УДК 681.3.10

С. И. Вяткин[1], А. Н. Романюк[2], О. В. Романюк[2]
[1]Институт автоматики и электрометрии Сибирского отделения РАН
просп. Академика В.А. Коптюга, 1, 630090 Новосибирск, Россия
[2]Винницкий национальный технический университет, Украина
Хмельницкое шоссе, 95, 21000 Винница, Украина

Базовые операции для формирования рельефа и изменения формы сложных функциональных поверхностей в системах компьютерной визуализации

Рассмотрены вопросы использования функций возмущения для визуализации поверхностей трехмерных объектов в реальном времени. Предложены методы задания и визуализации трехмерных поверхностей, адаптированные для реализации на графических ускорителях. Предложена реализация методов преобразования описывающей функции для геометрических операций: офсеттинга, проекции, метаморфозиса (морфинга), кручения, заметания.

Ключевые слова: *скалярные функции возмущения, свободные формы, единичный трехмерный куб, морфинг, офсеттинг, кручение.*

Введение

В настоящее время численное решение задач трехмерной геометрии, требующих больших вычислительных ресурсов, находит все большее применение в реальных приложениях. Это моделирование динамики твердых и деформируемых тел, решение задач в области механики, вычислительной биологии, молекулярного моделирования и т.д. Компьютерное трехмерное геометрическое моделирование позволяет получить и исследовать геометрические параметры объекта, проанализировать механические свойства, динамику поведения и взаимодействия объектов. Многообразие задач геометрического моделирования требует разработки эффективных методов и алгоритмов формирования компьютерных трехмерных геометрических моделей, обеспечивающих высококачественную и информативную визуализацию в реальном времени, используя стандартные современные программно-аппаратные средства.

Анализ методов и постановка задачи

Наиболее распространенное задание трехмерных моделей объектов для их визуализации — полигональное задание. Наряду с преимуществами такая модель

имеет и свои недостатки. Моделируя реальные объекты, строится приближенная полигональная модель. Для повышения качества изображения чаще всего необходимо увеличивать количество полигонов, что влечет за собой увеличение времени визуализации и объема используемой памяти. При изменении масштаба объекта сложно быстро и эффективно изменить количество полигонов для модели объекта. Предлагаемая концепция функционального задания моделей объектов виртуальной среды позволит устранить недостатки, характерные для полигонального задания моделей.

Концепция моделирования виртуальной среды на базе функционально заданных объектов может быть описана как алгебраическая система

$$(M, \Phi, W), \tag{1}$$

где M — множество геометрических объектов; Φ — множество геометрических операций; W — множество отношений на множество объектов.

Известны следующие функциональные способы задания примитивов: поверхностями свертки [1] — это интегральное представление неявно заданных поверхностей, известных в компьютерной графике как капельные модели [2], метасферами и мягкими объектами [3].

Предлагается для описания геометрических объектов использовать функции возмущения от базовых поверхностей [4] (рис. 1). Свободная форма есть композиция базовой поверхности и возмущений

$$F'(x, y, z) = F(x, y, z) + \sum_{i=1}^{N} f_i * R_i(x, y, z), \tag{2}$$

где $R(x, y, z)$ — возмущение; f_i — форм-фактор;

$$R_i(x, y, z) = \begin{cases} Q_i^3(x, y, z), if\ Q_i(x, y, z) \geq 0, \\ 0, if\ Q_i(x, y, z) < 0, \end{cases} \tag{3}$$

где $Q(x, y, z)$ — возмущающая квадрика.

Степень возмущения можно выбирать произвольно. От этого будет зависеть гладкость переходов от базовой поверхности до области с возмущением, и, соответственно, размер самого возмущения. Для того, чтобы поверхность была гладкой, степень должна быть больше двух. Это условие гарантирует непрерывность функции и ее производной.

Задача конструирования объекта сводится не к аппроксимации его поверхности примитивами, а к деформации базовой поверхности нужным образом.

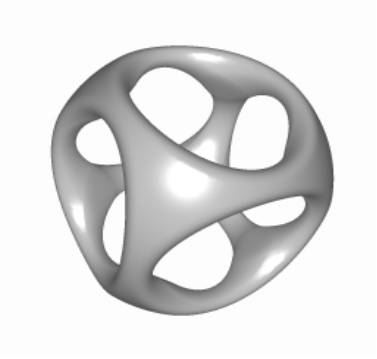

Рис. 1. Свободная форма на основе одной квадрики с пятью функциями возмущения

Описания объектов трехмерных сцен базовыми поверхностями и функциями возмущения имеет компактное описание, что позволяет уменьшить от 10 до 1000 раз объем передаваемых данных в зависимости от конкретных трехмерных сцен и моделей.

Цель работы — разработка базовых операций для формирования рельефа и изменения формы сложных функциональных поверхностей в системах компьютерной визуализации с использованием современных графических видеокарт.

Реализация геометрических операций и отношений над функционально заданными моделями объектов

Множество геометрических операций Φ математически выражается так:

$$\Phi_j:\ M^1 + M^2 + ... + M^n \rightarrow M, \tag{4}$$

Пусть объект G_1 определен как $f_1(X) \geq 0$. Унарная операция $(n = 1)$ объекта G_1 означает операцию $G_2 = \Phi_j(G_1)$ с определением

$$f_2 = \Psi(f_1(X)) \geq 0, \tag{5}$$

где Ψ — непрерывная вещественная функция одной переменной.

С помощью унарной операции офсеттинга можно создавать увеличенную или уменьшенную копию исходного объекта, то есть делать положительный или отрицательный офсеттинг соответственно (рис. 2).

Рис. 2. Положительный и отрицательный офсеттинг

Например, можно имитировать пульсацию тела. Пусть исходный объект задается функцией $f(X) > 0$, тогда при применении такой операции полученное тело будет описываться функцией $F = f(X) + C$, где $C < 0$ определяет отрицательный офсеттинг (сжатие), а $C < 0$ — положительный офсеттинг (расширение).

Следующая унарная операция — проекция (рис. 3). Пусть исходный объект $G_1 \subset E^n$ описывается функцией $f_1(x_1, x_2, ..., x_i, ..., x_n) \geq 0$, а его проекция $G_2 \subset E^{n-1}$ описывается функцией $f_2(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n) \geq 0$. Объект G_2 может быть определен как объединение сечений объекта G_1 гиперплоскостью $x_i = C_j$, где $C_{j+1} = C_j + \Delta x_i, j = 1, N,$ и $C_1 = x_{i\min}$. Пусть $f_{1j} = f_1(x_1, x_2, ..., x_{i-1}, C_j, x_{i+1}, ..., x_n)$ — функция для сечения. В итоге функция для проекции при $\Delta x_i \rightarrow 0$ есть объединение всех функций f_{1j}:

$$f_2 = f_{11} \vee f_{12} \vee \ldots \vee f_{1j} \vee \ldots \vee f_{1N}. \quad (6)$$

Рис. 3. Функционально заданный объект и его проекции

Пусть объекты G_1 и G_2 определены как $f_1(X) \geq 0$ и $f_2(X) \geq 0$. Бинарная операция $(n = 2)$ (4) объектов G_1 и G_2 означает операцию $G_3 = \Phi_j(G_1, G_2)$ с определением

$$f_3 = \Psi\,(f_1(X), f_2(X)) \geq 0, \quad (7)$$

где Ψ — непрерывная вещественная функция двух переменных.

Рассмотрим случай, когда необходимо из одного объекта плавно перейти к другому. Достичь такого эффекта можно, сконструировав новую функцию плотности из функций данных объектов.

Метаморфозис (морфинг) — это бинарная операция, которая преобразовывает первый заданый объект во второй с получением множества промежуточных форм. При метаморфозисе осуществляется плавный переход начального образа в конечный (рис. 4). Пусть мы имеем F_1, F_2 — значения функций возмущения первого и второго объектов, соответственно, тогда результирующая функция возмущения *F* вычисляется следующим образом:

$$F = \beta F_1 + (1 - \beta) F_2, \quad (8)$$

где β — положительная непрерывная функция.

Рис. 4. Морфинг негомеоморфных объектов

Морфинг для объектов, заданных функциями возмущения, значительно упрощается в сравнении с полигональными объектами. Морфинг для объектов на основе функций — это просто интерполяция этих функций. Заметим, что эти объекты могут быть различными, то есть иметь различные базовые квадрики, различные наборы функций возмущений. Изменяя функции возмущения, можно изме-

нить форму объекта. Интерполируя коэффициенты уравнения квадрики от исходного состояния до выбранного конечного, получим плавное перетекание объекта из одного состояния в другое. Такие же манипуляции можно проделывать и с базовой квадрикой. Так же можно интерполировать фактор возмущения. Интересных эффектов можно добиться изменением положения и масштаба квадрики возмущения относительно базовой.

Для многих объектов подобный эффект сложно реализовать при использовании полигональной модели задания. Но многие эффекты, связанные с преобразованием координат, которые реализуют с использованием полигональной модели, можно реализовать и на аналитически заданных объектах.

Кручение — это деформация тела, являющаяся частным случаем биективного отображения (рис. 5). Биективное отображение служит для определения деформаций исходных объектов. Для кручения исходного тела нужно найти его координаты x, y, z, и подвергнуть их преобразованию:

1) кручение вокруг оси Z:

$$t = (z - z_1)/(z_2 - z_1), \quad \theta = (1-t)\theta_1 + t\theta_2,$$
$$x_{new} = x\cos\theta + y\sin\theta, \quad y_{new} = -x\sin\theta + y\cos\theta,$$

где z_1, z_2 — координаты концов на z-интервале; θ_1, θ_2 — углы вращения в радианах для конечных точек;

2) кручение вокруг оси Y:

$$t = (y - y_1)/(y_2 - y_1), \quad \theta = (1-t)\theta_1 + t\theta_2,$$
$$z_{new} = z\cos\theta + x\sin\theta, \quad x_{new} = -z\sin\theta + x\cos\theta,$$

где y_1, y_2 — координаты концов на y-интервале; θ_1, θ_2 — углы вращения в радианах для конечных точек;

3) кручение вокруг оси X:

$$t = (x - x_1)/(x_2 - x_1), \quad \theta = (1-t)\theta_1 + t\theta_2,$$
$$y_{new} = y\cos\theta + z\sin\theta, \quad z_{new} = -y\sin\theta + z\cos\theta,$$

где x_1, x_2 — координаты концов на x-интервале; θ_1, θ_2 — углы вращения в радианах для конечных точек.

Рис. 5. Результаты реализации кручения

Заметание можно рассматривать как проекцию движущегося тела из 4D(x, y, z, t) в 3D(x, y, z) пространство. Далее отрисовывается тело каждый раз с новыми координатами, которые изменялись согласно определенному закону. При этом предыдущие полученные изображения сохраняются в памяти и используются для получения результата заметания. Получающаяся фигура — это объединение изображений заметающего тела при различных положениях (рис. 6).

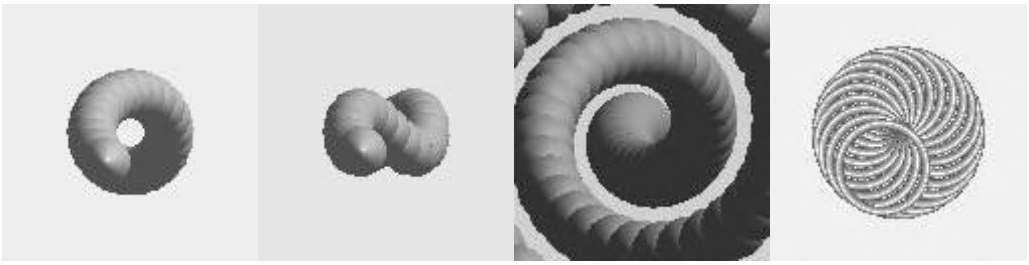

Рис. 6. Примеры заметания

Одним из примеров отношений может служить определение столкновений между объектами. Бинарное отношение есть множество множества $M^2 = M \times M$. Оно может быть определено как

$$S_j : M \times M \to I. \tag{9}$$

Пусть объекты G_1 и G_2 определены как $f_1(X) \geq 0$ и $f_2(X) \geq 0$. Бинарная операция пересечения объектов G_1 и G_2 определяется так:

$$S_c(G_1, G_2) = \begin{cases} 0, \text{ если } G_1 \cap G_2 = \varnothing, \\ 1, \text{ если } G_1 \cap G_2 \neq \varnothing. \end{cases} \tag{10}$$

Для вычисления S_c может быть использована функция $f_3(X) = f_1(X) \,\&\, f_2(X)$. Если $f_3(X) < 0$, то для любой точки пространства E^n $S_c = 0$.

С помощью особого теста на пересечение и бинарного поиска можно за постоянное число шагов (определяется заданной точностью) определить точку столкновения объектов, если оно имеет место.

Возможна реализация новых эффектов над геометрическими объектами с использованием предложенных операций. К недостаткам можно отнести сложность геометрической обработки и визуализации в реальном времени. Однако в настоящее время эту проблему можно решить применением высокопроизводительных параллельных многопроцессорных систем графических акселераторов.

Рендеринг геометрических объектов

Рассмотрим метод визуализации функционально заданных объектов, адаптированный для эффективной реализации графическими видеокартами [5]. Будем

считать, что сцена спроектирована в единичный трехмерный куб, а наблюдатель смотрит вдоль оси *Z* (рис. 7).

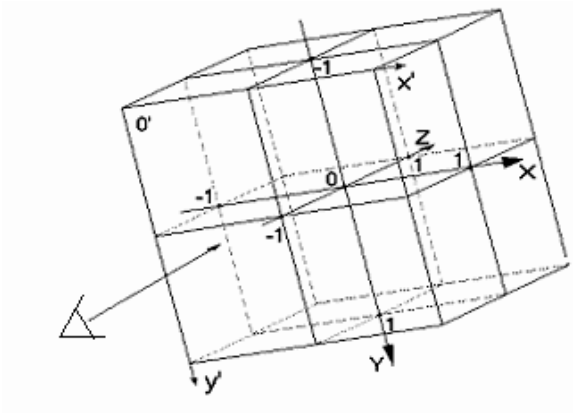

Рис. 7. Трехмерный единичный куб

Проекция сцены на плоскость *XY* будет представлять собой конечный набор значений. Весь куб можно разделить на «бруски» так, чтобы каждый брусок соответствовал пикселю на изображении. Каждый из брусков в дальнейшем делится вдоль оси *Z*, образуя набор вокселей. Так как размеры бруска в плоскости *XY* значительно меньше, чем в направлении оси *Z*, брусок можно рассматривать как луч. Таким образом, получим функцию плотности вдоль луча, которая зависит от одной переменной. Задача будет состоять в нахождении первой точки, в которой функция обращается в ноль. Найдя такую точку для каждого луча, будет известна глубина кадра. Далее в каждом пикселе можно вычислить нормаль. Имея данные о глубине и нормали в каждом пикселе, можно использовать модель локального освещения [6]. В итоге получится изображение гладкого объекта с учетом освещения. Таким образом, главной частью вычислений является эффективное нахождение первого пересечения луча с поверхностью.

В процессе вычисления точек поверхности объекта куб делится на меньшие части, для которых проверяется тест на пересечение с объектом. Процесс деления можно разбить на две стадии. В первую очередь куб делится на четыре части в плоскости *XY*. Далее рассматривается каждая часть отдельно. Если пересечение с заданным объектом не имеет место, то из дальнейшего рассмотрения эта часть исключается, и цвет всех пикселей, соответствующих этой части куба, принимает значение фона. С пересеченными объемами проводится аналогичная процедура деления. Процесс заканчивается, когда для рассматриваемой части соответствует только один пиксель. В дальнейшем каждую часть, с которой может быть пересечение, можно рассматривать как луч, направленный вдоль оси *Z*. Вторым шагом рассматриваются лучи, соответствующие полученным частям, и производится бинарный поиск для нахождения ближайшей точки пересечения с объемом.

Таким образом, деление куба ведется до тех пор, пока не достигается максимально установленный уровень рекурсии. Преимущество этого метода в том, что он позволяет на ранней стадии отбросить большие части пустого пространства. В процессе поиска вокселей, содержащих в себе участки поверхности объекта, фор-

мирующие изображение, осуществляется обход кубического пространства по четверичному дереву, листья которого являются корнями двоичных деревьев.

Основным отличием адаптированного метода является отсутствие четверичного деления. Поэтому куб делится на части в плоскости *XY* соответственно пикселям на изображении. Во второй части метод остается прежним. В этом случае уменьшение времени на визуализацию достигается за счет эффективного использования вычислительных ресурсов графического акселератора, что стало возможным благодаря использованию Compute Unified Device Architecture (CUDA) от компании NVIDIA. CUDA — это модель параллельного программирования с набором программных средств, которая позволяет реализовывать программы на языке C для исполнения на графическом акселераторе.

В адаптивном методе используется большое количество вычислительных процессоров, одновременно происходит проверка сразу нескольких лучей. В большей части графических видеокарт, поддерживающих CUDA, не менее ста двадцати восьми скалярных вычислительных ядер. Следовательно, будет отбрасываться достаточно большая часть куба. Стоит заметить, что в последовательном варианте (исходном методе), чтобы отбросить часть куба вблизи с объектом, необходимо поделить рассматриваемую область на более мелкие части. В параллельном варианте это не имеет значения, и большая часть будет сразу отброшена.

Во второй стадии метода (двоичное деление) принцип работы последовательного и параллельного вариантов практически не отличается. Отличие состоит только в способе задания объектов. В последовательном варианте объекты задавались в виде иерархической модели. В адаптивном методе возмущения задаются явно в виде квадрик. Это обусловлено тем, что при написании функций модели CUDA нельзя использовать рекурсию. Иерархическое представление является более естественным для такого метода задания объектов.

Заключение

Было реализовано приложение, которое визуализирует свободные формы, заданные квадриками с возмущениями на графическом акселераторе. В функции графической видеокарты входили расчет глубины кадра, нормалей и освещения. Центральный процессор использовался для геометрических преобразований. Для отображения изображения использовалась DirectX.

Проведено тестирование производительности предложенного варианта реализации. Тестирование производилось на компьютере с процессором Intel Core2 CPU E8400 3.0 GHz и графическим акселератором GeForce 8800 GTX. Проверялась производительность на четырнадцати различных сценах из свободных форм, заданных квадриками.

Время визуализации зависит от сложности объекта. Производительность сильно зависит от типа и скорости памяти. Наибольшей производительности можно достичь при использовании только регистров. Исследования показали, что 8192 регистров хватает для визуализации свободной формы с четырьмя возмущениями. Когда возмущений от пяти до десяти, то необходимо одновременно использовать память и регистры.

Тестирование показало, что при использовании четырех функций возмущения время визуализации с использованием видеокарты уменьшилось по сравнению с реализацией на центральном процессоре в семь раз.

С появлением новой унифицированной архитектуры GF100 от NVIDIA стала возможна анимация сложных функциональных поверхностей в реальном времени.

1. *Bloomenthal J.* Convolution Surfaces / J. Bloomenthal, K. Shoemake // SIGGRAPH'91, Computer Graphics. — 1991. — Vol. 25, N 4. — P. 251–256.

2. *Muraki S.* Volumetric Shape Description of Range Data Using «Blobby Model» / S. Muraki // Computer Graphics. — July, 1991. — 25(4). — P. 227–235.

3. *Wyvill G*. Data Structure for Soft Objects / G. Wyvill, C. McPheeters, B. Wyvill // The Visual Computer. — 1986. — 2(4). — P. 227–234.

4. *Вяткин С.И.* Моделирование сложных поверхностей с применением функций возмущения / С.И. Вяткин // Автометрия. — 2007. — Т. 43, № 3. — С. 40–47.

5. Геометрическое моделирование и визуализация функционально-заданных объектов / С.И. Вяткин, Б.С. Долговесов, А.В. Есин [и др.] // Автометрия. — 1999. — № 6. — С. 84.

6. *Романюк О.Н.* Високопродуктивні методи та засоби зафарбовування тривимірних графічних об'єктів. Монографія / О.Н. Романюк, А.В. Чорний. — Вінниця: УНІВЕСУМ-Вінниця, 2006. — 190 с.

Поступила в редакцию 15.09.2010