|  |  |
| --- | --- |
| Изображение выглядит как эмблема, символ, нашивка, герб  Автоматически созданное описание | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова» (БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-02 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Факультет |  | О |  | | Естественнонаучный |
|  |  | шифр |  | | наименование |
| Кафедра |  | О7 |  | | Информационные системы и программная инженерия |
|  |  | шифр |  | | наименование |
| Дисциплина |  | | | Компьютерная геометрия и графика | |

КУРСОВАЯ РАБОТА

на тему

|  |
| --- |
| Разработка трехмерной сцены «Зимнее танковое сражение» |
| с использованием библиотеки OpenGL |
|  |

|  |  |
| --- | --- |
| Выполнил студент группы | И913Б |
| Калягин В.Д. |  |
| Фамилия И.О. |  |
| **РУКОВОДИТЕЛЬ** | |
| Снижко Е.А. | |
| Фамилия И.О. Подпись | |
| Оценка | |
| « » | 20 г. |

**РЕФЕРАТ**

Отчет 17 с.,9 рис., 6 источн., 1 прил.

КОМПЬЮТЕРНАЯ ГРАФИКА, OPENGL, ШЕЙДЕРЫ, С++

В ходе выполнения курсовой работы была разработана программа, реализующая вывод на экран трехмерной сцена с возможностью перемещения по ней и взаимодействия.

**Содержание**

[Введение 4](#_Toc152864335)

[1 Постановка задачи 5](#_Toc152864336)

[1.1 Описание разрабатываемой программы 5](#_Toc152864337)

[1.2 Формальное описание поставленной задачи 5](#_Toc152864338)

[2 Описание разработанной программы 7](#_Toc152864339)

[2.1 Описание используемого алгоритма отображения сцены 7](#_Toc152864340)

[2.2 Описание алгоритма загрузки объектов 9](#_Toc152864341)

[2.3 Описание используемых шейдеров 10](#_Toc152864342)

[2.4 Описание дополнительных используемых алгоритмов и функций 10](#_Toc152864343)

[3 Использованные мультимедийные ресурсы и сторонние библиотеки 11](#_Toc152864344)

[4 Демонстрация работы 12](#_Toc152864345)

[Заключение 15](#_Toc152864346)

[Список использованных источников 16](#_Toc152864347)

[ПРиложение A 17](#_Toc152864348)

# **Введение**

Компьютерная графика — это область графики, предполагающая создание изображений при помощи компьютера. Одной из самых популярных библиотек для создания компьютерной графики является библиотека *OpenGL.* Она предоставляет собой высокопроизводительный инструмент, позволяющий создавать двумерные и трехмерные сцены различной сложности.

Целью курсовой работы является разработка трехмерной сцены «Зимнее танковое сражение».

Для достижения поставленной цели необходимо было решить следующие задачи:

* реализовать алгоритм загрузки моделей объектов сцены из файлов формата *OBJ*;
* реализовать загрузку текстур и их наложение на объекты сцены;
* разработать алгоритмы перемещения танка игрока;
* разработать алгоритмы перемещения камеры от 3-го лица;
* реализовать освещение сцены, танков;
* разработать алгоритм вращения башни танка игрока.

1. **Постановка задачи**

## **1.1 Описание разрабатываемой программы**

Разрабатываемая программа должна выводить на экран трехмерную сцену, содержащую танки «КВ-2»[1], «ПЦ3» и «Тигр». Также необходимо создать окружение в виде неба, ёлок и заснеженного ландшафта.

Необходимо реализовать перемещение игрока по сцене на танке «КВ-2» с разной скоростью, а также ограничить возможность его выезда за пределы сцены. Также вместе с танком игрок должны перемещаться камера и башня. Камера должна иметь вид от 3-го лица (снаружи танка). Башня должна вращаться вокруг своей оси вслед за камерой.

Сцена должна иметь дневное освещение.

Необходимо реализовать алгоритмы анимации вращения башни танка игрока.

## **1.2 Формальное описание поставленной задачи**

Необходимо разработать программу, которая будет выводить на экран трехмерную сцену, состоящую из нескольких объектов. Модели данных объектов должны загружаться из файлов формата *OBJ*[2]. Также необходимо загружать их текстуры и материалы. Должен быть реализован алгоритм вывода загруженных объектов на экран, наложение на них текстур и материалов.

Необходимо реализовать перемещение объекта танка игрока по карте с заданной скоростью, возможность изменять данную скорость и направление перемещения. Также необходимо реализовать расчет видовой матрицы для режима от 3-го лица.

Необходимо реализовать возможность вращения башни вокруг своей оси в соответствии с направлением камеры игрока.

Необходимо расположить на сцене фоновый, точечные и направленные источники света, реализовать шейдер, который просчитывает освещение для всех выводимых на экран объектов. Также должно быть реализовано перемещение источников света вместе с объектами, к которым они прикреплены.

Необходимо разработать алгоритмы расчета аффинных преобразований для всех объектов сцены согласно времени сцены.

1. **Описание разработанной программы**

## **2.1 Описание используемого алгоритма отображения сцены**

Для отрисовки сцены в программе создан класс *Scene*. Диаграмма данного класса представлена на рисунке 2.1.

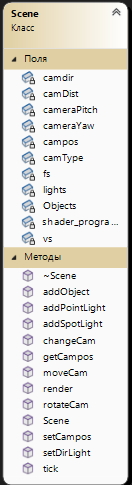


Рисунок 2.1 – Диаграмма класса *Scene*

Данный класс содержит в себе массив *Objects*, содержащий в себе все объекты сцены. Классы объектов являются наследниками класса *BaseObject*. Диаграмма всех классов объектов представлена на рисунке 2.2.

Изображение выглядит как текст, снимок экрана, Прямоугольник, дизайн

Автоматически созданное описание

Рисунок 2.2 – Диаграмма классов объектов

Каждый кадр сцены вызывается 2 метода сцены: *render(int WW,int WH)*, принимающий размеры окна и выводящий на экран все объекты сцены. Данный метод производит расчет видовой матрицы и матрицы проекции и вызывает метод *draw* каждого объекта массива *Objects*.

Также каждый кадр вызывается метод *tick(float tickl)*, принимающий длительность кадра и передающий всем объектам сцены длительность кадра, что обеспечивает их анимацию и перемещение.

Для вывода на экран объекты имеют метод *draw(glm::mat4 projection, glm::mat4 view, glm::vec3 campos, LightsInfo lights, GLuint shader\_programme)*, который принимает матрицы проекции, вида, информацию об источниках света и шейдерную программу. Данный метод вызывает метод *draw* каждой своей составной части и передает ей все свои параметры, а также матрицу модели объекта. Составные части объекта хранятся в массиве *subObjects* и представлены классм *SubObject*, диаграмма которого показана на рисунке 2.3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.3 – Диаграмма класса составной части объекта.

Класс *subObject* имеет *метод draw(glm::mat4 projection, glm::mat4 view, glm::mat4 model, glm::vec3 campos, LightsInfo lights, GLuint shader\_programme)*, который принимает матрицы проекции, вида, модели информацию об источниках света и шейдерную программу. Данный метод передает в шейдеры свои материалы, текстуры и источники света сцены. Далее при помощи функции *glDrawArrays* составная часть объекта выводится на экран.

## **2.2 Описание алгоритма загрузки объектов**

Для загрузки объекта из файла класс *Object* имеет метод *LoadFromOBJ(std::string filename, std::vector<SubObject\*>\* so, std::vector<Material>\* mtrs)* который принимает название файла и загружает из него материалы и составные объекты в соответствующие массивы. Из данного файла программа получает массивы координат вершин, текстурных координат и нормалей, а также массив индексов координат вершин, текстурных координат и нормалей. Поскольку Библиотека *OpenGL* не поддерживает только индексы для координат вершин данный метод преобразует их в 3 массива, не требующих индексов. Также данный метод получает из файла названия файлов с текстурами и загружает их.

Класс *SubObject* имеет конструктор *SubObject(std::vector<glm::vec3>\* points, std::vector<glm::vec2>\* textures, std::vector<glm::vec3>\* normals,Material \*mtr)* который принимает массивы координат вершин, нормалей и текстурных координат, а также материал объекта. Из данных массивов он генерирует буфер вершин, который уже используется при отображении сцены.

## **2.3 Описание используемых шейдеров**

В программе используется 2 шейдера: вершинный и фрагментный.

Вершинный шейдер принимает координаты вершины, ее нормаль и текстурные координаты, а также матрицы модели, вида и проекции. В нем производится преобразование нормали вершины и ее координат в соответствии с матрицей модели, и расчет положения вершины на экране в соответствии и матрицами модели, вида, проекции. Далее он передает в вершинный шейдер нормаль, координаты вершины и ее текстурные координаты.

Фрагментный шейдер принимает нормаль, координаты вершины, текстурные координаты, материал, информацию об источниках света и положение камеры. При помощи этого рассчитывается цвет для каждой точки на экране путем суммирования цвета, полученного фоновой, диффузной и отраженной компоненты света[3].

## **2.4 Описание дополнительных используемых алгоритмов и функций**

Также стоит упомянуть об алгоритмах расчета положения камеры, перемещения танка и вращения его башни.

Камера имеет вид от 3-го лица. Камера всегда направлена в одну точку, но ее положение при вращении изменятся. Для такого вида можно использовать сферическую систему координат. Центром сферы станет точка, в которую камера направлена, а положение камеры будет вычисляться углами тангажа, рыскания и радиуса сферы. Радиус задает удаленность камеры от 3-го лица.

1. **Использованные мультимедийные ресурсы и сторонние библиотеки**

При написании курсовой работы были использованы библиотеки *TinyOBJLoader*[4]и *STB*[5].

Библиотека *TinyOBJLoader* предоставляет широкий инструментарий для загрузки трехмерных моделей из файлов *OBJ* формата. Она позволяет загружать не только сам объект, но и его материалы из файла формата *MTL*. В данной программе используется функция данной библиотеки, которая принимает название файла, и создает 3 структуры: структура, содержащая все вершины, структура, содержащая индексы вершин всех объектов и структура материалов. Также данная функция способна выполнять триангуляцию, что необходимо, так как *OpenGL*[6] выводить только полигоны, имеющие 3 вершины.

*STB* – это совокупность библиотек, состоящих из одного заголовочного файла и выполняющих различные функции. В данной программе использована библиотека *STB\_Image*, позволяющая загружать картинки из файлов. Она содержит функцию *stbi\_load* которая принимает названия файла с картинкой и загружает ее в массив байт. Данная функция используется для загрузки текстур.

Обе библиотеки бесплатные и имеют открытый исходный код.

Также в данной программе используются трехмерные модели, которые находятся в открытом доступе на сайте «*Open3DModel*». Данный сайт содержит большую базу трёхмерных моделей, распространяемых бесплатно.

**4 Демонстрация работы**

На рисунке 4.1 представлена сцена в режиме камеры от 3-го лица.



Рисунок 4.1 – Режим камеры от 3-го лица

На рисунке 4.2 представлен танк противника «Тигр».

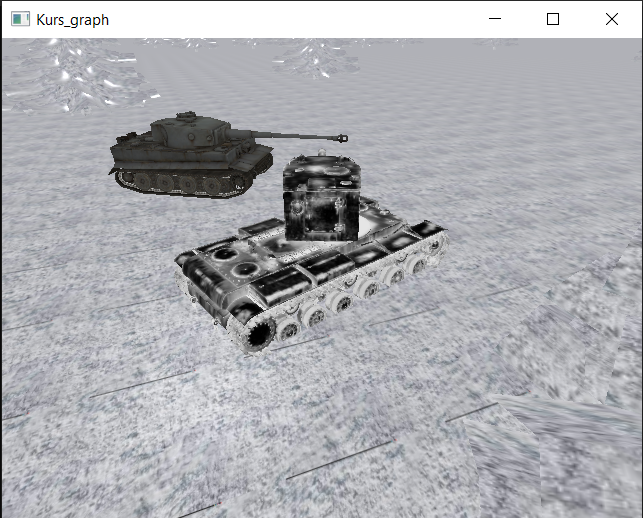


Рисунок 4.2 – Танк противника «Тигр».

На рисунке 4.3 представлена танк противника ПЦ3.



Рисунок 4.3 – Танк противника «ПЦ3»

На рисунке 4.4 показано зимнее окружение.

Изображение выглядит как снимок экрана, снег, небо, на открытом воздухе

Автоматически созданное описание

Рисунок 4.4 – Зимнее окружение

На рисунках 4.5 и 4.6 показан поворот башни танка игрока.

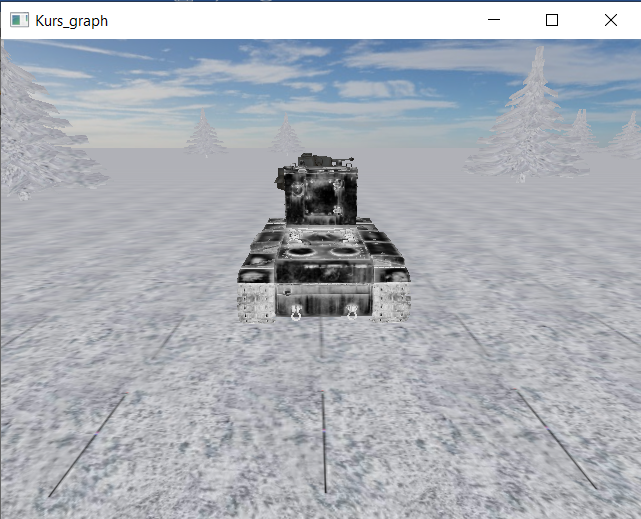


Рисунок 4.5 – Изначальное положение башни танка игрока

Изображение выглядит как снимок экрана, небо, снег, на открытом воздухе

Автоматически созданное описание

Рисунок 4.5 – Башня после поворота вокруг своей оси

# **Заключение**

В ходе выполнения курсовой работы была создана сцена «Зимнее танковое сражение».

Для загрузки моделей была использована библиотека *TinyOBJLoader*. Для загрузки текстур использовалась библиотека *STB\_Image*. Для реализации перемещения танка, камеры и поворота башни были разработаны соответствующие алгоритмы. Для расчета освещения и наложения текстур была разработана шейдерная программа.

Курсовая работа выполнена в *IDE Microsoft Visual Studio 2022* на языке *С*++ с использованием графической библиотеки OpenGL 3.0.

# **Список использованных источников**

1. Сайт «Энциклопедия военной техники». — URL: https://war-book.ru/linejnye-korabli-tipa-yamato/ (дата обращения 07.12.2023).
2. Справка с различными форматами файлов. — URL: https://people.sc.fsu.edu/~jburkardt/data/data.html (дата обращения 07.12.2023).
3. Статья про освещение с сайта «Компьютерная графика». — URL: https://compgraphics.info/3D/lighting/light\_sources.php (дата обращения 07.12.2023).
4. Документация библиотеки TinyOBJLoader. — URL: .https://github.com/tinyobjloader/tinyobjloader (дата обращения 07.12.2023).
5. Документация библиотеки STB. — URL: https://github.com/nothings/stb/blob/master/README.md (дата обращения 07.12.2023).
6. Документация библиотеки OpenGL. — URL: . https://opengl.org.ru/ (дата обращения 07.12.2023).
7. Сайт «Open3DModel». — URL: https://open3dmodel.com/

(дата обращения 07.12.2023).

# **ПРиложение A**

Исходный код программы предоставлен на репозитории *https://github.com/BeerStrike/Crsch\_4*.