МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

VI Международной молодежной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 24–26 мая 2018 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательский Дом Томского государственного университета
2018

4. *Ежов А.А, Шумский С.А.* Нейрокомпьютинг и его применение в экономике и бизнесе. – М.: МИФИ, 1998. – 224 с.
5. Geektimes Прогнозирование фондового рынка с помощью нейронных сетей [Электронный ресурс] // URL: https://geektimes.com/post/279170/ (дата обращения: 07.04.2018).
6. *Лесик И.А.* Решение задачи прогнозирования с использованием нейронных сетей прямого распространения на примере построения прогноза роста курса акций // Программные продукты и системы. – 2015. – Вып. 2. – С. 70–74.
7. MATLAB.Exponenta MATLAB&Toolboxes [Электронный ресурс] // URL: http://matlab.exponenta.ru/neuralnetwork/book2/11/newff.php (дата обращения: 15.02.2018).
8. Artificial neural network [Электронный ресурс] // URL: https://en.wikipedia.org/wiki/Artificial_neural_network (дата обращения: 10.04.2018).
9. NeuroPro Области применения искусственных нейронных сетей [Электронный ресурс] // URL: http://neuropro.ru/neu7.shtml (дата обращения: 15.02.2018).

ПОСТРОЕНИЕ РЕАЛИСТИЧНЫХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ GLSL-ШЕЙДЕРОВ OPENGL

Л.В. Чупрасова
Томский государственный университет
l.chuprasova@gmail.com

Введение

Компьютерная графика – это достаточно сложная, основательно проработанная дисциплина, предметом изучения которой является создание, хранение и обработка моделей и их изображений с помощью ЭВМ.

3D-графика применяется во множестве различных областей: в науке, для наглядного изображения каких-то не до конца изученных процессов, в видеоиграх, которые давно не ограничиваются компьютерной платформой, в кинематографе для создания сложнейших спецэффектов, в медицине, строительстве, рекламе, и т.д. Имея качественную модель, можно увидеть объект, который не существует, посмотреть его со всех сторон, окунуться в виртуальный мир компьютерных игр, сидя дома перед экраном компьютера.

Следуя правилам создания качественной 3D-модели, можно добиться эффекта присутствия, эффекта реальности, т.е. такие объекты отличаются от других правдоподобностью. В наше время реалистичные 3D-изображения являются пиком совершенства в игровой, рекламной, строительной индустрии.

Моделирование 3D-изображений – это достаточно ресурсоемкая задача, ставящаяся перед современными вычислительными системами, и, несмотря на постоянное совершенствование техники, стремительное развитие технологий, требования к задачам остаются неизменно высокими, т.к. с ростом возможностей растут и требования, т.е. помимо оптимизации технической базы требуется оптимизация на уровне прикладного программирования.

Одним из таких методов оптимизации является грамотное использование графической библиотеки OpenGL, а именно – программирование на шейдерах. Оно дает возможность переложить часть сложных вычислений, связанных с отрисовкой сцены, с центрального процессора на специально предназначенный для этого графический процессор, являющийся аппаратной частью видеокарты.

1. Библиотека OpenGL и шейдеры

OpenGL является одним из самых распространенных прикладных программных интерфейсов (API – Application Programming Interface) к графической аппаратуре для разработки приложений в области двумерной и трёхмерной графики.

Библиотека содержит в себе около 120 команд, которые может использовать программист для задания объектов и операций, необходимых для написания интерактивного графического приложения.

Основными особенностями этой графической библиотеки являются:

- Стабильность – все нововведения создаются таким образом, чтобы сохранялась совместимость с предыдущим обеспечением.
- Кроссплатформенность – независимо от используемой операционной системы OpenGL гарантирует одинаковый визуальный результат. Кроме того, эти приложения работают и на рабочих станциях, и на суперкомпьютерах.
- Интуитивно понятный интерфейс и продуманная структура, что обеспечивает легкость применения [1].

Вместе с выходом версии OpenGL 3.0 стала доступна шейдерная программа, именно она лежит в основе решения задач реализации реалистичного 3D-изображения.

Шейдерная программа – это небольшая программа, состоящая из программных модулей – шейдеров (вершинного и фрагментного, возможны и другие). Отличительной особенностью шейдеров является то, что они выполняются на аппаратном уровне оборудования, используя видеопамять и графический процессор (GPU) [2]. Программа заменяет часть графического конвейера видеокарты, без её использования мы не получим даже самую простую картинку на экране.

Программы, работающие с трёхмерной графикой, используют шейдеры для определения параметров геометрических объектов или изображения, для изменения изображения (для создания эффектов сдвига, отражения, преломления, затемнения с учётом заданных параметров поглощения и рассеяния света, для наложения текстур на геометрические объекты и др.)

Шейдеры делят на три типа, в зависимости от того, какой процессор будет их исполнять:

- Вершинный шейдер (vertex shader). Заменяет часть графического конвейера, выполняющего преобразования, связанные с данными вершин геометрических примитивов, такие, как умножение вершин и нормалей на матрицу проекции и моделирования, установка цветов вершин, установка материалов освещения. Он работает для каждой отрисованной вершины.
- Фрагментный или пиксельный шейдер (pixel shader). Заменяет часть графического конвейера, обрабатывает каждый полученный на предыдущих стадиях фрагмент, а именно – данные, связанные с пикселями. Например, получение данных из текстуры, просчёт освещения, просчёт смешивания. Пиксельный шейдер используется на последней стадии графического конвейера для формирования фрагмента изображения.
- Геометрический шейдер (geometry shader). В отличие от вершинного, геометрический шейдер способен обработать не только одну вершину, но и целый примитив. Такой тип шейдера используется редко [2].

Шейдеры для OpenGL пишутся на специальном языке GLSL. Шейдером называется независимо компилируемая единица, написанная на этом языке. GLSL-шейдеры принято хранить в виде исходных кодов. Исходные коды компилируются драйвером лишь после создания действующего контекста OpenGL. Шейдерная программа – это набор скомпилированных шейдеров, связанных вместе.

GLSL (OpenGL Shading Language) – язык высокого уровня для программирования шейдеров. Синтаксис языка базируется на языке программирования ANSI C, однако, из-за его специфической направленности для упрощения и повышения производительности из него были исключены многие возможности. В язык включены дополнительные функции и типы данных, например, для работы с векторами и матрицами [3].

2. Реалистичное изображение

Для создания реалистичных изображений необходимо определить как свойства самого объекта, так и свойства внешней среды.

Первая группа включает в себя параметры материала, из которого сделан объект, степень его прозрачности, а также текстура. Ко второй группе относятся количество и свойства источников света, модель освещения и отрисовка теней.

Каждое свойство задаётся последовательно в исходном коде программы, с помощью вызовов специальных команд OpenGL.

Для снятия нагрузки с центрального процессора была написана шейдерная программа, состоящая из двух вершинных и двух фрагментных шейдеров, в которой обрабатываются все данные, необходимые при построении качественных 3D-моделей.

Чтобы нарисовать какие-либо модели, создан класс Mesh.

2.1. Источники света

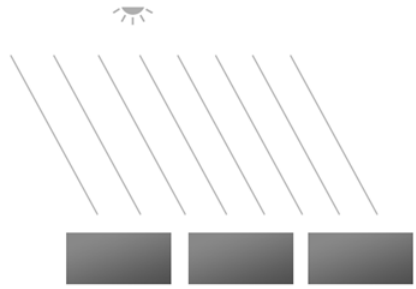
В компьютерной графике чаще всего работают с тремя типами источников света [4]:

- Точечный источник света (point light).
- Направленный источник света (directional light).
- Прожектор (spot light).

В работе настроен и реализован направленный источник света.

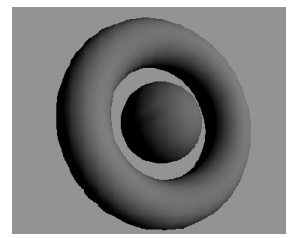
Направленный источник света находится в бесконечности, и свет распространяет в заданном направлении. Если источник света достаточно удалён от объекта, то все его приходящие лучи параллельны друг другу, поэтому создается впечатление, что свет направлен одинаково и не зависит ни от расположения источника света, ни от положения наблюдателя. Поэтому и расчёт освещения для всех объектов будет одинаковым. Такой свет идеально подходит для равномерного освещения.

Хорошей аналогией направленного источника света является солнце. Хоть оно и не бесконечно удалено от нас, этого расстояния достаточно, чтобы считать его таковым при расчётах [5].



Свет настраивается в исходном коде программы, но его свойства требуются при расчётах в построении модели освещения, которые производятся в шейдерной программе. В OpenGL нет встроенных функций передачи конкретных данных в шейдер, поэтому для работы с источником света создан класс Light.

Когда мы настраиваем все компоненты источника света, становится видно сцену с объектами:



2.2. Модель освещения

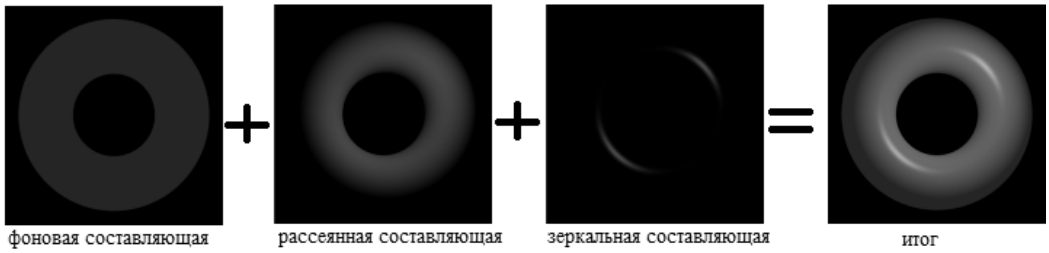
В любом трехмерном изображении использование какой-либо модели освещения придает реалистичность. Любая модель освещения включает в себя закон, по которому

рассчитывается освещённость точки в пространстве и метод закраски многоугольника. В OpenGL нет готовой реализации модели освещения.

В работе использована закраска методом Фонга, потому что она не требует больших вычислительных затрат, однако позволяет решить многие проблемы других методов.

Суть метода Фонга заключается в определении нормали к поверхности в каждой вершине полигона и дальнейшей интерполяции вектора по всему полигону. Далее вычисляется значение яркости для каждого пикселя, основываясь на значениях вектора нормали. При этом достигается лучшая аппроксимация кривизны поверхности и, следовательно, получается более реалистичное изображение.

Освещённость каждой точки складывается из трёх составляющих: фоновое освещение, рассеянный свет и бликовая составляющая. Свойства источника света определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения.



1. Фоновое освещение (ambient) – это постоянная в каждой точке величина надбавки к освещению. Теоретически, в сцене оно имитирует рассеянный свет, не имеющий конкретного источника. Поскольку центра и направления у него нет, оно совершенно одинаково для всех точек поверхности.

2. Рассеянный свет или диффузное отражение (diffuse) соответствует отражению света от неровной поверхности. При попадании на подобную поверхность свет рассеивается практически равномерно по всем направлениям

3. Зеркальный свет (specular) учитывает не только направление на источник света, но и местоположение наблюдателя. При попадании на поверхность подчиняется следующему закону: «падающий и отражённый лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части». Таким образом, отражённая составляющая освещённости в точке зависит от того, насколько близки направления на наблюдателя и отражённого луча [6].

2.3. Материал объекта

Также нужно определить материал объекта для того, чтобы знать, как этот объект будет реагировать на освещение. Свойства материала определяют следующими параметрами:

1. Восприятие фонового освещения (ambient). Фоновое освещение на практике используется, чтобы подсветить излишне тёмные участки.

2. Восприятие рассеянного, диффузного освещения (diffuse). Диффузное отражение придает объекту его естественный цвет.

3. Восприятие отражённого освещения (specular). Зеркальная компонента определяет отражающие свойства поверхности.

4. Самостоятельное свечение (emission). Самостоятельное свечение материала определяет интенсивность излучаемого света материалом.

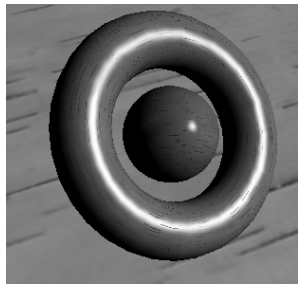
5. Коэффициент блеска (shiniess). Коэффициент блеска определяет степень зеркального отражения материала. Значение задается от 0 до 128 [7].

6. Текстурирование. Под текстурой понимается некоторое изображение, которое надо определённым образом нанести на объект, например, для придания иллюзии рель-

ефности поверхности. Изображение может быть любым, но чаще всего это шаблоны кирпича, дерева, листьев и другие, с помощью которых сцене добавляется дополнительный реализм.

Все свойства материалов также определены в исходном коде программы. При рендере сцены данные передаются в шейдер, в котором выполняется расчёт цвета для каждого пикселя с учётом всех обозначенных свойств. Для работы с материалом создан класс Material.

После добавления модели освещения и текстурирования получено следующее изображение:



2.4. Тени

Одной из главных задач при синтезе реалистичных изображений является построение теней. С помощью теней реализм виртуальной сцены значительно увеличивается, а также они помогают лучше понять взаимное расположение объектов и источника света.

К сожалению, в OpenGL нет функции построения теней, но мы можем реализовать сами один из нескольких известных алгоритмов с помощью этой библиотеки.

С физической точки зрения, тени можно разделить на две составляющие: полная (абсолютная) тень и полутень. Полная тень – это та часть затеняемого объекта, которая должна быть невидима совсем – тёмная, резко очерченная часть. При реализации алгоритма построения теней используется только полная тень.

Существует несколько алгоритмов при визуализации сцены с тенями. В настоящее время наиболее распространены три:

1. Проективные тени.
2. Метод теневых объёмов.
3. Карта теней.

Был использован алгоритм карты теней (Shadow maps). Это наиболее распространенный метод. Пожалуй, его суть является самой простой из всех. Метод базируется на том, что точки, находящиеся в тени – это те точки, которые “спрятаны” от источника света. Метод использует буфер глубины (Z-буфер).

Для того, чтобы определить видимые и невидимые точки, выполняем следующие действия:

1. Сцену выводим в Z-буфер из положения источника света, после чего создаем пустую текстуру для хранения буфера глубины (CreateDepthTexture). Содержимое буфера копируем в эту текстуру, которая и называется картой тени. Она хранит значения глубины ближайших к источнику света элементов сцены.

Так как при заполнении буфера глубины нет необходимости обрабатывать цвет пикселя, а нужно знать только его глубину, написан отдельный вершинный шейдер, в котором отключены все лишние расчёты, чтобы как можно больше ускорить работу программы, а т.к. цвет пикселя нас не интересует, фрагментный шейдер может быть пустым.

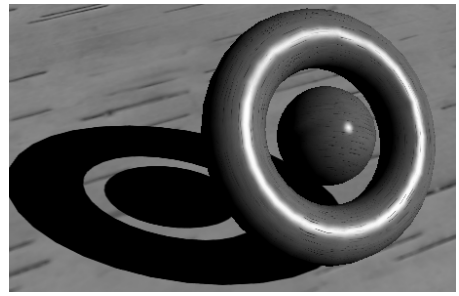
Также здесь важно отметить, что для разных типов источников света все происходит по-разному. Направленные источники света не имеют позиции в пространстве, к ним можно при определённом приближении отнести солнечный свет. Но для формиро-

вания карты теней эту позицию приходится выбирать, обычно её привязывают к положению наблюдателя, чтобы в карту попадали объекты, находящиеся в поле зрения наблюдателя. При рендеринге используют ортографическую проекцию.

2. Рисуем сцену из основной камеры, с точки зрения наблюдателя. Для того, чтобы понять, в тени точка или освещена, достаточно перевести координаты этой точки в пространство карты теней и сравнить. Переводим координаты точки из мировой системы координат в текстурные координаты, которые лежат в диапазоне [0;1]. Если полученные координаты не попали в диапазон [0;1], то точка не находится в карте теней, значит, она освещена. Сделав эту выборку, сравниваем глубину точки, сохранённой в карту теней и глубину точки, переведенной в систему координат источника. Точка оказывается в тени, если значение в карте теней меньше, т.е., если значение из карты теней меньше, то в этой точке есть какой-то объект, который находится ближе к источнику света, и мы находимся в его тени.

Скорость выполнения данной реализации метода не зависит ни от сложности затеняющих объектов, ни от сложности сцены, что обуславливает высокую эффективность [8].

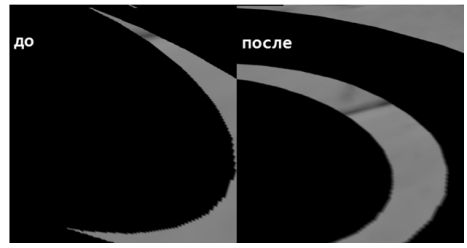
Реализация тени выглядит таким образом:



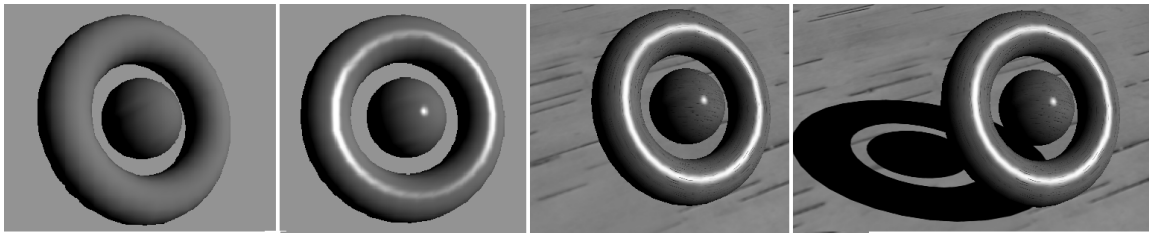
2.5. Сглаживание краев тени

Для сглаживания краёв тени использована техника Percentage Closer Filtering (PCF). Данная техника заключается в том, что производится несколько тестов на затенение в окрестностях искомого фрагмента [9], т.е. для каждого пикселя берётся среднее значение его соседних пикселей в карте теней. Таким образом, получается плавный переход из тёмной области в светлую область, граница теней будет более «мягкой».

Расчёт средних значений произведен также в основном шейдере.



Демонстрация:



Плоская закраска + свет + модель освещения Фонга + текстурирование + тень

Были проведены эксперименты, связывающие сложность сцены и время выполнения программы (ЦП: AMD E-350 Proceccor 1.60 GHz, GPU: Radeon HD 6330M, эффективная тактовая частота памяти 1800 МГц, память 1024 МБ).

Количество объектов сцены	Время в секундах
1	0.00145484
3	0.00274012
6	0.00380818
12	0.00420133
100	0.00551013

Так как все расчеты и прорисовка выполняются на графическом процессоре, изменение сложности сцены несущественно влияет на время выполнения программы.

Заключение

Перед нами была поставлена задача спроектировать реалистичную сцену, освещаемую методом Фонга и затенённую по алгоритму карты теней. Для этого были изучены некоторые методы стандарта OpenGL. Благодаря OpenGL и его шейдерной программе процесс построения этой модели был сведён к прикладному математическому решению и использованию небольшого количества ресурсов, что сводит математические расчёты к минимуму.

Задача была выполнена, разработана и реализована архитектура классов, способных отрисовывать трехмерные сцены с использованием основных методов OpenGL.

Создано демонстрационное приложение с использованием этой библиотеки. Реализованы алгоритмы 3D-графики: освещение методом Фонга с текстурированием объекта и отрисовка теней алгоритмом карты теней с использованием техники сглаживания Percentage Closer Filtering (PCF).

ЛИТЕРАТУРА

1. Графическая библиотека OpenGL. Учебно-методическое пособие [Электронный ресурс]. / Ю.М. Баяковский, А.В. Игнатенко, А.И. Фролов – Факультет вычислительной математики и кибернетики МГУ им. Ломоносова, 2003. – URL: https://rsdn.org/article/opengl/ogltut2.xml (дата обращения: 20.05.2018).
2. Шейдер [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki?curid=107896 (дата обращения: 20.05.2018).
3. OpenGL Shading Language [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/OpenGL_Shading_Language (дата обращения: 20.05.2018).
4. Компьютерная графика / теория, алгоритмы, примеры на C++ и OpenGL / Источники света [Электронный ресурс]. – URL: http://compgraphics.info/OpenGL/lighting/light_sources.php (дата обращения: 20.05.2018).
5. Learn OpenGL. Урок 2.5 – Источники света [Электронный ресурс]. – URL: https://habrahabr.ru/post/337642/ (дата обращения: 20.05.2018).
6. Компьютерная графика / теория, алгоритмы, примеры на C++ и OpenGL / Закон Ламберта. Модель отражения Фонга. Модель отражения Блинна – Фонга [Электронный ресурс]. – URL: http://compgraphics.info/3D/lighting/phong_reflection_model.php (дата обращения: 20.05.2018).
7. Свойства материала [Электронный ресурс]. – URL: http://metodpro.ru/index.php?type_page&katalog&id=999&met6 (дата обращения: 20.05.2018).
8. Реализация теней с помощью библиотеки OpenGL / Журавлев С.В., Михайлюк М.В., Торгашев М.А. [Электронный ресурс]. – URL: http://swsys.ru/print/article_print.php?id=863 (дата обращения: 20.05.2018).
9. Компьютерная графика лекция GLSL-тени от объектов [Электронный ресурс]. – URL: http://docplayer.ru/52035159-Kompyuternaya-grafika-lekciya-glsl-teni-ot-obektov.html (дата обращения: 20.05.2018).