|  |  |
| --- | --- |
|  | **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 КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***Визуализация*** ***кристалла кварца при помощи метода трассировки лучей***

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А.С.Миневска

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Ю.В.Строганов

(Подпись, дата) (И.О.Фамилия)

*2019 г.*

**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)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В.Рудаков

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2019г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_Компьютерная графика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_\_ИУ7-56Б\_\_\_\_\_\_Миневска Ани Стоянова\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсового проекта \_\_\_\_Программа визуализации кристалла кварца методом трассировки лучей.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КП (учебный, исследовательский, практический, производственный, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

***Задание***\_\_\_\_\_\_Разработать программу визуализации кристалла кварца из заданного набора методом трассировки лучей. В программе должны быть учтены освещенность, тип поверхности объектов, должны быть визуализированы тени и блики.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсового проекта:***

Расчетно-пояснительная записка на \_30-35\_ листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Ю.В.Строганов

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А.С. Миневска

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[Введение 5](#_Toc27565887)

[1. Аналитический раздел 7](#_Toc27565888)

[1.1. Формализация задачи 7](#_Toc27565889)

[1.2. Анализ алгоритмов метода трассировки лучей 9](#_Toc27565890)

[1.3. Анализ модели освещения 13](#_Toc27565891)

[2. Конструкторский раздел 15](#_Toc27565892)

[2.1. Описание выбранного алгоритма 15](#_Toc27565893)

[2.2. Описание ключевых моментов алгоритма 19](#_Toc27565894)

[2.3. Разработка типов данных и структуры программы 28](#_Toc27565895)

[3. Технологический раздел 30](#_Toc27565896)

[3.1. Выбор средств программной реализации 30](#_Toc27565897)

[3.2. Интерфейс программы 30](#_Toc27565898)

[4. Экспериментально-исследовательский раздел 31](#_Toc27565899)

[4.1. Примеры работы программы 31](#_Toc27565900)

[4.2. Исследование временных характеристик программы 32](#_Toc27565901)

[Заключение 36](#_Toc27565902)

[Список источников 37](#_Toc27565903)

# Введение

Проблема визуализации 3D изображения с помощью ЭВМ всегда остается актуальной. Создание компьютерных игр, симуляторов, анимационных фильмов, проектирование оптических систем [1, c. 1 - 2] – 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 мощным и универсальным методом создания фотореалистичных изображений [2, c. 180]. 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 графических объектов, геометрические и спектральные параметры которых задаются, в частности кристалл кварца.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Формализовать 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 структуру программы.
7. Разработать интерфейс.
8. Выбрать средства программной реализации.
9. Разработать программные модули.

# Аналитический раздел

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

## Формализация задачи

Для того, чтобы найти подходящее решение поставленной задачи, необходимо определить, какие будут входные данные, каков ход решения задачи и каким должен быть результат.

Входные данные – это характеристики объектов сцены. Объектами сцены в данной работе являются геометрические объекты и источники света. Источник 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 |
| Кристал кварца | Задается его центром. При создании он всегда направлен вдоль оси Oz | Задаются его радиусом (R), высотой и широтой |
| Треугольник | Задается при помощи его вершинами | Задаются его вершинами A, B и C |
| Прямоугольник | Задается при помощи 3х вершин | Задаются его вершинами A, B и C |
| Правильный многоугольник | Задается его центром. При создании он всегда направлен вдоль оси Oz | Задаются его радиусом (R), и количество ребер. |

При этом стоит отметить, что все геометрические объекты имплементируют интерфейс IRotate и их можно повернуть при помощи метода rotate(…). При этом применяется перспективное преобразование, ось z направлена «вглубь экрана», а наблюдать находится на конечном расстоянии от объектов сцены. Ось x направлена вправо, ось y вниз.

Для определения цвета геометрического объекта и источника будет использована цветовая модель RGB [3].

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.

## Анализ алгоритмов метода трассировки лучей

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

Главная 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 [4, c. 361]. 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.

**Классический алгоритм трассировки лучей**

В традиционной трассировке лучей лучи света обрабатываются в обратном направлении (backward ray tracing): 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 [4]. 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. Однако в данном алгоритме не учитывается диффузная составляющая интенсивности от соседних объектов.

**Алгоритм трассировки пути**

Алгоритм трассировки пути очень похож на алгоритм трассировки лучей, однако учитывает диффузную составляющую интенсивности света, полученную от других объектов сцены. 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. Однако для получения реалистичного изображения таких произвольных лучей для каждого пикселя должно быть создано несколько тысяч [5], 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, однако и получаемое изображение практически схоже с фотографией.

**Алгоритм бросания лучей (ray casting)**

Рейкастинг может быть представлен как сокращённая 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, пересечётся с поверхностью. Эта особенность делает невозможным точный рендеринг отражений, преломлений и естественной проекции теней с помощью рейкастинга. 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 [6].

**Сравнение визуальных характеристик изображений, полученных разными алгоритмами**

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

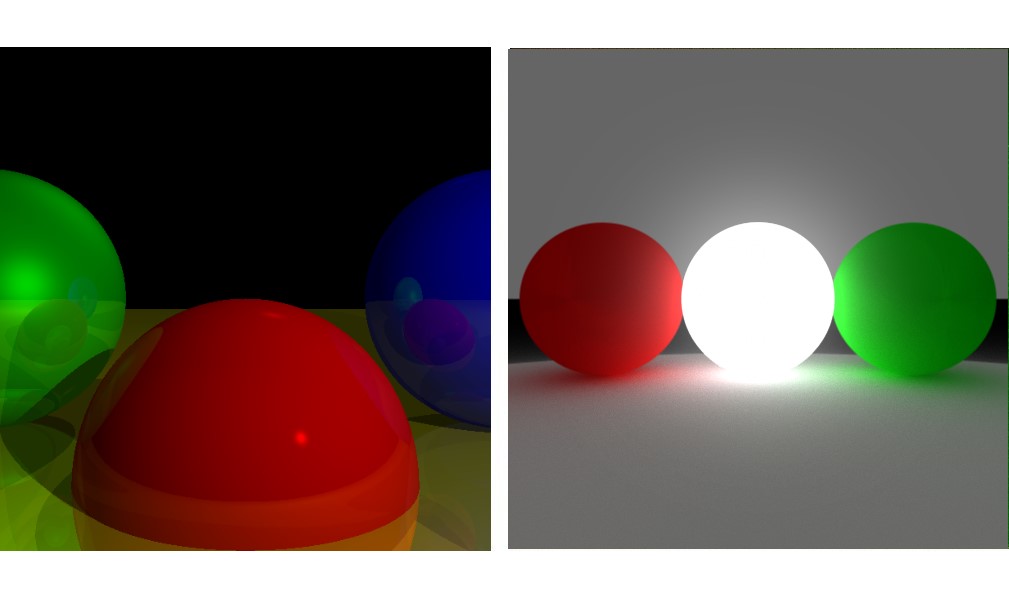


Рисунок 1. Изображения, полученные трассировкой лучей (слева) и трассировкой пути (справа)

Как видно из рисунка, трассировка лучей не дает такого визуального эффекта мягкого света и реалистичности изображения, как трассировка пути. Однако, на рисунке 2 показано, как меняется качество изображения, получаемого трассировкой пути, в зависимости от количества случайных лучей.

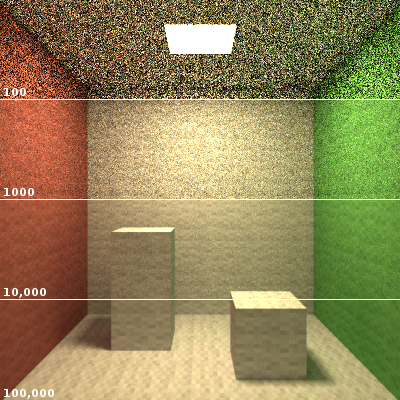


Рисунок 2. Шум на изображении в зависимости от количества случайных лучей

На рисунке 3 приведен пример полученного алгоритмом бросания лучей изображения. Как видно из рисунка, реалистичность полученного изображения намного хуже, чем у предыдущих двух алгоритмов.

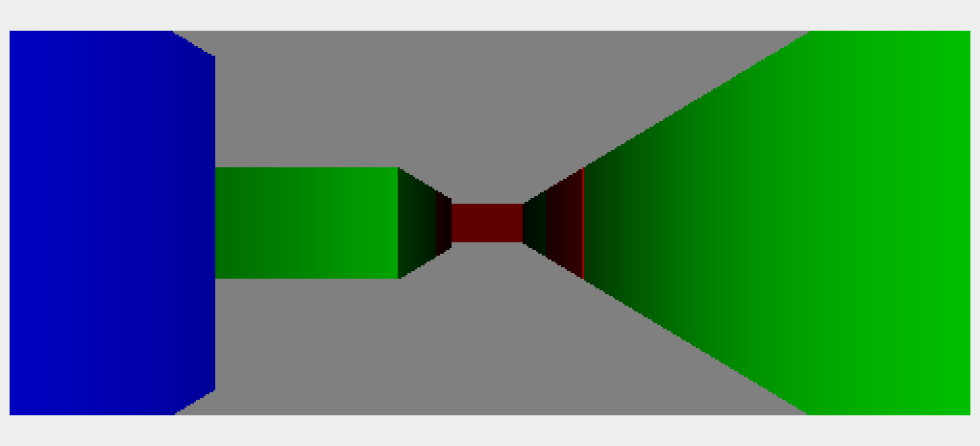


Рисунок 3. Изображение, полученное алгоритмом бросания лучей

**Выбор алгоритма**

Определим следующие критерии выбора алгоритма: визуальная реалистичность изображения и время работы алгоритма. Поскольку основным критерием 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, классический алгоритм трассировки лучей является наиболее подходящим для решения поставленных в этой работе задач.

## Анализ модели освещения

Как уже отмечалось ранее, для получения фотореалистичного изображения, необходимо использовать глобальную модель освещения, т.е. такую модель, которая учитывает все имеющиеся источники освещения. Источниками энергии могу быть не только 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. Одной из самых распространенных моделей, реализующих этот принцип, является модель отражения по Фонгу. На рисунке 4 представлены три компонента и результат их сложения.



Рисунок 4. Модель отражения по Фонгу

Выражение, с помощью которого можно вычислить интенсивность света в заданной точке для среды с одним источником освещения будет выглядеть следующим образом:

(1)

I – суммарная интенсивность света в заданной точке, – фоновая составляющая интенсивности, – коэффициент рассеянности света, – интенсивность источника света, – вектор из точки на поверхности к источнику освещения, – нормаль к поверхности в точке, – коэффициент диффузного отражения, – вектор отражения, – вектор к наблюдателю, – коэффициент блеска материала, – коэффициент зеркального отражения. Так как модель аддитивная, то для того, чтобы учитывать несколько источников освещения, необходимо вместо второго и третьего слагаемых взять суммы по всем источникам.

Однако для решения поставленной в данной работе задачи необходимо также учитывать компоненты, соответствующие глобальному зеркальному отражению и пропусканию. Согласно модели Уиттеда [4, с. 440], для учета интенсивности от отраженного и преломленного лучей можно использовать выражение (2), включающее в себя модель Фонга и учет нескольких источников.

(2)

m – количество источников, – интенсивность зеркального отражения света, – коэффициент пропускания, – интенсивность света, прошедшего по преломленному лучу.

Таким образом, для учета освещения необходимо задавать параметры материала, из которого будет состоять геометрический объект сцены.

# 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.

## Описание выбранного алгоритма

На рисунке 5 представлена обобщенная схема алгоритма трассировки, которая будет подробнее рассмотрена далее.

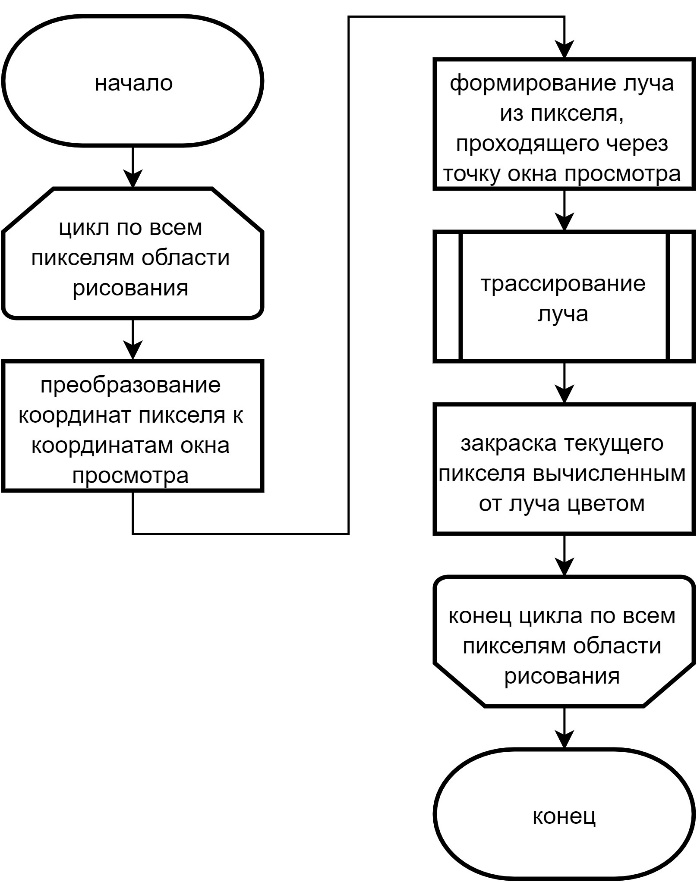


Рисунок 5. Обобщенная схема алгоритма

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

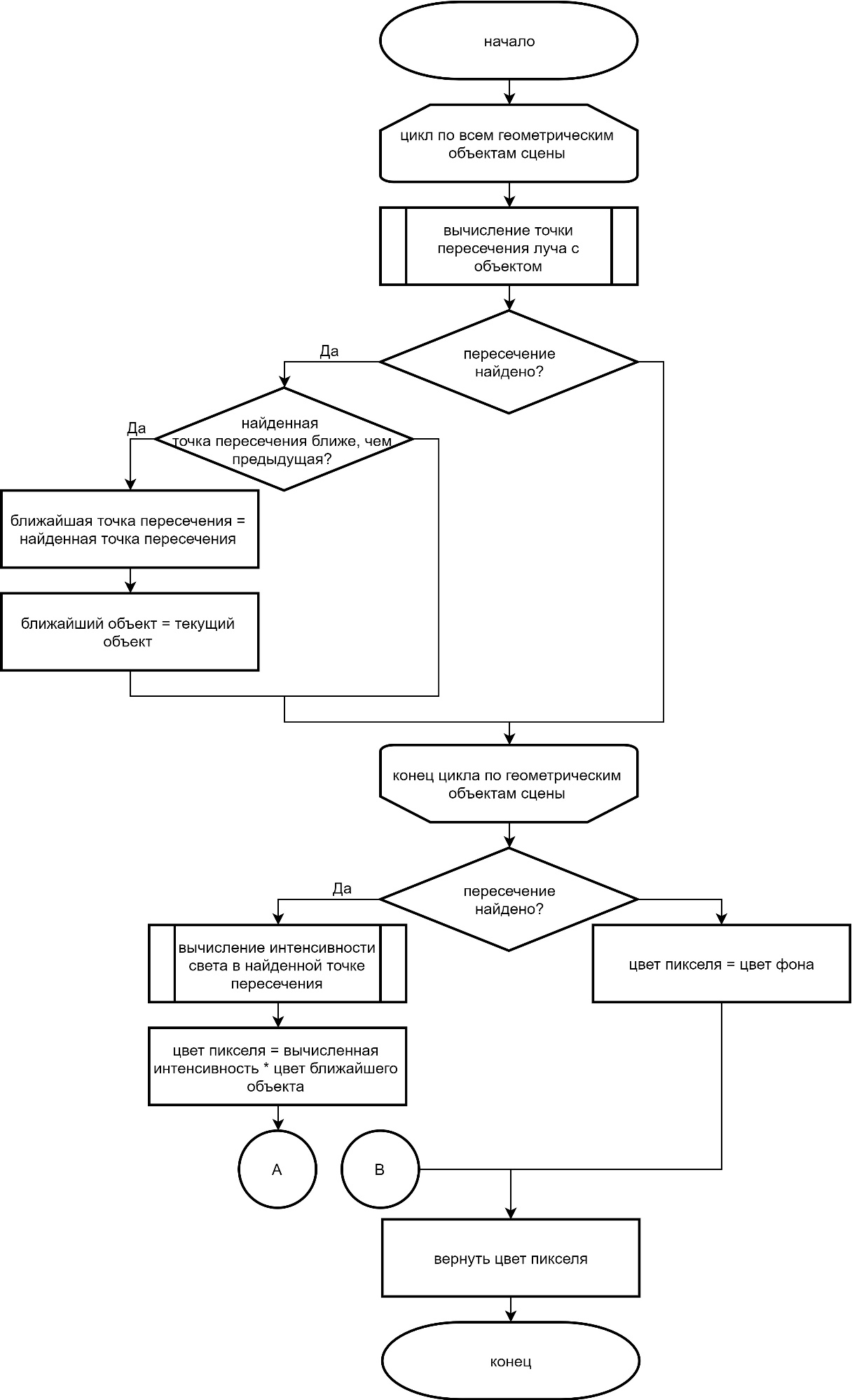


Рисунок 6. Схема алгоритма трассирования луча (часть 1)

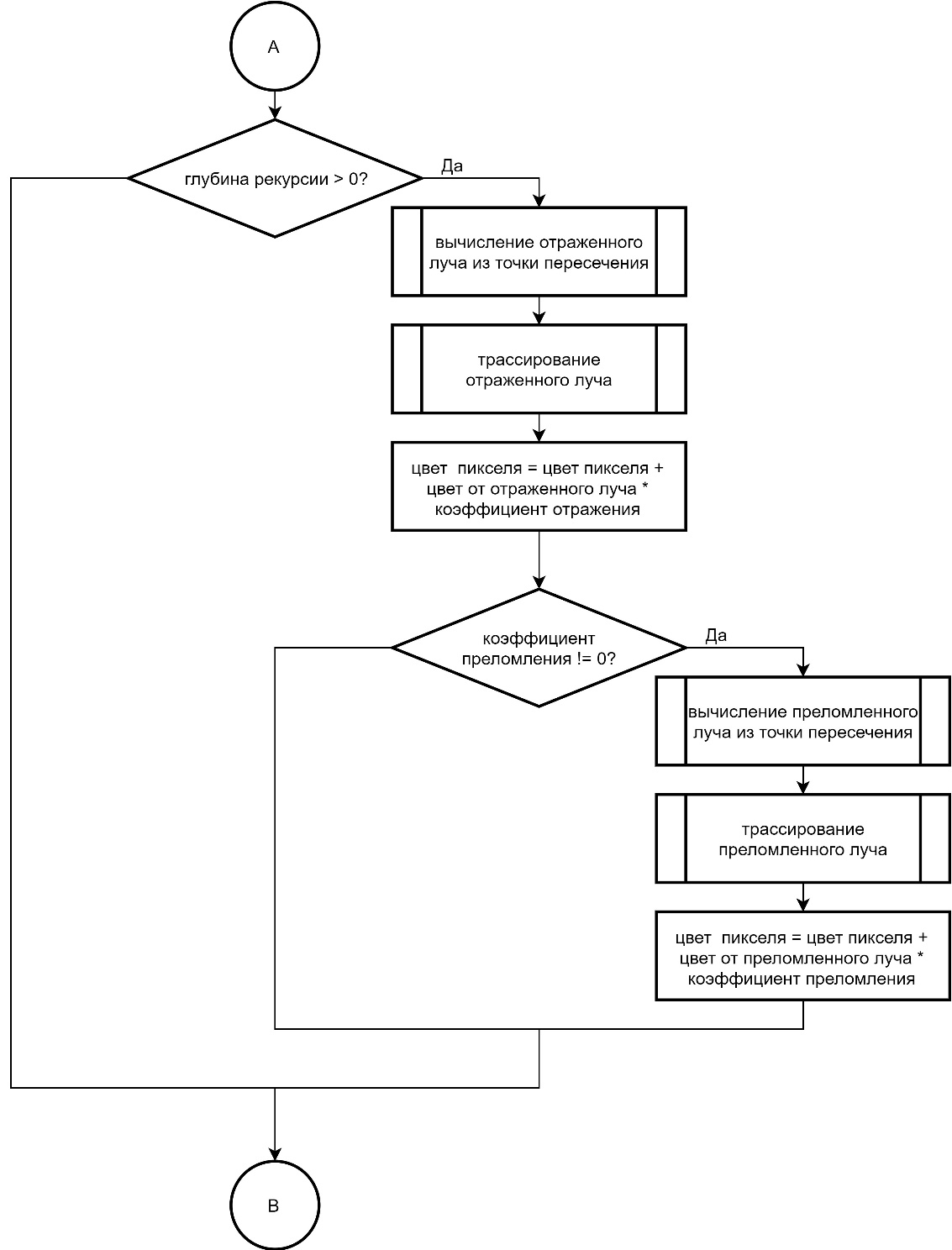


Рисунок 7. Схема алгоритма трассирования луча (часть 2)

Как видно из схемы этого участка алгоритма, функция трассирования луча вызывается рекурсивно для отраженного и преломленного лучей. Для изначального луча (от наблюдателя) задается глубина рекурсии, равная 3. 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 значения цвета, которые будут учитываться при закраске пикселя, до которого был выпущен изначальный луч.

На рисунке 8 будет представлен алгоритм вычисления интенсивности в точке.

|  |  |
| --- | --- |
|  |  |

Рисунок 8. Схема алгоритма вычисление интенсивности

## Описание ключевых моментов алгоритма

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

1. Вычисление точки пересечения трассируемого луча с объектом;
2. Вычисление нормали к поверхности в точке пересечения;
3. Вычисление координат отраженного луча;
4. Вычисление координат преломленного луча;
5. Определение коэффициентов диффузного отражения, зеркального отражения и пропускания для различного типа материалов.

**Вычисление точки пересечения трассируемого луча с объектом и нормали к поверхности объекта в этой точке**

Вычисление точки пересечения и нормали остается индивидуальным для каждого типа поверхности. Поскольку в данной работе необходимо визуализировать 6 типов объектов, рассмотрим решение проблемы только для них.

Для начала определим, что наиболее удобным способом представление лучей для решения данной задачи является параметрическое уравнение. Если известно, что луч проходит через точку O, и известно его направление (из т. O в т. V = V – O = ), то можно выразить любую точку P луча как

(3)

Если рассматривать выражение (3) в проекции на оси координат, то получится следующая система уравнений:

(4)

Теперь для того, чтобы найти точку пересечения луча и некоторой поверхности, необходимо выразить уравнение поверхности в параметрическом виде, решить уравнение относительно t и подставить в выражение (3). Так будет найдена точка P, которая будет одновременно принадлежать и лучу и поверхности. Если решение уравнений относительно t даст несколько результатов, то стоит выбрать наименьший, так как это значение будет соответствовать ближайшей к наблюдателю точке. Нахождение ближайшей к наблюдателю точки является необходимым условием алгоритма. При это значения t, меньшие нуля, следует игнорировать, так как точка, соответствующая этому значению, будет находится за камерой, т.е. вне поля зрения наблюдателя. Также если объект представляет собой соединение нескольких типов поверхности, то пересечения ищутся для каждой и затем также выбирается наименьшее значение t. Если ни одно из найденных значений t не подходит, то луч не пересекает объект. Далее рассмотрим конкретные примеры уравнений для нахождения точек пересечения и нормалей.

1. Шар.

Для того, чтобы отобразить шар, необходимо вычислить точку пересечения очередного трассируемого луча с поверхностью сферы, ограничивающей шар. Если т. C — центр сферы, а r — радиус сферы, то точки P на поверхности сферы удовлетворяют следующему уравнению:

(5)

В уравнении (5) P - C – вектор из C в P, а – скалярное произведение этого вектора на себя, что равно квадрату расстояния от точки C до точки P, или квадрату радиуса сферы. Подставляя в уравнение (5) уравнение (3) получим следующее уравнение относительно t:

(6)

Преобразовав выражение (6), получим следующее уравнение:

(7)

Уравнение (7) представляет собой квадратное уравнение, которое можно решить известными способами. Далее точка пересечения находится по правилам, описанным выше. Для нахождения нормали к поверхности сферы в найденной точке пересечения необходимо найти координаты вектора, который начинается в центре сферы и заканчивается в найденной точке, и нормализовать его.

1. Параллелепипед.

Поскольку в данной работе ребра параллелепипеда параллельны осям, то нахождение точек пересечения луча с параллелепипедом сводится к нахождению точек пересечения луча с плоскостями, параллельными плоскостям проекции. В книге [7, c. 355-361] подробно описывается метод Кая и Кадзии для нахождения t точки пересечения. Нормаль в зависимости от того, на какой грани параллелепипеда была найдена точка пересечения, будет единичным вектором вдоль или противоположно оси, перпендикулярно которой располагается грань с точкой пересечения.

1. Пирамида.

Правильная четырехгранная пирамида состоит из четырех треугольников и одного квадрата, поэтому наименьшее t для всей пирамиды будет наименьшим из найденных для вышеперечисленных примитивов. Для того, чтобы найти пересечение с треугольником, можно перейти к барицентрическим координатам [8]. Зная вершины пирамиды, можно задать треугольник тремя вершинами (a, b, c), а из них уже сформировать два вектора, выходящие из одной точки, длины которых равны длинам соответствующих ребер пирамиды. Тогда произвольная точка p на поверхности, в которой содержится треугольник, может быть задана следующим выражением:

(8)

В выражении (8) – коэффициенты барицентрической комбинации, для которых должно выполняться условие: . Для точек, расположенных внутри треугольника, каждый из этих коэффициентов должен быть больше нуля и меньше единицы. Можно выразить один из коэффициентов через два другие, например . Тогда выражение (8) примет следующий вид:

(9)

Чтобы найти точку пересечения нужно в левую часть уравнения (9) подставить правую часть выражения (3), совершив некоторые преобразования получим:

(10)

Используя систему (4), выражение (10) можно преобразовать к следующей системе:

(11)

Далее необходимо решить систему линейных уравнений (11) методом Крамера относительно t, . Нормаль к поверхности треугольника может быть вычислена как векторное произведение векторов, направленных по сторонам треугольника.

Чтобы найти пересечение с квадратом нужно найти точку пересечения с плоскостью, содержащую квадрат, и проверить, находится ли точка внутри квадрата. Для нахождения t можно использовать следующее выражение:

(12)

В выражении (12) M – точка, принадлежащая плоскости, O – точка, из которой начинается луч, – нормаль к плоскости, – направление луча. Нормаль к основанию пирамиды в этом проекте всегда имеет координаты (0, 1, 0).

1. Цилиндр.

Уравнение цилиндра можно записать в следующем виде:

(13)

Где x0, z0 – смещение центра основания цилиндра относительно начала координат, r – радиус цилиндра. Уравнение (13) – это уравнение бесконечного по оси y цилиндра. Чтобы отобразить цилиндр заданной высоты, необходимо проверить y-координату найденной точки пересечения, что её значение заключено в интервале [y0, y0+h], где y0 – у-координата заданной точки центра нижнего основания цилиндра, а h – заданная высота. Подставив в уравнение (13) правую часть уравнений системы (4) для соответственно проекций x и z, получим следующее уравнение:

(14)

Где (dx, dy, dz) = , (ox, oy, oz) = O, определенные для выражения (3). Уравнение (14) представляет собой квадратное уравнение. Нормаль к поверхности может быть найдена как нормализованный вектор, начало которого расположен в точке на оси конуса с y-координатой точки пересечения, а конец в самой точке пересечения. Цилиндр сверху и снизу ограничен диском. Для того, чтобы найти пересечение луча с диском, необходимо сначала найти точку пересечения луча с поверхностью, содержащей диск, по формуле (12). Далее необходимо проверить, что расстояние от найденной точки до центра основания цилиндра (с которым ищется пересечение) не превышает значения радиуса основания цилиндра (для ускорения вычислений можно сравнивать квадраты расстояний):

( (15)

Если луч пересекает цилиндр в одном из оснований, то координаты нормали в этой точке будут равны: (0, 1, 0), если пересечение найдено с нижним основанием, либо (0, -1, 0), если пересечение найдено с верхним основанием.

1. Конус.

Уравнение конуса может быть задано следующим образом [7, c.392]:

(16)

Где x0, y0, z0 – смещение центра основания конуса относительно начала координат, r – радиус основания конуса, h – высота. Уравнение (16) описывает бесконечный конус, чтобы получить только нижнюю часть конуса с заданной высотой, необходимо проверить, что y-координата найденной точки пересечения заключена в интервале [y0, y0+h], где y0 – у-координата заданной точки центра основания конуса, а h – заданная высота. В уравнение (16) необходимо подставить соответствующие значения из системы (4), тогда получится уравнение следующего вида:

(17)

Уравнение (13) является квадратным уравнением, решение которого находится известным способом. В данной задаче конус ограничен снизу, поэтому также необходимо учитывать тот факт, что луч может пересечь основание конуса. Тогда необходимо также проверять пересечение луча с диском, образованным нижним основанием. Нормаль к поверхности можно найти, вычислив градиент функции конуса:

(18)

1. Тор.

Уравнение тора может быть задано в следующем виде:

(19)

Где x0, y0, z0 – смещение центра направляющей окружности тора относительно начала координат, a – радиус направляющей окружности, b – радиус образующей окружности тора. Аналогичным образом, необходимо подставить правые части уравнений системы (4) в уравнение (19) и получить уравнение относительно t:

(20)

В уравнении (20):

,

,

,

,

.

Уравнение (20) имеет четвертую степень. Чтобы решить уравнение четвертой степени в общем виде, необходимо сначала методом Декарта-Эйлера [9] привести его к уравнению третьей степени, которое можно решить методом Кардано [10][11]. Нормаль к поверхности также находится через нахождение градиента:

(21)

**Вычисление координат отраженного и преломленного лучей**

Для того, чтобы ответить на вопрос, как вычисляются координаты отраженного и преломленного лучей, 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 [12, c. 69].

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

Закон отражения света (проиллюстрирован на рисунке 9): угол отражения β равен углу падения α.

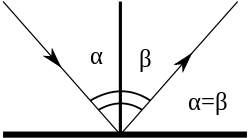


Рисунок 9. Иллюстрация закона отражения

Координаты отраженного луча могут быть вычислены аналитическим способом. Если – нормаль к поверхности в некоторой точке, а – падающий луч, направленный из точки, то отраженный луч может быть вычислен по следующей формуле:

(22)

Закон преломления света (проиллюстрирован на рисунке 10): 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 γ является величиной, неизменной для двух приведенных сред: sin(α)/sin(γ) = η, где η = η2/η1. Η1 и η2 – показатели преломления для первой и второй сред соответственно.

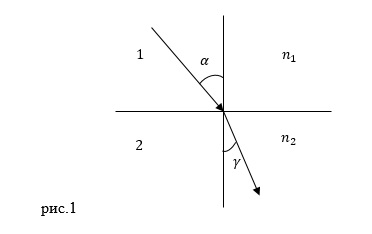


Рисунок 10. Иллюстрация закона преломления

Если известна нормаль к поверхности в точке, падающий луч , то координаты преломленного луча могут быть вычислены по следующей формуле:

(23)

В выражении (23) , – угол между падающим лучом и нормалью, может быть вычислен как частное от деления скалярного произведения этих векторов на произведение их длин.

**Определение коэффициентов, отражающих оптические свойства некоторых материалов**

В данной работе было выделено несколько типов материалов, оптическими свойствами которых будут обладать геометрические объекты сцены. Ниже в таблице 2 приведены коэффициенты пропускания, зеркального и диффузного отражения для выбранных материалов, а также коэффициент блеска материала. Эти параметры учитываются при расчете интенсивность света в точке (см. выражение (1)). Также необходимо определить относительный показатель преломления двух сред, который требуется в расчете преломленного луча (см. выражение (23)).

*Таблица 2 – Спектральные характеристики материалов*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Материал |  |  |  |  |  |
| Стекло | 0.01 | 0.2 | 0.79 | 125 | 1.0 / 1.5 |
| Металл | 0.1 | 0.7 | 0.0 | 50 | 1.5 |
| Зеркало | 0.0 | 1.0 | 0.0 | 50 | 1.5 |
| Пластик | 0.8 | 0.2 | 0.0 | 100 | 1.5 |
| Резина | 0.99 | 0.01 | 0.0 | 10 | 1.5 |

## 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 11 представлена диаграмма классов.

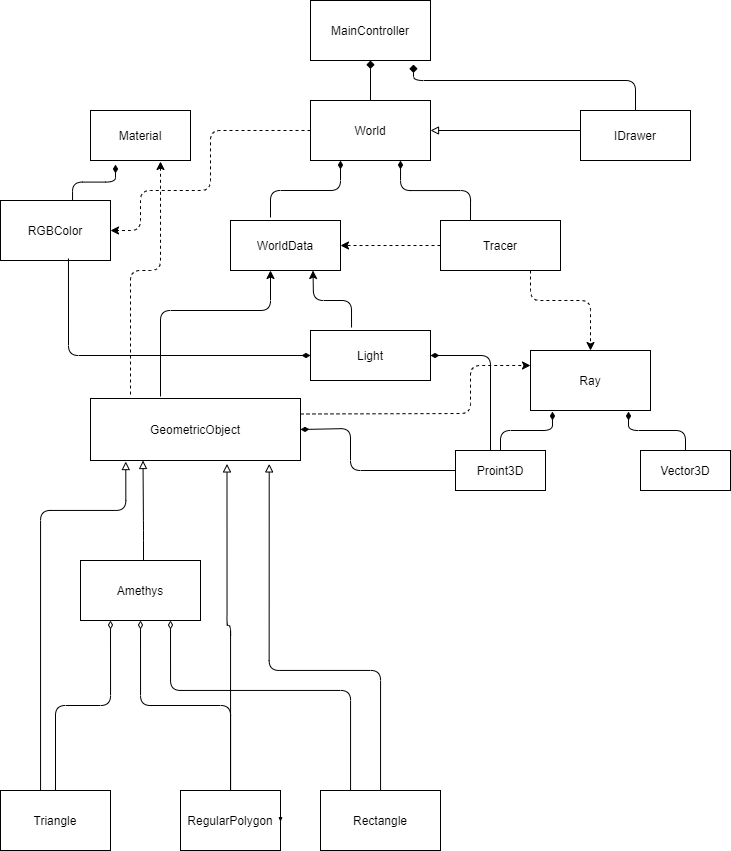
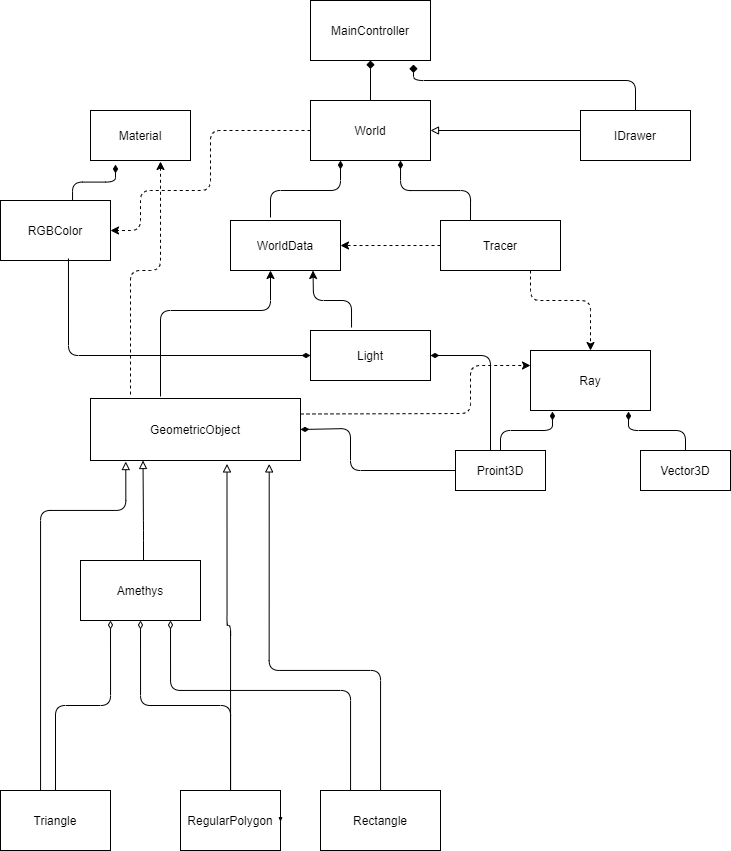


Рисунок 11. Диаграмма классов

Класс главного окна программы, через которое пользователь взаимодействует с программой, представляет класс MainController. В нем создается объект IDrawer, с помощью которого происходит закраска пикселей вычисленными значениями цветов. Также Config хранит объекты Material, в котором определены оптические свойства некоторых материалов. К тому же MainController содержит класс World, который в совокупности с классом WorldData представляет собой контейнер всех отображаемых на экране объектов. Класс World содержит класс Tracer, который непосредственно производит трассирование лучей, поэтому он использует класс, описывающий геометрический луч Ray, и класс WorldData. Класс Ray в свою очередь содержит класс Point3D, который задает точку, из которой начинается луч, и класс Vector3D, который описывает направление луча.

WorldData содержит два массива: массив указателей на объекты класса GeometricObject, который содержит всю информацию о каком-либо геометрическом объекте, и массив указателей на объекты класса Light. Поскольку свет задается точкой в трехмерном пространстве, иненсивность и цветом, то класс Light содержит класс RGBColor и класс, описывающий точку в пространстве – Point3D (интенсивность источника может быть описана встроенным типом данных, например, таким как double). Классы Amethyst, Rectangle, RegularPolygon, Rectangle, Triangle представляют собой производные от класса GeometricObject. В каждом из них определены методы нахождения точки пересечения и нормали к поверхности в заданной точке и описаны параметры, которые задают эти объекты.

# Технологический раздел

В данном разделе будут описаны особенности программной реализации проекта.

## 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 фреймворком javafx. Выбранная программа позволяет разработать графический пользовательский интерфейс, обеспечивает вывод графической информации, поддерживает средства отладки.

Используемый sdk: openjdk 11.

## Интерфейс программы

При запуске программы на пустом виджете рисования отображены координатные оси с единичными отрезками, фон черный.

Для того, чтобы добавить новый геометрический объект, необходимо в MainController проинициализировать его, выбрать необходимы материал.

Для каждого объекта задан способ определения его положения.

Необходимо указать цвет объекта. Цвет можно задать как вручную, указав в полях RGBColor значения для палитры RGB от 0 до 1.

После выставления всех параметров необходимо добавить его при помощи метода addObject(…)

Также можно добавить источники света, задавь поля положение, цвет, интенсивность соответственно точку, в которой будет располагаться источник, цвет его света и интенсивность света. Далее необходимо использовать word.addLight(…)

# Экспериментально-исследовательский раздел

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

## Примеры работы программы

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

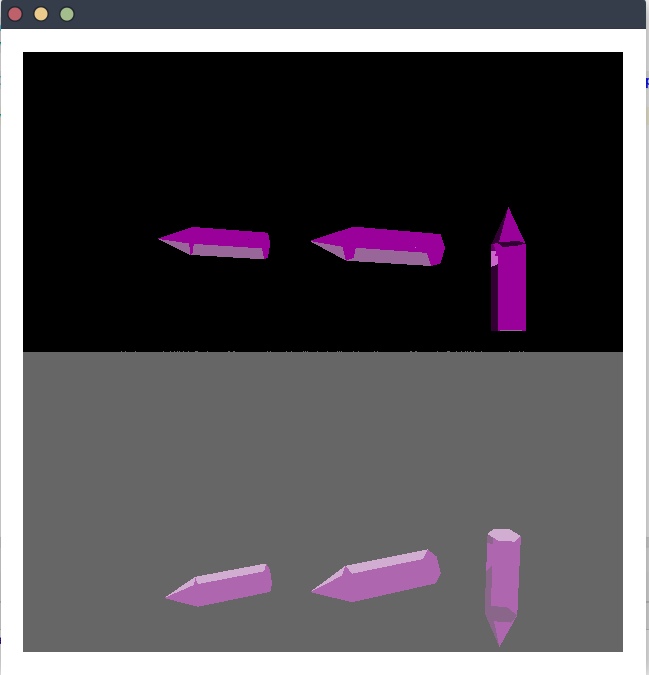


Рисунок 12. Пример работы программы (1)

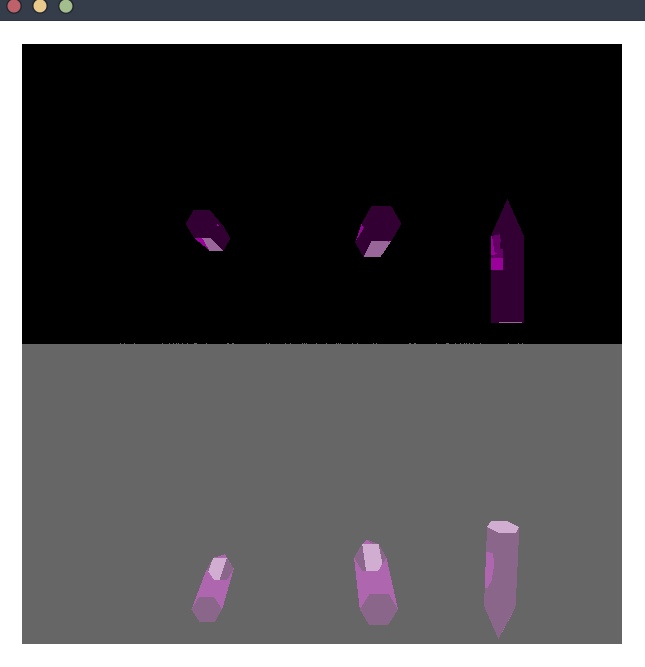
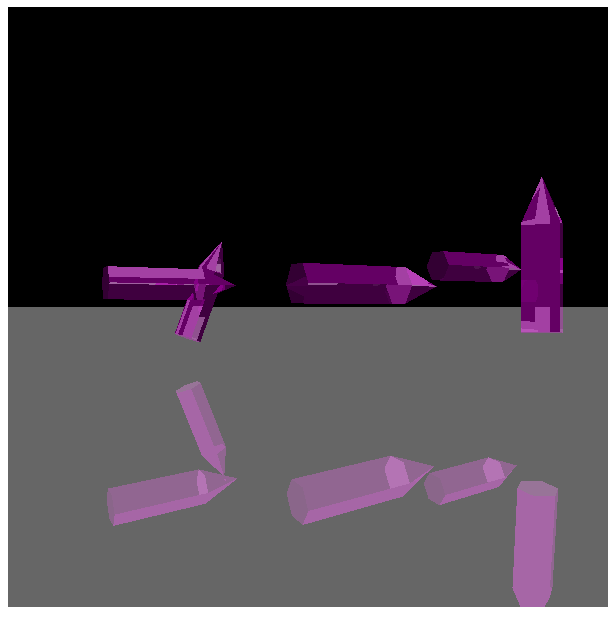


Рисунок 13. Пример работы программы (2)



*Рисунок 13. Пример работы программы (3)*

## Исследование временных характеристик программы

Для исследования временных характеристик программы использовался компьютер на базе четырехъядерного процессора Intel Core i5 частотой 3.5 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 резиновые кристаллы. Их количество изменялось от 1 до 10 включительно. Время замерялось при разном количестве рабочих потоков (от 1 до 4, далее 6, 8, 10). 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 .

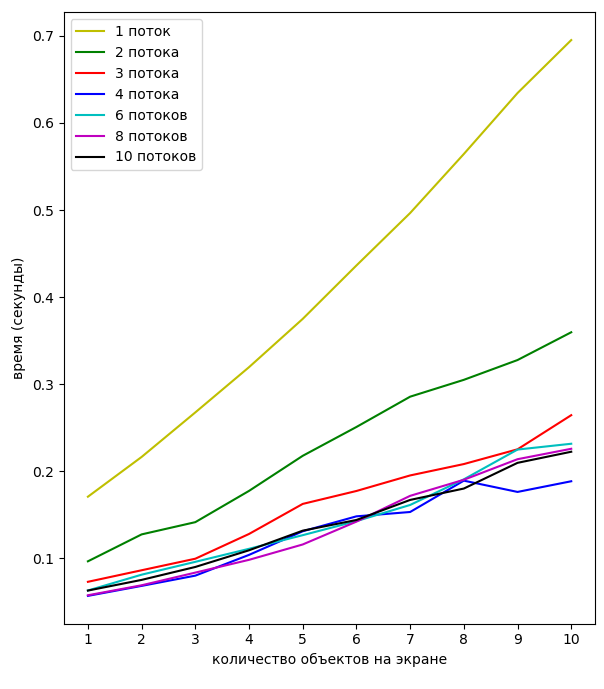


Рисунок 14. График зависимости времени работы программы от количества визуализируемых объектов

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 луче нужно трассировать (а трассирование луча включает в себя все самые трудоемкие вычисления в программе).

**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 9 включительно. Время замерялось при разном количестве рабочих потоков (1 и 4). 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 .

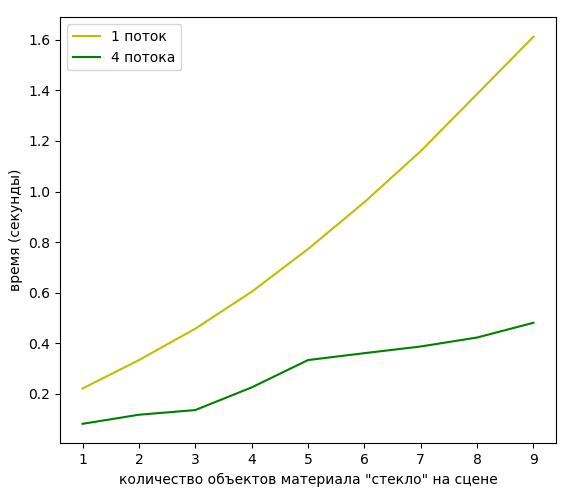


Рисунок 15. График зависимости времени выполнения программы от количества стеклянных объектов

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, на визуализацию 5 резиновых шаров одному потоку потребовалось 0.4 секунды, в то время как для отображения 5 стеклянных шаров – 0.7 секунд. Для четырех потоков: 0.13 секунды и 0.3 секунды соответственно. Таким образом, рендеринг объектов, для которых коэффициент пропускания не равен нулю занимает в 2.3 раза больше времени, чем рендеринг непрозрачных неотражающих объектов. Это опять же объясняется количеством трассируемых лучей: дополнительные преломленный и отраженный лучи.

# Заключение

В результате проделанной работы был разработан программный продукт, позволяющий создавать графические объекты с различными спектральными и геометрическими характеристиками. Для демонстрации был создан кристалл кварца, в частности аметист.

В процессе разработки был рассмотрен, проанализирован и реализован один из самых популярных на сегодняшний день методов построения реалистичного трехмерного изображения – метод трассировки лучей на примере моделирования кристалла кварца.

Данный программный продукт может быть использован для демонстрации спектральной и диффузной отражающей способностей различных материалов, а также для визуализации падения теней при различных условиях освещенности.

Реализованный алгоритм обратной трассировки лучей можно модернизировать и оптимизировать, например, проводить тесты на пересечения со сферической оболочкой каждого объекта или группы объектов, использовать принципы суперсэмплинга [13] для получения более реалистичной картины, а также перенести некоторые вычисления на аппаратную реализацию с помощью сторонних графических движков. А благодаря использованию объектно-ориентированного подхода при разработке, в программный продукт может легко быть модифицирован.

В ходе реализации курсового проекта были освоены и закреплены методы объектно-ориентированного анализа и проектирования, а также теоретические знания, полученные во время аудиторных занятий в вузе по дисциплинам профессионального цикла.

# Список источников

1. Волобой А.Г. Средства визуализации распространения световых лучей в задачах проектирования оптических систем / А.Г. Волобой, В.А. Галактионов, А.Д. Жданов, Д.Д. Жданов // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ - 2009. - №4. - С. 1 - 2.
2. Порев В.Н. Компьютерная графика / В.Н. Порев – СПб.:БХВ-Петербург, 2002 – 432 с., ил.
3. Горьков Алексей О цветовых пространствах. RGB. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/181580/– (дата обращения: 28.04.20).
4. Роджерс Д. Алгоритмические основы машинной графики / Д. Роджерс, Пер. с англ. -- М.: Мир, 1989. – 512 с., ил.
5. Станислав Марчевский. Трассировка пути на GPU. [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/132862/>. – (дата обращения: 02.12.2019).
6. Юрий Лаходюк. Raytracing rander на C [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/187720/> – (дата обращения: 28.04.2020).
7. Kevin Suffern. Ray Tracing from the Ground Up. / Kevin Suffern – A K Peters, Ldt, Wellesley, Massachusetts, 2007.
8. Abraham Albert Ungar Барицентрические координаты. [электронный ресурс]. – Режим доступа: <https://web.archive.org/web/20120519111614/http://www.worldscibooks.com/mathematics/7740.html> – (дата обращения: 28.04.2020).
9. Don Herbison-Evans. Solving Quartics and Cubics for Graphics. [Электронный ресурс]. – Режим доступа: <https://app-1492098746.000webhostapp.com/pubs/solving.html>. – (дата обращения: 8.12.2019).
10. Олег Одинцов. Формула Кардано для решения кубического уравнения. [Электронный ресурс]. – Режим доступа: <https://1cov-edu.ru/mat_analiz/funktsii/ratsionalnye/mnogochleny/kubicheskie-uravneniya/formula-kardano/>. – (дата обращения: 8.12.2019).
11. Иродов И.Е. Волновые процессы. Основные законы. / И.Е. Иродов - М.-С.-П.:Физматлит, 1999.
12. Kristof Beets. Supersampling. [Электронный ресурс]. – Режим доступа: <http://www.x86-secret.com/articles/divers/v5-6000/datasheets/FSAA.pdf> – (дата обращения: 28.04.2020).