*Федеральное государственное бюджетное образовательное учреждение*

*высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«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

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Разработка редактора композиций трехмерных графических

примитивов

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Луговой Д.М.

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Майков К.А.

(Подпись, дата) (И.О.Фамилия)

Москва, 2019

Оглавление

[Введение 4](#_Toc26716641)

[1. Аналитический раздел 5](#_Toc26716642)

[1.1 Анализ алгоритмов удаления невидимых линий и поверхностей 5](#_Toc26716643)

[Алгоритм Робертса 6](#_Toc26716644)

[Алгоритм Варнока 6](#_Toc26716645)

[Алгоритм, использующий Z-буфер 7](#_Toc26716646)

[Алгоритм трассировки лучей 8](#_Toc26716647)

[Критерии сравнения 8](#_Toc26716648)

[Выбор оптимального алгоритма 9](#_Toc26716649)

[1.2 Анализ алгоритмов закраски 10](#_Toc26716650)

[Простая закраска 10](#_Toc26716651)

[Закраска по Гуро 10](#_Toc26716652)

[Закраска по Фонгу 10](#_Toc26716653)

[Выбор алгоритма закраски 11](#_Toc26716654)

[2. Конструкторский раздел 12](#_Toc26716655)

[2.1 Структуры данных 12](#_Toc26716656)

[2.2 Общий алгоритм построения изображения 13](#_Toc26716657)

[2.3 Аффинные преобразования 14](#_Toc26716658)

[2.4 Камера и перспективная проекция 15](#_Toc26716659)

[2.5 Отбрасывание невидимых граней 17](#_Toc26716660)

[2.6 Отсечение по пирамиде видимости 17](#_Toc26716661)

[2. 7 Алгоритм Z-буфера 18](#_Toc26716662)

[2.8 Модель освещения 20](#_Toc26716663)

[Модель Ламберта 21](#_Toc26716664)

[Модель Фонга 22](#_Toc26716665)

[2.9 Построение теней 23](#_Toc26716666)

[3. Технологический раздел 25](#_Toc26716667)

[3.1 Выбор средств реализации 25](#_Toc26716668)

[3.2 Структура программы 26](#_Toc26716669)

[3.3 Интерфейс программы 28](#_Toc26716670)

[4. Исследовательский раздел 31](#_Toc26716671)

[4.1 Зависимость времени рендеринга от числа объектов 31](#_Toc26716672)

[4.2 Зависимость времени рендеринга от количества боковых граней 32](#_Toc26716673)

[Заключение 33](#_Toc26716674)

[Список использованных источников 34](#_Toc26716675)

# Введение

Целью данного курсового проекта является разработка редактора композиций, состоящих из графических примитивов, геометрические и спектральные характеристики которых задает пользователь. Для формирования более полного представления о полученной сцене должны присутствовать возможности передвижения камеры и изменения пространственных и спектральных характеристик источника освещения.

В рамках реализации проекта должны быть решены следующие задачи:

* Изучение и анализ существующих алгоритмов компьютерной графики, использующихся для создания реалистичной модели взаимно перекрывающихся объектов, и выбор наиболее подходящих для решения поставленной задачи.
* Проектирование архитектуры программы и ее интерфейса.
* Реализация выбранных алгоритмов и структур данных.
* Проведение исследования на основе разработанной программы.

# 1. Аналитический раздел

## 1.1 Анализ алгоритмов удаления невидимых линий и поверхностей

Основной задачей при построении реалистичного изображения является задача удаления объектов или их частей, которые перекрываются другими объектами, то есть являются невидимыми с точки зрения наблюдателя.

Выделяют две группы алгоритмов для ее решения:

* Алгоритмы, работающие в объектном пространстве. Данные алгоритмы имеют привязку к мировой или физической системе координат. Получаемые результаты ограничиваются только точностью вычислений, однако требуют большого объема вычислений, зависящего от требуемой точности и сложности поступающей на вход сцены. В эту группу входят алгоритм Робертса, алгоритм со списком приоритетов и т.д. [1]
* Алгоритмы, работающие 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 первой группы, и зависит от разрешающей способности экрана и количества объектов на сцене. Основными представителями данной группы являются алгоритм Варнока, алгоритм Z-буфера и алгоритм трассировки лучей. [1]

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

### Алгоритм Робертса

В соответствии с алгоритмом Робертса, вначале из каждого тела удаляются те ребра или грани, которые перекрываются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть перекрывается этими телами. [1]

Преимущество данного алгоритма в том, что математические методы, используемые в нем просты и точны.

Недостатком этого алгоритма является большая, по сравнению с алгоритмами, работающими в пространстве изображения, трудоемкость, которая пропорциональна квадрату количества объектов на сцене.

### Алгоритм Варнока

Основной идеей данного алгоритма является принцип "разделяй и властвуй", заключающийся в разбиении области рисунка на более мелкие подобласти. Для каждой подобласти определяются связанные с ней многоугольники и те из них, видимость которых тривиальна, изображаются на экране. В случае невозможности однозначно определить видимость части многоугольника разбиение области повторяется, и для каждой из вновь полученных подобластей рекурсивно применяется процедура определения видимости. Предполагается, что с уменьшением размеров области количество перекрывающих ее многоугольников сокращается. В результате работы алгоритма получаются области, содержащие не более одного многоугольника, либо разбиение продолжается до тех пор, пока размер области не станет равен одному пикселю. В этом случае для полученного пикселя вычисляется значение глубины каждого многоугольника (координата Z), и визуализируется тот из них, у которого значение этой координаты больше. [1]

Достоинством данного алгоритма является простота реализации и высокая эффективность в случае, если размеры перекрываемых областей невелики.

Его недостатком является использование рекурсивных вызовов, что значительно снижает скорость выполнения в случае больших размеров перекрываемых областей.

### Алгоритм, использующий Z-буфер

Данный алгоритм удаления невидимых поверхностей является одним из самых простых и широко используемых. Его идея заключается в использовании двух буферов: буфера кадра и буфера глубины, также называемого Z-буфером. Буфер кадра используется для хранения интенсивности каждого пикселя в пространстве изображения. В буфере глубины запоминается значение координаты Z (глубины) каждого видимого пикселя в пространстве изображения. В ходе работы алгоритма значение глубины каждого нового пикселя, заносимого в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксель расположен ближе к наблюдателю, чем пиксель, уже находящийся в буфере кадра, то новый пиксель заносится в буфер кадра и производится корректировка Z-буфера: в него заносится глубина нового пикселя. Если же значение глубины нового пикселя меньше, чем хранящееся в буфере, то осуществляется переход к следующей точке. [1]

Основными достоинствами данного алгоритма являются простота его реализации, корректная обработка случаев взаимных пересечений объектов, линейная зависимость трудоемкости от числа объектов на сцене, а также отсутствие необходимости предварительной сортировки объектов по глубине, то есть они могут обрабатываться в произвольном порядке.

К недостаткам данного алгоритма относят необходимость выделения памяти под два буфера, каждый из которых имеет размер равный количеству пикселей на экране.

### Алгоритм трассировки лучей

Идея данного алгоритма заключается в том, что для каждого пикселя картинной плоскости определяется ближайшая к нему грань, для этого через пиксель выпускается луч, находятся все пересечения луча с гранями и среди пересечений выбирается ближайшее. [1]

К достоинствам данного алгоритма можно отнести возможность получения изображения гладких объектов без аппроксимации их примитивами (например, треугольниками). Его вычислительная сложность линейно зависит от сложности сцены. Благодаря отслеживанию пути, пройденного лучом, появляется возможность реализовать глобальную модель освещения, учитывающую отражения и преломления света. Качество полученного изображения получается очень реалистичным, этот метод отлично подходит для создания фотореалистичных сцен.

Главным недостатком алгоритма трассировки является необходимость создавать большое число лучей, проходящих через сцену, которые могут раздваиваться на отраженный и преломленный лучи, для которых все вычисления повторяются. Это приводит к существенному снижению скорости работы программы.

### Критерии сравнения

При реализации редактора композиций из графических примитивов необходимо обеспечить плавную смену кадров при перемещении камеры и изменении объектов, поэтому основными критериями сравнения алгоритмов являются зависимость трудоемкости алгоритма от числа объектов на сцене и использование рекурсивных вызовов, требующее значительных временных затрат.

Так как среднестатистический компьютер имеет 8 ГБ оперативной памяти [3], то факт задействования дополнительной памяти под буферы не является весомым критерием и может быть опущен.

В связи с ограниченностью времени выполнения курсового проекта значимым критерием отбора алгоритма является простота его реализации, а также то, насколько широко используется алгоритм в современном программном обеспечении.

### Выбор оптимального алгоритма

По критерию использования рекурсивных вызовов оптимальными алгоритмами являются алгоритмы Робертса и Z-буфера, однако алгоритм Робертса имеет квадратичную зависимость от числа объектов на сцене, в то время как алгоритм Z-буфера – линейную [1].

Алгоритмы Варнока и Z-буфера имеют наименьшую сложность реализации, так как работают непосредственно с пикселями экрана и не требуют решения задачи нахождения пересечения граней объектов.

Наибольшее распространение в современном программном обеспечении получили алгоритмы трассировки лучей и Z-буфера: алгоритм Z-буфера используется в большинстве графических движков, а алгоритм трассировки лучей используется в ПО для создания фотореалистичных сцен.

Как видно из таблицы 1.1.1, алгоритмом, соответствующим всем выделенным критериям, является алгоритм Z-буфера. Поэтому именно этот алгоритм предлагается для реализации в рамках данного курсового проекта.

Таблица 1.1.1 – Сравнение алгоритмов удаления невидимых линий и поверхностей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Алгоритм Робертса** | **Алгоритм Варнока** | **Алгоритм Z-буфера** | **Алгоритм трассировки лучей** |
| **Использование рекурсивных вызовов** | - | + | - | + |
| **Зависимость трудоемкости от числа объектов (N)** |  | N, но зависит от положения объектов | N | N, но зависит от модели освещения |
| **Сложность реализации** | Средняя | Низкая | Низкая | Средняя |
| **Распространенность в современном ПО** | Низкая | Низкая | Широкая | Широкая |

## 1.2 Анализ алгоритмов закраски

Существуют три основных алгоритма, позволяющих закрасить полигональную модель: простая закраска, закраска по Гуро и закраска по Фонгу.

### Простая закраска

Суть данного алгоритма заключается в том, что для каждой грани объекта находится вектор нормали, и с его помощью в соответствии с выбранной моделью освещения вычисляется значение интенсивности, с которой закрашивается вся грань. [2]

Данный метод закраски обладает большим быстродействием, однако все пиксели грани имеют одинаковую интенсивность, и сцена выглядит нереалистично.

### Закраска по Гуро

Данный метод отличается от простой закраски тем, что разные точки грани закрашиваются с разными значениями интенсивности. Для это в каждой вершине грани находится вектор нормали и вычисляется значение интенсивности. Затем найденные значения интенсивности интерполируются по всем точкам грани. [2]

С помощью этого метода получаются достаточно реалистичные изображения, однако все объекты кажутся матовыми.

### Закраска по Фонгу

Закраска Фонга по своей идее похожа на закраску Гуро, но ее отличие состоит в том, что в методе Гуро по всем точкам полигона интерполирутся значения интенсивностей, а в методе Фонга – вектора нормалей, и с их помощью для каждой точки находится значение интенсивности. [2]

Эта закраска требует больших вычислительных затрат, чем предыдущие, однако она позволяет достигнуть лучшей локальной аппроксимации кривизны поверхности и, следовательно, с ее помощью получается более реалистичное изображение.

### Выбор алгоритма закраски

На рисунке 1.2.1 наглядно показаны различия рассмотренных методов закраски.



Рисунок 1.2.1 – Методы закраски (слева направо: плоская, Гуро, Фонга)

Алгоритм закраски Фонга требует большего числа вычислений по сравнению с другими, однако он дает наиболее реалистичное изображение, в частности зеркальных бликов. В данном курсовом проекте пользователь должен иметь возможность задавать спектральные характеристики объектов, в том числе коэффициенты зеркального отражения и блеска поверхности, поэтому для получения более реалистичной сцены используется метод закраски Фонга.

# 2. Конструкторский раздел

## 2.1 Структуры данных

Для формализации общего алгоритма синтеза изображения в данной программе, необходимо ввести определения использующихся в ней структур данных:

1. Сцена представляет собой список с произвольным числом моделей, объект камеры и объект источника освещения.
2. Модель включает в себя следующие данные:

* Массив вершин фигуры,
* Массив полигонов фигуры,
* Массив векторов нормалей к вершинам,
* Коэффициенты отражения и блеска поверхности,
* Цвет поверхности,
* Матрицы аффинных преобразований.

1. Камера содержит:

* Положение в пространстве,
* Значения углов тангажа и рыскания,
* Систему координат камеры, задаваемую тремя ортогональными векторами,
* Угол обзора и соотношение сторон экрана,
* Границы пирамиды видимости.

1. Источник освещения включает:

* Положение в пространстве,
* Цвет источника.

## 2.2 Общий алгоритм построения изображения

Вход: Геометрические характеристики моделей, камеры, источника освещения, спектральные характеристики моделей и источника освещения.

Выход: Построенная в буфере кадра сцена.

Общая схема алгоритма генерации изображения представлена на рисунке 2.2.1.



Рисунок 2.2.1 – Общая схема алгоритма синтеза изображения

## 2.3 Аффинные преобразования

В представленном алгоритме синтеза изображения первым этапом преобразования полигона перед его растеризацией является переход из пространства модели в мировое пространство. Это действие осуществляется с помощью матриц аффинных преобразований [5]. В данном курсовом проекте над объектами возможно произвести следующие операции:

* Поворот вокруг координатных осей

Поворот описывается углом θ и осью вращения. Матрицы поворота имеют вид:

* + Вокруг оси OX:
  + Вокруг оси OY:
  + Вокруг оси OZ:
* Перенос вдоль координатных осей

Перенос в трехмерном пространстве задается значениями переноса вдоль осей OX, OY, OZ - dx, dy, dz соответственно. Матрица переноса имеет вид:

## 2.4 Камера и перспективная проекция

Для перемещения по сцене используется камера, задаваемая точкой положения в пространстве, пирамидой видимости и собственной системой координат, которая состоит из трёх ортогональных векторов.

Пусть

* P – положение камеры,
* D - вектор взгляда,
* U – вектор вверх,
* R - вектор вправо.

Переход в пространство камеры осуществляется в 2 этапа:

1. Перенос полигона в отрицательную сторону от камеры на   
   расстояние P с помощью матрицы переноса [5]:
2. Преобразование полигона к системе координат камеры при помощи матрицы поворота [5]:

Для удобства пользователя в данном курсовом проекте управление системой координат камеры производится с помощью изменения углов Эйлера.

Пусть

* pitch – тангаж,
* yaw – рыскание.

Тогда вектор направления камеры можно вычислить по формулам:

После перехода в пространство камеры необходимо спроецировать полигон на картинную плоскость. В данном курсовом проекте используется перспективная проекция.

Пусть

* zoomx – приближение по X,
* zoomy – приближение по Y,
* ratio – соотношение сторон экрана,
* far – расстояние от камеры до задней грани пирамиды видимости,
* near – расстояние от камеры до передней грани пирамиды видимости,
* fov – вертикальный угол обзора.

Увеличение объектов по координатам X и Y можно вычислить по формулам [2]:

Для перехода в пространство отсечения используется матрица перспективной проекции [2]:

## 2.5 Отбрасывание невидимых граней

С помощью отбрасывания нелицевых граней моделей при построении изображения можно существенно сократить временные затраты, так как не полигоны, невидимые по отношению к камере, растеризоваться не будут.

Пусть

* – вектор нормали к грани модели,
* – вектор от камеры до любой точки грани.

Для определения видимости грани используется формула:

## 2.6 Отсечение по пирамиде видимости

Отсечение полигона по пирамиде видимости решает две задачи:

1. Удаление полигонов, лежащих за камерой, но проецируемых на картинную плоскость в перевернутом виде.
2. Сокращение времени на растеризацию полигонов за счет удаления их невидимых частей.

Для выполнения операции отсечения в данном курсовом проекте использовался алгоритм Сазерленда-Ходжмана [1]. Согласно рисунку 2.6.1, его идея состоит в том, чтобы последовательно отсекать полигон относительно каждой грани пирамиды видимости до тех пор, пока полигон не будет отсечен полностью либо пока все грани не будут пройдены.



Рисунок 2.6.1 – Схема алгоритма Сазерленда-Ходжмана

## 2. 7 Алгоритм Z-буфера

Для растеризации треугольного полигона сначала необходимо найти ограничивающий его прямоугольник, в котором это полигон содержится. Это делается для того, чтобы не тратить время на растеризацию пикселей, не являющихся частью полигона. Затем для каждого пикселя ограничивающего прямоугольника находятся его барицентрические координаты относительно вершин полигона.

Пусть

* A, B, C – вершины полигона,
* P – пиксель внутри ограничивающего прямоугольника.

Площадь треугольника можно найти по следующей формуле:

Тогда барицентрические координаты пикселя равны [2]:

В случае, если хоть одна из барицентрических координат отрицательна, то пиксель лежит вне полигона. Если пиксель лежит внутри треугольника, то найти значение его глубины можно по следующей формуле [5]:

Затем производится сравнение значения глубины точки со значением глубины из Z-буфера. Если глубина пикселя меньше, значит он лежит ближе к камере и должен быть растеризован. Происходит вычисление интенсивности пикселя, его значение заносится в буфер кадра, а в Z-буфер заносится значение глубины пикселя. Полная схема алгоритма Z-буфера представлена на   
рисунке 2.7.1.



Рисунок 2.7.1 – Схема алгоритма Z-буфера

## 2.8 Модель освещения

Все модели освещения делятся на два вида: глобальные и локальные. Глобальные модели учитывают возможности отражения и преломления света от объектов, не являющихся прямыми источниками освещения, поэтому они требуют значительных затрат.

Существуют более простые, локальные модели освещения, которые учитывают только свет от источника. Именно этот тип моделей используются в данном курсовом проекте. Выделяют две основные модели локального освещения: модель Ламберта и модель Фонга.

### Модель Ламберта

Данная модель является простейшей моделью освещения, так как учитывает только идеальное диффузное отражение света от тела [2]. На рисунке 2.8.1 показано, что согласно этой модели, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. При этом положение наблюдателя не имеет значения, т.к. диффузно отраженный свет рассеивается равномерно по всем направлениям.

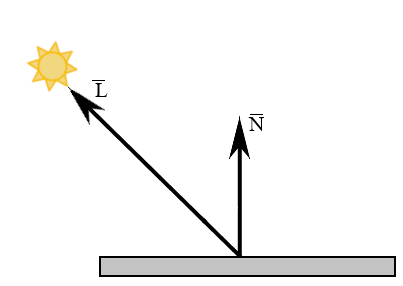


Рисунок 2.8.1 – Модель освещения Ламберта

Пусть

* – вектор от точки до источника,
* – вектор нормали,
* I – результирующая интенсивность света в точке,
* I0 – интенсивность источника,
* Kd – коэффициент диффузного освещения.

Формула расчёта интенсивности имеет следующий вид:

Из формулы (2.8.1) следует главный недостаток модели Ламберта – одинаковая интенсивность во всех точках, принадлежащих одной грани.

### Модель Фонга

Модель Фонга включает в себя модель Ламберта, но также учитывает тот факт, что кроме диффузного отражения на материале может появляться блик, то есть зеркальное отражение [2]. Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части как показано на рисунке 2.8.2.

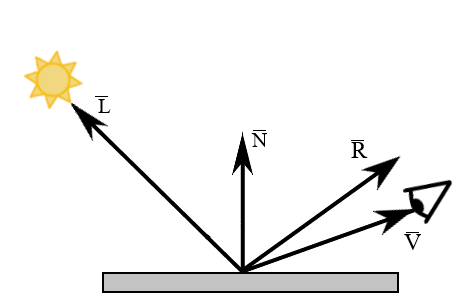


Рисунок 2.8.2 – Модель освещения Фонга

Отраженная составляющая освещенности в точке зависит от того, насколько близки направления на наблюдателя и отраженного луча.

Также в модели освещения Фонга используется понятие рассеянного освещения – это константа, которая прибавляется к интенсивности в точке для придания сцене большей реалистичности.

Таким образом, согласно модели Фонга интенсивность к точке складывается из 3 компонент: диффузной, зеркальной и рассеянной, как показано на рисунке 2.8.3.

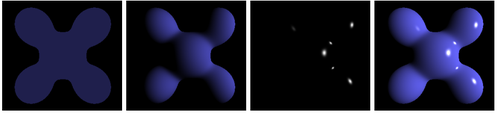


Рисунок 2.8.3 – Составляющие модели Фонга (слева направо: рассеянная, диффузная, зеркальная, суммарная)

Пусть

* – вектор отраженного луча,
* – вектор от точки до наблюдателя,
* Ip – интенсивность рассеянного освещения,
* кз – коэффициент зеркального освещения,
* ка – коэффициент рассеянного освещения,
* α - коэффициент блеска.

Формула для расчета интенсивности для модели Фонга имеет вид:

## 2.9 Построение теней

Для генерации теней в данном курсовом проекте используется метод теневых карт. Он основан на алгоритме Z-буфера: сначала камера устанавливается на позицию источника освещения и происходит построение сцены с занесением глубин в теневой буфер (карту теней). Все точки в этом буфере видимы с точки зрения источника, соответственно они освещены. После этого изображение строится уже с позиции наблюдателя, при этом для каждой точки буфера кадра расстояние до источника освещения сравнивается со значением из теневой карты: если это расстояние больше полученного значения из карты, то точка лежит в тени и отображается только с учетом рассеянного света. Иначе точка освещена и происходит расчет ее интенсивности со всеми составляющими освещения. [4]

На рисунке 2.9.1 представлены случаи затененной и освещенной точек.

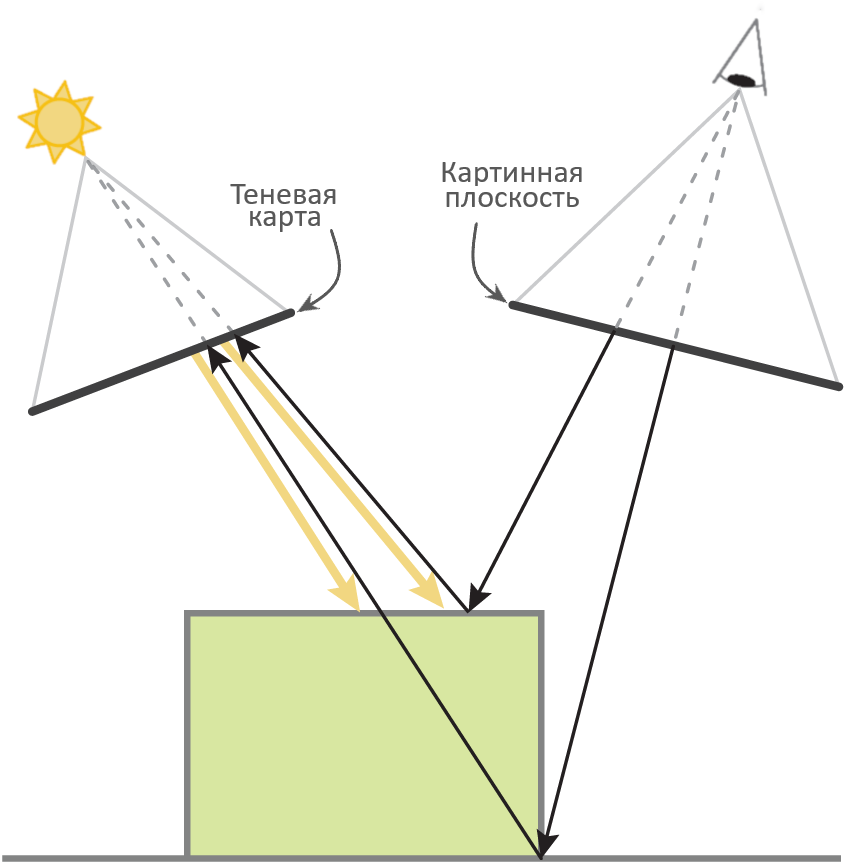


Рисунок 2.9.1 – Использование карты теней

Так как используемый источник освещения является точечным, то есть излучающим свет во всех направлениях, то для генерации теней необходимо произвести построение сцены 6 раз для разных направлений камеры: вдоль и против осей OX, OY и OZ. В результате образуется так называемая кубическая карта теней, в которой содержатся значения глубин всех освещенных участков сцены.

Этот способ генерации теней требует значительного числа вычислений, однако они выполняются только при перестроении сцены, то есть при изменении положения камеры заново вычислять значения в карте теней не нужно. Таким образом метод кубической карты теней позволяет сохранить плавность подачи картинки при перемещении по сцене, поэтому используется в данном курсовом проекте.

# 3. Технологический раздел

## 3.1 Выбор средств реализации

Для написания данного курсового проекта был использован язык C++. Этот язык поддерживает объектно-ориентированную модель разработки, что позволяет четко структурировать программу и легко модифицировать отдельные ее компоненты независимо от других. Язык C++ позволяет эффективно использовать ресурсы системы благодаря широкому набору функций и классов из стандартной библиотеки.

В качестве среды разработки был использован Qt Creator 4.10.2. Он обладает всем необходимым функционалом для написания, профилирования и отладки программ и создания графического пользовательского интерфейса. Данная среда поставляется вместе с фреймворком Qt 5.13.2 [6], классы и функции которого также были использованы при написании данного курсового проекта.

Критическим по скорости участком программы является попиксельное вычисление глубины для Z-буфера и интенсивности для буфера кадра. Эта часть была разбита на отдельные потоки при помощи класса стандартной библиотеки std::thread. Распараллеливание программы позволяет эффективно использовать свободные ресурсы системы и существенно сократить время на необходимые вычисления.

## 3.2 Структура программы

На рисунке 3.2.1 представлена схема разработанных классов.

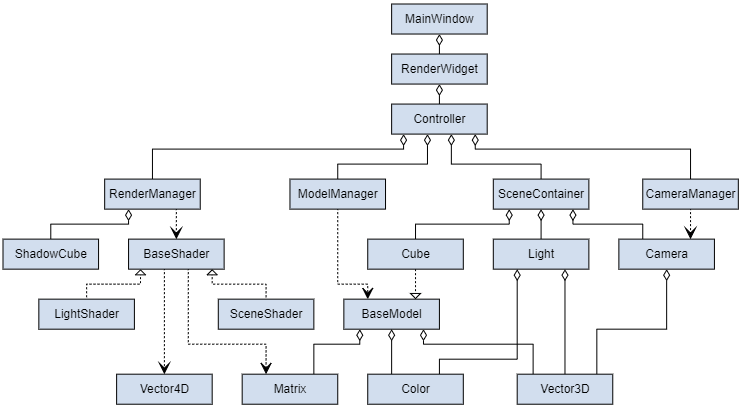


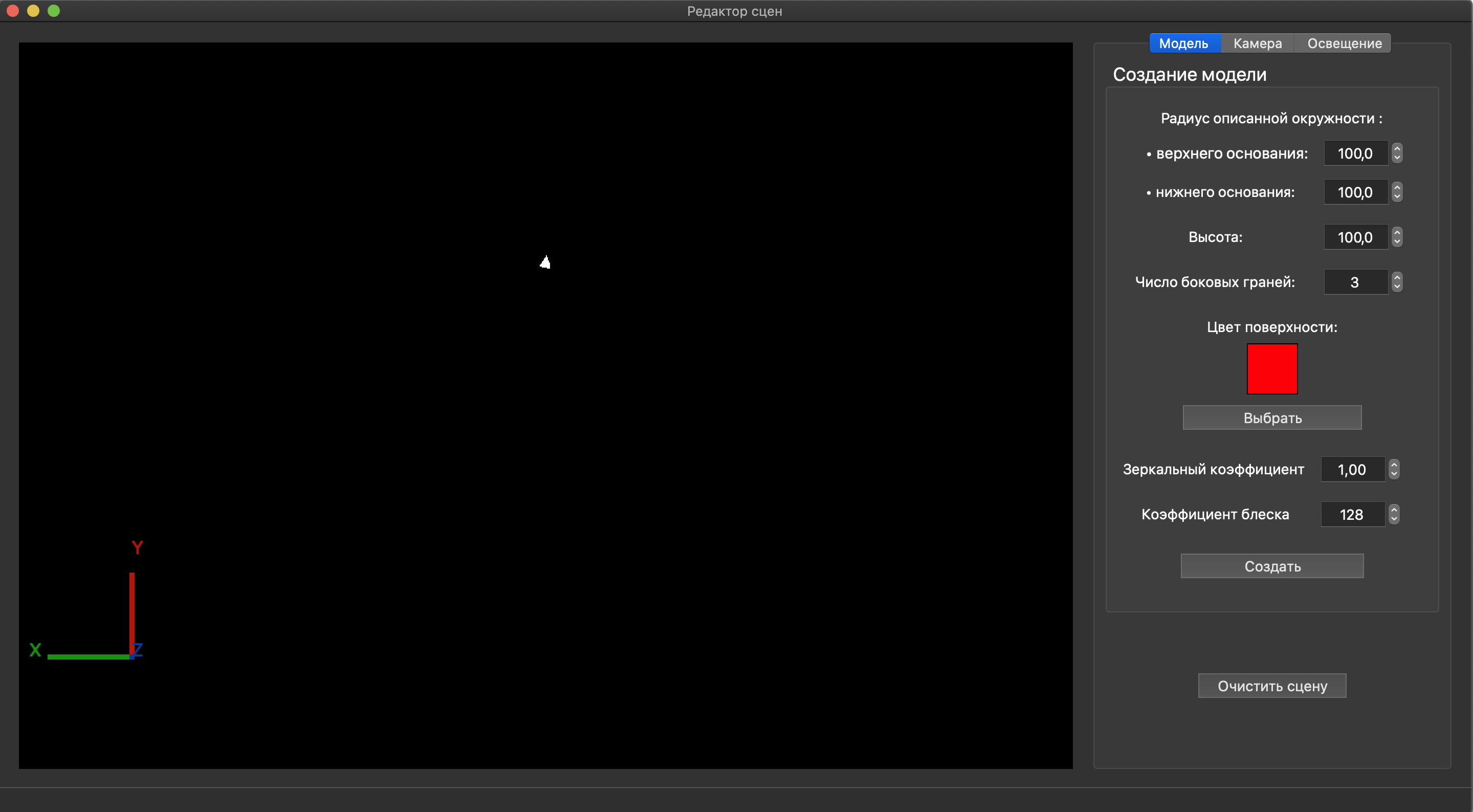
Рисунок 3.2.1 – Схема классов программы

Разработанная программа состоит из следующих классов:

* Базовые математические классы
  + Matrix – класс матриц;
  + Vector3D – класс векторов трехмерного пространства;
  + Vector4D – класс векторов в однородных координатах.
* Классы для работы с моделями
  + BaseModel – базовый класс моделей с возможностью перемещения по сцене и изменения спектральных характеристик;
  + Cube – класс куба с возможностью изменения радиусов верхнего и нижнего оснований, высоты и числа боковых граней.
* Вспомогательные классы сцены
  + Camera – класс камеры с возможностью перемещения по сцене;
  + Light – класс источника освещения с возможностью перемещения по сцене и изменения цвета.
* Классы шейдеров
  + BaseShader – базовый класс шейдера для преобразования вершин и вычисления интенсивности пикселей;
  + SceneShader – класс шейдера для визуализации сцены;
  + LightShader – класс шейдера для визуализации источника освещения.
* Классы управления сценой
  + ModelManager – класс для осуществления аффинных преобразований;
  + CameraManager – класс для осуществления видовых и перспективных преобразований;
  + RenderManager – класс для визуализации сцены;
  + SceneContainer – класс управления объектами сцены.
* Классы интерфейса
  + Controller – класс для взаимодействия управляющих классов с классами интерфейса;
  + RenderWidget – класс виджета сцены для вывода буфера кадра;
  + MainWindow – класс главного окна сцены.

## 3.3 Интерфейс программы

При запуске программы на виджете сцены отображается источник освещения и система координат в левом нижнем углу. Источник освещения отображается как пирамида цвета источника. Для управления сценой используются разделы в правой части экрана (рисунок 3.3.1).

  
Рисунок 3.3.1 – Интерфейс программы

Для создания модели пользователю необходимо ввести в разделе “Модель” ее геометрические и спектральные характеристики: радиусы верхнего и нижнего оснований, высоту, число боковых сторон, цвет поверхности, коэффициент блеска и зеркальный коэффициент   
(рисунок 3.3.1 а). При нажатии правой кнопкой мыши по модели на виджете сцены в разделе “Модель” отобразится секция с редактированием модели (рисунок 3.3.1 б). Для возврата в секцию создания необходимо нажать правой кнопкой мыши по пустой части виджета сцены.

Для перемещения по сцене используются клавиши: W – вперед, S – назад, A – влево, D – вправо. Для поворота камеры необходимо зажать левую кнопку мыши и переместить мышь.

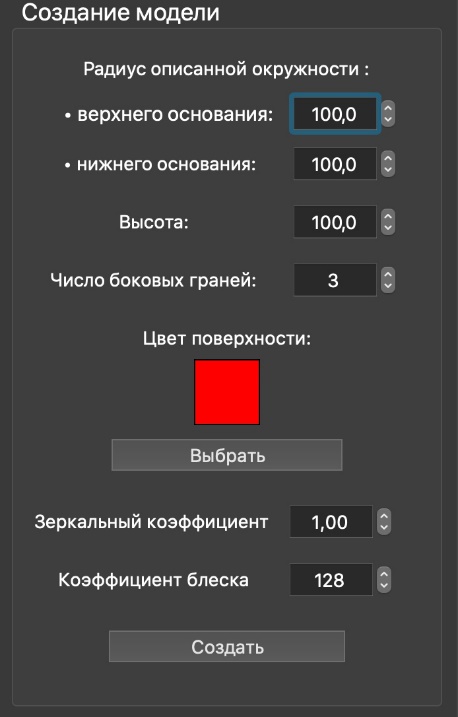
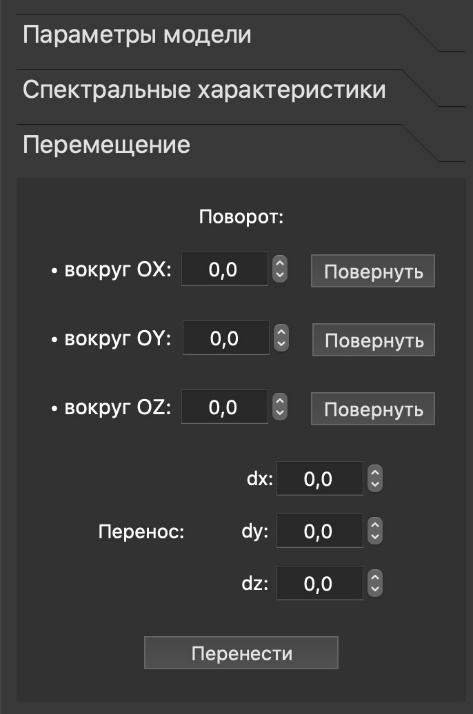
 а)  б)

Рисунок 3.3.1 – Раздел “Модель”: Создание (а) и редактирование (б)

В разделе “Камера” находятся следующие параметры: расстояния до ближней и дальней плоскостей пирамиды видимости, угол обзора, позиция камеры в пространстве, углы поворота, скорость перемещения и чувствительность мыши (рисунок 3.3.2).

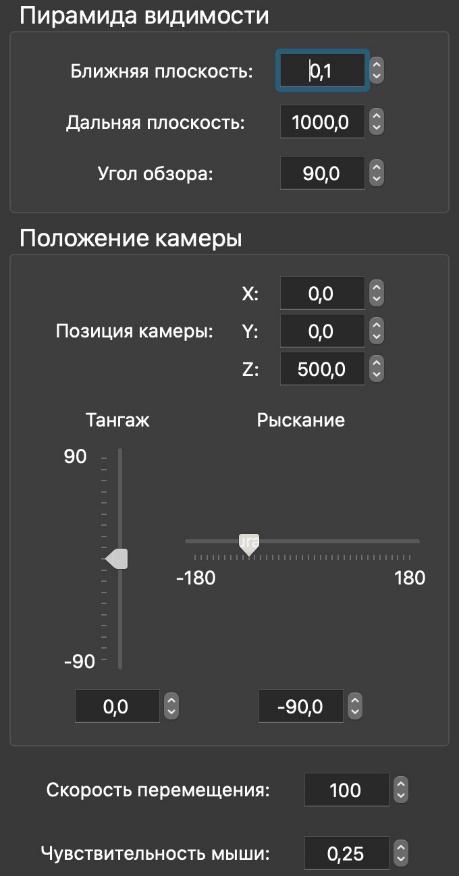


Рисунок 3.3.2 – Раздел “Камера”

Раздел “Освещение” содержит положение и цвет источника освещения (рисунок 3.3.3 а). При выборе цвета появляется диалоговое окно с палитрой цветов (рисунок 3.3.3 б)

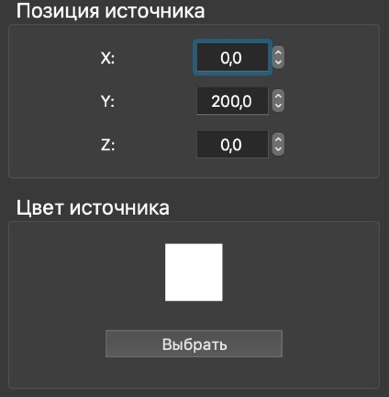
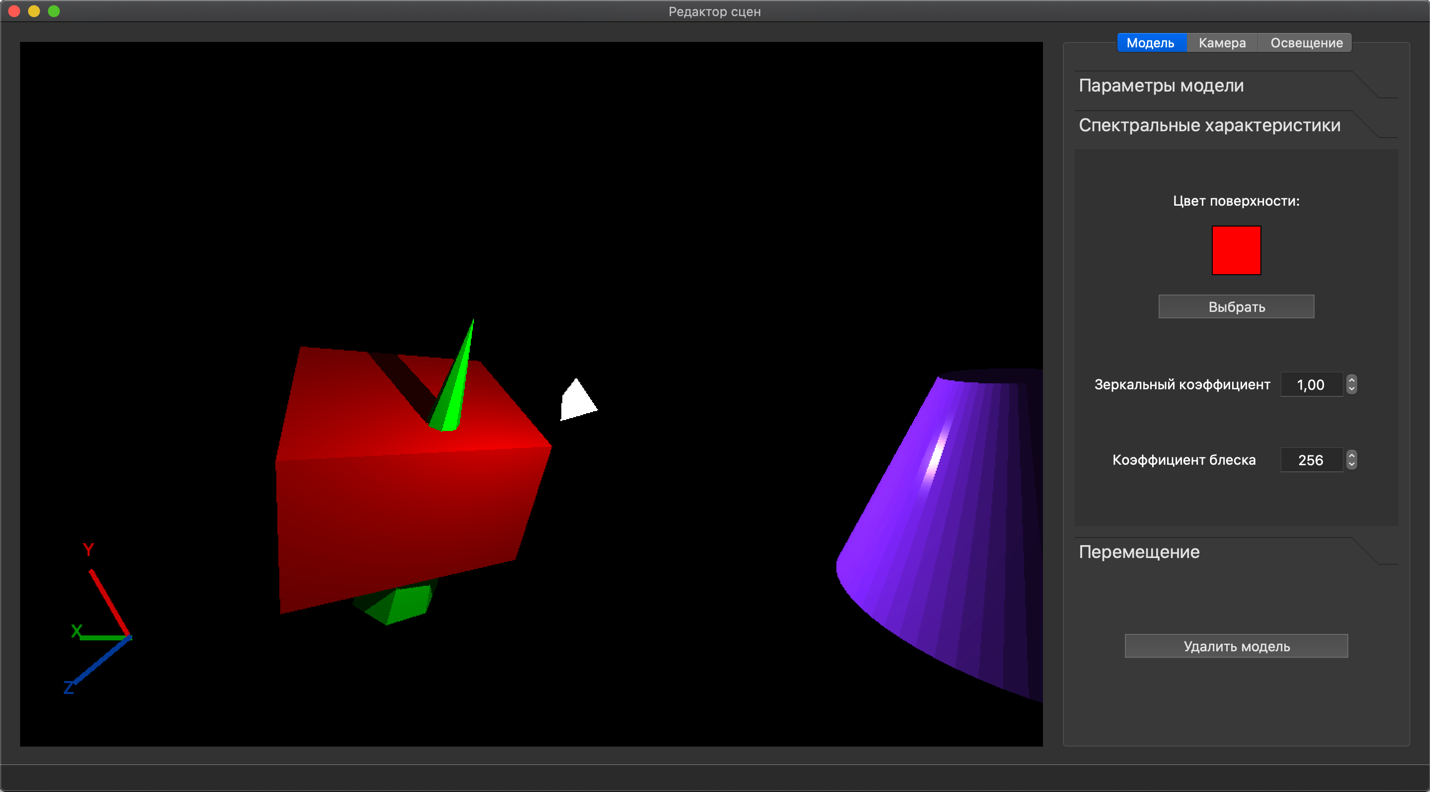
а)  б)

Рисунок 3.3.3 – Раздел “Освещение” (а) и окно выбора цвета (б)

На рисунке 3.3.4 приведен пример работы программы.

  
Рисунок 3.3.4 – Тестовая сцена

# 4. Исследовательский раздел

При исследовании временных характеристик разработанной программы использовался компьютер на базе 4-х ядерного процессора Intel Core i5 частотой 2300 MГц с 8 ГБ оперативной памяти типа LPDDR3 частотой 2133 МГц. Для замеров времени была использована библиотека chrono.

## 4.1 Зависимость времени рендеринга от числа объектов

Для исследования зависимости времени рендеринга изображения от числа объектов на сцене, использовались объекты с фиксированным числом граней, каждый объект имел освещенную и затененную части. Количество объектов менялось на сцене от 100 до 1000 с шагом 100, были рассмотрены случаи для 3-х, 6-ти и 9-тигранных объектов. Результаты проведенного исследования представлены на рисунке 4.1.1.

Рисунок 4.1.1 – График зависимости времени отрисовки от числа объектов

Как видно из графика, время рендеринга сцены зависит от количества объектов линейно, независимо от числа их боковых граней.

## 4.2 Зависимость времени рендеринга от количества боковых граней

Следующим этапом исследования разработанной программы является исследование зависимости времени построения сцены от числа боковых граней объектов при фиксированном количестве объектов. В ходе эксперимента количество боковых граней менялось от 50 до 500 с шагом 50, при этом были рассмотрены случаи для 2-х объектов с равным числом граней и для 2-х объектов, у одного из которых число граней менялось, а у другого было фиксировано. Результаты эксперимента показаны на рисунке 4.2.1.

Рисунок 4.4.2 - График зависимости времени отрисовки от числа боковых граней

Из проведенного эксперимента можно сделать вывод, что время рендеринга сцены линейно зависит от числа боковых граней объектов, при этом для 2-х объектов время растет вдвое быстрее, если количество граней у обоих увеличивается одновременно, чем в случае фиксированного количества граней у одного из них. В случаях фиксированного количества граней время растет одинаково, постоянная разница во времени объясняется разным числом граней у фиксированного объекта.

# Заключение

В результате проделанной работы был разработан программный продукт, позволяющий создавать и редактировать композиции из трехмерных графических примитивов.

В процессе разработки были рассмотрены, проанализированы и реализованы основные алгоритмы построения реалистичного трехмерного изображения: алгоритмы удаления невидимых линий, методы закраски, модели освещения, алгоритмы создания динамических теней.

Разработанный программный продукт предоставляет широкие возможности настройки геометрических и спектральных характеристик объектов, положения камеры и положения и цвета источника освещения. Были добавлены возможности изменения ориентации и положения камеры с помощью манипуляторов клавиатура и мышь.

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.

Быстродействие низкоуровневых частей программы, отвечающих за растеризацию полигонов, может быть улучшено за счет замены их программной реализации на поддерживаемую аппаратно из интерфейсов графических движков, например, OpenGL или DirectX.

Благодаря использованию объектно-ориентированного подхода при разработке, в программный продукт может легко быть модифицирован добавлением других типов моделей и шейдеров.

Программа курсовой работы полностью соответствует поставленному техническому заданию.

# Список использованных источников

1. Роджерс Д. Математические основы машинной графики. / Роджерс Д., Адамс Дж. – М.: Мир, 1989. – 512с.
2. Fletcher Dunn, Ian Parberry. 3D Math Primer for Graphics and Game Development / Fletcher Dunn, Ian Parberry. – Second edition. – Taylor and Francis Group, LLC, 2011. – 845с.
3. Данные об оборудовании пользователей Steam [Электронный ресурс]. – Режим доступа: URL: https://store.steampowered.com/hwsurvey/. – (Дата обращения: 25.10.2019).
4. Eric Lengyel. Mathematics for 3D Game Programming and Computer Graphics / Eric Lengyel. – Third edition. - Course Technology, a part of Cengage Learning, 2012. – 566с.
5. Learn computer graphics from scratch [Электронный ресурс]. – Режим доступа: URL: https://www.scratchapixel.com. – (Дата обращения: 06.09.2019).
6. Шлее М. Qt 5.10. Профессиональное программирование на C++ / Шлее М. – СПб: БХВ-Петербург, 2018. – 1072с.