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»

ОТЧЁТ О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Волков Георгий

Группа ИУ7-51Б

Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7

Студент: _____ Волков В.Г.
подпись, дата Фамилия, И.О.

Преподаватель: _____ Волкова Л.Л.
подпись, дата Фамилия, И. О.

Москва — 2022 г.

Идивидуальное задание:
Разработать программу для создания реалистических изображений с использованием алгоритма трссироваки лучей. Провести анализ существующих методов для построения реалистических изображений и выбрать самые подходящие.

Оглавление

Введение

Компьютерная графика — область деятельности, в которой вычислительные машины используются с целью создания, обработки и хранения графической информации.

В наши дни компьютерная графика применяется во всех областях жизни человека, что вызвано широким распространением ПК, мобильных телефонов и других устройств. Особенный интерес представляют алгоритмы построения реалистичных изображений в связи с ростом производительность процессоров, памяти и графических ускорителей. Эти алгоритмы способны учитывать множество физических ,таких как отражение, преломление, прозрачность, блеск и тень. Они являются крайне требовательными к ресурсам компьютера. Их скорость работы напрямую зависит от требований к качеству и реалистичности изображения, которое должно получиться в результате работы. Трудоемкость этих алгоритмов особенно проявляется при создании динамических сцен. Необходимо найти баланс между производительностью и реалистичностью получаемого изображения.

Целью работы является анализ и реализация алгоритма построения реалистичного изображения с применением трассировки лучей и алгоритмов для работы с камерой наблюдателя, моделью освещения и проекцией результата на экран.

Для достижения поставленной цели необходимо:

- провести анализ существующих алгоритмов построения реалистических изображений, текстурирования, моделей освещений, способов представления моделей и выбрать подходящие;
- реализовать выбранные алгоритмы и структуры данных;
- разработать ПО позволяющее отобразить результат;
- разработать интерфейс;
- провести исследование реализованного алгоритма.

Результатом работы будет написанное ПО для создания реалистических изображений с учётом цветов и текстур объектов и их оптических свойств, таких как отражение, преломление, прозрачность и блеск.

1 Аналитический раздел

В этом разделе будет представлен анализ существующих способов представления объектов, алгоритмов построения реалистического изображения, текстурирования и моделей освещения.

1.1 Описание трёхмерного объекта

В компьютерной графике существует множество способов представления объектов. В данной работе необходимо выделить такие способы, которые позволяет показать объём модели и наложить текстуру. В связи с этими требованиями можно рассмотреть следующие варианты:

- геометрические примитивы;
- воксельная модели;
- полигональные модели.

1.1.1 Геометрические примитивы

Примитив может быть описан некоторой функцией, принимающей параметры, например центр и радиус для сферы или ширину, высоту и глубину для параллелепипеда. В качестве таких примитивов могут выступать любые геометрические объекты: куб, конус, пирамида, сфера или цилиндр.

Достоинства:

- простота построения изображения модели;
- малое количество информации для хранения представления. объекта.

Недостатки:

- сложность создания реалистичной моделей из геометрических примитивов;
- трудность наложения текстур.

1.1.2 Воксельная модель

Двумерные модели можно опистать с помощью писелей. Аналогично можно и описывать трёхмерные модели из вокселей, маленьких кубиков. Они являются элементами объёмного изображения, содержащие значения элементов растра в трёхмерном виде.

Достоинства:

- хорошо подходит для моделирования непрерывных сред;
- хорошо подходит для трассировки.

Недостатки:

- большой расход памяти;
- низкое разрешение.

1.1.3 Полигональная модель

Полигональная модель использует полигональную сетку, которая является совокупностью связанных между собой выпуклых многоугольников (полигонов), аппроксимирующих поверхность модели. Представление модели в виде многоугольников упрощает их рендер. Чаще всего как полигон используются треугольник, так как он является простейшим многоугольником и все остальные многоугольники могут быть разбиты на треугольники.

Данным способом можно описать объекты любой формы с хорошим разрешением и детализацией. Детализация зависит от количества полигонов в сетке. Время рендера напрямую зависит от количества полигонов в модели.

Благодаря тому что полигоны это плоские многоугольники их легко использовать для имитации неровностей, изменяя их нормали, и текстурирования, заранее указав текстурные координаты.

Достоинства:

- широко используются в компьютерной графике;
- необходимо вычислять только координаты вершин при преобразованиях;
- небольшой объём данных при хорошей аппроксимации поверхности.

Недостатки:

- сложные алгоритмы визуализации;
- аппроксимация приводит к погрешностям.

1.1.4 Вывод

После анализа вышеописанных вариантов в качестве способа представления объектов сцены была выбрана полигональная модель. Так как в отличии от других вариантов с помощью полигональной сетки можно представить объекты любой формы в хорошем разрешении и большой детализацией, используя относительно небольшой объём памяти для хранения.

1.2 Алгоритм построения трёхмерного изображения

В этом разделе будут рассмотренны алгоритмы построяния трёхмерного изображения, которые будут использоваться в написанном ПО. Эти алгоритмы являются важнейшей частью всей программы и выполняют основную работу по рендеру результирубщего изображения.

1.2.1 Алгоритм с Z-буфером

Алгоритм работает в пространстве изображения и в нём используется 2 буфера: Z-буфер(буфер глубины) и буфер кадра, размер соответствуют количеству пикселей на экране. В Z-буфере находится информация о координате z для каждого пикселя, а в буфере кадра его интенсивность. В начале работы буферы заполняются минимальным значением координаты z и интенсивностью фона соответственно. Затем каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра, при этом не производится никакого начального упорядочения.

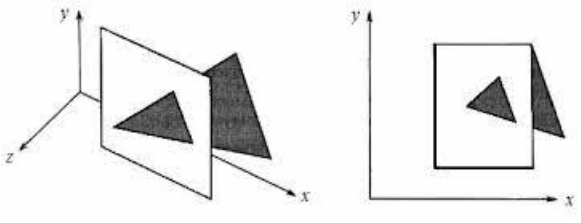

Рисунок 1.1. Пример работы алгоритма с использованием Z-буфера

В процессе работы глубина каждого нового пикселя сравнивается с глубиной, занесённой в буфер глубины. Если глубина нового пикселя меньше, то в буфер кадра заносится данные интенсивности нового пикселя, а в z-буфер новую координату z. Если же глибина нового пикселя больше, то данные в буферах не меняются.

Достоинствами данного алгоритма является простота его реализации и отсутствие сортировки элементов сцены. К недостаткам же можно отнести большой объём используемой памяти и трудоемкость реализации эффектов прозрачности и преломления, а также устранения ступенчатости.

1.2.2 Алгоритм с обратной трассировкой лучей

Является модификацией простого алгоритма трассировки лучей и отличается лишь тем что лучи испускается из камеры, а не из источников света.

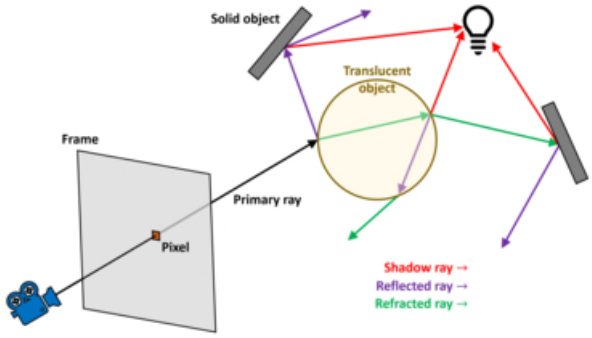

Рисунок 1.2. Пример работы алгоритма трассировки лучей

Сам алгоритм работает по следующему принципу: из камеры выпускается луч, который называется первичным, и ищется пересечения луча с каким-либо объектом сцены. Если пересечение не находится, то результатом является фоновая интенсивнось, иначе проверяется освещенность точки пересечения первичного луча и объекта сценаа всеми источниками света. Для это пускается теневой луч в направлении каждого источника света. Если он пересекает какой-либо объект, то источник света не видим из точки пересечения, то есть объект находится в тени относительного этого источника и соответственно он не освещает данную точку. Если же теневой луч не встретил никаких объектов на своём пути, то интенсивность рассчитывается в соответствии с моделью освещения. Результатом являет суммарная интенсивность от всех видимых источников света.

Если тела обладает отражающими свойства, то просчитывается и испускается отражённый луч, для которого рекурсивно выполняются те же операции как и для первично. Аналогично программа работает с преломлёнными лучами.

Достоинства данного алгоритма является высокий реализм получаемого изображения, учёт таких физических явлений как тень, преломление, отражение. Также алгоритм легко поддаётся распараллеливанию.

Основным недостатком является производительность. Каждый раз необходимо просчитывать множество новых лучей, что создаёт немалую нагрузку на вычислительные устройства.

1.2.3 Вывод

Для создания реалистичного изображения лучше всего подходит алгоритм трассировки лучей, так как он даёт наиболее приближенный к реальности результат и учитывает отражения, прозрачность и тени. Основным недостатком является его производительность. Алгоритм с использованием Z-буфера же можно использовать для предпросмотра сцены где может потребоваться рендер в реальном времени, которые трудно реализуем алгоритмом трассировкой лучей без соответствующего графического ускорителя.

1.3 Алгоритмы наложения текстур на трёхмерные объекты

1.3.1 Афинное текстурирование

Самый дешевый способ интерполяции между тремя текстурными координатами треугольника — использовать линейную интерполяцию с барицентрическими координатами. Введём координаты текстуры u, v. Они указвают на определённый пиксель текстуры, тексель. Для вершит полиговон изначльно заданы текстурные координаты, указываю на соответствующий тексель. Чтоб не высчитывать значения u и v для каждого пикселя проекции полигона, можно вопользоваться билинейной интерполяцией, используя уже известные координаты тексилей вершин треугольника. Основным недостатком этого метода является игнорирование координаты z, из-за чего текстура может искажатся и выглядить нереалистично.

1.3.2 Перспективно-корректное текстурирование

Искажения в афинном текстурировании вызваны допущем что u, v изменяются по экрану линейно. Это не так. Точные значения координат тексилей можно считать по точечным формулам, но это не эффективно. Проще воспользоваться фактом что u/z и v/z линейно зависят от координат проекции треугольника. Достаточно посчитать для каждой вершины $1/z$, u/z и v/z, а затем их линейно интерполировать по асему полигону. Сами же точные значения u и v считаюся как:

$$u = (u/z)/(1/z)$$
$$v = (v/z)/(1/z)$$

1.3.3 Вывод

Наиболее реалистичным методом является перспективно-корректное текстурирование, так как в его основе лежит точное тексутрирование, учитывающее перспективу в отличии от афинного метода.

1.4 Модель освещения трёхмерных объектов

1.4.1 Модель Ламберта

Простейшая модель освещения, чисто диффузное освещение. Считается, что свет падающий в точку, одинакового рассеивается по всем направлением полупространства. Таким образом, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. Считается по формуле:

$$I = k_d(\vec{n}, \vec{l}),$$

где

- k_d - коэффицент диффузного отражения;
- $\vec{n}$ - нормаль;
- $\vec{l}$ - еденичные вектор, напрвленный к источнику света.

1.4.2 Модель Фонга

Модель расчёта освещения трёхмерных объектов, в том числе полигональных моделей и примитивов, а также метод интерполяции освещения по всему объекту. Это локальная модель освещения, то есть она учитывает только свойства заданной точки и источников освещения, игнорируя эффекты рассеивания, линзирования, отражения от соседних тел. Эта модель состоит из диффузной составляющей и зеркальной. Благодаря зеркальной составляющей на объектах появляеюся блики. Интенсивность в точке зависит от того насколько близки отражённый вектор к вектору направленному из точки падения в сторону наблюдетеля. В модели учитывается интенсивность фонового, диффузного и зеркально освещёния. Для расчёта диффузной составляющей используется модель Ламберта. Считается по формуле:

$$I = k_a I_a + k_d(\vec{n}, \vec{l}) + k_s(\vec{n}, \vec{r})^n,$$

где

- k_d - коэффицент диффузного отражения;
- k_s - коэффицент зеракльного отражения;
- k_a - коэффицент рассеянного отражения;
- I_s - интенсивность фонового освещения;
- $\vec{n}$ - нормаль;
- $\vec{l}$ - еденичные вектор, напрвленный к источнику света;
- $\vec{l}$ - еденичные вектор, напрвленный к наблюдателю;
- n - степень, аппроксимирующая пространственное распределение зеркально отраженного света.

1.4.3 Модель Уиттеда

Модель освещенности Уиттеда является одной из самых распространенных и наиболее часто используемой моделью в методе трассировки лучей. Ипользует для расчёта интенсивности глобальноую модель освещения, учитывающую свет провзаимодействовавший с други объектами. Помимо учёта фоновой, диффузной и зеркальной компонент в этой модели ещё учитывается интернсивность отражённого и преломлённого света от других тел.

1.4.4 Вывод

Для получения наиболее реалистических изображений была выбрана модель Уиттеда как самая подходящая и реалистичная, так как она учитывает такие эффекты, как отражение, прозрачность, преломление, тень.

2 Конструкторский раздел

2.1 Реализация алгоритма обратной трассировки лучей

2.1.1 Алгоритм обратной трассировки лучей

Алгоритм обратной трассировки лучей работает следующим образом: из камеры пускается луч черерез каждый пиксель эрана и ищется его пересечения с объетами сцены. Луч выпущенный из камеры называется первичным. Пусть точка пересечения называется П1.

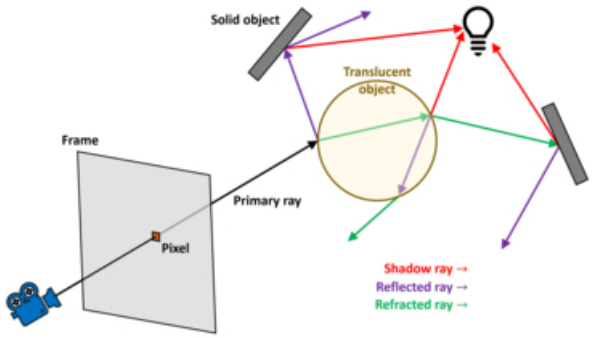

Рисунок 2.1. Схема работы алгоритма трассировки лучей

Далее для кажого источника света определяется, видна ли для него точка П1. Для это испускается теневой луч(красная стрелка на рисунке 2.1) из П1. Если он пересекается с какими-либо объектами, находящимися между П1 и источником света, то точка находится в тени от это источника и не освещается им. Далее освещени считается по некой модели. Результатом будет сумма интенсивностей всех видимых источников света для П1. Если материал имеет отражающие свойства то просчитывается и испускается отражённый луч света(фиолетовая стрелка на рисунке 2.1) и точно так же рекурсивно обрабатывается. Аналогично происходит с преломлёнными лучами(зелёная стрелка на рисунке 2.1), если материал имеет преломляющие свойства.

2.1.2 Нахождение пересечеия с полигоном

Самым известным тестом на пресечение луча и треугольника является бариоцентрический тест.

Введём следующие обозначения как на рисунке 2.3

- Z - точка пересечения;
- P - начало луча;
- t - расстояние от p до z;
- $\vec{d}$ - направление луча;
- V_0, V_1, V_2 - вершины треугольника;

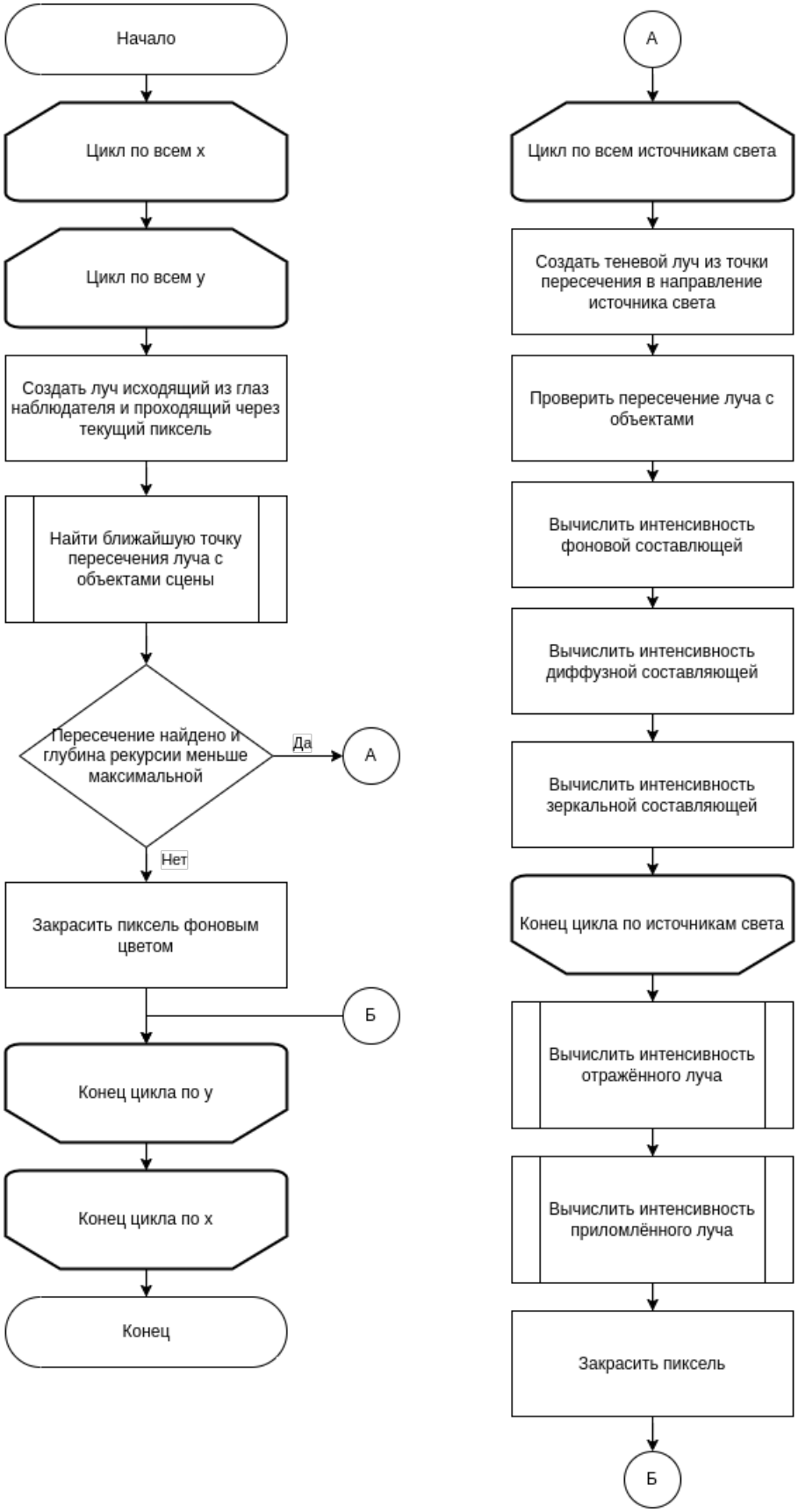

Рисунок 2.2. Блок-схема алгоритма трассировки лучей

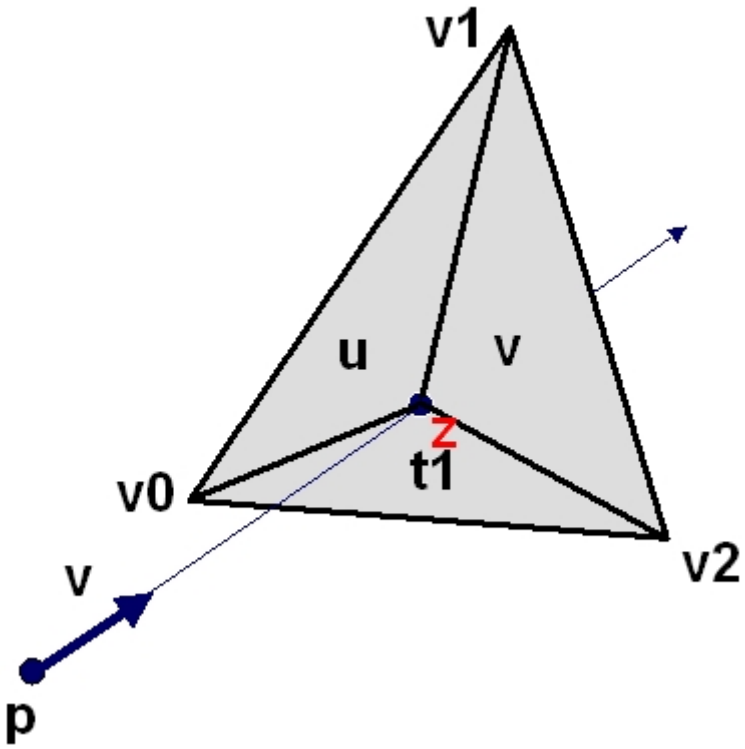

Рисунок 2.3. Схема для поиска пересечения луча и треугольника

- u, v, $t1$ - барицентрические координаты.

Барицентрические координаты представляют собой отношения площадей маленьких треугольников к большому треугольнику. Имея 3 точки на плоскости, можно выразить любую другую точку через ее барицентричечкие координаты.

$$Z(u,v) = (1-u-v) * V_1 + u * V_2 + v * V_0 \tag{2.1}$$

Уравнение (2.1) берется просто из определения барицентрических координат, выражая точку пересечения z.

$$Z(t) = P + t * \vec{d} \tag{2.2}$$

Уравнение (2.2) это параметрическое уравнение прямой.

$$P + t * \vec{d} = (1-u-v) * V_1 + u * V_2 + v * V_0 \tag{2.3}$$

Прировняв правые части уравнений (2.2) и (2.3) получаем третье уравнение, которое, по сути, является системой из 3-х уравнений с 3-мя неизвестными (u ,v, t).

Проведя алгебраические преобразования получим

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(\vec{V}, \vec{E_1})} * \begin{bmatrix} (\vec{Q}, \vec{E_2}) \\ (\vec{V}, \vec{T}) \\ (\vec{Q}, \vec{d}) \end{bmatrix}$$

где $\vec{E_1} = V_1 - V_0, \vec{E_2} = V_2 - V_0, \vec{T} = P - V_0, \vec{V} = (\vec{d} \times \vec{E_2}), \vec{Q} = (\vec{T} \times \vec{E_1})$.

2.1.3 Нахождение пересечеия с объемлющей оболочкой

При трассировке лучей крайне неэффективно искать пересечения с каждым полигоном объекта. Лучше поместить объект в объемлюющкю оболочку и сначала проверять пересечение с ней. Если луч не пересекает оболочку, то и не перескает объект сцены, находящийся в оболочке. В качестве такой оболочки будет использоваться сфера в связи спростотой поиска пересечения луча и сферы.

Из уравнение луча имеем $X = A + t\vec{d}$. Уравнение для точки на поверхности серы выгядит слудуюзим образом $|X - C|^2 = r^2$. Подставив X во второе уравнение получаем:

$$|A + t\vec{d} - C|^2 = r^2$$

Обозначим $\vec{s} = A - C$, тогда

$$|s + t\vec{d}|^2 = (\vec{s}, \vec{s}) + 2t * (\vec{s}, \vec{d}) + t^2 * (\vec{d}, \vec{d}) = r^2$$

Получилось квадратное уравнение относительно t. Дискриминант считается считается следующим образом:

$$D = 4 * ((\vec{s}, \vec{d})^2 - d^2 * (s^2 - r^2))$$

Если D < 0 то объект, находящийся в объемлющей сфере, сразу можно выбрсывать из рассмотрения, так как луч его точно не пересекает.

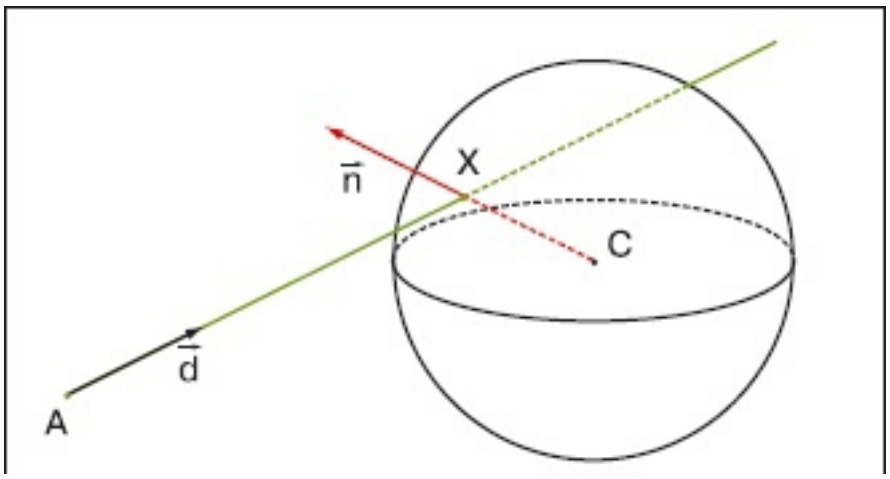

Рисунок 2.4. Пересечение сферы лучём

2.1.4 Ускорение алгоритма трассировки лучей

Распараллеливания алгоритмов часто используют для ускорения работы. Алгоритм трассировки лучей отлично поддаётся распраллеливанию посколько каждый пиксель экрана обрабатыевается независимо. Можно разбить экран на сектора в виде прямоугольников, которые будут обрабатываться параллельно, независимо друг от друга.

Также можно строить иерахическую структуру оболочек, что позволит отбрасывать сразу целые группы объектов, которе не пересекает данный луч.

2.2 Реализация модели освещения Уиттеда

2.2.1 Нахождение отражённого луча

Для нахождения направление отражённого луча $\vec{R}$ необходимо знать только направления нормали $\vec{N}$ и падающего луча $\vec{L}$.

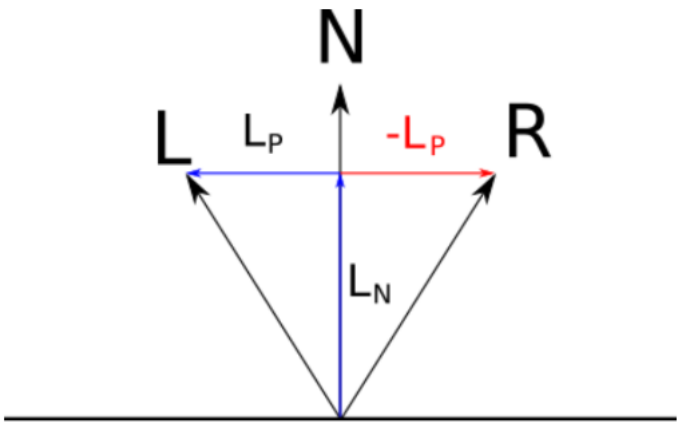

Рисунок 2.5. Разложение падающего луча

Падающий вектор $\vec{L}$ можно разложить на два проекции $\vec{L_N}$ и $\vec{L_P}$. Тогда $\vec{L} = \vec{L_N} + \vec{L_P}$. Так как $\vec{N}$ единичный вектор, то длинна прекции будет равна $(\vec{L}, \vec{N})$ и $\vec{L_N} = (\vec{L}, \vec{N})\vec{N}$. Следовательно $\vec{L_P} = \vec{L} - \vec{L_N}$. Отражённый же луч можно выразить как $\vec{R} = \vec{L_N} - \vec{L_P}$. Всё подставивив и упростив окончательно получаем $\vec{R} = 2(\vec{L}, \vec{N})\vec{N} - \vec{L}$.

2.2.2 Нахождение преломлённого луча

Преломлённый луч $\vec{P}$ можно найти изходя их того факта что падающий и преломлённый лучи лежат в одной плоскости и из закона Снелиуса, который записывается так:

$$sin(\alpha) * n_1 = sin(\gamma) * n_2$$

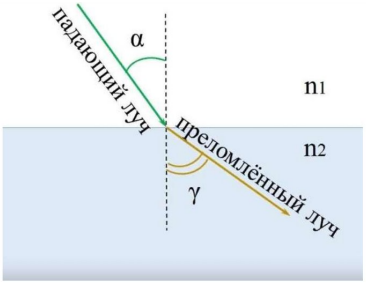

Рисунок 2.6. Преломление

Введём дополнительные обозначение $n = \frac{n_1}{n_2}$, $\vec{L}$ - падающий луч, $\vec{N}$ - нормаль. Можем получить уравнение для вектора преломлённого луча:

$$\vec{P} = n * (\vec{L} + cos(\alpha) * \vec{N}) - \vec{N}\sqrt{1 - sin^2(\gamma)}$$

Если подкоренное выражение отрицательно, то этот случай соответствует полному отражению.

2.2.3 Общая реализация модели освещения Уиттеда

Модель Уиттеда учитывает фоновое, диффузное и зеркальное освещение, а также эффеты отражения и преломления. Согласно этой модели суммарная интенсивность определяется следующим уравнением:

$$I = k_a I_a C + k_d I_d C + k_s I_s + k_r I_r + k_t I_t,$$

где

- k_a - коэффициент рассеянного отражения;
- k_d - коэффициент диффузного отражения;
- k_s - коэффициент зеокального отражения;
- k_r - коэффициент отражения;
- k_t - коэффициент преломления;
- I_a - интенсивность фонового компоненты;
- I_d - интенсивность диффузной компонеты;
- I_s - интенсивность зеркального компоненты;
- I_r - интенсивность отражённого луча;
- I_t - интенсивность преломлённого луча;
- C - цвет.

2.2.4 Расчёт интенсивностей

Интенсивность диффузного отражения не зависит от положения наблюдателья. Её интенсивность в точке можно вычислить по следующей формуле:

$$I_d = k_d * \sum I_i * (\vec{N}, \vec{L_i}),$$

где

- k_d - коэффициент диффузного отражения;
- I_i - интенсивность в точке, от i-го источника света;
- $\vec{N}$ - нормаль;
- $\vec{L_i}$ - единичный вектор, направленный в строну i-го источника света.

Интенсивность же зеркального отражения зависит от положения наблюдателья и её интенсивность в точке можно вычислить по следующей формуле:

$$I_d = k_s * \sum I_i * (\vec{S}, \vec{R_i})^n,$$

где

- k_s - коэффициент зеркального отражения;
- I_i - интенсивность в точке, от i-го источника света;
- $\vec{S}$ - единичный вектор, направленный в строну наблюдателя;
- $\vec{R_i}$ - единичный вектор, задающий направленние отражённого луча от i-го источника света;
- n - степень, аппроксимирующая пространственное распределение зеркально отраженного света.

В итоге получается следующая формула:

$$I = k_a \sum I_{ai} + k_d * \sum I_i * (\vec{N}, \vec{L_i}) + k_s * \sum I_i * (\vec{S}, \vec{R_i})^n + k_r I_r + k_t I_t,$$

2.3 Выбор используемых типов данных

В программе будут реализованны и использованны следующие структуры данных:

- Point3D - точка;
- Vector3D - ветор;
- Color - цвет;
- Polygon - полигон. Хранит индексы вершин, координат нормали и текстурных координат;
- SceneObject - объект сцены. Хранит координаты вершин, нормали, текстурные координаты и полигоны;
- LightObject - источник света. Хранит координаты источника света;
- Camera - камера. Хранит координаты, направление и вектор указывающий наверх;
- Scene - сцена. Хранит все объекты.

3 Заключение

Во время выполнения поставленной задачи были рассмотренны способы представления трёхмерных объектов, алгоритмы построения реалитического изображения, методы наложения текстур и модели освещения. Были проанализированны их достоинства и недостатки и выбраты самые подходящие для реализации поставленной задачи.

В ходе выполенения поставленной задачи мной были изучены возмоности QT Framework, полученны знания в такой перспективной сфере компьюторной графики, как трассировка лучей с глобальным освещением.

4 Источники

- Роджерс
- Ray tracing in one week
- Джеймс Д. Фоули, Андрис Ван Дам, Стивен К. Фейнер и Джон Ф. Хьюз, Принципы и практика компьютерной графики, второе издание.
- Компьютерная графика. Рейтрейсинг и растеризация