=

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 Информатики и систем управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Моделирование физически точных лучей света, проходящих через решётчатую поверхность и листву деревьев»

Студент ИУ7-54Б _____ Воякин А.Я.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Корниенко В.В.
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2020г.

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)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Компьютерная графика

Студент группы ИУ7-54б

Воякин Алексей Янович
(Фамилия, имя, отчество)

Тема курсового проекта Моделирование физически точных лучей света, проходящих через решётчатую поверхность и листву деревьев

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 программное обеспечение для визуализации ярких лучей, проходящих через решётчатую поверхность и листву деревьев. Реализовать интерфейс, который позволит выбирать из предложенного набора препятствия, представленные в виде объемных моделей, для искусственного источника света. Программный продукт должен предоставлять возможность размещения источников света, а также возможность просмотра сцены с фиксированного положения наблюдателя.

Оформление курсового проекта:

Расчетно-пояснительная записка на __24_ листах формата А4.
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 _____ В.В. Корниенко
(Подпись, дата) (И.О.Фамилия)

Студент _____ А.Я. Воякин
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

Введение.

Рендер (Рендеринг) — это процесс создания финального изображения или последовательности из изображений на основе двухмерных или трехмерных данных. Данный процесс происходит с использованием компьютерных программ и зачастую сопровождается трудными техническими вычислениями, которые ложатся на вычислительные мощности компьютера или на отдельные его комплектующие части.

Процесс рендеринга так или иначе присутствует в разных сферах профессиональной деятельности, будь то киноиндустрия, индустрия видеоигр или же видеоблогинг. Зачастую, рендер является последним или предпоследним этапом в работе над проектом, после чего работа считается завершённой или же нуждается в небольшой постобработке. Также стоит отметить, что нередко рендером называют не сам процесс рендеринга, а скорее уже завершённый этап данного процесса или его итоговый результат[2].

Если мы говорим о задаче рендеринга изображения, то чаще всего имеем в виду рендеринг в трёхмерной графике, так как это задача является востребованнее всего в этой сфере. Это связано с тем, что как такового трехмерного измерения в компьютерной графике не существует и работа ведется с двухмерным изображением.

Цель данной работы – реализовать построение трехмерной сцены и визуализацию световых столбов.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена

2) выбор существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену;

3) реализация данных алгоритмов для создания трехмерной сцены;

4) разработать программное обеспечение, которое позволит отобразить трехмерную сцену и визуализировать световые столбы.

1 Аналитическая часть

В данной части производится анализ и сравнение методов и алгоритмов предметной области, необходимых для выполнения поставленной задачи.

1.1 Постановка задачи

Разработать программу моделирования трёхмерного объекта и визуализации световых столбов, проходящих через листву/решётчатую поверхность. Каждый объект хранится в файле формата OBJ. Отображение каждого объекта на сцене происходит при помощи алгоритма z-буфера.

Процесс визуализации световых столбов происходит за счёт построения множества копий объекта увеличенного масштаба и сдвига на необходимую величину по обеим плоскостям. Можно выделить следующие задачи:

- реализация процедуры чтения из файлов формата obj;
- реализация создания объектов сцены;
- реализация алгоритма z-буфера;
- реализация метода тонирования Гуро для закраски объекта;
- реализация камеры и источника света;
- реализация пользовательского интерфейса.
- реализация построения копий объекта.

1.2 Алгоритм Брезенхема

Алгоритм Брезенхема построения отрезков выбирает оптимальные растровые координаты для представления отрезка, другими словами, данный алгоритм позволяет получить приближение идеальной прямой точками растровой сетки.

Основная идея алгоритма базируется на расстоянии между действительным положением отрезка и ближайшими координатами сетки. Это расстояние называют ошибкой[5].

В данном алгоритме проверяется принадлежность проекции к оси x или y. На какую ось проекция будет больше, на ту ось и смещается пиксель. По другой оси смещение на один пиксель происходить лишь в том случае, когда линия отклонилось от оси более чем на половину пикселя.

Преимущество:

• высокая скорость растрирования вектора.

Недостатки:

• при маленьком разрешении экрана наблюдается эффект ступенчатости;

• округление значений: действительные величины преобразуются в ближайшие целые числа.

1.3 Удаление невидимых линий

Удаление невидимых линий и поверхностей не самая простая задача в компьютерной графике: на данный момент не существует такого алгоритма, который работал бы быстро и с высокой детализацией результата. Под детализацией понимается: просчет теней, прозрачности, фактуры, отражения и преломления света.

Алгоритмы удаления невидимых линий и поверхностей делятся на два вида:

1. алгоритмы, работающие в пространстве изображения;
2. алгоритмы, работающие в объектном пространстве.

В алгоритмах первого вида находятся ближайшие точки сцены к наблюдателю и отображаются. Такой подход к удалению невидимых линий и поверхностей не требует высокой точности вычислений, поэтому сложности таких алгоритмов не превышает $O(c * n)$[4], где c - число пикселей, n - количество объектов. В алгоритмах данного вида точность ограничивается разрешающей способностью экрана.

В алгоритмах второго вида все действия происходят в физической системе координат, в ней описаны данные объекты. Сложность таких алгоритмов в среднем для n объектов $O(n^2)$[4]. Однако, медленная скорость работы обусловлена точными результатами. Качество, полученных изображений, не будет зависеть от разрешений экрана.

В рамках курсового проекта необходимо было выбрать такой алгоритм, который бы быстро удалял задние линии и поверхности, поэтому выбор осуществлялся между следующими алгоритмами:

1. алгоритм художника;
2. алгоритм Z-буфера;
3. алгоритм A-буфера.

1.3.1 Алгоритм художника

Алгоритм художника является простейшим вариантом решения задачи об удалении невидимых линий и граней в компьютерной графике.

Название «алгоритм художника» был использован в связи с тем, что его реализация схожа с техникой, которой пользуются многие живописцы: сначала рисуют дальние части сцены, затем ближние части. Так и в

алгоритме художника: многоугольники сортируются по координате z по возрастанию, в начале расположены те грани, которые расположены ближе всего к наблюдателю. После того, как грани отсортированы по расстоянию от наблюдателя, их начинают отрисовывать на сцене, начиная с самых дальних.

Главным недостатком данного алгоритма является неправильное вычислений задних граней при определенных сценариях: когда один многоугольник является относительно одной фигуры неэкранируемым, а относительно другой является экранируемым. Данную проблему можно наблюдать на рисунке 1.4.1.1.

Также этот алгоритм работает медленно, из-за отрисовки ненужных задних граней, которые по итогу будут закрашены ближней гранью к наблюдателю. И если задних граней будет много, то данный алгоритм будет весьма трудозатратным.

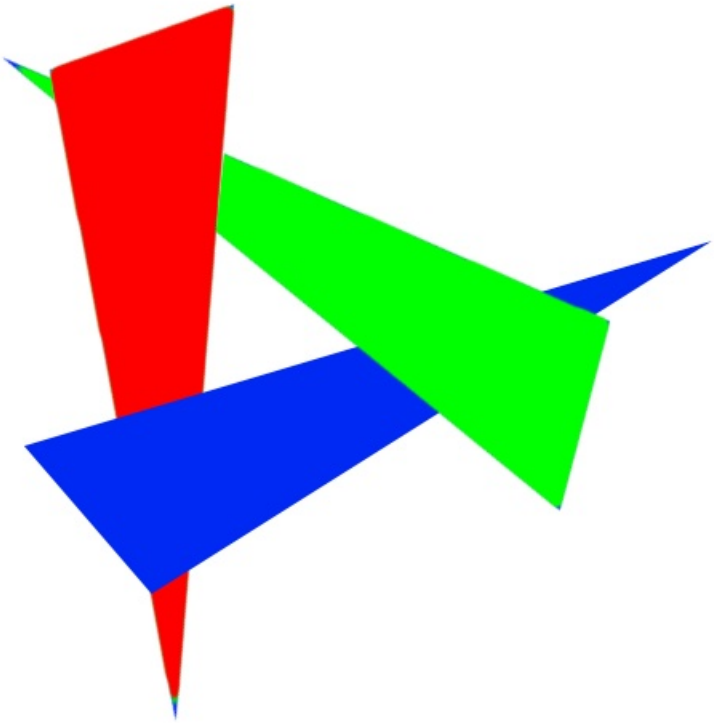

Рис. 1.4.1.1: Ошибочный вариант расположения фигур для алгоритма художника

1.3.2 Алгоритм Z-буфера

Алгоритм Z-буфера заключается в том, что помимо буфера кадра: место, где содержится информация о цвете для каждого пикселя в пространстве изображения, у нас также будет z-буфер, в котором будет хранится координата z, другими словами, глубина, для каждого пикселя.

Во время работы алгоритма, значение z-буфера или глубины сравнивается со значением, которое нужно занести в z-буфер. Если сравнение показывает, что новое значение расположено ближе к наблюдателю по оси z, то новый пиксел заносится в буфер кадра и z-буфер корректируется новым значением z. Визуализацию работы алгоритма можно посмотреть на рисунке 1.4.2.1.

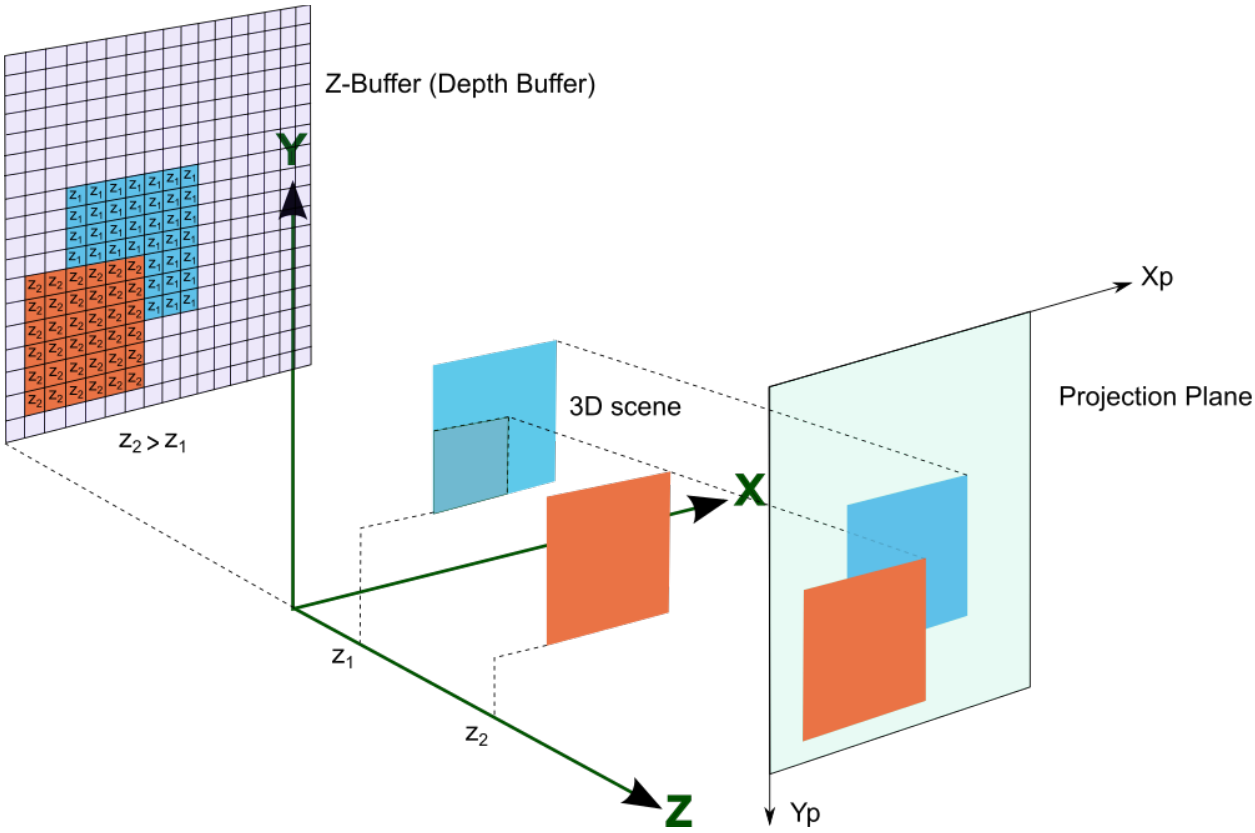

Рис. 1.4.2.1: Работа алгоритма z-буфер

Данный алгоритм является быстродействующим, из-за того, что в нем не выполняются лишние операции, как это было с отрисовкой задних граней в алгоритме художника.

Однако, есть два существенных недостатка: первый заключается в том, что для хранения буфера кадра и z-буфера нужно выделить большой объем памяти[1]. Второй недостаток заключается в том, что пиксели в буфер кадра заносятся в произвольном порядке, поэтому появляются трудности с реализацией эффектов прозрачности или просвечивания. Второй недостаток можно не учитывать, если заранее известно, из каких материалов состоит объект.

1.3.3 Алгоритм A-буфера

Данный алгоритм иногда называют модификацией z-буфера. Данный алгоритм специально предназначен для осуществления эффекта усреднения

по области, прозрачности и наложения полигонов. На замену z-буфера приходит буфер накопления, который помимо координаты z хранит данные о поверхности. Пример работы можно увидеть на рисунке 1.4.3.1.

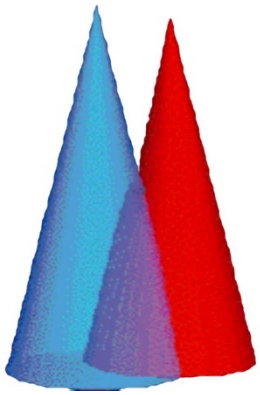

Рис. 1.4.3.1: Работа алгоритма a-буфер

1.4 Метод тонирования Гуро

Метод тонирования Гуро основан на интерполяции интенсивности, данный метод подход к закраске объекта, позволяет устранить дискретность изменения интенсивности.

Процесс закраски Гуро можно разделить на четыре этапа:

1. вычисление нормалей ко всем полигонам;
2. определение нормали в вершинах путем усреднения нормалей по всем полигональным граням;
3. вычисление значений интенсивности в вершинах, используя нормали в вершинах и применяя произвольный метод закраски;
4. закрашивание многоугольника путем линейной интерполяции значений интенсивности в вершинах (сначала вдоль каждого ребра, а затем между ребрами).

Интерполяция интенсивностей работает следующим образом: для всех ребер запоминается начальная интенсивность, изменение интенсивности при каждом шаге по координате y. Затем, заполнение видимого интервала производится путем интерполяции между значениями интенсивности на ребрах, ограничивающих интервал (рис. 1.5.1).

На рисунке 1.5.2 приведены три формулы вычисления интенсивности цвета для каждой границы строки от Ia до Ib, приведенной на рисунке 1.5.1.

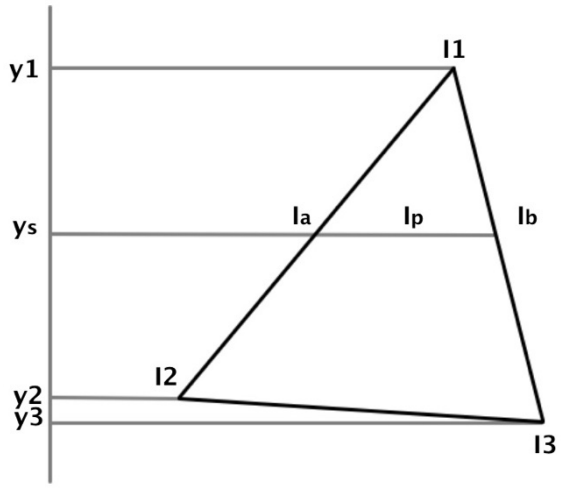

Рис. 1.5.1: Интерполяция интенсивностей

$$I_a = I_1 \frac{y_s - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2}$$

$$I_b = I_1 \frac{y_s - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_s}{y_1 - y_3}$$

$$I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$

Рис. 1.5.2: Формулы вычисления интенсивности цвета

1.5 Выводы из аналитического раздела

В данном разделе были описаны: алгоритм Брезенхема, файлы формата Wavefront obj, алгоритм художника, алгоритм z-буфера, a-буфера, метод тонировки Гуро. Также было проведено сравнение алгоритмов удаления невидимых линий и поверхностей.

2 Конструкторская часть.

В данном разделе будет приведена блок-схема алгоритма z-буфера. Также будет описан процесс рендера изображения.

2.1 Требования к программе

Программа должна предоставлять следующие возможности:

- визуальное отображение сцены;
- визуальное отображение источника света;
- визуальное отображение световых столбов;
- поворот объекта.

2.2 Выбор алгоритма для удаления невидимых линий

Стоит задача визуализировать трёхмерный объект и поворачивать его относительно его центра. Алгоритмы, работающие в трехмерном пространстве, и алгоритм художника не подходят, ввиду трудозатратности: рендеринг объекта и визуализация световых столбов будет выполняться очень долго.

Если выбирать среди описанных ранее алгоритмов z-буфер и a-буфер, то самым логичным решением будет выбор алгоритма z-буфер: объекты не состоят из прозрачных или просвечиваемых материалов, поэтому алгоритм a-буфер будет выполнять ненужную работу.

2.3 Процесс отрисовки изображения

В качестве алгоритма визуализации объекта был выбран алгоритм z-буфер. Данный выбор был аргументирован в аналитической части.

Основные этапы работы данного алгоритма:

1. инициализация и заполнение буфера кадра фоновым значением интенсивности цвета;
2. инициализация и заполнение z-буфера максимальным значением z;
3. преобразование объекта в растровую форму (порядок преобразования не имеет значения);
4. для каждого пиксела в многоугольнике вычислить его глубину z(x, y) или расстояние по оси z. Сравнить вычисленную глубину z(x, y) со значением в zbuffer(x, y);
5. Если значение z(x, y) будет меньше значения zbuffer(x, y), тогда заменяем значение zbuffer(x, y) на z(x, y) и вычисляем новое значение интенсивности цвета для точки с координатами (x, y, z(x, y));
6. в противном случае, никаких действий не выполнять.

Процесс рендера программном модуле проходит через 4 этапа:

1. преобразование мировых координат к координатам сцены;

2. удаление невидимых линий и поверхностей при помощи алгоритма z-буфер;

3. вычисление интенсивности цвета для граней методом тонирования Гуро;

4. вывод изображения на экран.

Пункты 2 и 3 выполняются совместно.

Схема работы алгоритма представлена на рисунках 2.2.1 - 2.2.2.

Рис. 2.2.1: Схема работы алгоритма z-буфер часть 1

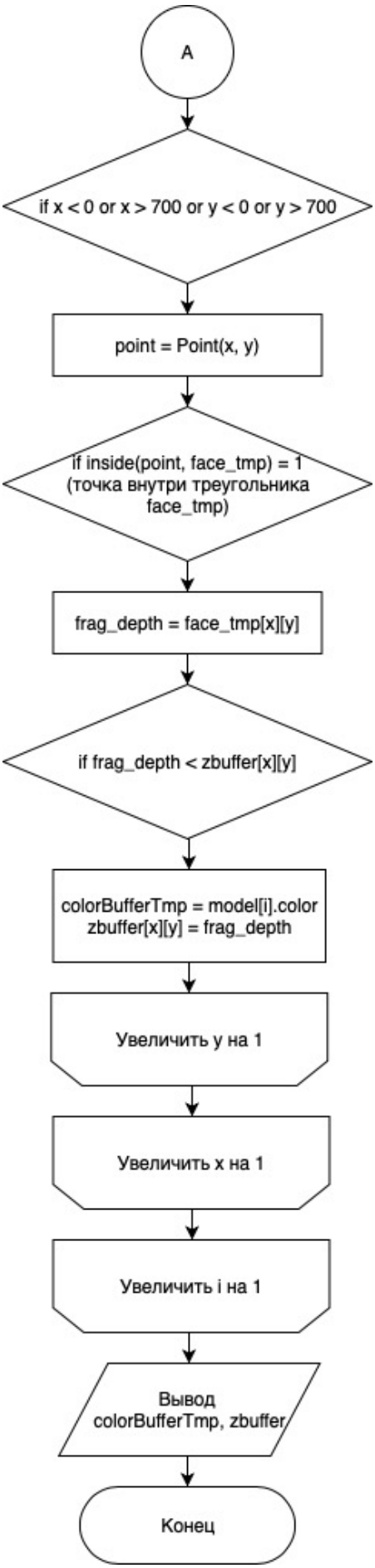

Рис. 2.2.2: Схема работы алгоритма z-буфер часть 2

2.4 Предоставляемый функционал

Интерфейс представляет собой некое пространство, в котором пользователь может выполнять различные преобразования с моделью.

Можно выделить следующие возможности:

• выбор модели для рендеринга (дерево или решётка);
• вращение координат модели (влево или вправо);

2.5 Вывод

В данном разделе была приведена схема работы алгоритма z-буфер, была описана работа рендера изображения в целом. Также были описаны возможности пользователя.

3 Технологическая часть.

В данном разделе будут рассмотрены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач.

3.1 Требования к программному обеспечению

Программное обеспечение должно реализовывать поставленную на курсовой проект задачу. В программном модуле должен присутствовать интерфейс для взаимодействия с объектом. Программа не должна аварийно завершаться, сильно нагружать процессор, а также не должна задействовать большой объем оперативной памяти.

3.2 Выбор и обоснование языка и среды программирования.

Для разработки данной программы применён язык C++ и среда Qt Creator по следующим причинам:

- широкие возможности реализации алгоритмов с высокой эффективностью;
- значительный набор стандартных классов и процедур;
- большое количество виджетов и их гибкость;
- удобство отладки и редактирования программы;
- активное использование сигнально-слотовой связи для реализации асинхронного взаимодействия классов.

Версия компилятора C++: GNU++11 [-std=gnu++11]

3.3 Использованные типы и структуры данных

В реализованной программе использовались некоторые базовые типы и структуры данных языка C++, такие как integer, float, char. Также использовался шаблон std::vector<T>, расположенный в заголовочном файле <vector.h> в пространстве имен std. Vector использовался для хранения большого числа пикселей, нормалей, вершин, граней. Еще была заимствована библиотека glm (OpenGL Mathematics) для работы с матрицами и векторами. Встроенная структура данных QImage необходима для задания разрешения полотна и для отрисовки(высвечивания) на нем пикселей.

Кроме того, в программе используются специально разработанные для поставленной задачи классы:

- Model - класс, назначение которого - считывать, хранить, обеспечивать доступ и необходимые операции для работы с прочитанной из obj файла моделью. Во время запуска программы в данную структуру данных считывается информация из obj файла;
- Data - структура, предназначенная для хранения изображения виджета, задействованного в оконном приложении;
- IShader - структура данных, используемая для определения интенсивности цвета для определенной грани, которая подается в функцию vertex;

- MWidget - класс, который связывает в себе большинство вышеперечисленных структур данных. Данный класс нужен для считывания данных с файла, работы с полотном. Почти все действия, происходящие в программе, связаны с данной структурой данных.

3.4 Формат хранения файла

Формат файла описания геометрии OBJ (Wavefront object) — это разработка Wavefront Technologies для анимационного пакета Advanced Visualizer. Формат файла является открытым и был принят другими разработчиками приложений 3D графики. Он может быть экспортирован/импортирован во многих программах, связанных с многомерным моделированием. По большей части это общепринятый формат[3].

В файле формата .obj хранятся данные вершин, элементы, кривые произвольной формы, связь между поверхностями свободной формы. В рамках курсового проекта были использованы следующие данные:

• геометрические вершины (v);

• вершины нормалей (vn);

• грани (f).

В очень сложных по детализации файлах формата obj типов данных намного больше, их насчитывается больше 30 разных видов.

Формат записи данных в файл можно увидеть на рисунке 1.3.1.

```
g objMesh
v −0.631852 0.857320 −0.007216
v −0.631852 0.834007 −0.011006
v −0.631852 0.857320 0.007222
vt 0.000000 0.000000
vt 0.030303 0.000000
vt 0.000000 0.045455
vn −1.000000 0.000000 0.000000
vn −1.000000 0.000000 0.000000
vn −1.000000 0.000000 0.000000
f 1/1/1 2/2/2 3/3/3
f 4/4/4 3/3/3 2/2/2
f 5/5/5 6/6/6 7/7/7
```

Рис. 1.3.1. Хранение данных в файлах формата OBJ.

3.5 Оформление итогового изображения

Требование к итоговому изображению:

- Хорошая детализация изображения: отсутствие «рваных пикселей»;
- Правильное отображение световых столбов.

В связи с установленными требованиями к изображению было принято решение установить разрешение полотна 400 x 400 пикселей. Данное разрешение было установлено с целью снижения затрат по времени на рендер изображения и отрисовку световых столбов.
Итоговое изображение продемонстрировано на рисунке 3.4.1.

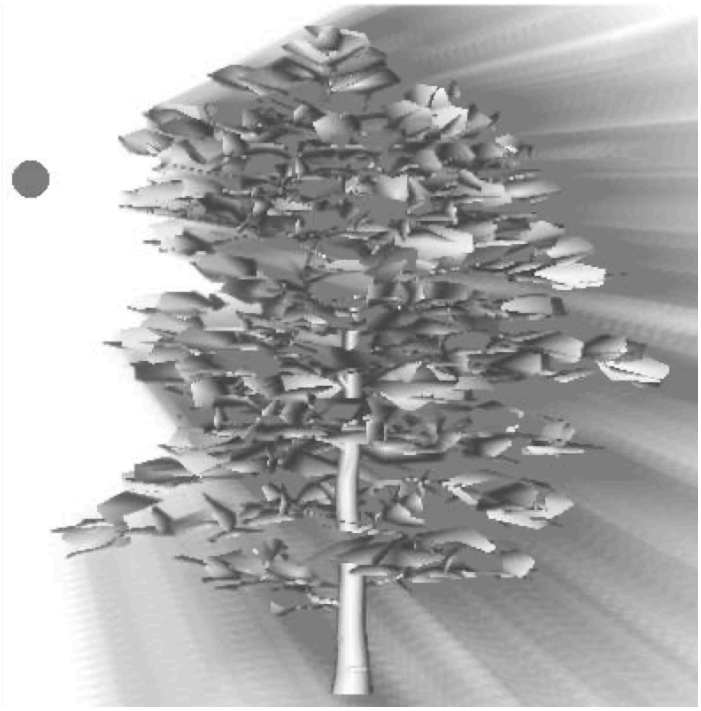

Рис. 3.4.1: Пример работы программы

3.6 Использованные типы и структуры данных

После запуска программы будет построено стартовое изображение сцены и интерфейс для взаимодействия с объектом, см. на рисунке 3.5.1.

Рис. 3.5.1: Стартовое изображение программы

Интерфейс разбит на 2 блока по типу взаимодействия с полотном, см. рисунок 3.5.1:

1) Блок «Выбор объекта» отвечает за выбор из представленных объектов для отрисовки и построения световых столбов:
   a) Кнопка «Дерево» очищает сцену и моделирует объект дерева в изначальном положении и строит световые столбы для него.
   b) Кнопка «Решётка» очищает сцену и моделирует объект решётки в изначальном положении и строит световые столбы для него.
2) Блок «Поворот объекта» отвечает за поворот координат объекта для наглядной демонстрации отображения световых столбов:
   a) Кнопка «<-» очищает сцену, поворачивает координаты объекта влево относительно центра объекта и моделирует новый объект с построенными световыми столбами.
   b) Кнопка «->» очищает сцену, поворачивает координаты объекта вправо относительно центра объекта и моделирует новый объект с построенными световыми столбами.

3.7 Пользовательский интерфейс.

Программа имеет русскоязычный интерфейс, организованный в виде набора виджетов, позволяющих пользователю после запуска программы выбирать объект для отрисовки и поворачивает его влево или вправо.
Доступ ко всем функциям и настройкам осуществляется непосредственно через компоненты основного окна программы.

4 Экспериментально-исследовательская часть.

4.1 Исследование зависимостей времени генерации сцены от кол-ва полигонов объекта.

Суть эксперимента заключается в наблюдении зависимости изменения времени генерации отрисовки сцены и световых столбов от кол-ва полигонов объекта. Точные результаты исследования представлены в таблице 4.1.1.

Название объекта	Количество полигонов объекта	Время генерации и отрисовки световых столбов, с
Решётка	4212	4,59
Дерево	11685	28,01

Таблица 4.1.1. Зависимость времени генерации и отрисовки сцены от кол-ва полигонов объекта.

4.2 Исследование зависимостей времени генерации сцены от кол-ва виртуальных планов для построения световых столбов.

Суть эксперимента заключается в наблюдении зависимости изменения времени генерации отрисовки сцены и световых столбов от кол-ва виртуальных планов для построения световых столбов. Точные результаты исследования представлены в таблице 4.2.1.

Чем меньше виртуальных планов, тем хуже будет выглядеть результат отображения световых столбов (световые столбы будут «рваными»)

При кол-ве виртуальных планов равном нулю, световые столбы не рисуются.

Кол-во виртуальных планов	Время генерации и отрисовки световых столбов для объекта «Дерево», с	Время генерации и отрисовки световых столбов для объекта «Решётка», с
0	0,75	0,39
10	4,32	1,43

15	8,88	2,25
25	28,01	4,59

Таблица 4.2.1. Зависимость времени генерации и отрисовки сцены от кол-ва виртуальных планов.

4.3 Вывод

В данном разделе было проведено исследование временных затрат от количества полигонов объекта, загружаемых на сцену, для программы с различными способами хранения данных. Увеличение скорости работы рендера достиглось за счет того, что мы храним считанный из файла объект в собственной структуре данных Model, поэтому нам не приходится загружать его каждый раз из файла.

Заключение.

Разработанный программный комплекс отвечает всем предъявляемым к нему требованиям и обеспечивает возможность моделирования световых столбов. Проведены исследования работы алгоритмов на различных наборах исходных данных.

Алгоритм z-буфера показал себя с лучшей стороны скоростью отрисовки сцены, но он не характеризуется реалистичностью синтезированного изображения из-за отсутствия таких эффектов, как освещённость, отражение, преломление и т.д. Сымитировать подобные эффекты возможно, накладывая на объекты заранее просчитанные текстуры освещённости, что в совокупности с эффективными алгоритмами оптимизации по быстродействию приближает изображение к реальности.

Список использованной литературы

1. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.
2. Шлее М., Qt 4.8 Профессиональное программирование на C++. – СПб.:БХВ-Петербург, 2012 – 912 с.:
3. Страуструп Б., Язык программирования C++. Специальное издание. Пер. с англ. – М.:Издательство Бином, 2012 г. – 1136 с.: ил.
4. Проблемы трассировки лучей – из будущего в реальное время. [Электронный ресурс]. – Режим доступа: https://nvworld.ru/articles/ray-tracing/3/
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж., Приёмы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.: ил. (Серия «Библиотека программиста»)
6. Костикова Е., Методы интерактивной визуализации динамики жидких и газообразных сред. – М: МГУ, 2009 – с. 20-23.
7. Particle Systems. [Электронный ресурс]. – Режим доступа: http://homepages.inf.ed.ac.uk/tkomura