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**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА *К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ НА ТЕМУ:*

«Моделирование геометрических тел простой формы с изменяемым коэффициентом отражения»

Студент группы **ИУ7-54Б** _____ (Подпись, дата) **Разин А.В.** (И.О. Фамилия)

Руководитель ВКР _____ (Подпись, дата) **Куров А.В.** (И.О. Фамилия)

Нормоконтролер _____ (Подпись, дата) **Кузнецова О.В.** (И.О. Фамилия)

2023 г.

Содержание

1 Введение

Компьютерная графика играет важнейшую роль в современной жизни. Она позволяет нам видеть различные задачи в совершенно новом свете, а также создавать и исследовать визуально привлекательные и живые изображения. С ее помощью мы можем представлять друг другу информацию с помощью цветного и анимированного содержимого. Получение качественного изображения необходимо во многих областях, включая медицину, искусство и игры. Особенно важной становится задача генерации реалистичного изображения, с развитием технологий люди привыкают к все более правдоподобной картинке. Чаще всего люди подмечают нереалистичность изображения при наблюдении света: его отражения и преломления.

Целью данной работы является разработка программного обеспечения, моделирующего отражения от геометрических тел.Для достижения поставленной цели требуется решить следующие задачи:

1. Формализировать представление объектов сцены и описать их
2. Проанализировать алгоритмы построения реалистичных изображений и теней
3. Выбрать наилучшие алгоритмы для достижения цели из рассмотренных
4. Проанализировать полученную модель взаимодействия света с объектами
5. Выбрать програмные средства для реализации модели
6. Реализовать полученную модель
7. Создать интерфейс
8. Провести замеры временных характеристик полученной модели

2 Аналитическая часть

2.1 Формализация объектов синтезируемой сцены

На визуализируемой сцене могут находится следующие объекты:

1. **Точечный источник света** Данный источник света излучает свет во всех направлениях, интенсивность света убывает при отдалении от источника. Источник характеризуется:

   (a) Положением в пространстве

   (b) Интенсивностью

   При расчете отражений будет использоваться интенсивность источника,для расчета интенсивности пикселей. Цвет свечения будет описываться через значения RGB.

2. **Объекты сцены**

   Объектами сцены являются трехмерные примитивы:

   2.1. **Шар**

   Для описания шара потребуется:

   2.1.1. Радиус

   2.1.2. Координаты центра

   2.2. **Куб**

   Для описания куба потребуется:

   2.2.1. Координаты центра

   2.2.2. Координаты вершин относительно центра

   2.2.3. Индексы соединенных вершин(Ребра)

   2.3. **Конус**

   Для описания конуса потребуется:

   2.3.1. Координаты центра окружности основания конуса

2.3.2. Радиус основания конуса

2.3.3. Координата верхней точки конуса

2.4. **Цилиндр**

Для описания цилиндра потребуется:

2.4.1. Координаты центра нижнего основания цилиндра

2.4.2. Координата центра верхнего основания конуса

2.4.3. Радиус цилиндра

Заметим, что каждый из примитивов также должен описываться своим цветом в формате RGB, а также коэффициентами рассеянного, диффузного,зеркального отражения

3. **Камера**

В данном случае камера может описываться:

3.1. Координатами своего положения

3.2. Вектором направления взгляда

3.3. Правым вектором пространства камеры

3.4. Верхним вектором пространства камеры

2.2 Анализ моделей отражения

Свет отраженный от объекта может быть диффузным и зеркальным. Диффузное отражение происходит, когда свет поглощается поверхностью, а затем вновь испускается, в данном случае отражение равномерно рассеивается по всем направлениям и положение наблюдателя не имеет значения.Зеркальное отражение происходит от внешней поверхности объекта, оно является направленным и зависит от положения наблюдателя. Так как отражение происходит от внешней части объекта ,то отраженный свет сохраняет свойства падающего, например в случае если белый свет отражается от красного тела, отраженный свет также будет нести в себе часть красного цвета. Для расчета интенсивности света данных отражений существует несколько моделей:

1. Модель Ламберта

2. Модель Фонга [1]

2.2.1 Модель Ламберта

В данной модели рассматривается диффузная составляющая отражения.

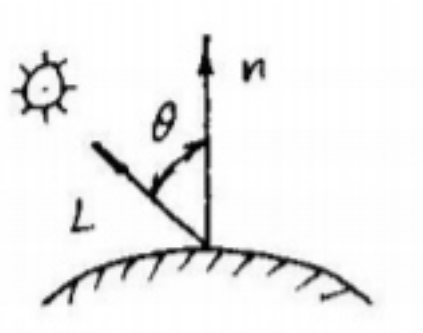

Рисунок 2.1 – Модель ламберта

Считается, что интенсивность отраженного света пропорциональна косинусу угла между направлением света и нормалью к поверхности:

$$I = k_a I_a + I_l k_l \cos\theta \quad 0 \leq \theta \leq \pi/2. \tag{2.1}$$

В формуле 2.1:

1. k_a, k_d - коэффициенты рассеянного, диффузного отражения соответственно

2. I_a, I_l - интенсивность рассеянного и диффузного отражения

3. $theta$ - угол между нормалью к поверхности и направлением света

Заметим что значения приведенных коэффициентов лежат на отрезке от 0 до 1.

Однако интенсивность света должна убывать с увеличением расстояния от источника до объекта, эмпирически было выведено следующее соотношение:

$$I = k_a I_a + \frac{I_l k_l \cos\theta}{d + K}. \tag{2.2}$$

В данном случае добавлены d, K, в случае если точка наблюдения на бесконечности, то d - определяется положением объекта, ближайшего к точке наблюдения, то есть он будет освещаться с максимальной интенсивностью источника, а дальние объекты - с уменьшенной. K в данном случае - произвольная постоянная. [1]

2.2.2 Модель Фонга

Данная модель также учитывает зеркальную составляющую отражения

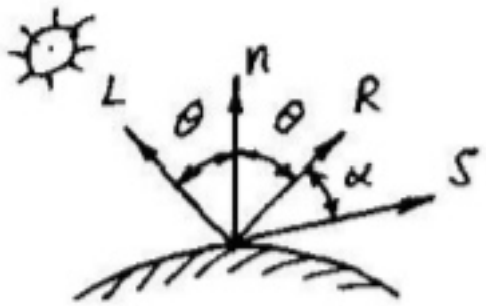

Рисунок 2.2 – Расчет зеркальной составляющей отражения

Зеркальная составляющая отражения имеет следующий вид:

$$I_s = I_l \omega(i, \lambda) \cos^n \alpha. \tag{2.3}$$

В формуле 2.3 символы соответственно означают:

1. $\omega(i, \lambda)$ - кривая отражения,показывающая отношение зеркально отраженного света к падающему, как функцию угла падения i и длинны волны λ

2. α - угол между отраженным лучом и вектором, проведенным из точки падения луча в точку наблюдения

3. n - степень, аппроксимирующая пространственное распределение отраженного света

4. I_l - интенсивность падающего луча

Функция $\omega(i, \lambda)$ сложна, так что ее заменяют константой k_s, получаемой экспериментально.

Таким образом формула принимает следующий вид:

$$I = k_a I_a + k_d I_l(\hat{n} \cdot \hat{L}) + k_s I_l(\hat{S} \cdot \hat{R})^n. \tag{2.4}$$

В данном случае косинусы вычисляются с помощью скалярного произведения нормированных векторов:

1. $\hat{n}$ - вектор нормали поверхности в точке падения
2. $\hat{L}$ - вектор падающего луча
3. $\hat{S}$ - вектор наблюдения
4. $\hat{R}$ - вектор отражения

Символ ˆ означает, что данный вектор нормированный.[1]

2.2.3 Выбор модели отражения

Данные модели описывают простую (локальную) модель освещения,то есть модель освещения в которой учитывается свет, попадающий в рассматриваемую точку от источника света. Для получения отражений стоит использовать глобальную модель освещения, в которой также учитывается интенсивность света, отраженного от других поверхностей, пример описания глобальной модели излучения можно найти в секции 2.4.1. Заметим, что глобальная модель в случае алгоритма Уиттеда освещения использует идею модели Фонга (см. 2.9).

2.3 Анализ алгоритмов закраски

Для сглаживания и добавления реализма изображениям необходимо использовать алгоритмы закраски. В данном случае будут рассмотрены 2 алгоритма закраски:

1. Простая закраска
2. Закраска методом Гуро
3. Закраска методом Фонга

2.3.1 Простая закраска

В случае использования простой закраски считается,что и источник света и наблюдатель находятся в бесконечности, так что диффузная составляющая одинакова (она зависит от угла падения), вектор наблюдения также будет для всех точек одинаковым, то есть зеркальная составляющая также не будет изменяться (см. 2.2). Таким образом данный алгоритм является быстрым, однако в случае, если закрашиваемая грань является результатом аппроксимации тела, будет заметен резкий переход между интенсивностями. [1]

2.3.2 Закраска методом Гуро

В случае использования метода Гуро можно получить сглаженное изображение, для этого сначала определяется интенсивность вершин многоугольника, а затем с помощью билинейной интерполяции вычисляется интенсивность каждого пиксела на сканирующей строке.

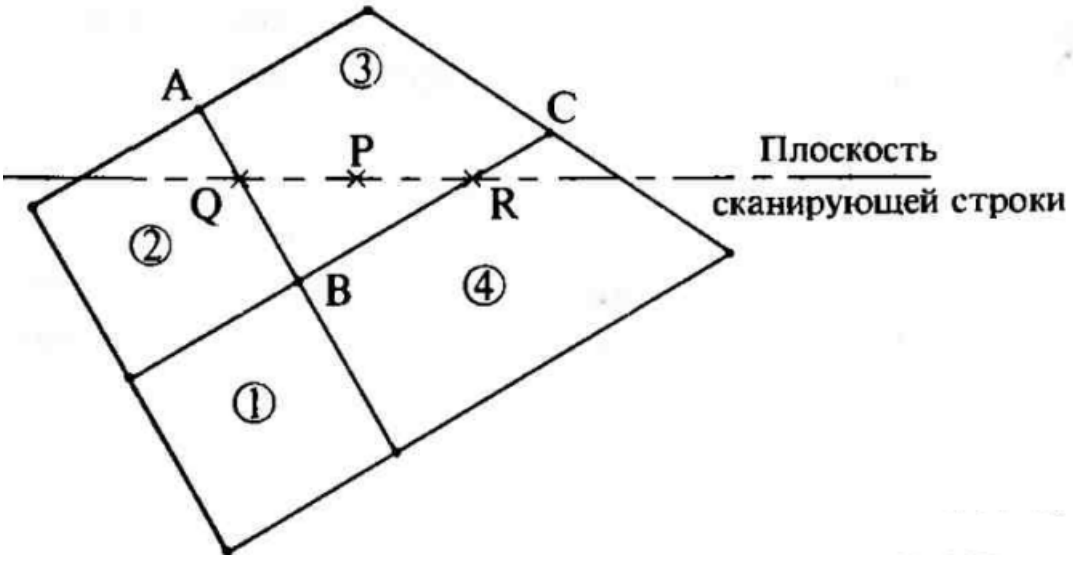

Рисунок 2.3 – Пример полигональной поверхности

Например, рассмотрим участок полигональной поверхности на рисунке 2.3. Значение интенсивности в точке P определяется линейной интерполяции значений интенсивностей в точках Q и R. Для получения интенсивности в точке Q можно провести линейную интерполяцию интенсивности в вершинах A и B (см. формула 2.5).

$$I_Q = uI_A + (1-u) * I_B \quad 0 \leq u \leq 1, t = \frac{AQ}{AB}. \tag{2.5}$$

Таким же образом рассчитывается интенсивность в точке B, после чего интерполяция используется еще раз, для поиска значения интенсивности в точке P (см. формула 2.6). [1]

$$I_P = tI_Q + (1-t) * I_R \quad 0 \leq t \leq 1, t = \frac{QP}{QR}. \tag{2.6}$$

Недостатки метода:

1. Появление полос Маха
2. Не учитывает кривизны поверхности (случай, если нормали поверхностей одинаково ориентированы)

Достоинства метода:

1. Получение сглаженного изображения
2. Небольшие трудозатраты

2.3.3 Закраска методом Фонга

Закраска Фонга требует больших вычислительных затрат, однако она решает множество проблем метода Гуро.В данном методе вместо интерполяции интенсивностей света, производится интерполяция вектора нормали.Таким образом учитывается кривизна поверхности. В случае картинки 2.3, нормали

в соответствующих точках рассчитывалась бы формулами 2.7. [1]

$$\begin{aligned} n_Q &= un_A + (1-u)n_B \quad 0 \le u \le 1 \\ n_R &= wn_B + (1-w)n_C \quad 0 \le w \le 1 \\ n_P &= tn_Q + (1-t)n_R \quad 0 \le t \le 1. \end{aligned} \tag{2.7}$$

где

$$u = \frac{AQ}{AB}$$

$$w = \frac{BR}{BC} \tag{2.8}$$

$$t = \frac{QP}{QR}$$

Недостатки метода:

1. Высокая трудозатратность алгоритма

Достоинства метода:

1. Получение сглаженного изображения
2. Учет кривизны поверхностей
3. Улучшенная реалистичность зеркальных бликов (по сравнению с методом Гуро)‘’

2.3.4 Выбор метода закраски

На картинах 2.4 и 2.5 представлены различные методы закраски. Заметим, что наиболее реалистичным методом закраски является закраска Фонга, так как она учитывает кривизну тел и лучше визуализирует блики, так что для высокого качества изображения стоит воспользоваться методом Фонга.

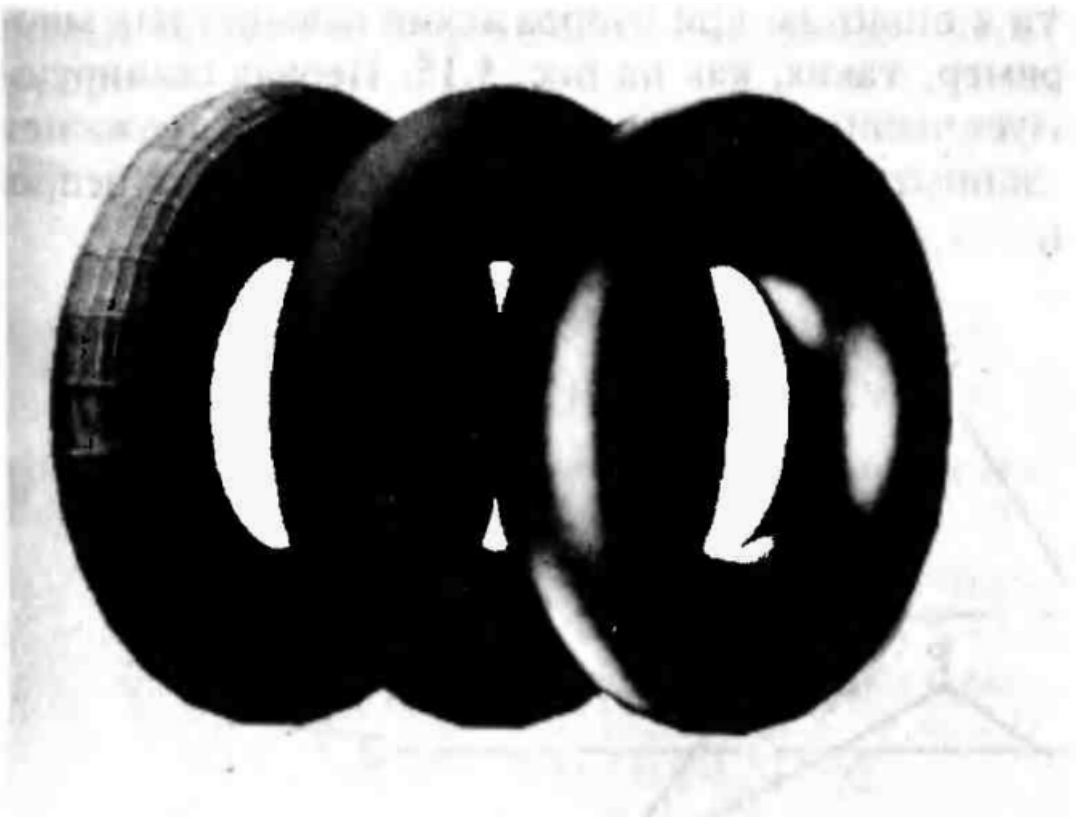

Рисунок 2.4 – Сравнение методов закраски: слева - простая, посередине - методом Гуро, справа - методом Фонга

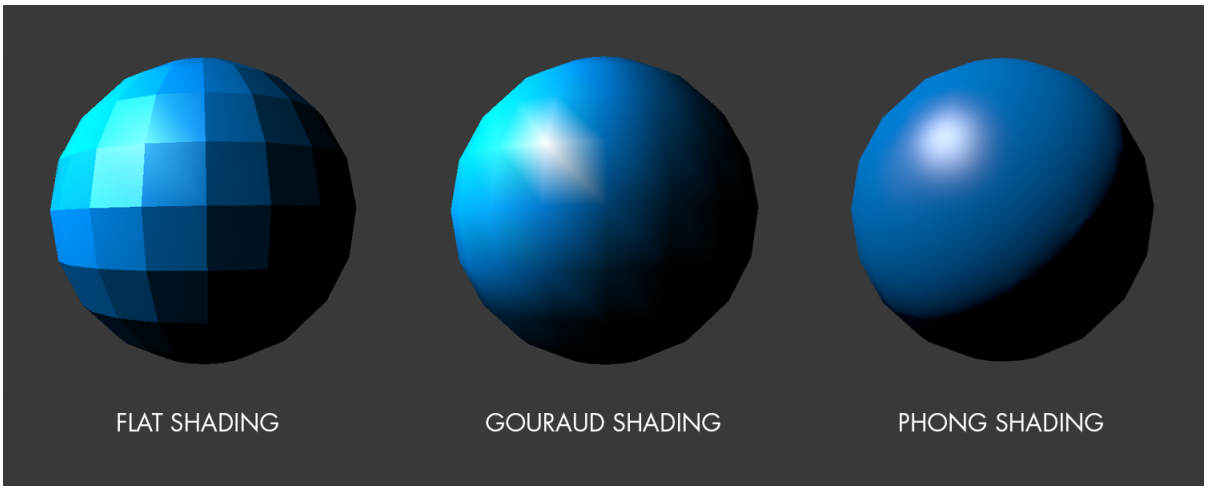

Рисунок 2.5 – Сравнение бликов при различных методах закраски

2.4 Анализ алгоритмов визуализации

Для начала рассмотрим пример отражения лучей:

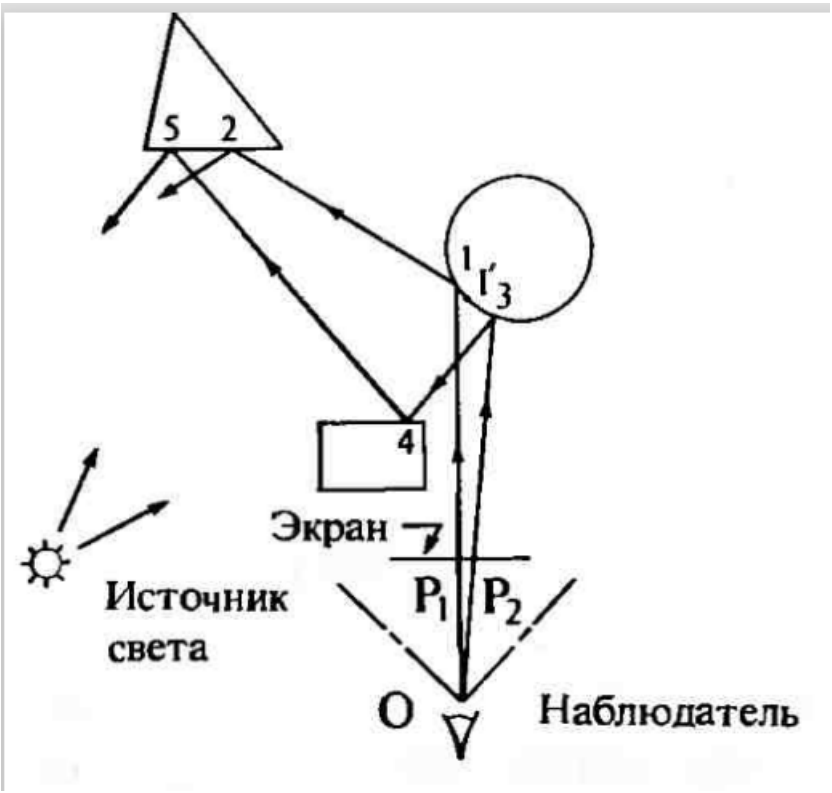

Рисунок 2.6 – Пример трассировки луча

Заметим что на рисунке 2.6 призма, загороженная от наблюдателя параллелепипедом,становится видимой из-за отражения в сфере. Точка 5 видима, так как отражается от обратной стороны параллелепипеда в точке 4 к точке 3 на сфере, а затем к наблюдателю. Таким образом, при создании глобальной модели освещения, алгоритмы , основанные на удалении невидимых поверхностей не будут давать изображения необходимого качества. Таким образом глобальная модель освещения является частью алгоритмов выделения видимых поверхностей путем трассировки лучей[1]

При построении реалистичного изображения необходимо с полированными поверхностями необходимо визуализировать отражения света от тел. Существуют множество подходов для создания реалистичных изображений:

1. Трассировка световых лучей (Ray tracing)
2. Отображения отражений (Reflection mapping)
3. Трассировка лучей в пространстве изображения(Screen-space reflections)

2.4.1 Алгоритм трассировки лучей

Основная идея алгоритма - симуляция физического процесса прохождения света

В реальной жизни объекты являются видимыми, в случае если они отражают свет от источника, после чего данные лучи света попадают в человеческий глаз.Аналогичная идея заложена в данном способе создания изображения - необходимо отследить движение лучей света. Заметим, что отслеживать путь всех лучей света не стоит, так как это неэффективно, при построении изображения внимание следует уделять объектам видимыми со стороны наблюдателя. В таком случае можно отслеживать лучи света, исходящие из точки наблюдения, т. е. производить трассировку лучей в обратном направлении.В нашем случае лучи стоит проводить через центры пикселей изображения, считается ,что наблюдатель находится на бесконечности, из-за чего все лучи параллельны оси OZ.[1, 3]

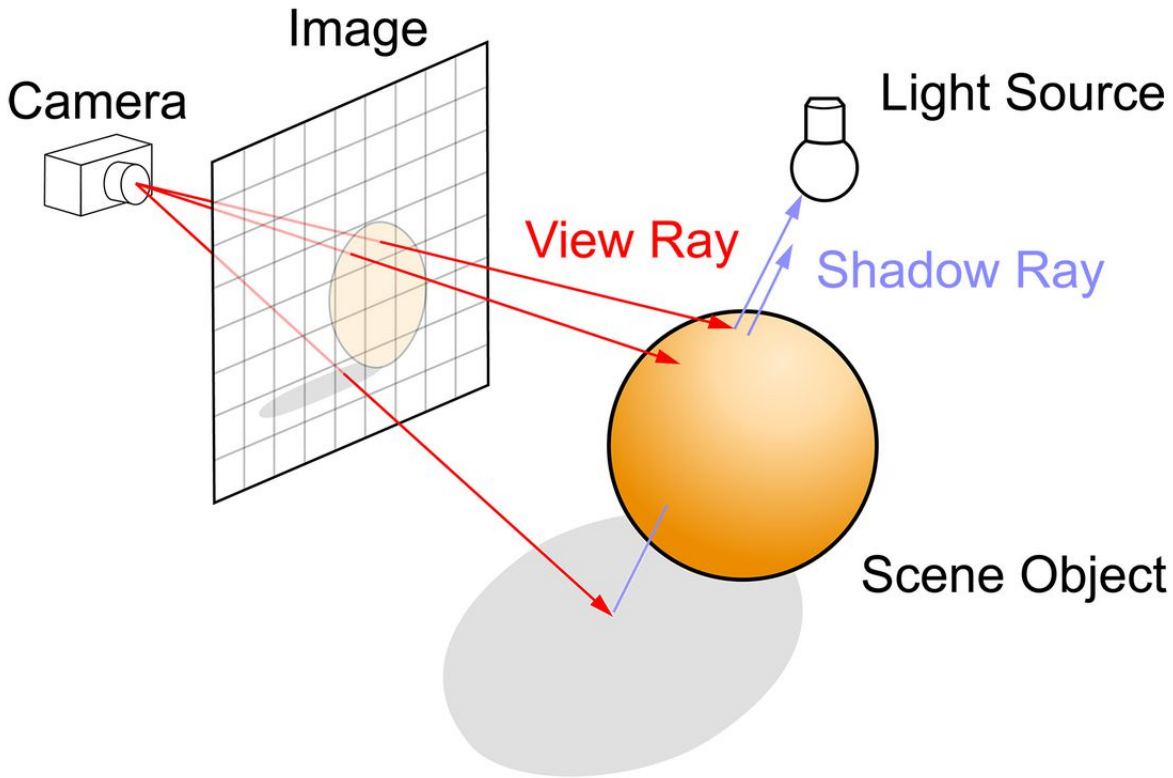

Рисунок 2.7 – Пример трассировки луча

Первые работы принадлежат Уиттеду и Кэю.Алгоритм Уиттеда более общий и часто используется.[Роджерс с.438] Уиттед пользуется моделью , в

которой диффузная и зеркальная составляющие отражения рассчитываются подобно локальной модели (см. 2.2). Диффузное отражения одинаково во всех направлениях, так что наибольшую проблему представляет расчет зеркальных отражений.[1]

Рисунок 2.8 – Расчет зеркального отражения луча в алгоритме Уиттеда

На рисунке 2.8 луч **V** падает на поверхность в точку **Q**, после чего отражается в направлении **r** и преломляется в направлении **p**. В данном случае:

1. I_t - интенсивность света, проходящего по преломленному лучу **p**.
2. η - показатели преломления сред(влияют на направление преломленного луча)
3. $\hat{S}$,$\hat{R}$ - полученные вектора наблюдения и отражения
4. $\hat{L_j}$ - Вектор к источнику света j

Тогда наблюдаемая интенсивность **I** выражается формулой:

$$I = k_a I_a + k_d \sum_j I_{l_j}(\hat{n} \cdot \hat{L_j}) + k_s \sum_j I_{l_j}(\hat{S} \cdot \hat{R}_j)^n + k_s I_s + k_t I_t. \qquad (2.9)$$

В формуле 2.9 соответственно означают:

1. k_a, k_d, k_s - коэффициенты рассеянного, диффузного,зеркального отражения соответственно

2. k_t - коэффициент пропускания

3. n - степень пространственного распределения Фонга

В данном случае знак ˆ означает что данный вектор нормализован. Значения коэффициентов определяются внешней средой,свойствами материала объектов и длинной волн света Таким образом возможно посчитать интенсивность света для отраженной и преломленной части луча. После чего полученные вычисления необходимо выполнить еще раз для отраженного и преломленного луча и т. д. , а также сложить полученные интенсивности. Теоретически свет может отражаться бесконечно, так что стоит ограничить число рассматриваемых отражений либо определенным числом, либо не рассматривать лучи с интенсивностью меньше определенного значения Таким образом данный алгоритм имеет асимптотику $O(N \cdot C \cdot 2^K)$, где **N** - количество тел,**C** - количество испускаемых лучей, **K** - количество рассматриваемых отражений света. [1]

Достоинства алгоритма:

1. Создание реалистичных изображений [1, 3, 4]

2. Возможность наблюдения физических явлений, так как алгоритм симулирует поведение света в реальной жизни [1, 3, 4]

Недостатки алгоритма:

1. Время работы [1, 3, 4]

2. Количество требуемой памяти, так как в памяти необходимо хранить все отраженные и преломленные лучи, полученные при предыдущих расчетах [1, 3, 4]

2.4.2 Трассировка лучей в пространстве изображения

Основная идея алгоритма - симуляция физического процесса прохождения света, рассматривая только видимые объекты
Обычно при необходимости расчета отражений и теней уже известны объекты, которые находятся на сцене.При использовании SSR(Screen-space reflections),ис информация о имеющихся объектов из-за чего трудозатраты на создание изображения заметно сокращаются. Асимптотическая оценка данного алгоритма аналогична асимптотической оценке алгоритма трассировке лучей, однако в данном случае будут анализироваться только видимые объекты.[4]

Перед началом алгоритма требуется информация для каждого пикселя:

1. Координата Z наиближайшей к наблюдателю поверхности
2. Нормаль данной поверхности

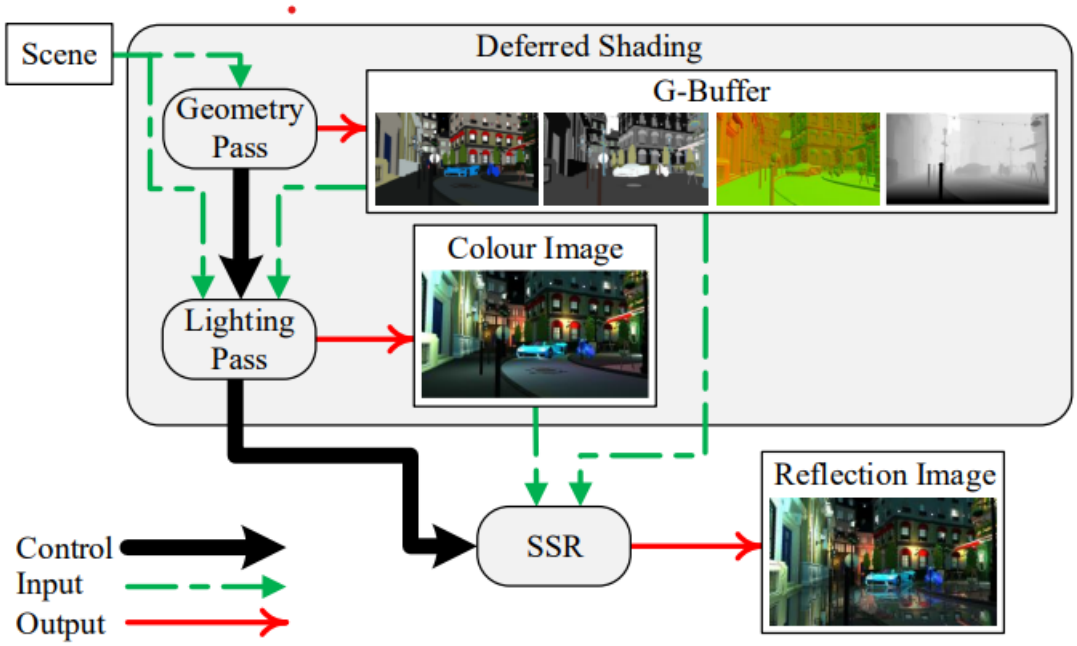

Рисунок 2.9 – Поток данных при использовании SSR

До начала работы самого алгоритма необходимо подготовить данные, что происходит в два этапа:

1. Геометрический проход(Geometry pass)
2. Световой проход(Lightning pass)

На картинке 2.9 используется понятие **G-buffer**, данный буфер содержит все необходимые данные для начала работы алгоритма,данные для данного буфера будут получены после геометрического прохода. В общем случае он содержит для каждого пикселя:

1. Нормали к видимым поверхностям
2. Значение z наиближайшей видимой фигуры
3. Свойства материалов, значимые для трассировки света(коэффициенты диффузного и зеркального отражения)

При световом проходе для каждого пикселя выбираются источники,которые влияют на его интенсивность Работа SSR аналогична работе алгоритма **Ray tracing**,однако информация о видимых объектах уже получена и будут рассматриваться только они. Из-за этого,если часть объекта не видима то изображение будет не корректным, как ,например, на картинке 2.10.[4, 5]

Рисунок 2.10 – Некорректный расчет отражений при использовании SSR

Достоинства алгоритма:

1. Меньшее число трудозатрат,чем при использовании алгоритма трассировки лучей [4]

Недостатки алгоритма:

1. При наличии невидимых для наблюдателя частей объектов в отражении изображение будет некорректным [4]

2. Искаженем Искажение геометрии при генерации изображений [4]

2.4.3 Отображение отражений

Основная идея - создание текстуры с уже просчитанными отражениями,после чего наложить ее на зеркальный предмет
Для того чтобы получить текстуру,достаточно предварительно рассчитать изображения в 6 направлениях для необходимого объекта, получив кубическую карту, ее пример представлен на картинке 2.11.[5]

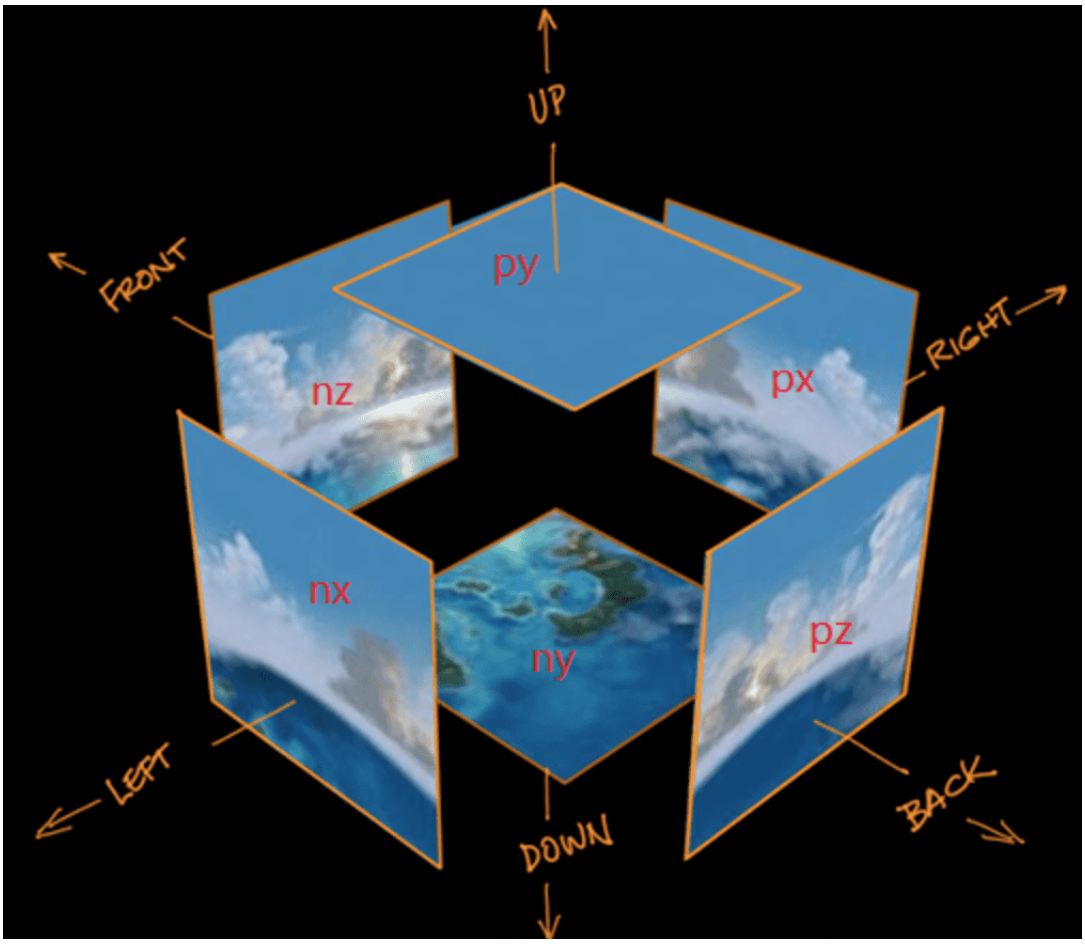

Рисунок 2.11 – Пример кубической карты(cube maps)

Например для объекта 2.12, будет получена карта 2.13

Рисунок 2.12 – Положение объекта для построения карты

Рисунок 2.13 – Полученная кубическая карта

Однако динамически обновлять данную текстуру очень трудозатратно (необходимо отрсиовать сцену 6 раз).Асимптотическая оценка зависит от алгоритма, выбранного для построения изображения каждой грани куба.

Достоинства алгоритма:

1. При статических текстурах не тратится время на расчет отражений и теней [5]

Недостатки алгоритма:

1. Расчет изменяющейся картинки очень трудозатратен [5]

2.4.4 Выбор оптимального алгоритма

При реализации отражений примитивов точность их представления играет решающую роль.Единственный алгоритм из рассмотренных, который позволяет представить максимально реалистичное изображение - **алгоритм трассировки лучей**.Так как выбранные примитивы простые, то при использовании данного алгоритма вычислительная сложность не будет являться критически высокой. Алгоритм алгоритм трассировки в пространстве изображения хорошо показывает себя при необходимости генерации неправдоподобных отражений, кубические карты используются при возможности создания статических изображений, эти ограничения не позволяют их использовать при генерации реалистичных изображений.

2.5 Выводы из аналитического раздела

В данном разделе были проанализированы модели отражения, методы закраски и алгоритмы создания отражений. Таким образом были выбраны:

1. **Алгоритм создания отражений** - Алгоритм обратной трассировки лучей
2. **Модель отражения** - Модель отражения Фонга
3. **Метод закраски** - Метод закраски Фонга

Входными данными для полученной системы будут являться:

1. Интенсивность источника
2. Спектральные характеристики материала примитива

3. Положение источника

4. Положение примитива

5. Угол поворота примитива

3 Конструкторская часть

3.1 Общий алгоритм построения изображения

Рассмотрим алгоритм построения кадра, пока что не рассматривая конкретную реализацию алгоритма трассировки лучей на картинке 3.1. Заметим, что при перемещении наблюдателя, необходимо заново приводить все рассматриваемые объекты к пространству камеры.

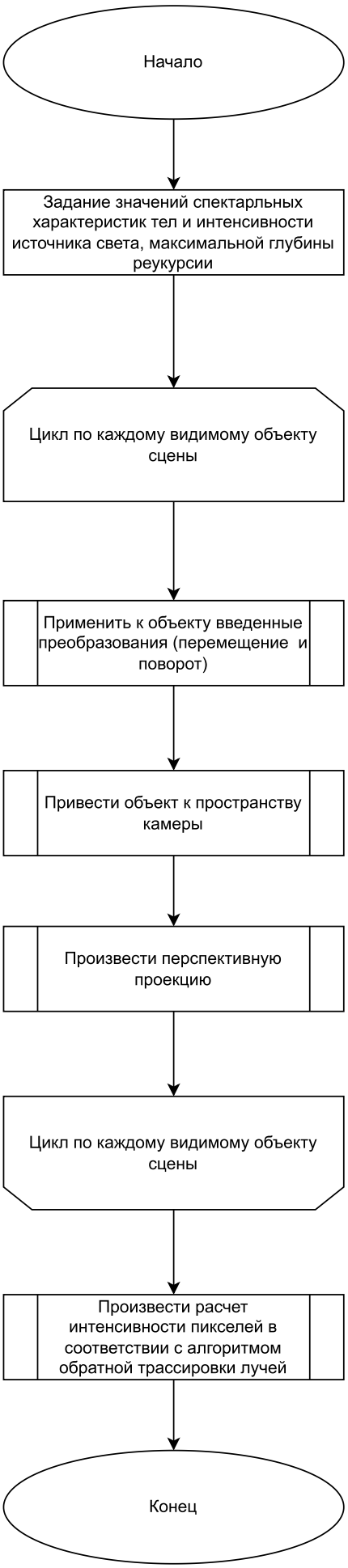

Рисунок 3.1 – Общий алгоритм построения кадра

3.2 Алгоритм обратной трассировки лучей

Алгоритм трассировки лучей для 1 пикселя приведен на картинке 3.2. Входными данными для него являются: описания объектов, число пикселей на экране, интенсивность источника света.

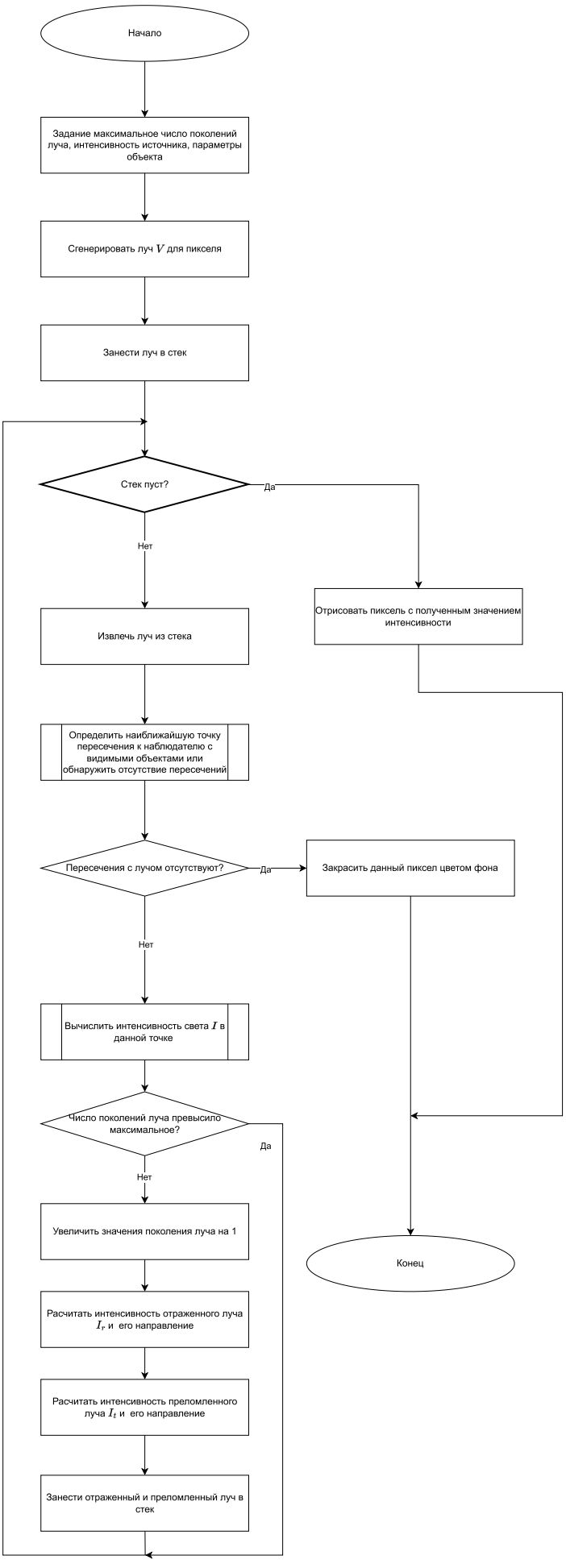

Рисунок 3.2 – Алгоритм трассировки лучей для 1 пикселя

Теоретически свет может отражаться бесконечно, введение ограничения на максимальное число поколений луча позволит ограничить время построения изображения. Необходимо отметить, что каждый луч при пересечении с поверхностью образует 2 луча: преломленный и отраженный, таким образом можно построить дерево расчета лучей. Пример такого дерева приведен на картинке 3.3, на данном дереве левая ветвь соответствует лучам, полученным в результате отражения, правая - в результате преломеления, стрелками указан обход дерева, при спуске по дереву луч строится, при поднятии по дереву интенсивность вычисляется. Можно заметить что максимальное поколение лучей является максимальной высотой получаемого дерева.

В данном случае при помещении луча в стек кладутся его следующие параметры:

1. Вычисленная интенсивность
2. Параметры луча для составления его уравнения
3. Текущее поколение [1]

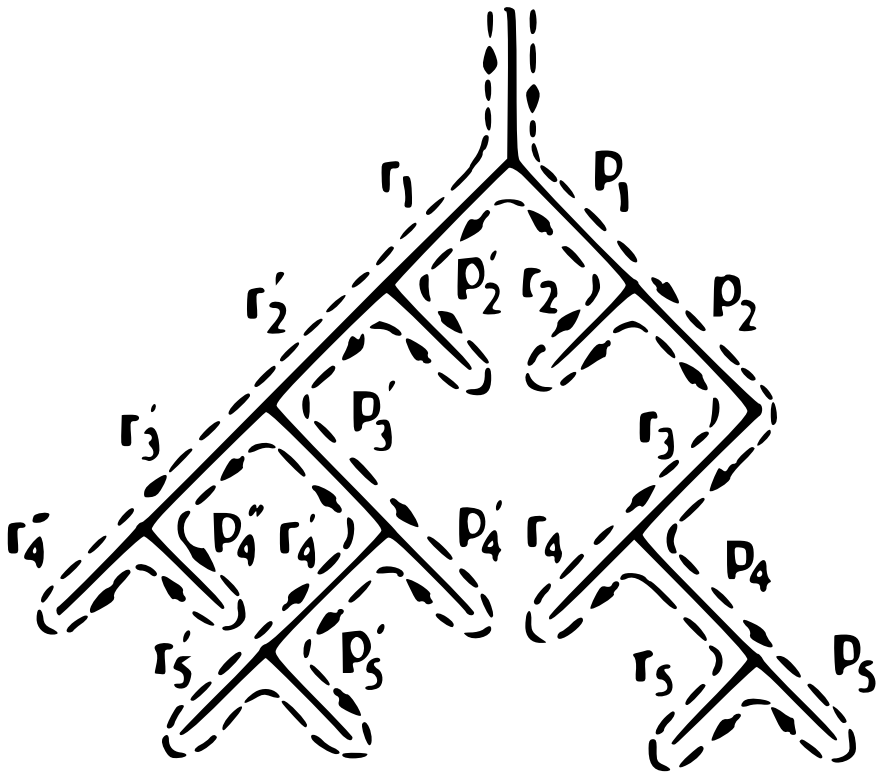

Рисунок 3.3 – Пример дерева трассировки луча

3.2.1 Расчет интенсивности света в соответствии с выбранной моделью

На картинке 3.2 необходимо рассчитать интенсивность света, рассмотрим как это будет происходить в соответствии с моделью освещения Уиттеда. (см 2.8) Если изображение цветное то приведенный блок (см. 3.4) выполняется трижды - для каждого из основных цветов. Идея данного алгоритма проста: в случае если вектор от источника света к рассматриваемой точке пересекает непрозрачное тело, источник света не освещает данную точку и мы переходим к следующему, если существует пересечение с прозрачными объектами, то они поглотят часть света и его интенсивность снизится. Заметим, что в данной реализации преломление света в прозрачных телах не учитывается. Считается, что нормаль к поверхности в заданной точке можно получить из описания объекта. (см. 2.1). Рисунок 3.4 соответствует этапу «Вычислить интенсивность света I в данной точке» в алгоритме обратной трассировки лучей (см. 3.2). Данная модель освещения может быть модифицирована для учета преломления света. [1]

3.2.2 Нахождение пересечения с объектами сцены

Один из самых трудозатратных этапов данного алгоритма - поиск пересечения с объектами сцены. Необходимо быстро определять координаты точки пересечения испущенного луча с данными примитивами.

Для начала необходимо ввести уравнение самого луча (см. **??**).

$$P(t) = \vec{E} + t\vec{D}, t \geq 0 \tag{3.1}$$

Также уравнение 3.1 имеет запись

$$\begin{aligned} x(t) &= x_E + tx_D \\ y(t) &= y_E + ty_D \\ z(t) &= z_E + tz_D. \end{aligned} \tag{3.2}$$

Таким образом луч определяется: точкой обзора - $\vec{E} = (x_E, y_E, z_E)$ и вектором

направления - $\vec{D} = (x_D, y_D, z_D)$. Значение t определяет направление луча: в случае если $tge0$, точка на луче находится после точки обзора, иначе - за.Таким образом для поиска наиближайшей точки пересечения, необходимо найти наименьшее неотрицательное значение t.[1, 7]

Уравнение сферы Сфера с единичным радиусом может быть задана следующим образом:

$$\vec{P} \cdot \vec{P} = 1 \tag{3.3}$$

Для получения условия пересечений достаточно подставить уравнение луча из 3.1 в 3.3 и решить полученное уравнение относительно t.

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.4}$$

После проведенных преобразований будет получена формула 3.4, где:

1. $a = \vec{D} \cdot \vec{D}$
2. $b = 2\vec{E} \cdot \vec{D}$
3. $c = \vec{E} \cdot \vec{E} - 1$

В случае если вещественные решения уравнения 3.4 отсутствуют, то пересечения луча также отсутствуют, если только одно решение - существует одно пересечение и т. д. Если значение отрицательное то точка пересечения находится за точкой наблюдения и нас не интересует.[7]

Уравнение цилиндра Уравнение 3.5 задает бесконечный цилиндр.

$$x^2 + y^2 = 1 \tag{3.5}$$

Для ограничения цилиндра необходимо ввести требования к значению z цилиндра:

$$x^2 + y^2 = 1, z_{min} < z < z_{max} \tag{3.6}$$

Для нахождения пересечений подставим 3.2 в 3.5, после чего получим уравнение 3.7

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.7}$$

В формуле 3.7:

1. $a = x_D^2 + y_D^2$

2. $b = 2x_E x_D + 2y_E y_D$

3. $c = x_E^2 + y_E^2 - 1$

После вычисления значения по данной формуле, будет получено одно или несколько значений t, для конечного циллиандра необходимо вычислить значения z по формуле 3.2 ($z_1 = z_E + t_1 z_D, z_2 = z_E + t_2 z_D$), после чего проверить соответствуют ли полученные значения z условию $z_{min} < z < z_{max}$. Также необходимо проверить пересечения луча с верхней и нижней окружностями цилиндра, для этого можно воспользоваться формулами:

$$\begin{aligned} x^2 + y^2 = 1, z = z_{min} \\ x^2 + y^2 = 1, z = z_{max} \end{aligned} \tag{3.8}$$

Если z_1, z_2 лежат с разных сторон от $z_{\min}$, то луч проходит через ближайшую окружность цилиндра и вычислить значение t можно следующим способом:

$$t_3 = \frac{z_{\min} - z_E}{z_D}. \tag{3.9}$$

Аналогично находится и точка пересечения с дальней окружностью ($z_{\max}$) цилиндра.[7]

Уравнение конуса Конус задается уравнением 3.10

$$x^2 + y^2 = z^2. \tag{3.10}$$

После введения данного уравнения точка пересечения вычисляется аналогично точке пересечения с цилиндром. Нельзя забывать, что также как и в работе с цилиндром, для ограничения фигуры необходимо вводить условия $z_{\min} < z < z_{\max}$.

Уравнение плоскости Плоскость может определяться вектором нормали $\vec{N}$, и вершиной на плоскости Q. Таким образом точка P принадлежит плоскости, если

$$\vec{N} \cdot (\vec{P} - \vec{Q}) = 0 \tag{3.11}$$

Для нахождения пересечения необхрдимо подставить 3.1 в 3.11. После

выполнения преобразований будет получено выражение 3.12.

$$t = \frac{\vec{N} \cdot (\vec{Q} - \vec{E})}{\vec{N} \cdot \vec{D}} \tag{3.12}$$

В случае, если $t \geq 0$ тогда точка пересечения $\vec{E} + t\vec{D}$. Если $\vec{N} \cdot \vec{D} = 0$ тогда луч параллелен плоскости и точка пересечения отсутствует.[7]

Трансорфмация примитивов Заметим что в записанных нами формулах наложены ограничения на положение фигур:

1. Сфера и цилиндр были расмотрены только с единичным радиусом
2. Цилиндр и конус совмещены по оси с осью z
3. Конус имеет единичнный уклон

Для поиска пересечения необходимо найти преобразования, периводящие примитив из начала координат и данных условий в требуемые (перенос, поворот, масштабирование). После чего применить обратные преобразования к построенному лучу. Пусть объект $\hat{B}$ проходит трансформацию TRS, чтобы попасть в требуемую позицию $B = TRS\hat{B}$, в таком случае обратное преобразование луча будет выглядеть следующим образом (см. 3.13).

$$\begin{aligned} \hat{E} &= S^{-1}R^{-1}T^{-1}\vec{E} \\ \hat{D} &= S^{-1}R^{-1}\vec{D}. \end{aligned} \tag{3.13}$$

После нахождднния пересечения преобразованного луча с исходным объектом будет получено значение t, что позволяет посчитать $\vec{P} = \vec{E} + t\vec{D}$.

3.3 Перспектива

Для получения перспективы необходимо скорректировать направление каждого луча. Перед тем как получить наше изображение необходимо спроецировать его на картинную плоскость. все лучи должны исходить из одной точки и проходить через полученные соответствующие пиксели (см. 3.5).Для описания координатной системы камеры введены: вектора $-w$ - вектор направления взгляда, e - точка наблюдения, v - вектор направленный «вверх»

(англ. up vector), необходимый для построения базиса. В данном случае картинная плоскость находится на удалении d в направлении взгляда $-\mathrm{w}$, каждый луч будет иметь различное направление в зависимости от пересекаемого пикселя.

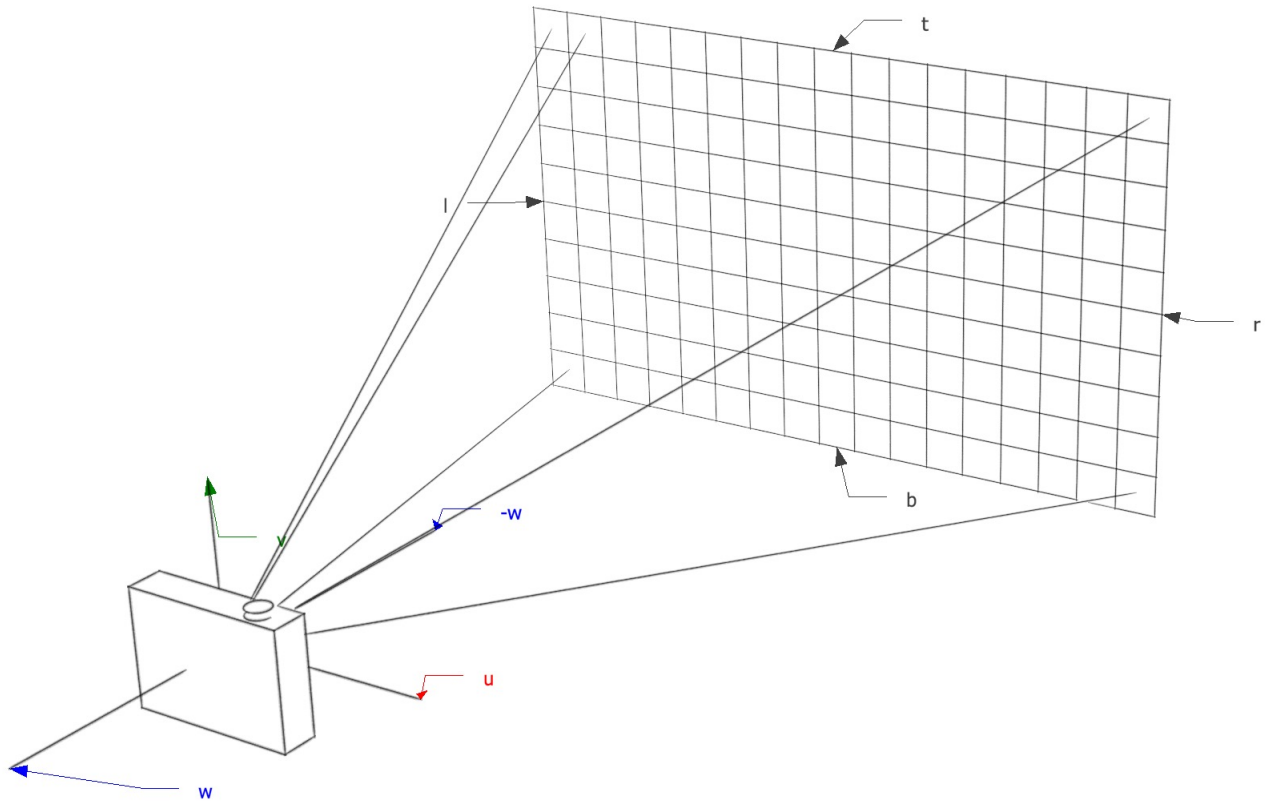

Рисунок 3.5 – Перспективный вид

Соответственно необходимо корректировать вектор направления испускаемых лучей (см. 3.1) следующим образом:

$$D = -d\mathrm{w} + u\mathrm{n} + u\mathrm{v} \tag{3.14}$$

где u, v получены из расчета луча для каждого пикселя (см. 3.15).

$$\begin{aligned} u &= l + (x + 0.5)\frac{r - l}{n_x} \\ v &= b + (y + 0.5)\frac{t - b}{n_y} \end{aligned} \tag{3.15}$$

В выражениях 3.15 символы соответственно означают:

1. l, r - позиции левого и правого краев изображения соответственно
2. t, b - позицции верхнего и нижнего краев изображения соответственно
3. x, y - координаты пикселя
4. n_x, n_y - количество пикселей по соответствующим осям изображения

После выполненных преобразований будет получено представления луча, для которого будут вычисляться пересечения с объектами.[8]

3.4 Вывод

В данном разделе был разобрана реализация выбранного алгоритма построения изображения, а также рассмотрены случаи поиска пересечений лучей к конкретным примитивам.Была затронута тема построения перспективы для выбранного алгоритма и математическое обоснование его работы.

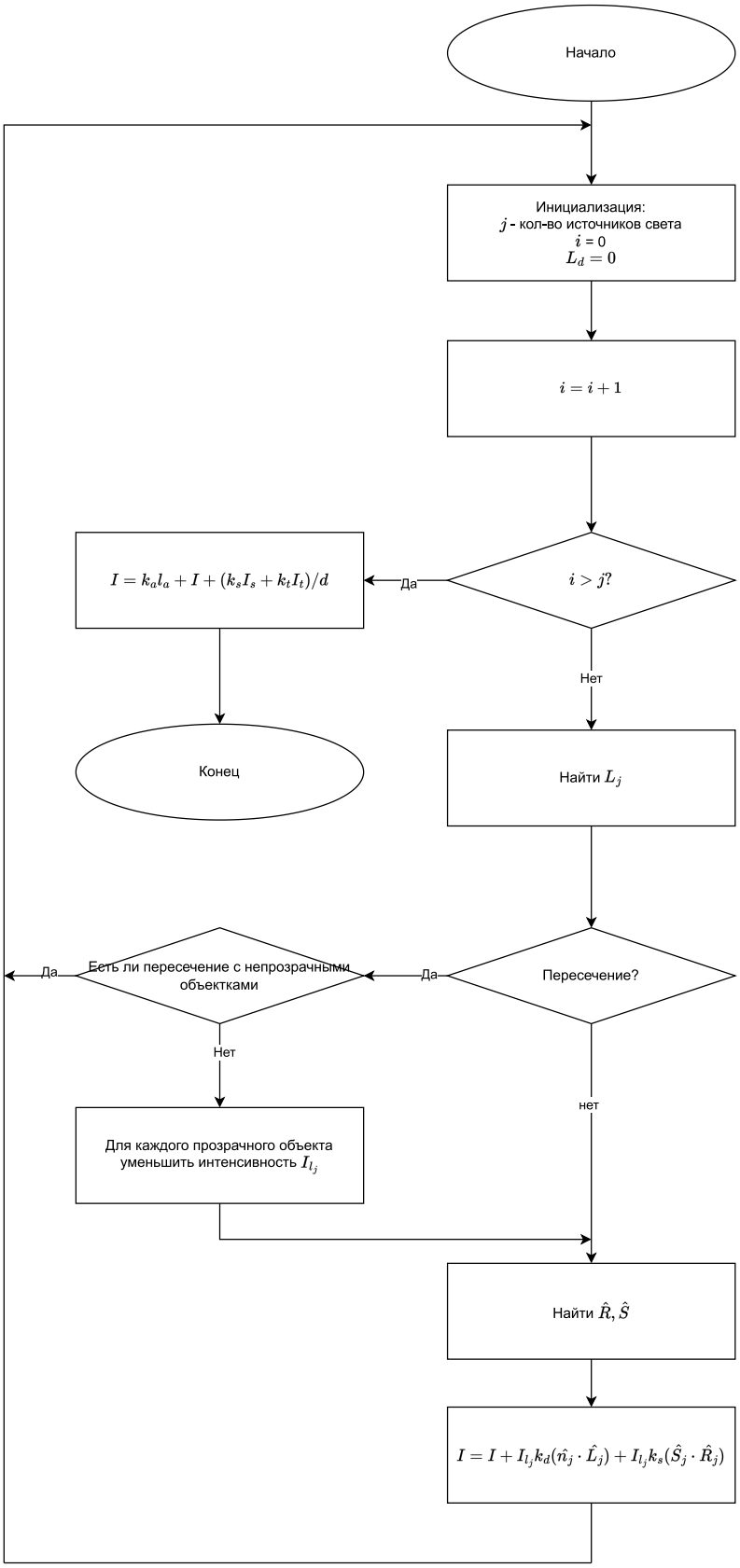

Рисунок 3.4 – Алгоритм вычисления интенсивности света для модели Уиттеда

Литература

[1] Роджерс Д. Алгоритмические основы машинной графики. - 1-е изд. - Москва: Мир, 1989. - 512 с.

[2] Модель глобального освещения с трассировкой лучей [Электронный ресурс]. – Режим доступа: https://vunivere.ru/work71759/page3 (дата обращения 15.07.23)

[3] Современное состояние методов расчета глобальной освещенности в задачах реалистичной компьютерной графики [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/sovremennoe-sostoyanie-metodov-raschyota-globalnoy-osveschyonnosti-v-zadachah-realistichnoy-kompyuternoy-grafiki/viewer (дата обращения 15.07.23)

[4] Screen Space Reflection Techniques [Электронный ресурс]. – Режим доступа: https://ourspace.uregina.ca/handle/10294/9245 (дата обращения 15.07.23)

[5] Отражение в играх. Как работают, различия и развитие технологий [Электронный ресурс]. – Режим доступа: https://clck.ru/34zZCf (дата обращения 15.07.23)

[6] Простые модели освещения [Электронный ресурс]. – Режим доступа: https://grafika.me/node/344 (дата обращения 15.07.23)

[7] Ray tracing primitives [Электронный ресурс]. – Режим доступа: https://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node2.html (дата обращения 20.07.23)

[8] Ray tracing [Электронный ресурс]. – Режим доступа: https://www.mauriciopoppe.com/notes/computer-graphics/ray-tracing/ (дата обращения 23.07.23)