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

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Моделирование движения поезда

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Уласик Е.А.

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Новик Н.В.

(Подпись, дата) (И.О.Фамилия)

Москва, 2019

Оглавление

[Введение 3](#_Toc28338660)

[1. Аналитический раздел 4](#_Toc28338661)

[2. Конструкторский раздел 8](#_Toc28338662)

[2.1 Общий алгоритм программы 8](#_Toc28338663)

[2.2 Общий алгоритм растеризации изображения 9](#_Toc28338664)

[2.3 Реализация алгоритма z-буфера 10](#_Toc28338665)

[2.4 Аффинные преобразования 12](#_Toc28338666)

[2.5 Реализация перемещения поезда 12](#_Toc28338667)

[2.6 Реализация камеры 13](#_Toc28338668)

[2.7 Реализация освещения 16](#_Toc28338669)

[3. Технологический раздел 17](#_Toc28338670)

[3.1 Диаграмма классов 17](#_Toc28338671)

[3.2 Интерфейс программы 19](#_Toc28338672)

[Заключение 21](#_Toc28338673)

[Список использованной литературы 22](#_Toc28338674)

# Введение

В наши дни компьютерная графика используется во многих сферах деятельности. С помощью её методов можно создать изображения, которые используются, например, в компьютерных играх и фантастических фильмах; она позволяет моделировать различные ситуации, которые сложно воспроизвести в реальном мире. Таким образом, можно утверждать, что создание реалистичных компьютерных изображений весьма актуальная и востребованная задача.

Целью данного курсового проекта является разработка программы моделирования движения поезда, состоящего из локомотива и нескольких вагонов. На поверхности земли могут располагаться деревья, положение которых задаётся пользователем.

В рамках реализации проекта должны быть решены следующие задачи:

* Анализ алгоритмов компьютерной графики, использующихся для создания изображения.
* Выбор наиболее оптимальных алгоритмов для решения задачи.
* Организация структур данных.
* Проектирование архитектуры программы и её интерфейса.
* Реализация выбранных алгоритмов и структур данных.

# Аналитический раздел

Наиболее распространёнными алгоритмами для отсечения невидимых линий являются алгоритм Робертса, алгоритм Варнока, алгоритм трассировки лучей и алгоритм, использующий z-буфер.

Алгоритм Робертса

Алгоритм Робертса прежде всего удаляет из каждого тела те рёбра или грани, которые экранируются самим телом. Далее определяется какая часть или части ребра тела перекрываются другими телами. [1]

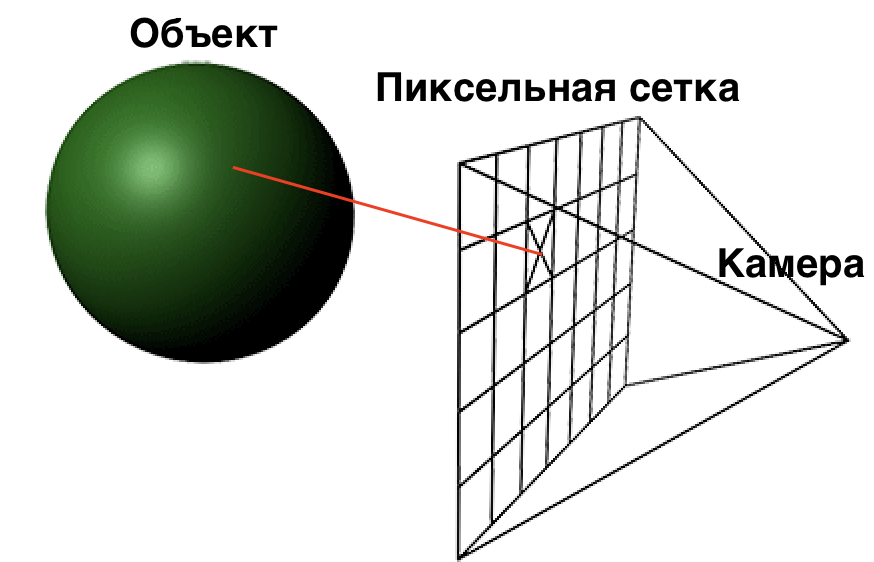
Преимущество данного алгоритма в том, что математические методы, которые используются в нём, просты и точны.

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

Алгоритм трассировки лучей

В этом методе для каждого пикселя картинной плоскости определяется ближайшая к нему грань. Для этого из каждого пикселя выпускается луч, по которому определяются пересечения с гранями объекта. Среди них выбирается ближайшие к камере.[1]

На рисунке 1 приведена иллюстрация алгоритма трассировки лучей:



*Рисунок 1 Иллюстрация работы трассировки лучей*

Преимуществами данного алгоритма являются возможность получения гладких объектов без аппроксимации их примитивами и простота реализации.

Главный недостаток алгоритма – низкая производительность. Для получения изображения необходимо создать огромное количество лучей, проходящих через сцену и отражаемых от объекта.

Алгоритм Варнока

Существует несколько версий данного алгоритма.[2] В основу алгоритма положено разбиение области рисунка на более мелкие подобласти. Для каждой подобласти определяется связанные с ней многоугольники и самые крупные из них изображаются на экране. Предполагается, что уменьшение размеров области приводит к уменьшению количества многоугольников, которые её перекрывают. Разбиение продолжается до того момента, пока область не будет размером в один пиксель.

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

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

Алгоритм, использующий Z-буфер

Z-буфер используется для запоминания атрибутов каждого пикселя в пространстве изображения. Другими словами – это буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пикселя в пространстве изображения. В процессе работы глубина или значение z каждого нового пикселя сравнивается с глубиной того пикселя, который уже занесён в буфер. Если сравнение показывает, что новый пиксель расположен ближе, то пиксель изображается, а старое значения буфера заменяется новым. [1]

Несомненным плюсом данного алгоритма является его простота, которая не мешает решению задачи удаления поверхностей и визуализации их пересечения. В этом алгоритме не тратится время на сортировку элементов сцены, что дает преимущество в скорости работы.

Недостаток алгоритма – большой объём требуемой памяти для буфера. Однако для современных компьютеров этот недостаток не является существенным.

Выбор оптимальных алгоритмов

По критерию использования ресурсов процессора оптимальным является алгоритм z-буфера: его зависимость от числа объектов линейна. Алгоритм Робертса имеет сложность , а алгоритм Варнока использует рекурсивные вызовы и его сложность зависит от положения объектов.

По критерию простоты оптимальными следует считать алгоритм Варнока и z-буфер. Они работают непосредственно с пикселями экрана и не требуют поиска пересечения граней объектов.

Основными критериями выбора алгоритмов для решения поставленной задачи являются скорость работы и простота реализации. Таким образом, алгоритм, использующий Z-буфер, соответствует обоим критериям.

Выбор закраски

Существует три наиболее часто используемых алгоритма закраски многоугольников: простой, по Гуро и по Фонгу.

Простой алгоритм закраски

Каждому многоугольнику присваивается некоторый цвет. После этого, задаётся вектор направления света. Далее находится угол между нормалью поверхности многоугольника и вектором направления света. По полученному значению определяются оттенки цветов. Чем больше повёрнута к источнику поверхность объекта, тем ярче будет она закрашена. Таким образом, различные грани закрашиваются разными цветами, при этом все точки одной грани имеют одинаковый цвет.[2]

Преимуществом данного алгоритма является его высокое быстродействие и простота реализации.

Закраска по Гуро

Для этого алгоритма нормаль вычисляется не для всей поверхности, а для каждой вершины. С помощью этих значений вычисляется интенсивность, которая затем интерполируется по всем точкам грани.

Закраска по Фонгу

Идея алгоритма Фонга похожа на идею алгоритма Гуро, но, в отличие от Гуро, в данном алгоритме интерполируется значение нормали, по которой и находится значение интенсивности пикселя.

Эта закраска требует больших вычислительных затрат, однако позволяет получить более реалистичное изображение.

На рисунке 2 приведены различия методов закраски (1 – простой алгоритм, 2 – по Гуро, 3 – по Фонгу).



*Рисунок 2 Иллюстрация различия методов закраски*

Таким образом, модели, закрашенные методами Фонга и Гуро, очень хорошо детализированы, но, помимо критерия реалистичности, стоит вопрос о быстродействии. Для выполнения данной задачи эти алгоритмы слишком медленны. По этой причине был выбран простой алгоритм закраски.

# Конструкторский раздел

В данном разделе приведены общий алгоритм программы, общий алгоритм растеризации изображения, описания реализаций z-буфера, движения поезда, камеры и освещения, схемы реализованных алгоритмов; даны формулы матриц аффинных преобразований.

## 2.1 Общий алгоритм программы

1. В программу загружаются вершины фигур и порядок их соединения из внешних файлов.
2. На экране формируются статические объекты: поверхность земли и железная дорога.
3. Производится размещение объектов и камеры на начальное положение.
4. Производятся преобразования из декартовой системы координат в экранную в несколько шагов.
5. Отображение данных объектов.
6. С помощью интерфейса можно задать положение деревьев на поверхности земли и количество вагонов состава.
7. После нажатия на клавишу “запуск”, на экране появляется модель поезда, которая движется из одного края экрана, в другую.
8. После того, как поезд закончил движение, можно изменить количество вагонов и начать движение заново.

## 2.2 Общий алгоритм растеризации изображения

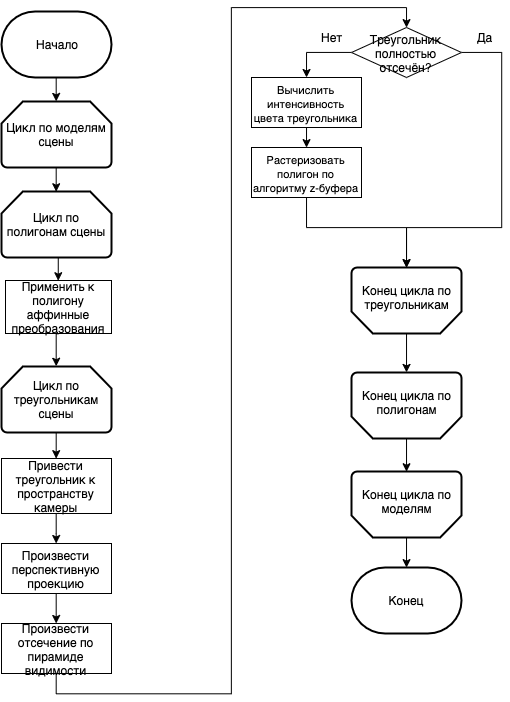
Вход:

* характеристики объектов (положение вершин, цвет);
* характеристики камеры (положение камеры, вектор взгляда);

Выход:

* Выведенное на экран изображение;

На рисунке 3 приведена схема алгоритма растеризации изображения.

**

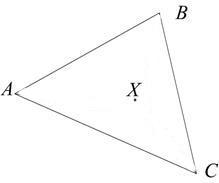
*Рисунок 3 Схема общего алгоритма растеризации изображения*

## 2.3 Реализация алгоритма z-буфера

Изначально буфер кадра заполняется значением . Затем выполняется растровая развёртка каждого треугольника сцены. Для сравнения значения буфера кадра и глубины пикселя каждого треугольника, возникает необходимость в определении точного значения координаты z точки. Для решения данной задачи были использованы барицентрические координаты.[4]

Пусть даны вершины треугольника A, B, C (x, y, z);

Пусть дана некоторая точка X c её координатами x и у, принадлежащая этому треугольнику. На рисунке 4 дана иллюстрация задачи.



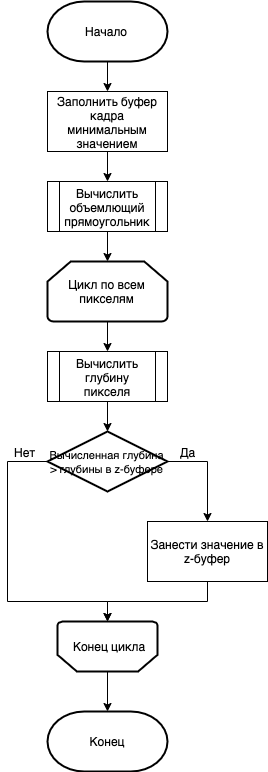
*Рисунок 4 Иллюстрация задачи*

Тогда барицентрические координаты точки равны [3]:

где S – площадь треугольника, которая может быть вычислена лишь один раз.

Если хотя бы одна из барицентрических координат отрицательна, то точка лежит вне треугольника. Иначе координату z следует вычислить по формуле (4):

Полученное значение z сравнивается со значением z в буфере по тем же координатам x, y. В результате сравнения, если значение z точки больше, то пиксель должен быть растеризован, а значение, хранящиеся в буфере, заменено на значение z текущей точки. На рисунке 5 приведена схема алгоритма z-буфера.



*Рисунок 5 Схема алгоритма растеризации с z-буфером*

## 2.4 Аффинные преобразования

В представленном общем алгоритме растеризации изображения происходит применение аффинных координат к полигонам. Это действие осуществляется с использованием матриц аффинных преобразований. Для задачи данной курсовой работы потребовались следующие операции:

* Матрицы перемещения (5):
* Матрицы поворота (вокруг осей x (6), y (7) и z (8) соответственно):
* Матрица однородного масштабирования (9):

## 2.5 Реализация перемещения поезда

Движение корпуса осуществляется с использованием матрицы переноса(5).

Для колёс движение разбивается на два этапа:

* вращение с использованием матрицы поворота вокруг своей оси, вычисляемой по формуле (10)

, (10)

где – матрица поворота, *T' –* транспонированная матрица переноса на новое место поворота, *T* – матрица переноса на изначальную позицию. [6]

* перемещение матрицей переноса (5).

## 2.6 Реализация камеры

Перемещение обзора сцены осуществляется с помощью движения камеры. Перенос объектов в пространство камеры происходит с помощью аффинных преобразований:

Далее необходимо спроецировать полигон на экранную плоскость. Для этого используется матрица перспективной проекции пирамиды видимости(11):

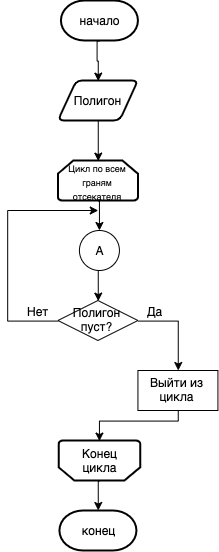
где – увеличение по X, – увеличение по Y, *far* – расстояние от камеры до задней грани пирамиды видимости, *near* – расстояние от камеры до передней грани пирамиды видимости [2].

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

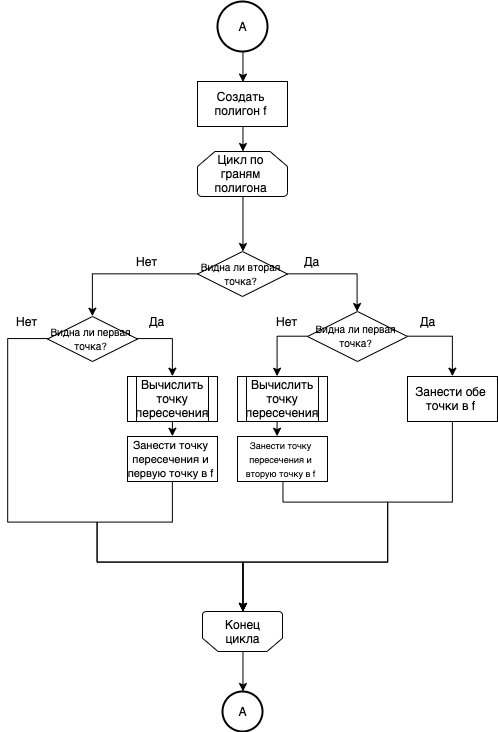
где fov – вертикальный угол обзора.

где ratio – соотношение сторон экрана.

Далее происходит отсечение невидимых линий методом последовательного отсечения Сазерленда-Ходжмана по граням пирамиды видимости. В этом алгоритме исходный и каждый из промежуточных многоугольников отсекается последовательно относительно одной плоскости.[1] Отсечение происходит по всем граням пирамиды видимости или пока многоугольник не отсечётся полностью. На рисунках 6-7 приведены схемы алгоритма Сазерленда-Ходжмана:



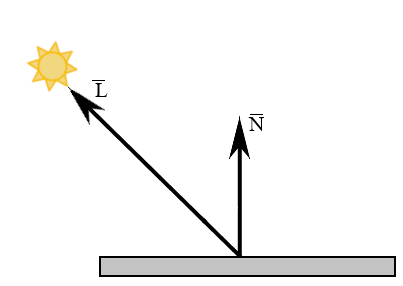
*Рисунок 6 Схема алгоритма Сазерленда-Ходжмана (часть 1)*



*Рисунок 7 Схема алгоритма Сазерленда-Ходжмана (часть 2)*

## 2.7 Реализация освещения

Была реализована модель Ламберта. Это простейшая модель освещения. Считается, что свет, падающий в точку, одинакового рассеивается по всем направлением полупространства. Таким образом, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения.[5] На рисунке 8 приведена модель освещения Ламберта:



*Рисунок 8 Модель освещения Ламберта*

Пусть

– вектор от точки поверхности до источника освещения,

– вектор нормали,

Формула расчёта интенсивности имеет следующий вид:

где I – результирующая интенсивность света в точке, I0 – интенсивность источника, Kd – коэффициент диффузного освещения.

Из формулы (12) следует главный недостаток модели Ламберта – одинаковая интенсивность во всех точках, принадлежащих одной грани.

Для реализации данной модели нормаль треугольника скалярно перемножается с вектором освещения. Результат векторного произведения является интенсивность освещения источником данного треугольника. Далее следует умножить полученную интенсивность на цвет треугольника.

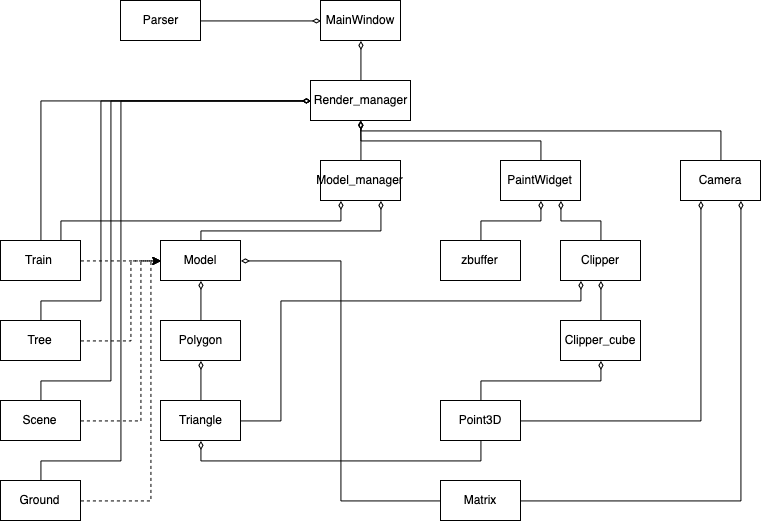
# Технологический раздел

Для написания данного курсового проекта в качестве языка программирования был выбран С++. Этот язык поддерживает объектно-ориентированную модель разработки, что позволяет структурировать программу и легко её модифицировать. Язык C++ позволяет эффективно использовать ресурсы системы благодаря широкому набору функций и классов.

Для реализации проекта была выбрана среда разработки Qt Creator. Данная среда обладает удобным редактором кода, отладчиком, дизайнером интерфейса. Главная причина выбора данной среды – полученный опыт работы во время курса “Компьютерной графики”.

## 3.1 Диаграмма классов

На рисунке 9 приведена диаграмма классов.



*Рисунок 9 Диаграмма классов*

Разработанная программа состоит из следующих классов:

Класс ввода:

* Parser – класс чтения модели из файла.

Базовые математические классы:

* Matrix – класс матриц;
* Point3D – класс точки пространства.

Классы объектов:

* Model – класс модели. Содержит все основные функции;
* Train – класс поезда. Отвечает за вращение колёс. Наследник Model;
* Tree – класс дерева. Наследник Model;
* Ground – класс поверхности земли. Наследник Model;
* Scene – класс сцены. В нём формируются все основные статические объекты сцены. Наследник Model.

Вспомогательные классы сцены:

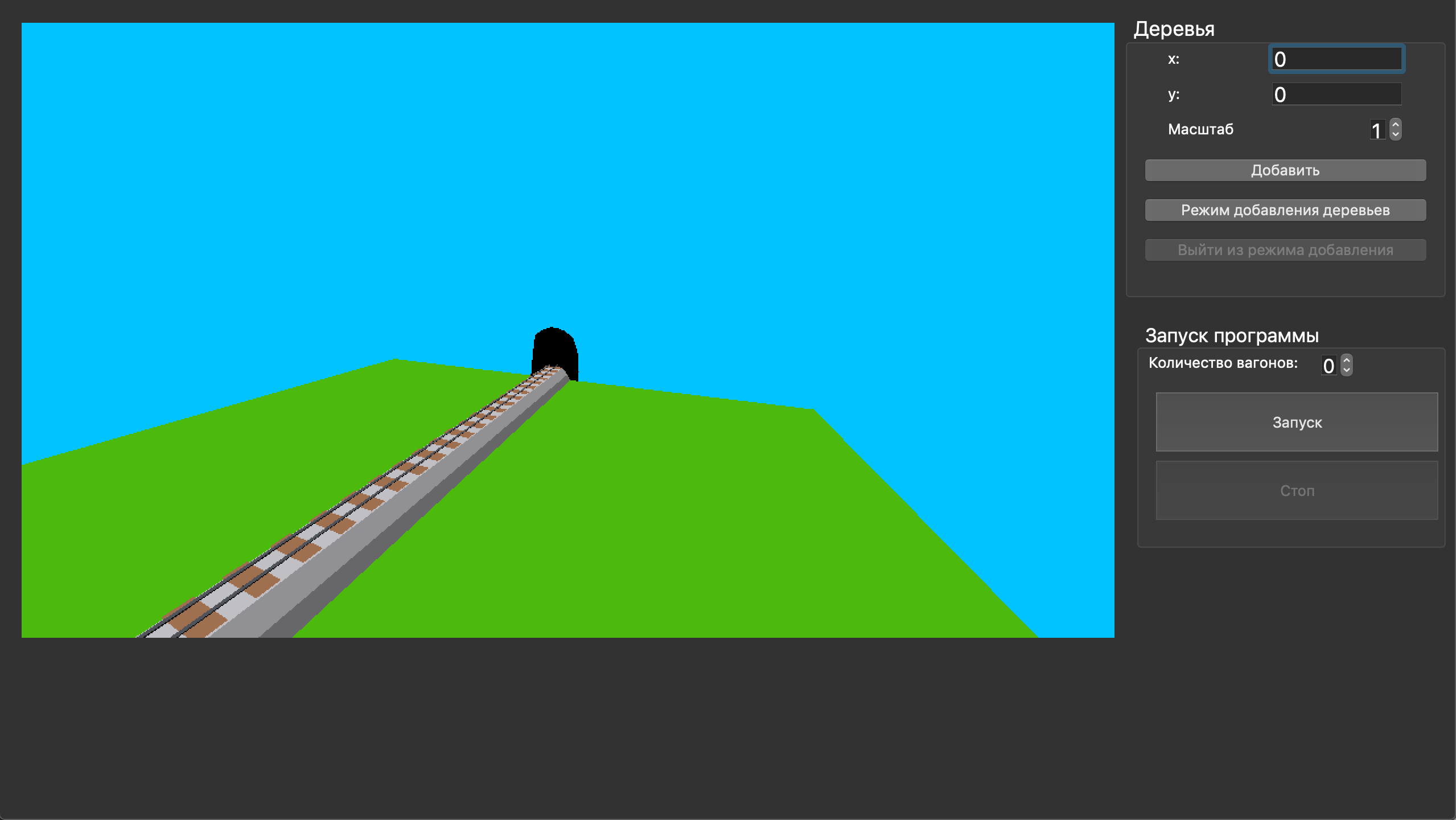
* Polygon – класс полигона. Содержит атрибуты позиции своего центра, свой цвет, имя и массив треугольников;
* Triangle – класс треугольника. Содержит точки-вершины треугольника;
* Camera – класс камеры с возможностью перемещения по сцене;
* Clipper – класс-отсекатель по алгоритму Сазерленда-Ходжмена;
* Clipper\_cube – класс, содержащий 4-хмерный куб для отсечения алгоритмов Сазерленда Ходжмена;
* Zbuffer – класс, содержащий буфер кадра. Используется в алгоритме z-буфера;
* Model\_manager – класс для осуществления преобразований модели.

Классы интерфейса:

* MainWindow – класс главного окна сцены;
* Render\_manager – класс посредник между интерфейсом и программой;
* PaintWidget – класс виджета для вывода буфера кадра.

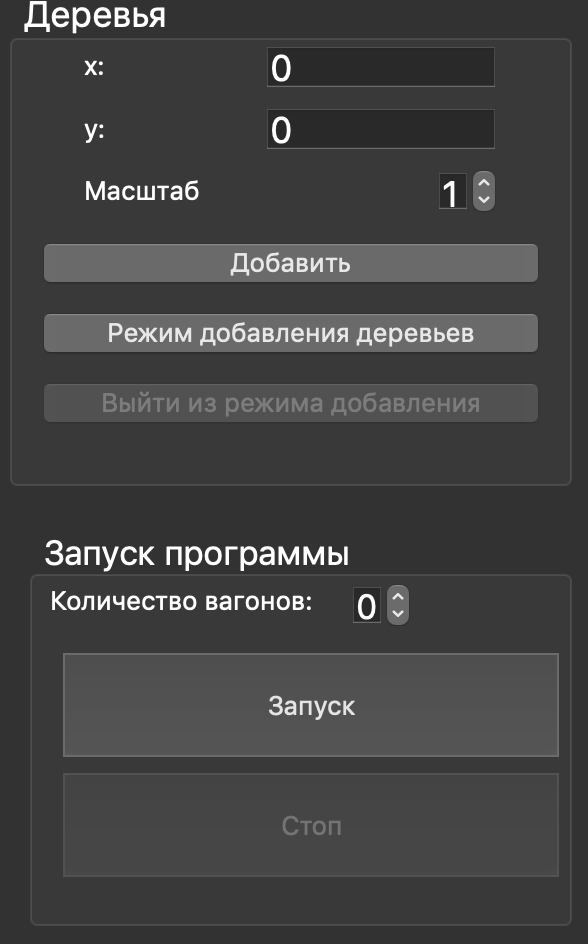
## 3.2 Интерфейс программы

При запуске программы в окне отображается плоскость земли, модель железной дороги и тоннели. Пользователь может взаимодействовать с программой с помощью панели управления и клавиш клавиатуры. На рисунке 10 приведён начальный экран программы.



*Рисунок 10 Начальный экран программы*

На рисунке 11 приведена панель управления программой.



*Рисунок 11 Панель управления*

Взаимодействие с камерой осуществляется через клавиши клавиатуры:

* Клавиша W – движение вперёд;
* Клавиша S – движение назад;
* Клавиша A – движение влево;
* Клавиша D – движение вправо;
* Клавиша Q – движение вверх;
* Клавиша E – движение вниз;
* Клавиши управления курсором – поворот камеры на соответствующие направление.

Используя панель управления, можно выбрать расстановку деревьев на поверхности земли. Если пользователь поставит дерево на поверхность железной дороги, то ему будет выведено сообщение об ошибке. Расстановка деревьев осуществляется через поля ввода координат x и у. Также можно выбрать из 3 вариантов размер дерева. По нажатию на кнопку “режим добавления деревьев” камера будет размещена над поверхностью земли. В этом режиме пользователь может, используя устройство ввода – мышь, нажимать на точку, в которой требуется расположить дерево. Пользователь может вернуться в исходное положение камеры, нажав на кнопку “выйти из режима добавления”.

Непосредственно перед запуском поезда, пользователь также может выбрать количество вагонов (от 0 до 5). По нажатию на кнопку “Запуск” начнётся движение поезда. Поезд пройдёт от одного тоннеля до другого. После пересечения плоскости тоннеля каждый вагон проходит некоторое расстояние, после чего перестаёт отображаться. После того, как последний вагон пройдёт тоннель, можно повторить запуск.

# Заключение

При написании курсового проекта были рассмотрены и проанализированы основные алгоритмы построения реалистичного трехмерного изображения: алгоритм Робертса, алгоритм Варнока, алгоритм трассировки лучей, алгоритм, использующий z-буфер, алгоритм простой закраски, алгоритмы закраски по Гуро и по Фонгу. Проанализированы их достоинства и недостатки, и возможность их использования для решения поставленной задачи. Выбранные алгоритмы – алгоритм z-буфер и простой алгоритм закраски – были реализованы.

Разработанная программа позволяет получать на экране дисплея модель движения поезда по железной дороге. Движение поезда было реализована с использованием операций переноса и поворота. Операции переноса были использованы, например, при движении корпуса. Операции поворота использовались при повороте колёс состава.

В интерфейсе программы были добавлены возможности выбора количества вагонов поезда и расположения деревьев. Расстановку деревьев можно производить, используя устройства ввода клавиатуру или мышь. С помощью поля масштаб регулируется размер деревьев.

Приложение имеет несколько направлений дальнейшего развития. Например, на данном этапе не была реализована возможность пользовательского импорта моделей, наложения текстур.

# Список использованной литературы

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. Computer Graphics from scratch URL: https://www.gabrielgambetta.com/computer-graphics-from-scratch/introduction.html (дата обращения: 13.12.19).
4. Learn Computer Graphics From Scratch! URL: https://www.scratchapixel.com/index.php (дата обращения: 17.12.19).
5. Митин А.И., Свертилова Н.В. Компьютерная графика. - 2-е изд. - Москва, Берлин: Директ-Медиа, 2016. – 256с.
6. Сиротин Э.Е., Перфильев Д.А., Кушнаренко А.В. Компьютерная геометрия и графика. - СПб.: Машиностроение, 1980. - 66 с.