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:

Моделирование реалистического изображения строительных инструментов, деталей

Студент ИУ7-53Б _____ Идрис Магомет-Салиевич Ильясов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Ольга Владимировна Кузнецова
(Подпись, дата) (И.О.Фамилия)

Москва, 2019 г.

Реферат

Целью данного курсового проекта является изучение алгоритмов компьютерной графики построения реалистических трехмерных изображений: удаление невидимых граней, закраска. В результате выполнения работы получено приложение, написанное на языке программирования C++ с использованием среды разработки Qt Creator.

Отчет содержит 26 стр., 7 рис., 6 источн..

Введение

С каждым годом виртуальная реальность все сильнее проникает в обычную человеческую жизнь и используется в различных сферах: в развлекательной для создания компьютерных игр и спецэффектов в кинолентах, в научной для моделирования и проведения экспериментов, в коммерческой для наглядного отображения данных.

Одним из быстро развивающихся направлений использования виртуальной реальности и компьютерной графики является моделирование и визуализация реалистичного трехмерного изображения. Для удовлетворения растущих потребностей в скорости синтеза и реалистичности полученного изображения разрабатываются новые алгоритмы и совершенствуются уже существующие.

Целью данной работы является разработка программного продукта для моделирования реалистического изображения строительных инструментов и деталей.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать существующие методы и алгоритмы удаления невидимых граней и поверхностей, закраски;
- разработать или адаптировать существующие алгоритмы, необходимые для решения поставленной задачи;
- спроектировать программный продукт, моделирующий реалистическое изображение строительных инструментов и деталей;
- реализовать программное обеспечение.

1. Аналитический раздел

В данном разделе содержится постановка решаемой задачи, проводится анализ существующих алгоритмов построения трехмерных изображений, выбираются наиболее подходящие алгоритмы для достижения цели работы.

1.1. Формализация объектов моделируемой сцены

В компьютерной графике существует несколько вариантов геометрического представления трехмерных объектов:

- Каркасная модель
- Поверхностная модель
- Объемная модель

Каркасная модель представляет геометрию трехмерного объекта, описывая положение контуров и граней, является наименее информативной. Поверхностная модель представляет объект в виде набора ограничивающих поверхностей, она не дает представления о том, по какую сторону материал, по какую пустота. Объемная модель является наиболее информативной среди рассматриваемых моделей. В объемной модели добавляется информация о том, где материал, а где пустота.

В рамках рассматриваемой задачи объекты, расположенные на сцене, лучше всего описываются через поверхностную модель, так как каркасные модели недостаточно информативны, а объемные модели слишком информативны и ресурсоемки.

Сцена состоит из следующих объектов:

- Точечный источник света, который представляет собой материальную точку, расположенную за камерой и из которой исходят лучи во время направления равномерно. Положение задается трехмерными координатами.
- Строительные инструменты и детали, являющиеся полигональными моделями, заданными множеством многоугольников и нормалей.

На рисунке 1.1 приведена функциональная модель решаемой задачи.

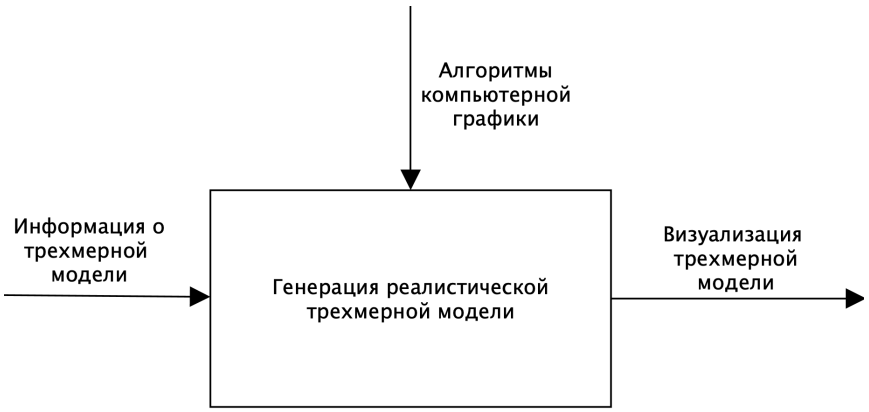

Рисунок 1.1. Функциональная модель IDEF0 нулевого уровня.

1.2. Ограничения

В рамках данной работы запрещено использование готовых сторонних библиотек для написания приложений, использующих двухмерную и трехмерную компьютерную графику.

1.3. Критерии выбора алгоритма построения трехмерной сцены

Для решения поставленной задачи и получения наилучшего результата из множества алгоритмов нужно выбрать алгоритм, соответствующий следующим критериям:

- простота реализации алгоритма (простой алгоритм проще отлаживать в случае возникновения ошибок);
- наилучшая скорость работы алгоритма (для получения максимально быстрого отклика от программы и плавной картинки).

1.4. Описание алгоритмов

Далее будут рассмотрены алгоритмы удаления невидимых поверхностей, закрашивания видимых поверхностей и операции преобразования точек.

1.4.1. Базовая математика

Для реализации преобразований точек и элементарных законов, описанных в п. 1.1., необходимо прибегнуть к базовой математике.

Перемещение точки

Перемещение точки — наиболее простая операция преобразования в пространстве, ее можно описать формулой:

$$\begin{cases} x_1 = x_0 + dx \\ y_1 = y_0 + dy \\ z_1 = z_0 + dz \end{cases}$$

где x_0, y_0, z_0 — начальные координаты точки,

x_1, y_1, z_1 — преобразованные координаты точки,

dx, dy, dz — приращение по трем координатам.

Поворот точки

Поворот точки осуществляется с указанием центра поворота. Пусть точка M с координатами (x, y) при повороте с центром в точке $A(x_0, y_0)$ на угол φ переходит в точку $M'(x', y')$, тогда

$$\begin{cases} x' = xcos\varphi - ysin\varphi \\ y' = xsin\varphi + ycos\varphi \end{cases}$$

Масштабирование точки

Масштабирование точки также производится с указанием центра масштабирования. Пусть точка M с координатами (x, y) при масштабировании с центром в точке $A(x_0, y_0)$ на значения коэффициентов kx, ky переходит в точку $M'(x', y')$, тогда

$$\begin{cases} x' = xkx + (1 - kx)x_0 \\ y' = yky + (1 - ky)y_0 \end{cases}$$

1.4.2 Алгоритмы удаления невидимых линий и поверхностей

Задача удаления невидимых линий и поверхностей является одной из самых главных и наиболее сложных в компьютерной графике. Алгоритмы удаления невидимых линий и поверхностей [2] служат для определения линий и рёбер, поверхностей или объёмов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства.

Для этой задачи, одной из центральных в машинной графике, не существует единого и наилучшего способа удаления. Сложность задачи привела к появлению большого числа различных способов её решения. Многие из них ориентированы на специализированные приложения.

Все алгоритмы удаления невидимых линий (поверхностей) включают в себя сортировку. Главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения. Также существует сортировка по загораживанию тел друг другом. Эффективность этих алгоритмов зависит в большей мере от эффективности процесса сортировки.

Существует тесная взаимосвязь между скоростью работы алгоритма и детальностью его результата. Ни один из алгоритмов не может достигнуть хороших оценок для двух этих показателей одновременно.

Алгоритмы удаления делятся на две группы:

1. работающие в объектном пространстве (мировая система координат; высокая точность);
2. работающие в пространстве изображений (система координат связана с дисплеем; точность ограничена разрешающей способностью экрана).

Сложность для алгоритма, работающего в объектном пространстве $\sim N^2$, где N — количество объектов; для алгоритма, работающего в пространстве изображения $\sim(N * M)$, где M — число пикселей.

$N^2 < N * M$, но алгоритмы, работающие в пространстве изображений, на практике могут быть эффективнее в силу свойства когерентности при растровой реализации (рядом расположенные пиксели чаще всего будут обладать одинаковыми свойствами).

Алгоритм с Z-буфером

Данный алгоритм является одним из простейших алгоритмов удаления невидимых поверхностей, но требует большого объема памяти. Обычный буфер кадра хранит коды цвета для каждого пикселя в пространстве изображения. Идея алгоритма состоит в том, чтобы для каждого пикселя дополнительно хранить еще и координату Z, или глубину. При занесении очередного пикселя в буфер кадра значение его Z-координаты сравнивается с Z-координатой пикселя, который уже находится в буфере. Если Z-координата нового пикселя больше, чем координата старого, т.е. он ближе к наблюдателю, то атрибуты нового пикселя и его Z-координата заносятся в буфер, если нет, то ничего не делается.

Время работы алгоритма не зависит от сложности сцены. Многоугольники, составляющие сцену, могут обрабатываться в произвольном порядке. Для сокращения затрат времени нелицевые многоугольники могут быть удалены.

Таким образом, основные преимущества алгоритма — его скорость работы, простота реализации и возможность распараллеливания (т.к. каждый многоугольник можно обрабатывать по отдельности), из недостатков можно выделить большое потребление памяти и небольшую точность вычислений.

Алгоритм Робертса

Данный алгоритм удаляет невидимые линии в объектном пространстве. Он работает только с выпуклыми телами в пространстве объектов. Каждый объект сцены представляется многогранным телом, полученным в результате пересечения плоскостей. Т.е. тело описывается списком граней, состоящих из ребер, которые в свою очередь образованы вершинами.

Вначале из описания каждого тела удаляются нелицевые плоскости, экранированные самим телом. Затем каждое из ребер сравнивается с каждым телом для определения видимости или невидимости. Т.е. объем вычислений растет как квадрат числа объектов в сцене. Наконец, вычисляются новые ребра, полученные при протыкании телами друг друга.

К преимуществам алгоритма можно отнести точность вычислений и возможность распараллеливания, а к недостаткам скорость работы, сложность реализации и строгую ориентацию метода только на выпуклые многогранники.

Алгоритм Варнока

Алгоритм Варнока основывается на рекурсивном разбиении экрана. Алгоритм работает в пространстве изображения и анализирует область на экране дисплея (окно) на наличие в них видимых элементов. Если в окне нет изображения, то оно просто закрашивается фоном. Если же в окне имеется элемент, то проверяется достаточно ли он прост для визуализации. Если объект сложный, то окно разбивается на более мелкие, для каждого из которых выполняется тест на отсутствие или простоту изображения. Рекурсивный процесс разбиения может продолжаться до тех пор, пока не будет достигнут предел разрешения экрана.

Основные плюсы алгоритма – простота реализации и возможность распараллеливания (т.к. каждую четверть изображения можно вычислять в отдельном потоке), недостатки – скорость выполнения (т.к. алгоритм рекурсивный) и небольшая точность вычислений.

Алгоритм трассировки лучей

В алгоритме трассировки лучей генерируется пучок лучей, выходящих от наблюдателя во всевозможные направления. Проблема данного алгоритма состоит в том, что лишь малая часть этих лучей дойдут до источника, соответственно, повлияют на изображение сцены. При этом для каждого выпущенного от

наблюдателя луча необходимо будет производить расчеты, что в итоге сказывается на эффективности алгоритма.

Положительной стороной данного алгоритма является универсальность: можно использовать в сложных сценах, а также есть возможность использования в параллельных вычислительных системах (т.к. расчет отдельной точки выполняется независимо от других точек).

Недостатком данного алгоритма служит большое количество необходимых вычислений для синтеза изображения.

Вывод

В ходе анализа алгоритмов удаления невидимых линий и поверхностей установлено, что наиболее подходящий алгоритм в соответствии с выбранными критериями — алгоритм с Z-буфером.

1.4.3 Алгоритмы закраски поверхностей

Целью данных алгоритмов является получение сглаженного изображения.

Простая закраска

В простой модели освещения отраженный от объекта свет может быть диффузным. Диффузное отражения света происходит, когда свет проникает под поверхность объекта, поглощается, а затем вновь испускается, соответственно, свет рассеивается равномерно по всем направлениям. Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта [3, 5, 6].

$$I = I_0 * cos(\alpha),$$

где I – результирующая интенсивность света в точке,

I_0 – интенсивность источника,

α – угол между нормалью к поверхности и вектором направления света.

К преимуществам данного метода можно отнести простоту реализации и быстроту выполнения. Недостаток данного алгоритма — невысокая реалистичность изображения.

Закраска по Гуро

Данный алгоритм предполагает билинейную интерполяцию интенсивностей — это определение значений функции в точке, лежащей внутри какого-то интервала. За счёт создаётся иллюзия гладкой криволинейной поверхности. Для работы алгоритма необходимо вычесть, помимо нормали к каждому полигону, нормали к каждой вершине (усреднение нормалей примыкающих полигонов).

Положительной стороной данного алгоритма является реалистичность изображения и простота реализации. Недостатки: скорость выполнения [5].

Закраска по Фонгу

Алгоритм основан на билинейной интерполяции нормалей. Из-за этого достигается лучшая локальная аппроксимация кривизны поверхности.

Плюсы алгоритма: реалистичное изображение, блики от поверхности выглядят наиболее правдоподобно. Минусы: скорость выполнения [5].

Вывод

В ходе анализа алгоритмов закраски поверхностей установлено, что наиболее подходящий алгоритм в соответствии с выбранными критериями — простая закраска.

1.5. Вывод

В результате рассмотрения алгоритмов выбраны следующие алгоритмы:

1. алгоритм с Z-буфером (для удаления нелицевых поверхностей и граней);

2. простой алгоритм закраски (для сглаживания изображения);

2. Конструкторский раздел

В данном разделе рассматриваются используемые типы и структуры данных, диаграмма классов, схемы алгоритмов для Z-буфера и простого метода освещения.

Программа должна обладать следующей функциональностью:

1. визуализировать трехмерную сцену, состоящую из объектов, описанных в п. 1.1, в режиме реального времени с учетом логики существования объектов сцены;

2. предоставлять в интерфейсе возможность добавлять и удалять объекты, производить над ними простейшие операции преобразования.

2.1. Схемы алгоритмов

На приведенных ниже рисунках представлены схемы используемых алгоритмов:

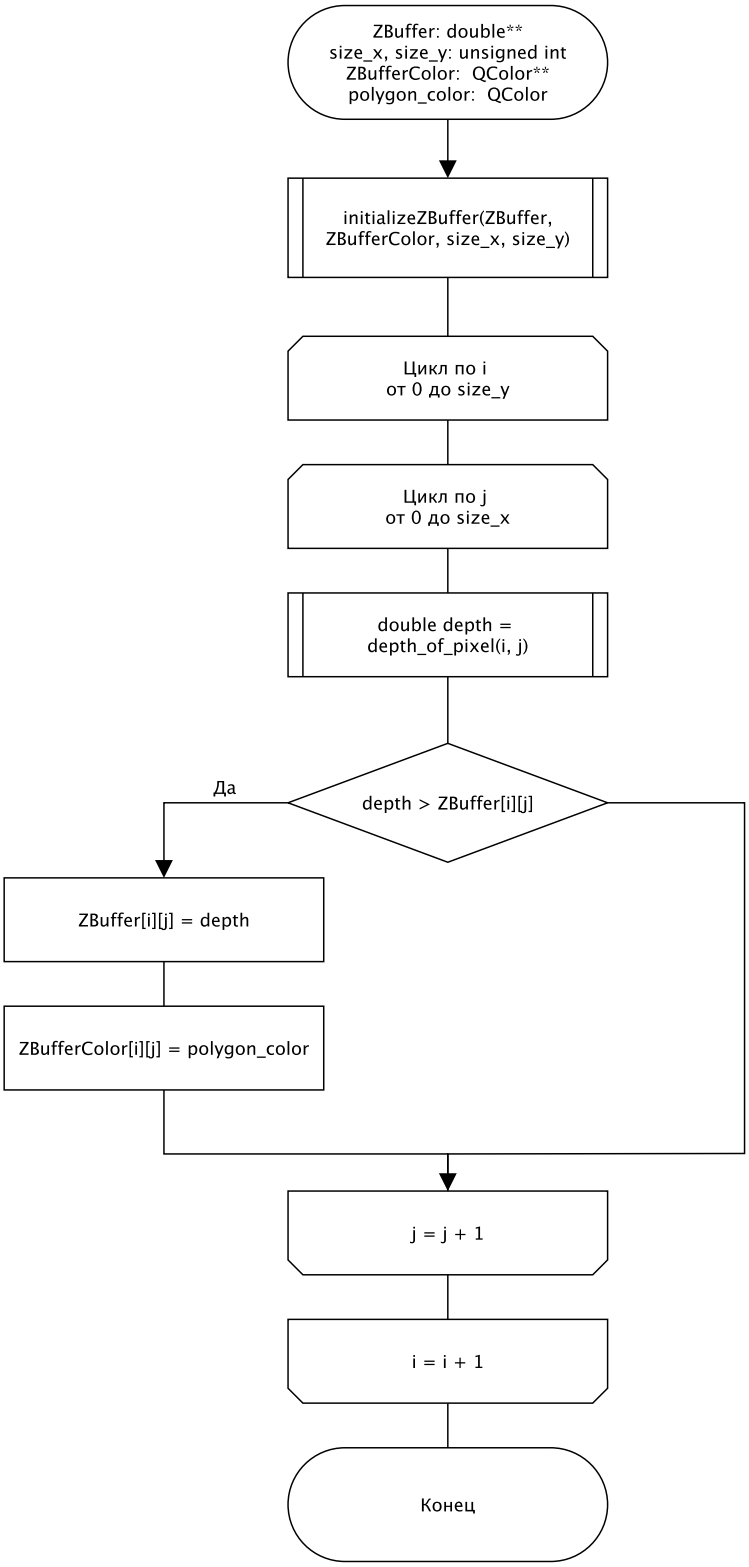

Рисунок 2.1. Схема алгоритма удаления невидимых поверхностей с Z-буфером.

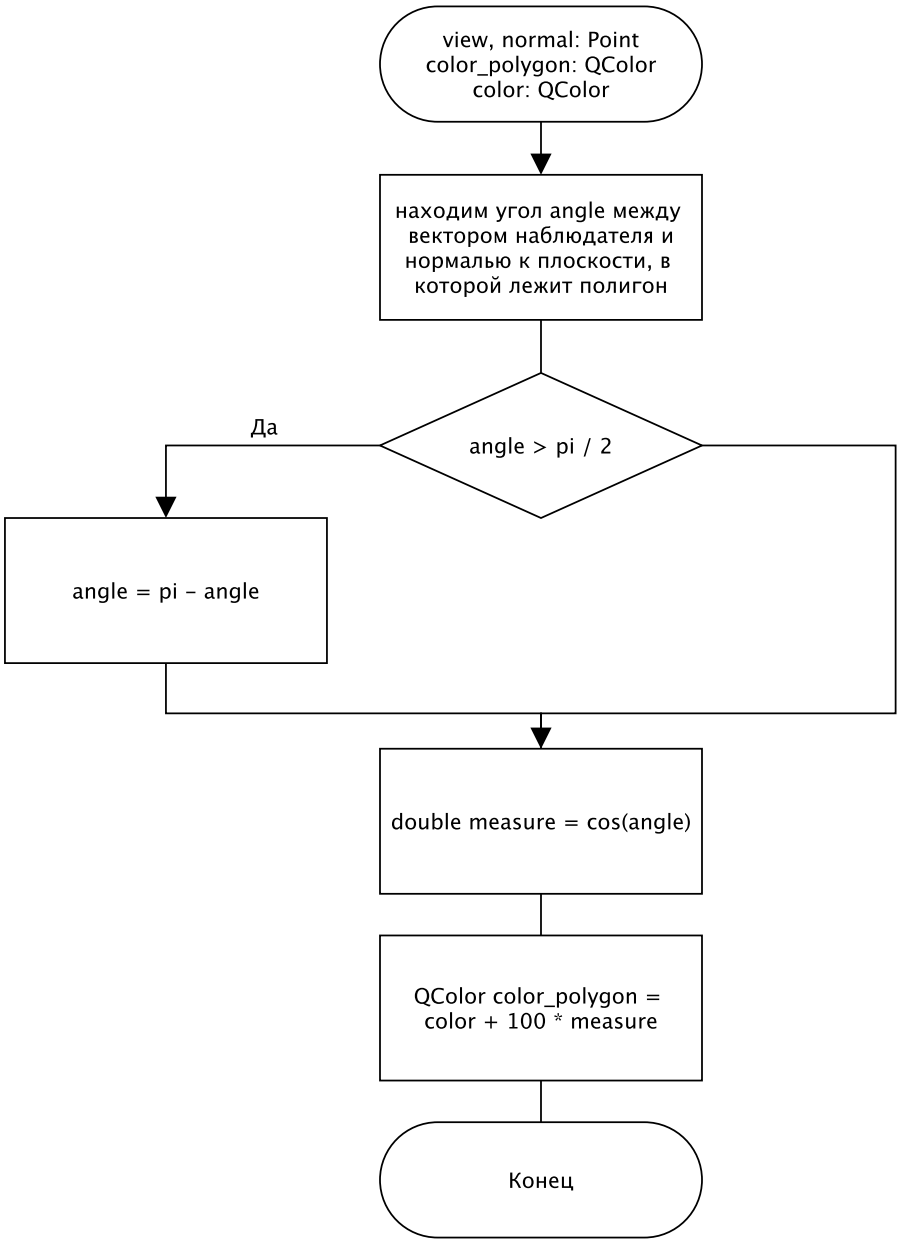

Рисунок 2.2. Схема алгоритма простой закраски.

2.2. Используемые типы и структуры данных

Чтобы реализовать построение выделенной трехмерной сцены, понадобятся структуры данных, описывающие точку, полигон, плоскость, объект, модель, камеру, Z-буфер.

Используемые структуры данных и типы приведены ниже в листингах 2.1-2.7.

Листинг 2.1. Структура точки

```
struct Point
{
    double x;
    double y;
    double z;
};
```

Листинг 2.2. Структура полигона

```
struct Polygon
{
    std::vector<Point> points;
    QColor polygon_color;
    Flatness flatness;

};
```

Листинг 2.3. Структура плоскости

```
struct Flatness
{
    double a;
    double b;
    double c;
    double d;
};
```

Листинг 2.4. Структура объекта

```
struct Object
{
    std::vector<Polygon> polygons;
    Point center;
};
```

Листинг 2.5. Структура модели

```
struct Model
{
    int L;
    int H;
    int W;
};
```

Листинг 2.6. Структура камеры

```
struct Camera
{
    Point coordinates;
    Point direction;
};
```

Листинг 2.7. Структура Z-буфера

```
struct ZBuffer
{
    unsigned size_x;
    unsigned size_x;
    ZBufferCell **cells;

};

struct ZBufferCell
{
    double depth;
    QColor color;
};
```

2.3. Диаграмма классов

Ниже на рисунке 2.3 приведена диаграмма классов.

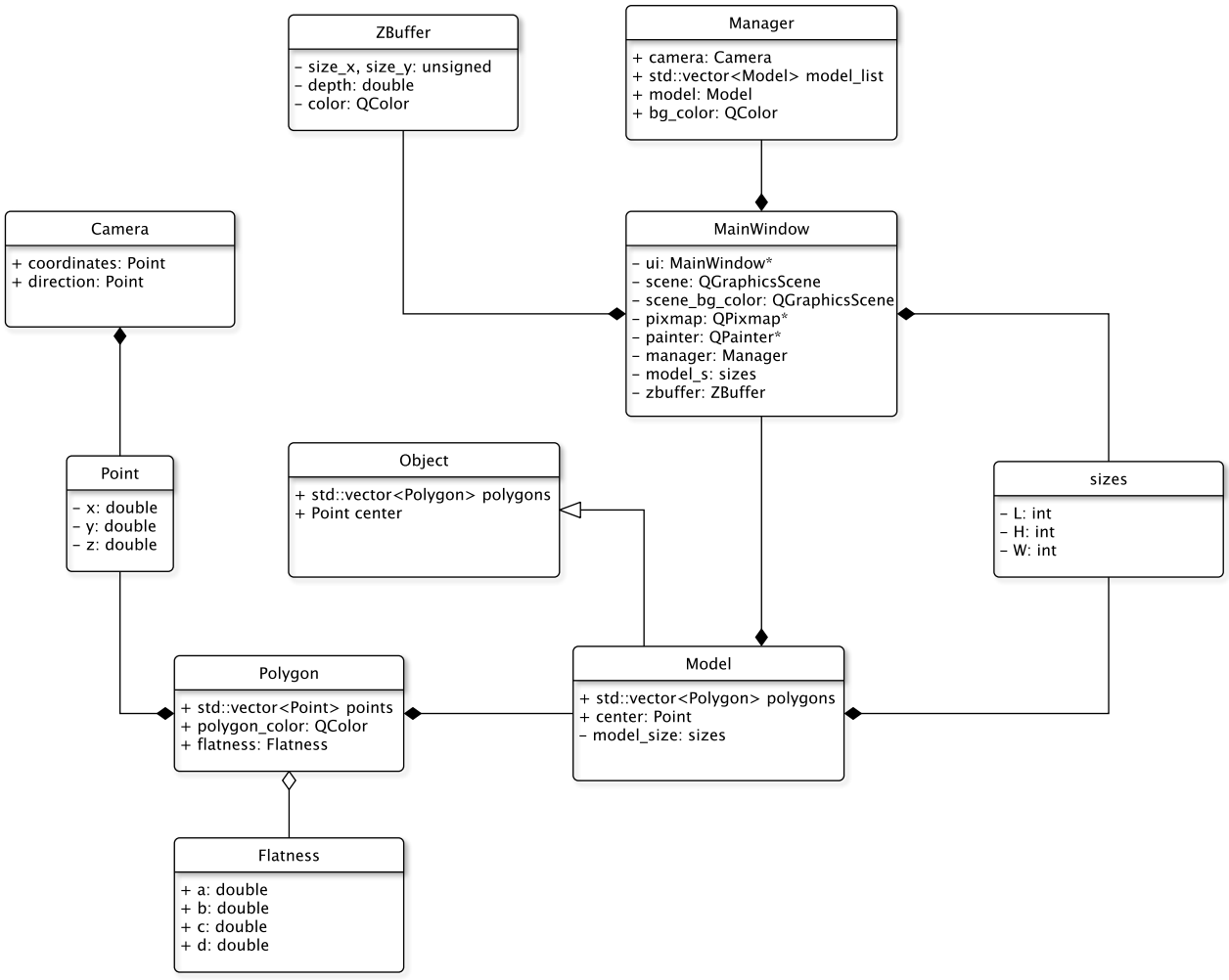

Рисунок 2.3. Диаграмма классов

2.4. Вывод

В данном разделе были рассмотрены схемы используемых алгоритмов, используемые типы и структуры данных и диаграмма классов.

3. Технологический раздел

В данном разделе рассмотрен выбор средств программной реализации, описаны основные моменты программной реализации.

3.1. Выбор языка программирования и среды разработки

В качестве языка программирования выбран язык C++. Этим языком программирования поддерживается объектно-ориентированный подход программирования, что позволяет программисту естественным образом декомпозировать задачу и легко модифицировать программу в случае необходимости.

Для реализации проекта в качестве среды разработки выбрана среда программирования Qt Creator 5.13.2. Данная среда обладает удобным редактором кода и отладчиком, а также широким набором настроек.

3.2. Описание входных данных

В данной программе входными данными являются текстовые файлы с расширением .txt, в них содержится информация о синтезируемой 3D-модели (количество полигонов, вершины многоугольников).

Формат файла выглядит следующим образом:

- общее число полигонов (целое число);
- количество вершин у полигона (целое число) и координаты точек (x, y, z):

```
3
4 -54 4
56 -54 4
56 -106 4
```

- координаты центра модели (x, y, z):

```
30 0 -150
```

3.3. Инструкция по запуску программного обеспечения

Для корректного запуска программного обеспечения каких-либо требований нет. 3D-модели можно добавить на сцену, открыв текстовый файл с требуемым форматом, описанным в п.3.2.

3.4. Описание интерфейса программы

На рисунке 3.1 представлен интерфейс разработанного ПО.

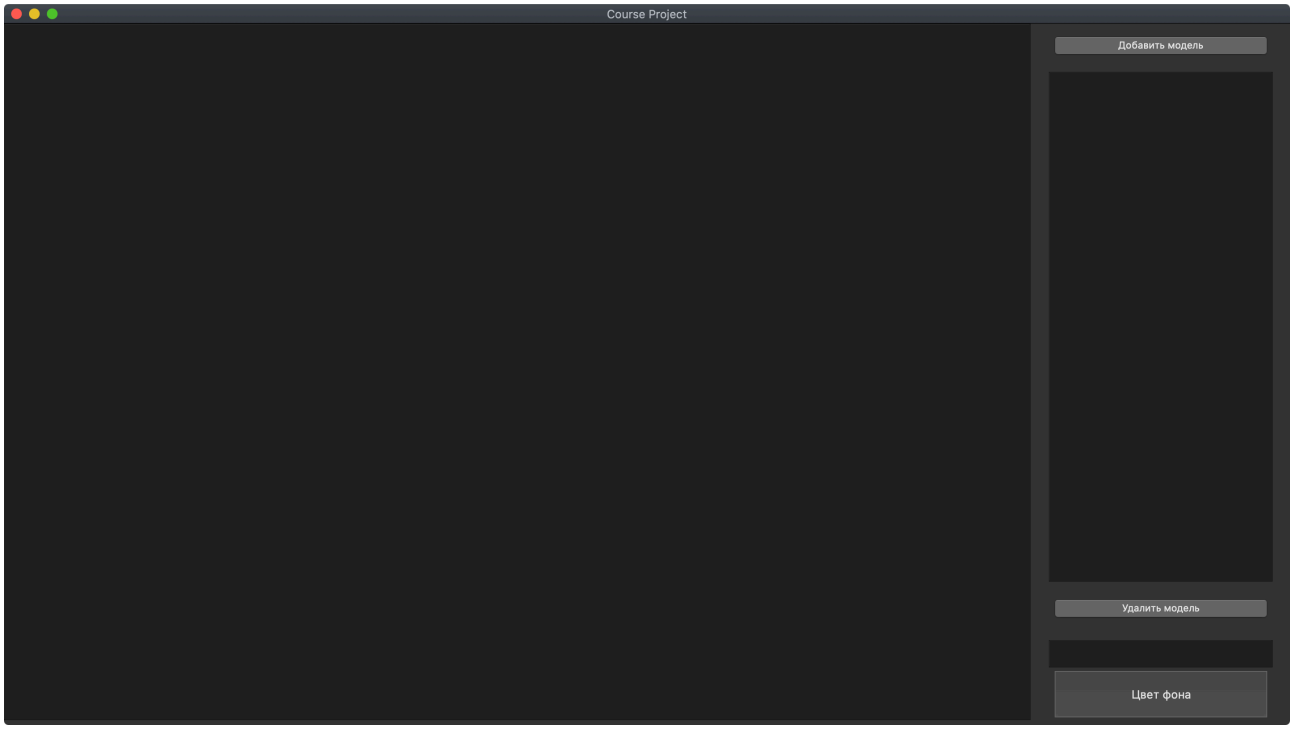

Рисунок 3.1. Интерфейс программы

Пользователю доступна возможность добавления трехмерной модели при помощью диалогового окна с выбором файлов и изменения цвета фона.

Положение выбранной в правой части интерфейса модели может быть изменено с помощью клавиш «W», «A», «S», «D». Для приближения и отдаления модели можно воспользоваться кнопками «Q» и «E». При помощи кнопок «I», «K», «L», «J», «U», «O» можно производить поворот выбранной трехмерной модели вокруг оси OX, OY, OZ соответственно. При необходимости пользователю доступна возможность удаления выбранной модели с холста.

3.5. Вывод

В данном разделе были выбраны средства реализации, рассмотрен интерфейс программы.

4. Экспериментальный раздел

В данном разделе приводится результат проведенного исследования по апробации программного продукта.

4.1. Апробация ПО

Целью исследования является апробация работы программы. Ниже на рисунках 4.1-4.2 приведены примеры отрисовки трехмерной модели молотка.

Рисунок 4.1. Пример отрисовки трехмерной модели.

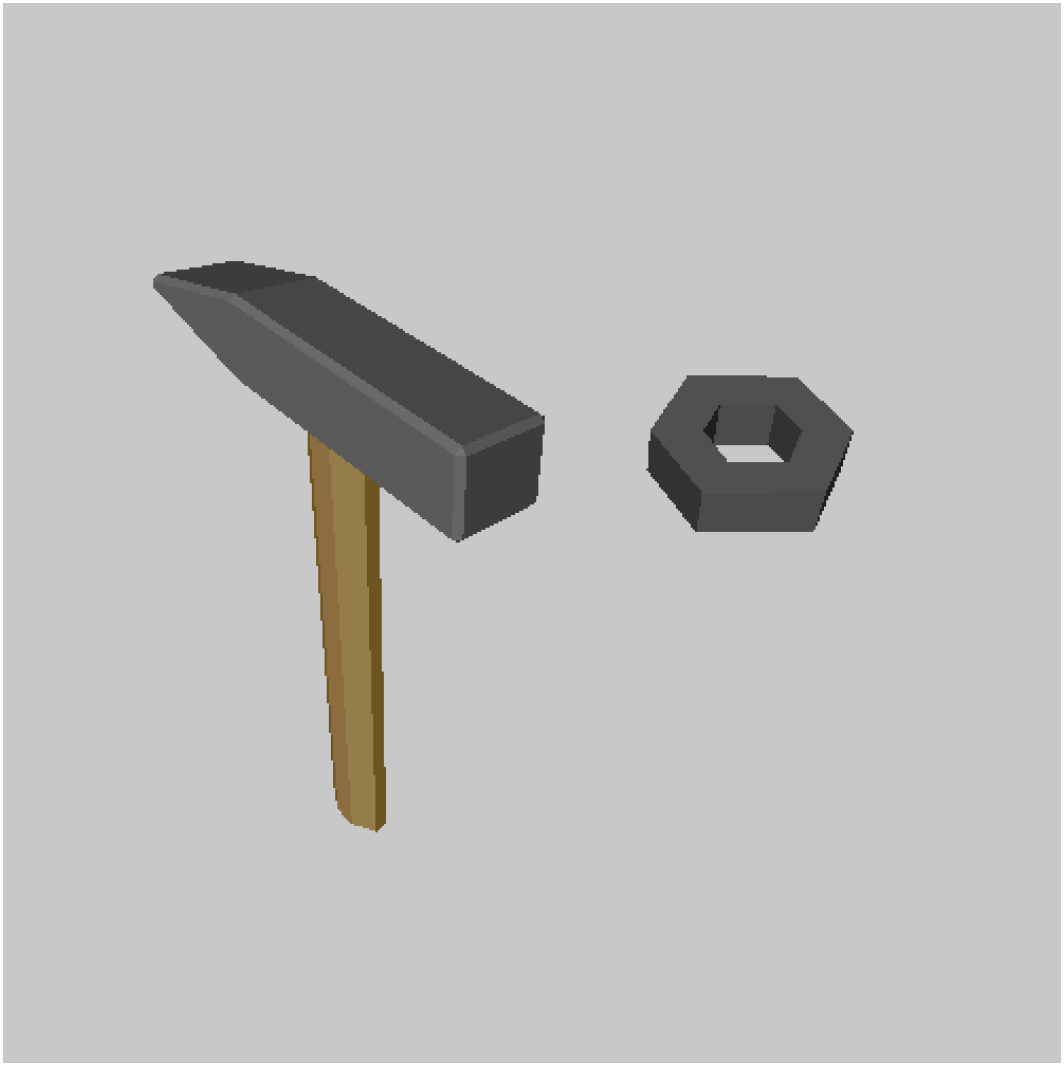

Рисунок 4.2. Пример отрисовки трехмерной модели.

4.2. Вывод

В ходе эксперимента были получены удовлетворительные результаты по моделированию трехмерного реалистического изображения модели строительного инструмента, детали. Все объекты в соответствии с расположением источника света закрашиваются корректно.

Заключение

В ходе выполнения работы были проанализированы существующие алгоритмы удаления невидимых линий и поверхностей, закраски, указаны их преимущества и недостатки.

Разработаны собственные и адаптированы существующие структуры данных и алгоритмы, необходимые для решения поставленной задачи.

Спроектировано и реализовано программное обеспечение, моделирующее реалистическое изображение трехмерной модели строительного инструмента и деталей.

Список литературы

1. Роджерс Д., Адамс Дж. Математические основы машинной графики: Пер. с англ. - М.: Мир, 2001. - 604с.

2. Порев В.Н. Компьютерная графика. –СПб.: БХВ-Петербург, 2002. – 432с.

3. Модели затенения. Плоская модель. Затенение по Гуро и Фонгу. [Электронный ресурс]. – Режим доступа: https://compgraphics.info/3D/lighting/shading_model.php, свободный – (22.10.2019)

4. Шикин Е.В., Боресков А.В. Компьютерная графика. Полигональные модели. – М.: ДИАЛОГ- МИФИ, 2001.-464с.

5. Трехмерная графика с нуля. Часть 2: растеризация. [Электронный ресурс]. — Режим доступа: https://habr.com/ru/post/342708/, свободный - (05.10.2019)

6. Основные алгоритмы компьютерной графики. Реалистичное представление сцен. [Электронный ресурс]. – Режим доступа: https://bourabai.ru/graphics/0211.htm, свободный — (20.10.2019)