Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный
технический университет»

На правах рукописи

Коротких Анна Анатольевна

Исследование и программная реализация параллельных алгоритмов обратной трассировки лучей

Направление подготовки 02.04.03
«Математическое обеспечение и администрирование
информационных систем»

АВТОРЕФЕРАТ
МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

2017

Работа выполнена в ФГБОУ ВО
«Комсомольский-на-Амуре государственный технический университет»

Научный руководитель: кандидат физико-математических наук, доцент, доцент кафедры «Прикладная математика и информатика»
Лошманов Антон Юрьевич

Рецензент: кандидат физико-математических наук, научный сотрудник Института машиноведения и металлургии ДВО РАН
Севастьянов Георгий Мамиевич

Защита состоится 22 июня 2017 г. в 13.30 часов по адресу: 681000, г. Комсомольск-на-Амуре, пр. Ленина, 27, ауд. 321/3.

Автореферат разослан 16 июня 2017 г.

Секретарь ГЭК А.А. Сиротин

ОБЩАЯ ХАРАКТЕРИСТИКА ДИССЕРТАЦИОННОЙ РАБОТЫ

Актуальность темы. На протяжении практически всей истории компьютерной графики наиболее актуальной задачей являлось получение изображений, неотличимых от реальной фотографии. Обычно для того чтобы получить изображение, задается информация о геометрии визуализируемой сцены, информация о материалах (степень отражения, преломления и т.п.), позиция и яркость источников света, положение камеры.

В основе алгоритмов фотореалистичного синтеза изображений лежат различные физические принципы. Построив достаточно сложную модель света, учитывающую все законы физики, можно добиться неотличимости синтезированного изображения от фотографии. Однако, как правило, чем больше модель света учитывает различных эффектов, тем большей вычислительной сложностью обладают алгоритмы синтеза изображений.

Рендеринг – процесс создания реалистичных изображений на экране, использующий математические модели и формулы для добавления цвета, тени и т.д. Одна из основных задач рендеринга - создание изображения, максимально приближенного к действительности. Понятно, что для этого компьютер должен учитывать все законы реального мира. По этой причине количество вычислений может быть достаточно большим.

Целью данной работы является программная реализация параллельных алгоритмов обратной трассировки лучей и исследование их влияния на скорость генерации растровых изображений.

Для достижения указанной цели поставлены следующие *задачи*:

– рассмотреть и изучить магистерскую диссертацию В.С. Валовой 2015 г. «Исследование и программная реализация параллельных алгоритмов и технологий платформы WPF для генерации растровых изображений»;

– изучить основные понятия многопоточного программирования и его подходы;

– изучить и сравнить различные методы нахождения пересечения прямой и параллелепипеда;

– исследовать влияние применения параллельных алгоритмов обратной трассировки лучей на скорость генерации растрового изображения;

– программная реализация параллельных вычислений алгоритма обратной трассировки лучей и вывода изображения на экран.

Объектом исследования является процесс рендеринга трехмерных сцен.

Предметом исследования являются параметры, влияющие на скорость рендеринга трехмерных сцен.

Для решения поставленных задач использовались следующие *методы* исследования: теоретические (сравнение, анализ) и эмпирические (тестирование, изучение литературы и результатов деятельности).

Научная новизна исследования заключается в следующем:

– применение многопоточности к алгоритму обратной трассировки лучей (так как этот метод исключительно удачно подходит для параллельных вычислений);

– исследование применения совместного использования параллельных вычислений и новых технологий WPF для увеличения скорости рендеринга.

Достоверность и обоснованность результатов исследования. Основные положения и выводы, полученные в диссертации, достаточно обоснованы и аргументированы. Сформулированная в диссертации научная задача, заключающаяся в применение параллельных алгоритмов и новых технологий (WPF) для увеличения скорости рендеринга растровых изображений, была исследована и решена на основе корректного использования многопоточного программирования, технологии Windows Presentation Foundation.

Достоверность основных выводов и результатов диссертации подтверждается:

1. Обоснованием выбора технологии Windows Presentation Foundation вместо интерфейса Windows Forms, применения многопоточного программирования при рендеринге трехмерных сцен;
2. Использованием современного апробированного научно-методического аппарата для формализации и решения сформулированной в диссертации научной задачи;
3. Полнотой опубликования результатов исследования и их широко апробацией.

Практическая значимость полученных результатов исследований состоит в том, что разработанные алгоритмы позволяют увеличить скорость математических вычислений и вывод растрового изображения на экран.

В основу диссертационной работы положены результаты исследований:

1. Исследование влияния параллельных алгоритмов на скорость генерации растрового изображения.

2. Исследование влияния применения новых технологий (технология Windows Presentation Foundation) на вывод растрового изображения.

Апробация результатов. Результаты работы докладывалась на:

– 46-ой научно-технической конференции студентов и аспирантов «Научно-техническое творчество аспирантов и студентов», Комсомольск-на-Амуре, апрель 2016 г.

– 47-ой научно-технической конференции студентов и аспирантов «Научно-техническое творчество аспирантов и студентов», Комсомольск-на-Амуре, апрель 2017 г.

Публикации. По результатам выполненных в диссертации исследований автором опубликовано 2 работы:

– журнал «Евразийский союз ученых» (ЕСУ) № 1 (22). Ч. 3. 2016;

– сборник материалов 46-ой научно-технической конференции студентов и аспирантов «Научно-техническое творчество аспирантов и студентов» (Комсомольск-на-Амуре, апрель 2016 г.).

Структура и объем. Магистерская диссертация состоит из введения, общей характеристики, трех глав, заключения и списка литературы. Объем работы – 105 страниц, в том числе 67 рисунков, 2 таблицы и 1 приложение.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Введение раскрывает актуальность темы, определяются цели и задачи исследования, объект, предмет, указываются научная новизна, практическая значимость, достоверность и обоснованность результатов исследования.

В первой главе дается понятие многопоточного программирования, его основные подходы и классификация.

Многопоточное программирование является одной из важных особенностей языка C#. Отличительной чертой многопоточной программы является то, что она состоит из двух или более частей, выполняемых параллельно. Каждая часть такой программы называется потоком и определяет отдельный путь выполнения команд. Таким образом, многопоточная обработка является особой формой многозадачности.

Благодаря встроенной в C# поддержке многопоточной обработки сводятся к минимуму или вообще устраняются многие трудности, связанные с организацией многопоточной обработки в других языках программирования.

Всего различают две разновидности многозадачности: на основе процессов (многозадачность организуется для параллельного выполнения программ) и на основе потоков (многозадачность организуется для параллельного выполнения отдельных частей одной программы).

С выпуском версии 4.0 в среде .NET Framework появились два важных дополнения, имеющих отношение к многопоточным приложениям:

1. TPL (библиотека распараллеливания задач) – усовершенствует многопоточное программирование двумя основными способами: она упрощает создание и применение многих потоков и позволяет автоматически использовать несколько процессоров. Благодаря этим двум особенностям, TPL рекомендуется в большинстве случаев к применению для организации многопоточной обработки.

Применяя TPL, параллелизм в программу можно ввести двумя основными способами:

Параллелизмом данных. Основная идея заключается в том, что одна операция над совокупностью данных разбивается на два параллельно выполняемых потока или больше, в каждом из которых обрабатывается часть данных.

Параллелизмом задач – вычислительная задача разбивается на несколько относительно самостоятельных подзадач и каждый процессор загружается своей собственной подзадачей. Чем больше подзадач, тем большее число процессоров можно использовать, тем большей эффективности можно добиться.

2. PLINQ (параллельный язык интегрированных запросов) – дает возможность составлять запросы, для обработки которых автоматически используется несколько процессоров, а также принцип параллелизма, когда это уместно.

Во второй главе рассматриваются методы трассировки лучей, выявляются преимущества и недостатки метода обратной трассировки лучей, а так же приводятся примеры пересечения луча с некоторыми объектами, например, с шаром, эллиптическим цилиндром, с круговым тором.

Трассировка лучей - это метод, позволяющий восстановить непрерывное изображение из дискретного пиксельного представления

(матрицы дисплея) с помощью математической модели интенсивности освещения. Каждой точке плоскости экрана соответствует точка *x*, освещенность в которой может быть рассчитана, например, при помощи интеграла вида:

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

$L(\phi_i, \theta_i)$ - это функция, описывающая общее освещение, падающее в точку *x* под всеми возможными углами в пределах полусферы. $R(\phi_i, \theta_i, \phi_r, \theta_r)$ - двунаправленная функция распределения отражения. Она полностью описывает характер взаимодействия света с конкретной поверхностью, связывая интенсивность и угол падающего света с интенсивностью и углом отраженного или пропущенного прозрачной поверхностью света.

Метод обратной трассировки лучей был разработан в начале 80-х годов и применялся для создания высококачественных реалистичных изображений не в реальном масштабе времени. К примеру, построение сцен на машине с процессором 80286 занимало несколько дней, а то и недель. Не смотря на такую медленную скорость сцены, качество получаемого изображения было максимально приближено к реальности. Метод обратной трассировки лучей позволяет получать такие эффекты, как отражение, преломление, затенение и т.д. Достоинство данного метода заключается также в том, что можно работать со сценами, заданными не набором треугольников для аппроксимации гладких поверхностей, а, собственно, самими этими поверхностями в математической форме. Таким образом, задача построения изображения такой поверхности решается точно, а не приближённо, как если бы такая поверхность была аппроксимирована набором треугольников.

Третья глава посвящена алгоритмам поиска пересечений, а именно анализу эффективности методов нахождения пересечения прямой и параллелепипеда.

Алгоритмы нахождения пересечений (или в более простом случае, определение факта пересечения или не пересечения) различных трехмерных объектов очень важны при решении геометрических задач. Алгоритмы нахождения пересечений прямой и линейных или квадратичных поверхностей достаточно широко изучены. Однако, решение задач о нахождении пересечений с другими поверхностями представляет определенные трудности. Часто в компьютерной графике прибегают к приему, когда объект помещается в более простой объект (экстенд), пересечение с которым достаточно легко вычислить. В

качестве таких экстендов используются кубы (параллелепипеды), сферы, цилиндры и т.д.

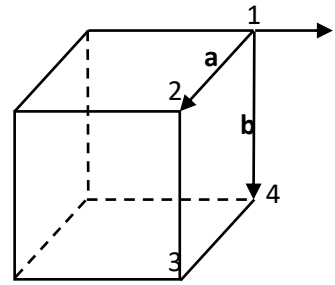
В данной работе рассматриваются различные методы определения пересечений прямой и параллелепипеда (далее в работе будет использоваться термин куб). Предложены два новых метода решения этой задачи. Представленные методы анализируются с точки зрения эффективности работы (быстродействия).

Алгоритмы нахождения пересечений играют важную роль во многих геометрических задачах и CAD/CAM-системах. Пересечение линий и замкнутых поверхностей может рассматриваться как обобщение хорошо известной в компьютерной графике задачи отсечения. Пересечение прямой или луча с поверхностью – это ключевая проблема, решаемая во всех алгоритмах трассировки лучей.

Рассматриваемые методы пересечения с плоскостями:

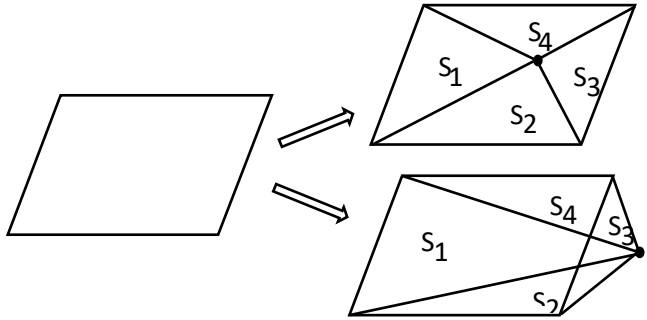
Пересечение с шестью плоскостями

Наиболее простой способ определения пересечения прямой и куба – это определение пересечений прямой с плоскостями, совпадающими с шестью гранями куба. При этом после нахождения пересечения с плоскостью необходимо проверить попадает ли точка пересечения внутрь соответствующей грани.



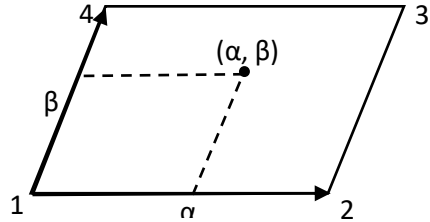
Таким образом, этот метод разбивается на две подзадачи: сначала определить пересечение с плоскостью, а затем уже входит ли точка в грань.

Проверка попадания точки внутрь грани может быть выполнена несколькими способами. Один из них заключается в вычислении площадей четырехугольника и четырех треугольников. Если сумма площадей четырех треугольников ($S_1 + S_2 + S_3 + S_4$) равна площади четырехугольника S, то точка принадлежит грани.



Двухпараметрическое представление грани

Существенный недостаток предыдущего метода – это то, что необходимо найти точку пересечения прямой и плоскости, а уже потом проверить попадание в грань. Можно сократить объем вычислений, если находить только точки пересечения, попадающие в рассматриваемую грань.



Если рассмотреть произвольный параллелограмм, то координаты любой точки, принадлежащей данному параллелограмму могут быть записаны такой системой:

$$\begin{cases} x = (x_2 - x_1) \propto + (x_4 - x_1)\beta + x_1, \\ y = (y_2 - y_1) \propto + (y_4 - y_1)\beta + y_1, \\ z = (z_2 - z_1) \propto + (z_4 - z_1)\beta + z_1, \end{cases}$$

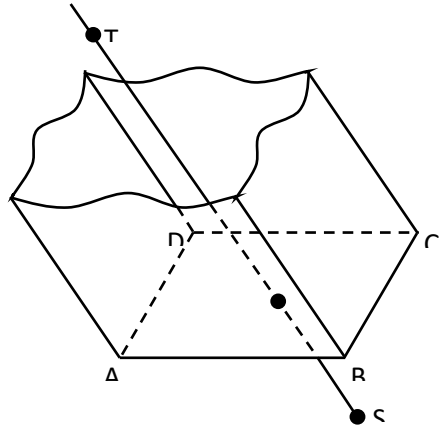
где $\propto \in [0;1]$ и $\beta \in [0;1]$.

И, если рассмотреть эту систему совместно с системой параметрического представления прямой, проходящей через точки T и S, то получим систему трех линейных алгебраических уравнений с тремя неизвестными ($\propto$, β и t). Решить её можно любым способом.

Пересечение полупространств

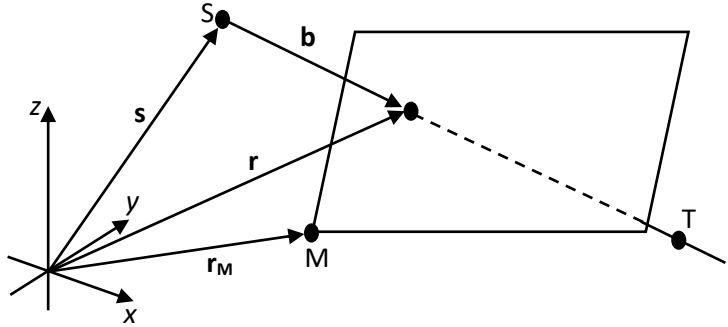
Рассмотрим четырехугольник и прямую. Нам необходимо проверить пересекает или нет прямая рассматриваемый четырехугольник. Построим четыре плоскости, параллельные прямой и проходящие через стороны четырехугольника. Прямая пересекает четырехугольник, если

она лежит внутри полученного таким образом четырехугольного цилиндра.



Подход, основанный на алгоритме отсечения Кируса-Бека

Рассмотрим одну из граней куба.



Введем понятие «пятно грани» – это точка, принадлежащая грани. В качестве пятен граней куба удобно использовать его вершины. На рисунке в качестве пятна будет точка M с радиус-вектором $\vec{r}_M$. Пусть $\vec{r}$ – радиус-вектор точки пересечения (x, y, z) прямой ST и грани, $\vec{n}$ – вектор внешней нормали рассматриваемой грани. Тогда уравнение плоскости, в которой лежит грань, может быть записано в векторном виде:

$$(\vec{r} - \vec{r}_M) \cdot \vec{n} = 0. \qquad (2)$$

Но для радиус-вектора $\vec{r}$ справедливо следующее соотношение:

$$\vec{r} = \vec{s} + \vec{b}t,$$

где $\vec{b} = (x - S_x)\vec{i} + (y - S_y)\vec{j} + (z - S_z)\vec{k}$, а t – вещественный параметр.

Подставляя выражение для $\vec{r}$ в (2) и выражая параметр t, получим:

$$t = \frac{\vec{n} \cdot (\vec{r}_M - \vec{s})}{\vec{n} \cdot \vec{b}}. \quad (3)$$

Анализируя теперь знаменатель выражения (3), можно сказать как направлен вектор $\vec{b}$ относительно рассматриваемой грани (плоскости) – во внешнее полупространство или во внутреннее. Так, если $\vec{n} \cdot \vec{b} > 0$, то вектор выходит во внешнее полупространство, а если $\vec{n} \cdot \vec{b} < 0$ – входит во внутреннее. В связи с этим можно вычислить значения соответствующих параметров t_{out} и t_{in}.

Если же $\vec{n} \cdot \vec{b} = 0$, то вектор $\vec{b}$ и соответствующая грань (плоскость) параллельны и не пересекаются. В этом случае надо проанализировать знак числителя выражения (3) и принять или не принять решение об отбрасывании отрезка целиком. Так, если $\vec{n} \cdot (\vec{r}_M - \vec{s}) < 0$, то отрезок лежит вне той части пространства, которую занимает куб. Поэтому в этом случае можно сразу сделать вывод, что данный отрезок не пересекается с кубом.

Было проведено сравнительное исследование скорости генерации и построения растрового изображения трехмерной сцены размером 564 × 839 точек с использованием описанных выше методов определения пересечений прямой и куба. Для исследования было выбрано три компьютера со различными характеристиками.

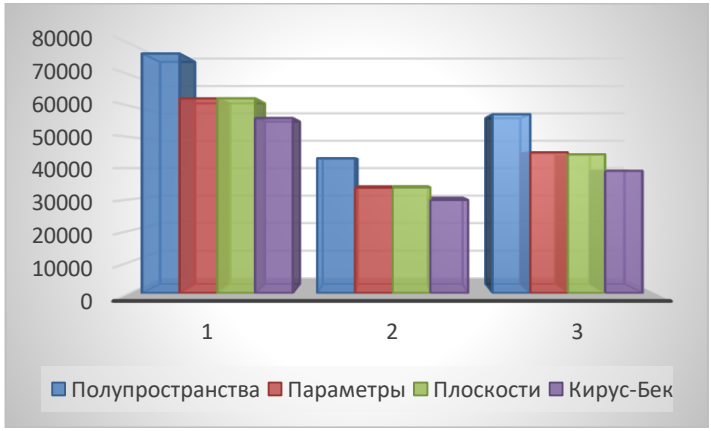


Рисунок 1 – Результаты эксперимента.

Дальнейшее направление работы по ускорению рендеринга сцены было связано с распараллеливанием метода обратной трассировки лучей.

В идеальном случае система из n процессоров могла бы ускорить вычисления в n раз. Естественно, встает вопрос о том, на какое реальное ускорение можно рассчитывать. Ответ на этот вопрос в какой-то мере дает закон Амдала.

Закон Амдала – иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей.

Результаты проведенных исследований по распараллеливанию метода обратной трассировки лучей показывают качественное совпадение с кривой Амдала (рисунок 2).

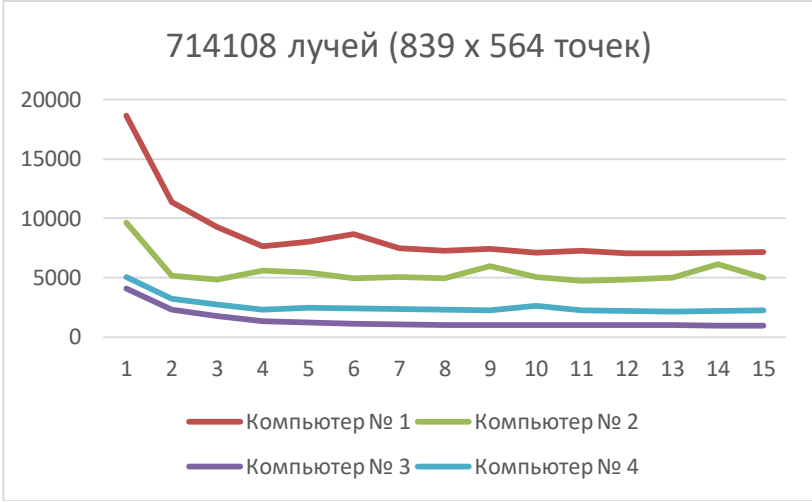


Рисунок 2 – Результаты эксперимента

СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ИССЛЕДОВАНИЯ

1. Носков А.А., Коротких А.А., Лошманов А.Ю. Анализ эффективности методов нахождения пересечения прямой и куба в алгоритме трассировки лучей // журнал «Евразийский союз ученых» (ЕСУ) № 1 (22). Ч. 3. 2016. С. 129–133.

2. Коротких А.А., Лошманов А.Ю. Трассировка лучей. Алгоритмы поиска пересечений// сборник материалов 46-ой научно-технической конференции студентов и аспирантов «Научно-техническое творчество аспирантов и студентов» апрель 2016. С. 300-301.