

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Аналитическая часть	6
1.1 Описание трёхмерного объекта	6
1.1.1 Геометрические примитивы	6
1.1.2 Воксельная модель	6
1.1.3 Полигональная модель	7
1.1.4 Вывод	7
1.2 Алгоритм построения трёхмерного изображения	8
1.2.1 Алгоритм с Z-буфером	8
1.2.2 Алгоритм с обратной трассировкой лучей	9
1.2.3 Вывод	10
1.3 Алгоритмы наложения текстур на трёхмерные объекты	10
1.3.1 Аффинное текстурирование	10
1.3.2 Перспективно-корректное текстурирование	10
1.3.3 Вывод	11
1.4 Модель освещения трёхмерных объектов	11
1.4.1 Модель Ламберта	11
1.4.2 Модель Фонга	11
1.4.3 Модель Уиттеда	12
1.4.4 Вывод	13
2 Конструкторская часть	14
2.1 Реализация алгоритма обратной трассировки лучей	14
2.1.1 Алгоритм обратной трассировки лучей	14
2.1.2 Нахождение пересечения с полигоном	16
2.1.3 Нахождение пересечения с объемлющей оболочкой	17
2.1.4 Ускорение алгоритма трассировки лучей	18
2.2 Реализация модели освещения Уиттеда	19
2.2.1 Нахождение отражённого луча	19
2.2.2 Нахождение преломлённого луча	19
2.2.3 Общая реализация модели освещения Уиттеда	20
2.2.4 Расчёт интенсивностей	20

2.3	Выбор используемых типов данных	21
	ЗАКЛЮЧЕНИЕ	22
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	23

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 производительности 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.

Цель: описать аналитический и конструкторский разделы курсовой работы по дисциплине "Компьютерная графика".

Задачи:

- провести анализ существующих методов для построения реалистических изображений и выбрать самые подходящие;
- описать в конструкторском разделе реализацию метода построения трёхмерной сцены и освещения.

1 Аналитическая часть

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 позволяют 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 описать с помощью пикселей. 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.[3]

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;

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 рассмотрены алгоритмы построения трёхмерного изображения, которые будут использоваться в написанном ПО. Эти алгоритмы являются важнейшей частью всей программы и выполняют основную работу по рендеру результирующего 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 упорядочивания.

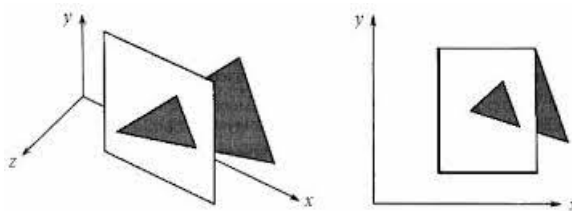


Рисунок 1 – Пример работы алгоритма с использованием Z-буфера

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 глубина 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.

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 интенсивность, иначе проверяется освещенность точки пересечения первичного луча и объекта сцены 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 относительно 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.[1]

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 Аффинное текстурирование

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

1.3.2 Перспективно-корректное текстурирование

Искажения в аффинном текстурировании вызваны допущением 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 всему полигону. Сами же точные значения u и

v считаются 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 текстурирование, учитывающее перспективу в отличие от аффинного метода.

1.4 Модель освещения трёхмерных объектов

1.4.1 Модель Ламберта

Простейшая модель освещения, чисто диффузное освещение. Считается, что свет падающий в точку, одинаково рассеивается по всем направлениям 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 - коэффициент диффузного отражения;
- \vec{n} - нормаль;
- \vec{l} - единичный вектор, направленный 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 появляются блики. Интенсивность в точке зависит от того насколько

близки отражённый вектор к вектору направленному из точки падения в сторону наблюдателя. В модели учитывается интенсивность фонового, диффузного и зеркально освещения. Для расчёта диффузной составляющей используется модель Ламберта.[5]

Считается по формуле:

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

где:

- k_d - коэффициент диффузного отражения;
- k_s - коэффициент зеркального отражения;
- k_a - коэффициент рассеянного отражения;
- I_s - интенсивность фонового освещения;
- \vec{n} - нормаль;
- \vec{l} - единичный вектор, направленный к источнику света;
- \vec{r} - единичный вектор, направленный 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. Использует для расчёта интенсивности глобальную модель освещения, учитывающую свет, взаимодействовавший с другими объектами. Помимо учёта фоновой, диффузной и зеркальной компонент в этой модели ещё учитывается интенсивность отражённого и преломлённого света от других тел. Согласно этой модели суммарная интенсивность определяется следующим уравнением:

$$I = k_a I_a C + k_d I_d C + k_s I_s + k_r I_r + k_t I_t, \quad (1)$$

где

- k_a - коэффициент рассеянного отражения;

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

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 Конструкторская часть

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 через каждый пиксель экрана и ищется его пересечения с объектами сцены. Луч выпущенный из камеры называется первичным. Пусть точка пересечения называется П1.

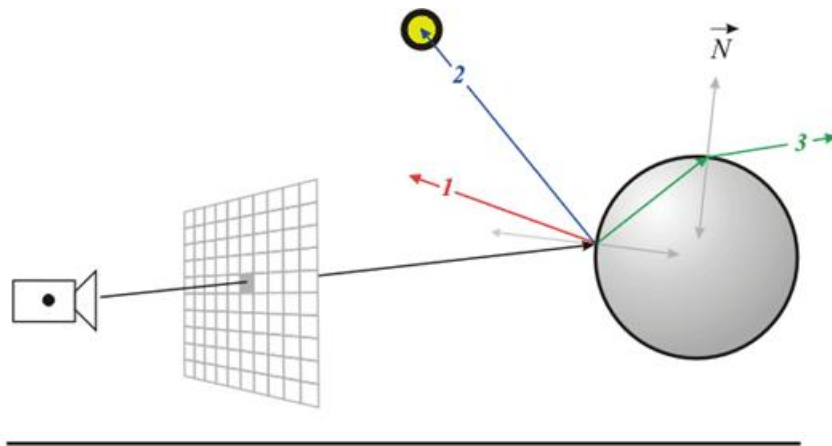


Рисунок 2 – Схема работы алгоритма трассировки лучей

Далее для каждого источника света определяется, видна ли для него точка П1. Для это испускается теневой луч(луч 2) 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 освещение считается по модели. Результатом будет сумма интенсивностей всех видимых источников света для П1. Если материал имеет отражающие свойства то просчитывается и испускается отражённый луч света(луч 1) и точно так же рекурсивно обрабатывается. Аналогично происходит с преломлёнными лучами(луч 3), если материал имеет преломляющие свойства.

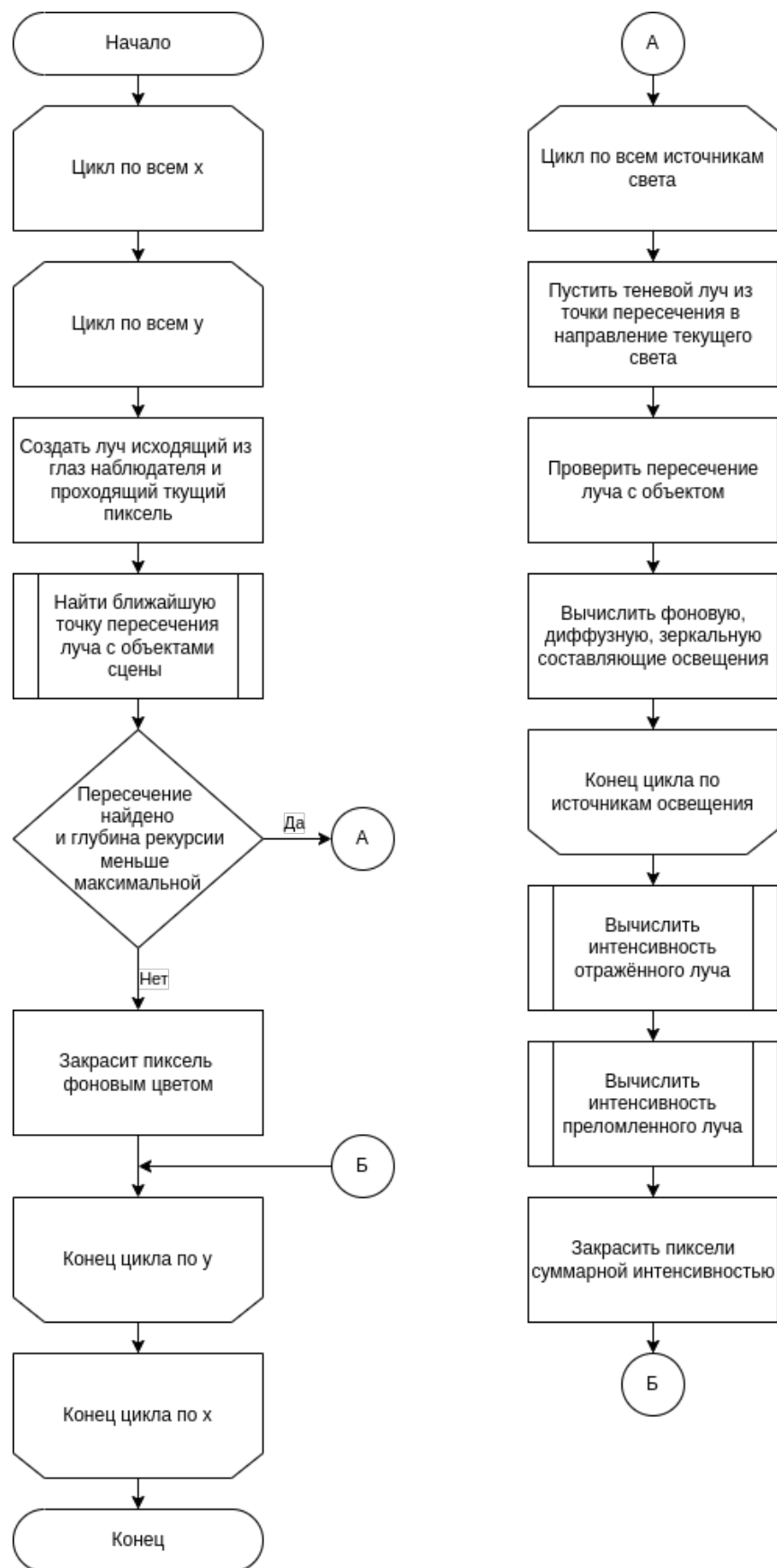


Рисунок 3 – Блок-схема алгоритма трассировки лучей

2.1.2 Нахождение пересечения с полигоном

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

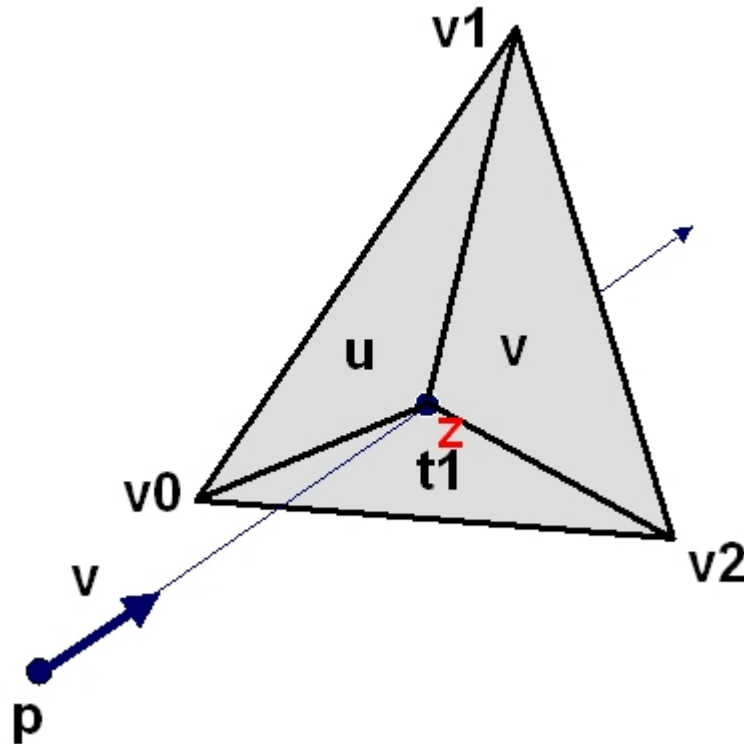


Рисунок 4 – Схема для поиска пересечения луча и треугольника

Введём следующие обозначения как на рисунке 4

- Z - точка пересечения;
- P - начало луча;
- t - расстояние от p до z ;
- \vec{d} - направление луча;
- V_0, V_1, V_2 - вершины треугольника;
- u, v, t_1 - 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 барицентрические координаты. Если каждая из этих координат будет больше или равна нулю, то искомая точка принадлежит треугольнику.[4]

$$Z(u, v) = (1 - u - v) * V_1 + u * V_2 + v * V_0 \quad (2)$$

Уравнение (2) берется просто из определения барицентрических координат, выражая точку пересечения z .

$$Z(t) = P + t * \vec{d} \quad (3)$$

Уравнение (3) это параметрическое уравнение прямой.

$$P + t * \vec{d} = (1 - u - v) * V_1 + u * V_2 + v * V_0 \quad (4)$$

Приравняв правые части уравнений (3) и (4) 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 пересечения с объемлющей оболочкой

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

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

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

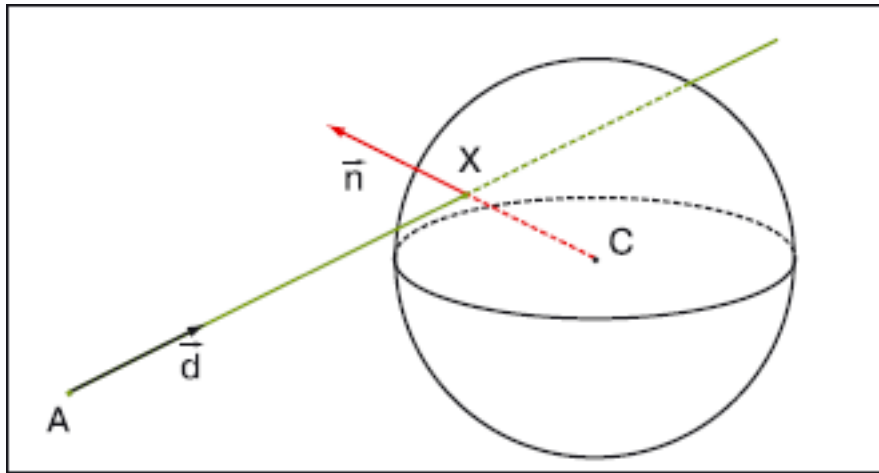


Рисунок 5 – Пересечение сферы лучём

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 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 выбрасывать из рассмотрения, так как луч его точно не пересекает.

2.1.4 Ускорение алгоритма трассировки лучей

Распараллеливание алгоритмов часто используют для ускорения работы. Алгоритм трассировки лучей отлично поддается распараллеливанию, поскольку каждый пиксель экрана обрабатывается независимо. Можно разбить экран на сектора в виде прямоугольников, которые будут обрабатываться параллельно, независимо друг от друга. Также можно строить иерархическую структуру оболочек, что позволит отбрасывать сразу целые группы объектов, которые не пересекает данный луч [2].

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} .

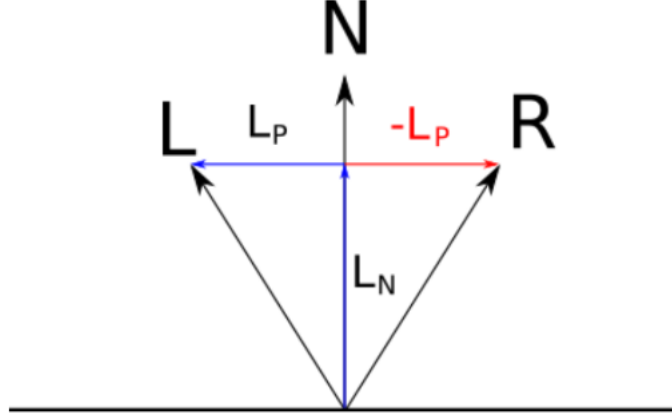


Рисунок 6 – Разложение падающего луча

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 проекции 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 подставив 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 исходя из того факта что падающий и преломлённый лучи лежат в одной плоскости и из закона Снелиуса, который записывается так:

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

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

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

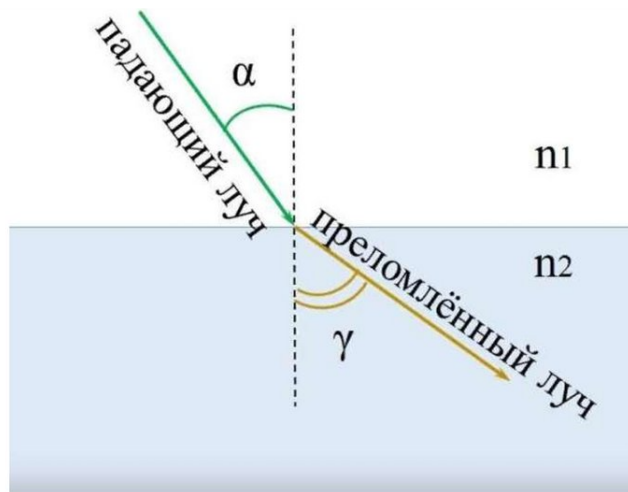


Рисунок 7 – Преломление

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 эффекты отражения и преломления, рекурсивно вычисляя интенсивность этих лучей. Согласно этой модели суммарная интенсивность определяется уравнением 1.

2.2.4 Расчёт интенсивностей

Интенсивность диффузного отражения не зависит от положения наблюдателя. 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 сторону i -го источника света.

Интенсивность же зеркального отражения зависит от положения наблю-

дателя 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 сторону наблюдателя;
- \vec{R}_i - единичный вектор, задающий направление 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 - вектор;
- Color - цвет;
- Polygon - полигон. Хранит индексы вершин, нормалей и текстурных координат;
- SceneObject - объект сцены. Хранит вершины, нормали, текстурные координаты и полигоны;
- LightObject - источник света. Хранит координаты источника света;
- Camera - камера. Хранит координаты, направление и вектор указывающий вверх;
- Scene - сцена. Хранит все объекты.

ЗАКЛЮЧЕНИЕ

Во время выполнения данной работы были рассмотрены способы представления трёхмерных объектов, были выбраны алгоритмы построения реалистического изображения, методы наложения текстур и модели освещения. Были проанализированы их достоинства и недостатки. Выбраны самые подходящие алгоритмы, удовлетворяющие выдвинутым требованиям реалистичности и быстродействия.

В ходе выполнения поставленной задачи мной были получены знания в такой перспективной сфере компьютерной графики, как трассировка лучей с глобальным освещением.

Также в ходе выполнения работы были выполнены следующие задачи:

- проанализированы существующие алгоритмы построения реалистических изображений, текстурирования, модели освещения, способы представления моделей и выбраны подходящие;
- описаны в конструкторской части реализуемые методы построения трёхмерного изображения и освещения.

Поставленная цель была достигнута: описаны аналитический и конструкторский разделы курсовой работы по дисциплине "Компьютерная графика".

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Роджерс Д. Алгоритмические основы машинной графики. — М.: Мир, 1989. — 512 с.
2. Гамбетта Г. Компьютерная графика. Рейтрейсинг и растеризация. — СПб.: Питер, 2022. — 224 с.
3. Методы представления дискретных данных [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index, свободный – (21.09.22)
4. Пересечение луча и треугольника [Электронный ресурс]. – Режим доступа: <http://ray-tracing.ru/articles213>, свободный – (23.09.22)
5. Простые модели освещения [Электронный ресурс]. – Режим доступа: <http://grafika.me/node/344/>, свободный – (19.09.22)