НЕКОТОРАЯ РЕАЛИЗАЦИЯ ТРАЕКТОРИИ ОБЪЕКТА UNITY

И. А. Смоляк, Е. В. Крот

Белорусский государственный университет; г. Минск;
smolakilona0@gmail.com;
Krotev120@gmail.com;
науч. рук. – О. М. Кветко, ст. преп.

В данной статье мы опишем как попасть в цель по заданной траектории, реализуя её в платформе Unity. Изначально расскажем, что такое Unity. Затем определим конкретное условие нашей задачи, рассмотрим подробно траекторию мяча и выведем формулу скорости для объекта с помощью математических формул. Опишем реализацию в Unity, в которой будем использовать ранее выведенную формулу, и покажем работу данной программы.

Ключевые слова: платформа Unity, заданная траектория, формула скорости.

При разработке видеоигр часто встречается задача вычисления определённого угла для попадания в цель. Сегодня это очень распространено. Мы хотим представить вам баллистическую траекторию объекта в Unity.

ПОДРОБНОЕ ОПИСАНИЕ ЗАДАЧИ И РЕАЛИЗАЦИЯ В UNITY

Что такое Unity?

Unity — это ведущая в отрасли платформа, предоставляющая инструменты для разработки и администрирования удивительных игр и других интерактивных продуктов реального времени, а также их публикации на широком спектре устройств.

Определение условия

Предположим, что в нашей игре есть два объекта, один из которых должен попасть мячом в другой по заданной траектории.

Траектория мяча

Создадим мяч в определённой начальной точке, из которой мы будем бросать его и зададим скорость. При этом рассчитаем скорость таким образом, чтобы она была направлена под таким углом и имела такую величину, чтобы попасть точно в наш объект, который мы задали. Мяч будет вести себя физично, то есть единственный раз мы зададим скорость и дальше он будет лететь по траектории корректно и как в реальной жизни.

Подробное описание задачи

Перейдём к описанию. У нас есть оси X, Y. Бросаем из центра координат и хотим попасть в какую-то случайную точку A, которую мы нарисуем произвольно, по такой траектории будет лететь мяч. Точка A и является нашей целью, куда мы хотим попасть. Следовательно, бросаем мяч

мы под каким-то углом α и скоростью v. Зависимость. Важно понимать, что траектория может быть разной. Если угол α меньше, то скорость v, следовательно, больше и наоборот. То есть это бесконечное количество разных траекторий, в зависимости от того, какой у нас угол.

Для начала примем, что угол α мы знаем, то есть он равен какой-то константе. Например, 45˚, потом в инспекторе мы можем изменить это значение.

Напишем уравнения, по которым мы рассчитаем нашу скорость.

Уравнения движения нашего объекта в проекциях на оси X и Y:

По X: положение объекта будет изменяться по закону:

$$x = v_x * t \qquad (1)$$

По этой оси мяч будет лететь прямолинейно равномерно с одинаковой скоростью, так как сопротивление воздуха мы учитывать не будем.

По Y: положение объекта будет изменяться по закону

$$y = v_y * t + (a * t^2) / 2 \qquad (2)$$

Так как действует сила гравитации, движение получается равноускоренное, следовательно, добавляем ускорение, умноженное на время в квадрате и пополам.

Получаем систему:

$$\begin{cases} x = v_x * t \\ y = v_y * t + \dfrac{a * t^2}{2} \end{cases} \qquad (3)$$

Где v_x – проекция скорости на ось X: $v_x = v * \cos(\alpha)$, v_y – проекция скорости на ось Y: $v_y = v * \sin(\alpha)$,

a – ускорение свободного падения: $a = g = 9{,}806$ м·с$^{-2}$

Подставим это в систему :

$$\begin{cases} x = v * \cos(\alpha) * t \\ y = v * \sin(\alpha) * t + \dfrac{g * t^2}{2} \end{cases} \qquad (4)$$

Нам известны x и y – координаты второго объекта(цели), угол α мы задаём, g константа.

Неизвестны скорость v и время t, величина, которая изменяется.

Чтобы решить эту систему уравнений и выразить из неё скорость v, мы можем из первого уравнения выразить t и подставить в систему (4) во второе уравнение:

$$t = \frac{x}{v * \cos(\alpha)} \qquad (5)$$

$$y = v * \sin(\alpha) * \frac{x}{v * \cos(\alpha)} + \frac{g * x^2}{2 * v^2 * (\cos(\alpha))^2} = x * \operatorname{tg}(\alpha) + \frac{g * x^2}{2 * v^2 * (\cos(\alpha))^2} \qquad (6)$$

Выражаем скорость из уравнения (6):

$$v = \pm \frac{\sqrt{g * x^2}}{\sqrt{2 * (y - x * tg(\alpha)) * \cos(\alpha)}} \quad (7)$$

и переносим её в Unity.

Реализация на Unity

Создадим объекты: пушку, бросающую мяч(shooter), цель(target) и сам мяч(ball).

Значит, у нас есть объект shooter, у него есть spawn, из которого мы будем бросать мяч. Она находится ровно в центре этого объекта и может вращаться по оси X, а ось Z направлена вперёд как у shooter, так и у spawn.

Создадим новый скрипт(ball), и он будет назначен на объект shooter.

Создадим пару полей: SpawnTransform – точка из которой мы бросаем, TargetTransform – это наша цель, AngleInDegrees – это угол, который мы будем в инспекторе задавать в градусах, bullet – мяч.

Начнём с того, что наш объект будем поворачиваться вслед за нашим игроком.

Vector3 fromTo= TargetTransform.position-transform.position это вектор, который направлен от объекта, который бросает, к его цели.

Нас интересует проекция этого вектора, чтобы наш объект поворачивался только по двум осям, поэтому создаем переменную Vector3 fromToXZ=new Vector3(fromTo.x,0f, fromTo.z), которая будет равна проекции этого вектора в плоскости XZ. То есть исключили ось Y и получили проекцию вектора на плоскость XZ.

Для того, чтобы наш объект поворачивался вслед за своей целью, воспользуемся функцией transform.rotation=Quaternion.LookRotation (fromToXZ, Vector3.up), указываем вектор, в направлении которого мы хотим смотреть fromToXZ и ось вверх Vector3.up. То есть наш объект, который бросает мяч, должен смотреть в этом направлении, но кроме этого он должен вращаться вокруг этой оси. Мы задаём определённое направление, вертикальная ось должна быть направлена вверх, тогда он будет ориентирован ровно как нам нужно.

Теперь реализуем поворот пушки в соответствии с тем углом, который мы зададим. Выбираем Spawn и будем из инспектора передавать значение угла поворота по x.SpawnTransform.LocalEulerAngles= new Vector3(-AngleInDegrees,0f,0f). Теперь этот угол управляет поворотом нашей пушки.

Перейдём к нашей формуле. Для начала создадим сам объект, мяч: public GameObject Bull. Переменные float x= fromToXZ.magnitude- это расстояние от объекта, бросающего мяч, до цели в горизонтальной плоскости или, одновременно, это длина вектора fromToXY, а float y=

fromTo.y. Теперь напишем формулу и будем использовать для удобства переменную v^2 и потом извлечем из неё корень. Задаём Physics.gravity.y = 9.8.

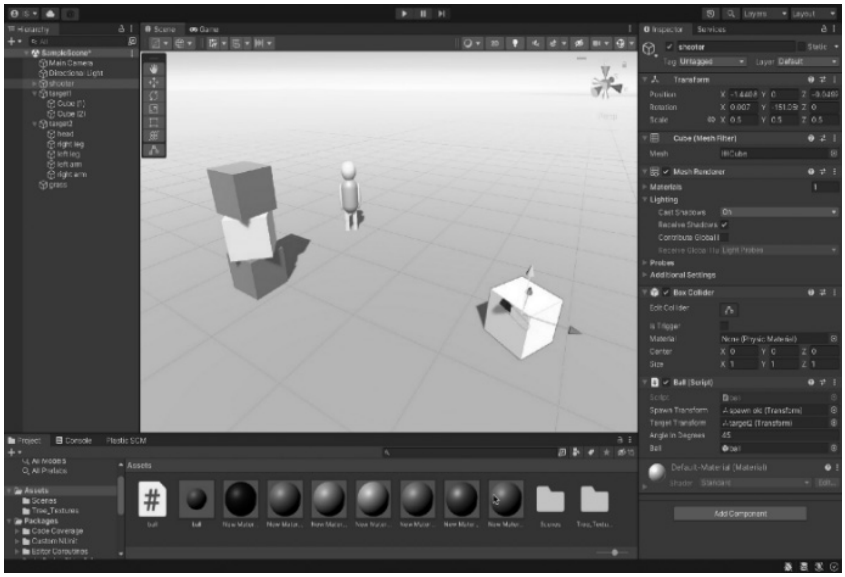
float v^2 = (g * x^2) / 2 * $\left(y - x * Mathf.tan(\text{AngleInRadians})\right) *$ Mathf.cos(AngleInRadians). Используется α в радианах, поэтому создадим ещё одну переменную float AngleInRadians = AngleInDegrees * $Mathf.PI$ / 180. И теперь, чтобы получить v, нам нужно извлечь корень

float v =Mathf.Sqrt(Mathf.Abs(v^2)), и возьмем модуль v^2, так как извлекать корень из отрицательного числа нельзя.

Таким образом, скорость мы рассчитали. Теперь создадим функцию public void Shot(), в которую поместим всё, кроме поворота пушки и добавим функцию GameObject = newBull Instantiate(Bull, SpawnTransform.position, Quaternion.identity). По умолчанию поворот равен 0, так как в данном случае нам это неважно. И затем добавляем к newBullt скорость: newBull = GetComponent<Rigidbody>().velocity= SpawnTransform.forward * v. Скорость это величина векторная, поэтому нужно понять в каком направлении она должна быть. Направление у нас будет направлено так же, как и Spawn. Так как в Updata() нам каждый раз не нужно считать эту скорость, а лишь только тогда, когда мы бросаем. И из Updata() будем вызывать, если например нажать кнопку мыши: if (Input.GetMouseButtonDown){Shot();}.

Исходный код выглядит следующим образом:

Прикрепляем скрипт к shooter, заполняем новые поля объектами spawn, target, ball и устанавливаем необходимый угол в градусах. Запускаем программу и проверяем её работу:



Библиографические ссылки

1. https://www.youtube.com/playlist?list=PL0lO_mIqDDFVNOKquWCHh4n-Ird5HRB_1
2. https://www.youtube.com/watch?v=Mofp_Z0kiBw