Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н. Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА«Программное обеспечение ЭВМ и информационные технологии»

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

Визуализации сфер методом трассировки лучей.

Студент: А. К. Чаушев И.О. Фамилия
(Подпись, дата)

Руководитель проекта: Т. Н. Романова И.О. Фамилия
(Подпись, дата)

Москва 2020

Техническое задание

Разработать программу визуализации сфер методом трассировки лучей. В программе должны быть учтены освещенность, тип поверхности объектов (матовые, металлические и стеклянные), должны быть визуализированы тени и блики.

Содержание

Введение

Одной из главных проблем синтеза реалистичных изображений является значительный объем вычислений и, как результат, большое время работы алгоритма. Трассировка лучей участвует в создание компьютерных игр, симуляторов, анимационных фильмов, проектирование оптических систем [6] – это далеко не полный список тех областей, в которых получение реалистичного изображения является первостепенной задачей. Для того, чтобы получить максимально похожее на реальное изображение, можно использовать методы визуализации, в основе которых лежат принципы, аналогичные реальным физическим процессам. Одним из таких методов и является трассировка лучей. Стоит отметить, что на сегодняшний день он считается наиболее универсальным методом создания фотореалистичных изображений [5]. Широкий спектр применения метода трассировки лучей, простое объяснение его принципов, а также возможность отрисовки максимально реалистичных изображений – основные причины, по которым этот метод был выбран в качестве темы курсового проекта. Целью данного курсового проекта является изучение возможностей метода трассировки лучей и создание программного продукта, в котором возможна визуализация плоскостей и сфер.

Для достижения поставленной цели необходимо решить следующие задачи:

1) формализировать решаемую задачу;

2) рассмотреть и проанализировать существующие алгоритмы решения задачи, выбрать подходящий;

3) изучить физические принципы и законы, по которым строится выбранный метод визуализации;

4) изучить математическую модель, описывающую работу метода.

5) выбрать наиболее подходящую реализацию выбранного алгоритма;

6) разработать архитектуру и структуру программы;

7) разработать интерфейс;

8) выбрать средства программной реализации;

9) разработать программные модули;

1 Аналитический раздел

В данном разделе будет дана формализация задачи курсового проекта, а также будут рассмотрены алгоритмы метода трассировки лучей.

1.1 Анализ алгоритмов метода трассировки лучей

Проведем анализ предметной области, в соответствии с которым выберем подходящий ход решения поставленной задачи.

Идея метода трассировки лучей заключается в том, что наблюдатель видит любой объект посредством испускаемого некоторым источником света, который падает на этот объект и затем каким-то путем доходит до наблюдателя [7]. Таким образом, проследив все лучи, исходящие из источника света и достигшие наблюдателя, можно синтезировать реалистичное изображение.

Все алгоритмы метода трассировки лучей работают в пространстве изображения. Данные алгоритмы предполагают привязку к системе координат экрана или картинной плоскости, на которую производится проецирование изображаемых объектов. Объем требуемых вычислений значительно меньше, чем у алгоритмов, работающих в объектном пространстве, и зависит от разрешающей способности экрана и количества объектов на сцене.

Для выбора, наиболее подходящего для достижения поставленных задач алгоритма, необходимо осуществить краткий обзор существующих, отобрать критерии для сравнения и выявить алгоритм, который удовлетворяет всем или большинству критериев.

Исходя из цели данной работы, необходимо выбрать алгоритм, который сможет поддерживать глобальную модель освещения, синтезировать реалистичное изображение за приемлемое для пользователя время, а также использовать по возможности минимальное количество ресурсов.

1.1.1 Алгоритм трассировки лучей

В алгоритме трассировке лучей, лучи света обрабатываются в обратном направлении, луч испускается из камеры вглубь сцены через пиксель окна вывода. Каждый луч, исходящий от наблюдателя (камеры), проходит через центр пикселя на растре до сцены. Траектория каждого луча отслеживается, чтобы определить, какие именно объекты сцены (если таковые существуют) пересекаются данным лучом. Если луч пересекает объект, то определяются все возможные точки пересечения луча и объекта. Эти пересечения упорядочиваются по глубине. Пересечение с минимальным значением z представляет видимую поверхность для данного пикселя [7]. Характеристики этого объекта используются для определения цвета пикселя. Далее из найденной точки пересечения выпускаются новые лучи: до источников освещения (чтобы определить не является ли поверхность объекта в текущей точке теневой), отраженный и преломленные лучи. Таким образом, алгоритм рекурсивно вычисляет интенсивность света в точке пересечения, учитывая влияние соседних объектов и всех источников освещения.

1.1.2 Алгоритм трассировки пути

Алгоритм трассировки пути очень похож на алгоритм трассировки лучей. При трассировке пути из найденной точки пересечения кроме луча до источника света, выпускается новый луч в случайном направлении. Этот луч трассируется до тех пор, пока он не пересечется с источником света, что может и не случится. При трассировке пути путь луча может пересечься с множеством диффузных поверхностей до того, как пересечься с источником света [3], результаты трассировки которых потом сводятся к среднему значению. В зависимости от количества выпущенных случайных лучей будет меняться

уровень так называемого шума изображения: чем больше выпущено лучей, тем меньше шума. Такой алгоритм требует гораздо больше ресурсов, чем алгоритм трассировки лучей.

1.1.3 Алгоритм бросания лучей

Алгоритм бросания лучей может быть представлен как сокращенная и существенно более быстрая версия алгоритма трассировки лучей. Метод бросания лучей не вычисляет новые тангенсы лучей света, которые возникнут после того, когда луч, который проецируется от глаза к источнику света, пересекается с поверхностью. Эта особенность делает невозможным точный рендеринг отражений, преломлений и естественной проекции теней с помощью бросания лучей. В результате синтезируемое изображение теряет реалистичность, так как глобальная модель освещения не реализована. Высокая скорость вычисления сделала рейкастинг удобным методом рендеринга в ранних компьютерных играх с трёхмерной графикой реального времени [2].

1.2 Сравнение визуальных характеристик изображений, полученных разными алгоритмами

На рисунке 1 - 3 представлены изображения, полученные в результате работы алгоритма трассировки лучей, трассировки пути и алгоритма бросания лучей.

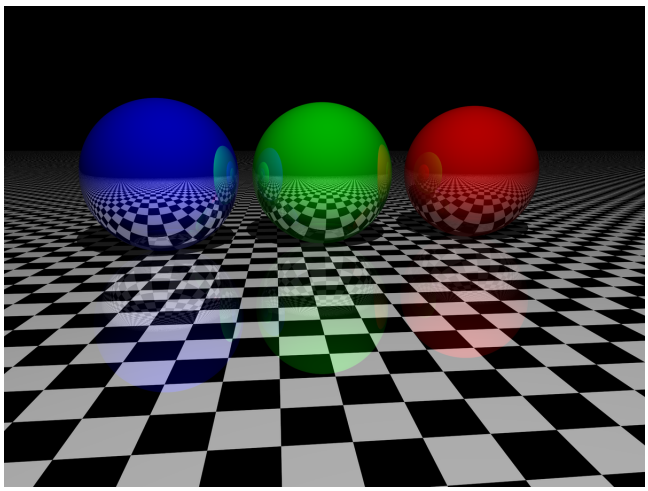
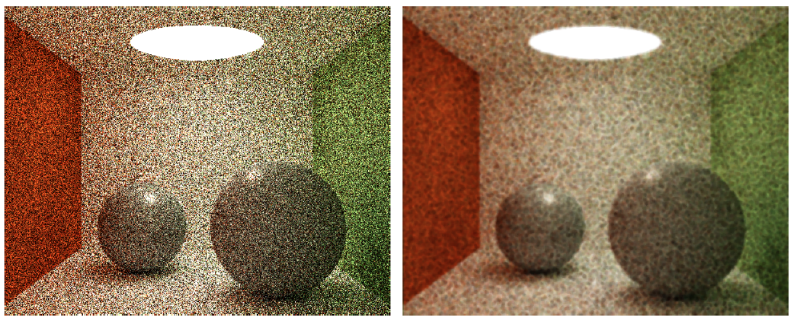
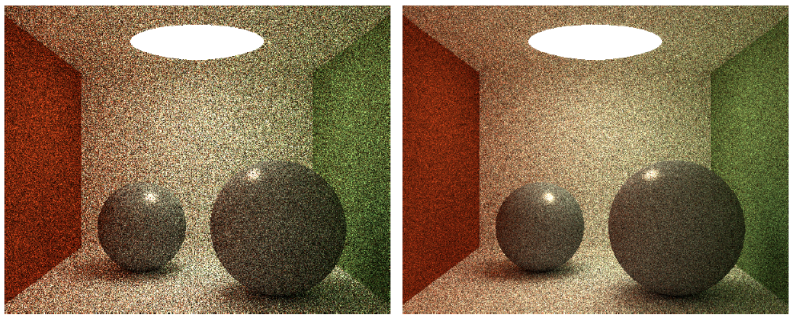


Рисунок 1 – Изображение полученное методом трассировкой лучей.



(a) Under-sampled. 10 samples / 14 seconds

(b) Smoothed Image. 10 samples / 15 seconds



(c) 20 samples / 28 seconds

(d) 50 samples / 67 seconds

Рисунок 2 – Изображении полученное методом трассировки путей.

Рисунок 3 – Изображение, полученное алгоритмом бросанию лучей.

1.3 Вывод

Как видно из рисунка 1, трассировка лучей дает реалистичности изображения. Однако, на рисунке 2 показано, как меняется качество изображения, получаемого трассировкой пути, в зависимости от количества случайных лучей. На рисунке 3 приведен пример полученного алгоритмом бросания лучей изображения. Как видно из рисунка, реалистичность получаемого изображения намного хуже.

1.4 Выбор алгоритма

Определим следующие критерии выбора алгоритма: визуальная реалистичность изображения и время работы алгоритма. Поскольку основным критериям качества синтезируемого изображения в данной работе будет рассматриваться реалистичность, то алгоритм бросания лучей определим как неподходящий. Для достижения необходимой реалистичности, время выполнения алгоритма трассировки пути значительно выше, чем время выполнения классического алгоритма трассировки лучей (в каждой точке пересечения при трассировке лучей испускается 2 луча, при трассировке пути (в зависимости от сцены) от 10 000). Таким образом, алгоритм трассировки лучей является наиболее подходящим для решения поставленных в этой работе задач.

1.5 Анализ модели освещения

Модель освещения предназначена для того, чтобы рассчитать интенсивность отраженного к наблюдателю света в каждой точке (пикселе) изображения [1]. Для получения фотореалистичного изображения, необходимо использовать глобальную модель освещения - модель, которая учитывает все имеющиеся источники освещения. Источниками энергии могут быть не толь-

ко первичные источники света, но и другие отражающие объекты. Отражение может быть как зеркальным, так и диффузным. Моделирование расчёта освещения объекта обычно представляет из себя расчёт трёх составляющих: фоновой, рассеянной (диффузной) и бликовой.

1.5.1 Модель освещения Фонгу

Модель отражения Фонга является довольно популярной и широко используемой в компьютерной графике. На рисунке 4 представлены три компонента и результат их сложения.

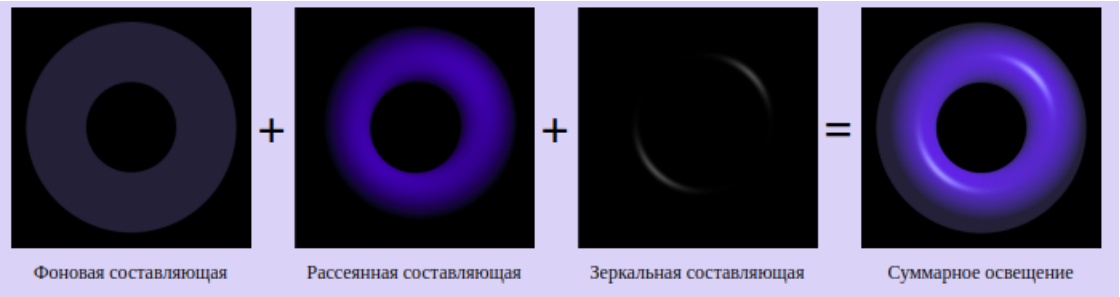


Рисунок 4 – Модель отражения Фонгу.

В этой модели анализируются только световые лучи, испускаемые светоизлучающими поверхностями - источниками, и их взаимодействие с поверхностями сцены, не учитываются способности материала излучать свет, затухания света с расстоянием [10].

1.5.2 Модель освещения Кука-Торренса

Модель Кука-Торренса – отличная модель для зеркальных бликов на поверхностях с микро гранями. На рисунке 5 представлены результаты моделью Кука-Торренса.

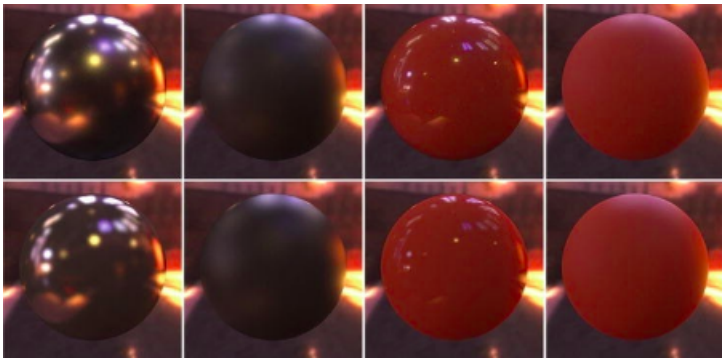


Рисунок 5 – Модель отражения Фонгу.

Так как эта модель используется для расчета отраженного света, то рассеянный свет вычисляется по классической формуле Ламберта, в которой освещенность точки зависит только от угла между нормалью к поверхности в данной точки, и положением источника света. Вычисляется как скалярное произведение нормали и нормализованного положения источника света:

$$K_d = N \cdot L \tag{1}$$

Количество отраженного света зависит от трех факторов [9]:

1) коэффициент Френеля (F);

2) геометрическая составляющая, учитывающая самозатенение (G);

3) компонент, учитывающий шероховатость поверхности (D);

Общая формула для вычисления отраженного света такова:

$$K = \frac{F \cdot G \cdot D}{(\vec{V} \cdot \vec{N}) \cdot (\vec{L} \cdot \vec{N})} \tag{2}$$

1.6 Постановка задачи

Входные данные – это характеристики объектов сцены. Объектами сцены в данной работе являются геометрические объекты и источники света. Ис-

точник света задается положением в трехмерном пространстве, цветом и интенсивностью. Один геометрический объект задается его положением в трехмерном пространстве, размерами и качественными характеристиками, такими как цвет и материал. Далее в таблице 1 представлены правила, по которым задаются геометрические объекты.

Таблица 1 – Правила определения геометрических объектов сцены

Название	Положение	Параметры
Сфера	Задается его центром.	Задается его радиусом (R) и материалом.
Плоскость	Задается при помощи его центром и вектором нормали.	Задается матетриал.
Источник света	Задается его центром и вектором направлений.	Задается интенсивность и его тип.

Для определения цвета геометрического объекта и источника была использована цветовая модель RGB [4].

Ход решения задачи – определение всех характеристик объектов сцены, выполнение инструкций выбранного алгоритма метода трассировки лучей, вывод результата на экран.

Результат решения задачи – визуализированные на экране заданные геометрические объекты с учетом заданных источников освещения. При этом синтезированное изображение должно обеспечивать формирование у наблюдателя ощущения реалистичности изображенных объектов.

2 Конструкторский раздел

В данном разделе будет подробно описан выбранный алгоритм, будет описана его математическая модель. Также будут представлены схемы алгоритма и представлены основные структуры данных.

2.1 Описание выбранного алгоритма

На рисунках 6 - 7 представлена показана схема работы функции трассирования луча

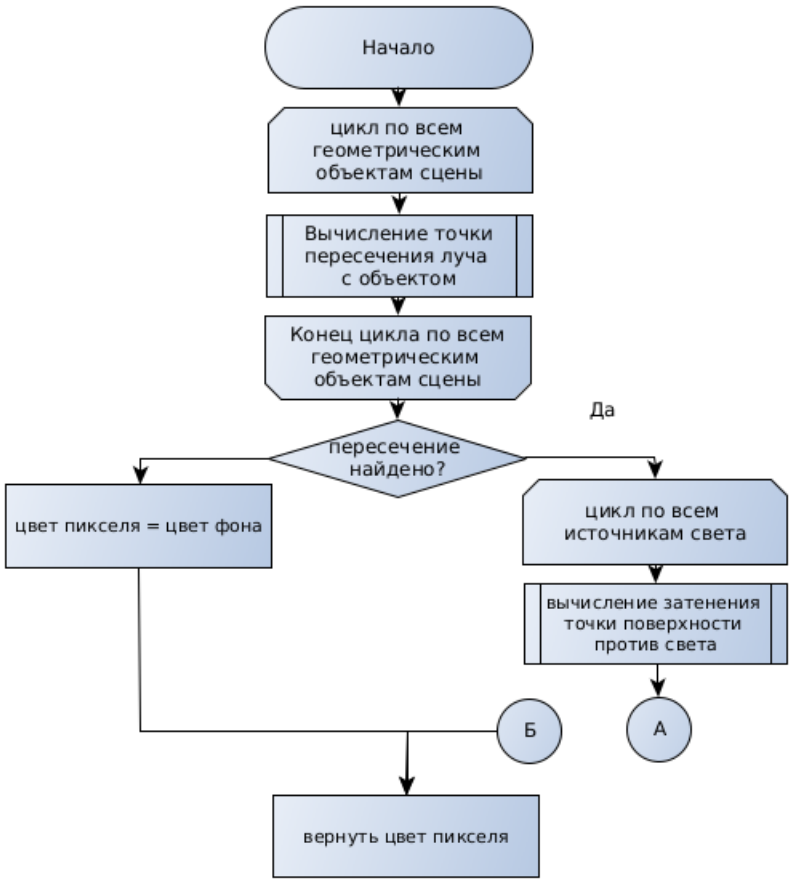


Рисунок 6 – Схема алгоритма трассировки луча (часть 1)

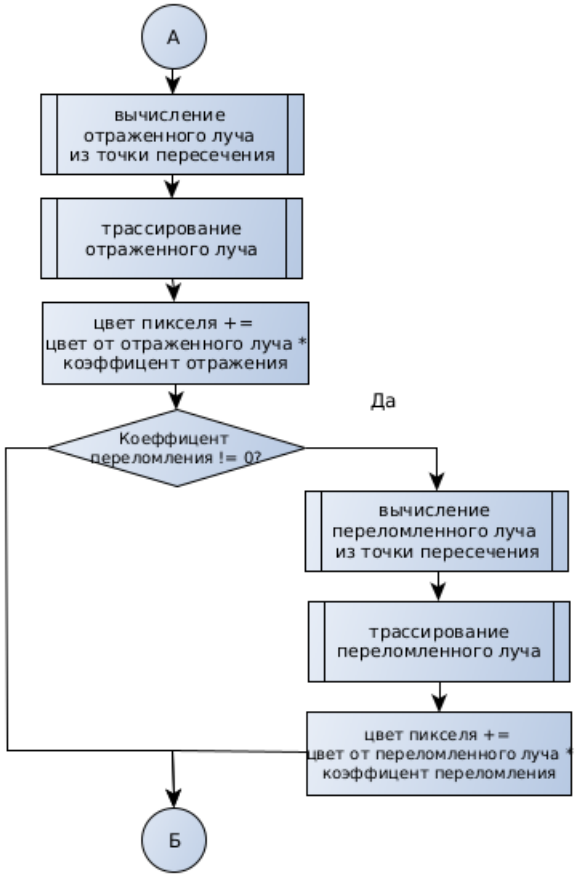


Рисунок 7 – Схема алгоритма трассировки луча (часть 2)

Как видно из схемы этой части алгоритма, функция трассировки луча вызывается рекурсивно для отраженного и преломленного луча. Для изначального луча (от наблюдателя) задается глубина рекурсии. Далее в каждом уровне рекурсии это значение уменьшается на единицу таким образом гарантирован выход из рекурсии, когда значение глубины станет равным нулю. Тогда из всех рекурсивных вызовов будут возвращены значение цвета, которые будут учитываться при закраске пикселя, до которого был выпущен изначальный луч.

2.2 Описание ключевых моментов алгоритма

Алгоритм обратной трассировки луча содержит несколько ключевых моментов, которые требуют более детального рассмотрения, а именно:

1) вычисление точки пересечения трассирующего луча с объектом;
2) вычисление нормали к поверхности в точки пересечение;
3) вычисление координат отраженного луча;
4) вычисление координат преломленного луча;
5) определение коэффициентов диффузного отражение, зеркального отражения и пропускания для различного типа материалов;

2.2.1 Вычисление точки пересечения со сферой

Пусть из точки O выпущен луч в направлении $\vec{d}$, как показано на рисунке 8

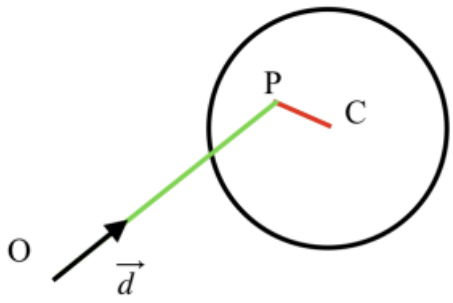


Рисунок 8 – Пересечение луча со сферой.

Точкой пересечения луча и сферы является P – решение системы

$$\begin{cases} \vec{p} = \vec{o} + t\vec{d} \\ (\vec{p} - \vec{c}) = r^2 \end{cases} \tag{3}$$

$(\vec{o} + t\vec{d} - \vec{c})^2 = r^2$ обозначим $\vec{s} = \vec{c} - \vec{a}$, тогда $(t\vec{d} - \vec{s})^2 = r^2$

$t^2\vec{d}^2 - 2t \cdot (\vec{d}, \vec{s}) + \vec{s}^2 = r^2$, где (d, s) – скалярное произведение $\vec{d}$ и $\vec{s}$, равное сумме попарных произведений их координат

Получилось квадратное уравнение 4

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

Если D $<$ 0, то луч проходит мимо сферы, если D $\geqslant$ 0, то уравнение имеет действительные корни eq:t12

$$t_{1,2} = \frac{(\vec{d}, \vec{s}) \pm \sqrt{D}}{\vec{d}^2} \tag{5}$$

Наименьшее положительное значение t, если оно существует, дает ответ задачи. Если же положительного значения нет, то луч сферу не пересекает.

Если точка пересечения найдена, то нас будет интересовать также нормаль к поверхности в этой точке, так как эта нормаль определяет направление отраженного луча, формула 6.

$$\vec{n} = \frac{\vec{p} - \vec{o}}{|\vec{p} - \vec{o}|} \tag{6}$$

2.2.2 Вычисление точки пересечения с плоскостью

Плоскость задается общим уравнением: $(\vec{p}, \vec{n}) = r$, где $\vec{n}$ – вектор нормали с плоскости, имеющий единичную длину, а r – растояния от начала координат до плоскости (с точностью до знака). См. рисунок 9

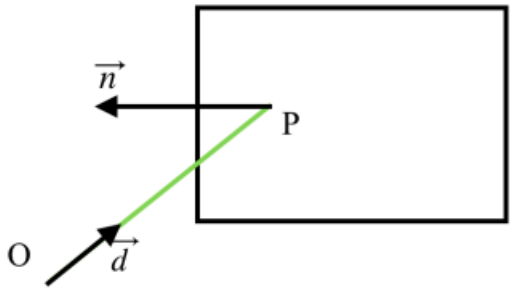


Рисунок 9 – Пересечение луча с плоскостью.

Точкой пересечения луча и плоскоти является P – решение системы 7

$$\begin{cases} \vec{p} = \vec{o} + t\vec{d} \\ (\vec{p} - \vec{c}) = r \end{cases} \tag{7}$$

Тогда решением будет 8

$$t = \frac{r - (\vec{o}, \vec{n})}{(\vec{d}, \vec{n})} \tag{8}$$

Если $(\vec{d}, \vec{n}) = 0$,то луч параллелен плоскости, т.е. не пересекает ее. Если же $(\vec{d}, \vec{n}) \neq 0$ то вычисляем t. Тогда если его значение положительно, то луч пересекает плоскость.

2.2.3 Вычисление координат отраженного и преломленного луча

Для того, чтобы ответить на вопрос, как вычисляются координаты отраженного и преломленного луча, необходимо обратиться к такому разделу оптики, как геометрическая оптика.

Геометрическая оптика позволяет рассматривать луч света в виде геометрической линии, вдоль которой и распространяется свет. При этом также геометрически можно показать, как свет может отразиться от поверхности, преломиться или пройти через нее. Основные законы геометрической оптики, включающие в себя закон прямолинейного распространения света, закон преломления и закон отражения, позволяют точно смоделировать путь луча от источника к наблюдателю или наоборот. Рассмотрим эти вышеперечисленные законы более подробно [8].

Закон прямолинейного распространения света: в оптически однородной среде свет распространяется прямолинейно.

Закон отражения света (проиллюстрирован на рисунке 10): угол отражения β равен углу падения α.

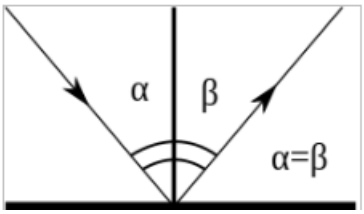


Рисунок 10 – Иллюстрация закона отражения.

Координаты отраженного луча могут быть вычислены аналитическим способом. Если $\vec{N}$ – нормаль к поверхностьям и $\vec{L}$ падающий луч, направленный из точки, то отраженный луч $\vec{R}$ может быть вычислен по следующей формуле 9:

$$\vec{R} = 2\vec{N}(\vec{N}, \vec{L}) - \vec{L} \tag{9}$$

Закон преломления света (проиллюстрирован на рисунке 11): падающий и

преломленный лучи, также как перпендикуляр к границе раздела двух сред, восстановленный в точке падения луча, лежат в одной плоскости. Отношение sin угла падения α к sin угла преломления является величиной, неизменной для двух приведенных сред: $\sin(\alpha)/\sin(\gamma) = \eta$, где $\eta = \eta_2/\eta_1$. $H_1\eta_2$– показатели преломления для первой и второй среды соответственно.

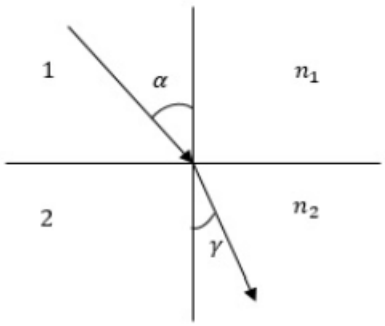


Рисунок 11 – Иллюстрация закона преломления

Если известна нормаль $\vec{N}$ к поверхности в точке, падающий луч $\vec{L}$, то координаты преломленного луча $\vec{T}$ могут быть вычислены по следующей формуле:

$$\vec{T} = \eta\vec{L} + \theta \tag{10}$$

В выражении 10 $\eta = \frac{\eta_1}{\eta_2}$, θ – угол между падающим лучом и нормалью, может быть вычислен как частное от деления скалярного произведения этих векторов на произведение их длин.

2.2.4 Определение коэффициентов, отражающих оптические свойства некоторых материалов

В данной работе было выделено несколько типов материалов, оптическими свойствами которых будут обладать геометрические объекты сцен. Ниже в таблице 2 приведены коэффициенты пропускания, зеркального и диффузного

отражения для выбранных материалов, а также коэффициент блеска материала. Также необходимо определить относительный показатель преломления двух сред, который требуется в расчете преломленного луча (см. выражение (10)).

Таблица 2 – Спектральные характеристики материалов

Материал	k_d	k_s	k_l	α	η
Стекло	0.01	0.2	0.79	125	1.0/1.5
Метал	0.1	0.7	0.0	50	1.5
Матовый	0.99	0.01	0.0	10	1.5

2.3 Разработка типов данных и структуры программы

Для реализации поставленной работы потребуется несколько пользовательских типов данных, которые будут отражать структуру объектов на сцене. Учитывая то, что эти типы данных должны содержать некоторую информацию об объекте, а также предоставлять возможность совершать какие-то действия над этими объектами, то удобнее всего будет представить разрабатываемые типы данных в виде классов. На рисунке 12 представлена диаграмма классов.

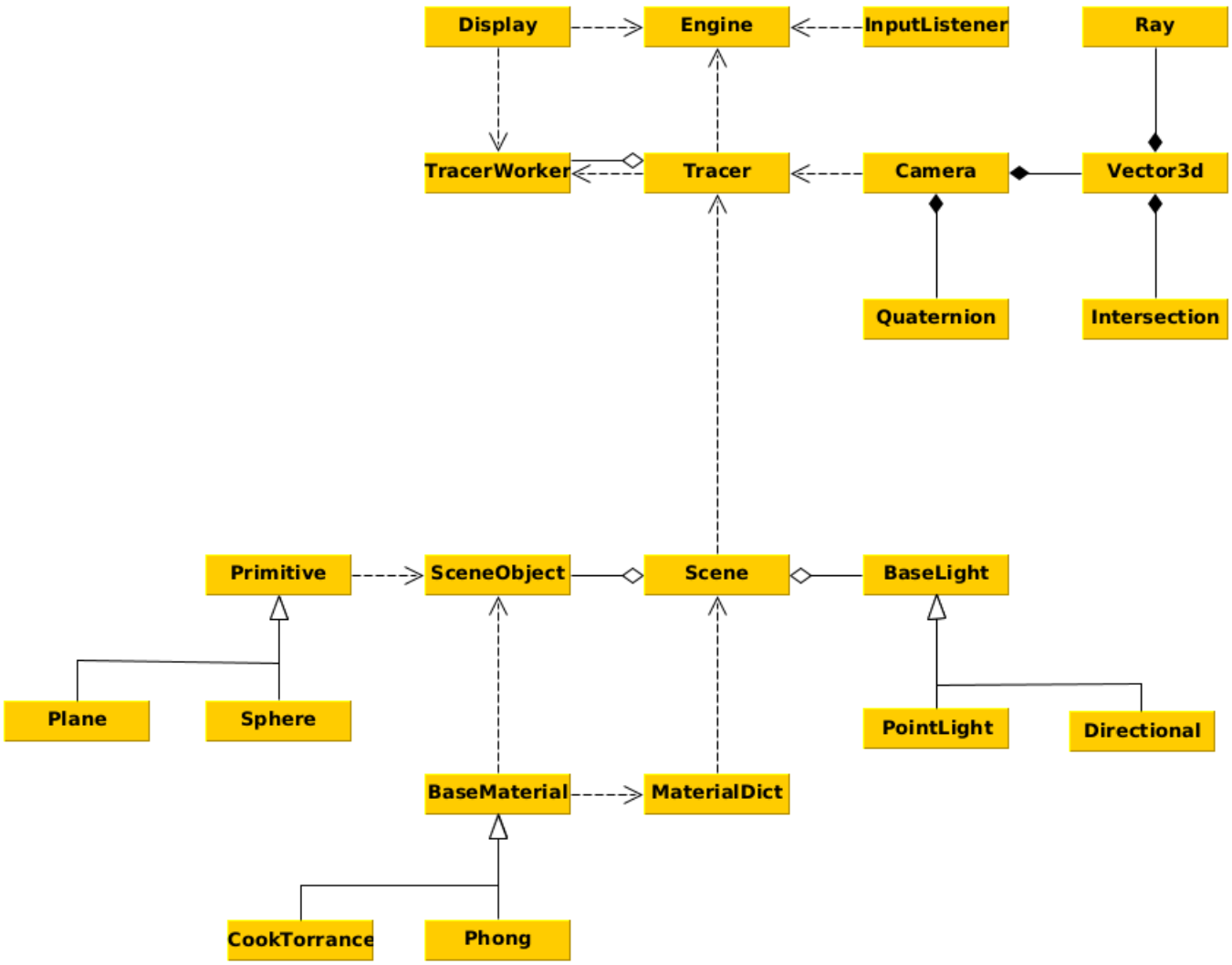


Рисунок 12 – Диаграмма классов

Главный класс который отвечает за цикл отрисовки является Engine. В нем есть класс Display который отвечает за экран, с помощью этого класса происходит закраска пикселей вычисленными значениями цветов. Также в классе Engine есть класс Tracer. Tracer производит трассирование лучей. К тому же Tracer содержит класс TracerWorker который отвечает за трассировки в определенном квадрате экрана. Класс Scene являеся зависимость класса Tracer. Scene содержит в себя информацию о сцене, массив объектов класса SceneObject, который принимает в себя список объектов класса Primitive и класс BaseMaterial который содержит информацио о материале объекта, также у класса Scene есть список из объектов класса BaseLight – это список источники света. В коде нашей камере будет соответствовать класс Camera. Для поворота камеры использован класс Quaternion. Vector3d – класс описывающий точку в пространстве. Класс Ray в свою очередь содержит класс Vector3d, который задает точку, из которой начинается луч, и класс еще раз Vector3d, который описывает направление луча. В этот класс реализован метод calculateCameraRay с помощью который строится луч выходящий из заданной точки и отвечающий требованиям камеры. Классы Plane, Sphere представляют собой производные от класса Primitive. В каждом из них определены методы нахождения точки пересечения и нормали к поверхности в заданной точке и описаны параметры, которые задают эти объекты.

3 Технологический раздел

В данном разделе будут описаны особенности программной реализации проекта.

3.1 Выбор средств программной реализации

Для написания данного курсового проекта был использован язык программирования Java. Этот язык поддерживает объектно-ориентированную модель разработки, что позволяет четко структурировать программу и легко модифицировать отдельные ее компоненты независимо от других. Язык Java позволяет эффективно использовать ресурсы системы благодаря широкому набору функций и классов. В качестве среды разработки была выбрана программа Intellij Idea. совместно с библиотекой java awt. Выбранная программа позволяет разработать графический пользовательский интерфейс, обеспечивает вывод графической информации, поддерживает средства отладки.

3.2 Интерфейс программы

При запуске программы отрисовывается сцена. Для того чтобы переместить камеру используются кнопки W – вперед, S – назад, A – влево, D – вправо. для поворота головы(камеры) используются стрелки, а так же кнопки Q, E.

Для того, чтобы добавить новый геометрический объект, необходимо в задать его в Сцену по правилам задания объекта См. таблицу (1).

3.3 Хранение и обмен данными в системе

Данные считываются из файла, содержащего в себе:

1-я строка содержит булевая для состояние отладчика.

2-я строка значение эпсилон

3-я строка маштаб

4-я строка делители экрана

5-я строка высоту экрана

6-я строка максимальную возможную рекурсию

7-я строка ширину экрана

8-я строка название программы.

4 Экспериментально-исследовательский раздел

В данном разделе будут представлены примеры работы программы, а также приведен анализ временных характеристик программы в зависимости от количества рабочих потоков.

4.1 Примеры работы программы

Далее на рисунках 13 – 15 будут приведены снимки экрана, на которых виден результат работы программы.

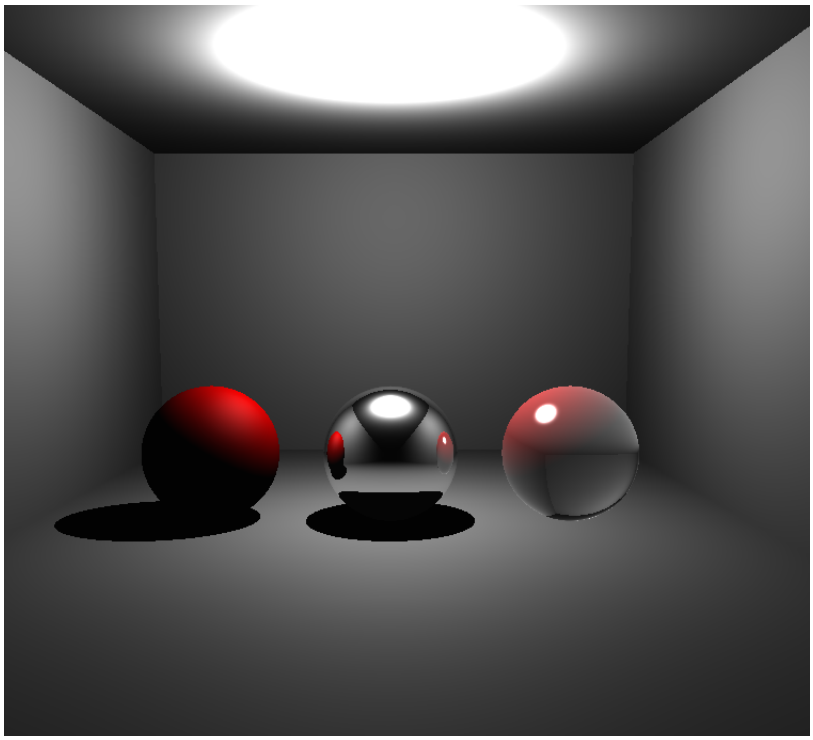


Рисунок 13 – Пример работы программы (1)

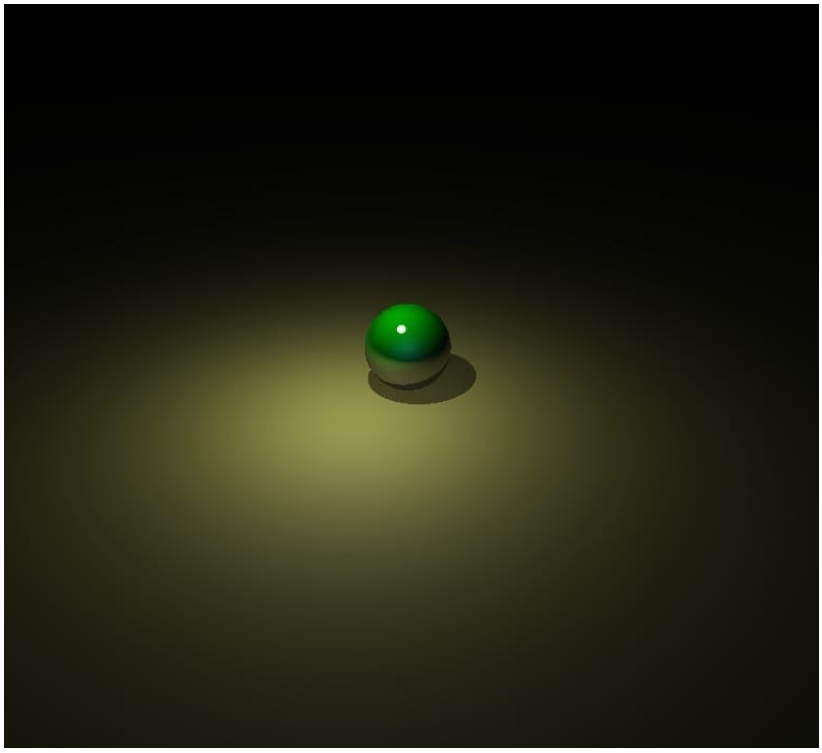


Рисунок 14 – Пример работы программы (2)

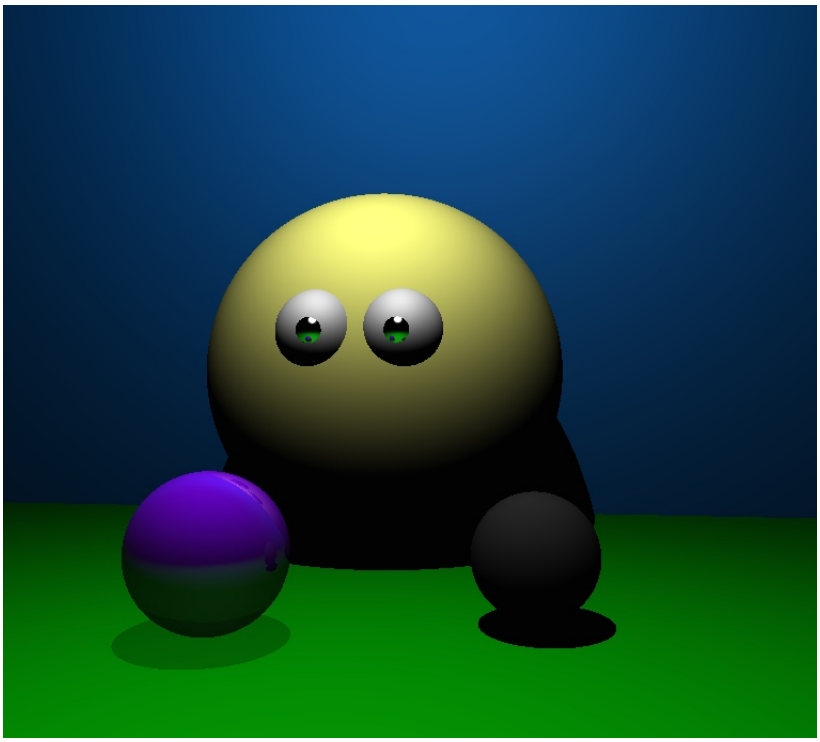


Рисунок 15 – Пример работы программы (3)

4.2 Исследование временных характеристик программы

Для исследования временных характеристик программы использовался компьютер на базе четырехъядерного процессора Intel Core i5 частотой 2.6 ГГц с 8 ГБ оперативной памяти.

Зависимость времени рендеринга от количества отображаемых на экране объектов для разного количества рабочих потоков

При проведении экспериментов на сцену по одному добавлялись одинаковые матовые сферы. Их количество изменялось от 1 до 25 включительно. Время замерялось при разном количестве рабочих потоков. Результат одного эксперимента рассчитывался как средний из результатов проведенных испытаний с одинаковыми входными данными. Количество повторов каждого эксперимента = 10. Результаты проведенного исследования представлены на рисунке 16.

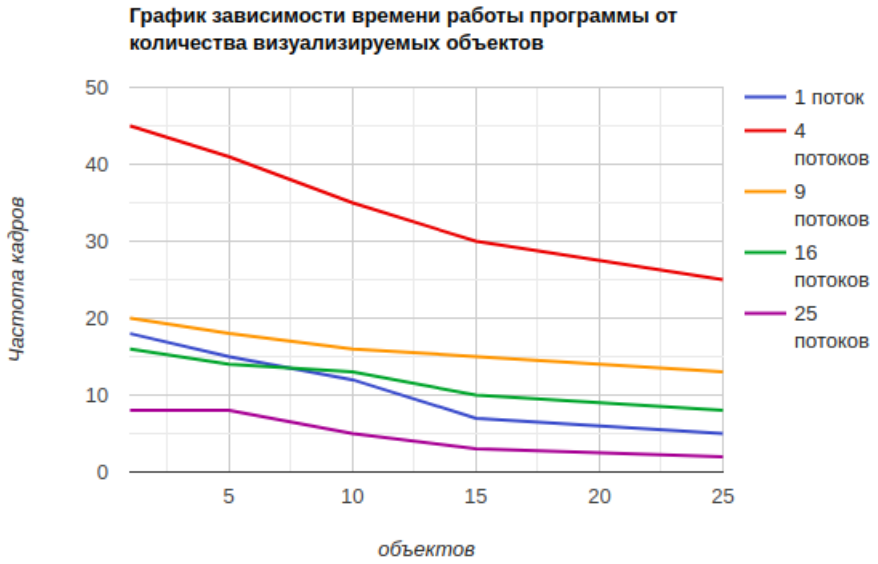


Рисунок 16 – График зависимости времени работы программы от количества визуализируемых объектов.

Как видно из графика, программа с один рабочим потоком работает медленнее, чем с большим количеством потоков. При этом время выполнения программы уменьшается пропорционально количеству рабочих потоков. Однако следует заметить, что при количестве рабочих потов больше четырех время выполнения программы начинает расти. Это связано с тем, что время на маршрутизацию потоков (создание дополнительных потоков, простой потоков в очереди, переключение контекста) превышает время выполнения основных вычислений. Таким образом, самое оптимальное количество потоков - четыре, что соответствует количеству логических процессоров компьютера. Также из графика видно, что время выполнения растет с увеличением количества отображаемых объектов. Данный факт можно объяснить так: чем больше объектов на сцене, тем больше лучей нужно трассировать.

Зависимость времени рендеринга от количества объектов с материалом «стекло»

При проведении экспериментов на сцену по одному добавлялись одинаковые стеклянные шары. Их количество изменялось от 1 до 25 включительно. Время замерялось при разном количестве рабочих потоков . Таким образом, одному измерению времени соответствует определенное количество объектов на сцене и определенное количество рабочих потоков. Результат одного эксперимента рассчитывался как средний из результатов проведенных испытаний с одинаковыми входными данными. Количество повторов каждого эксперимента = 10. Результаты проведенного исследования представлены на рисунке 17.

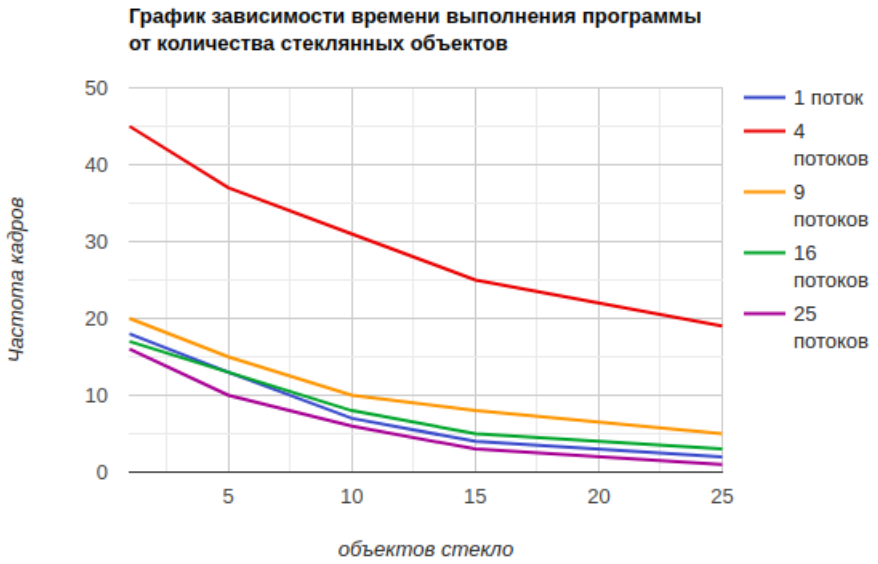


Рисунок 17 – График зависимости времени выполнения программы от количества стеклянных объектов.

Из графика видно, что время выполнения программы растет нелинейно в зависимости от количества «стеклянных» объектов на сцене. При этом при сравнении двух графиков видно, что, например, при визуализация 5 шаров с использованием 4 потоков, программа выдает 42 кадров в секунду, в то время как для отображения 5 стеклянных шаров – 36 кадров в секунду. Таким образом, рендеринг объектов, для которых коэффициент пропускания не равен нулю занимает больше времени, чем рендеринг непрозрачных неотражающих объектов. Это опять же объясняется количеством трассируемых лучей: дополнительные преломленный и отраженный луч.

Заключение

В результате проделанной работы выполнены следующие задачи:

1) поставлена задача;
2) рассмотренны и проанализированны существующие алгоритмы решения задачи, выбрать подходящий;
3) изучены физические принципы и законы, по которым строится выбранный метод визуализации;
4) изучинны математическую модель, описывающую работу метода.
5) выбран наиболее подходящую реализацию выбранного алгоритма;
6) разработана архитектура и структура программы;
7) разработан интерфейс;
8) выбранны средства программной реализации;
9) разработаны программные модули;

Достигнута цель проекта – изучение возможностей метода трассировки лучей и создание программного продукта, в котором возможна визуализация плоскостей и сфер.

В результате проделанной работы был разработан программный продукт, позволяющий создавать графические объекты с различными спектральными и геометрическими характеристиками.

Список литературы

[1] Бобков А.. Модели освещения. [ЭЛ. РЕСУРС] Режим доступа: https://alexander
bobkov.ru/opengl/tutorials/tutorial4. (дата обращения: 07.11.2020).

[2] Википедия. Ray casting. [ЭЛ. РЕСУРС] Режим доступа: https://ru.wikipedia.org
(дата обращения: 07.11.2020).

[3] Марчевский С.. Трассировка пути на GPU. [ЭЛ. РЕСУРС] Режим доступа: https://habr.com/ru/post/132862/. (дата обращения: 07.11.2020).

[4] Википедия. RGB. [ЭЛ. РЕСУРС] Режим доступа: https://ru.wikipedia.org/wiki/
(дата обращения: 07.11.2020).

[5] Порев В.Н. *Компьютерная графика.* СПб.:БХВ-Петербург, 2002.

[6] А.Г. Волобой. *Средства визуализации распространения световых лучей в задачах проектирования оптических систем.* 2009.

[7] Роджерс Д. *Алгоритмические основы машинной графики.* Д. Роджерс, Пер. с англ. – М.: Мир, 1989.

[8] Иродов И.Е. *Волновые процессы. Основные законы.* И.Е. Иродов - М.-С.-П.:Физматлит, 1999.

[9] Резник С.. Быстрая реализация модели освещения Кука-Торренса. [ЭЛ. РЕСУРС] Режим доступа: https://gamedev.ru/code/articles/Cook-Torrance.
(дата обращения: 07.11.2020).

[10] Кулагин Д.. Модель отражения Фонга. [ЭЛ. РЕСУРС] Режим доступа: https://compgraphics.info/3D/lighting/phong$_r e f l e c t i o n_m$odel.php. (дата обращения: 07.11.2020).