



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

K KУРСОВОЙ РАБОТЕ

на тему:

«»

Студент Компьютерная графика
(Группа)

(Подпись, дата)

Компьютерная графика
(И. О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

Моделирование геометрических

2023 г.

Разин А./ИУ7-54Б Кузнецова О. В.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Аналитическая часть	5
1.1 Формализация объектов сцены	5
1.2 Анализ моделей отражения	6
1.2.1 Модель Ламберта	6
1.2.2 Модель Фонга	8
1.2.3 Выбор модели отражения	10
1.3 Анализ алгоритмов визуализации	10
1.3.1 Алгоритм трассировки лучей	12
1.3.2 Трассировка лучей в пространстве изображения	14
1.3.3 Трассировка пути	16
1.3.4 Выбор оптимального алгоритма	20
2 Конструкторская часть	21
2.1 Требования к программному обеспечению	21
2.2 Общий алгоритм построения изображения	21
2.3 Выбор типов и структур данных	22
2.4 Алгоритм обратной трассировки лучей	24
2.4.1 Нахождение пересечения с объектами сцены	25
2.5 Перспектива	29
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	31

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 [1].

Целью данной работы является разработка программного обеспечения, моделирующего отражения от геометрических тел.

Для достижения поставленной цели требуется решить следующие задачи:

1. формализовать 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 программные средства для реализации модели;
6. реализовать полученную модель;
7. создать интерфейс;
8. провести замеры времени построения кадра от количества и типа примитивов на сцене.

1 Аналитическая часть

1.1 Формализация объектов сцены

На визуализируемой сцене могут находиться следующие объекты.

1. Точечный источник света

Данный источник света излучает свет во всех направлениях, интенсивность света убывает при отдалении от источника. Источник характеризуется положением в пространстве и интенсивностью. При расчете отражений будет использоваться интенсивность источника, для расчета интенсивности пикселей. Цвет свечения будет описываться через значения RGB.

2. Сфера

Сфера описывается радиусом и координатами ее центра.

3. Куб

Для описания куба необходимо несколько параметров.

- Координаты центра.
- Размеры куба по каждой из осей.
- Угол поворота по каждой из осей.

4. Конус

Конус описывается следующими параметрами.

- Половинный угол при вершине осевого сечения конуса.
- Высота конуса.
- Позиция вершины конуса.
- Вектор высоты конуса.

5. Цилиндр

Цилиндр описывается некоторыми параметрами.

- Координаты центра одного основания цилиндра.

- Координаты центра другого основания цилиндра.
- Радиус цилиндра.

6. Камера

Камера описывается несколькими параметрами.

- Координатами своего положения.
- Вектором направления взгляда.
- «Правым» вектором пространства камеры.
- «Верхним» вектором пространства камеры.

Каждый из примитивов также должен описываться своим цветом в формате RGB, а также коэффициентами рассеянного, диффузного, зеркального отражения.

1.2 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 [2]:

1. модель Ламберта;
2. модель Фонга.

1.2.1 Модель Ламберта

В данной модели рассматривается диффузная составляющая отражения.

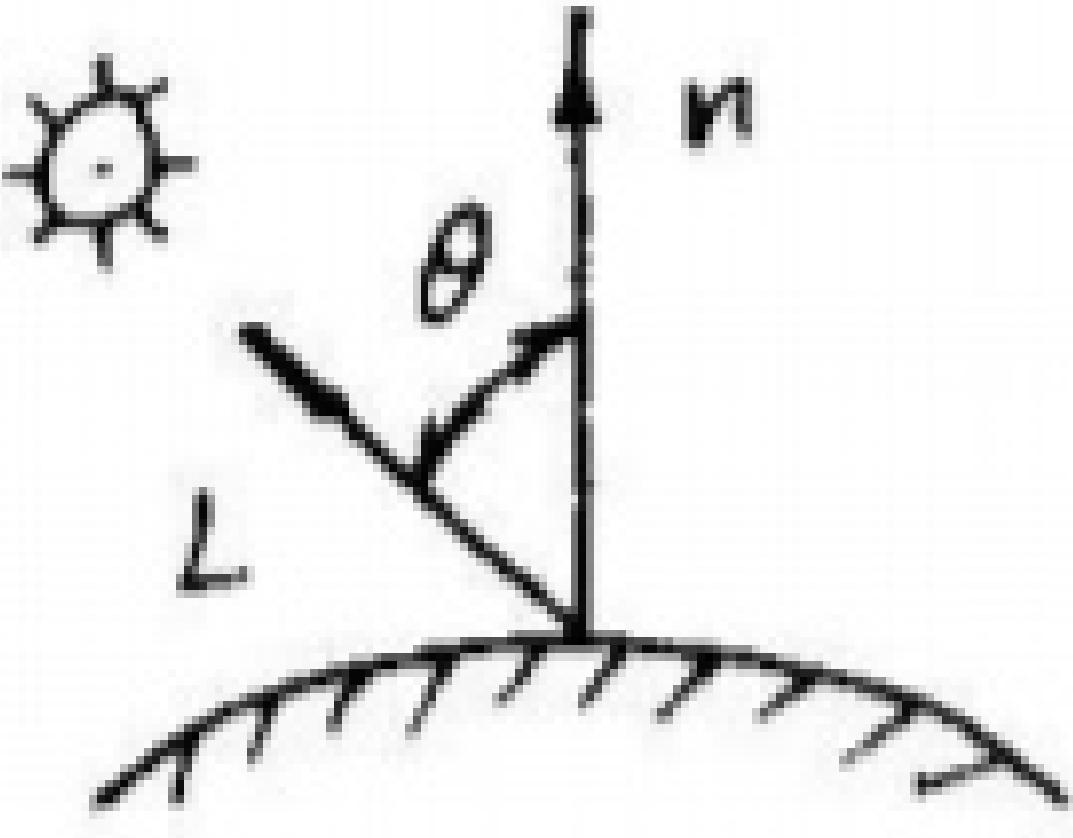


Рисунок 1.1 – Модель Ламберта

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

$$I = k_a I_a + I_l k_l \cos \theta \quad 0 \leq \theta \leq \pi/2. \quad (1.1)$$

В формуле (1.1):

1. k_a, k_d - коэффициенты рассеянного, диффузного отражения соответственно;
2. I_a, I_l - интенсивность рассеянного и диффузного отражения;
3. θ - угол между нормалью к поверхности и направлением света.

Заметим что значения приведенных коэффициентов лежат на отрезке от 0 до 1 [2].

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

$$I = k_a I_a + \frac{I_l k_l \cos \theta}{d + K}. \quad (1.2)$$

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. Таким образом K - произвольная постоянная [2].

1.2.2 Модель Фонга

Данная модель также учитывает зеркальную составляющую отражения

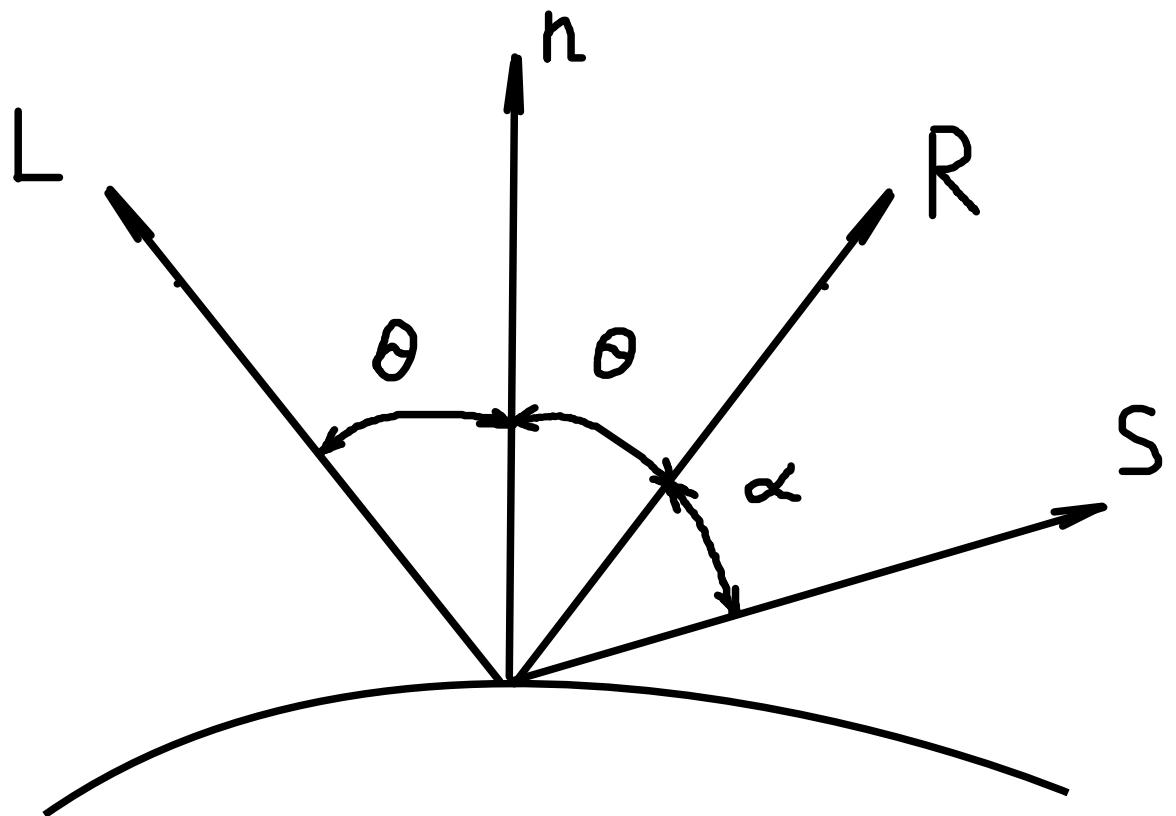


Рисунок 1.2 – Модель Фонга

Зеркальная составляющая отражения имеет следующий вид:

$$I_s = I_l \omega(i, \lambda) \cos^n \alpha. \quad (1.3)$$

В формуле (1.3) символы соответственно означают:

1. $\omega(i, \lambda)$ - кривая отражения, показывающая отношение зеркально отраженного света к падающему, как функцию угла падения i и длины 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 [2].

Таким образом формула принимает следующий вид:

$$I = k_a I_a + k_d I_l (\hat{n} \cdot \hat{L}) + k_s I_l (\hat{S} \cdot \hat{R})^n. \quad (1.4)$$

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 $\hat{\cdot}$ означает, что данный вектор нормированный [2].

1.2.3 Выбор модели отражения

Данные модели описывают простую (локальную) модель освещения, то есть модель освещения в которой учитывается свет, попадающий в рассматриваемую точку только 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 1.3.1. Заметим, что глобальная модель в случае алгоритма Уиттеда освещения использует идею модели Фонга (см. 1.5).

1.3 Анализ алгоритмов визуализации

Приведен пример отражения лучей:

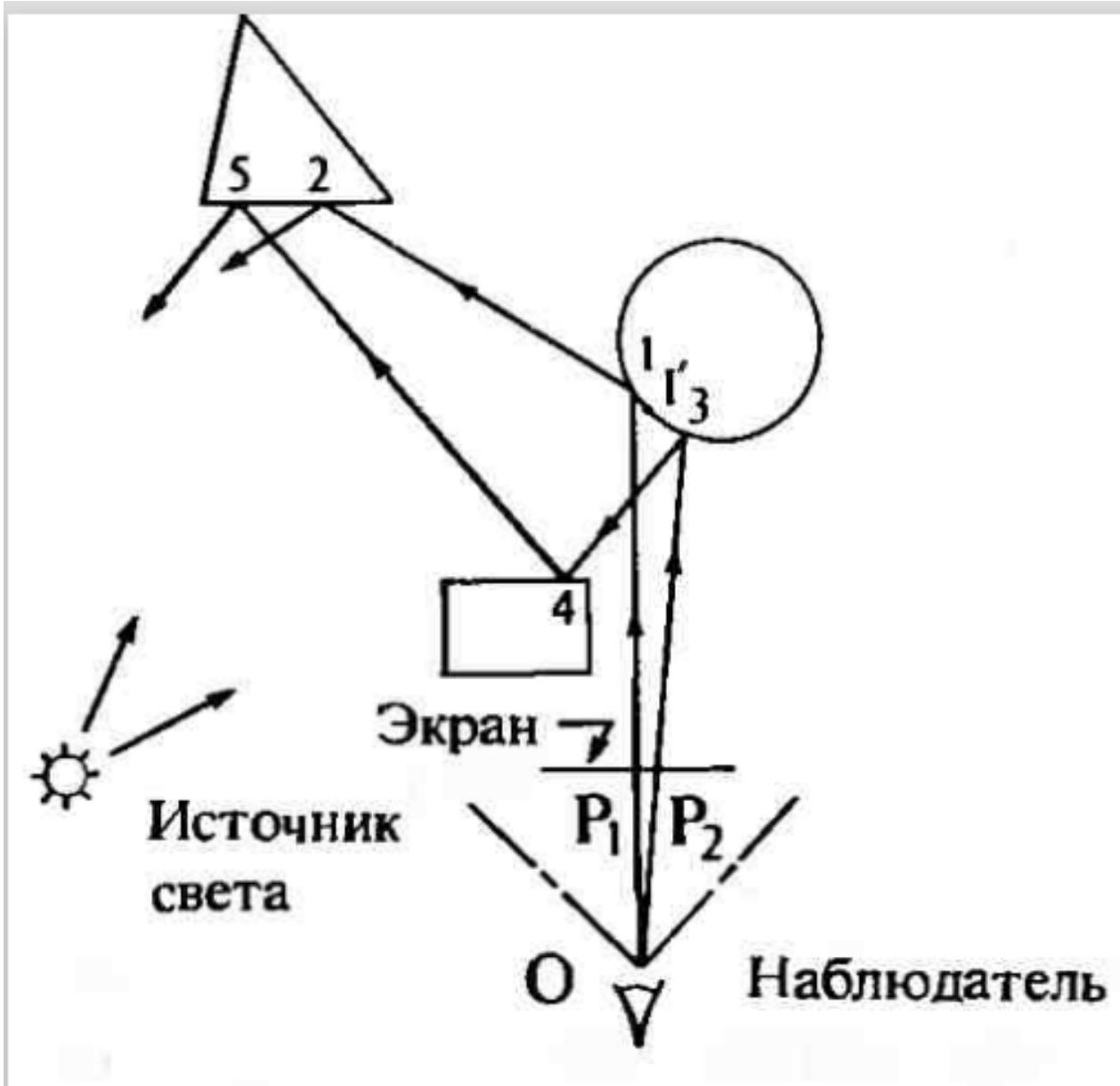


Рисунок 1.3 – Пример трассировки луча

На рисунке 1.3 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 [2].

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

1. трассировка световых лучей (англ. Ray tracing);
2. трассировка пути (англ. Path tracing);
3. трассировка лучей в пространстве изображения (англ. Screen reflections).

1.3.1 Алгоритм трассировки лучей

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. 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 данном случае лучи стоит проводить через центры пикселей изображения, считается, что наблюдатель находится на бесконечности, из-за чего все лучи параллельны оси OZ [modern_ray_tracing; 2].

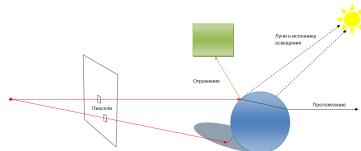


Рисунок 1.4 – Пример трассировки луча

Первые работы принадлежат Уиттеду и Кэю. Алгоритм Уиттеда более общий и часто используется. Уиттед пользуется моделью, в которой диффузная и зеркальная составляющие отражения рассчитываются подобно локальной модели (пример приведен на картинке 1.2) [2].



Рисунок 1.5 – Расчет зеркального отражения луча в алгоритме Уиттеда

На рисунке 1.5 луч \mathbf{V} падает на поверхность в точку \mathbf{Q} , после чего отражается в направлении \mathbf{r} и преломляется в направлении \mathbf{p} . В данном случае:

1. I_t - интенсивность света, проходящего по преломленному лучу \mathbf{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 \mathbf{I} выражается формулой:

$$I = k_a I_a + k_d \sum_j I_{l_j} (\hat{n} \cdot \hat{L}_j) + k_s \sum_j I_{l_j} (\hat{S} \cdot \hat{R}_j)^n + k_s I_s + k_t I_t. \quad (1.5)$$

В формуле (1.5) 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 $\hat{\cdot}$ 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. [2]

Достоинства алгоритма:

1. создание реалистичных изображений. [**modern _ ray _ tracing; SSR; 2**];
2. возможность наблюдения физических явлений, так как алгоритм симулирует поведение света в реальной жизни [**modern _ ray _ tracing; SSR; 2**].

Недостатки алгоритма:

1. долгое время получения кадра. [**modern _ ray _ tracing; SSR; 2**];
2. количество требуемой памяти, так как в памяти необходимо хранить все отраженные и преломленные лучи, полученные при предыдущих расчетах [**modern _ ray _ tracing; SSR; 2**].

1.3.2 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 алгоритма трассировки лучей в пространстве изображения (англ. Screen-space reflections, SSR),

используется информация о имеющихся объектах 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. [SSR]

Перед началом алгоритма требуется информация для каждого пикселя:

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

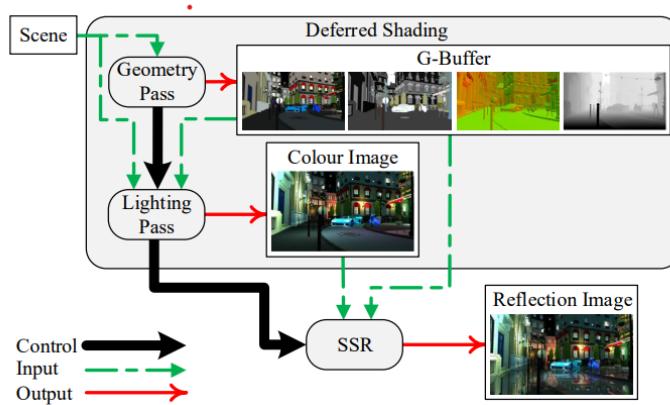


Рисунок 1.6 – Поток данных при использовании SSR

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

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

На картинке 1.6 используется понятие **G-buffer**, этот 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 nearest 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 некорректным, как, например, на картинке 1.7 [SSR; **reflexion_types**].



Рисунок 1.7 – Некорректный расчет отражений при использовании SSR

Достоинства алгоритма:

1. меньшее время получения кадра, чем при использовании алгоритма трассировки лучей [SSR].

Недостатки алгоритма:

1. При наличии невидимых для наблюдателя частей объектов в отражении изображение будет некорректным [SSR];
2. Искажение геометрии при генерации изображений. [SSR]

1.3.3 Трассировка пути

В обратной трассировке лучей считается, что отражение луча идеально, и через каждый пиксель экрана трассируется один луч, однако ввиду данных допущений изображения не являются полностью физически корректными [ray_path_tracing]. Данный алгоритм вычисляет интенсивность освещенности с помощью метода Монте-Карло, так как отражения лучей имеют

случайный характер [**monte_carlo**]. Для реализации метода Монте Карло для каждого пикселя генерируется несколько лучей, после чего результирующее значение их интенсивностей усредняется [**path_tracing_def**].

Каждый раз, когда луч пересекается с поверхностью, выпускается теневой и случайный отраженный луч. Теневой луч — луч, с помощью которого учитывается прямое освещение (свет излучаемый источником света) для данной точки выборки. Данный луч проводится из точки пересечения к объекту, являющимся источником света в данной сцене. В случае если данный луч пересекает объект сцены, то в данной точке интенсивность прямого света равна 0. Однако в таком случае объект может получить свет от других объектов сцены с помощью отраженных лучей (непрямое освещение) при пересечении трассируемого луча с объектами, в случае если точка пересечения освещена прямым освещением, интенсивность данного освещения будет учитываться при расчете интенсивности света исходного луча. Таким образом при каждом столкновении луча с примитивом учитывается прямое и непрямое освещение, которое будет учитываться при расчете интенсивности отраженного луча. Из-за того что направление лучей рассчитывается случайно, при малом количестве генерируемых лучей на пиксель лучи не пересекут исходные объекты, что образует шумы, которые представлены на картинке 1.8 [**path_tracing; ray_path_tracing**]. Для избежания шумов необходимо увеличить число генерируемых лучей на 1 пиксель. Из-за необходимости в генерации большого количества лучей данный алгоритм трудозатратнее предыдущих и не позволяет наблюдать сцену в реально времени из-за шумов [**path_tracing**]. Трассировка пути позволяет получить более реалистичное изображение, на картинках 1.9 и 1.10 представлены два изображения, полученные с помощью трассировки лучей и трассировки пусти соответственно [**cyber_trce**].

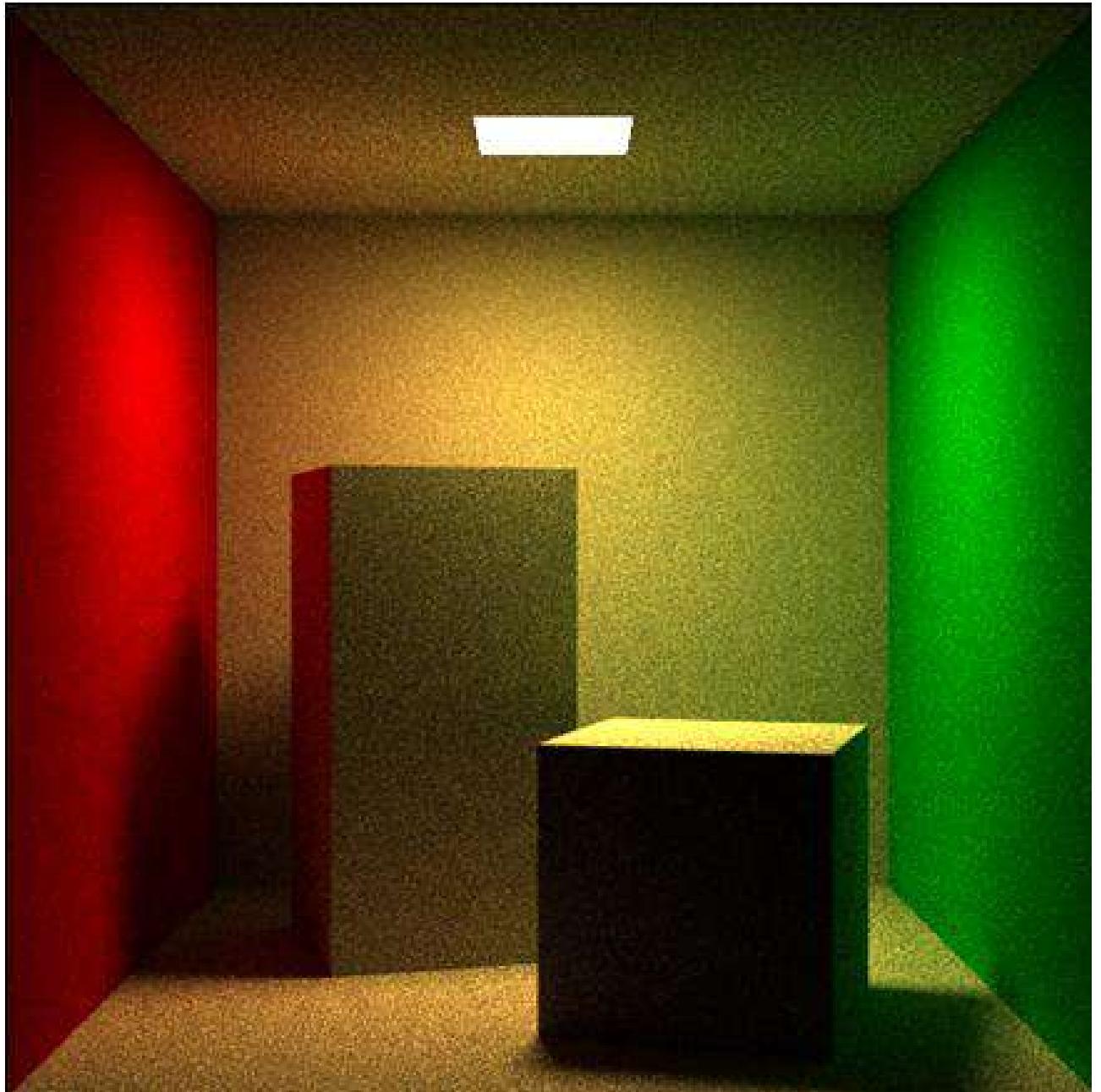


Рисунок 1.8 – Пример шума при использовании трассировки пути



Рисунок 1.9 – Пример кадра комнаты в игре Cyberpunk 2077 с помощью трассировки лучей



Рисунок 1.10 – Пример кадра комнаты в игре Cyberpunk 2077 с помощью трассировки пути

1.3.4 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 для выбранных примитивов поиск пересечений возможен аналитически, то при использовании данного алгоритма вычислительная сложность не будет являться критически высокой [3]. Алгоритм трассировки в пространстве изображения позволяет генерировать неправдоподобные отражения, кубические карты используются при возможности создания статических изображений, эти ограничения не позволяют их использовать при генерации реалистичных изображений.

Выводы

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

1. **Алгоритм создания отражений** - Алгоритм обратной трассировки лучей.
2. **Модель отражения** - Модель отражения 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.

2 Конструкторская часть

2.1 Требования к программному обеспечению

Программа должна предоставлять следующие возможности:

1. Задание положения источника света
2. Изменения положения камеры и направления ее взгляда
3. Визуализация трехмерных примитивов: шар, куб, цилиндр, конус
4. Визуализация отражений, света и теней в соответствии с параметрами примитивов
5. Поддержка перемещения и поворота заданных примитивов, а также изменения их цвета
6. Изменения интенсивности источника света

2.2 Общий алгоритм построения изображения

Рассмотрим алгоритм построения кадра, пока что не рассматривая конкретную реализацию алгоритма трассировки лучей на картинке 2.1.



Рисунок 2.1 – Общий алгоритм построения кадра

2.3 Выбор типов и структур данных

Для формирования общего алгоритма синтеза изображения, необходимо ввести определения соответствующих структур данных.

1. Структура источника света описывается следующими полями:

- position — вектор положения источника света в пространстве;
 - intensivity — интенсивность источника света по каждой составляющей RGB.
2. Структура камеры описывается следующими полями:
- position — вектор положения камеры в пространстве;
 - view — вектор направления взгляда камеры;
 - up — текущий «верхний» вектор камеры;
 - right — текущий «правый» вектор камеры.
3. Структура материала объекта описывается следующими полями:
- Color — цвет данного материала;
 - kA — коэффициент рассеянного освещения;
 - kD — коэффициент диффузного освещения;
 - kS — коэффициент спектрального освещения.
4. Структура луча описывается следующими полями:
- position — точка, лежащая на данном луче;
 - direction — вектор задающий направление луча;
 - t — вещественное число, задающее точку на данном луче.
5. Также вводится структура пересечения луча с примитивом:
- t — вещественное число, задающее точку на трассируемом луче;
 - point — координаты точки пересечения луча с примитивом;
 - normal — нормаль данного примитива, проведенная из точки пересечения;
 - material — материал пересеченного примитива;
 - tracedRay — трассируемый луч.
6. Сцена представляет собой массив с заранее определенным максимальным числом примитивов.

7. Примитивы задаются аналогично их формальному описанию, приведенному в 1.1.

2.4 Алгоритм обратной трассировки лучей

Алгоритм трассировки лучей для 1 пикселя приведен на картинке ??.
Входными данными для него являются: описания примитивов, интенсивность источника света, положение источника света, максимальное число отражений луча.

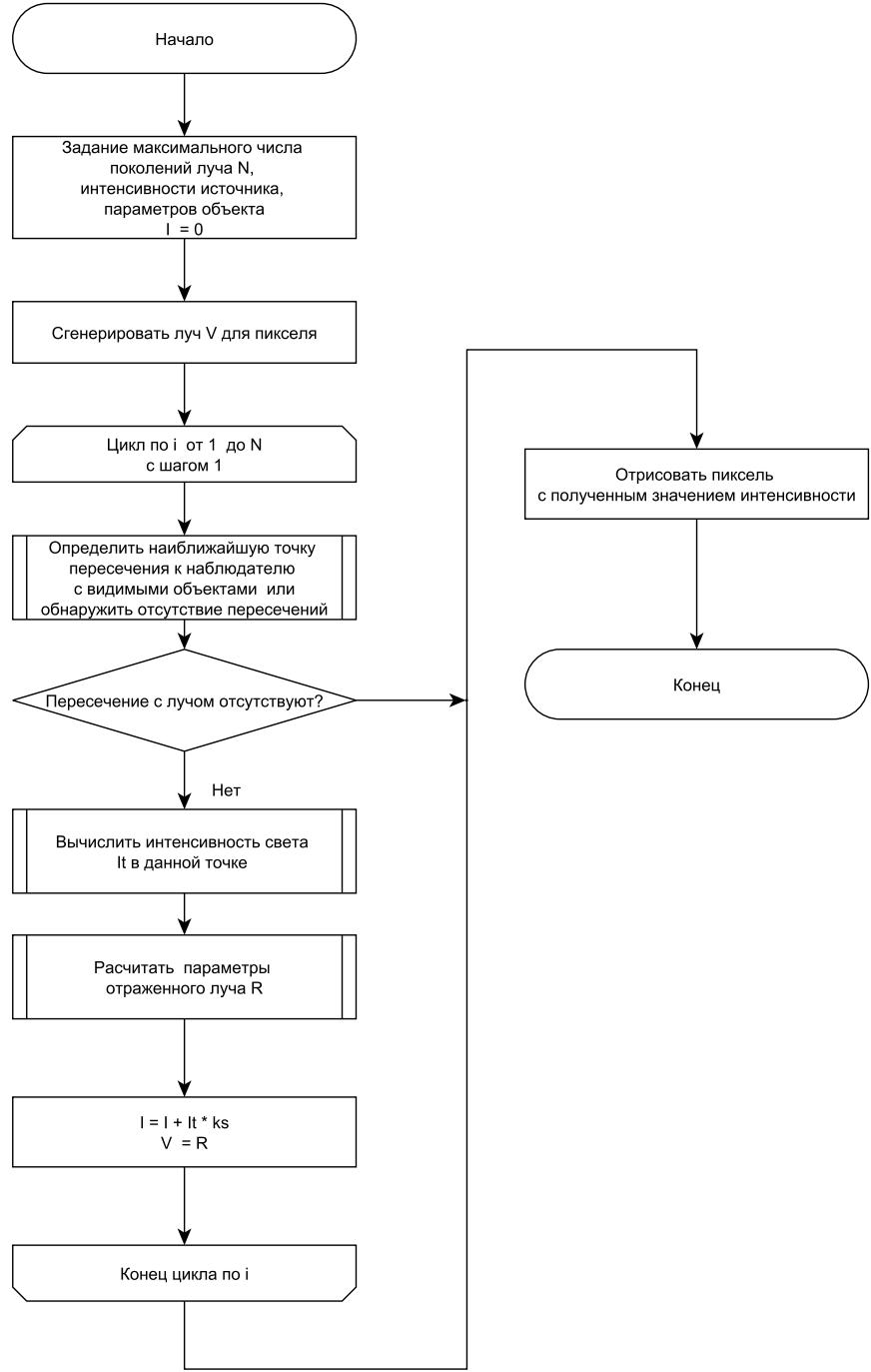


Рисунок 2.2 – Алгоритм трассировки лучей для 1 пикселя

Теоретически свет может отражаться бесконечно, введение ограничения на максимальное число поколений луча позволит ограничить время построения изображения, получая реалистичное поведение света.

2.4.1 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.

Необходимо ввести уравнение самого луча (2.1).

$$P(t) = \vec{E} + t\vec{D}, t \geq 0 \quad (2.1)$$

Также уравнение 2.1 имеет запись

$$\begin{aligned} x(t) &= x_E + t x_D \\ y(t) &= y_E + t y_D \\ z(t) &= z_E + t z_D. \end{aligned} \quad (2.2)$$

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 $t \geq 0$, точка на луче находится после точки обзора, иначе - за. Таким образом для поиска nearestнейшей точки пересечения, необходимо найти наименьшее неотрицательное значение t . [2; 3]

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

$$\vec{P} \cdot \vec{P} = 1 \quad (2.3)$$

Для получения условия пересечений достаточно подставить уравнение луча из 2.1 в 2.3 и решить полученное уравнение относительно t .

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (2.4)$$

После проведенных преобразований будет получена формула 2.4, 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 2.4 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.[3]

Уравнение цилиндра Уравнение 2.5 задает бесконечный цилиндр.

$$x^2 + y^2 = 1 \quad (2.5)$$

Для ограничения цилиндра необходимо ввести требования к значению z цилиндра:

$$x^2 + y^2 = 1, z_{\min} < z < z_{\max} \quad (2.6)$$

Для нахождения пересечений подставим 2.2 в 2.5, после чего получим уравнение 2.7

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (2.7)$$

В формуле 2.7:

1. $a = x_D^2 + y_D^2$
2. $b = 2x_Ex_D + 2y_Ey_D$
3. $c = x_E^2 + y_E^2 - 1$

После вычисления значения по данной формуле, будет получено одно или несколько значений t , для конечного цилиндра необходимо вычислить значения z по формуле 2.2 ($z_1 = z_E + t_1 z_D, z_2 = z_E + t_2 z_D$), 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:

$$\begin{aligned} x^2 + y^2 = 1, z = z_{\min} \\ x^2 + y^2 = 1, z = z_{\max}. \end{aligned} \quad (2.8)$$

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:

$$t_3 = \frac{z_{\min} - z_E}{z_D}. \quad (2.9)$$

Аналогично находится и точка пересечения с дальней окружностью (z_{\max}) цилиндра.[3]

Уравнение конуса

Конус задается уравнением 2.10

$$x^2 + y^2 = z^2. \quad (2.10)$$

После введения данного уравнения точка пересечения вычисляется аналогично точке пересечения с цилиндром. Нельзя забывать, что так же 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

$$\vec{N} \cdot (\vec{P} - \vec{Q}) = 0. \quad (2.11)$$

Для нахождения пересечения необходимо подставить 2.1 в 2.11. После выполнения преобразований будет получено выражение 2.12.

$$t = \frac{\vec{N} \cdot (\vec{Q} - \vec{E})}{\vec{N} \cdot \vec{D}} \quad (2.12)$$

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.[3]

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

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

Для поиска пересечения необходимо найти преобразования, переводящие 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. 2.13).

$$\begin{aligned} \hat{E} &= S^{-1}R^{-1}T^{-1}\vec{E} \\ \hat{D} &= S^{-1}R^{-1}\vec{D}. \end{aligned} \quad (2.13)$$

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

2.5 Перспектива

Для получения перспективы необходимо скорректировать направление каждого луча. Перед тем как получить наше изображение необходимо спроектировать его на картинную плоскость. все лучи должны исходить из одной точки и проходить через полученные соответствующие пиксели (см. 2.3). 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 $-w$, каждый луч будет иметь различное направление в зависимости от пересекаемого пикселя.

Рисунок 2.3 – Перспективный вид

Соответственно необходимо корректировать вектор направления испускаемых лучей (см. 2.1) следующим образом:

$$D = -dw + un + uv \quad (2.14)$$

где u, v получены из расчета луча для каждого пикселя (см. 2.15).

$$\begin{aligned} u &= l + (x + 0.5) \frac{r - l}{n_x} \\ v &= b + (y + 0.5) \frac{t - b}{n_y}. \end{aligned} \quad (2.15)$$

В выражениях 2.15 символы соответственно означают:

1. l, r - Позиции левого и правого краев изображения соответственно
2. t, b - Позиции 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.[4]

Выводы

В данном разделе была 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. Были описаны основные структуры данных, необходимы для решения поставленных задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Компьютерная графика и сферы ее применения. — Режим доступа: <https://moluch.ru/archive/294/66793/?ysclid=1plohpfg4b711269405> (дата обращения 01.12.23).
2. Роджерс Д. Алгоритмические основы машинной графики //. — Издательство «Мир». Редакция литературы по математическим наукам, 1989. — С. 512.
3. Простые модели освещения [Электронный ресурс]. — Режим доступа: <https://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node2.html> (дата обращения 20.07.23).
4. Ray tracing [Электронный ресурс]. — Режим доступа: <https://www.mauriciopoppe.com/notes/computer-graphics/ray-tracing/> (дата обращения 23.07.23).