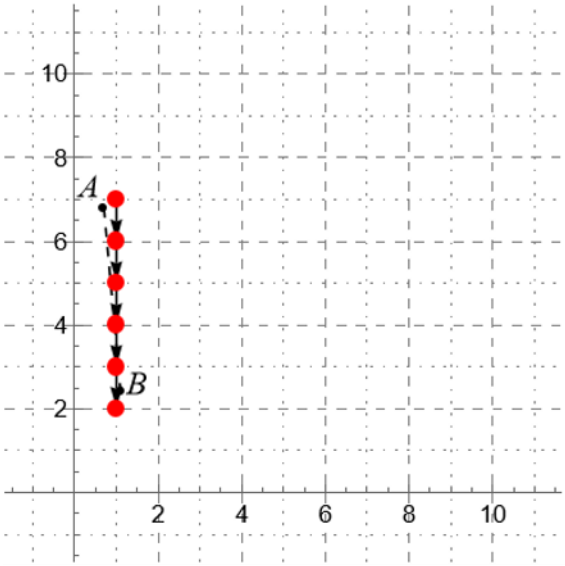
1.9. (Перейти к условию).

```
{{5, 3}, {4, 4}, {4, 5}, {3, 6},
 {3, 7}, {2, 8}}
"DSDSD"
```



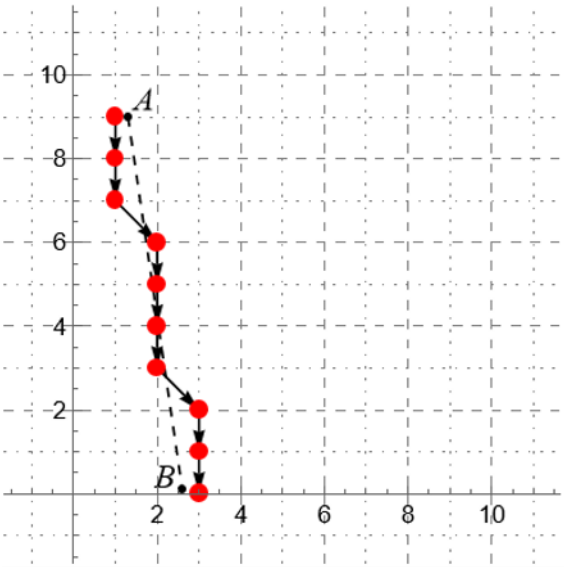
1.10. (Перейти к условию).

```
{{1, 7}, {1, 6}, {1, 5}, {1, 4},
 {1, 3}, {1, 2}}
"SSSSS"
```



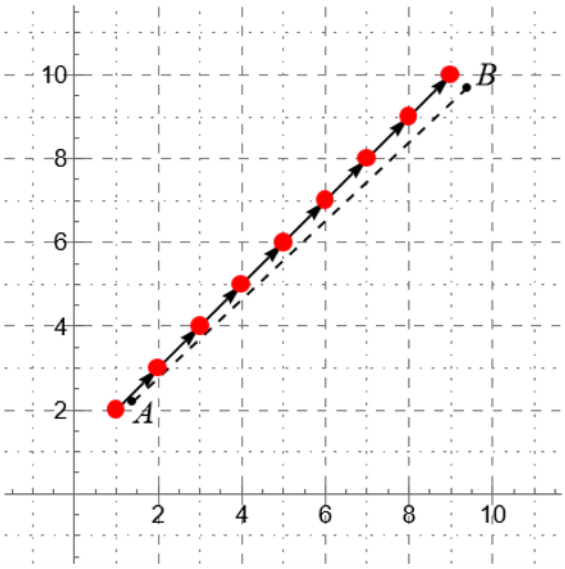
1.11. (Перейти к условию).

```
{{1, 9}, {1, 8}, {1, 7}, {2, 6},
 {2, 5}, {2, 4}, {2, 3}, {3, 2},
 {3, 1}, {3, 0}}
"SSDSSSDSS"
```



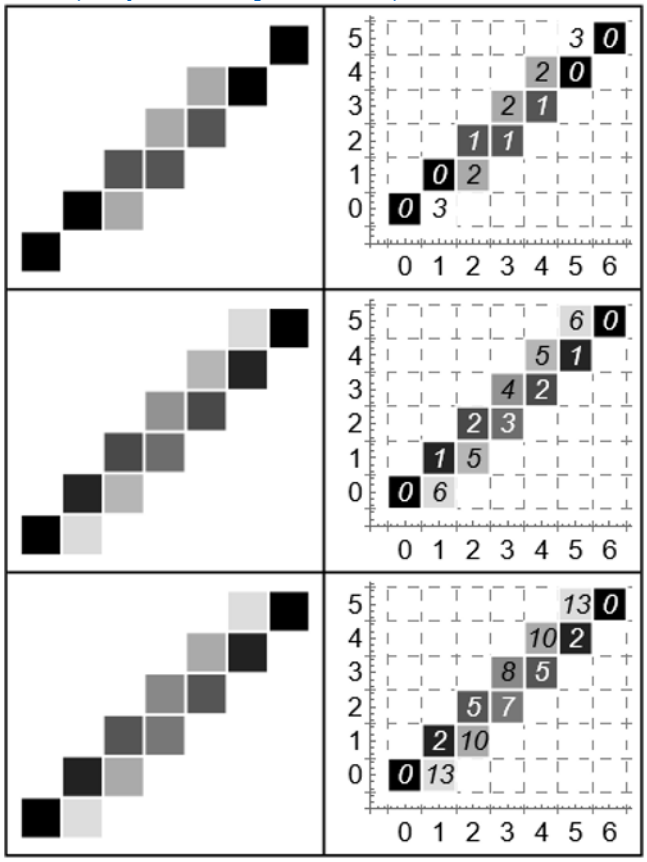
1.12. (Перейти к условию).

```
{{1, 2}, {2, 3}, {3, 4}, {4, 5},
 {5, 6}, {6, 7}, {7, 8}, {8, 9},
 {9, 10}}
"DDDDDDDD"
```

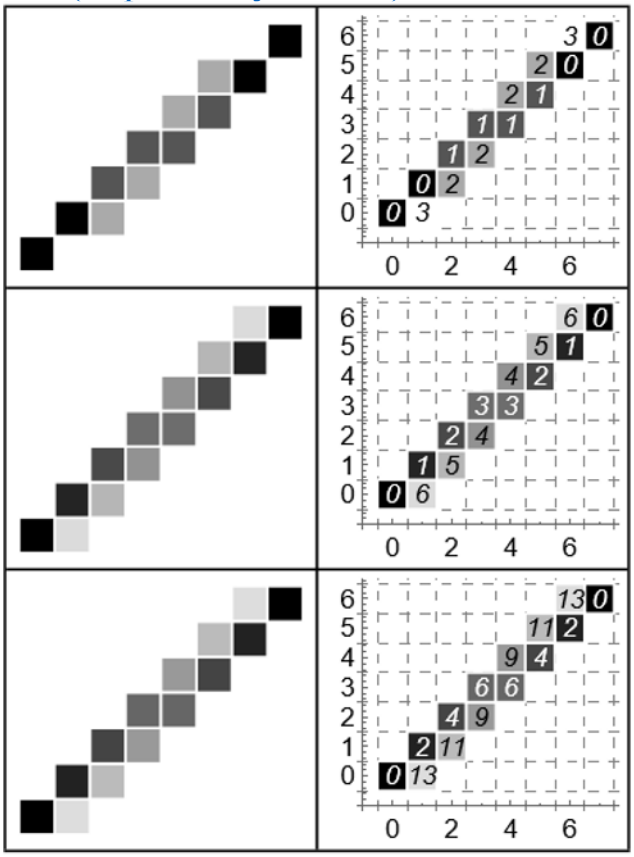


2. Для каждого примера представлены ответы для 4, 8 и 16 оттенков серого. В левой колонке каждой таблицы для сравнения приведены сами итоговые отрезки, в правой колонке – значения для оттенков серого, соответствующие тому или иному пикселю.

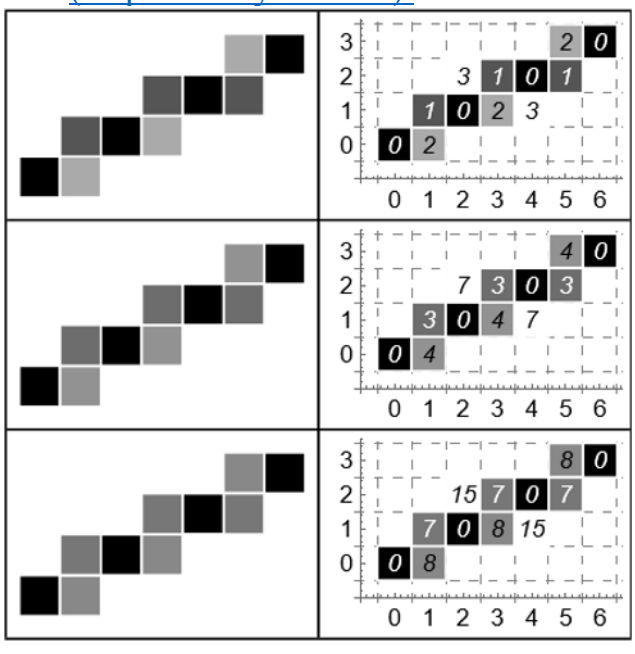
2.1. (Перейти к условию).



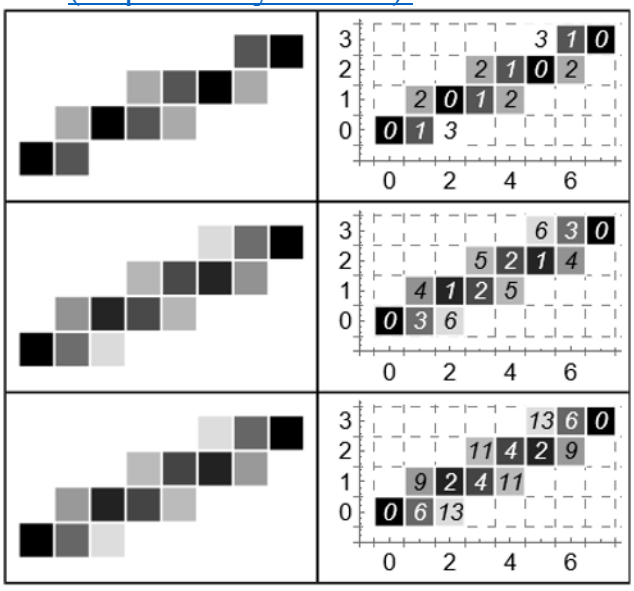
2.2. (Перейти к условию).



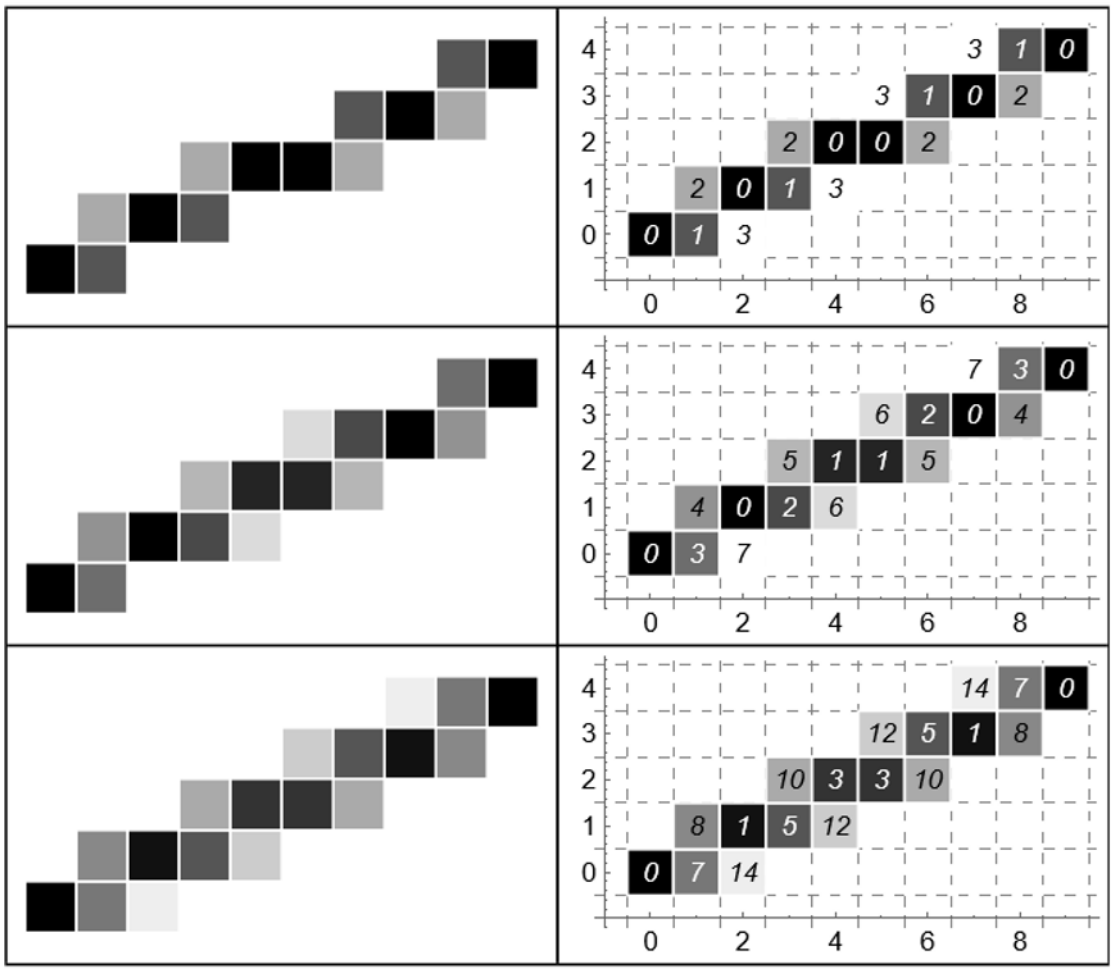
2.3. (Перейти к условию).



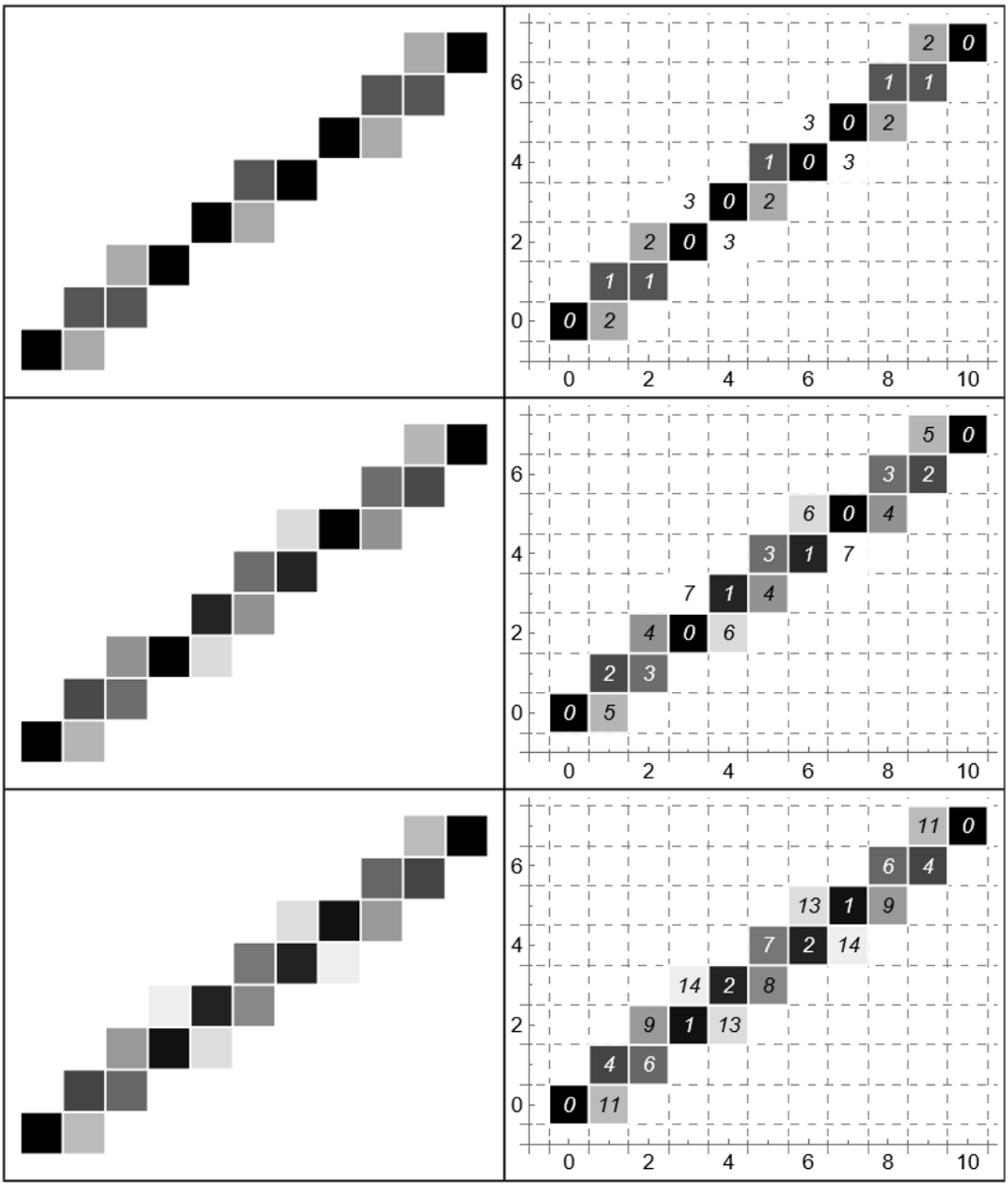
2.4. (Перейти к условию).



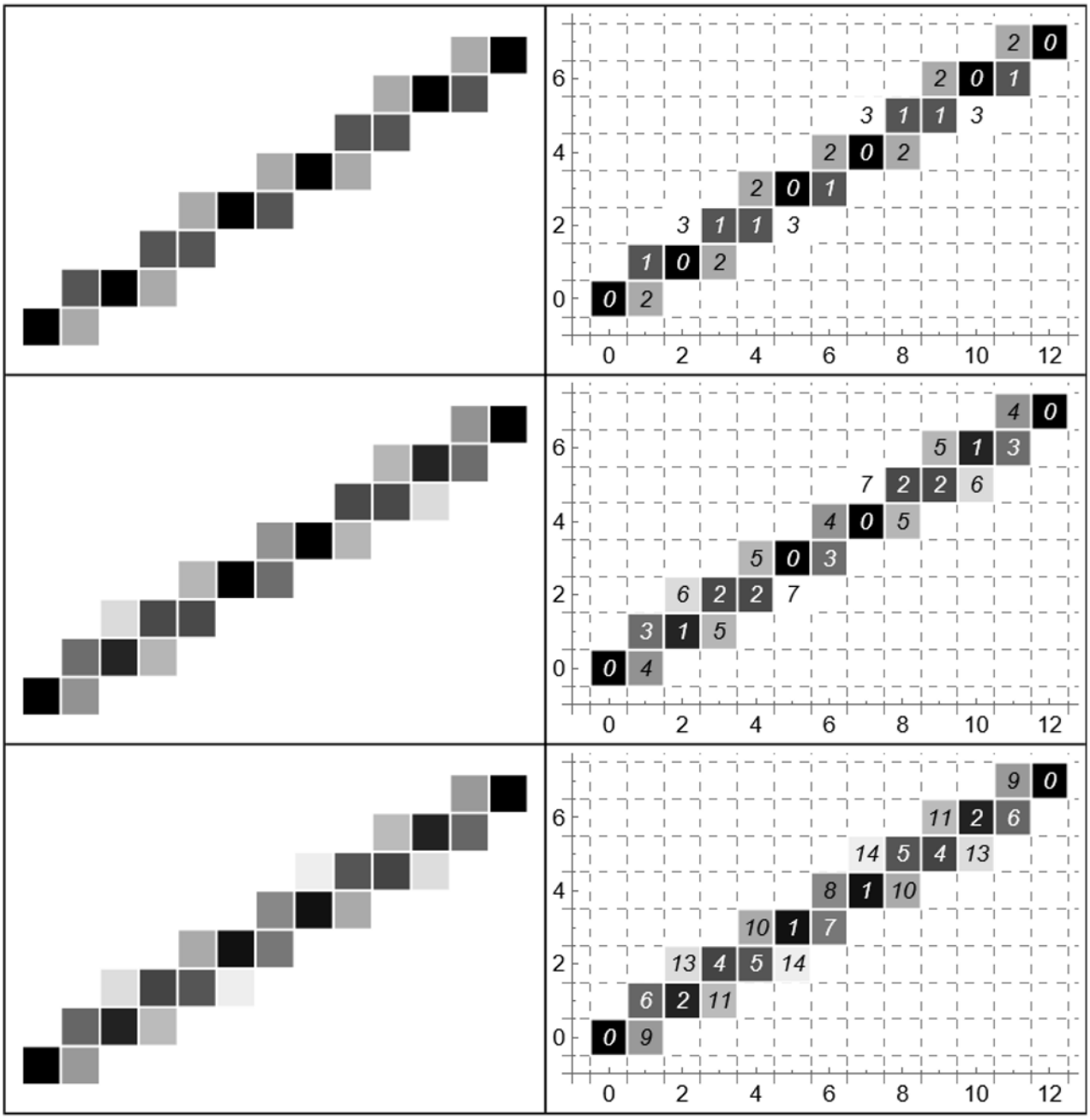
2.5. (Перейти к условию).



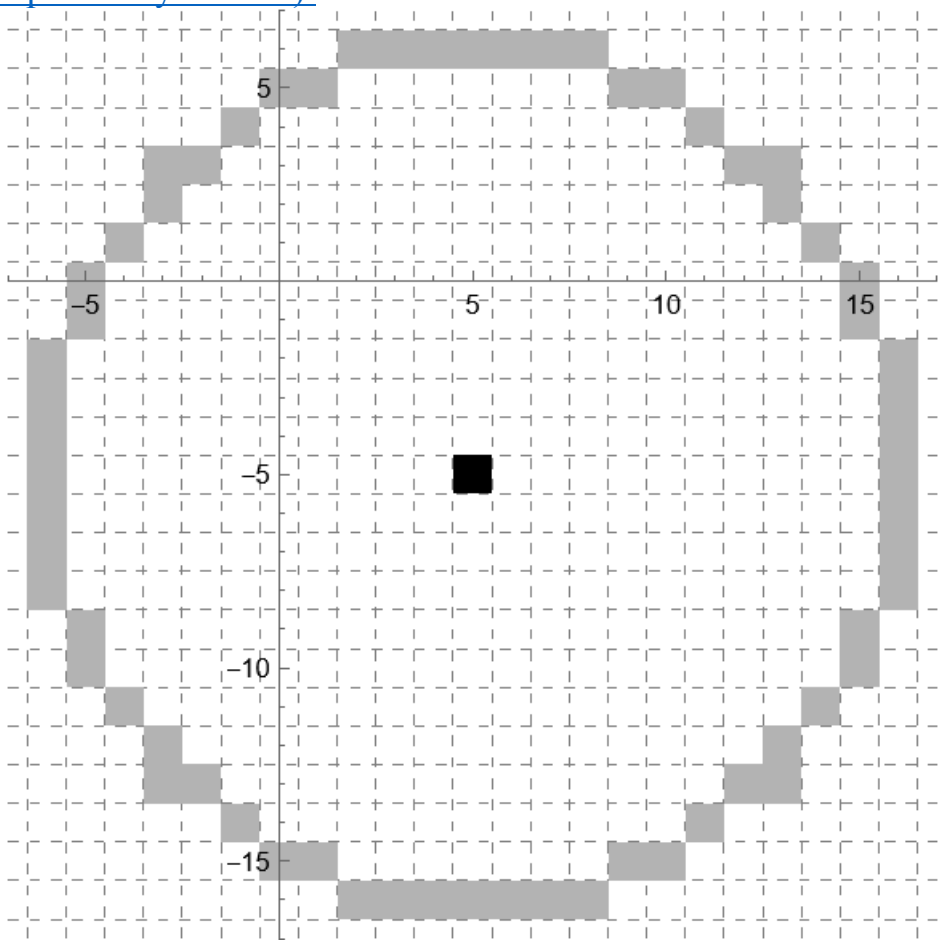
2.6. (Перейти к условию).



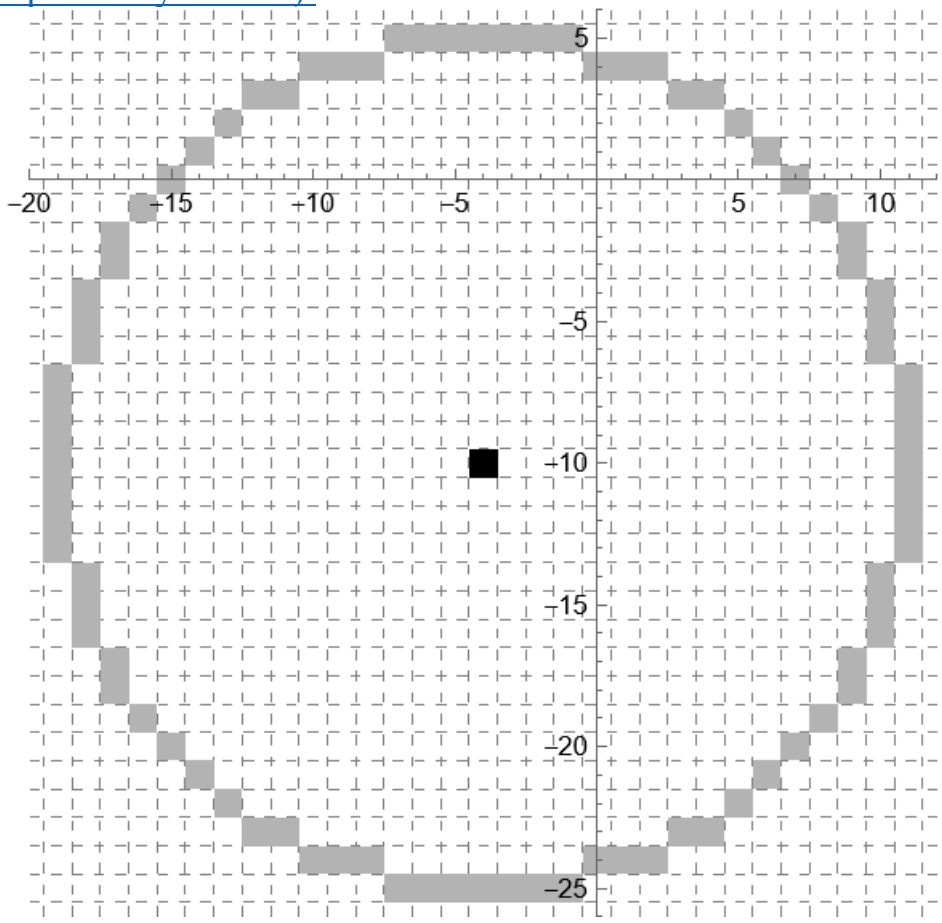
2.7. (Перейти к условию).



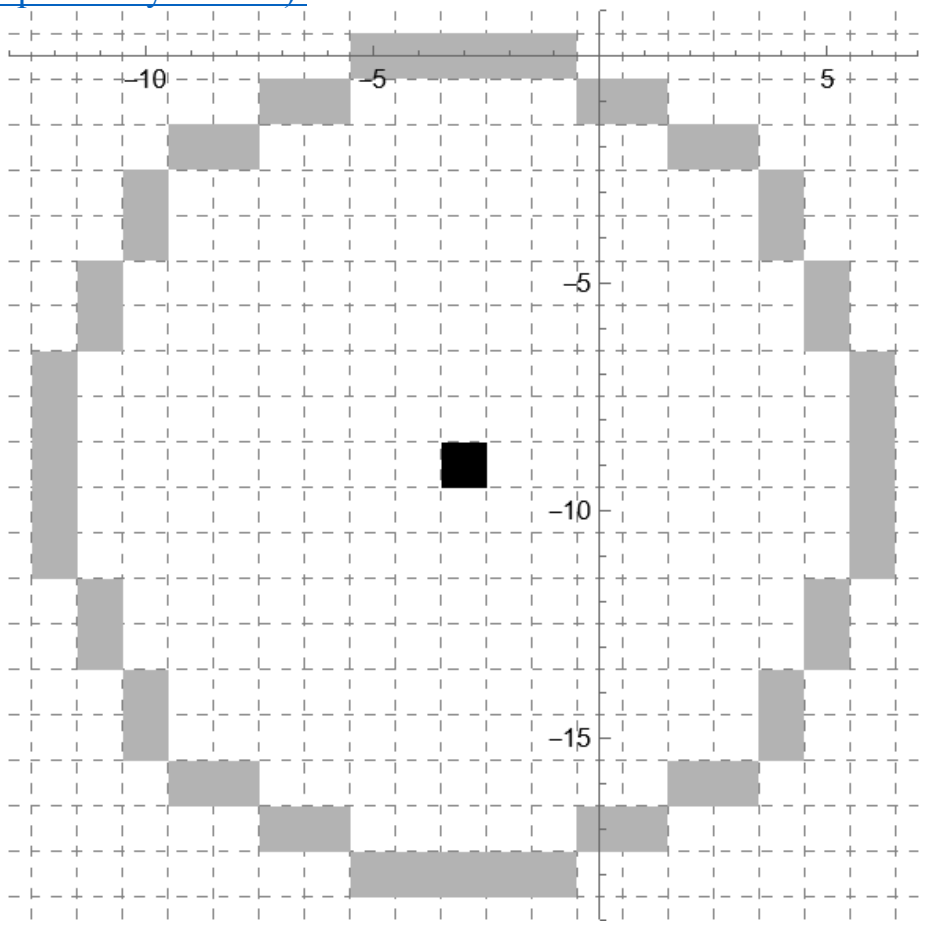
3.1. (Перейти к условию).



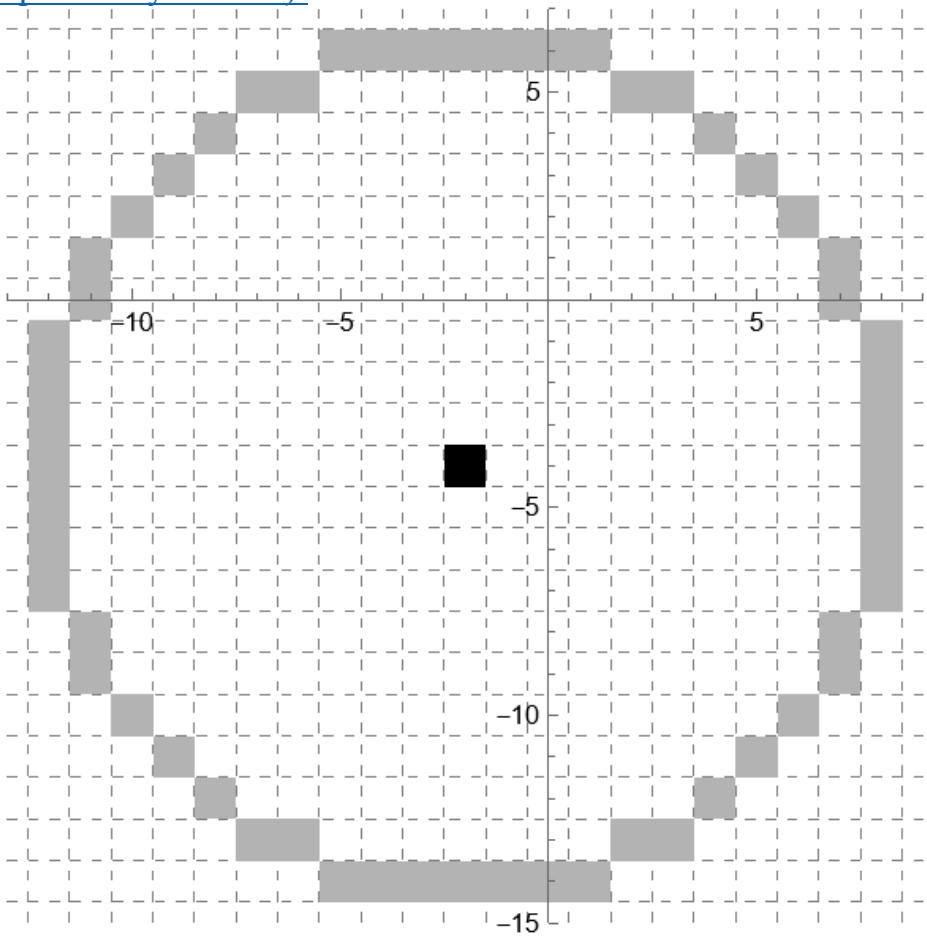
3.2. (Перейти к условию).



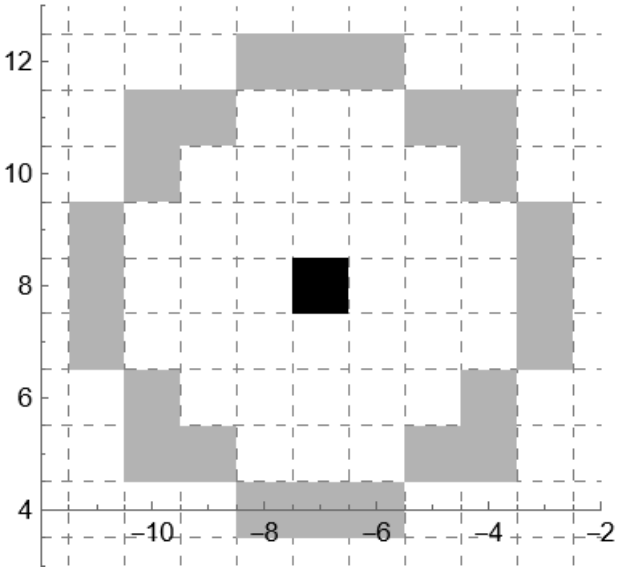
3.3. (Перейти к условию).



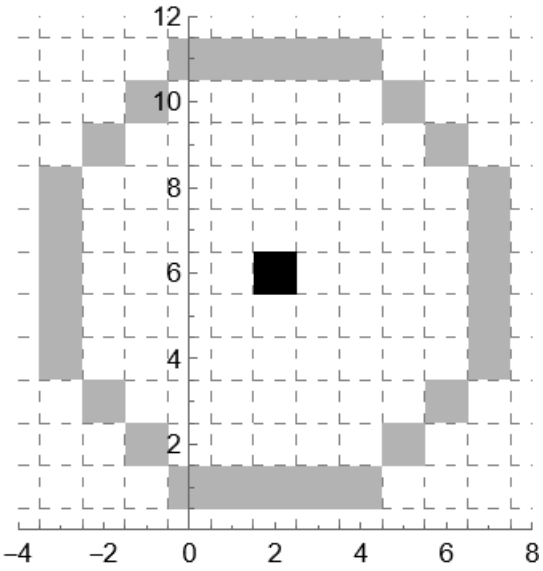
3.4. (Перейти к условию).



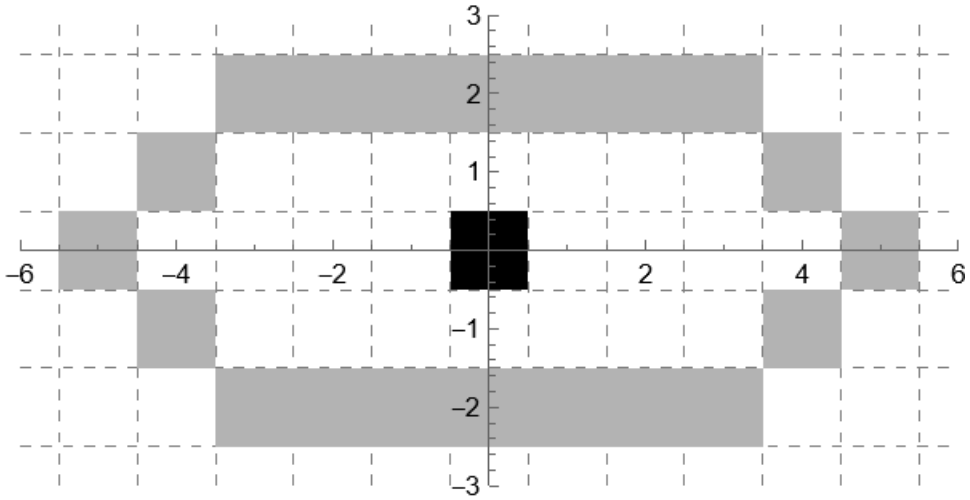
3.5. (Перейти к условию).



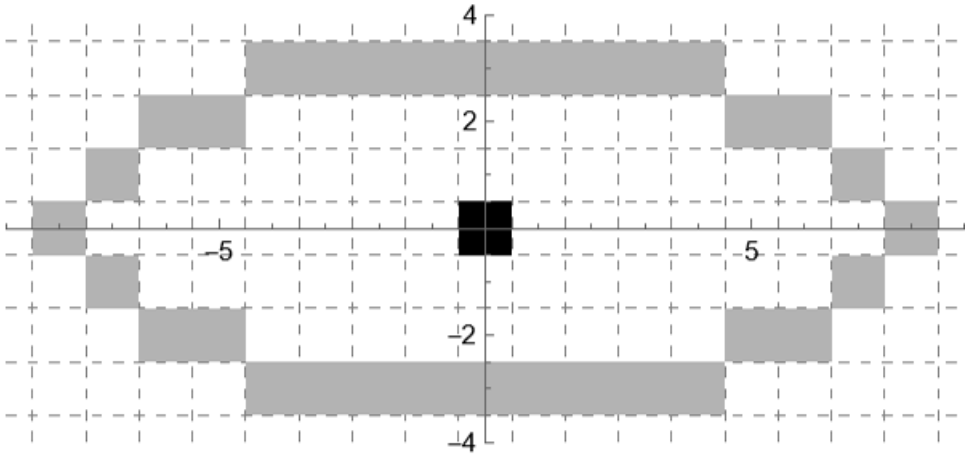
3.6. (Перейти к условию).



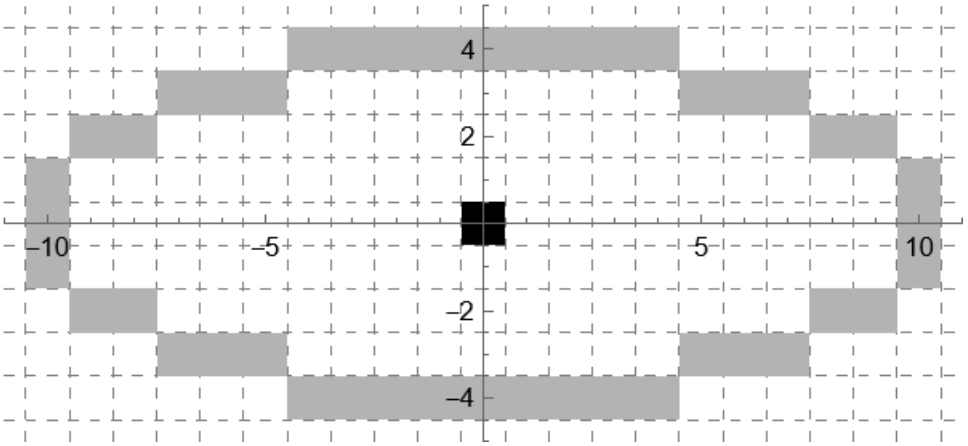
3.7. (Перейти к условию).



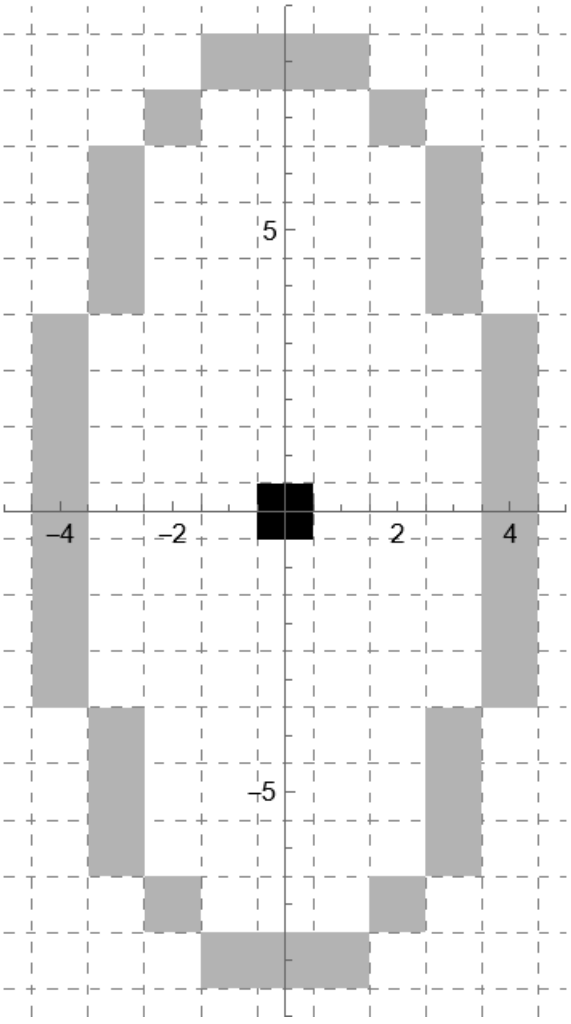
3.8. (Перейти к условию).



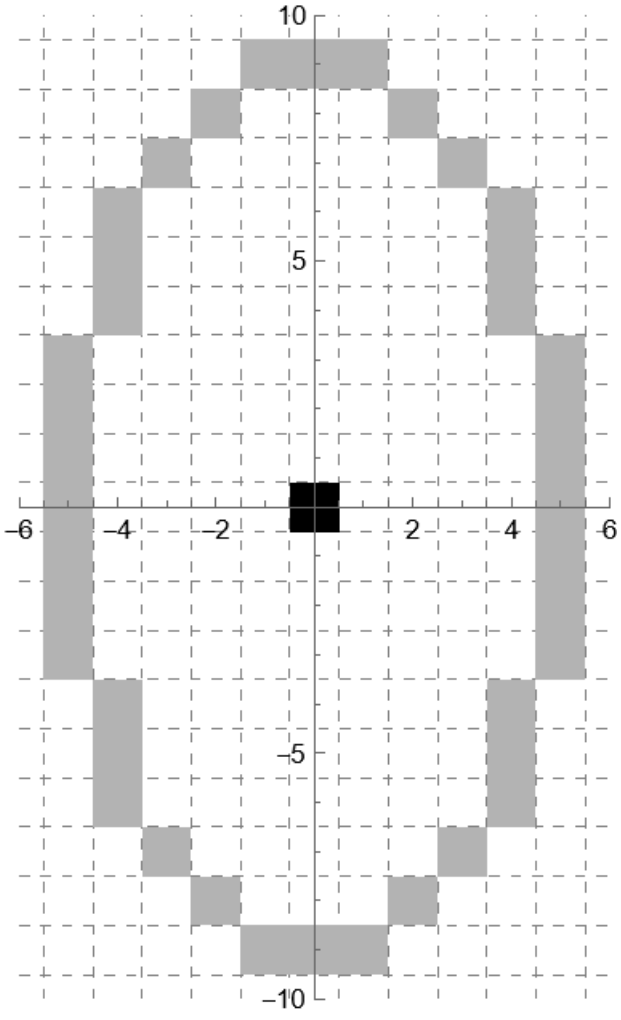
3.9. (Перейти к условию).



3.10. (Перейти к условию).

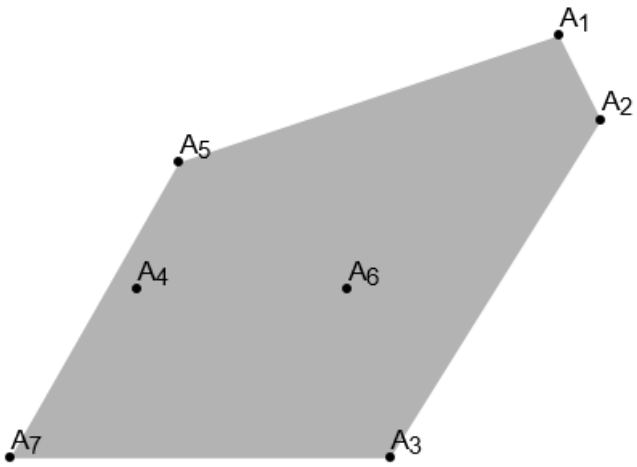


3.11. (Перейти к условию).

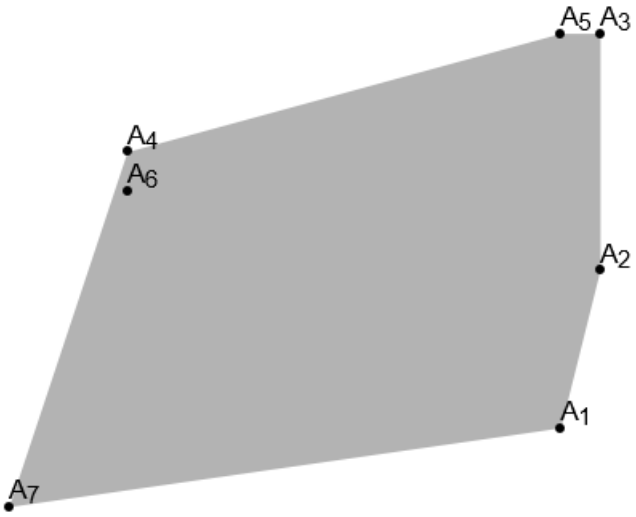


Ответы к задачам подраздела 2.4.

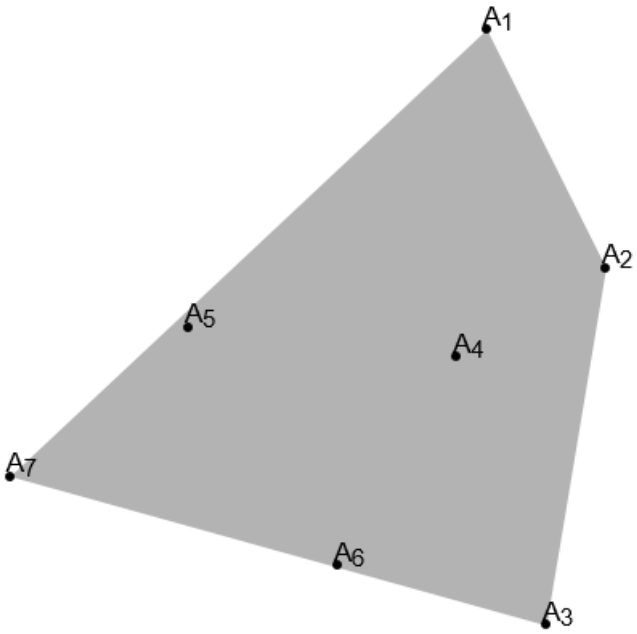
1. (Перейти к условию).



2. (Перейти к условию).

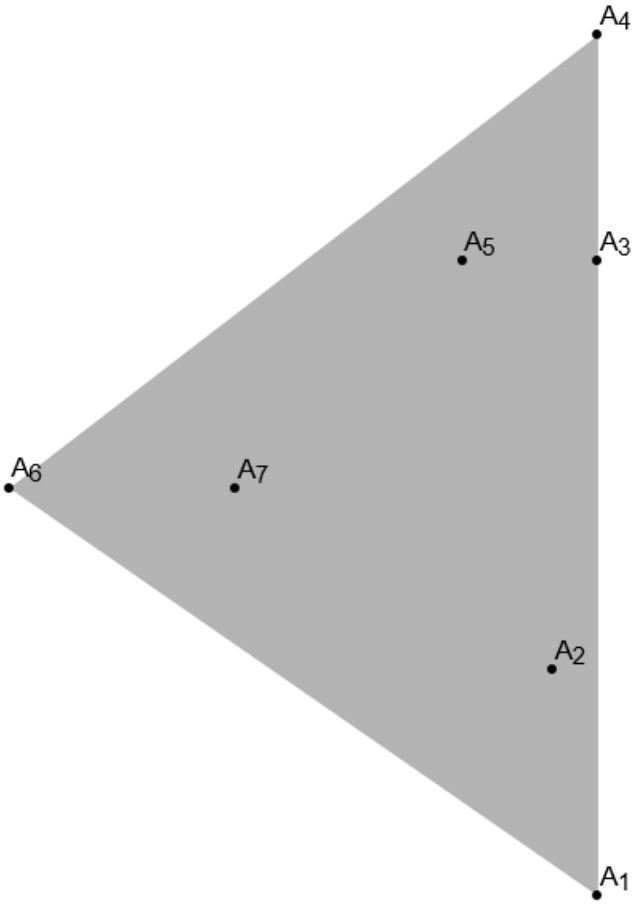


3. (Перейти к условию).

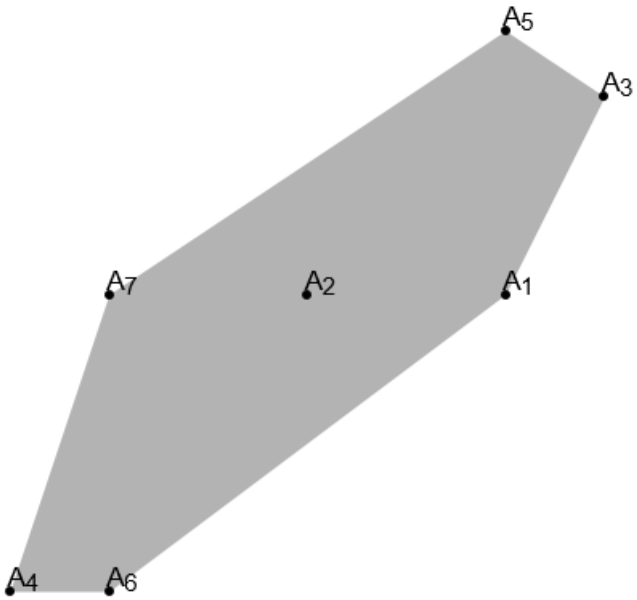


4. (Перейти к условию).

Точка A_3 не является вершиной выпуклой оболочки, т.к. она лежит на отрезке A_4A_1

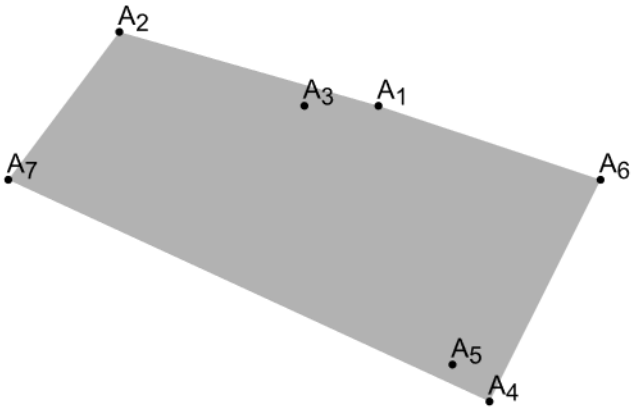


5. (Перейти к условию).



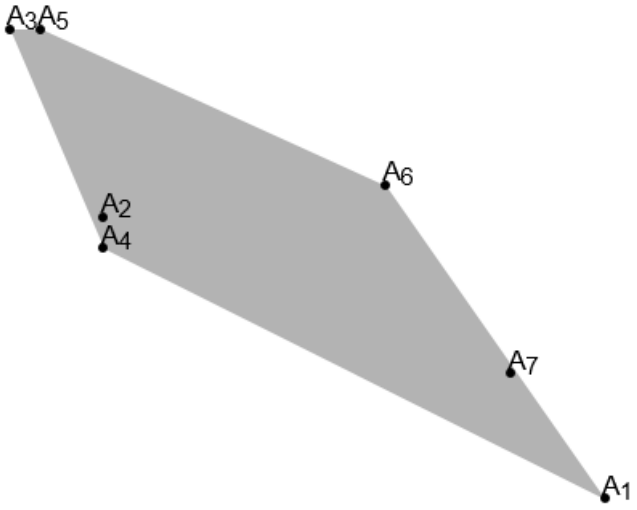
6. (Перейти к условию).

Вершина A_1 является вершиной выпуклой оболочки и не лежит на отрезке A_2A_6

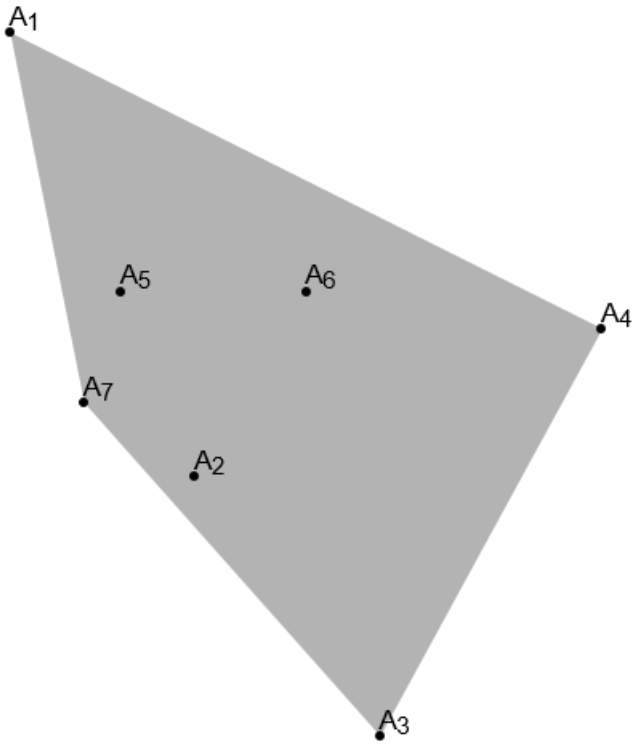


7. (Перейти к условию).

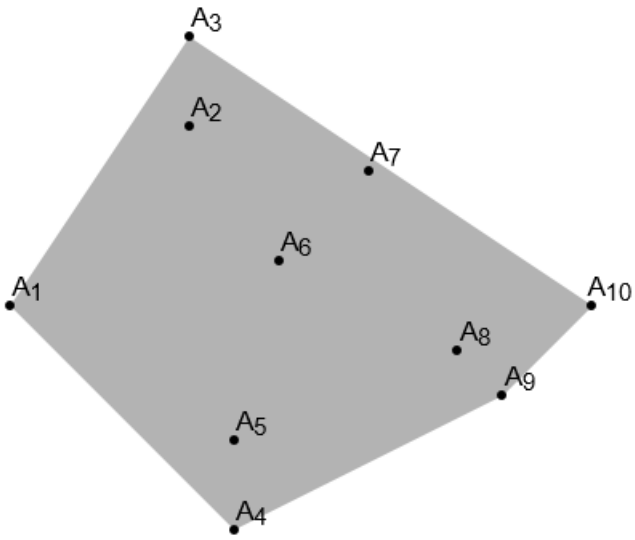
Точка A_7 не является вершиной выпуклой оболочки и не лежит на отрезке A_6A_1



8. (Перейти к условию).

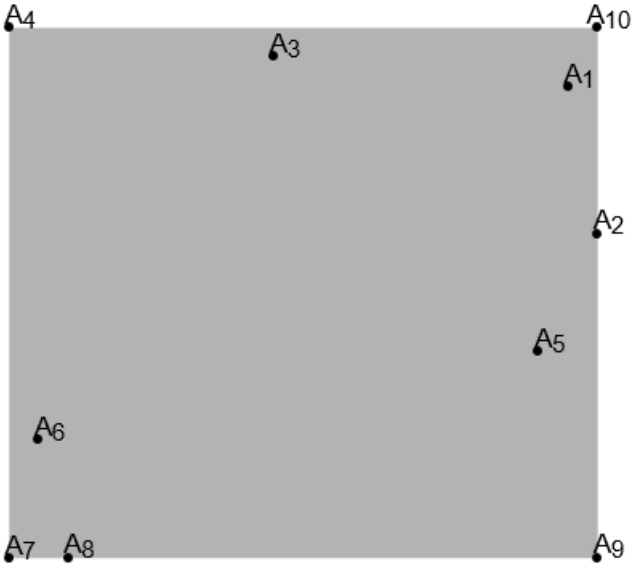


9. (Перейти к условию).



10. (Перейти к условию).

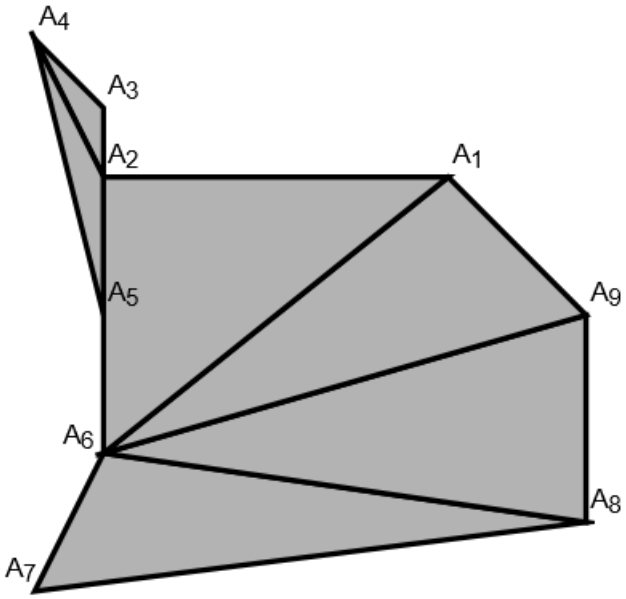
Точки A_2 и A_8 не являются вершинами выпуклой оболочки, т.к. A_2 лежит на отрезке $A_{10}A_9$, A_8 – на отрезке A_7A_9



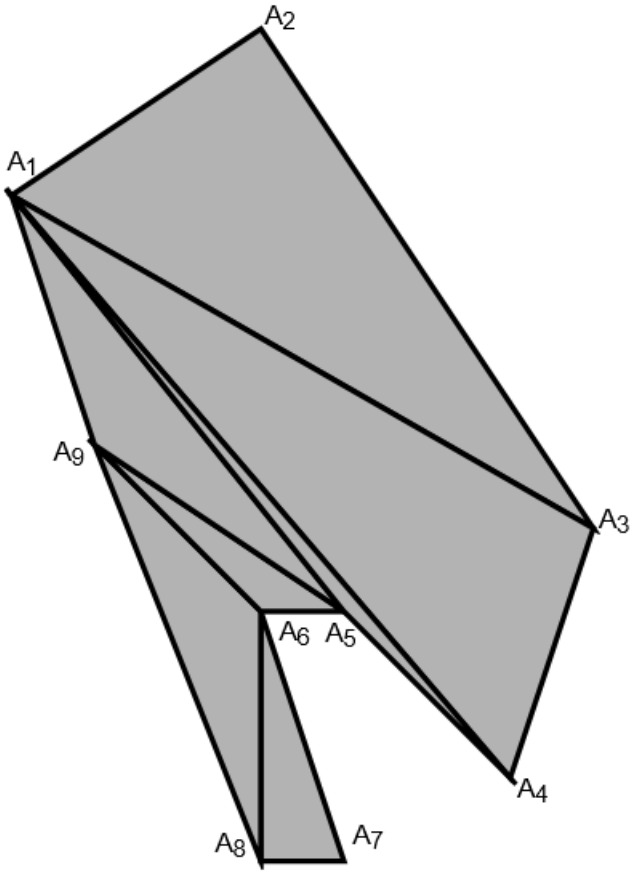
Ответы к задачам подраздела 2.5.

1. В ответах приведены примеры триангуляций. Возможны и другие результаты.

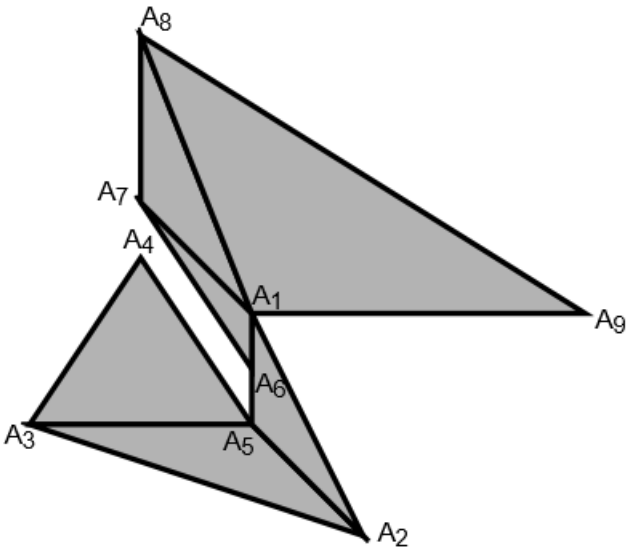
1.1. (Перейти к условию).



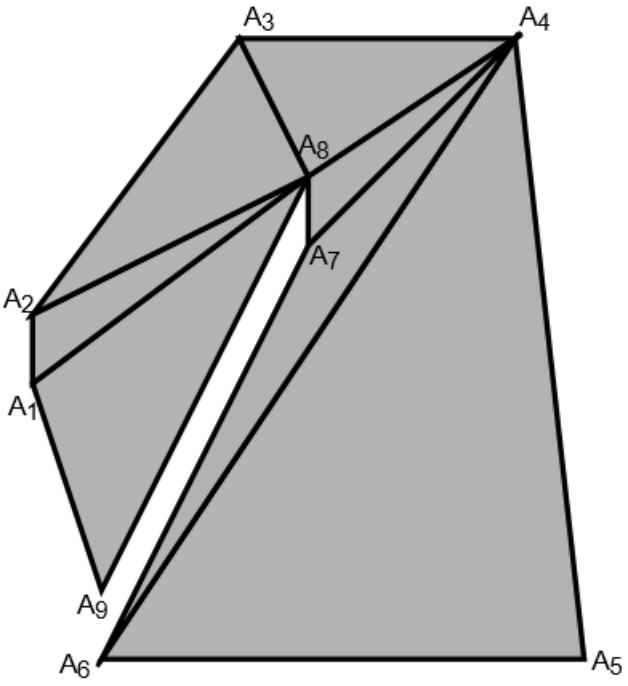
1.2. (Перейти к условию).



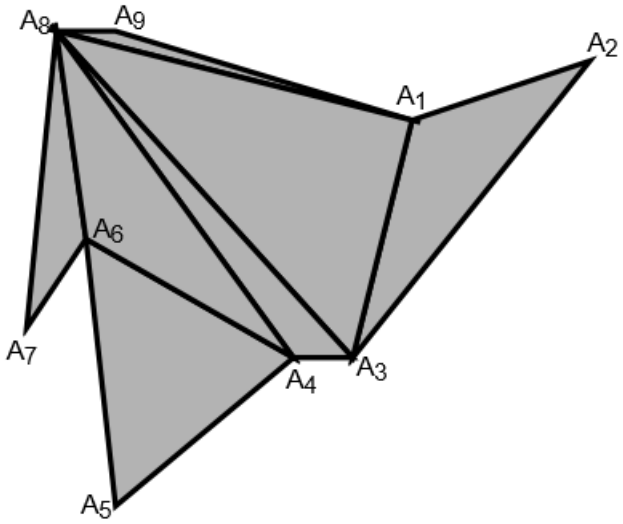
1.3. (Перейти к условию).



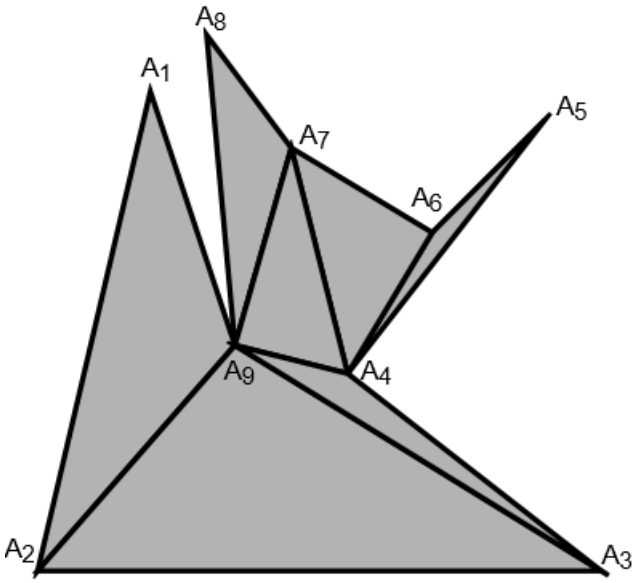
1.4. (Перейти к условию).



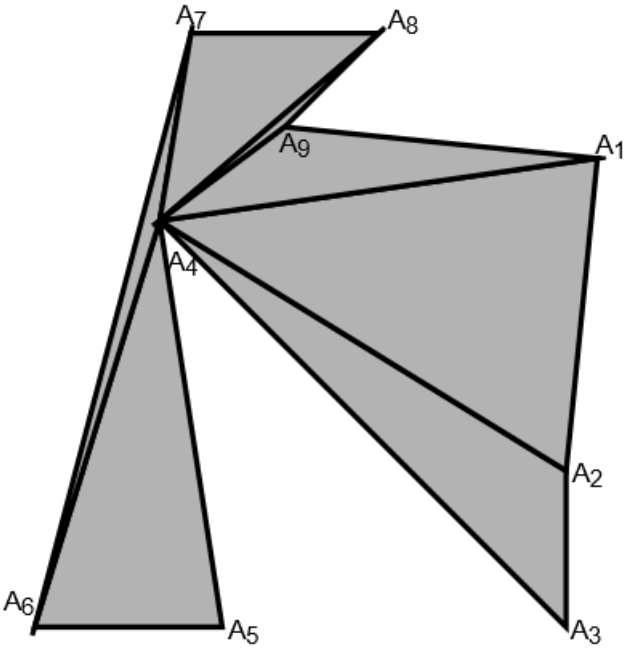
1.5. (Перейти к условию).



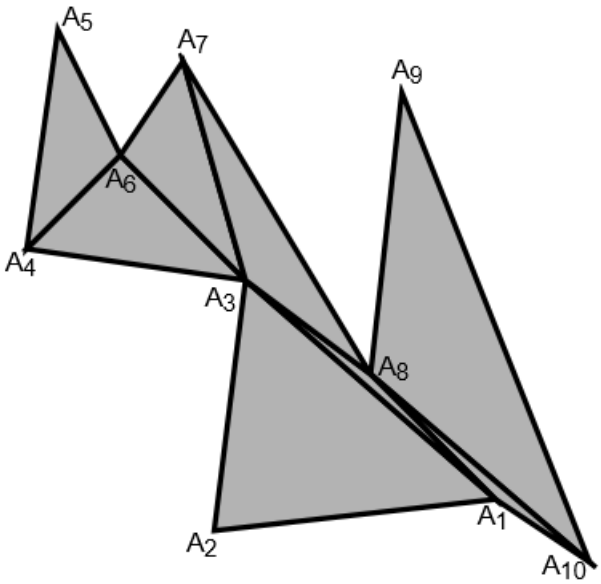
1.6. (Перейти к условию).



1.7. (Перейти к условию).

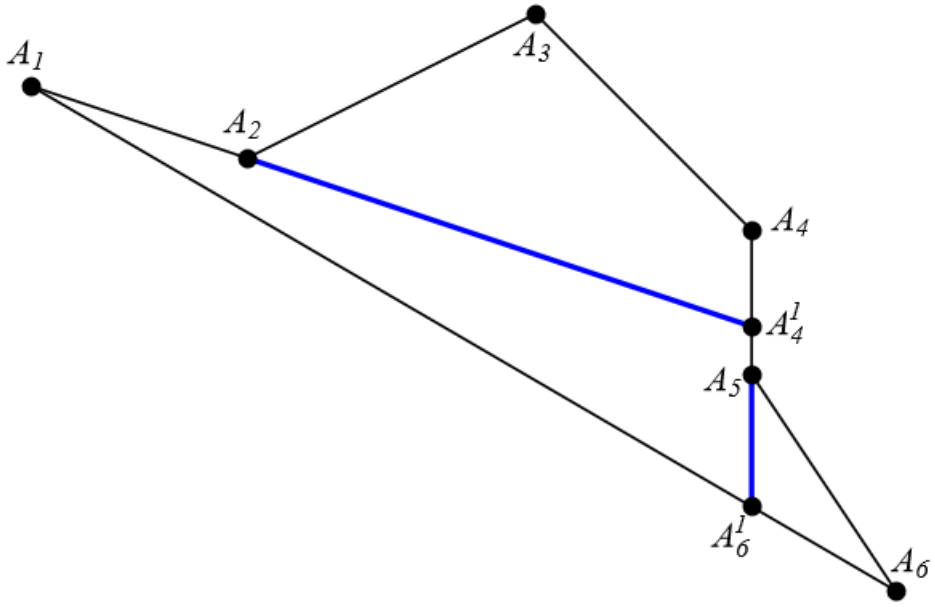


1.8. (Перейти к условию).

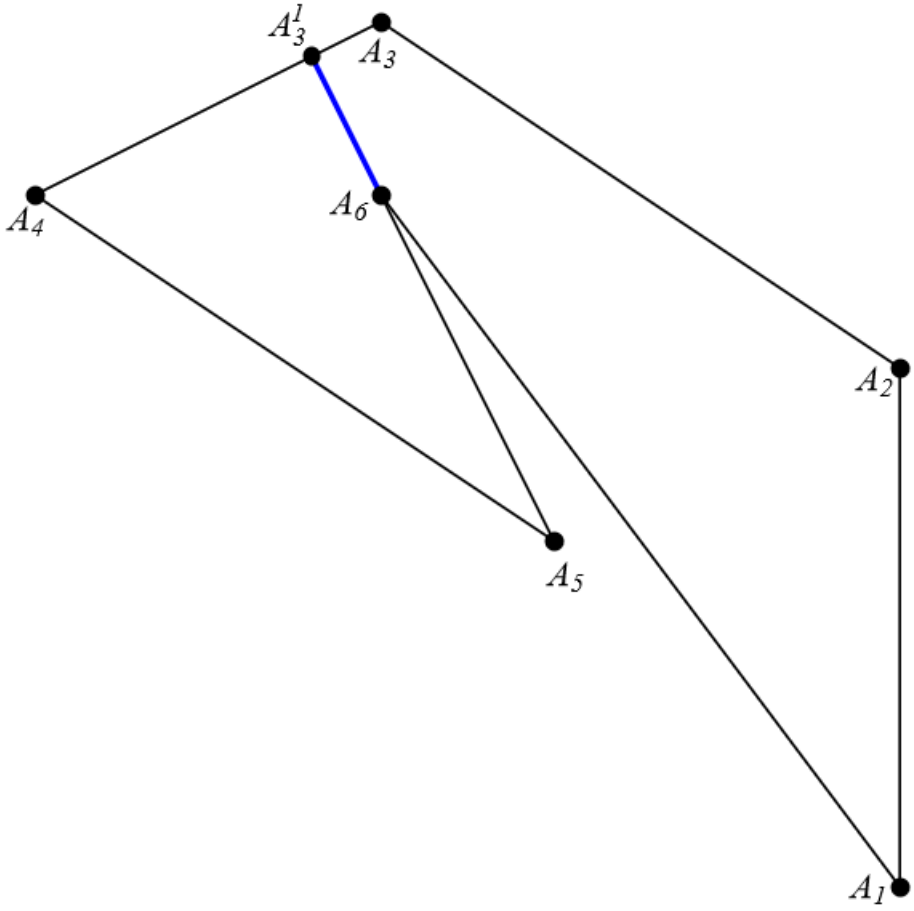


2. В ответах к этой задаче приведены также координаты новых узлов, порождаемых проводимыми хордами.

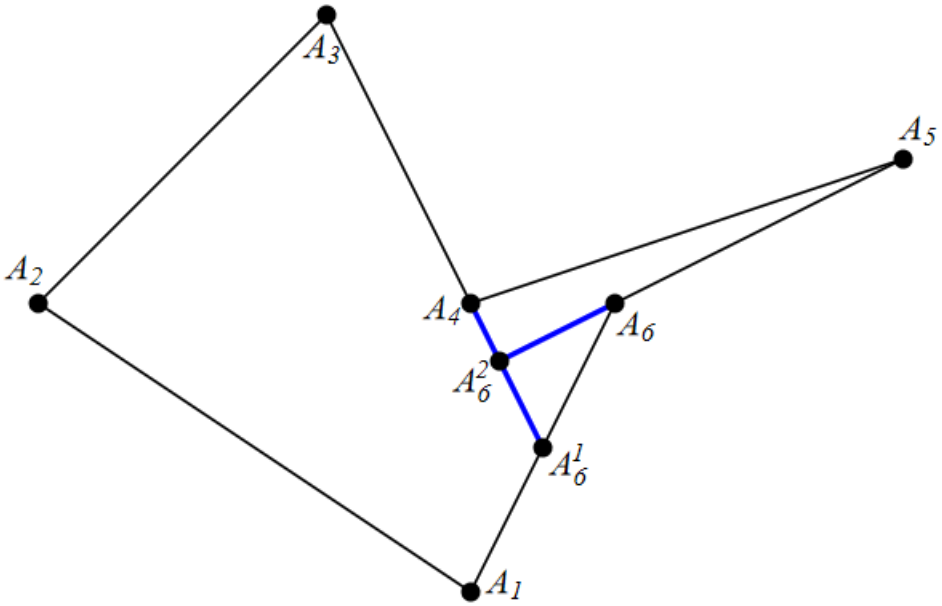
2.1. (Перейти к условию) $A_4^1(10,-10/3)\in A_4A_5$, $A_6^1(10,-35/6)\in A_6A_1$.



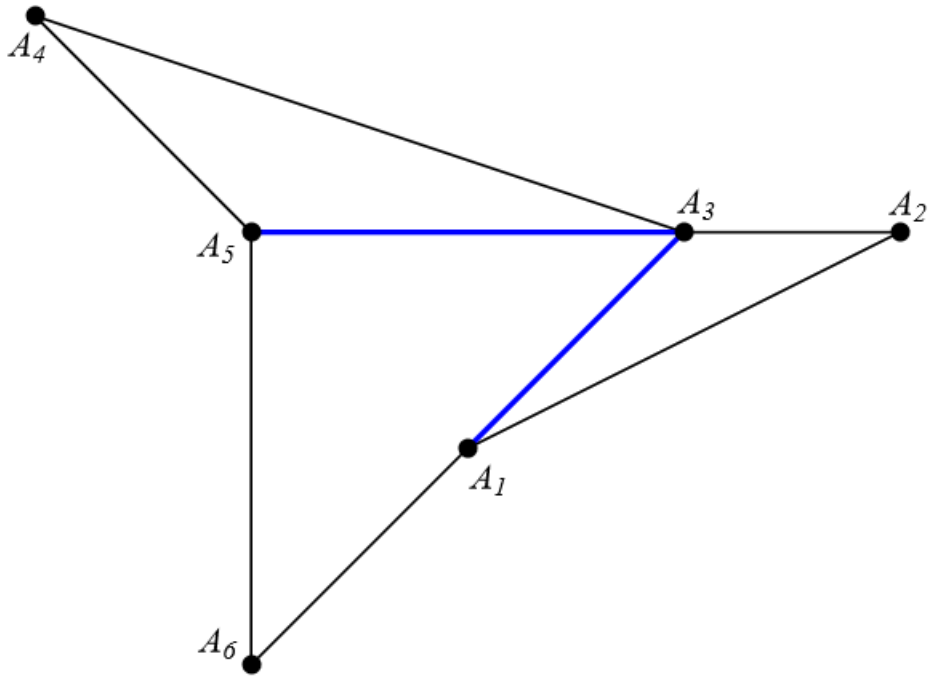
2.2. (Перейти к условию) $A_3^1(-17/5,24/5)\in A_3A_4$.



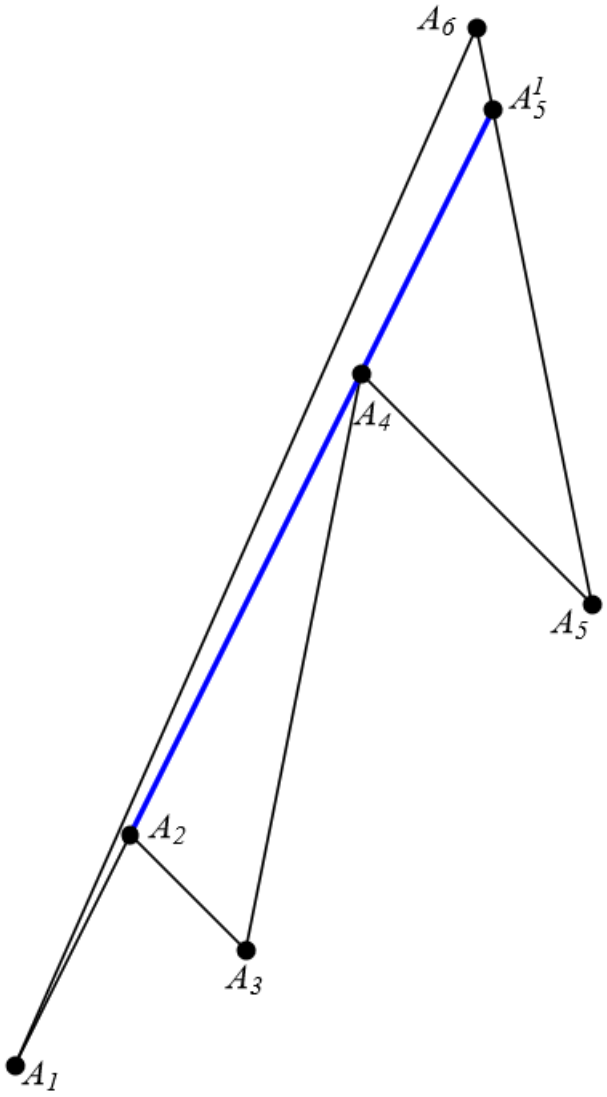
2.3. (Перейти к условию) $A_6^1(1/2,1) \in A_6A_1$, $A_6^2(1/5,8/5) \in A_6^1A_4$.



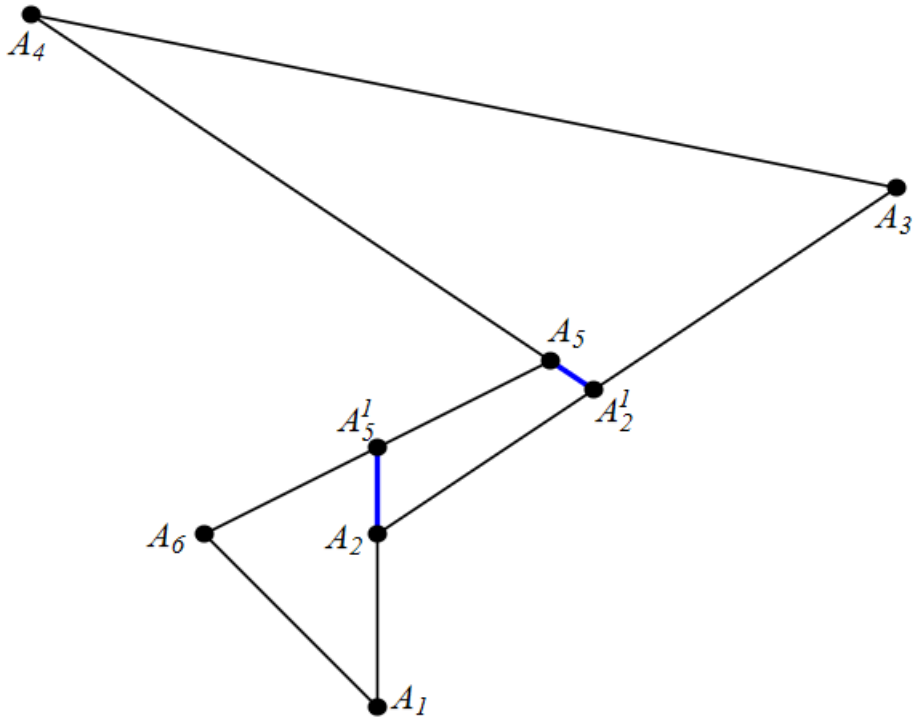
2.4. (Перейти к условию) Все хорды попали в вершины первоначального полигона.



2.5. (Перейти к условию) $A_5^1(29/7, 72/7) \in A_5A_6$.

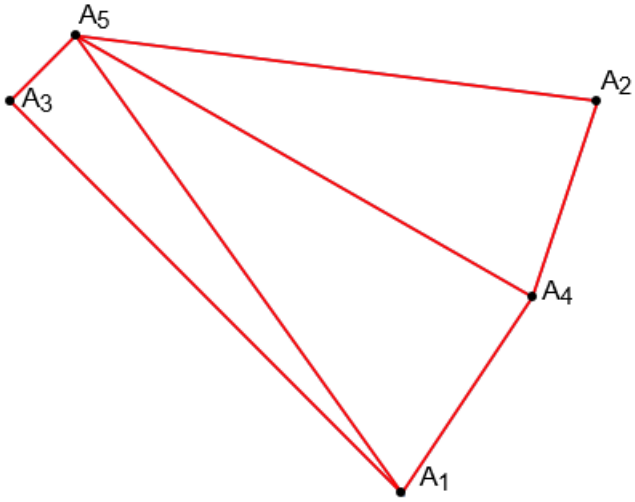


2.6. (Перейти к условию) $A_5^1(0,3/2) \in A_5A_6$, $A_2^1(5/4,11/6) \in A_2A_3$.

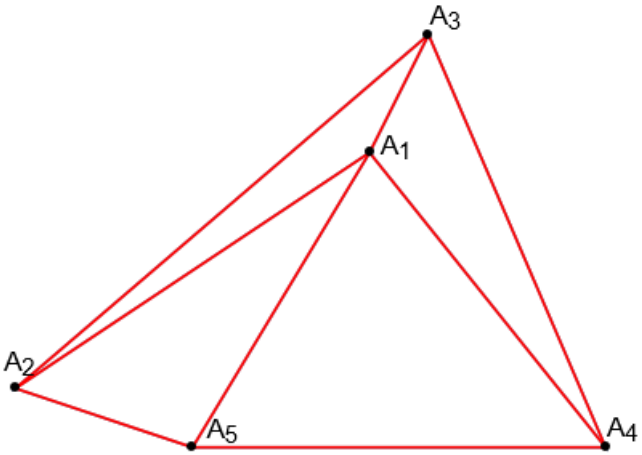


Ответы к задачам подраздела 2.6.

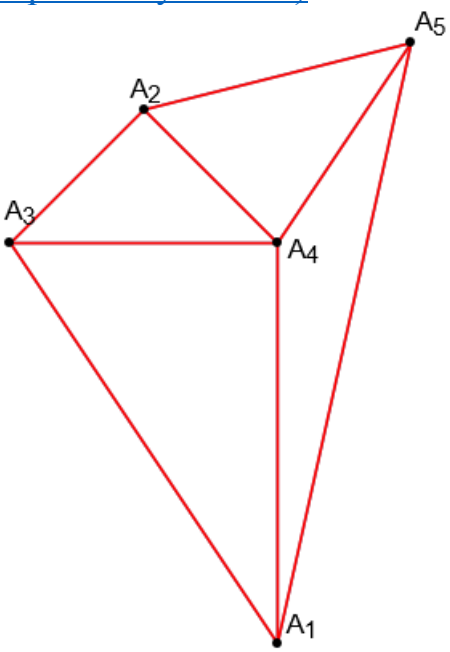
1.1. (Перейти к условию)



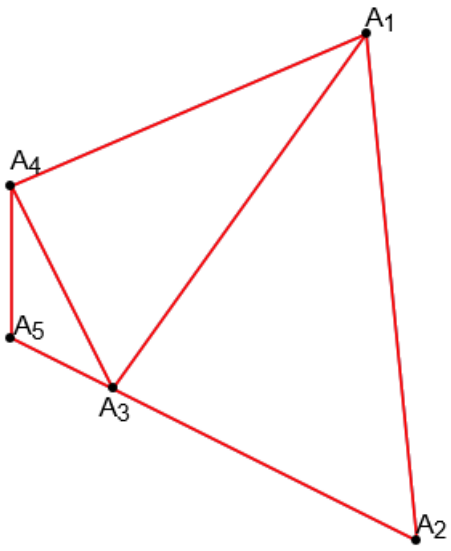
1.2. (Перейти к условию)



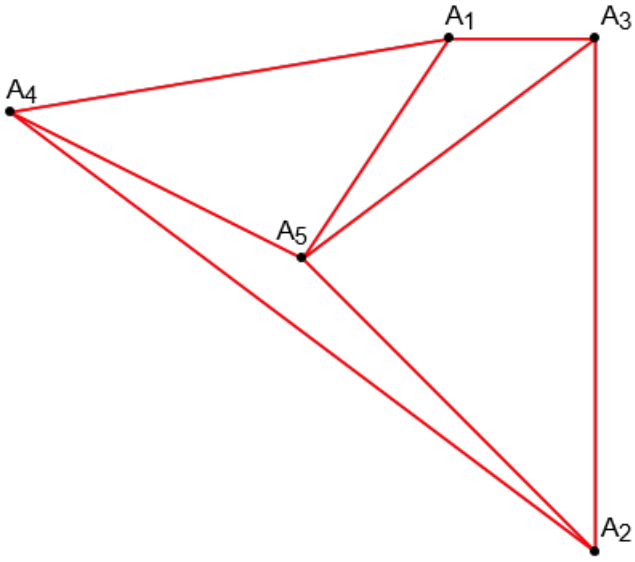
1.3. (Перейти к условию)



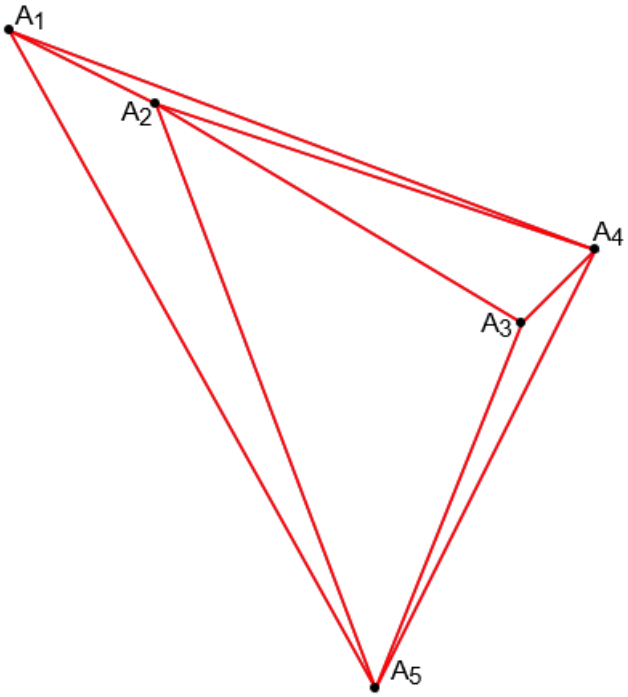
1.4. (Перейти к условию)



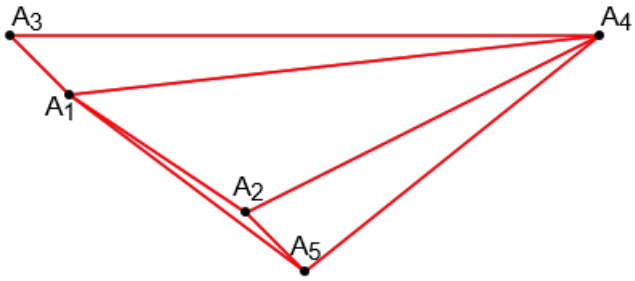
1.5. (Перейти к условию)



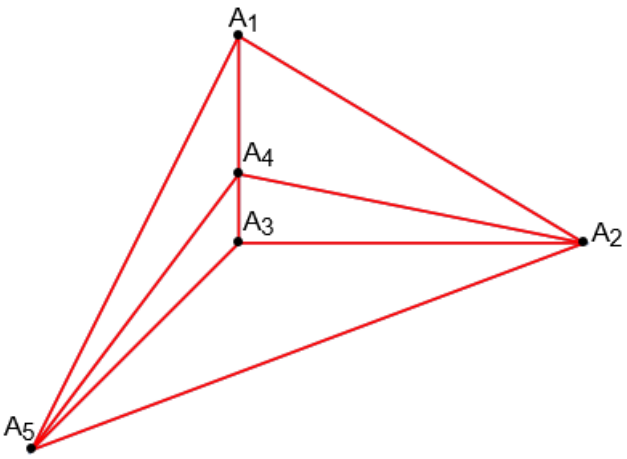
1.6. (Перейти к условию)



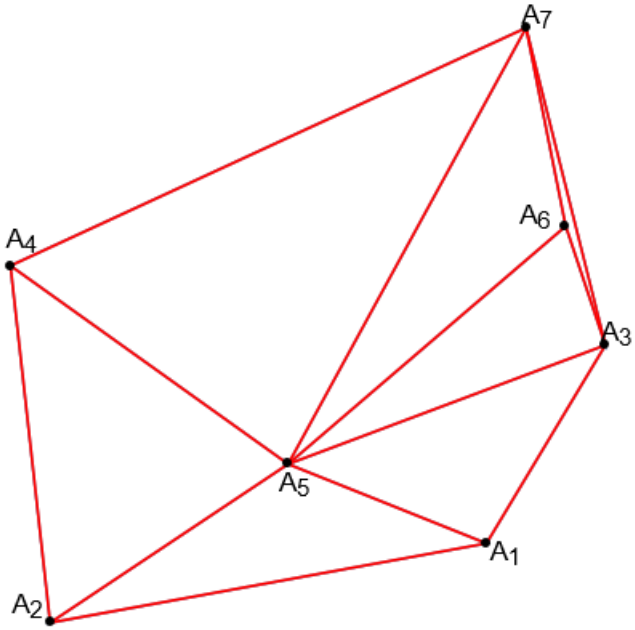
1.7. (Перейти к условию)



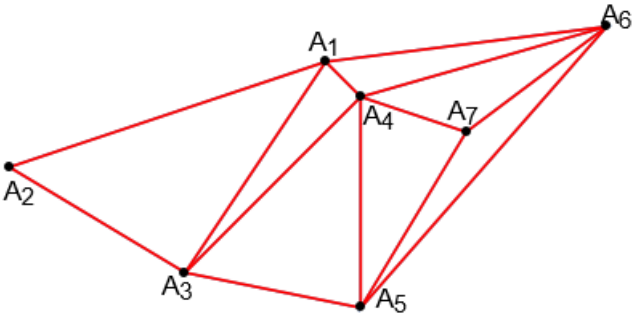
1.8. (Перейти к условию)



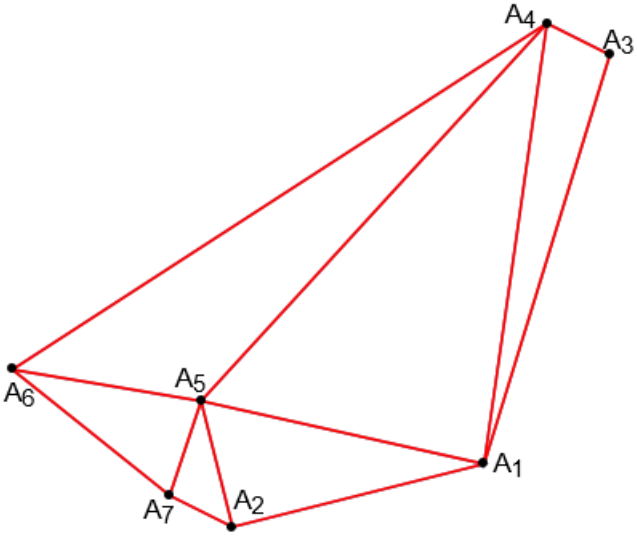
1.9. (Перейти к условию)



1.10. (Перейти к условию)

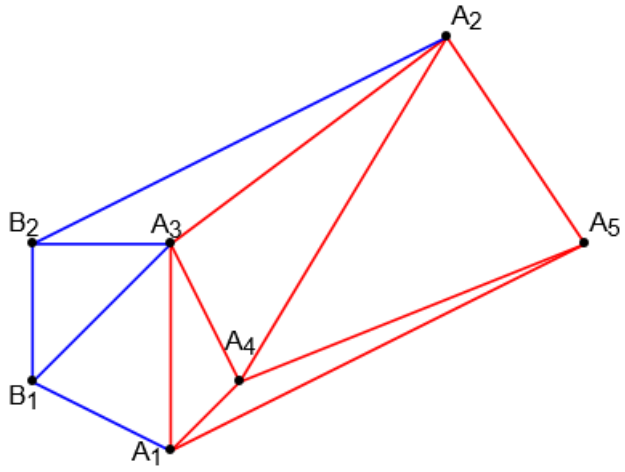


1.11. (Перейти к условию)

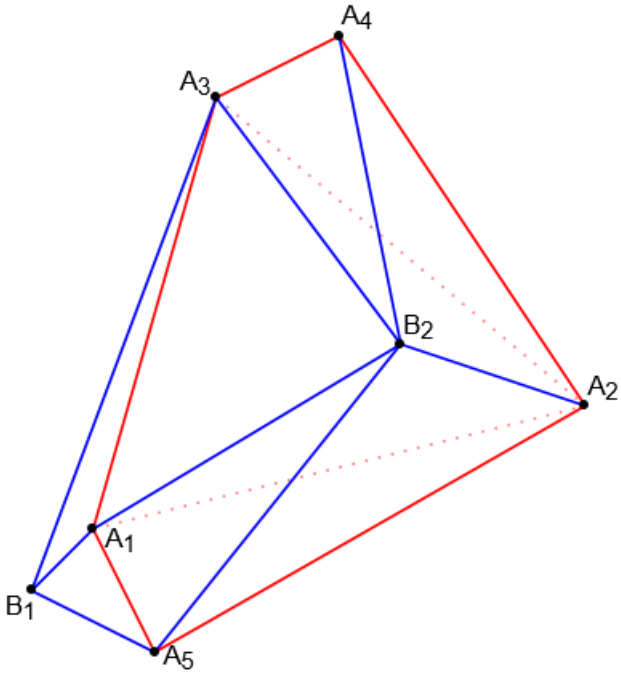


2–3. На чертежах ниже рёбра исходных и итоговой триангуляций обозначены следующим образом: синие – добавленные, красные пунктирные – удалённые рёбра, красные сплошные – рёбра, не претерпевшие изменений.

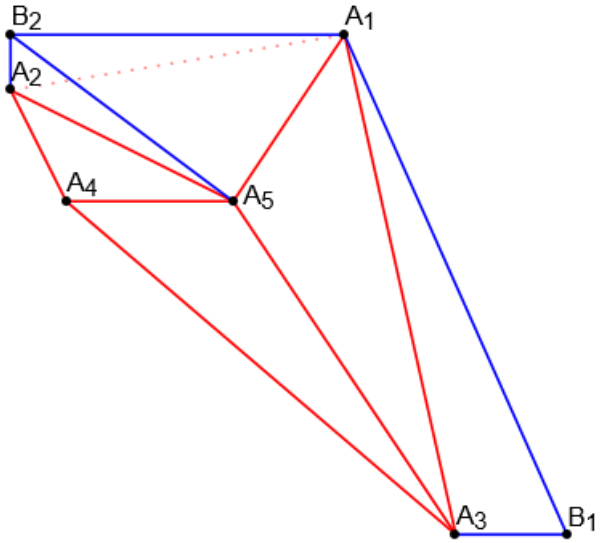
2.1. (Перейти к условию)



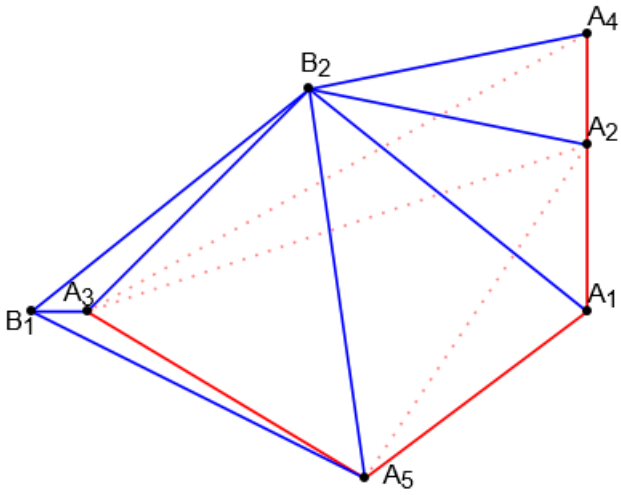
2.2. (Перейти к условию)



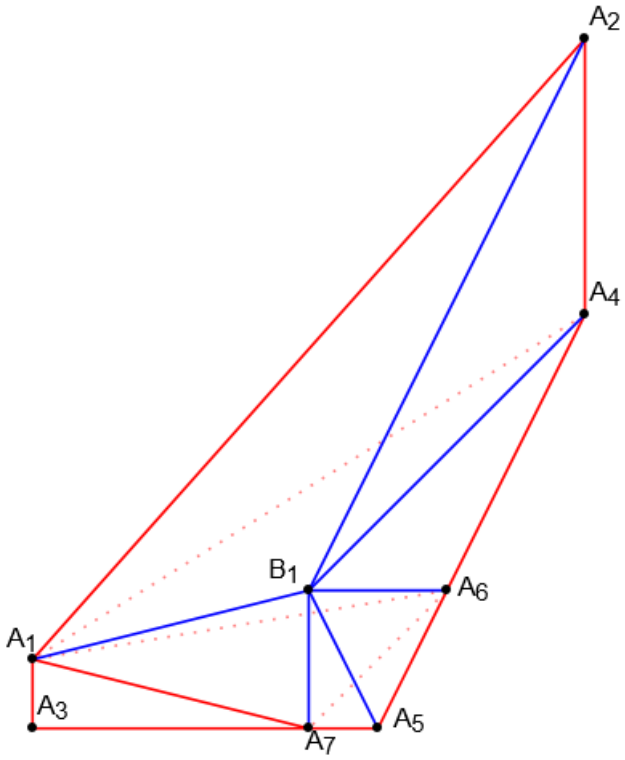
2.3. (Перейти к условию)



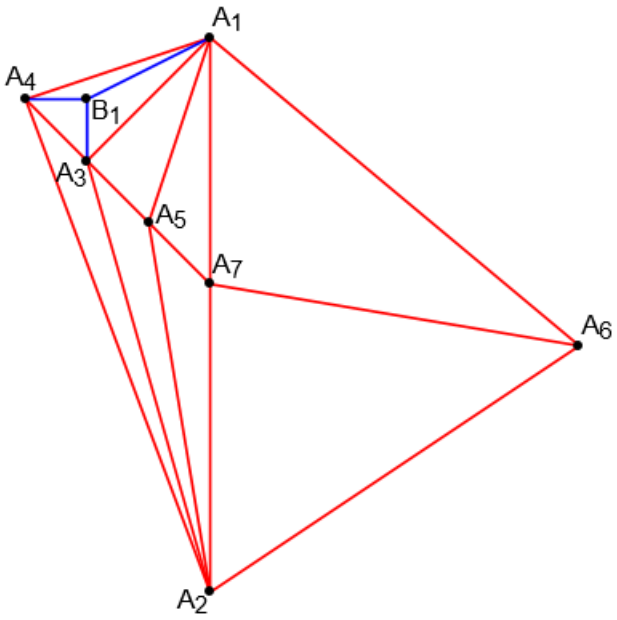
2.4. (Перейти к условию)



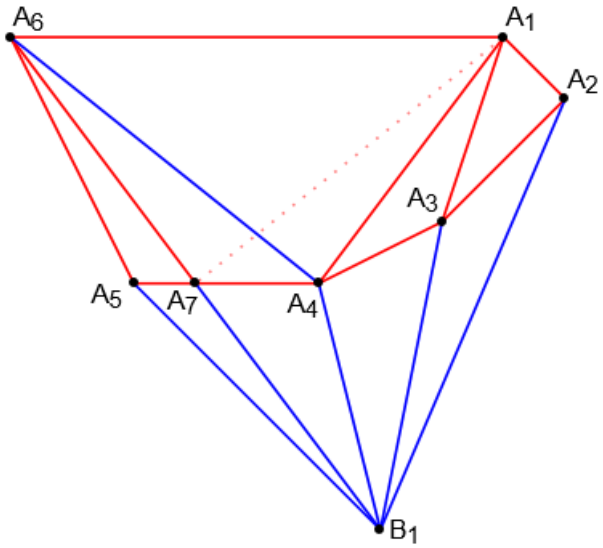
2.5. (Перейти к условию)



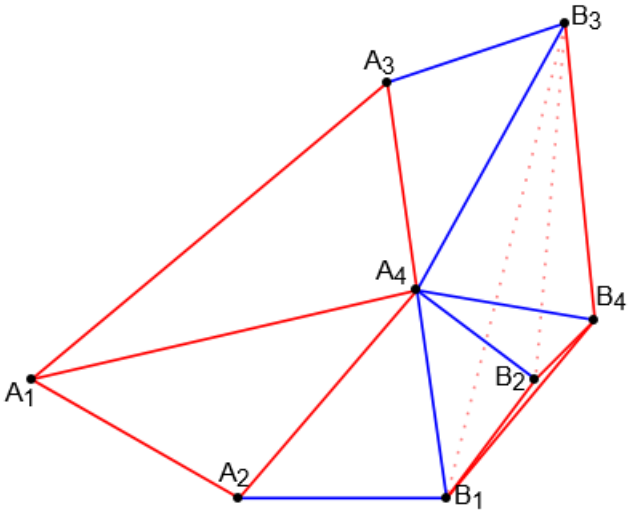
2.6. (Перейти к условию)



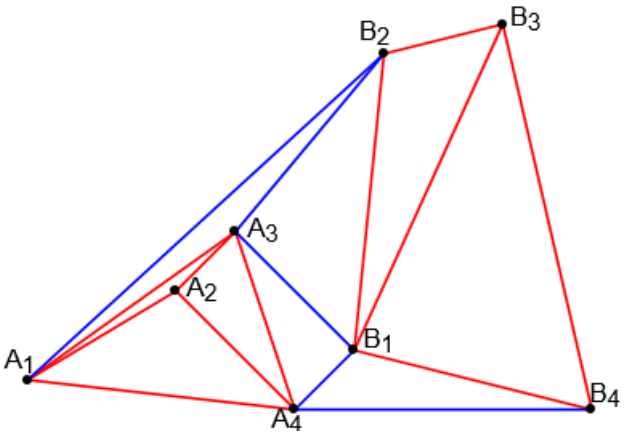
2.7. (Перейти к условию). Точки A_1, A_4, A_6 и A_7 лежат на одной окружности, поэтому делать флип ребра A_1A_7 на A_4A_6 не нужно, но можно.



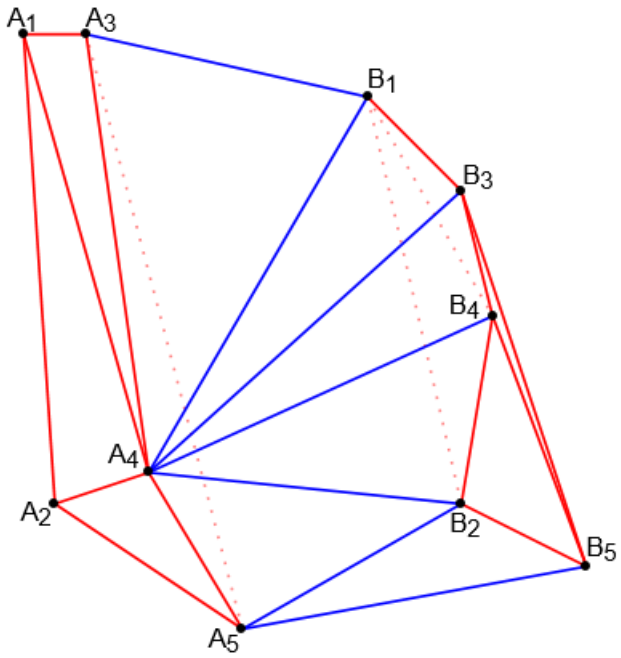
3.1. (Перейти к условию)



3.2. (Перейти к условию)



3.3. (Перейти к условию)



Ответы к задачам подраздела 2.7.

1. (Перейти к условию). В приведённых ниже ответах через p обозначена точка начала, $\vec{v}_{\text{отр}}$ – направляющий вектор отражённого луча.

1.1. $p(0, 3, 1)$, $\vec{v}_{\text{отр}}(16,5,-18)$,

1.2. $p(0, 0, 0)$, $\vec{v}_{\text{отр}}(11,5,19)$,

1.3. $p(5, 3, -1)$, $\vec{v}_{\text{отр}}(-23,-17,-7)$,

1.4. луч не падает на поверхность,

1.5. $p\left(11/3,-10,14/3\right)$, $\vec{v}_{\text{отр}}(35,-39,20)$,

1.6. $p(5, 0, 3)$, $\vec{v}_{\text{отр}}(37,13,14)$,

1.7. $p(2, 3, 4)$, $\vec{v}_{\text{отр}}(1,2,1)$,

1.8. $p(3, 3, 0)$, $\vec{v}_{\text{отр}}(87,18,-7)$,

1.9. $p(2, -2, 1)$, $\vec{v}_{\text{отр}}(2\ln 2+1,2\ln 2(1-\ln 2),2\ln^2 2+2\ln 2-1)$.

2. (Перейти к условию). В приведённых ниже ответах через p обозначена точка начала, $\vec{v}_{\text{прел}}$ – направляющий вектор преломлённого луча.

2.1. $p\left(-1/5,13/5,-4\right)$, $\vec{v}_{\text{прел}}\left(-16\sqrt{6}+2\sqrt{10},-7\sqrt{6}-6\sqrt{10},-5\sqrt{6}+2\sqrt{10}\right)$,

2.2. $p\left(7/2,-1/2,0\right)$, $\vec{v}_{\text{прел}}(-1,1,2)$,

2.3. $p\left(-29/16,13/16,17/16\right)$, $\vec{v}_{\text{прел}}\left(2\sqrt{7}+5,-3\sqrt{7}+3,\sqrt{7}-1\right)$,

2.4. $p\left(-7/2,11/2,5\right)$, $\vec{v}_{\text{прел}}\left(1,1,-4+\sqrt{10}\right)$,

2.5. $p(0, 0, 0)$, $\vec{v}_{\text{прел}}(12\sqrt{2}+\sqrt{41},9\sqrt{2}+4\sqrt{41},16\sqrt{2}-3\sqrt{41})$,

2.6. $p\left(-13/2,5/2,5\right)$, $\vec{v}_{\text{прел}}\left(-\sqrt{10}-1,\sqrt{10}+1,-\sqrt{10}+2\right)$.

3. (Перейти к условию). В приведённых ниже ответах через q_i обозначена тень точки a_i (при условии видимости всех теней).

3.1. Тени не видны наблюдателю.

3.2. $q_1(-12,12,-2)$, $q_5(-13/2,-29/4,3/4)$, $q_6(-1,-37/7,-25/7)$.

3.3. $q_2(3,-7,5)$, $q_5(0,-14/5,-14/5)$.

3.4. $q_1(-3,-35,23)$, $q_2(0,-21,27/2)$, $q_3(3,-6,3)$, $q_4(4,2,-7/2)$, $q_5(5,-11,11)$, $q_6(0,-18,21/2)$.

3.5. $q_1(-4/5,26/5,0)$, $q_3(-18/5,39/10,-1)$, $q_4(-33/5,12/5,-2)$, $q_5(-4/5, 37/10,1)$, $q_6(-3/5,42/5,-2)$. Точка a_2 лежит на подложке П, поэтому тени не отбрасывает.

3.6. $q_2(-7,-3,-7/2)$.

4. (Перейти к условию). В приведённых ниже ответах через q_i обозначено отражение точки a_i (при условии его наличия).

4.1. $q_1(2,-22/7,-17/7)$, $q_2(2/11,-36/11,-31/11)$,

4.2. $q_1(4/3,1/3,2/3)$, $q_2(-1/3,5/3,1/3)$,

4.3. $q_2(8/7,1,-19/7)$,

4.4. $q_1(2/27,4/27,2/27)$, $q_3(-2/3,-1/3,1/3)$,

4.5. $q_2(3/19,-3/19,-18/19)$,

4.6. $q_1(17/3,19/6,26/3)$, $q_2(-1/3,-4/3,17/3)$, $q_3(-1,-7/4,11/2)$.

5. (Перейти к ответу) В приведённых ниже ответах через q_i обозначено преломлённое изображение точки a_i (при условии его наличия).

5.1. $q_1\left(-4,1+4\sqrt{7/3},0\right)$, $q_2\left(-4,-3+\sqrt{7/3},-1\right)$, $q_3\left(-4,1+7\sqrt{7/3},0\right)$.

5.2. Никаких преломленных изображений видно не будет, так как выполняется равенство $\vec{s}\cdot\vec{n}=0$, где $\vec{n}$ – нормальный вектор плоскости П (происходит наблюдение в торец плоской преломляющей поверхности).

5.3. $q_1\left(\frac{-66+13\sqrt{2}}{14},\frac{-18-13\sqrt{2}}{7},\frac{8+26\sqrt{2}}{7}\right)$, $q_2\left(\frac{-12+17\sqrt{2}}{28},\frac{-51-17\sqrt{2}}{14},\frac{-59+34\sqrt{2}}{14}\right)$, $q_3\left(\frac{-16+11\sqrt{2}}{28},\frac{-61-11\sqrt{2}}{14},\frac{-25+22\sqrt{2}}{14}\right)$,

5.4. $q_1\left(\sqrt{3/2},1-\sqrt{3/2},0\right)$, $q_2\left(3+1/\sqrt{6},3-1/\sqrt{6},0\right)$,

5.5. $q_1(4/3,7/6,8/3)$,

5.6. $q_3(-72/25,-2,79/25)$.

3. РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

В течение семестра по учебной дисциплине предусмотрен контроль знаний в форме 2 коллоквиумов и 2 контрольных работ. В настоящем ЭУМК приведены вопросы второго коллоквиума и варианты второй контрольной работы, посвящённые в том числе данной части дисциплины. Также приведён список вопросов к экзамену, часть из которых относится к темам, затронутым в настоящем ЭУМК.

3.1. Вопросы к коллоквиуму №2

Ниже приведён перечень вопросов, которые могут быть заданы студентам на коллоквиуме по данному разделу дисциплины. Рекомендуется составлять варианты, содержащие по два вопроса (один из вопросов 1–15 и один из вопросов 16–30). Также рекомендуется оценивать ответ студента исходя из полноты ответа, владения отдельными терминами и темами заданий в целом, способностью применять наиболее подходящие способы решения поставленных задач. Жирным выделены вопросы, относящиеся к настоящему ЭУМК.

1. **Алгоритм Коэна-Сазерленда**
2. **Алгоритм Лианга-Барски**
3. **Алгоритм Кируса-Бека**
4. **Алгоритм DDA-линии**
5. **Алгоритм Брезенхема**
6. **Алгоритм Кастла-Питвэя**
7. **Алгоритм Ву**
8. **Алгоритм Джарвиса**
9. **Алгоритм Грэхема**
10. **Алгоритм Эндрю**
11. **Алгоритм "быстрой оболочки"**
12. **Алгоритмы отсечения в трёхмерном пространстве. Алгоритм Робертса**
13. **Алгоритмы отсечения в трёхмерном пространстве. Алгоритм художника**
14. **Алгоритмы отсечения в трёхмерном пространстве. Алгоритм z-буфера**
15. **Триангуляция полигона**
16. **Триангуляция точек. Триангуляция Делоне**
17. **Основные принципы оптической геометрии. Отражение световых лучей через криволинейную и плоскую поверхность**

18. **Основные принципы оптической геометрии. Преломление световых лучей через криволинейную и плоскую поверхность**
19. **Основные принципы оптической геометрии. Тень точки на плоской подложке**
20. **Основные принципы оптической геометрии. Наблюдаемое отражение точки на плоской идеально отражающей поверхности**
21. **Основные принципы оптической геометрии. Наблюдаемое преломление точки через плоскую идеально преломляющую поверхность**
22. Приближение кривых. Интерполяционный многочлен Лагранжа
23. Приближение кривых. Интерполяция кривых по методу наименьших квадратов
24. Приближение кривых. Интерполяционный многочлен Эрмита
25. Приближение кривых. Аппроксимация кривых по методу наименьших квадратов
26. Приближение кривых. Линии Безье
27. Приближение кривых. Составные интерполяционные кривые
28. Приближение кривых. Сплайны
29. Моделирование поверхностей. Кинематические поверхности
30. Моделирование поверхностей. Полиномиальные поверхности

3.2. Варианты контрольной работы №2

Вариант 1

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(6,\ 0),\ B(0,\ -1),\ C_1(-1,4),\ C_2(-2,0),\ C_3(0,-2),\ C_4(3,0),\ C_5(5,2).$$

2. Вычислить матрицу тела для тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$:

$$A_1(5,0,5),\ A_2(4,1,1),\ A_3(6,3,1),\ A_4(8,1,7).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(7,5),\ A_2(2,4),\ A_3(6,3),\ A_4(5,6),\ A_5(5,7),\ A_6(2,6),\ A_7(7,7).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(4,10),\ A_2(10,1),\ A_3(6,-10),\ A_4(-7,-10),\ A_5(-6,-7),$$
$$A_6(-2,-8),\ A_7(5,5),\ A_8(4,5),\ A_9(-9,-5).$$

Вариант 2

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(5,\ 4),\ B(-7,\ -3),\ x_{\min} = -4,\ y_{\min} = -3,\ x_{\max} = 3,\ y_{\max} = 3.$$

2. Осуществить отсечение отрезка AB внешним выпуклым отсекателем с известной матрицей тела C:

$$A(1,\ 2,\ 2),\ B(1,\ 3,\ 4),\ C = \begin{pmatrix} -2 & -6 & -3 & 24 \\ 8 & 6 & 3 & -24 \\ -10 & -12 & 3 & 30 \\ 4 & 12 & -3 & -12 \end{pmatrix}.$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(2,2),\ A_2(7,1),\ A_3(4,5),\ A_4(5,6),\ A_5(6,1),\ A_6(2,6),\ A_7(4,0).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(-4,-10),\ A_2(10,-4),\ A_3(9,8),\ A_4(8,9),\ A_5(0,10),$$
$$A_6(7,2),\ A_7(-3,-6),\ A_8(-3,1),\ A_9(-7,-7).$$

Вариант 3

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(7,\ 8),\ B(-3,\ 1),\ x_{\min} = -1,\ y_{\min} = 1,\ x_{\max} = 5,\ y_{\max} = 7.$$

2. Определить видимость граней тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$, для ближнего наблюдателя S:

$$A_1(7,6,1),\ A_2(10,0,10),\ A_3(5,4,9),\ A_4(9,8,2),\ S(-2,-5,2).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(2,5),\ A_2(3,6),\ A_3(4,5),\ A_4(7,5),\ A_5(6,4),\ A_6(6,3),\ A_7(1,4).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(8,10),\ A_2(-8,8),\ A_3(-10,4),\ A_4(-3,8),\ A_5(2,6),$$
$$A_6(2,2),\ A_7(-3,-7),\ A_8(8,-4),\ A_9(10,4).$$

Вариант 4

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x=x_{\min}$, $x=x_{\max}$, $y=y_{\min}$, $y=y_{\max}$, а также растеризовать результат этого отсечения:

$$A(16,-1),\ B(4,4),\ x_{\min}=4,\ y_{\min}=-3,\ x_{\max}=13,\ y_{\max}=5.$$

2. Осуществить отсечение отрезка AB внешним выпуклым отсекателем с известной матрицей тела C:

$$A(3,2,0),\ B(3,0,5),\ C=\begin{pmatrix} 3 & 2 & 2 & -19 \\ -6 & -1 & -4 & 38 \\ -3 & -1 & -1 & 17 \\ 9 & 1 & 4 & -47 \end{pmatrix}.$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(2,1),\ A_2(1,2),\ A_3(2,0),\ A_4(7,6),\ A_5(6,6),\ A_6(6,3),\ A_7(7,3).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(6,7),\ A_2(-2,3),\ A_3(-3,5),\ A_4(-5,4),\ A_5(7,-1),$$
$$A_6(-1,-8),\ A_7(-4,-3),\ A_8(-7,5),\ A_9(4,9).$$

Вариант 5

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(3, -3),\ B(14, 0),\ x_{\min} = 5,\ y_{\min} = -4,\ x_{\max} = 11,\ y_{\max} = 5.$$

2. Определить видимость граней тетраэдра с вершинами в точках A_i, $i = \overline{1,4}$, для ближнего наблюдателя S:

$$A_1(8,8,4),\ A_2(2,8,4),\ A_3(1,1,0),\ A_4(1,8,0),\ S(-2,\ 0,\ -5).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(6,0),\ A_2(3,5),\ A_3(7,6),\ A_4(2,3),\ A_5(2,1),\ A_6(1,1),\ A_7(2,0).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(8,3),\ A_2(2,7),\ A_3(8,-3),\ A_4(-7,-8),\ A_5(8,-5),$$
$$A_6(9,-9),\ A_7(10,-4),\ A_8(8,8),\ A_9(7,5).$$

Вариант 6

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(-3, -3),\ B(3, -1),\ C_1(5,0),\ C_2(3,4),\ C_3(-2,-1),\ C_4(-2,-2),\ C_5(4,-2).$$

2. Вычислить матрицу тела для тетраэдра с вершинами в точках A_i, $i = \overline{1,4}$:

$$A_1(5,1,2),\ A_2(9,4,3),\ A_3(0,5,1),\ A_4(2,7,5).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(3,2),\ A_2(0,0),\ A_3(3,7),\ A_4(7,1),\ A_5(1,1),\ A_6(7,7),\ A_7(5,7).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(-2,-9),\ A_2(7,-1),\ A_3(7,3),\ A_4(1,1),\ A_5(0,9),$$
$$A_6(0,7),\ A_7(-2,7),\ A_8(1,-1),\ A_9(-9,-8).$$

Вариант 7

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(3, 6), B(-4, 2), C_1(2,2), C_2(4,3), C_3(0,5), C_4(-2,5), C_5(-1,3).$$

2. Осуществить отсечение отрезка AB внешним выпуклым отсекателем с известной матрицей тела $\mathcal{C}$:

$$A(0, 0, 0), B(1, 0, 2), \mathcal{C} = \begin{pmatrix} 3 & -2 & -4 & 8 \\ 6 & 2 & -5 & -8 \\ -12 & -1 & 16 & 4 \\ 3 & 1 & -7 & 5 \end{pmatrix}.$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(6,3), A_2(7,6), A_3(5,1), A_4(7,2), A_5(4,2), A_6(7,4), A_7(4,3).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(7,-8), A_2(9,-9), A_3(9,9), A_4(4,10), A_5(7,9),$$
$$A_6(-8,7), A_7(-4,-1), A_8(1,-5), A_9(1,-3).$$

Вариант 8

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(5, -1), B(1, -6), x_{\min} = -4, y_{\min} = -5, x_{\max} = 4, y_{\max} = 0.$$

2. Осуществить отсечение отрезка AB внешним выпуклым отсекателем с известной матрицей тела $\mathcal{C}$:

$$A(3, 2, 1), B(4, 0, 2), \mathcal{C} = \begin{pmatrix} 2 & 0 & 3 & -11 \\ 5 & 7 & -10 & -10 \\ 0 & -1 & 0 & 5 \\ -4 & 0 & 1 & 15 \end{pmatrix}.$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(0,2),\ A_2(5,6),\ A_3(2,5),\ A_4(7,4),\ A_5(2,4),\ A_6(7,6),\ A_7(4,7).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(-9,2),\ A_2(-4,1),\ A_3(0,-3),\ A_4(2,-6),\ A_5(9,-6),$$
$$A_6(6,9),\ A_7(3,7),\ A_8(1,7),\ A_9(-8,8).$$

Вариант 9

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(3,\ 10),\ B(9,\ 1),\ x_{\min} = -2,\ y_{\min} = 4,\ x_{\max} = 6,\ y_{\max} = 13.$$

2. Осуществить отсечение отрезка AB внешним выпуклым отсекателем с известной матрицей тела C:

$$A(2,\ 3,\ 4),\ B(2,\ 2,\ 3),\ C = \begin{pmatrix} -5 & -3 & 9 & -3 \\ 9 & 2 & -6 & 2 \\ -1 & -4 & -5 & 30 \\ -3 & 5 & 2 & -12 \end{pmatrix}.$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(5,7),\ A_2(0,1),\ A_3(5,1),\ A_4(3,3),\ A_5(3,1),\ A_6(3,5),\ A_7(2,1).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(0,-2),\ A_2(0,-8),\ A_3(-10,8),\ A_4(-9,7),\ A_5(5,7),$$
$$A_6(7,3),\ A_7(5,-1),\ A_8(1,-2),\ A_9(1,-10).$$

Вариант 10

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, а также растеризовать результат этого отсечения:

$$A(-1,\ -2),\ B(7,\ 1),\ x_{\min} = -2,\ y_{\min} = -4,\ x_{\max} = 4,\ y_{\max} = 2.$$

2. Вычислить матрицу тела для тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$:

$$A_1(7,0,2),\ A_2(3,1,2),\ A_3(6,5,5),\ A_4(8,0,3).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(1,6),\ A_2(2,0),\ A_3(4,5),\ A_4(3,5),\ A_5(6,7),\ A_6(5,2),\ A_7(6,0).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(-5,-8),\ A_2(-5,-10),\ A_3(-6,-5),\ A_4(-2,-3),\ A_5(5,7),$$
$$A_6(8,8),\ A_7(1,-4),\ A_8(9,-9),\ A_9(-1,-6).$$

Вариант 11

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(1,-3),\ B(-1,7),\ C_1(0,5),\ C_2(-2,4),\ C_3(0,1),\ C_4(2,-1),\ C_5(3,0).$$

2. Определить видимость граней тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$, для дальнего наблюдателя $\vec{s}$:

$$A_1(5,6,0),\ A_2(0,1,10),\ A_3(0,7,2),\ A_4(5,5,8),\ \vec{s}(-5,-3,-5).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(3,1),\ A_2(5,1),\ A_3(6,2),\ A_4(2,1),\ A_5(1,2),\ A_6(6,5),\ A_7(7,1).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_8A_1$:

$$A_1(0,1),\ A_2(-2,-8),\ A_3(9,-4),\ A_4(4,2),\ A_5(-3,3),\ A_6(-8,3),\ A_7(-3,-7),\ A_8(-1,-1).$$

Вариант 12

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(0,2),\ B(3,0),\ C_1(-2,0),\ C_2(2,-2),\ C_3(3,-2),\ C_4(5,3),\ C_5(3,4).$$

2. Вычислить матрицу тела для тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$:

$$A_1(10,10,4),\ A_2(7,0,0),\ A_3(10,8,3),\ A_4(4,0,0).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(6,6),\ A_2(2,5),\ A_3(1,0),\ A_4(7,3),\ A_5(7,7),\ A_6(1,5),\ A_7(5,5).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(7,-9),\ A_2(3,8),\ A_3(0,4),\ A_4(-2,8),\ A_5(-4,2),$$
$$A_6(-6,2),\ A_7(-5,-4),\ A_8(0,-2),\ A_9(4,-8).$$

Вариант 13

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(-4,\ 0),\ B(3,\ -1),\ C_1(-2,3),\ C_2(2,-1),\ C_3(5,0),\ C_4(5,1),\ C_5(1,4).$$

2. Определить видимость граней тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$, для дальнего наблюдателя $\vec{s}$:

$$A_1(10,1,4),\ A_2(7,7,1),\ A_3(9,5,1),\ A_4(5,8,5),\ \vec{s}(2,1,1).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(5,5),\ A_2(7,2),\ A_3(1,5),\ A_4(0,4),\ A_5(0,5),\ A_6(5,3),\ A_7(1,1).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(10,6),\ A_2(-1,1),\ A_3(6,3),\ A_4(10,-2),\ A_5(-8,-7),$$
$$A_6(-8,1),\ A_7(0,-4),\ A_8(-9,3),\ A_9(-6,9).$$

Вариант 14

1. Осуществить отсечение отрезка AB внешним выпуклым отсекателем $C_1C_2C_3C_4C_5C_1$:

$$A(5,\ 3),\ B(3,\ 6),\ C_1(-2,3),\ C_2(1,-1),\ C_3(5,1),\ C_4(4,4),\ C_5(1,5).$$

2. Вычислить матрицу тела для тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$:

$$A_1(1,10,10),\ A_2(6,5,7),\ A_3(4,8,10),\ A_4(4,1,1).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(7,0),\ A_2(4,1),\ A_3(2,2),\ A_4(4,4),\ A_5(2,6),\ A_6(5,0),\ A_7(2,4).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(-5,1),\ A_2(1,3),\ A_3(0,6),\ A_4(5,4),\ A_5(-3,-8),$$
$$A_6(-8,-6),\ A_7(-10,0),\ A_8(-3,-7),\ A_9(-4,-4).$$

Вариант 15

1. Осуществить отсечение отрезка AB внешним регулярным отсекателем, ограниченным прямыми $x=x_{\min}$, $x=x_{\max}$, $y=y_{\min}$, $y=y_{\max}$, а также растеризовать результат этого отсечения:

$$A(17,\ 12),\ B(3,\ 0),\ x_{\min}=4,\ y_{\min}=3,\ x_{\max}=14,\ y_{\max}=11.$$

2. Определить видимость граней тетраэдра с вершинами в точках A_i, $i=\overline{1,4}$, для дальнего наблюдателя $\vec{s}$:

$$A_1(7,3,7),\ A_2(3,2,10),\ A_3(10,0,7),\ A_4(7,4,9),\ \vec{s}(4,1,3).$$

3. Построить выпуклую оболочку для множества точек $\{A_i\}_{i=1}^{7}$:

$$A_1(7,7),\ A_2(3,7),\ A_3(0,7),\ A_4(6,3),\ A_5(3,5),\ A_6(5,4),\ A_7(0,3).$$

4. Осуществить триангуляцию полигона $A_1A_2...A_9A_1$:

$$A_1(0,7),\ A_2(10,2),\ A_3(2,-10),\ A_4(0,5),\ A_5(1,-8),$$
$$A_6(-1,-10),\ A_7(-5,-6),\ A_8(-3,7),\ A_9(-3,-4).$$

3.3. Примерный перечень вопросов к экзамену

Ниже приведены полный перечень вопросов к экзамену. Жирным выделены вопросы, относящиеся к темам, затронутым в настоящем ЭУМК.

1. Моделирование прямой и плоскости
2. Взаимное положение графических элементов на плоскости и в пространстве
3. Плоские полигоны и их свойства
4. Расположение точек относительно полигона
5. Понятие аффинного пространства
6. Декартовы системы координат в аффинном пространстве
7. Преобразования аффинных пространств. Аффинные преобразования
8. Движения в аффинном пространстве
9. Понятие и основные типы проективных преобразований. Ортографические проекции
10. Понятие и основные типы проективных преобразований. Аксонометрические проекции
11. Понятие и основные типы проективных преобразований. Косоугольные проекции
12. Понятие и основные типы проективных преобразований. Центральные проекции
13. **Базовые растровые алгоритмы**
14. **Выпуклая оболочка множества точек. Алгоритмы построения выпуклой оболочки**
15. **Триангуляция полигонов методом хорд**
16. **Триангуляция полигонов диагоналями. Триангуляция полигонов методом отрезания ушей**
17. **Триангуляция Делоне множества точек на плоскости**
18. **Алгоритмы отсечения и удаления невидимых рёбер. Алгоритм Сазерленда-Коэна**
19. **Алгоритмы удаления невидимых граней. Метод трассировки лучей. Алгоритм художника. Алгоритм Робертса. Метод z-буфера**
20. **Основные понятия геометрической оптики. Законы геометрической оптики. Отражение и преломление светового луча**
21. **Методы построения оптических эффектов. Условия видимости оптических эффектов. Построение тени точки и других геометрических объектов**
22. **Методы построения оптических эффектов. Условия видимости оптических эффектов. Построение отражения точки и других геометрических объектов**
23. **Методы построения оптических эффектов. Условия видимости оптических эффектов. Построение преломления точки и других геометрических объектов**

24. Приближение кривых и поверхностей. Интерполяция линий и поверхностей методом Лагранжа и методом наименьших квадратов

25. Приближение кривых и поверхностей. Аппроксимация линий и поверхностей методом наименьших квадратов

26. Классификация поверхностей. Построение каркаса для каркасной поверхности

27. Классификация поверхностей. Моделирование кинематических поверхностей

28. Цветовые модели. Геометрическая интерпретация и преобразования цветовых моделей

29. Моделирование освещённости и цвета поверхностей. Функция закраски Фонга

30. Моделирование освещённости и цвета поверхностей. Методы закраски поверхностей

31. Моделирование трёхмерных тел

4. ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

4.1. Фрагмент учебной программы

Цели и задачи учебной дисциплины

Цель учебной дисциплины «Математические методы компьютерной графики» – ознакомить студентов с математическими методами графического представления разнообразных объектов, особенностями их моделирования и визуализации, обусловленными свойствами самих объектов, их расположением и сферами применения полученных изображений.

Задачи учебной дисциплины:

1. Закрепить ранее изученные и изучить новые математические модели основных геометрических объектов, их взаиморасположения и взаимодействия;
2. Изучить основные преобразования геометрических пространств, широко используемые при моделировании и визуализации компьютерной графики;
3. Сформировать навыки решения типовых задач в области компьютерной графики.

Место учебной дисциплины. В системе подготовки специалиста с высшим образованием учебная дисциплина относится **к модулю** «Компьютерная графика» компонента учреждения образования.

Связи с другими учебными дисциплинами, включая учебные дисциплины компонента учреждения высшего образования, дисциплины специализации и др.

Дисциплина «Математические методы компьютерной графики» тесно связана со следующими дисциплинами:

- Модуль «Высшая математика»: «Аналитическая геометрия», «Алгебра и теория чисел» и «Математический анализ»;
- Модуль «Дискретная математика и алгоритмы»: «Теория графов» и «Алгоритмы и структуры данных»;
- «Методы вычислений».

Методы, излагаемые в данном курсе, используются в дисциплине «Программирование компьютерной графики».

Требования к компетенциям

Освоение учебной дисциплины «Математические методы компьютерной графики» должно обеспечить формирование следующих базовых профессиональных и специализированных компетенций:

базовые профессиональные компетенции:

БПК-1. Применять аппарат дифференциального и интегрального исчисления, методы аналитической геометрии и линейной алгебры для построения математических моделей и решения прикладных задач;

специализированные компетенции:

СК-2. Применять полученные теоретические и практические навыки для решения задач компьютерной графики в профессиональной деятельности.

В результате освоения учебной дисциплины студент должен:

знать:

- типы компьютерной графики;
- основные математические модели геометрических объектов;
- методы и алгоритмы определения их расположения относительно друг друга и наблюдателя;
- особенности визуализации геометрических объектов и их свойств (цвета, освещения, прозрачности);

уметь:

- строить математические модели геометрических объектов;
- выбирать и применять методы и алгоритмы визуализации для решения типовых задач вычислительной геометрии и компьютерной графики;

владеть:

- основными методами и алгоритмами моделирования и визуализации геометрических объектов.

Структура учебной дисциплины

Дисциплина изучается во 2 семестре. Всего на изучение учебной дисциплины «Математические методы компьютерной графики» отведено:

– для очной формы получения высшего образования – 216 часов, в том числе 102 аудиторных часа, из них: лекции – 68 часов, лабораторные занятия – 30 часов, управляемая самостоятельная работа – 4 часа.

Трудоемкость учебной дисциплины составляет 6 зачетных единиц.

Форма промежуточной аттестации – экзамен.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Раздел 3. Основные алгоритмы вычислительной геометрии

Тема 3.1. Алгоритмы отсечения и удаления невидимых линий и поверхностей

Отсечение отрезков. Алгоритм Коэна-Сазерленда. Алгоритм Лианга-Барски. Алгоритм Кируса-Бека. Алгоритм Николла-Ли-Николла.

Алгоритмы отсечения полигонов. Алгоритм Сазерленда-Ходжмана. Алгоритм Вейлера-Азертона.

Алгоритмы отсечения полиэдров. Алгоритм Робертса и его модификации. Алгоритм Варнока. Алгоритм художника. Алгоритм z-буфера.

Тема 3.2. Базовые растровые алгоритмы

Введение в растеризацию линий. Алгоритм DDA-линии (цифровой дифференциальный анализатор). Алгоритм Брезенхема. Алгоритм Кастла-Питвея. Растеризация со сглаживанием (антиалиасингом). Алгоритм Ву. Растеризация окружностей и эллипсов.

Тема 3.3. Построение выпуклой оболочки

Алгоритмы построения выпуклой оболочки для множества точек на плоскости. Алгоритмы Джарвиса, Грэхема, Эндрю, «быстрой оболочки».

Тема 3.4. Триангуляция

Понятие триангуляции. Структуры данных для представления триангуляции.

Триангуляция полигона. Триангуляция хордами. Триангуляция по диагоналям. Триангуляция методом «отрезания ушей». Контролируемая триангуляция полигона.

Триангуляция множества точек. Оптимальная триангуляция. Жадный алгоритм. Триангуляция Делоне, её свойства. Классификация алгоритмов построения триангуляции Делоне. Итеративные алгоритмы. Алгоритмы построения триангуляции Делоне слиянием. Алгоритмы прямого построения триангуляции Делоне.

Тема 3.5. Алгоритмы геометрической оптики

Основные задачи и базовые законы геометрической оптики. Метод трассировки лучей. Понятие световой энергии.

Пересечение луча с поверхностью. Отражение и преломление луча. Закон Бугера прохождения света через непрозрачную среду.

Лучевые методы моделирования оптических эффектов и их свойства. Тень, отражение, преломление геометрических объектов. Геометрические и вычислительные особенности оптических эффектов.

4.2. Рекомендуемая литература

Основная

1. Математические методы компьютерной графики : электронный учебно-методический комплекс для специальности: 6-05-0533-11 «Прикладная информатика», профилизация «Информационные аналитические системы». В 3 ч. Ч. 1. Математические основы компьютерной графики / С. В. Шолтанюк ; БГУ, Фак. прикладной математики и информатики, Каф. компьютерных технологий и систем. – Минск : БГУ, 2023. – 240 с. – Режим доступа: https://elib.bsu.by/handle/123456789/303481. – Дата доступа: 18.05.2025.

2. Васильков, Д. М. Геометрическое моделирование и компьютерная графика: алгоритмы и вычислительные основы [Электронный ресурс] : курс лекций / Д. М. Васильков. – Минск : БГУ, 2011. – Режим доступа: https://elib.bsu.by/handle/123456789/27612. – Дата доступа: 18.05.2025.

3. Никулин, Е. А. Компьютерная графика. Модели и алгоритмы: учебное пособие / Е. А. Никулин. – 2-е изд., стер. – Санкт-Петербург: Лань, 2021. – 708 с.

4. Никулин, Е. А. Компьютерная графика. Оптическая визуализация : учеб. пособие для студ. напр. подготовки "Информатика и вычислительная техника" / Е. А. Никулин. – Санкт-Петербург ; Москва ; Краснодар : Лань, 2023. – 196 с.

5. Роджерс, Д. Алгоритмические основы машинной графики : Пер. с англ. / Д. Роджерс ; под ред. Ю. М. Баяковского и В. А. Галактионова – М.: Мир, 1989. – 512 с.

Дополнительная

6. Sproull, R. F. A Clipping Divider / R. F. Sproull, I. E. Sutherland // AFIPS ‘68 (Fall, part I): Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, part I. – New York: ACM, 1968. – P. 765–775. – DOI: 10.1145/1476589.1476687.

7. Newman, W. M. Principles of Interactive Computer Graphics / W. M. Newman, R. F. Sproull. – 2nd ed. – New York : McGraw-Hill, 1979. – 607 p.

8. Liang, Y.-D. A New Concept and Method for Line Clipping / Y.-D. Liang, B. A. Barsky // ACM Transactions on Graphic. – 1984. – Vol. 3, No. 1. – P. 1–22. – DOI: 10.1145/357332.357333.

9. Sobkow, M. S. A Fast Two-Dimensional Line Clipping Algorithm via Line Encoding / M. S. Sobkow, P. Pospisil, Y.-H. Yang // Computer & Graphics. – 1987. – Vol. 11, Iss. 4. – P. 459–467. – DOI: 10.1016/0097-8493(87)90061-6.

10. Cyrus, M. Generalized Two- and Three-Dimensional Clipping. / M. Cyrus, J. Beck // Computers & Graphics. – 1978. – Vol. 3, Iss. 1. – P. 23–28. – DOI: 10.1016/0097-8493(78)90021-3.

11. Nicholl, T. M. An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis / T. M. Nicholl, D. T. Lee, R. A. Nicholl // Computer Graphics. – 1987. – Vol. 21, No. 4. – P. 253–262. – DOI: 10.1145/37402.37432.

12. Sutherland, I. E. Reentrant Polygon Clipping / I. E. Sutherland, G. W. Hodgman // Communications of the ACM. – 1974. – Vol. 17, Iss. 1. – P. 32–42. – DOI: 10.1145/360767.360802.

13. Weiler, K. Hidden Surface Removal using Polygon Area Sorting / K. Weiler, P. Atherton // Computer Graphics. – 1977. – Vol. 11, Iss. 2. – P. 214–222. – DOI: 10.1145/965141.563896.

14. Bentley, J. L. Algorithms for Reporting and Counting Geometric Intersections. / J. L. Bentley, T. Ottmann // IEEE Transactions on Computers. – 1979. – Vol. C-28, No. 9. – P. 643–647. – DOI: 10.1109/TC.1979.1675432.

15. Roberts, L. G. Machine Perception of Three-Dimensional Solids : thesis … for the degree of Dr. of Philosophy / Lawrence Gilman Roberts ; Massachusetts Institute of Technology. – Cambridge, MA, 1963. – 82 l. – Mode of access: https://dspace.mit.edu/handle/1721.1/11589. – Date of access: 18.05.2025.

16. Study For Applying Computer-Generated Images to Visual Simulation : tech. report (final) / Apollo Systems, General Electric Company ; perf.: R. A. Schumacker, B. Brand, M. G. Gilliland, W. H. Sharp. – Daytona Beach, FL, 1969. – 131 p. – AFHRL-TR-69-14. – Mode of access: https://apps.dtic.mil/sti/citations/AD0700375. – Date of access: 18.05.2025.

17. Sutherland, I. E. A Characterization of Ten Hidden-Surface Algorithms / I. E. Sutherland, R. F. Sproull, R. A. Schumacker // ACM Computing Surveys. – 1974. – Vol. 6, Iss. 1. – P. 1–55. – DOI: 10.1145/356625.356626.

18. Fuchs, H. On Visible Surface Generation by A Priori Tree Structures / H. Fuchs, Z. M. Kedem, B. F. Naylor // SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques, Seattle, WA, July 14–18, 1980 / New York : Association for Computing Machinery, 1980. – P. 124–133. – DOI: 10.1145/800250.807481.

19. Catmull, E. Computer Display of Curved Surfaces / E. Catmull // Seminal Graphics: Pioneering Efforts That Shaped the Field, Volume 1 / ed.: R. Wolfe. – New York : Association for Computing Machinery, 1998. – P. 35–41. – DOI: 10.1145/280811.280920.

20. Catmull, E. E. A Subdivision Algorithm for Computer Display of Curved Surfaces : thesis ... for the degree of Dr. of Philosophy / Edwin Earl Catmull ; University of Utah. – Salt Lake City, UT, 1974. – 83 l.

21. A Hidden Surface Algorithm for Computer Generated Halftone Pictures : tech. report / University of Utah ; perf.: J. E. Warnock. – Salt Lake City, UT, 1969. – V, 26 p. – RADC-TR-69-249.

22. Bresenham, J. E. Algorithm for Computer Control of a Digital Plotter / J. E. Bresenham // IBM Systems Journal. – 1965. – Vol. 4, No. 1. – P. 25–30. – DOI: 10.1147/sj.41.0025.

23. Castle, C. M. A. An Efficient Structural Technique for Encoding ‘Best-fit’ Straight Lines // C. M. A. Castle, M. L. V. Pitteway // The Computer Journal. – 1987. – Vol. 30, Iss. 2. – P. 168–175. – DOI: 10.1093/comjnl/30.2.168.

24. Wu, X. An Efficient Antialiasing Technique / X. Wu // Computer graphics. – 1991. – Vol. 25, No. 4. – P. 143–152. – DOI: 10.1145/127719.122734.

25. Bresenham, J. E. A Linear Algorithm for Incremental Digital Display of Circular Arcs / J. E. Bresenham // Communications of the ACM. – 1977. – Vol. 20, Iss. 2. – P. 100–106. – DOI: 10.1145/359423.359432.

26. Jarvis, R. A. On the Identification of the Convex Hull of a Finite Set of Points in the Plane / R. A. Jarvis // Information Processing Letters. – 1973. – Vol. 2, Iss. 1. – P. 18–21. – DOI: 10.1016/0020-0190(73)90020-3.

27. Graham, R. L. An Efficient Algorith for Determining the Convex Hull of a Finite Planar Set / R. L. Graham // Information Processing Letters. – 1972. – Vol. 1, Iss. 4. – P. 132–133. – DOI: 10.1016/0020-0190(72)90045-2.

28. Andrew, A. M. Another Efficient Algorithm for Convex Hulls in Two Dimensions / A. M. Andrew // Information Processing Letters. – 1979. – Vol. 9, Iss. 5. – P. 216–219. – DOI: 10.1016/0020-0190(79)90072-3.

29. Gamby, A. N. Convex-Hull Algorithms: Implementation, Testing, and Experimentation / A. N. Gamby, J. Katajainen // Algorithms. – 2018. – Vol. 11, Iss. 12. – Article No. 195. – 27 p. – DOI: 10.3390/a11120195.

30. Greenfield, J. S. A Proof for a QuickHull Algorithm / J. S. Greenfield // Electrical Engineering and Computer Science - Technical Reports. – 1990. – Article No. 65. – 21 p. – Mode of access: https://surface.syr.edu/eecs_techreports/65/. – Date of access: 18.05.2025.

31. Скворцов, А. В. Алгоритмы построения и анализа триангуляции / А. В. Скворцов, Н. С. Мирза. – Томск: Изд-во Том. ун-та, 2006. – 168 с.

32. Никулин, Е. А. Модернизация алгоритма контролируемой триангуляции полигонов / Е. А. Никулин // Труды Нижегородского государственного технического университета им. Р.Е. Алексеева. – 2011. – №2 (87). – С. 62–67.

33. Santos, F. A better upper bound on the number of triangulations of a planar point set / F. Santos, R. Seidel // Journal of Combinatorial Theory, Series A. – 2003. – Vol. 102, Iss. 1. – P. 186–193. – DOI: 10.1016/S0097-3165(03)00002-5.

34. Mulzer, W. Minimum-weight triangulation is NP-hard / W. Mulzer, G. Rote // Journal of the ACM. – 2008. – Vol. 55, Iss. 2. – Article No. 11. – 29 p. – DOI: 10.1145/1346330.1346336.

35. Delaunay, B. Sur la sphère vide. A la mémoire de Georges Voronoï / B. Delaunay // Bulletin de l'Académie des Sciences de l'Union des Républiques Soviétiques Socialistes. VII série. Classe des sciences mathématiques et naturelles. – 1934. – Iss. 6. – P. 793–800.

36. Xia, G. The Stretch Factor of the Delaunay Triangulation Is Less than 1.998 / Ge Xia // SIAM Journal on Computing. – 2013. – Vol. 42, Iss. 4. – P. 1620–1659. – DOI: 10.1137/110832458.

37. Lee, D. T. Two Algorithms for Constructing a Delaunay Triangulation / D. T. Lee, B. J. Schachter // International Journal of Computer & Information Sciences. – 1980. – Vol 9. – P. 219–242. – DOI: 10.1007/BF00977785.

ПРИЛОЖЕНИЕ 1

ДОКАЗАТЕЛЬСТВО КОРРЕКТНОСТИ АЛГОРИТМА КАСТЛА-ПИТВЕЯ

Алгоритм Кастла-Питвея построения кода Ротштейна для растеризации отрезка AB с началом в точке $A(0, 0)$ и концом в точке $B(x_B, y_B)$, $x_B > y_B > 0$, описан в подразделе 1.3.4. Доказательство его корректности приведено отдельно в данном приложении из-за громоздкости, а также в силу того, что для его понимания необходимы некоторые предварительные сведения из арифметики.

1.1. Цепные дроби и их построение

Цепной, или **непрерывной дробью** называется выражение следующего вида:

$$n_0 + \cfrac{1}{n_1 + \cfrac{1}{n_2 + \cfrac{1}{n_3 + \ldots}}},$$

где $n_0 \in \mathbb{Z}$, n_1, n_2, n_3, … – некоторые натуральные числа. Они называются **элементами** цепной дроби. Цепная дробь обозначается следующим образом: $[n_0; n_1, n_2, n_3, \ldots]$. Цепная дробь может быть конечной $\left(\text{например, } \frac{2}{9} = \cfrac{1}{4 + \frac{1}{2}},\ \frac{7}{5} = 1 + \cfrac{1}{2 + \frac{1}{2}} \right)$ или бесконечной $\left(\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}},\ \pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \ldots}} \right)$, причём у числа $\sqrt{2}$ дальнейшие элементы цепной дроби все равны 2, в цепной дроби, равной π, закономерность её элементов пока не обнаружена. Число элементов конечной цепной дроби (без учёта целой части) будем называть её **порядком**.

Ясно, что конечная цепная дробь равна некоторому рациональному числу, так как при помощи последовательного домножения на её элементы числителей

и знаменателей внутренних дробей можно добиваться их уменьшения, пока не получится обыкновенная дробь (например, $\dfrac{1}{2+\dfrac{1}{1+\dfrac{1}{2}}} = \dfrac{1}{2+\dfrac{2}{2+1}} = \dfrac{1}{2+\dfrac{2}{3}} = \dfrac{3}{6+2} = \dfrac{3}{8}$). Верно и обратное, т.е. всякое рациональное число представимо в виде конечной цепной дроби. Для доказательства этого факта используется алгоритм Евклида нахождения наибольшего общего делителя (НОД) двух натуральных чисел.

1.1.1. Алгоритм Евклида и его связь с цепными дробями

Пусть даны два натуральных числа a и b, причём $a > b$. Тогда их НОД (т.е. наибольшее натуральное число, которое делит нацело оба этих числа) можно найти по **алгоритму Евклида**, который заключается в следующем:

1. Число a делится с остатком на b:

$$a = n_0 b + r_1, \tag{1.1}$$

где $n_0 \in \mathbb{N}$ – частное, $r_1 \in \mathbb{Z},\ 0 \le r_1 \le b-1$ – остаток.

2. Если $r_1 = 0$, то это означает, что число a разделилось нацело на b, откуда следует, что НОД(a, b) = b – алгоритм завершает свою работу. В противном случае b делится с остатком на ненулевой остаток r_1:

$$b = n_1 r_1 + r_2, \tag{1.2}$$

где $n_1 \in \mathbb{N},\ r_2 \in \mathbb{Z},\ 0 \le r_2 \le r_1 - 1.$

3. Пока не получим нулевой остаток, последовательно делим полученные остатки друг на друга:

$$
\begin{aligned}
r_1 &= n_2 r_2 + r_3, \\
r_2 &= n_3 r_3 + r_4, \\
&\ldots \\
r_{k-2} &= n_{k-1} r_{k-1} + r_k, \\
r_{k-1} &= n_k r_k.
\end{aligned}
\tag{1.3}
$$

Нулевой остаток должен получиться рано или поздно, так как справедливы неравенства $b > r_1 > r_2 > \ldots > r_k$, а строго убывающая последовательность натуральных чисел не может быть бесконечной[1].

4. Искомый наибольший общий делитель принимается равным последнему ненулевому остатку r_k.

Рассмотрим два взаимно простых числа a и b, т.е. НОД(a, b) = 1, причём $a > b$. Применяя к ним алгоритм Евклида, можно по формулам (1.1) – (1.3) вычислить числа n_i, $i = \overline{0,k}$. Рассмотрим дробь $\frac{a}{b}$:

$$
\frac{a}{b} = \frac{n_0 b + r_1}{b} = n_0 + \frac{r_1}{b} = n_0 + \frac{1}{b/r_1} = n_0 + \frac{1}{(n_1 r_1 + r_2)/r_1} =
$$

$$
= n_0 + \frac{1}{n_1 + \dfrac{r_2}{r_1}} = \ldots = n_0 + \cfrac{1}{n_1 + \cfrac{1}{n_2 + \cfrac{1}{\ldots + \cfrac{1}{n_k}}}}.
$$

Таким образом, получена конечная цепная дробь для произвольного рационального числа $\frac{a}{b}$.

1.1.2. Свойства конечных цепных дробей

В перечисленных ниже свойствах подразумевается, что дроби являются несократимыми.

1. Для всякого рационального числа существует ровно два представления в виде цепной дроби:

[1] Подобные рассуждения получили название «метод бесконечного спуска»

$$\frac{a}{b}=[n_0;n_1,n_2,...,n_{k-1},n_k]=[n_0;n_1,n_2,...,n_{k-1},n_k-1,1]. \qquad (1.4)$$

Несложно проверить эквивалентность двух записей:

$$n_0+\cfrac{1}{n_1+\cfrac{1}{n_2+\cfrac{1}{...+\cfrac{1}{n_k}}}}=n_0+\cfrac{1}{n_1+\cfrac{1}{n_2+\cfrac{1}{...+\cfrac{1}{(n_k-1)+\cfrac{1}{1}}}}}.$$

Других цепных дробей, равных $\frac{a}{b}$, нет в силу единственности частных n_0, n_1, …, n_k, полученных при работе алгоритма Евклида. ■

Из двух записей в формуле (1.4) чаще используется первая. Иногда конечная цепная дробь с последним элементом, бóльшим единицы, называется её каноническим представлением.

2. Связь между цепными дробями, равными взаимно обратные числам: $\frac{a}{b}=[n_0;n_1,...,n_k]\Leftrightarrow\frac{b}{a}=[0;n_0,n_1,...,n_k]$. Выполнение этого свойства легко проверить, применив операцию обращения к цепной дроби.

3. Рассмотрим цепную дробь $\frac{p}{q}=[0;n_1,...,n_k]$, которая, очевидно, является рациональным числом в интервале (0, 1). Если из этой цепной дроби убрать несколько последних элементов, получим новую цепную дробь $\frac{p_i}{q_i}=[0;n_1,...,n_i]$. Такая дробь называется **подходящей** для исходной цепной дроби. Подходящие дроби обладают рядом уникальных свойств, однако приведём здесь только одно из них: подходящая дробь больше исходной цепной дроби, если её порядок i нечётный, и меньше, если её порядок чётный.

◆ Ясно, что выполняется следующее неравенство:

$$n_i<n_i+\cfrac{1}{n_{i+1}+\cfrac{1}{...+\cfrac{1}{n_k}}}, \qquad (1.5)$$

что эквивалентно $[n_i;]<[n_i;n_{i+1},...,n_k]$. При обращении обеих частей этого неравенства знак меняется: $[0;n_i]>[0;n_i,n_{i+1},...,n_k]$. Добавляя теперь к обеим частям по натуральному числу n_{i-1}, имеем $[n_{i-1};n_i]>[n_{i-1};\ \ n_i,n_{i+1},...,n_k]$. Повторяя эту процедуру в общей сложности i раз, получим, что знак неравенства (1.5) поменялся i раз, откуда и следует выполнение свойства. ■

4. При уменьшении последнего элемента дроби $[n_0;n_1,n_2,...,n_{k-1},n_k]$ происходит уменьшение всего числа, если порядок k чётный, либо увеличение, если k нечётно; при увеличении последнего элемента дроби подчёркнутые слова следует изменить на противоположные по значению:

$$[n_0;n_1,...,n_{k-1},n'_k]-[n_0;n_1,...,n_{k-1},n_k]\begin{cases}>0 \text{ при чётном } k \text{ и } n'_k>n_k,\\ <0 \text{ при чётном } k \text{ и } n'_k<n_k,\\ <0 \text{ при нечётном } k \text{ и } n'_k>n_k,\\ >0 \text{ при нечётном } k \text{ и } n'_k<n_k.\end{cases}$$

Например, $\frac{1}{2}>\frac{1}{3}>\frac{1}{4}$, $\frac{1}{1+\frac{1}{2}}<\frac{1}{1+\frac{1}{3}}<\frac{1}{1+\frac{1}{4}}$. Это свойство следует из предыдущего.

5. При добавлении нового элемента цепной дроби она увеличится, если её исходный порядок был чётным, и уменьшится, если её порядок был нечётным:

$$[n_0;n_1,n_2,...,n_{k-1},n_k,n_{k+1}]-[n_0;n_1,n_2,...,n_{k-1},n_k]\begin{cases}>0 \text{ при чётном } k,\\ <0 \text{ при нечётном } k.\end{cases}$$

Данное свойство непосредственно следует из свойства 3.

1.2. Связь цепных дробей с алгоритмом Кастла-Питвея

1.2.1. Алгоритм медиантных сближений

Глядя на рисунок 1.69, можно заметить, что в ходе работы алгоритма Кастла-Питвея происходит постепенное построение целого растрированного отрезка, причём текущее построение хранится в одной из переменных m_1 и m_2.

Обозначим через (x_1, y_1), (x_2, y_2) координаты точек концов отрезков m_1 и m_2 соответственно. Тогда в результате следующей итерации будет построен отрезок с концом в точке с координатами $(x_1 + x_2, y_1 + y_2)$. Докажем первоначально, что выполняя определённые действия согласно алгоритму Кастла-Питвея, можно получить произвольную точку с координатами (x_B, y_B), $x_B > y_B > 0$, НОД$(x_B, y_B) = 1$, если исходные координаты равны (1, 0) и (1, 1).

Обозначим через k_1 и k_2 угловые коэффициенты прямых, проходящих соответственно через точки (x_1, y_1) и (x_2, y_2) (на рисунке 1.69 лучи, лежащие на этих прямых, обозначены пунктирными линиями), через k – угловой коэффициент прямой, проходящей через начало координат и точку $(x_1 + x_2, y_1 + y_2)$. Тогда справедливы равенства $k_1 = \frac{y_1}{x_1}$, $k_2 = \frac{y_2}{x_2}$. Определим операцию, аналогичную конкатенации двух кодов Ротштейна, следующим образом:

$$k = k_1 \oplus k_2 = \frac{y_1 + y_2}{x_1 + x_2}.$$

Число k, определяемое по такой формуле, называется **медиантой** двух дробей k_1 и k_2. Нетрудно убедиться, что $k_1 < k_1 \oplus k_2 < k_2$. В зависимости от того, в который из интервалов $(k_1, k_1 \oplus k_2)$ и $(k_1 \oplus k_2, k_2)$ попадает число $k_B = \frac{y_B}{x_B}$, те же рассуждения применяются к концам этого интервала. Тем самым обеспечивается сближение двух лучей с текущими угловыми коэффициентами к искомой точке (x_B, y_B). Таким образом, из алгоритма Кастла-Питвея следует **алгоритм медиантных сближений** к числу k_B путём построения последовательности медиант, которая должна рано или поздно дать дробь $k_B = \frac{y_B}{x_B}$:

1. Начальные дроби равны $k_1 := \frac{0}{1}$, $k_2 := \frac{1}{1}$.
2. Вычисляется медианта $k := k_1 \oplus k_2$.
3. Если $k = k_B$, то алгоритм завершает работу.
4. Если $k > k_B$, то происходит переопределение верхнего ограничения: $k_2 := k$. Происходит возврат к пункту 2.
5. Если $k < k_B$, то происходит переопределение нижнего ограничения: $k_1 := k$. Происходит возврат к пункту 2.

Пример работы этого алгоритма при $x_B = 22$, $y_B = 9$:

$$k_B = \frac{9}{22}: \quad \begin{array}{c|c|c|c|c|c|c|c|c} k_2 & \frac{1}{1} & \frac{1}{2} & - & - & \frac{3}{7} & \frac{5}{12} & \frac{7}{17} & \frac{9}{22} \\ \hline k_1 & \frac{0}{1} & - & \frac{1}{3} & \frac{2}{5} & - & - & - & \end{array} \tag{1.6}$$

В левом столбце выписаны начальные значения верхнего и нижнего ограничений на дробь k_B, в последующих столбцах – медианты значений из предыдущего столбца (прочерк означает, что значение переменной на этой итерации не изменилось), которые помещаются в верхнюю строку, если она оказалась больше дроби k_B, либо в нижнюю, если она оказалась меньше неё. Последняя медианта оказалась равна k_B, таким образом, алгоритм завершён.

1.2.2. Доказательство корректности

Для доказательства корректности алгоритма медиантных сближений необходимо доказать следующие два пункта:

- все полученные дроби являются несократимыми,
- рано или поздно действительно получится исходный угловой коэффициент k_B.

Докажем ещё одно свойство цепных дробей: медианта цепной дроби $[0; n_1,...,n_{k-1},n_k]$ и подходящей для неё дроби $[0;n_1,...,n_{k-1}]$ равна $[0;n_1,...,n_{k-1},n_k+1]$.

◆ Обозначая первую дробь через $\frac{b}{a}$, вторую – через $\frac{q}{p}$, $a > b$, $p > q$, НОД(a, b) = НОД(p, q) = 1. Если выполняются неравенства $n_{k-1} \neq 1$, $n_k \neq 1$, то согласно алгоритму Евклида цепные дроби строятся по следующим формулам:

$$\begin{array}{cc} a = n_1 b + r_1, & p = n_1 q + r_1', \\ b = n_2 r_1 + r_2, & q = n_2 r_1' + r_2', \\ r_1 = n_3 r_2 + r_3, & r_1' = n_3 r_2' + r_3', \\ \ldots & \ldots \\ r_{k-4} = n_{k-2} r_{k-3} + r_{k-2} & r_{k-4}' = n_{k-2} r_{k-3}' + 1, \\ r_{k-3} = n_{k-1} r_{k-2} + 1, & r_{k-3}' = n_{k-1}. \\ r_{k-2} = n_k, & \end{array} \tag{1.7}$$

Складывая равенства, стоящие в одной строке, а также добавляя по единице к обеим частям нижнего равенства первого столбца, получим следующие:

$$a+p=n_1(b+q)+(r_1+r_1'),$$
$$b+q=n_2(r_1+r_1')+(r_2+r_2'),$$
$$r_1+r_1'=n_3(r_2+r_2')+(r_3+r_3'),$$
$$\ldots$$
$$r_{k-4}+r_{k-4}'=n_{k-2}(r_{k-3}+r_{k-3}')+r_{k-2}+1,$$
$$r_{k-3}+r_{k-3}'=n_{k-1}(r_{k-2}+1)+1,$$
$$r_{k-2}+1=n_k+1.$$

Таким образом, получим результат работы алгоритма Евклида для чисел $a+p$ и $b+q$, также оказавшихся взаимно простыми. Отсюда и следует требуемое:

$$\frac{1}{n_1+\dfrac{1}{\ldots+\dfrac{1}{n_{k-1}+\dfrac{1}{n_k}}}}\oplus\frac{1}{n_1+\dfrac{1}{\ldots+\dfrac{1}{n_{k-1}}}}=\frac{b}{a}\oplus\frac{q}{p}=\frac{b+q}{a+p}=\frac{1}{n_1+\dfrac{1}{\ldots+\dfrac{1}{n_{k-1}+\dfrac{1}{n_k+1}}}}.$$

Заметим, что формулы (1.7) справедливы и в случае, когда $n_{k-1}=1$ или $n_k=1$. Таким образом, свойство полностью доказано. ■

Возвращаясь к алгоритму медиантных сближений, представим угловой коэффициент k_B в виде цепной дроби: $k_B=[0;n_1,\ldots,n_k]$. Вначале согласно алгоритму получаются цепные дроби 1-ого порядка $\frac{1}{2}$, $\frac{1}{3}$, …, $\frac{1}{n_1}$, бóльшие k_B, причём дробь $\frac{1}{n_1}=[0;n_1]$ является подходящей для k_B. Затем идёт дробь $\frac{1}{n_1+1}$, которая уже меньше, чем k_B, и которую можно представить в виде цепной дроби 2-ого порядка: $\frac{1}{n_1+1}=\frac{1}{n_1+\frac{1}{1}}=[0;n_1,1]$. Легко видеть, что ранее полученная дробь $\frac{1}{n_1}=[0;n_1]$ является подходящей и для неё, поэтому следующие вычисляемые дроби согласно доказанному выше свойству представимы в виде цепных дробей 2-ого порядка: $\frac{1}{n_1+\frac{1}{2}}$, $\frac{1}{n_1+\frac{1}{3}}$, …, $\frac{1}{n_1+\frac{1}{n_2}}$. Все они меньше исходного углового коэффициента k_B по свойствам 3 и 4 из

пункта 1.1.2. Получили новую подходящую дробь $\dfrac{1}{n_1+\dfrac{1}{n_2}}$, после которой уже идут цепные дроби 3-его порядка: $\dfrac{1}{n_1+\dfrac{1}{n_2+\dfrac{1}{1}}}$, $\dfrac{1}{n_1+\dfrac{1}{n_2+\dfrac{1}{2}}}$, …, $\dfrac{1}{n_1+\dfrac{1}{n_2+\dfrac{1}{n_3}}}$. Это продолжается до тех пор, пока не получится искомая дробь k_B, которая выражается некоторой конечной цепной дробью.

С учётом всех этих замечаний схему (1.6) можно представить несколько в ином виде:

$$k_B=\frac{9}{22}=\frac{1}{2+\dfrac{1}{2+\dfrac{1}{4}}}$$

k_2	$\frac{1}{1}$	$\frac{1}{2}$	–	–	$\frac{1}{2+\frac{1}{2+\frac{1}{1}}}$	$\frac{1}{2+\frac{1}{2+\frac{1}{2}}}$	$\frac{1}{2+\frac{1}{2+\frac{1}{3}}}$	$\frac{1}{2+\frac{1}{2+\frac{1}{4}}}$
k_1	$\frac{0}{1}$	–	$\frac{1}{2+\frac{1}{1}}$	$\frac{1}{2+\frac{1}{2}}$	–	–	–	–

Из этой схемы видно, что в верхней строке расположены вначале n_1-1 дробей (не считая самый левый столбец), затем в нижней – n_2, затем снова в верхней – n_3 дроби и т.д., пока не вычислена дробь k_B. Это согласуется с алгоритмом Кастла-Питвея, в котором, как нетрудно заметить, происходит нахождение НОД чисел x_B и y_B по алгоритму Евклида на основе вычитаний, причём вначале происходит n_1-1 вычитание из переменной *x* переменной *y* (не считая самую первую операцию вычитания, не входящую в условный цикл), затем n_2 вычитания из переменной *y* переменной *x* и т.д. Таким образом, доказано, что алгоритм Кастла-Питвея даёт код Ротштейна, который, вообще говоря, кодирует некоторую ломанную линию с началом в начале координат и с концом в точке с нужными координатами (x_B, y_B).

1.3. Построение кода Ротштейна в алгоритме Кастла-Питвея

Для доказательства того, что алгоритм Кастла-Питвея строит именно растеризацию отрезка, причём такую же, как и по Брезенхему, докажем вначале некоторые леммы.

Лемма 1. Две дроби k_- и k_+, полученные по алгоритму медиантных сближений, медианта которых даёт угловой коэффициент k_B исходного отрезка (т.е. две дроби из последнего столбца схемы (1.6)), являются самыми близкими приближениями соответственно снизу и сверху из всех дробей со знаменателем, меньшим x_B.

◆ Для доказательства этого факта обратимся к **последовательности Фарея** – это возрастающая последовательность из всех несократимых дробей со знаменателем, не превышающим некоторое число n, из отрезка [0, 1]. При заданном натуральном n последовательность Фарея для этого числа обозначается $F(n)$. Например, для $n = 6$ получается такая последовательность:

$$F(6) = \left\{\frac{0}{1}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{1}{1}\right\}.$$

Последовательность Фарея $F(n)$ можно построить по следующему алгоритму, который является модификацией алгоритма медиантных сближений:

1. Крайние члены последовательности равны $k_1 := \frac{0}{1}$ и $k_2 := \frac{1}{1}$.
2. Вычисляется медианта $k = k_1 \oplus k_2$.
3. Если знаменатель дроби k оказался не больше n, то она вставляется между k_1 и k_2 в искомую последовательность, а также происходит переопределение $k_2 := k$.
4. Если знаменатель дроби k превысил n, то она пропускается, а переменным k_1 и k_2 присваиваются значения, следующие за ними в уже имеющейся последовательности.
5. Если $k_1 = \frac{1}{1}$, то алгоритм завершает работу.
6. Возврат к шагу 2.

Отметим, что этот алгоритм обладает свойствами, доказанными в пункте 1.2.2, т.е. в результате его работы получаются несократимые дроби со знаменателями, не бóльшими n. Схема построения последовательности Фарея на примере последовательности $F(6)$ приведена на рисунке 1.1, где красной точкой обозначено текущее значение переменной k_1, синей – значение k_2.

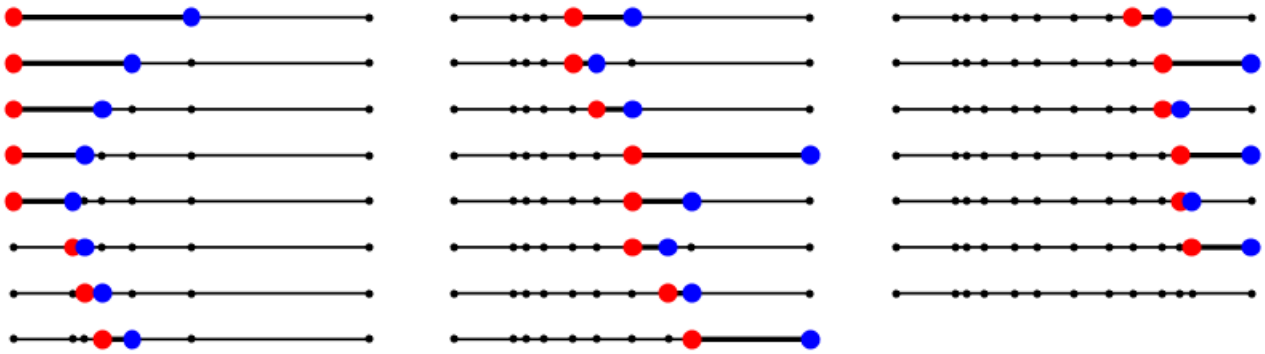


Рисунок 1.1 – Построение последовательности Фарея на единичном отрезке

Утверждение леммы 1 эквивалентно тому, что две дроби должны быть соседними для k_B в последовательности $F(x_B)$. Если это было бы не так, то, например, между дробями k_- и k_B находилась бы по крайней мере ещё одна дробь, которая равнялась бы их медианте: $k'_- = k_- \oplus k_B$. У этой несократимой дроби знаменатель превышал бы число x_B – противоречие. Таким образом, лемма 1 доказана. ■

Следующие леммы сформулированы и доказаны самими Кастлом и Питвеем непосредственно в их статье [22]. Далее через $L(x, y)$ будем обозначать код Ротштейна для растрированного отрезка с началом в начале координат и концов в точке с координатами (x, y).

Лемма 2. Если несократимая дробь $\frac{y_+}{x_+}$ является лучшим приближением сверху для $\frac{y_B}{x_B}$ среди всех дробей со знаменателем, меньшим x_B, то первые x_+ элементов кода Ротштейна растрированного отрезка AB с началом в точке $A(0, 0)$ и концом в точке $B(x_B, y_B)$ равны $\sim L(x_+, y_+)$, т.е. совпадают с перевёрнутым кодом Ротштейна для растрированного отрезка, проведённого от точки A к точке $B_+(x_+, y_+)$.

◆ Рассмотрим два отрезка AB и AB_+ с концами в точках с целочисленными координатами. Обозначим точку $B'_+\left(x_+, \frac{y_B}{x_B}x_+\right)$, которая является пересечением отрезка AB со столбцом растровой сетки, на которой расположена точка B_+ . Ясно, что треугольник $AB_+B'_+$ не содержит целочисленных точек, так как иначе можно было бы подобрать лучшее приближение к дроби $\frac{y_B}{x_B}$, нежели $\frac{y_+}{x_+}$ (Рисунок 1.2).

Не будет содержать треугольник $AB_+B'_+$ и точек с координатами вида $\left(i, j+\frac{1}{2}\right)$, $i, j \in \mathbb{Z}$. Если бы такая точка лежала внутри треугольника $AB_+B'_+$ (Рисунок 1.3), то, откладывая отрезок $B''_+C' = AB''_+$ на луче AB''_+, получим

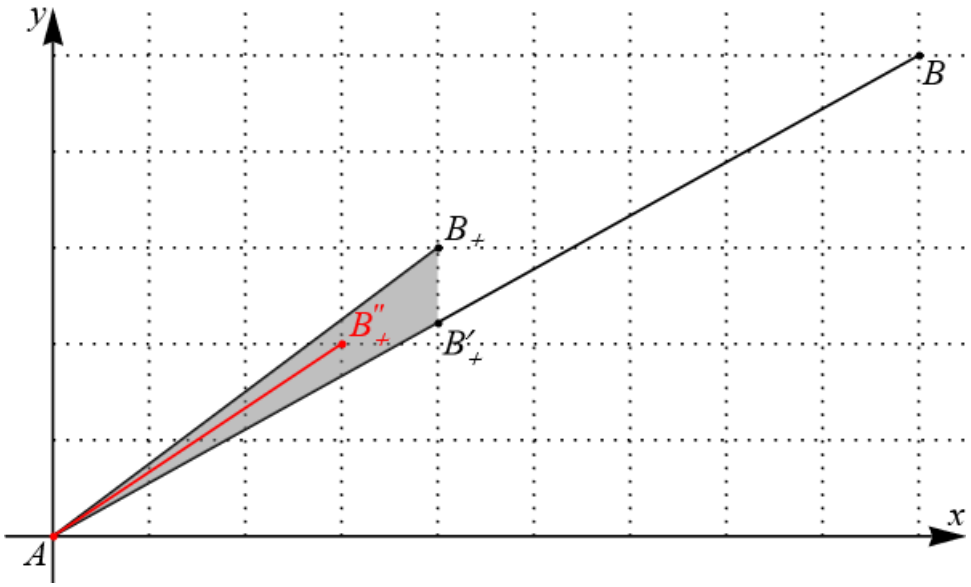


Рисунок 1.2 – Наличие целочисленной точки B''_+ внутри треугольника приводит к появлению отрезка, более близкого к исходному

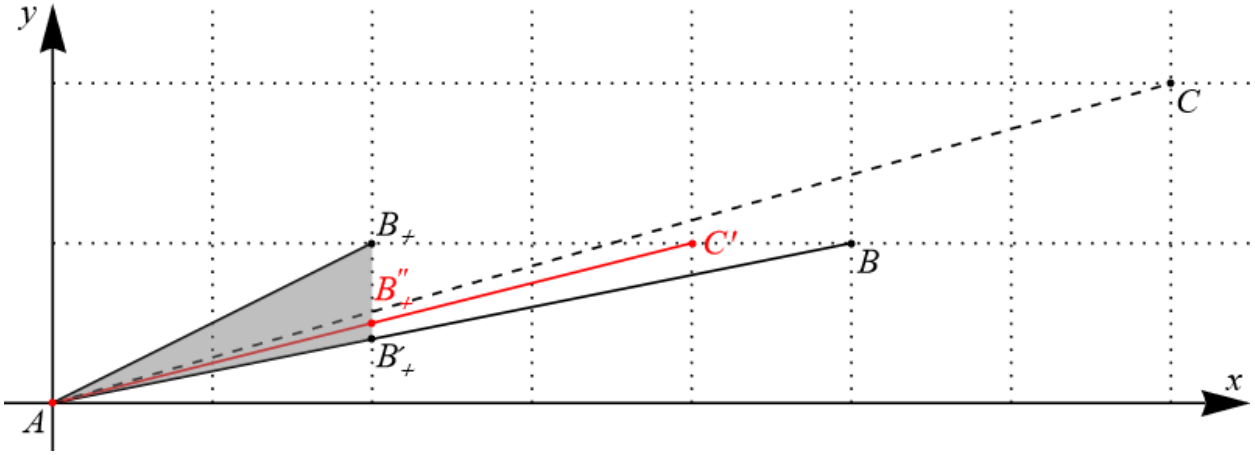


Рисунок 1.3 – Наличие точки B''_+ с целочисленной абсциссой и ординатой с дробной частью 1/2 внутри треугольника приводит к появлению целочисленного отрезка, более близкому к следующей медианте

целочисленный отрезок AC' с концом в точке $C'(2i, 2j+1)$ с взаимно простыми координатами (так как отрезок AB''_+ не проходит через целочисленные точки в силу доказанного выше свойства треугольника $AB_+B'_+$), являющийся более близким, чем хотя бы один из отрезков AB или AB''_+, к отрезку AC, конец которого $C(x_B + x_+, y_B + y_+)$ удовлетворяет равенству $\overrightarrow{AC} = \overrightarrow{AB} + \overrightarrow{AB_+}$. С учётом несократимости дробей $\frac{y_B}{x_B}$ и $\frac{y_+}{x_+}$ получим противоречие лемме 1.

Итак, имеем, что в силу отсутствия в треугольнике $AB_+B'_+$ точек с координатами вида (i, j) и $\left(i, j+\frac{1}{2}\right)$, $i, j \in \mathbb{Z}$, ординаты каждой пары

пересечений $\left(i, \frac{y_B}{x_B} i\right)$ и $\left(i, \frac{y_+}{x_+} i\right)$ двух отрезков со столбцами растровой сетки $x = i,\ i = \overline{1, x_+}$, имеют равные целые части, а их дробные части одновременно лежат в одном из промежутков: $\left[0, \frac{1}{2}\right]$ либо $\left[\frac{1}{2}, 1\right)$. Во втором случае оба пересечения округляются к пикселю $\left(i, \left\lceil \frac{y_B}{x_B} i \right\rceil\right) = \left(i, \left\lceil \frac{y_+}{x_+} i \right\rceil\right)$. В первом случае имеет округление в нижнюю сторону: $\left(i, \left\lfloor \frac{y_B}{x_B} i \right\rfloor\right) = \left(i, \left\lfloor \frac{y_+}{x_+} i \right\rfloor\right)$. Это равенство, однако, может нарушиться в случае, когда отрезок AB''_+ проходит через точку с координатами вида $\left(i, j + \frac{1}{2}\right)$. Этого можно избежать, используя свойство квазисимметричности кода Ротштейна, описанное в пункте 1.3.4. Согласно нему, при перевороте кода Ротштейна получается та же растеризация для отрезка за тем исключением, что при прохождении отрезка ровно посередине между двумя пикселями происходит округление не вверх, а вниз (см. рисунок 1.68). Пример применения этого свойства продемонстрирован на рисунке 1.4 (точки на нём имеют координаты $B(5, 1)$ и $B_+(4,1)$). Таким образом, лемма 2 доказана. ■

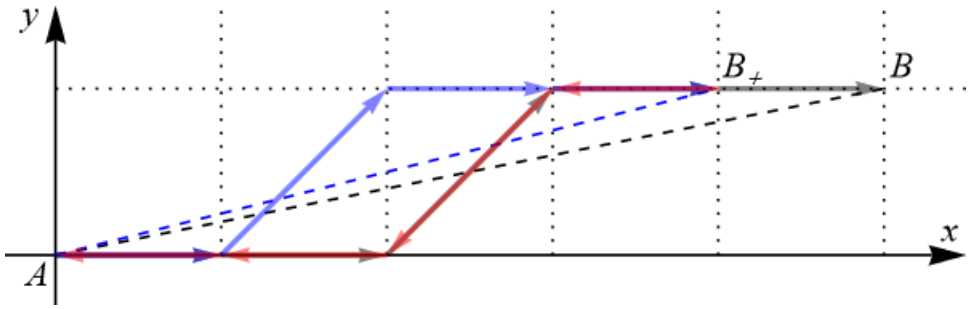


Рисунок 1.4 – Переворот кода Ротштейна для обеспечения округления вниз

Лемма 3. Если несократимая дробь $\frac{y_-}{x_-}$ является лучшим приближением снизу для $\frac{y_B}{x_B}$ среди всех дробей со знаменателем, меньшим x_B, то первые x_- элементов кода Ротштейна растрированного отрезка AB с началом в точке $A(0, 0)$ и концом в точке $B(x_B, y_B)$ равны $L(x_-, y_-)$, т.е. совпадают с кодом Ротштейна для растрированного отрезка, проведённого от точки A к точке $B_-(x_-, y_-)$. Эта лемма доказывается аналогично лемме 2, за исключением замечания с обоснованием переворота кода Ротштейна.

Лемма 4. Код Ротштейна растрированного отрезка AB равен $\sim L(x_+, y_+) \oplus \sim L(x_-, y_-)$.

◆ Согласно лемме 1, дроби $k_- = \frac{y_-}{x_-}$ и $k_+ = \frac{y_+}{x_+}$ должны получаться в результате алгоритма медиантных сближений. Отсюда следует, что их медианта равна дроби k_B, а $\overrightarrow{AB_+} + \overrightarrow{AB_-} = \overrightarrow{AB_-} + \overrightarrow{AB_+} = \overrightarrow{AB}$ (Рисунок 1.5). Из доказательства леммы 2 также следует, что внутри параллелограмма AB_+BB_- нет целочисленных точек, следовательно, искомый растрированный отрезок представим в виде кода Ротштейна, являющегося конкатенацией кодов для отрезков AB' и $B'B$, где $B'\left(x_+, \frac{y_B}{x_B} x_+\right)$ – точка на отрезке AB с абсциссой, равной абсциссе точки B_+. Растеризация отрезка AB', согласно лемме 2, представляется кодом $\sim L(x_+, y_+)$. Код для отрезка $B'B$, нетрудно заметить, равен $\sim L\left(x_-, \frac{y_B}{x_B} x_-\right)$, т.е. перевёрнутому коду для отрезка AB'', где точка $B''\left(x_-, \frac{y_B}{x_B} x_-\right)$ является точкой на отрезке AB с абсциссой, равной абсциссе точки B_-. В свою очередь, согласно лемме 3 имеем $\sim L\left(x_-, \frac{y_B}{x_B} x_-\right) = \sim L(x_-, y_-)$. Таким образом, получается, что код $\sim L(x_+, \ y_+) \oplus \sim L(x_-, y_-)$. является кодом Ротштейна для отрезка AB, что и требовалось доказать. ■

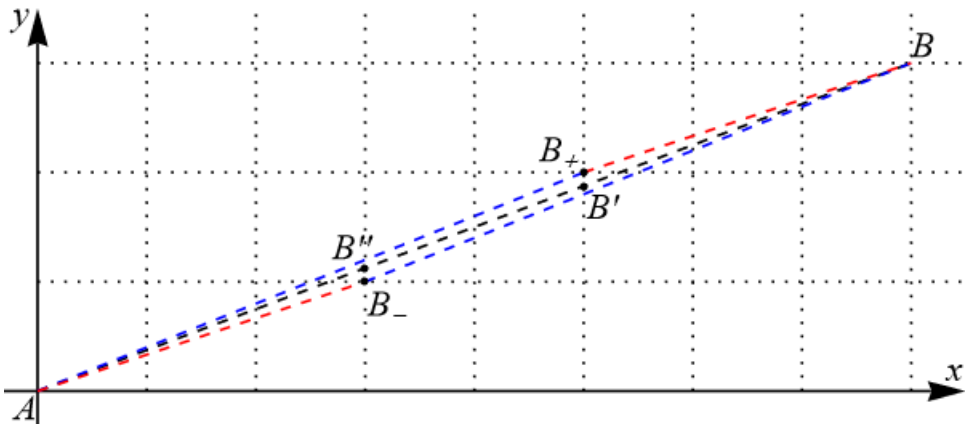


Рисунок 1.5 – Построение отрезка AB при помощи его сближений AB_+ и AB_-

Возвращаясь к самому алгоритму Кастла-Питвея, исходя из всего изложенного выше, получим следующее:

1. Начальные приближения снизу и сверху равны соответственно $\frac{y_-}{x_-} := \frac{0}{1}$, $\frac{y_+}{x_+} := \frac{1}{1}$. Согласно леммам 2 и 3, в переменных m_1 и m_2 должны храниться коды Ротштейна $L(x_-, y_-)$ и $\sim L(x_+, y_+)$ соответственно. Начальные значения, очевидно, равны $L(1, 0) = \text{S}$, $\sim L(1, 1) = \text{D}$.
2. Далее на $n_1 - 1$ итерации происходит переопределение верхнего приближения: $\frac{y_+}{x_+} := \frac{y_-}{x_-} \oplus \frac{y_+}{x_+}$. Переменной m_2 должен быть присвоен код $\sim L(x_- + x_+, y_- + y_+) = \sim(\sim L(x_+, y_+) \oplus \sim L(x_-, y_-)) = \sim(m_2 \oplus \sim m_1) = m_1 \oplus \oplus \sim m_2$.
3. Далее на n_2 итерациях переопределяется нижнее приближение: $\frac{y_-}{x_-} := \frac{y_-}{x_-} \oplus \frac{y_+}{x_+}$. Код Ротштейна m_1 для неё становится равным $L(x_- + x_+, y_- + y_+) = \sim L(x_+, y_+) \oplus \sim L(x_-, y_-) = m_2 \oplus \sim m_1$.
4. В конце работы алгоритма получатся лучшие приближения $\frac{y_-}{x_-}$ и $\frac{y_+}{x_+}$, которым соответствуют коды Ротштейна, равные m_1 и $\sim m_2$ соответственно. Значит, искомый код Ротштейна для исходного отрезка равен $\sim(\sim m_2) \oplus \sim m_1 = m_2 \oplus \sim m_1$.

Таким образом, алгоритм Кастла-Питвея полностью обоснован в случае $\text{НОД}(x_B, y_B) = 1$. Если координаты конца заданного отрезка имеют НОД, больший единицы, то это означает, что отрезок проходит через целочисленные пиксели $\left(\frac{x_B}{\text{НОД}(x_B, y_B)} i, \frac{y_B}{\text{НОД}(x_B, y_B)} i \right)$, $i = \overline{1, \text{НОД}(x_B, y_B)}$. Этими пикселями исходный отрезок делится на $\text{НОД}(x_B, y_B)$ маленьких. Для их растеризации достаточно растеризовать первый из них по алгоритму Кастла-Питвея, результат которого повторяется $\text{НОД}(x_B, y_B)$ раз, и получается растеризация для всего отрезка.