МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математической физики

Дипломная работа

Моделирование отражения радиоволн от сложных полигональных моделей

Выполнил:
Ефремов Андрей Сергеевич,
502 группа
Научный руководитель:
к.ф-м.н. Березин С. Б.
Рецензент:
к.ф.-м.н. Сазонов В.В.

Москва 2013

Введение

В радиолокации интересной является задача моделирования распространения электромагнитного сигнала и определение зон, затемененных объектом, а также областей, распространения переотраженного сигнала. Особенно это важно при использовании радиотехнических систем на протяженных объектах, так как возникают дополнительные ошибки измерения радиолокационных параметров, например, при стыковках космических кораблей и работе космических станций [1]. Так при стыковке в зону между принимающей антенны и передающей попадают элементы конструкций, порождая переотражение и искажение сигнала. Возможным решением этой проблемы является выбор зон пространства, в котором такое влияние переотражений минимально.

Целью данной работы является создание программного обеспечения, позволяющее визуализировать распространение сигнала в пространстве и его отражение от сложных полигональных моделей.

Однако, при попытках детального моделирования описанного выше процесса, мы неминуемо приходим к тому, что используемые компьютерные модели космических объектов обладают огромным количеством полигонов, а вычислительных ресурсов даже современных персональных компьютеров не хватает для их обработки.

Проведем простые предварительные расчеты. Если наша модель содержит 10 000 полигонов, то для того, чтобы определить, какие грани являются первичными для всей модели нам потребуется обработать порядка 10^8 полигонов (без алгоритмических оптимизаций, «влоб»). Мы же в работе будем использовать модель космического корабля «Союз» [2], содержащий 152 000 полигонов. Это число внушительно, просто необходимы большие мощности. Поэтому для расчетов будем использовать вычисления на графических ускорителях и технологию CUDA.

Графические ускорители выросли из задач обработки и формирования изображения на экране компьютера постепенно переродившись в массивно-параллельные процессоры общего назначения. Сам термин GPU (Graphics Processing Unit) относительно новый и впервые был использован корпорацией Nvidia, в качестве обозначения того, что графические ускорители стали мощными программируемыми устройствами пригодными для решения более широкого класса задач, не связанных с графикой [3].

Первые графические ускорители представляли из себя простые растеризаторы, однако эту простую задачу делали быстрее универсального процессора, что и привело к распространению графических ускорителей. Основная причина этого – ускоритель мог обрабатывать хоть и простую, но зато масштабную работу – обрабатывать сразу много отдельных пикселов [3].

По мере развития функциональность увеличивалась. Фактически графические ускорители стали представлять из себя SIMD-процессоры (Single Instruction Multiple Data), то есть параллельные устройства, способные одновременно выполнять одну и ту же операцию над многими данными. Экспоненциальный рост производительности и функциональности дал развитие направлению GPGPU (General-Purpose computing on Graphics Processing Units).

Все это открывает новые возможности при реализации приложений, требущих больших объемов специфических вычислений. И если раньше ресурсоемкие задачи и можно было решить, то только с использованием суперкомпьютеров и кластеров, то теперь это представляется возможным на обычном пользовательском компьютере.

На сегодняшний день существуют несколько технологий для разработки приложений, использующих для вычислений графические ускорители: OpenCL, CUDA, ATI Stream Technology. В нашей работе мы остановимся на технологии Nvidia CUDA и будем использовать её.

На то есть несколько причин. Nvidia CUDA хорошо зарекомендовала себя при решении многих ресурсоемких задач: моделирование гидродинамики [4], волн цунами [5], вычисления нейронных сетей [6].

Модели отражения

Радиоволна, как и любая другая электромагнитная волна, отражается от препятствий, причем препятствием является любая неоднородность электрических или магнитных параметров среды, то есть объект отражает электромагнитную энергию, в случае если проводимость, диэлектрическая или магнитная проницаемость отличается от соответствующих параметров среды. Так как от поверхности в результате процесса отражения отходит электромагнитная энергия, то можно считать, что отражатель сам является источником (вторичным) электромагнитного излучения. Это явление можно также представить себе следующим образом: при попадании волны на поверхность предмета, она в нем порождает вынужденные колебания зарядов, которые синхронны с колебаниями падающей волны. Колебания зарядов создают токи смещения и токи проводимости, и сами становятся источниками излучения. Каждый такой элементарный ток в рамках достаточного малого точечного объема можно считать источником новой сферической волны. Общий результат таких элементарных токов (в соответствии с принципом Гюйгенса-Френеля [8]) суммируется с различными фазовыми соотношениями и может принимать разные значения, а падающая волна отражается во всех направлениях. Значит, энергия переизлучается в различных направлениях неравномерно [7].

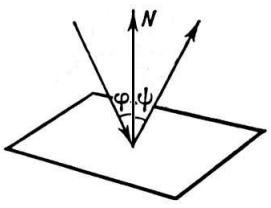
Радиоволны и световые волны имеют одну и ту же физическую электромагнитную природу, поэтому многие модели поведения света известные из оптики (кстати как и способы получения фотореалистических изображений компьютерной геометрии) применимы в нашем случае. Рассмотрим основные модели отражения радиоволн: зеркальное, диффузное и резонансное. Подробнее информацию об этом можно найти в [7].

Зеркальное отражение

Условие возникновения: линейные размеры отражающей поверхности много больше длины волны ($l >> \lambda$), а сама поверхность является достаточно гладкой ($h << \lambda$), где l – наименьшией линейный размер цели; λ – длина волны; h – высота неровностей поверхности.

Свойства зеркального отражения известны еще с давних времен и являются следствиями применения принципа Ферма к отражающей поверхности [9]:

- луч отраженный, луч падающей, а также нормаль отражающей поверхности, проведенная к точке падения, лежат в одной плоскости;
- угол падения равен углу отражения.



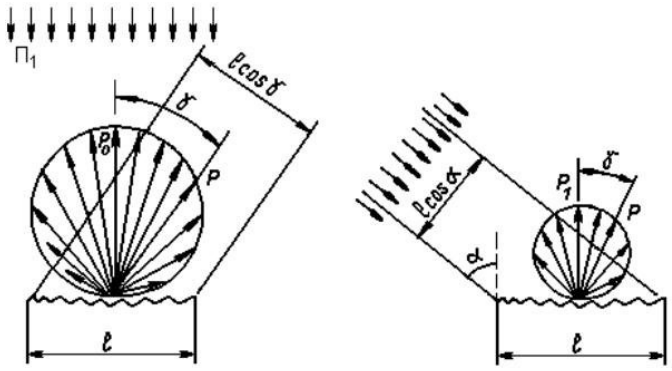
Интенсивность отражённого света характеризуется коэффициентом отражения и зависит от соотношения показателей преломления сред, от угла падения и поляризации падающего пучка лучей. Количественно эту зависимость выражают формулы Френеля [10].

Диффузное отражение

Условие возникновения: линейные размеры отражающей поверхности много больше длины волны ($l >> \lambda$), а неровности поверхности имеют порядок длины волны (или больше) и расположены хаотично, то есть поверхность является шераховатой (матовой) ($h \geq \lambda$).

При диффузном отражении энергия рассеивается во всех направлениях. Для матовых поверхностей применимы законы Ламберта [11]: для потока излучения, падающего нормально к матовой поверхности, мощность вторичного излучения под углом γ к нормали пропорциональна $cos\gamma$.

$$P = P_0 \cdot cos\gamma \tag{1}$$



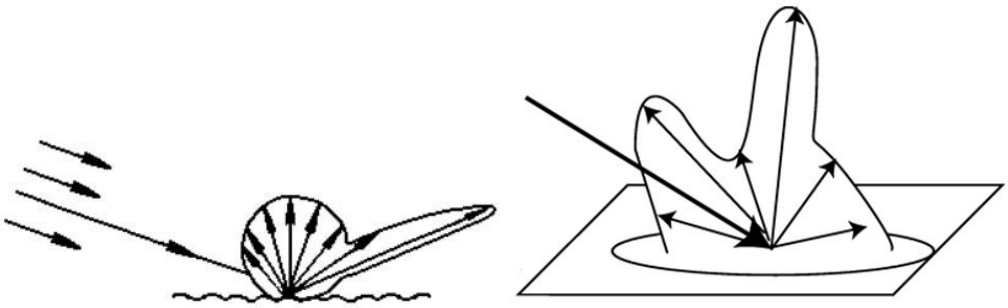
Если поток излучения падает под углом α к поверхности, то

$$P = P_0 \cdot cos\alpha \cdot cos\gamma \tag{2}$$

где P_0 – мощность, которую принимал бы приемник, если бы облучение шло с "зенита то есть с увеличением α уменьшается перехватываемый поверхностью падающий поток, отчего уменьшается освещенность и яркость.

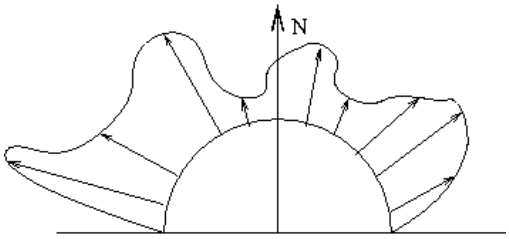
Условия зеркальности и диффузно отражающих поверхностей являются расплывчатыми. Здесь [7] дается строгая формулировка и вывод критерия различия зеркальных и матовых поверхностей. Скажем лишь, что тип поверхностей зависит сразу от трех факторов: высоты неровностей поверхности h, длины волны λ и угла падения ϕ.

Закон Ламберта – закон идеального рассеивания света, то есть это некоторая модель, к которой можно приблизиться, но которой не существует в естественных условиях. В реальной жизни поверхности обычно дают отражение, представляемое смесью диффузной и зеркальной состовляющих.



Кроме того, на практике, часто встречаются материалы имеющие несколько максимумов под разными углами и дающие в соответствующих направлениях из-за этого блеск (glossy materials).

Все вышеперечисленные свойства можно смоделировать воспользовавшись функцией, характеризующей степень отражения в данном направлении. Такая функция называется ДФО – двулучевая функцией отражения (BRDF – Bidirectional Reflection Distribution Function). Так для диффузных моделей ДФО равна константе.



Именно эту функцию мы и будем использовать в нашей работе. Таким образом достигается некоторая универсальность – наша программа сможет моделировать не только какой-то один конкретный вид отражения, а появляется возможность это задать в достаточно независимом виде.

Резонансное отражение

Список моделей отражения был бы не полон если не упомянуть резонансное отражение. Как уже было отмечено, падающая волна создает вынужденные колебания свободных или связанных зарядов в поверхности предмета. Тело, способное переизлучать электромагнитную волну, обладает собственной частотой колебаний частиц, несущих электрический заряд. Причем, если частота колебаний падающей волны совпадает с собственной частотой колебаний в поверхности, то имеем не что иное как явление резонансного отражения. В этом случае появляется ярко выраженная направленность вторичного излучения. Подробнее об этом явлении можно прочитать в [7].

Математическая постановка задачи

Предположим, у нас есть ненаправленная антенна, представляемая точечным источником (расстояние до поверхности значительно больше размеров антенны) с излучающей мощностью P_0. Тогда вся энергия распределяется равномерно по всей поверхности некоторой сферы радиуса R, площадь поверхности которой равна $4\pi R^2$. Таким образом, источник электромагнитного излучения создает плотность потока мощности

$$J = \frac{P_0}{4\pi R^2} \tag{3}$$

Следовательно, каждая точка поверхности $M(x, y, z)$, которая располагается в зоне прямой видимости, получает от источника излучения в секунду энергию мощности на единицу площади равную

$$dP_{in} = \left\langle \vec{J}(x, y, z) \cdot \vec{d\sigma} \right\rangle = J(x, y, z) \cdot d\sigma \cdot cos\phi, \tag{4}$$

где $d\sigma$ – окрестность точки $M(x, y, z)$, а ϕ – угол между вектором $\vec{J}(M)$ и нормалью к поверхности $d\sigma$, направленной в противоположную сторону от источника излучения.

Значит, учитывая (4), первично-освещенная поверхность получает в секунду энергию мощности, равную

$$P_{in} = \int\limits_S \left\langle \vec{J}(x, y, z) \cdot \vec{d\sigma} \right\rangle, \tag{5}$$

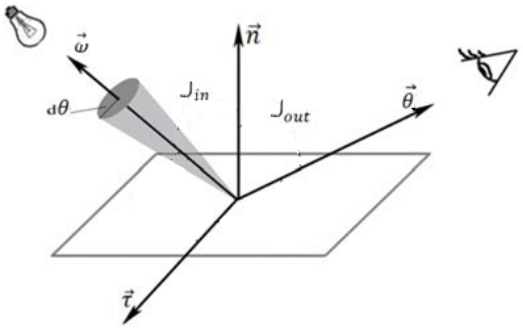
где S – первично-освещенная поверхность.

Часть энергии попадаемые в точки первично-освещенных поверхностей поглощается, часть отражается. Точки, отражающие электромагнитные волны ведут себя как вторичные источники электромагнитного излучения, а значит вышепроделанные умозаключения к ним тоже применимы.

После многократного отражения и переотражения получаем, что для каждой точки $M(x, y, z)$ справедливо

$$J_{out}(x, y, z, \vec{\omega}) = \int\limits_\Omega J_{in}(x, y, z, \vec{\theta}) \cdot f(x, y, z, \vec{\omega}, \vec{\theta}) \cdot \left\langle -\vec{\theta} \cdot \vec{n} \right\rangle \cdot d\theta, \tag{6}$$

где J_{out} – поток из точки (x, y, z) в направлении вектора $\vec{\omega}$, а J_{in} – поток входящего в точку (x, y, z) излучения из направления вектора $\vec{\theta}$. В данном уравнении интеграл взят по Ω – совокупности входящих направлений в точку $M(x, y, z)$, θ – телесный угол, порожденный вектором $\vec{\theta}$. Способ отражения от поверхности, а также соотношение отраженной энергии и поглощенной задано в общем случае двулучевой функцией отражения $f(x, y, z, \vec{\omega}, \vec{\theta})$.



Уравнение (6) является частным случаем более общего уравнения, называемого уравнением рендеринга [12]. Существует несколько подходов к его решению. Рассмотрим их.

Обзор методов решения уравнения рендеринга

Алгоритм излучательности (radiosity)

Рассмотрим частный случай уравнения рендеринга для диффузных материалов. Двулучевая функция отражения в этом случае представляет собой константу и уравнение сводится к тривиальному.

$$J_{out}(x,y,z,\vec{\omega}) = J_{emit}(x,y,z,\vec{\omega}) + \int\limits_{\Omega} J_{in}(x,y,z,\vec{\theta}) \cdot f(x,y,z,\vec{\omega},\vec{\theta}) \cdot \left\langle -\vec{\theta} \cdot \vec{n} \right\rangle \cdot d\theta, \quad (7)$$

где J_{emit} – излученная мощность точкой. Как мы уже упоминали, для диффузных материалов, значения ДФО не зависят ни от входящих, ни от исходящих направлений

$$f(x,y,z,\vec{\omega},\vec{\theta}) = \frac{\eta(x,y,z)}{\pi} \quad (8)$$

$$\pi \cdot J_{out} = \pi \cdot J_{emit} + \eta \cdot \int_{\Omega} J_{in}(x,y,z,\vec{\theta}) \cdot \left\langle -\vec{\theta} \cdot \vec{n} \right\rangle \cdot d\theta \quad (9)$$

Далее, преобразуя интеграл по телесному углу в интеграл по всем точкам поверхности сцены, а также разбивая поверхность на непересекающиеся полигоны с постоянной излучательностью, в конечном итоге, получаем следующую систему линейных уравнений

$$B_i = E_i + \eta_i \cdot \sum_j F_{ij} B_j, \quad (10)$$

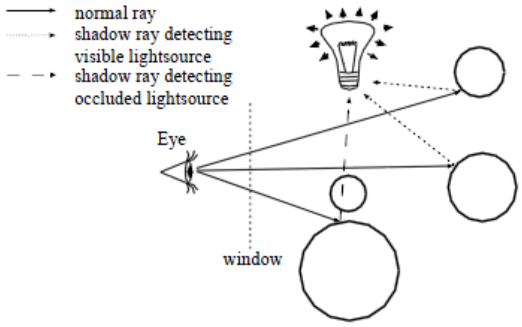
где B_i – результирующая излучательность полигона, E_i – излучаемая энергия полигоном, η_i – коэффициент отражения, F_{ij} – форм-фактор.

Подробнее с преобразованиями и самой реализации метода можно ознакомиться здесь [13].

Минус данного метода в том, что он позволяет моделировать, в принципе, конкретный тип материала – диффузный. В реальной жизни, как уже упоминалось, это встречается не всегда.

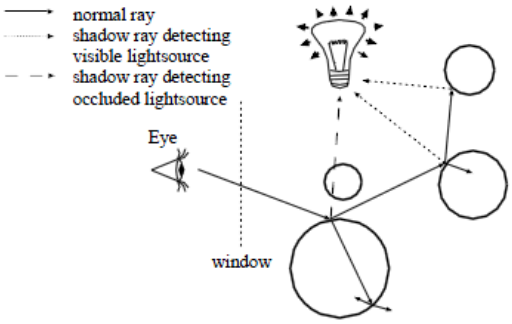
Алгоритм «бросания лучей» ray-casting

Фактически, данный метод учитывает только первично-освещенные точки поверхности, а значит он также помогает произвести расчет теней. Из точки выпускается луч по направлению к источнику. Если этот луч имеет пересечения с каким-либо фрагментом поверхности, то значит, что точка, из которой был выпущен луч, находится в тени. В ином случае, рассчитываем долю энергии получаемую точкой [13].



Алгоритм visibility ray-tracing и трассировка фотонов

Предыдущий алгоритм учитывал только первично-освещенные точки. Повысив дальность прохода лучей, то есть увеличив количество отражений и преломлений получим более точную модель. Различия visibility ray-tracing и трассировки фотонов лишь в направлении (от источника или же к источнику).



Метод Монте-Карло (на базе трассировки лучей) и стохастические методы

Все пространство разбивается равномерной сеткой. Количество узлов в трехмерном пространстве при этом равно N^3. Узлы, по которым ведется интегрирование, выбираются случайно. Соответственно, при увеличении числа узлов интеграл, вычисленный методом Монте-Карло [13], приближается к точному значению. Но при недостаточном количестве узлов получаем шумы. Шумы убираются размытием.

Гибридные алгоритмы

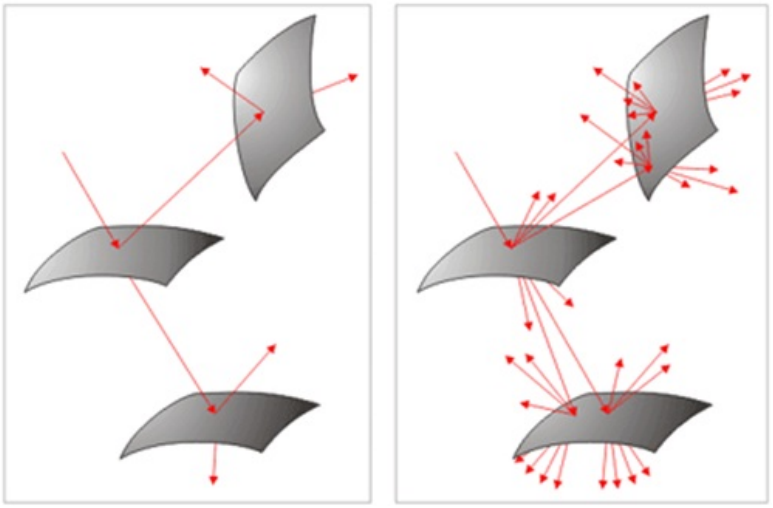
Каждый из приведенных выше алгоритмов обладает своими сильными и слабыми сторонами. Так алгоритм излучательности позволяет быстро рассчитать освещенность, но только для диффузных поверхностей. С другой стороны, алгоритм трассировки лучей имеет в своих недостатках низкую скорость, но является более универсальным. На этой основе появляется гибридные многопроходные алгоритмы,

в которых расчет делится на несколько независимых этапов. Так, использование только алгоритма излучательности позволяет рассчитывать взаимодействие и взаимовлияние диффузных поверхностей в пространстве, трассировка же учитывает зеркальную составляющую. Характерным примером данной группы алгоритмов являются алгоритмы Сллиона, Ширли, Чена [13].

Метод распределенной трассировки лучей

Рассмотренные выше подходы в рамках трассировки лучей можно использовать для расчета сложных материалов с неоднозанчным поведением. Пуская лучи в нескольких разных направлениях получаем возможность моделировать поверхности с произвольной функцией ДФО [14].

Данный подход увеличивает количество испускаемых лучей, а значит вычислительная сложность возрастает по сравнению с вышеприведенными алгоритмами обычной трассировки. Такова цена за универсальность двулучевой функции отражения.



Основу этого алгоритма мы и изберем для реализации нашей программы

Реализация

Составим некоторый алгоритм решения нашей задачи. Напомним, что никаких оптимизаций в алгоритме не проводилось, то есть расчеты велись «влоб». Так как это никоим образом не помешает анализу и сравнению производительности процессора и графического ускорителя, а также даст некоторую верхнюю практическую оценку вычисления задачи.

Пересечение отрезка и многоугольника

Задача определения пересечения отрезка и многоугольника разбивается на следующие этапы:

1. Определить точку пересечения M плоскости многоугольника и прямой, содержащей отрезок.
2. Находится ли точка пересечения M внутри многоугольника.
3. Находится ли точка пересечения M внутри отрезка.

Описанные выше задачи решаются стандартными методами аналитической геометрии [15]

Первичные грани

1. Берем полигон модели α.
2. Определяем её геометрический центр – точку P.
3. Для всех граней модели проверяем, пересекает ли её отрезок PL (кроме исходного полигона α, центром которого является P), где L – источник электромагнитного излучения.
4. В случае отсутствия таких граней, помечаем исходный полигон α как первичный и рассчитываем какую энергию в секунду получает наша поверхность

$$P_{in} = \frac{P_0 S cos\phi}{4\pi R^2}, \tag{11}$$

где P_0 – мощность исходного источника излучения L, ϕ – угол между вектором $\vec{LP}$ и нормалью $\vec{n}$, направленной в сторону L, расстояние $R = PL$, а S – площадь полигона.

5. Повторяем шаги 1-4 для всех полигонов поверхности.

Параллелизация на графическом ускорителе выполнялась на всех уровнях: как на внешнем цикле 1-4, так и на внутреннем 3. Причем внутренний цикл распределялся между нитями одного блока, а внешний – между блоками. При выполнении алгоритма на ускорителе модель космического корабля хранилась в глобального типа памяти.

Вторичные грани

1. Проведем расчеты первично-освещенных граней.
2. Для каждой первичной грани α выполним следующие шаги:

3. Определим центр первично-освещенной грани P.

4. Для каждого полигона модели β с геометрическим центром V, если не существует элементов моделей, пересекающих отрезок PV, будем считать его вторично-освещенным (аналогично как при определении первичных).

5. Мощность, получаемая вторично-освещенным полигоном от первично-освещенного фрагмента в этом случае рассчитываем по формуле

$$\Delta P'_{in} = \frac{P_{in} f S' cos\phi'}{2\pi R'^2}, \tag{12}$$

где P_{in} – получаемая мощность первичной грани от источника излучения, f – двулучевая функция отражения ($P_{in} \cdot f = P_{out}$ в направлении проверяемой грани), S' – площадь полигона β, расстояние $R' = PV$, угол ϕ' между вектором $\vec{PV}$ и нормалью $\vec{n}'$ к полигону β, направленной в сторону точки P.

Выполнив весь внешний цикл 2-5 получим

$$P'_{in} = \sum_j \Delta P'_{in_j}, \tag{13}$$

то есть влияние всех первичных граней сложится.

Параллелизация выполнялась на всех уровнях: на внешнем цикле 2-5, на среднем 4 и на внутреннем 4. При реализации на CUDA внутренний цикл делился между нитями одного блока. Остальные циклы (средний и внешний) делились между блоками упорядоченными в двумерную сетку. Аналогично, как и в предыдущем алгоритме, модель при выполении программы на видеокарте, хранилась в глобальной памяти.

Если грань по итогом работы двух процедур не относилась ни к первичной, ни к вторичной, то она помечалась особым цветом.

На основе рассчитанного освещения граней строилась сфера вокруг модели – сфера освещения. Расчет получаемой мощности полигонами сферы проводился аналогичными указанными выше способами.

Также, программа строит несколько проекций сфер на плоскость и получает изображение.

Результаты

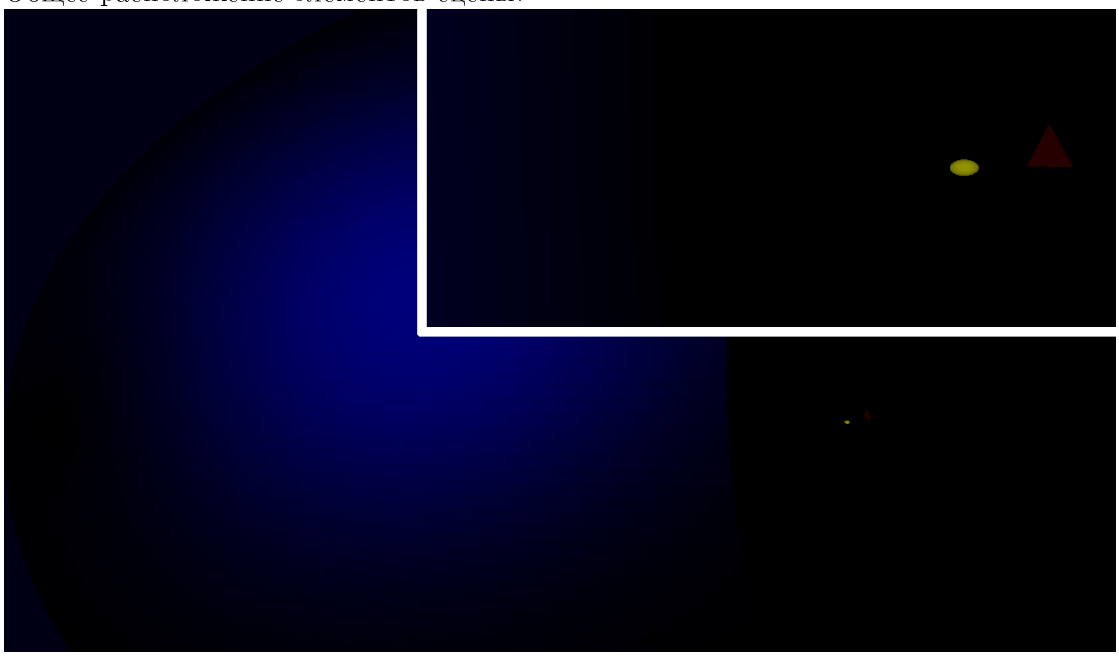
Результатом работы стало программное обеспечение, написанное на языке программирования C, работающее на операционных системах Linux и Windows. Основные возможности программы:

1. Загружать модель из файла (Wavefront OBJ geometry format, .obj) и отображать ее в трехмерном пространстве с использованием технологии OpenGL.
2. Возможность ставить источник излучения в любую точку пространства.
3. Проводить расчеты первичных и вторично-освещенных граней как на GPU (с использованием технологии CUDA), так и на CPU.
4. Сохранять проведенные расчеты в файл (.obj). Дополнительная информация хранится в комментариях особого вида, что, фактически, не повреждает сам файл.
5. Проводить расчет сферы освещения.
6. Возможность делать проекции сферы освещения с сохранением в файл картинки (Bitmap picture, .bmp).

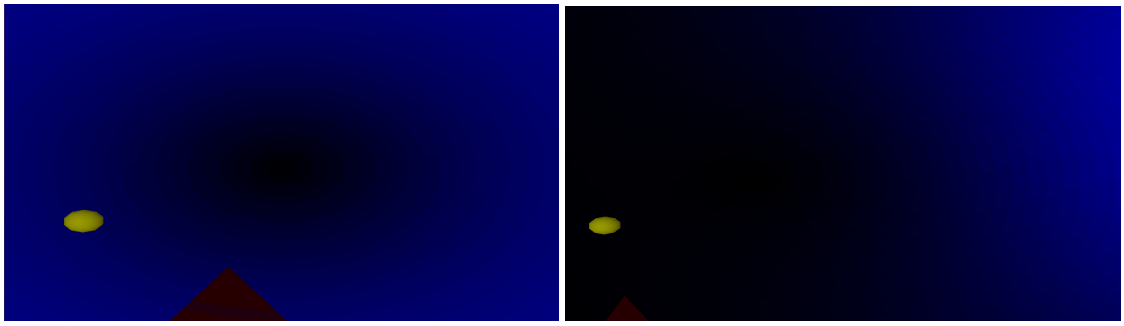
Диффузное и зеркальное

Как мы уже упомянули, в программе есть возможность задавать функцию ДФО вручную. Рассмотрим работу программы на простом примере: один треугольный полигон (красный) и один точечный источник излучения (желтая сфера).

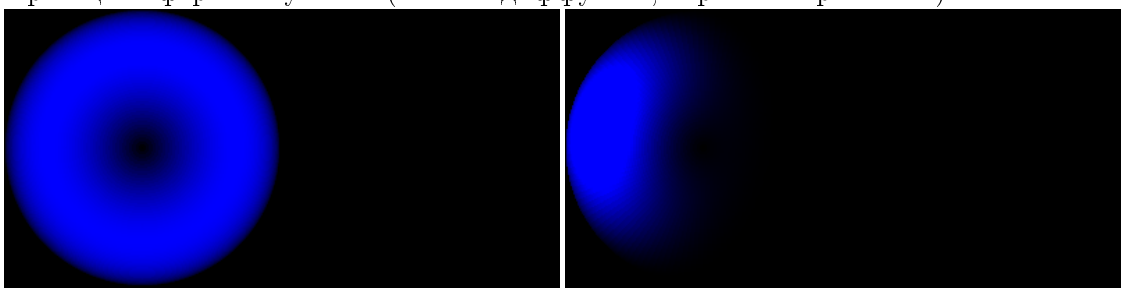
Общее расположение элементов сцены:



Скриншоты программы для диффузного рассеивания по Ламберту (слева) и зеркального отражения (справа):



Проекции сферы излучения (слева – диффузное, справа – зеркальное):

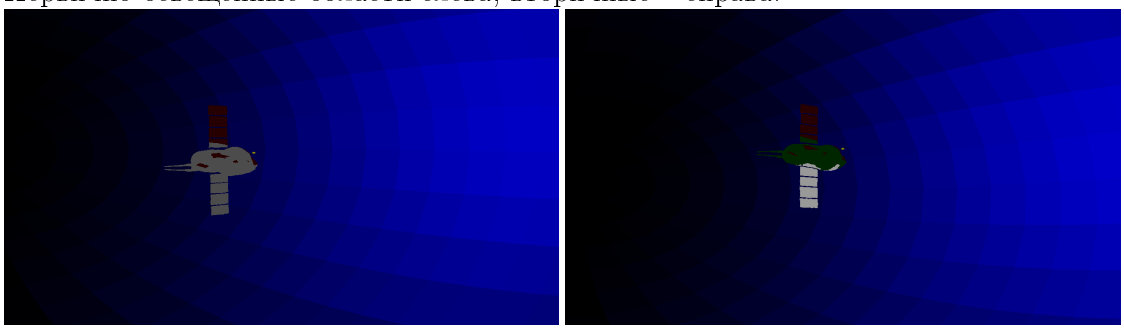


Таким образом, у нас есть возможность достаточно гибко моделировать различные материалы.

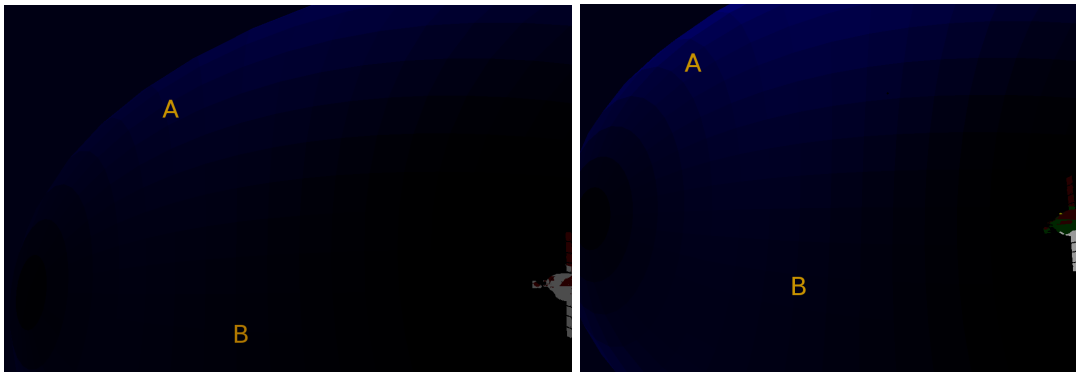
Первичные и вторичные

Рассмотрим одно положение объектов и сравним разницу между картинами в первом случае создаваемой только первично-освещенными областями и во втором – совокупностью первичных и вторичных групп полигонов.

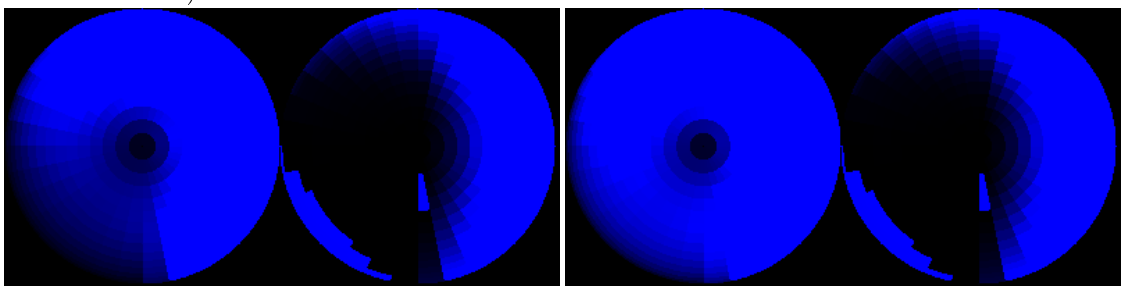
Первично-освещенные области слева, вторичные – справа:



Областью A обозначено отражение от верхней антенной части корабля. Область B – пространство, излучение на карте проекций которого изменилось.



Проекции сферы излучения (слева – только первичные, справа первичные и вторичные области):



По картам заметно, левая нижняя область сферы (область B) стала светлее. Таким образом, в этой области существенный вклад вносит вторичное отражение.

Таким образом, можно анализировать проекции сферы и выявлять области, в которых

1. излучение изменяется под влиянием вторичных зон (либо присутствует только переотраженный сигнал, либо присутствуют оба сигнала, но значения второго оказываются существенными),

2. излучение не подвергается изменениям (значение вторичного сигнала не существенно).

Сравнение производительности

Проведем сравнение производительностей по моделированию поставленных нами задач. Помимо параллельной версии программы, запускаемой на графическом ускорителе, была написана последовательная версия программы, запускаемая на центральном процессоре.

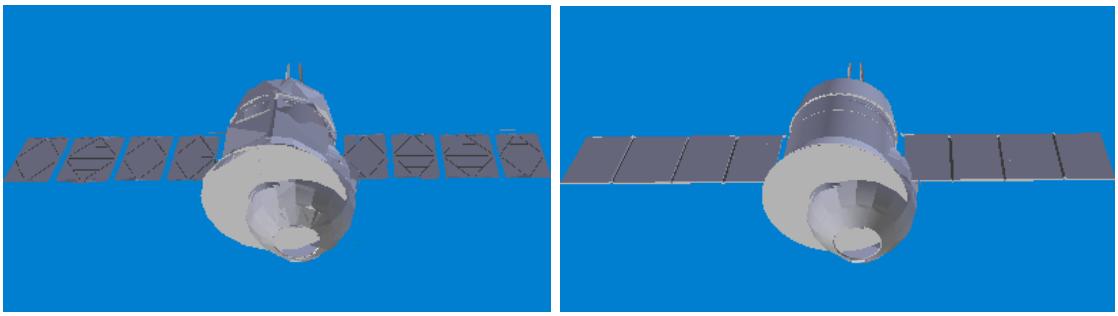
Характеристики центрального процессора (CPU): модель Intel Core i5 2430M, четыре ядра с тактовой частотой 2400 MHz.

Несмотря на то, что процессор многоядерный, программа была написана последовательно, то есть во время исполнении занято было только одно ядро.

Характеристики графического ускорителя (GPU): чипсет Nvidia GeForse GT 540M (Compute Capability 2.1).

Сравнение проводилось на пяти разной степени детализированности моделях космического корабля. Задача на GPU запускалась на сетке 65535 блоков x 512 нитей.

Подробности распределения вычислений между блоками и нитями, а также особенности использования типов памяти графического ускорителя рассматривались в разделе **Реализация** этой работы.



Расчет первичных зон отражения

время, сек/полигонов	**7391**	**12466**	**58503**	**109215**	**152027**
CPU	13.01	55.11	1159.20	3954.45	7332.93
GPU	1.06	3.65	70.91	210.12	361.64
ускорение	12.26	15.07	16.35	18.81	20.28

Таким образом, видно, что чем крупнее задача, тем сильнее получаем ускорение.

Выводы

По результатам работы были достигнуты следующие цели:

1. написано программное обеспечение, помогающее в анализировании областей влияния зон первичного и вторичного отражения;

2. написанный параллельный вариант программы, позволяет рассчитывать сцену и создавать карты проекции за время, значительно меньшее, чем последовательный.

Однако, расчет влияние зон вторичного отражения является трудоемким процессом, поэтому одним из путей улучшения решения задачи, приводимой в данной работе, является использование оптимальный алгоритмов, например, использование регулярной сетки. Все трехмерное пространство разбивается сеткой, а пересечение ищется только по тем кубикам, через которые прошел луч. Это совершается, например, алгоритмами Брезенхема [16] или 3DDA-обходом (алгоритм Fujimoto [17]).

CUDA – достаточно сложный, но при этом гибкий инструмент. Добиться более высокой производительности приложения представляется возможным использованием многих низкоуровневых оптимизаций. Несмотря, на известные проблемы острой нехватки регистров графического процессора при трассировки лучей [3] определенное ускорение путем оптимизации типа использованной памяти получить можно.

Список литературы

[1] *С.Б. Медведев, В.В. Сазонов, Х.У. Сайгираев*, Моделирование зон неустойчивой работы радиотехнической измерительной системы с активным ответом во время сближения и стыковки космических кораблей с Международной Космической Станцией // Математическое моделирование, 24(2):151–160, 2012. [html]

[2] Легедарный корабль «Союз» // Новости Космонавтики, апрель 2002

[3] *А.В. Боресков, А.А. Харламов*, Основы работы с технологией CUDA // ДМК Пресс, 2010

[4] *С.Б. Березин, В.М. Пасконов, Н.А. Сахарных*, Моделирование трехмерных течений методом расщепления с использованием параллельной архитектуры ГПУ // Вычислительные методы и программирование, 13: 75-81, 2012. [pdf]

[5] *М.А. Курако*, Оптимизация производительности вычислений для моделирования цунами на параллельных архитекутурах // Молодёжь и наука: Сборник материалов VII Всероссийской научно-технической конференции студентов, аспирантов и молодых учёных, посвященной 50-летию первого полета человека в космос [pdf]

[6] *В.В. Парубец, О.Г. Берестнева, Д.В. Девятых*, Применение технологии CUDA для ускорения вычислений в нейронных сетях // Известия Томского политехнического университета, том 320, выпуск 5 (2012)

[7] *П.В. Маковецкий, В.Г. Васильев*, Отражение радиолокационных сигналов – лекции // Ленинградский институт авиационного приборостроения, 1975

[8] Принцип Гюйгенса-Френеля // Большая советская энциклопедия (3-е издание)

[9] *Schuster A.*, An Introduction to the Theory of Optics. //London: Edward Arnold (1904), 340 p.

[10] *Hecht E.*, Optics. //Addison Wesley, 1987. 457 p.

[11] *Frank Pedrotti, Leno Pedrotti*, Introduction to Optics. // Prentice Hall, 1993. ISBN 0135015456.

[12] *J. Kajiya*, The rendering equation // ACM SIGGRAPH Computer Graphics, 1986. 20. N 4. P. 143-150

[13] *Андрей Лебедев*, История развития алгоритмов глобального освещения // Компьютерная графика и мультимедиа, 2011 [html]

[14] *Cook R., Porter T., Carpenter L.*, Distributed ray tracing // SIGGRAPH Comput. Graph, 1984. 18. N 3. P. 137-145.

[15] *Moller T., Trumbore B.*, Fast, Minimum Storage Ray-Triangle Intersection // J. Graphics Tools, 1997, v.2(1), p.21 – 28.

[16] *Роджерс Д.*, Алгоритмические основы машинной графики //Мир, 1989. — С. 512.

[17] , Анализ алгоритмов трассировки лучей для реалистичной визуализации трехмерных сцен и способов уменьшения их вычислительной сложности.

И.А. Запорожченко, М.А. Григорьев, С.А. Зори, //Цифровая обработка сигналов и изображений. – с. 353.