Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и Системы Управления»

Кафедра «Программное Обеспечение ЭВМ и Информационные Технологии»

Отчет по учебной практике

Студент Панчук Артем Игоревич

Группа ИУ7-42

Название (тема) практики «Программа редактирования трехмерных тел с использованием логических операций»

Научный руководитель
д-р т. н., профессор Майков К.А. _____
подпись

Руководитель практики
от МГТУ им. Н.Э. Баумана
к.т.н., доцент Куров А.В. _____
подпись

Содержание

Введение

Задачами дисциплины является формирование у обучающихся на основе ранее полученных теоретических знаний следующих умений, навыков и компетенций в области разработки программного обеспечения, включающей решение следующих задач:

- составление технического задания на разрабатываемое ПО;
- выбор, разработку или модификацию метода решаемой задачи;
- разработку алгоритмов решения задачи;
- выбор структур данных;
- выбор среды программирования;
- разработку форматов входных и выходных данных;
- разработку пользовательского интерфейса;
- разработку тестовых наборов данных;
- проведение тестирования отдельных модулей и комплексного тестирования;
- подготовку документации на разработанный программный продукт.

Целью практики является подготовка к профессиональной деятельности в ходе разработки законченного программного продукта в области компьютерной графики с подготовкой необходимой документации.

Основная часть

Перед нами стоит задача не только построения трехмерного изображения сцены с объектами и источниками света, но и также реализации возможности редактирования объектов с помощью логических операций. Поэтому отдельно рассмотрим алгоритмы, позволяющие:

- получить изображение трехмерных объектов (алгоритмы графики);
- осуществить редактирование трехмерных объектов (алгоритмы редактирования).

Алгоритмы графики

Рендеринг

Для рендеринга используем алгоритм «Distance-Aided Ray Marching» или «реймарчинг». В случае сложных поверхностей не нужно искать пересечение так, как это делают обычно – вычисляя геометрически пересечения прямой и поверхности явно. Вместо этого существует этот алгоритм, позволяющий сделать это итеративно, даже для тех поверхностей, для которых явная формула пересечения луча и поверхности вообще не существует (или ее очень сложно вывести).

Этот алгоритм схож с трассировкой лучей и бросанием лучей.

Достоинства:

- возможность рендеринга гладких объектов без аппроксимации их полигональными поверхностями (например, треугольниками); вычислительная сложность метода слабо зависит от сложности сцены;
- высокая алгоритмическая распараллеливаемость вычислений — можно параллельно и независимо направить два и более лучей, разделять участки (зоны экрана) на разных узлах кластера и т.д;
- отсечение невидимых поверхностей, перспектива и корректное изменения поля зрения являются логическим следствием алгоритма.

Недостатки:

- Серьёзным недостатком метода является производительность. Метод растеризации и сканирования строк использует когерентность данных,

чтобы распределить вычисления между пикселями. Впрочем, это влечет появление некоторых других преимуществ, таких как возможность направить больше лучей, чем предполагалось для устранения контурных неровностей в определённых местах модели. Также это регулирует отражение лучей и эффекты преломления, и в целом — степень фотореалистичности изображения.

Пусть существует функция f(x,y,z), задающая поверхность уравнением вида f(x,y,z) = 0. Для краткости можно будем писать f(p) = 0. Если в функцию f вместо p подставить, некоторую точку, не лежащую на поверхности, мы получим некое не равное нулю число. Нетрудно заметить, что это число будет представлять собой знаковое (хотя это может зависеть от формулы) расстояние до поверхности. Дальше мы «шагаем» по лучу на полученное расстояние и применяем эту процедуру еще раз, до тех пор, пока наше расстояние не станет меньше заданного порога. Следует отметить только, что размер шага было бы хорошо ограничивать снизу некоторой константой, для т.к. в противном случае трассировка будет очень медленной вблизи границ объектов.

Еще одна тонкость, для того чтобы не продолжать реймарчинг до бесконечности, необходимо в самом начале найти пересечение луча и ограничивающего всю сцену «bounding box-а». Как только выходим за эти границы (t > tmax) – нужно остановить реймарчинг. Либо можем просто ограничить максимальное число шагов по лучу какой-либо достаточно большой константой.

Нормаль есть не что иное, как градиент к поверхности. Вычисляем ее численно.

Освещение

Каждой точке в двухмерном пространстве экрана соответствует точка на некоторой поверхности (эту точку можно найти с помощью прослеживания пути луча, выпущенного из виртуального глаза через точку экрана в сцену; назовем ее точкой x). Освещенность точке поверхности x вычисляется при помощи интеграла следующего вида:

$$I\left(\phi_r, \theta_r\right) = \int_{\phi_i} \int_{\theta_i} L\left(\phi_i, \theta_i\right) R\left(\phi_i, \theta_i, \phi_r, \theta_r\right) d\phi_i d\theta_i$$

Это есть уравнение светопереноса. $L(\varphi i, \theta i)$ – это функция, описывающая общее освещение, падающее в точку x под всеми возможными углами в пределах полусферы. $R(\varphi i, \theta i, \varphi r, \theta r)$ – BRDF (bidirectional reflectance distribution function – двунаправленная функция распределения отражения или ДФО). Эта функция полностью описывает свойства взаимодействия конкретной поверхности со светом. Фактически, ДФО просто связывает по некоторому закону интенсивность и угол падающего света с интенсивностью и углом отраженного или пропущенного прозрачной поверхностью света.

Вычисляемая интегрированием интенсивность освещения в точке $I(\varphi r, \theta r)$ является не числом, а функцией. Другими словами, она дает значения интенсивности света, отражаемой поверхностью под разными углами. Таким образом, выполняя интегрирование по полусфере приходящего (падающего) в точку освещения L с учетом свойств поверхности R, мы получаем новую функцию распределения освещения в пространстве, обусловленную свойствами отражения поверхности (материала) в точке x. Хотя если мы рассматриваем луч, который трассировали через пиксел из виртуального глаза, то направление конечно одно и задано этим лучом.

Одна из наиболее часто используемых BRDF, представляющая полностью диффузную поверхность – BRDF Ламберта.

Считать интеграл освещенности точно – довольно дорого. Однако, существует ряд аппроксимаций, позволяющих примерно оценить его значение, не выполняя трассировку лучей по всем направлениям. Во-первых, следует разделить непрямое освещение на «резкоменяющееся» и «плавноменяющееся» – соответственно высокочастотную компоненту и низкочастотную компоненты. Высокочастотную составляющую считают, как и раньше, трассировкой лучей. Однако, для нее количество лучей обычно небольшое. Например, вам нужен всего 1 отраженный луч чтобы посчитать отражения на поверхности.

Для низкочастотной компоненты, обуславливающей диффузные переотражения света (которую дольше всего считать), можно применить аппроксимацию, позволяющую получить очень похожий на правду, но не совсем точный результат.

Идея «Ambient Occlusion» похожа на трассировку лучей по всем направлениям, но более простая в вычислительном плане. Мы предполагаем что вторичный, диффузно-переотраженный свет относительно равномерно распределен в пространстве. Такое освещение можно заменить неким

источником, так называемым «Environment Light». Он может быть задан панорамой, куб-мапом, или в простейшем случае просто – числом (3 числами – для красного, синего и зеленого). То есть мы говорим, пусть весь наш вторичный свет от относительно-далеко стоящих поверхностей задан некой константой.

Тогда, в той точке, где мы хотим посчитать освещение, мы испускаем лучи по всем направлениям, но не трассируем их в сцену, а лишь пытаемся вычислить, какой процент света закрыт близлежащими поверхностями.

То есть, фактически мы просто ограничили длину трассировки луча. Однако, во многих случаях «Ambient Occlusion» можно вычислить, не прибегая к трассировке лучей, заменив ее более простой апроксимацией, ведь нам не нужно в действительности находить пересечения, необходимо лишь знать — какой процент (примерно) нашего воображаемого источника закрыт.

К сожалению, оказывается, что даже «Ambient Occlusion» вычислить довольно трудоемко, особенно если речь идет об интерактивной визуализации. Однако, в случае неявно задаваемых поверхностей, существует упрощение, довольно точно приближающее «Ambient Occlusion». Идея состоит в том, чтобы взять несколько точек в направлении нормали к поверхности. И посмотрев в них расстояние до поверхности, оценить, насколько эта точка закрыта близлежащей геометрией. Это как раз наш случай, потому что у нас есть такая функция – это просто левая часть уравнения поверхности f(x,y,z) = 0. То есть это знакомая нам функция f(p). «Ambient Occlusion» рассчитанный таким способом, оказывается выгоднее чем трассировка первичного луча.

Алгоритмы редактирования

Все примитивы мы можем задать функциями расстояний. Тогда и булевы операции над ними мы так же зададим через функции расстояний. К примеру, псевдокодом сферу и цилиндр можем задать так:

```
float sdSphere( vec3 p, float s ) { return length(p)-s; }

float sdCylinder( vec3 p, vec3 c ) { return length(p.xz-c.xy)-c.z; }
```

Так же выразим операции объединение, пересечение, разность:

```
float opU( float d1, float d2 ) { return min(d1,d2); }

float opS( float d1, float d2 ) { return max(-d1,d2); }

float opI( float d1, float d2 ) { return max(d1,d2); }
```

Такое представление крайне удобно использовать при выбранной модели рендеринга. Результат выполнения операции так же представляется в виде функции. Дальнейшее его использование в других операциях представляет собой просто композицию функций.

Эти функции намного проще хранить, чем, к примеру, полигоны объекта. Единственным недостатком может проявиться проблема с производительностью, но при правильном распределении вычислений между ЦПУ и ГПУ можно добиться отзывчивого редактирования с рендерингом в режиме реального времени.

Заключение

В ходе практики мы подготовились к профессиональной деятельности, занимаясь разработкой законченного программного продукта в области компьютерной графики с подготовкой необходимой документации. Мы изучили поставленные задачи, нашли необходимые алгоритмы решения этих задач, проанализировали их, оптимизировали для применения к нашей ситуации.

Литература

- Функции расстояний (http://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm);
- Алгоритм Distance-Aided Ray Marching (http://ray-tracing.ru/articles231.html);
- Трассировка лучей (http://www.iquilezles.org/www/material/nvscene2008/rwwtt.pdf);
- Трассировка лучей (https://en.wikipedia.org/wiki/Ray_tracing_(graphics));
- Бросание лучей (https://en.wikipedia.org/wiki/Ray_casting);
- Модель Ламберта (https://en.wikipedia.org/wiki/Lambertian_reflectance);

Приложения

- Программа с примером рендеринга алгоритмом реймарчинга https://github.com/nopjia/WebGL-RayMarch;