|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Жигалкин Дмитрий Романович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_ИУ7-45Б\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стационарная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия \_\_\_\_\_\_МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Жигалкин Д.Р.\_**\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Куров А.В.**\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2020 г.*

**Индивидуальное задание:**

Разработать программное обеспечение для создания реалистичного изображения методом обратной трассировки лучей. Изображение представляет собой сцену, на которую можно установить только заданные в программе объекты: сферу, конус, плоскость, а также выбрать их характеристики: цвет, зеркальные отражения (по шкале), отражающую способность (можно выбрать предел рекурсии) (по шкале), прозрачность (по шкале). Также можно выбрать заданные в программе источники света: направленный, точечный и окружающее освещение и выбрать их интенсивность. В программе можно менять положение объектов, источников света (кроме окружающего освещения). Камеру наблюдателя можно перемещать на заданные координаты и поворачивать на заданный угол вокруг заданной оси.

Возможности сохранить полученное изображение – нет. Окно программы имеет фиксированный размер.

Оглавление

[Введение 4](#_Toc26537825)

[1. Аналитическая часть 5](#_Toc26537826)

[1.1 Обзор и анализ существующих программных систем и обоснование необходимости разработки 5](#_Toc26537827)

[1.2 Формализация объектов синтезируемой сцены 7](#_Toc26537827)

[1.3 Обоснование использованных алгоритмов 8](#_Toc26537828)

[1.4 Описание трехмерных преобразований сцены 9](#_Toc26537837)

[1.5 Выводы из аналитического раздела 10](#_Toc26537838)

[2. Конструкторская часть 11](#_Toc26537839)

[2.1 Описание алгоритма трассировки лучей 11](#_Toc26537827)

[2.2 Уравнение лучей 12](#_Toc26537827)

[2.3 Пересечение луча и сферы 13](#_Toc26537827)

[2.4 Освещение 15](#_Toc26537827)

[2.5 Моделирование диффузного отражения 16](#_Toc26537827)

[2.6 Уравнение диффузного отражения 18](#_Toc26537827)

[2.7 Тени 19](#_Toc26537827)

[2.8 Зеркальное отражение 22](#_Toc26537827)

[2.9 Моделирование зеркального отражения 23](#_Toc26537827)

# Введение

В компьютерной графике на сегодняшний день большое внимание уделяется алгоритмам получения реалистических изображений. Эти алгоритмы являются самыми затратными по времени. Обусловлено это тем, что они должны предусматривать множество физических явлений, таких как преломление, отражение, рассеивание света. Профессиональные программы для кинематографа учитывают еще больше явлений (дифракцию, интерференцию, зависимость коэффициентов преломления, отражения, поглощения от длины волны падающего света, вторичное, третичное отражение света).

В моей курсовой работе для рендеринга применяется алгоритм обратной трассировки. На сегодняшний день он считается одним из лучших для формирования реалистических изображений. Его используют большинство трехмерных графических редакторов. Применяется так же алгоритм z-буфера. Подобным методом пользуются в программах, где крайне важна скорость.

Цель данной работы – моделирование реалистичной сцены, расположенной за прозрачной поверхностью.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена, и дать описание выбранных свойств;

2) выбор и/или модифицирование существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену;

3) реализация данных алгоритмов для создания трехмерной сцены;

4) разработать программное обеспечение, которое позволит отобразить трехмерную сцену и визуализировать оптические эффекты.

# Аналитическая часть

* 1. **Обзор и анализ существующих программных систем и обоснование необходимости разработки**

На данный момент существует множество программных систем с открытым исходным кодом для работы над трассировкой лучей, но я выделил для себя основные:

* BRL-CAD
* Blender

BRL-CAD–представляет собой мощную 3Dсистему автоматизированного проектирования составных объемных тел методом конструктивной блочной геометрии. Включает в себя интерактивный геометрический редактор, параллельную трассировку лучей, рендеринг и геометрический анализ.

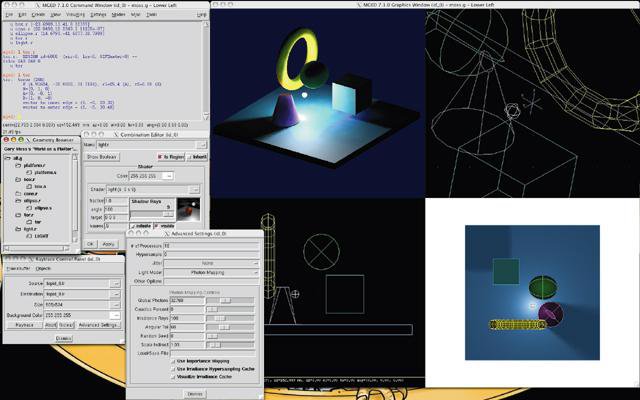


Рисунок 1.1 Интерфейс программы BRL-CAD.

Недостатки: большой объем программы, непростой интерфейс для неподготовленного пользователя.

Преимущества: кроссплатформенность, регулярные обновления, большие возможности для подготовленного пользователя.

Blender – профессиональное свободное программное обеспечение для создания трехмерной графики, включающее в себя множество возможностей, в том числе и рендеринг.

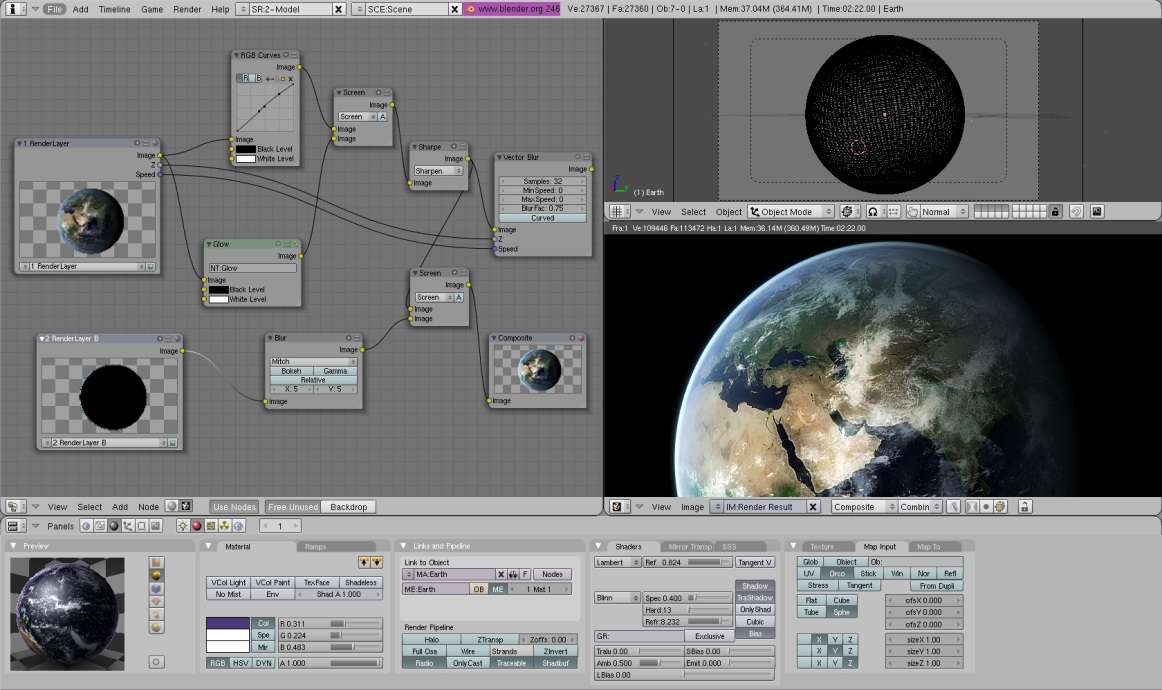


Рисунок 1.2 Интерфейс программы Blender.

Недостатки: Blender имеет репутацию программы, сложной для изучения. Практически каждая функция имеет соответствующее ей сочетание клавиш. Высокие системные требования для комфортной работы с приложением.

Преимущества: характерной особенностью является небольшой размер. Поддержка разнообразных геометрических примитивов, включая полигональные модели, универсальные встроенные механизмы рендеринга.

Вывод из проведенного анализа: не смотря на то, что существует немало многофункционального программного обеспечения для работы с трассировкой лучей, характерным для него недостатком является сложный и не совсем понятный для неподготовленного пользователя интерфейс. В своем программном продукте я хочу устранить данный недостаток.

## 1.2 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов.

* Источников света – представляют собой 3 подобъекта, которые расположены на бесконечности:

1. Окружающее освещение. Описывается коэффициентом освещенности.
2. Точечный источник света. Описывается тремя координатами положения и коэффициентом освещенности.
3. Направленный источник света. Описывается вектором направления и коэффициентом освещенности.

* Плоскость земли – ограничивающая плоскость. Изначально расположена внизу экрана, параллельна oXZ.
* Объемные тела, каждое из которых характеризуется следующими основными параметрами: координаты центра, цвет, коэффициент диффузного отражения, коэффициент зеркального отражения, прозрачность.

Они представляют собой 2 подобъекта:

1. Тела вращения:

Сфера. Помимо основных параметров задается еще и радиус.

1. Тела, заданные гранями:

Конус. Помимо основных параметров задается высота.

* Луч – невидимый объект, который задается параметрическим уравнением P= O + t \* direction. Где O – координаты точки начала луча, t - параметр, изменяя который можно получить любую точку на луче, direction – вектор направление луча.

## 1.3 Обоснование использованных алгоритмов

Главным алгоритмом, используемым в программе для удаления невидимых граней, а так же моделирования эффекта отражения, является алгоритм обратной трассировки лучей. Выбор был сделан в его пользу, так как в сравнении с его аналогом – алгоритмом трассировки лучей он оказался быстрее за счет того, что обрабатываются только те пиксели, на которые направлена камера.

Этот алгоритм позволил мне добиться максимальной реалистичности изображения. Он позволяет смоделировать распространение света в пространстве, учитывая законы геометрической оптики. Алгоритм можно быстро модернизировать, добавив в него обработку новых световых явлений. Алгоритм позволяет строить качественные тени с учетом большого числа источников света. Другие алгоритмы не позволяют так просто и качественно строить тени.

Достоинством алгоритма является то, что он не требователен к памяти, в отличие от алгоритма z-буфера. А недостатком является то, что работает он сравнительно долго и не позволяет строить изображения в реальном времени.

Для визуализации диффузного отражения был использован закон Ламберта, который понятен и удобен при трассировке лучей.

Моделируя зеркальное отражение, я сделал выбор в пользу модели распределения Фонга. Хотя она и не основана на физических явлениях и является произвольной, она проста в вычислениях и хорошо выглядит.

## 1.4 Описание трехмерных преобразований сцены

Сдвиг точки:



Масштабирование относительно начала координат:



Любое вращение в трёхмерном пространстве может быть представлено как композиция поворотов вокруг трёх ортогональных осей (например, вокруг осей декартовых координат). Этой композиции соответствует матрица, равная произведению соответствующих трёх матриц поворота.

Матрицами вращения вокруг оси декартовой системы координат на угол {\displaystyle \alpha }α в трёхмерном пространстве с неподвижной системой координат являются:

* Вращение вокруг оси х:
* Вращение вокруг оси у:
* Вращение вокруг оси z:

## 1.5 Выводы из аналитического раздела

В данном разделе был рассмотрен алгоритм обратной трассировки лучей для удаления невидимых граней. Были изучены его преимущества и недостатки, проведено сравнение с алгоритмом Z-буфера. Также были определены и формализованы объекты сцены, преобразования сцены с объектами.

# Конструкторская часть

* 1. **Описание алгоритма трассировки лучей.**

Методы трассировки лучей на сегодняшний день считаются наиболее мощными методами создания реалистических изображений. Универсальность методов трассировки в значительной степени обусловлена тем, что в их основе лежат простые и ясные понятия, отражающие наш опыт восприятия окружающего мира.

Рассмотрим, как формируется изображение. Изображение получается из-за того, что свет попадает в камеру. Выпустим из источников света множество лучей. Назовем их первичными лучами. Часть этих лучей улетит в свободное пространство, а часть попадет на объекты. На них лучи могут преломиться, отразится. При этом часть энергии луча поглотится. Преломленные и отраженные лучи образуют множество вторичных лучей. Далее эти лучи опять же преломятся и отразятся и образуют новое поколение лучей. В конечном итоге часть лучей попадет в камеру и сформирует изображение.

Существуют алгоритмы, работающие по такому алгоритму. Но они крайне неэффективны, так как большинство лучей, исходящих из источника, не попадают в камеру. А приемлемая картинка получается, если трассировать большое число лучей, что займет очень много времени. Данный алгоритм называется прямой трассировкой лучей.

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. Этот метод разработали в 80-х годах Уиттед и Кэй. В этом методе отслеживаются лучи не от источников, а из камеры. Таким образом, трассируется определенное число лучей, равное разрешению картинки.

Предположим, что у нас есть камера и экран, находящийся на расстоянии d от нее. Разобьем экран на квадратики. Дальше будем по очереди проводить лучи из камеры в центр каждого квадратика (первичные лучи). Найдем пересечение каждого такого луча с объектами сцены и выберем среди всех пересечений самое близкое к камере. Далее, применив нужную модель освещения, можно получить изображение сцены. Это самый простой метод трассировки лучей. Он позволяет лишь отсечь невидимые грани.

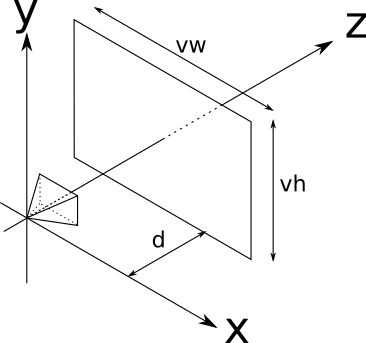


Рисунок 2.1 Расположение камеры и сцены

Но можно пойти дальше. Если мы хотим смоделировать такое явление, как отражение, нам необходимо из самого близкого пересечения пустить вторичные лучи. Например, если поверхность отражает свет и она идеально ровная, то необходимо отразить первичный луч от поверхности и пустить по этому направлению вторичный луч.   
 О прозрачности. Когда луч падает на прозрачную поверхность, мы пускаем вторичный луч и вычисляем дополнительный цвет — цвет света, проходящего сквозь объект.

* 1. **Уравнение лучей.**

Наилучшим способом представления лучей для нашей цели будет использование параметрического уравнения. Мы знаем, что луч проходит через O (точка, в которой находится камера), и мы знаем его направление (из O в V), поэтому мы можем выразить любую точку P луча как

где t — произвольное действительное число.  
Давайте обозначим , то есть направление луча, как  тогда уравнение примет простой вид

* 1. **Пересечение луча и сферы.**

Что такое сфера? Сфера — это множество точек, лежащих на постоянном расстоянии (называемом радиусом сферы) от фиксированной точки (называемой центром сферы):

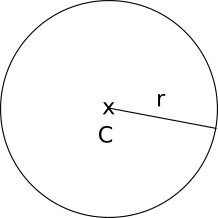


Рисунок 2.2 Изображение сферы.

Если C — центр сферы, а r — радиус сферы, то точки P на поверхности сферы удовлетворяют следующему уравнению:

Расстояние между P и C- это длина вектора из P в C:

Длина вектора — это квадратный корень его скалярного произведения на себя:

Или

Поскольку P — это одна и та же точка в обоих уравнениях, мы можем заменить P в первом на выражение для P во втором. Это даёт нам:

Это квадратное уравнение, тогда его решения

Это точно соответствует случаям, когда луч не пересекает сферу, луч касается сферы и луч входит и выходит из сферы:

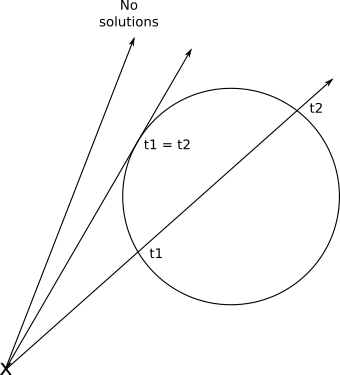


Рисунок 2.3 Возможные пересечения луча и сферы

* 1. **Освещение.**

Описание некоторых упрощающих допущений. Во-первых, мы объявим, что всё освещение имеет белый цвет. Это позволит нам охарактеризовать любой источник освещения единственным действительным числом i, называемым яркостью освещения.   
 Во-вторых, мы избавимся от атмосферы. Это значит, что освещение не становятся менее яркими, независимо от их дальности.

Источники освещения:

* Точечные источники.
* Направленные источники
* Окружающее освещение

Точечный источник (является аппроксимацией лампы накаливания) испускает свет из фиксированной точки в пространстве, называемой его позицией. Свет испускается равномерно во всех направлениях; именно поэтому его также называют всенаправленным освещением. Следовательно, точечный источник полностью характеризуется его позицией и яркостью.  
 Давайте зададим вектор  как направление из точки P в сцене к источнику освещения Q. Этот вектор, называемый световым вектором, просто равен . Заметьте, что поскольку Q фиксирована, а P может быть любой точкой сцены, то в общем случае  будет разным для каждой точки сцены.

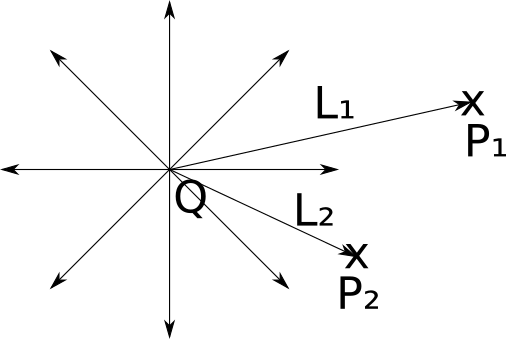


Рисунок 2.4 Точечный источник освещения

Для аппроксимации Солнца мы зададим направленные источники освещения. Как и точечные источники, направленный источник имеет яркость, но в отличие от них, у него нет позиции. Вместо неё у него есть направление. Можно воспринимать его как бесконечно удалённый точечный источник, светящий в определённом направлении. В случае точечных источников нам нужно вычислять новый световой вектор  для каждой точки P сцены, но в этом случае  уже задан.

Третий тип источников освещения, называется окружающим освещением, которое характеризуется только яркостью. Считается, что оно носит безусловный вклад освещения в каждую точку сцены. Это очень сильное упрощение чрезвычайно сложного взаимодействия между источниками освещения и поверхностями сцены, но оно передает правильную картину.

**2.5 Моделирование диффузного отражения.**

Когда луч света падает на матовый объект, то из-за неровности его поверхности на микроскопическом уровне, он отражает луч в сцену равномерно во всех направлениях, то есть получается «рассеянное» («диффузное») отражение.

С другой стороны, количество отражённого света зависит от угла между лучом света и поверхностью. Интуитивно это понятно— энергия, переносимая лучом, в зависимости от угла должна распределиться по меньшей или большей поверхности, то есть энергия на единицу площади, отражённая в сцену, будет соответственно выше или ниже:

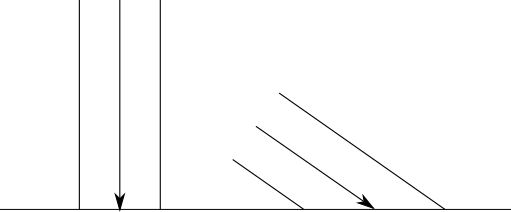


Рисунок 2.5 Падение светового луча

Итак, луч света с направлением и яркостью падает на поверхность с нормалью . Какая часть отражается обратно в сцену как функция от

Для геометрической аналогии давайте представим яркость света как «ширину» луча. Его энергия распределяется по поверхности размером . Когда  и  имеют одно направление, то есть луч перпендикулярен поверхности, , а это значит, что энергия, отражённая на единицу площади равна падающей энергии на единицу площади; . С другой стороны, когда угол между  и  приближается к 90, A приближается к ∞, то есть энергия на единицу площади приближается к 0; . Но что происходит в промежутках? Ситуация отображена на схеме ниже, Мы знаем , . Нам нужно вычислить .

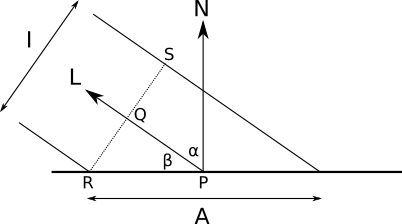


Рисунок 2.6 Падение луча на поверхность под углом

Поскольку технически луч света не имеет ширины, поэтому мы будем считать, что всё происходит на бесконечно малом плоском участке поверхности. Пусть , по определению она перпендикулярна , который также является направлением . Значит треугольник Очевидно, что угол. Рассмотрим треугольник : .

По определению

 и , то есть его можно вычислить как

Итого

Мы получили уравнение, связывающее отражённую часть света с углом между нормалью к поверхности и направлением света.

Заметьте, что при углах больше  значение  становится отрицательным. Если мы не задумываясь используем это значение, то в результате получим источники света, вычитающие свет. Это не имеет никакого физического смысла - угол больше  просто означает, что свет на самом деле достигает задней части поверхности, и не вносит свой вклад в освещение освещаемой точки. То есть если становится отрицательным, то мы считаем его равным 0.

**2.6 Уравнение диффузного отражения.**

Теперь мы можем сформулировать уравнение для вычисления полного количества света, полученного точкой с нормалью в сцене с окружающим освещением яркостью и точечных или направленных источников света с яркостью и световыми векторами или известными (для направленных источников), или вычисленными для P (для точечных):

Стоит снова повторить, что члены, в которых  не должны прибавляться к освещённости точки.

**2.7 Тени.**

Тени появляются там, где есть свет, но его лучи не могут достичь объекта, потому что на их пути есть другой объект.

Можно выделить 2 случая:

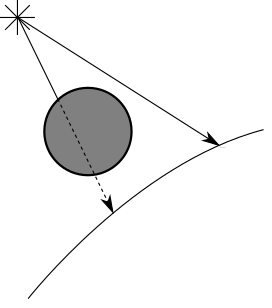


Рисунок 2.7 Возникновение тени.

Начнём с направленного источника. Мы знаем P. Это точка, которая нас интересует. Мы знаем  - это часть определения источника освещения. Имея , мы можем задать луч, а именно , который проходит из точки до бесконечно отдалённого источника освещения. Пересекает ли этот луч другой объект? Если нет, то между точкой и источником ничего нет, то есть мы можем вычислить освещённость от этого источника и прибавить его к общей освещённости. Если пересекает, то мы игнорируем этот источник.  
 Мы уже знаем, как вычислить ближайшее пересечение между лучом и сферой; мы используем его для трассировки лучей от камеры. Мы снова можем использовать его для вычисления ближайшего пересечения между лучом света и остальной сценой.

Однако параметры немного отличаются. Вместо того, чтобы начинаться с камеры, лучи испускаются из . Направление равно не а . И нас интересуют пересечения со всем после  на бесконечное расстояние. Это значит, что  и

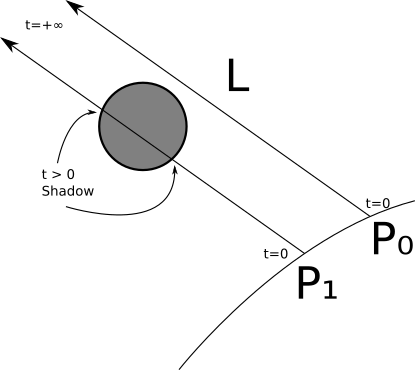


Рисунок 2.8 Испускание лучей.

Мы можем обрабатывать точечные источники очень похожим образом, но с двумя исключениями. Во-первых, не задан , но его очень просто вычислить из позиции источника и P. Во-вторых, нас интересуют любые пересечения, начиная с P, но только до L (в противном случае, объекты за источником освещения могли бы создавать тени!). То есть в этом случае  и

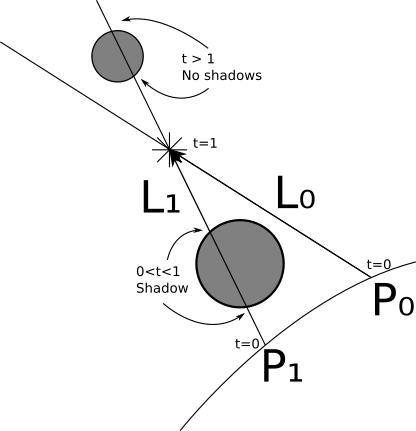
**

Рисунок 2.9 Ситуация с разными параметрами t.

Существует один пограничный случай, который нам нужно рассмотреть. Возьмём луч . Если мы будем искать пересечения, начиная с , то мы, вероятнее всего, найдём саму P при , потому что P действительно находится на сфере, и . Другими словами, каждый объект будет отбрасывать тени на самого себя.  
 Простейший способ справиться с этим — использовать в качестве нижней границы значений t вместо 0 малое значение ϵ. Геометрически, мы хотим сделать так, чтобы луч начинается немного вдали от поверхности, то есть рядом с P, но не точно в P. То есть для направленных источников интервал будет  а для точечных — .

**2.8 Зеркальное отражение.**

Для идеально отполированного зеркала падающий луч света  отражается в единственном направлении . Именно это позволяет нам чётко видеть объекты в зеркале: для каждого падающего луча  есть единственный отражённый луч . Но не каждый объект отполирован идеально, хотя большая часть света отражается в направлении , часть его отражается в направлениях, близких к ; чем ближе к , тем больше света отражается в этом направлении. «Блеск» объекта определяет то, насколько быстро отражённый свет уменьшается при отдалении от :

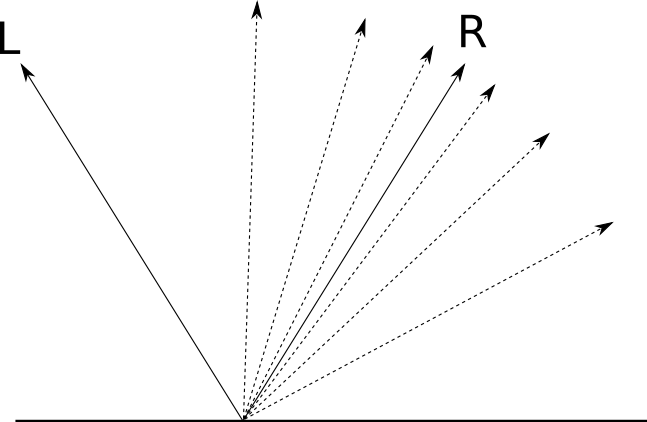


Рисунок 2.10 Отражение света.

Нас интересует то, как выяснить, какое количество света от  отражается обратно в направлении нашей точки обзора (потому что это свет, который мы используем для определения цвета каждой точки). Если  — это «вектор обзора», указывающий из  в камеру, а α — угол между  и , то вот, что мы имеем:

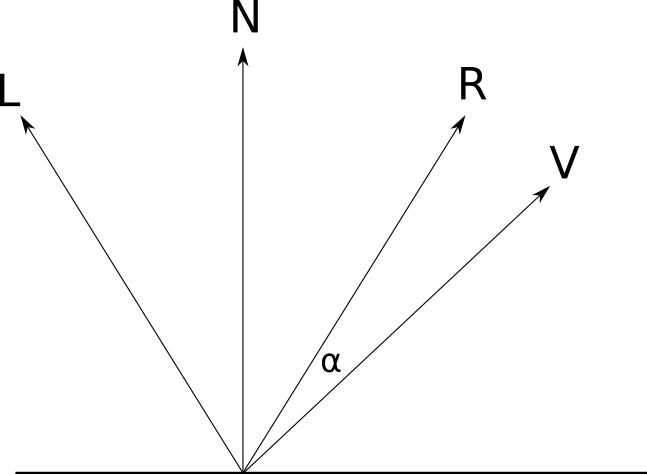


Рисунок 2.11 Зависимость угла падения.

При  отражается весь свет. При   свет не отражается. Как и в случае с диффузным отражением, нам нужно математическое выражение для определения того, что происходит при промежуточных значениях α.

**2.9 Моделирование зеркального отражения.**

Блеск — мера того, насколько быстро функция отражения уменьшается при увеличении . Очень простой способ получения различных кривых блеска заключается в вычислении степени  некоего положительного показателя s. Поскольку , то очевидно, что .То есть  ведёт себя точно так же, как , только «уже». Вот  для разных значений s:

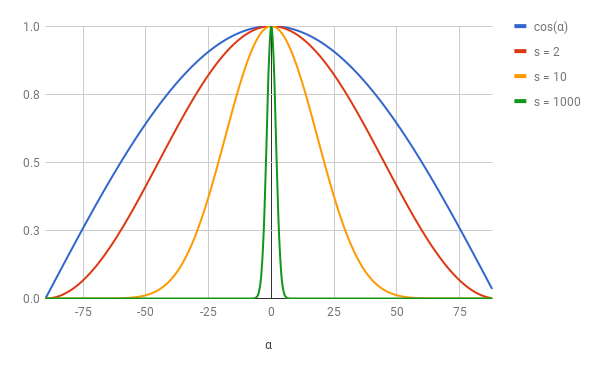


Рисунок 2.12 График .

S обычно называют показателем отражения, он является свойством поверхности. Объединим всё вместе. Луч  падает на поверхность в точке P, где нормаль равна  а показатель отражения — s. Какое количество света отразится в направлении обзора ?

Это значение равно , где α — это угол между  и , который в свою очередь является отражённым относительно . То есть первым шагом будет вычисление  из  и . Мы можем разложить . на два вектора . и , таких, что  где  параллелен , а  перпендикулярен :

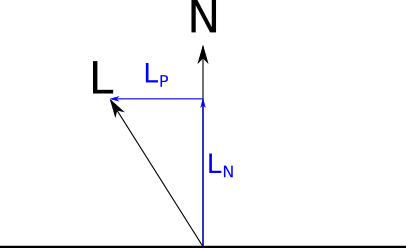


Рисунок 2.13 (а) Разложение вектора

 — это проекция  на ; по свойствам скалярного произведения и исходя из того, что , длина этой проекции равна ⟨, ⟩. Мы определили, что  будет параллелен , поэтому =⟨, ⟩. Поскольку , мы можем сразу получить   
Теперь посмотрим на ; поскольку он симметричен   относительно , его компонент, параллельный , тот же, что и у , а перпендикулярный компонент противоположен компоненту . То есть :

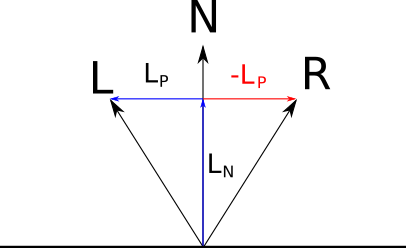


Рисунок 2.13 (б) Разложение вектора

Подставляя полученные ранее выражения, мы получим

и немного упростив, получаем

Теперь мы готовы записать уравнение «зеркального» отражения:

Как и в случае диффузного освещения, cos(α) может быть отрицательным, и мы снова должны это игнорировать. Кроме того, не каждый объект должен быть блестящим. Для таких объектов (который мы будем представлять через s = −1) значение «зеркальности» вообще не будет вычисляться.