

$$\text{СКО} = \left(\frac{1}{N} \sum_{i=1}^N (\hat{S}_i - \text{Suct}_i)^2 \right)^{\frac{1}{2}},$$

где $\hat{S}_i$ – оцененный отсчет ПФ; Suct_i – отсчет истинной ПФ; N – число отсчетов ПФ.

Результаты моделирования приведены на рис. 5.

Как следует из рис. 5, использование априорных сведений о количестве лучей в канале позволяет повысить точность оценки передаточной функции канала распространения радиоволн. Модификация методов может быть применена для решения задачи эквалайзирования в системах беспроводного широкополосного доступа, вычислительные зат-

раты на ее реализацию минимальны, а выигрыш от применения существенный.

Выводы

Показано, что модификация методов оценки импульсной характеристики канала распространения радиоволн ее прореживанием от шумовых отсчетов (оставляют 6–20 наиболее вероятных отсчетов максимальной амплитуды) приводит к уменьшению ошибки оценки передаточной функции до 3 раз при отношении сигнал/шум 3 дБ.

Работа выполнена при поддержке Министерства образования и науки РФ в соответствии с договором 13.G25.31.0011 от 07.09.2010 в порядке реализации Постановления № 218 Правительства РФ.

СПИСОК ЛИТЕРАТУРЫ

1. Айфичер Э., Джервис Б. Цифровая обработка сигналов: практический подход. 2-е изд. Пер с англ. – М.: Издат. дом «Вильямс», 2004. – 992 с.
2. Summary of delay profiles for MBWA. 2011. URL: <http://www.ieee802.org/20/Contribs/C802.20-03-77.pdf> (дата обращения: 08.05.2011).
3. Channel Models: A Tutorial. 2011. URL: http://www.cse.wustl.edu/~jain/cse574-08/ftp/channel_model_tutorial.pdf (дата обращения: 08.05.2011).
4. Тисленко В.И. Статистическая теория радиотехнических систем. – Томск: Том. гос. ун-т систем управления и радиоэлектроники, 2003. – 153 с.
5. Сергиенко А.Б. Цифровая обработка сигналов. – СПб.: Питер, 2005. – 604 с.
6. Уидроу Б., Стирнз С. Адаптивная обработка сигналов. – М.: Радио и связь, 1989. – 439 с.
7. Оппенгейм Э. Применение цифровой обработки сигналов. – М.: Мир, 1980. – 550 с.
8. Тихонов В.И. Оптимальный прием сигналов. – М.: Радио и связь, 1983. – 320 с.
9. Тисленко В.И., Савин А.А. Синтез квазиоптимального фильтра для оценки временного положения импульсного сигнала известной формы при многолучевом распространении радиоволн // Известия вузов. Радиоэлектроника. – 2006. – Т. 1. – № 6. – С. 56–62.

Поступила 10.05.2011 г.

УДК 519.688;004.92

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ОПТИМИЗАЦИИ МОДЕЛИ ГЛОБАЛЬНОГО ОСВЕЩЕНИЯ

В.В. Парубец, О.Г. Берестнева

Томский политехнический университет
E-mail: VParubets.mail@gmail.com

Рассматривается математическая модель построения освещения в базе сферических функций для трёхмерной сцены. Предлагается вариант распараллеленного предварительного расчёта коэффициентов разложения точек сцены по базису с применением технологии nVidia CUDA. Приводится сравнение производительности при счёте на центральных и графических процессорах.

Ключевые слова:

Обобщённая двулучевая функция распределения света, nVidia CUDA.

Key words:

Bidirectional reflectance distribution function, nVidia CUDA.

Введение

Расчёт освещения в базе сферических функций является одним из хорошо зарекомендовавших себя алгоритмов, применяющихся во множестве приложений компьютерной графики. Основной проблемой данного метода является длительный и ресурсоёмкий предварительный расчёт сфе-

рических гармоник для каждой точки трёхмерной сцены. Такая задача может быть распараллелена по данным. В описанном случае для наибольшей производительности решено применить технологию, позволяющую получить доступ к вычислительным ресурсам графических процессоров – CUDA от компании nVidia.

Данная работа даёт возможность сравнить производительность центральных и графических процессоров, а также увидеть пользу от оптимизации модели под параллельные алгоритмы.

Математическая модель

Данная модель представляет собой реализацию обобщённой двулучевой функции распределения света (ОДФРС) [1–3]. Для каждого класса материалов, присутствующих в сцене рассчитываются две функции: двулучевая функция отражённого света (ДФОС) и двулучевая функция рассеянного света (ДФРС), а материал в свою очередь представляется в виде комбинации идеального диффузного и идеального зеркального материалов. ОДФРС так же представляется в виде комбинации ДФОС и ДФРС.

Двулучевая функция отражённого света

ДФОС определяется как отношение количества энергии (света), отраженной в направлении ω_0 , к количеству энергии, попавшей на поверхность с направления ω_i . Пусть количество энергии, отраженной в направлении ω_0 , равно L_0 , а количество энергии, пришедшей с направления ω_i , равно E_i . Тогда ДФОС равна:

$$\text{ДФОС}(\omega_0, \omega_i) = \frac{L_0}{E_i},$$

где ω_0 и ω_i – дифференциальные телесные углы, которые можно однозначно задать двумя углами в сферической системе координат (азимут и зенитом).

Приближения ДФОС

ДФОС – функция, зависящая от двух дифференциальных телесных углов. Её можно представить как отображение пространства точек на сфере в пространство функций на сфере [1, 2]:

$$\text{ДФОС} : \omega_i \rightarrow f.$$

Для идеальной зеркальной поверхности ДФОС будет являться обратной изотропной симметричной вокруг направления ω_i функцией, то есть:

$$\text{ДФОС}(\omega_i, \omega_k) = \text{ДФОС}(R_{\pi}\omega_k, R_{\pi}\omega_i),$$

где R_{π} – оператор поворота сферы вокруг центра через нормаль к поверхности на π .

Для идеальной зеркальной поверхности ДФОС будет равна 1, если направление ω_i есть ни что иное, как отражение направления ω_i от нормали к поверхности, а в остальных случаях – 0.

Таким образом, ДФОС можно представить в следующем виде:

$$\text{ДФОС}(\omega_i, \omega_k) = (R_{\omega_k} f)(\omega_i),$$

где ω_k – направление отражения от нормали к поверхности; R_{ω_k} – оператор поворота на соответствующие направлению ω_k азимутальный и зенитный

углы; f – функция вида $f(\theta, \varphi) = \begin{cases} 1, & \varphi = \theta = 0 \\ 0, & \text{otherwise} \end{cases}$, разложенная по базисным функциям.

Остаётся сформировать оператор поворота базисных функций на заданные азимутальный и зенитный углы. Это линейный оператор, представимый в виде квадратной матрицы и действующий на коэффициенты разложения. В силу инвариантности весовых классов базисных функций относительно оператора поворота, матрица будет иметь блочно-диагональный вид.

Элементы подматриц данной матрицы вычисляются как [4]:

$$R_i^{jj}(\alpha, \beta) = \int_0^{2\pi} \int_0^{\pi} Y_i^j(\theta + \alpha, \varphi + \beta) Y_i^j(\theta, \varphi) \sin(\theta) d\theta d\varphi,$$

что соответствует повороту базиса на углы α и β .

Замечание: индексация в подматрицах ведётся с середины подматрицы (т. е. если подматрица, например, 3×3 , то индексация будет в интервале $[-1; 1]$).

При $\beta=0$ вычисление подматриц поворота сильно упрощается:

$$R_i^{jj}(\alpha) = \begin{cases} \cos(j \cdot \alpha), & j = i \\ \sin(|j| \cdot \alpha), & j = -i \\ 0, & |j| \neq |i| \end{cases}$$

Оператор поворота факторизуется:

$$R_i(\alpha, \beta) = R^x \left(-\frac{\pi}{2} \right) R^z(\beta) R^x \left(\frac{\pi}{2} \right) R^z(\alpha).$$

Так как рассматриваемая функция симметрична относительно оси X , то первый оператор поворота тождественен:

$$R_i(\alpha, \beta) = R_i^z(\beta) R_i^x \left(\frac{\pi}{2} \right) R_i^z(\alpha).$$

В результате поворот сводится к умножению на три матрицы, две из которых имеют мало ненулевых элементов, а матрица поворота вокруг оси X может быть заранее вычислена.

Двулучевая функция рассеянного света

ДФРС определяется как отношение количества энергии (света), рассеянной в направлении ω_0 , к количеству энергии, попавшей на поверхность со всех направлений видимой полусферы. Пусть количество энергии, рассеянной в направлении ω_0 , равно L_0 , тогда будет рассматриваться равномерное распределение рассеянного света: ДФРС является независимой от направления взгляда (ω_0) [1, 2]:

$$\text{ДФРС}(\omega_i) = \frac{L}{E_i},$$

где L – количество энергии, рассеянной одинаково во всех направлениях. Включив в ДФРС функцию видимости, мы внесем в неё учёт самозатенения:

$$\text{ДФРС}(\omega_i) = \frac{L}{E_i} V_i,$$

где ω_0 – принимает значение 0 или 1, в зависимости от того, перекрывается поток света приходящий в данном направлении геометрией объекта

или нет. Таким образом, с помощью ДФРС можно представить любую точку объекта без видозависимого освещения (бликов и т. д.).

Приближение ДФРС

ДФРС является ни чем иным, как скалярным полем на сфере (так как ω_i можно однозначно представить в виде точки со значением на единичной сфере).

В качестве функционального базиса на сфере подходит базис присоединённых сферических функций действительного переменного, образующих на сфере полный ортонормированный базис функций [1–3]:

$$y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} K_l^m \cos(m\varphi) P_l^m(\cos(\theta)), & m > 0 \\ \sqrt{2} K_l^m \sin(-m\varphi) P_l^{-m}(\cos(\theta)), & m < 0, \\ K_l^0 P_l^0(\cos(\theta)), & m = 0 \end{cases}$$

где $K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$ – коэффициент нормировки; $P_l^m(x)$ – присоединённый многочлен Лежандра.

ДФРС представляется в данном базисе в виде:

$$f_x(\theta, \varphi) \approx \sum_{l=0}^n \sum_{m=-l}^l c_l^m Y_l^m(\theta, \varphi),$$

где коэффициенты определяются как:

$$c_l^m = \int_s f_x(s) Y_l^m(s) ds.$$

Так как ДФРС задана численно, и вычисление функции видимости для каждого направления для многих сцен является задачей трудоёмкой в плане вычислений, то встаёт вопрос о том, как наиболее точно вычислить коэффициенты, посчитав при этом как можно меньше значений ДФРС. Для этих целей выбран метод Монте-Карло. Он имеет линейную сходимость вне зависимости от размерности интеграла:

$$c_l^m \approx \frac{\sum_{s \in \Omega} f_x(s) Y_l^m(s)}{4\pi \|\Omega\|} + o\left(\frac{1}{\|\Omega\|}\right),$$

где s – точка на единичной сфере.

Множество определим следующим соотношением:

$$\Omega = \{\omega_i = (2 \arccos(\sqrt{1-x_i}), 2\pi y_i) \mid x_i, y_i \in R(0,1)\}.$$

Таким образом обеспечивается равномерное распределение направлений в полном телесном угле.

Для каждого направления заранее рассчитываются значения базисных функций. После чего идёт проекция ДФРС каждой вершины на заранее выбранное количество базисных функций.

Визуализация результата

Яркость каждой точки объекта вычисляется как [2, 3]:

$$L = kL_D + (l-k)L_S,$$

где k – коэффициент диффузного отражения ($k < 1$); l – коэффициент полного отражения.

Диффузная составляющая

$$L_D = \sum_l \sum_m c_{(d)}^m c_{(l)}^m.$$

Отражённая составляющая

$$L_S = \sum_i c_{(s)}^i c_{(l)}^i \frac{\sum_j c_{(d)}^j c_{(s)}^j}{\cos(\eta)},$$

где $\frac{\sum_j c_{(d)}^j c_{(s)}^j}{\cos(\eta)}$ – функция видимости, вычисляемая

для расчёта самозатенения; η – угол между направлением ω_i и нормалью к поверхности.

Преимущества и недостатки метода

Преимущества:

- Хорошо адаптирован для расчёта глобального освещения или освещения от неточечных источников.
 - Исключены ошибки, связанные с поворотом источников света вследствие выбранного базиса на сфере.
 - Позволяет полностью перевести расчёт освещения на процессор видеокарты.
- Недостатки:*
- Необходимость выполнения предварительного расчёта коэффициентов разложения ДФРС и ДФОС для каждой вершины сцены.
 - Метод слабо применим только для динамичной геометрии.
 - Возможно возникновение неточностей в местах с малой плотностью геометрии и около объектов, отбрасывающих тени, – артефакты освещения.

Применение технологии nVidia CUDA

для ускорения предварительного расчёта

Центральные процессоры разработаны для самого широкого спектра задач и ограниченно используют возможности параллельных вычислений. Увеличение их производительности в основном связано с увеличением тактовой частоты и размеров кэш-памяти. Программирование на ЦПУ (центральных процессорных устройствах) для ресурсоёмких научных вычислений подразумевает тщательное структурирование данных и порядка инструкций для эффективного использования всех уровней кэш-памяти [5, 6]. Напротив, графические процессоры изначально спроектированы для параллельной обработки данных (например, закрасивания массива пикселей), когда обработка отдельных элементов может производиться независимо. ГПУ (графические процессорные устройства) являются примером потоковых процессоров, которые используют явный параллелизм по данным для увеличения скорости вычислений и уменьшения зависимости от задержек доступа к памяти. За последние 10 лет ГПУ эволюционировали

от устройств с фиксированной функциональностью для ускорения примитивных графических операций до программируемых процессоров, превосходящих обычные ЦПУ при выполнении векторизуемых операций.

В концепции потоковых вычислений данные представляются в виде потоков из независимых элементов, а независимые стадии обработки (выборы операций) – в виде ядер. Ядра можно представить как функции преобразования элементов входных потоков в элементы выходных потоков. Такое представление позволяет параллельно применять ядро сразу ко многим элементам входного потока. Вместо подгонки формата данных под архитектуру кэшей каждого конкретного ЦПУ, данные разбиваются на потоки, разные по содержанию и способу обработки. Предсказуемость потокового доступа к памяти позволяет выполнять его параллельно с вычислениями.

nVidia CUDA – это технология параллельных вычислений, позволяющая разрабатывать программы для ГПУ на стандартном языке программирования *C* с несколькими расширениями, причём производительность этих программ хорошо масштабируется числом ядер. Она разработана компанией *nVidia* и в настоящий момент работает только с её ГПУ, начиная с поколения *G8x*.

Преимущества технологии *CUDA* перед использованием стандартных графических библиотек *OpenGL* и *DirectX*:

- программирование на широко известном языке *C* с несколькими простыми расширениями;
- произвольная адресация при записи в память;
- доступ к программируемой разделяемой памяти;
- значительно меньшие накладные расходы на взаимодействие ГПУ и ЦПУ, некоторые операции выполняются асинхронно;
- побитовые операции над целыми числами, отсутствующие в шейдерной модели *SM3.0*;
- двойная точность операций с плавающей запятой.

Недостатки:

- примитивные структуры данных;

- нет механизмов абстракций: средств объектно-ориентированного и функционального программирования;
- нет управления обработкой с ГПУ (в коде ядра нельзя запускать и останавливать ядра, инициировать обмен данными с ЦПУ);
- обработка параллельной записи по одному адресу только для некоторых целочисленных операций;
- отсутствие поддержки рекурсивных функций.

Принцип параллельной обработки в данной работе

Учитывая проблемы исходной модели и возможности, предоставляемые технологией *CUDA*, очевиден вариант параллельного расчёта коэффициентов разложения базисных гармоник для отдельных вершин. При учёте возможности взаимодействия интерфейса *OpenGL* непосредственно с *CUDA*, обмен памяти сокращается, поскольку все необходимые данные (координаты вершин, нормали и т. д.) вместе с данными, необходимыми *CUDA* для расчётов, размещаются в памяти видеокарты.

Результаты

В результате выполненной работы было создано приложение, позволившее оценить как качество итогового освещения, так и разницу в производительности графических и центральных процессоров.

Для тестирования использовалась система на базе центрального процессора *AMD Athlon64 X2 mobile*, 2 Гб оперативной памяти и видеокартой *nVidia GeForce 8400GS mobile* с памятью, расширяемой до 384 Мб.

В качестве тестовой сцены применялась модель помещения с мебелью, состоящая из примерно 20 тыс. полигонов. Сравнительные результаты визуализации приведены в таблице.

Из таблицы видно, что видеокарта даже самой нижней бюджетной модели обходит по скорости работы центральный процессор несколько более высокого ранга. Однако необходимо отметить, что данные результаты были получены не с первого ра-

Таблица. Сравнительные результаты визуализации

Размер базиса	32	48	64	128	256
Скорость визуализации (кадры/с)	18,5	18,5	18,5	18,5	18,5
Ступенчатость краёв тени	Резкая	Заметная	Заметная	Малозаметная	Незаметная
Равномерность текстуры тени	Тень закрашена неравномерно	Тень закрашена равномерно, но ненасыщенно	Тень закрашена равномерно, но ненасыщенно	Тень закрашена равномерно	Тень закрашена равномерно
Наличие артефактов	Повсеместно во всей сцене	Повсеместно во всей сцене	Только в местах со сложной геометрией	Только в местах со сложной геометрией	Только в местах со сложной геометрией
Время счёта на ЦПУ	38 с	62 с	5 мин 42 с	35 мин 12 с	4 ч 20 мин
Время счёта на ГПУ	42 с	55 с	2 мин 15 с	15 мин 6 с	2 ч 18 мин

за, т. к. несмотря на привычный язык программирования ГПУ, разработка приложений для *CUDA* содержит массу технических тонкостей, особенно при работе с памятью. Невнимательное отношение ко всем деталям может привести к тому, что приложение на ГПУ будет работать значительно медленнее, чем на ЦПУ.

Немаловажным моментом является то, что размер базиса не влияет скорость итоговой визуализации.

СПИСОК ЛИТЕРАТУРЫ

1. Cohen M., Wallace J. Radiosity and Realistic Image Synthesis. – Cambridge: Academic Press Professional, 1993. – P. 8–10, 89, 190–194.
2. Капляян А.А. Технология предрасчёта освещения модели для получения мягких теней от динамических неточечных источников света // Gamedev.ru. 2005. URL: http://kri-conf.ru/2005/rec/KRI_2005_Programming_03apr_gal12_01_Anton_Kaplanyan_Akella.ppt (дата обращения: 25.06.2011).
3. Sloan P.-P. Stupid Spherical Harmonics // Game Developers Conference 2008, February 2008 (updated 2/10/2010). URL: <http://www.ppsloan.org/publications/StupidSH36.pdf> (Дата обращения 25.06.2011).
4. Капляян А.А. Advances of Realtime Rendering // SIGGRAPH 2010. – Лос-Анджелес, 2010. URL: <http://advances.realtimeren->

Выводы

1. Построена параллельная реализация расчёта освещения в базисе сферических функций.
2. Показана возможность применения графических процессоров в задачах компьютерной графики в целях оптимизации вычислений, не связанных непосредственно с визуализацией.
3. Произведено сравнение быстродействия центральных и графических процессоров в задаче потоковых вычислений.

[dering.com/s2010/Kaplanyan-CryEngine3 %28SIGGRAPH%202010 %20Advanced%20RealTime%20Rendering%20Course%29.pdf](http://dering.com/s2010/Kaplanyan-CryEngine3%28SIGGRAPH%202010%20Advanced%20RealTime%20Rendering%20Course%29.pdf) (дата обращения: 25.06.2011).

5. Боярченко А.С., Поташиников С.И. Использование графических процессоров и технологии CUDA для задач молекулярной динамики // Вычислительные методы и программирование. Т. 10. 2009. URL: http://num-meth.srcc.msu.ru/zhurnal/tom_2009/pdf/v10r102.pdf (дата обращения: 25.06.2011).
6. Евстигнеев Н.М. Интегрирование уравнения Пуассона с использованием графического процессора технологии CUDA // Вычислительные методы и программирование. – 2009. – Т. 10. URL: http://num-meth.srcc.msu.ru/zhurnal/tom_2009/pdf/v10r128.pdf (дата обращения: 25.06.2011).

Поступила 01.09.2011 г.