И. А. РАДКЕВИЧ, К. М. ШЕСТАКОВ

ПРОГРАММНЫЕ ТЕХНОЛОГИИ В ОБУЧЕНИИ ОПЕРАТОРОВ СЛОЖНОЙ ТЕХНИКИ

Компьютерная графика реального времени, ориентированная на визуализацию трехмерных сцен, достигла на сегодняшний день существенных успехов. Она находит широкое применение в сложных системах визуализации для тренажерных комплексов (авиационных, космических, морских, автомобильных, военных и т.п.).

Примерами таких комплексов могут служить тренажеры ТКНО–950 и ТКНТ–3Б, выпускаемые в Республике и предназначенные для обучения экипажей бронемашин БМП–2, танков Т–72. В ходе их создания решены задачи:

- создания виртуальных миров, в которых функционируют имитируемые объекты;
- моделирования физических процессов;
- моделирования реакции сложной техники на действия органов управления;
- моделирования отказов оборудования;
- контроля и учета действий обучаемых;
- автоматического выставления оценок;
- ведения базы данных по обучаемым;
- функционирования до 9 рабочих мест в сети без нарушения взаимодействия и эффекта реального времени.

Однако гибкое задание полигональной обстановки и объектов трехмерных сцен с последующим преобразованием и сканированием в плоскости изображений по прежнему актуально. Требуется хранить и отображать сцены, содержащие существенно большее число полигонов, чем реализовано в современных системах [1, 2, 3]. В данной работе представлены алгоритмы для визуализации больших объемов полигональных данных, с возможностью динамической подкачки ближнего окружения, и минимизации отклонений растеризации.

Первым шагом на пути уменьшения допустимых отклонений растеризации является введение динамической детализации с возможностью контроля точности.

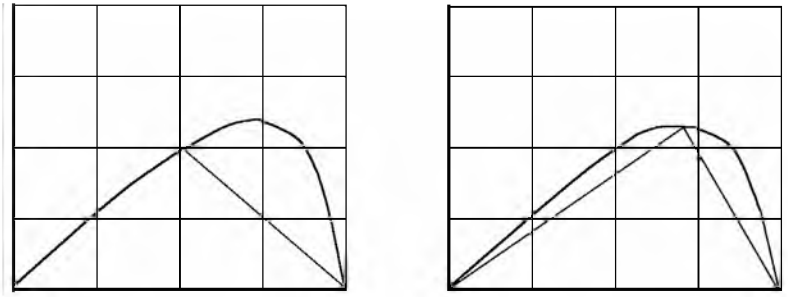
Наиболее удобным представлением для хранения и доступа к данным в данном случае является двоичное дерево. Это позволяет работать с большими объемами данных ввиду того, что скорость доступа пропорциональна $\log_2 N$, где N – полное число полигонов в кластере.

Вот так выглядит структура в C++, описывающая узел такого дерева:

```
struct mTriangleTreeNode
{
    char node_type; // NT_ROOT | NT_NODE | NT_LEAF
    unsigned int ancestor;
    unsigned int descender1, descender2;
    unsigned int compare, comp_claster;
    mPolyData  vis;
    float errorLevel;
    bool forceSplit;
    bool IsInTriangulation;
}
```

За основу при разработке алгоритма был взят алгоритм ROAM, разработанный совместно сотрудниками Лос-Алмосовской и Ливерморской лабораторий для задач авиасимуляторов. В ROAM строится дерево треугольников с ориентацией на равномерную разрядную сетку, то есть мы не могли полностью контролировать положение вершин в пространстве на этапе построения дерева, контролю поддавалась лишь высота. Разработанный алгоритм опирается на неравномерные разрядные сетки. Рассмотрим основные достоинства и недостатки такого подхода (рис. 1).

Первым очевидным достоинством является то, что при использовании неравномерных разрядных сеток мы можем достигать более высокой точности при более низком числе полигонов. В простейшем случае для низко-дисперсных сегментов ландшафта мы можем просто устанавливать вершину в максимуме функции (отметим, что для этого необходимо сначала перейти в пространство, являющееся касательным для предыдущего уровня приближения). Далее, очевидным плюсом является возмож-



а *б*

Рис. 1. Равномерная (*а*) и неравномерная (*б*) разрядные сетки

ность построения ландшафтов не задаваемых однозначными функциями высоты от координаты. То есть мы можем моделировать сегменты со сложной структурой: отвесные склоны, обрывы, навесы или даже пещеры. (единственным условием является односвязность поверхности). Однако при этом увеличивается объем памяти для хранения данных о поверхности (расчеты показали 8 % увеличение при построении горной местности), усложнилось построение триангуляционного дерева и расчет физических процессов.

Для того чтобы оценить качество динамической детализации необходим критерий, позволяющий определить достаточна или нет полигональная плотность в данном сегменте поверхности. Наиболее очевидным и наиболее удобным критерием в данном случае является полигональное отклонение - нормированное пиксельное отклонение ландшафта текущего уровня приближения от целевого ландшафта. Очевидно, что вычисление полигонального отклонения на каждом шаге построения триангуляции резко увеличит необходимые вычислительные объемы. Для того, чтобы избежать этого вводится статическая характеристика для каждого узла триангуляционного дерева – так называемый коэффициент ошибки. Эта величина рассчитывается на этапе построения триангуляционного дерева, соответственно уменьшая вычислительную нагрузку в процессе моделирования. Для вычисления нормированной ошибки для кадра надо умножить вектор отклонения на проекционную матрицу:

$$\text{Ошибка} = \|(0, \xi, z_c)\,\mu\| \qquad (1)$$

или в аппроксимации для случая перспективной проекции:

$$\text{Ошибка} = \frac{\square}{(z_c \cdot \square)}, \qquad (2)$$

где: z_c – расстояние до центра в пространстве камеры, μ – проекционная матрица, θ – коэффициент перспективной проекции

В процессе построения триангуляции для процесса деления мы берем коэффициент ошибки для текущего узла, получаем саму ошибку и сравнивая ее с максимально-допустимой принимаем решение о том, следует ли делить и дальше или можно остановится на достигнутой точности. Соответственно для процесса слияния мы берем коэффициент ошибки для предка текущего треугольника, получаем саму ошибку и принимаем решение, следует ли объединять текущий треугольник.

Рассмотрим, как будут выглядеть операции деления/слияния для случая неравномерной разрядной сетки (рис. 2).

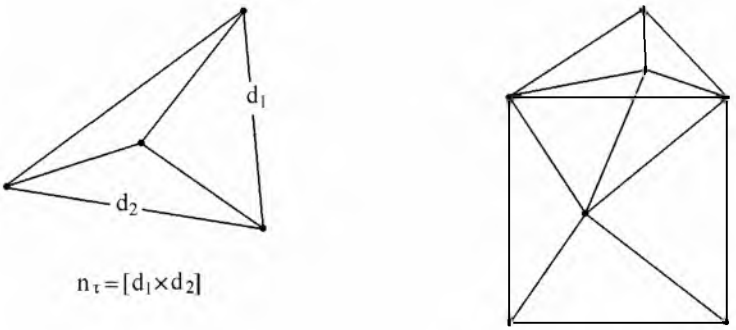


Рис. 2. Операции деления при неравномерной сетке

Во-первых, обратим внимание на то, что два парных треугольника вовсе не обязательно лежат в одной плоскости. Таким образом, нам надо ввести какое-то пространство касательное к данному. Будем называть касательным пространство с базисом состоящим из векторов $\mathbf{d}_1$, $\mathbf{n}_\tau$, $[\mathbf{d}_1 \cdot \mathbf{n}_\tau]$. Причем координата h, использовавшаяся в функции расчета коэффициента ошибки, будет в данном случае координатой по 2-му базисному вектору ($\mathbf{n}_\tau$). Таким образом, на этапе построения дерева мы берем 2 парных треугольника, получаем базис касательного пространства, определяем наиболее подходящий метод постановки новой вершины, рассчитываем ее координаты в касательном пространстве и проецируем назад в мировое пространство.

На сегодняшний день используется два метода постановки новой вершины: медианный (используется медиана функции высоты) и по максимуму (вершина ставится в локальном максимуме (моде) функции высоты, найденном методом координатного спуска).

Представленный алгоритм показал высокую эффективность при построении реальных полигонов.

ЛИТЕРАТУРА

1. *Gotsman C., Gumhold S., Kobbelt L.* Simplification and compression of 3d meshes // Tutorials on multiresolution in geometric modeling. Springer. 2002.
2. *Duchaineau M., Wolinsky M., Sigeti D. E.* ROAMing terrain: Real-time optimally adapting meshes // IEEE Visualization. 2001.
3. *Carl J.* Direct3D immediate mode: DirectX 7.0 programmer's reference/ Microsoft Corporation.1999. 640 p.