МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

ОТЧЕТ
по курсовой работе

Тема Разработка трехмерной сцены «Жизнь Пи»
с использованием библиотеки OpenGL

Обучающегося группы О729Б (группа) Веремчук А.О. (Фамилия и инициалы)

Направление подготовки / специальность 09.03.04 (индекс) Программная инженерия (полное наименование направления подготовки / специальности)

Направленность образовательной программы **Разработка программно-информационных систем** (профиль / специализация / магистерская программа)

Дисциплина (модуль) Компьютерная геометрия и графика

Руководитель: _____ (подпись)

к.пед.н., доцент (ученая степень, ученое звание) Снижко Е.А. (Фамилия ИО)

Оценка: _____

« » _____ 20 24 г.

Обучающийся: _____ (подпись)

Веремчук А.О. (Фамилия ИО)

« » _____ 20 24 г.

САНКТ-ПЕТЕРБУРГ
2024 г.

РЕФЕРАТ

Пояснительная записка 20 с., 10 рис., 8 источников, 1 приложение.

OPENGL, ТЕКСТУРА, ШЕЙДЕР, МАССИВ, ВЕКТОР

Объектом разработки является трёхмерная сцена «Жизнь Пи».

Целью курсовой работы является разработка трехмерной сцены «Жизнь Пи».

В ходе выполнения курсовой работы была создана трёхмерная сцена с использованием графической библиотеки OpenGL. На объекты сцены была наложена текстура, заданы источники света, реализована анимация, а также управление объектами и камерой с помощью клавиатуры.

В результате курсовой работы была разработана трёхмерная сцена «Жизнь Пи» в IDE Microsoft Visual Studio 2022 на языке C++ с использованием графической библиотеки OpenGL.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютерная графика используется в таких сферах, как кино, мультфильмы, компьютерные игры, приложения и многое другое [1]. Это высоко востребованное умение в современном мире, поэтому владение навыками визуальной графики полезно разработчикам [2].

Целью курсовой работы является разработка трехмерной сцены «Жизнь Пи».

Для достижения поставленной цели необходимо было решить следующие задачи:

- создать объекты сцены из фильма «Жизнь Пи» с использованием графических возможностей библиотеки OpenGL;
- реализовать наложение текстур на объекты;
- разработать алгоритмы анимации перемещения кита, волн, лодки, тигра и человека;
- реализовать освещение Луны, океана и кита с помощью средств библиотеки OpenGL;
- разработать модуль камеры для визуального взаимодействия с трехмерной моделью «Жизнь Пи»;
- разработать алгоритмы управления и взаимодействия со сценой, ее объектами и камерой.

1 Постановка задачи

В рамках курсовой работы необходимо разработать трёхмерную сцену «Жизнь Пи» [3]. Сцена представляет собой кадры из фильма «Жизнь Пи», где в ночном океане главные герои – тигр и человек (Пи), сидящие в лодке, встречают кита, который проплывает рядом с ними. Сцена состоит из восьми объектов: сферы, двух прямоугольников океана, лодки, тигра, человека, Луны и кита. В рамках сцены должны двигаться прямоугольники океана, чтобы осуществлять движение волн; лодка вместе с тигром и человеком должна покачиваться из стороны в сторону, имитируя качку; кит должен при нажатии клавиши «2» осуществлять следующее движение: опускаться вертикально под воду, на середине пути от своего начального положения до лодки показывать спину над водой, около лодки подниматься вертикально вверх из воды и потом опускаться обратно. Так же должно быть реализовано движение камеры во всех направлениях и возвращение её к начальной позиции при нажатии клавиши «1».

2 Моделирование сцены и отдельных объектов

Для моделирования сцены были скачаны с интернет-ресурса Free3D файлы формата obj для четырёх объектов: лодки, тигра, человека и кита [4]. Эти объекты были загружены с помощью функции LoadOBJ, которая принимает следующие параметры: расположение файла, массив вершин, текстурные координаты, массив нормалей.

Для начала был создан вектор, отвечающий за координаты расположения объекта, для последующей анимации. Потом с помощью функции LoadOBJ считывался файл модели и загружались данные объекта, такие как: координаты вершин, нормали и текстурные координаты. Последний этап – это загрузка объекта в шейдерную программу. Загрузка модели представлена во фрагменте программы ниже.

```
    // Задаём позицию лодки
    glm::vec3 boatPosition(0.0f, 0.0f, 0.0f);
    //Загрузка координат из файла
    loadOBJ("Objects\\boat.obj",           temp_vertices,
temp_textures, temp_normals);
    size = temp_vertices.size();
    //Создание модели
    Model boat(window,1);
    //Загрузка данных модели
    boat.load_coords(temp_vertices.data(), size);
    boat.load_normals(temp_normals.data(),
temp_normals.size());
    boat.load_texcoord(temp_textures.data(),
temp_textures.size());
    // загрузка в шейдер
    boat.load_shaders("matvs.glsl", "matfs.glsl");
```

После загрузки модели происходит отрисовка объектов в основном цикле программы. Первым этапом идёт очистка матрицы вида, потом

преобразование модели с помощью аффинных преобразований: для сдвига модели использовалась функция glm::translate, в которую передавались координаты расположения модели; для поворота использовалась функция glm::rotate, с помощью неё удалось расположить объекты на осях согласно сценарию сцены; для масштабирования использовалась функция glm::scale. Последний этап – это отрисовка модели, с помощью функции render, в параметры которой передаётся матрица вида, матрица представления, матрица проецирования, свет и примитив. Отрисовка объекта показана во фрагменте программы ниже.

```
    // ЛОДКА
    // очищение матрицы вида
    MMatr = WorldMatrix;
    // Ниже представлены Аффинные преобразования
    // перемещение объекта
MMatr = glm::translate(glm::mat4(1.0f), boatPosition);
    //лодка качается
    //повороты и вращение
    MMatr = glm::rotate(MMatr, glm::radians(rotation),
glm::vec3(0.0, 0.0, 1.0));
    MMatr = glm::rotate(MMatr, glm::radians(-90.0f),
glm::vec3(1.0, 0.0, 0.0));
    MMatr = glm::rotate(MMatr, glm::radians(-90.0f),
glm::vec3(0.0, 0.0, 1.0));
    // масштабирование
    MMatr = glm::scale(MMatr, glm::vec3(0.01, 0.01,
0.01));
    // отрисовка
    boat.render(MMatr, VMatr, PMatr, Lights,
GL_TRIANGLES);
```

Сфера, представляющая собой небо, и Луна были отрисованы с помощью функции genSphere, которая принимает такие параметры: массив индексов, массив вершин, массив текстурных координат, массив нормалей, радиус, количество медиан и параллелей. Две плоскости были созданы с помощью набора вершин, текстур и индексов, задающих порядок обхода вершин в массиве.

3 Наложение текстуры

Для большей реалистичности объектов сцены с интернет-ресурсов [5]-[7] были скачены текстуры для объектов: сферы, двух плоскостей, океана, Луны, тигра, человека и кита. У лодки был задан материал.

При загрузке объектов с текстурами были использованы текстурные фрагментный и вершинный шейдеры. При создании модели окна, указывалась, какая будет модель, для правильности выбора шейдера (0 – цветная, 1 – с материалом, 2 – с текстурой). Во фрагменте программы ниже представлен вершинный шейдер для наложения текстуры.

```
#version 400
in vec3 vertex_position;
in vec3 vertex_normal;
in vec2 vertex_texture;
uniform mat4 M;
uniform mat4 V;
uniform mat4 P;
out vec3 normal;
out vec3 f_pos;
out vec2 TexCoord;
void main() {
    //Преобразование координат фрагмента с учётом матрицы модели
    f_pos = vec3(M * vec4(vertex_position, 1.0));
    //Преобразование координат нормалей с учётом матрицы модели
    normal = mat3(transpose(inverse(M))) * vertex_normal; ;
    TexCoord = vertex_texture;
    //Обычное вычисление координат фрагмента
    gl_Position = P * V * vec4(f_pos, 1.0);}
```

На сферу была наложена текстура звёздного неба, так как действие сцены происходит ночью, рисунок 1.

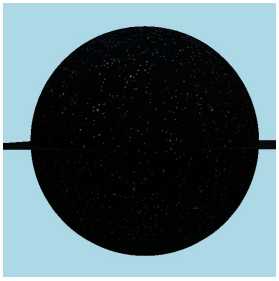

Рисунок 1 – Сфера (звёздное небо)

На две плоскости океана была наложена тёмная текстура волн, имитирующая ночной океан, рисунок 2.

Рисунок 2 – Две плоскости океана

На модель тигра была наложена полосатая текстура шерсти тигра, рисунок 3.

Рисунок 3 – Тигр

На модель человека была наложена текстура кожи, схожая с цветом кожи главного героя индуса Пи, рисунок 4.

Рисунок 4 – Человек

На сферу, имитирующую Луну, была наложена текстура лунной поверхности, рисунок 5.

Рисунок 5 – Луна

На модель кита была наложена серая текстура кожи, похожая на кожу серого кита, рисунок 6.

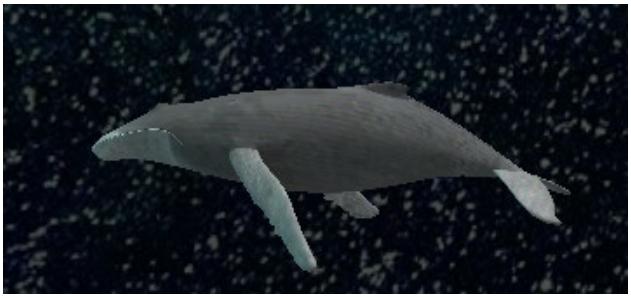

Рисунок 6 – Кит

У модели лодки был задан материал, имитирующий железо, покрашенное белой краской с помощью функции setMaterial. Первый параметр – фоновая компонента света, светло-серый цвет; второй параметр – диффузная компонента света, белый цвет; третий параметр – отражённая компонента света, светло-серый; четвёртый параметр – степень отражения [8]. Задание материала лодки показано во фрагменте кода ниже.

```
boat.setMaterial(glm::vec3(0.9f, 0.9f, 0.9f), glm::vec3(1.0f, 1.0f, 1.0f), glm::vec3(0.9f, 0.9f, 0.9f), 128 * 0.5);
```

Модель лодки с заданным материалом показана на рисунке 7.

Рисунок 7 – Лодка

4 Освещение

Для освещения объектов сцены было задано фоновое освещение и точечное освещение трёх объектов: Океана, Луны и кита.

Чтобы создать эффект флуоресцентной воды около лодки, голубой источник света был помещён под неё. Для достижения лазурного цвета параметры фонового света ambient и диффузного света diffuse отражают только голубой и зелёные цвета, параметр отражённого света specular задавал белый цвет бликов. Фрагмент программы, задающий голубой цвет воды, располагается ниже.

```
//ГОЛУБОЙ
P.ambient = glm::vec3(0.0f, 0.1f, 0.1f);
P.diffuse = glm::vec3(0.0f, 0.8f, 0.8f);
P.specular = glm::vec3(1, 1, 1);
P.position = glm::vec3(-1, -10, 0);
P.constant = 1;
P.linear = 0.01;
P.quadratic = 0.000000001;
Lights.pointLights.push_back(P);
```

Флуоресцентная вода, заданная с помощью голубого источника света показана на рисунке 8.

Рисунок 8 – Голубой свет

У Луны и кита был задан белый свет. Но у Луны квадратичная составляющая затухания quadratic меньше, чем у кита, это означает, что затухание света будет очень слабым при увеличении расстояния, и свет Луны будет сильнее. Фрагмент кода, описывающий свет Луны представлен ниже.

```
// БЕЛЫЙ 1 ЛУНА
P.ambient = glm::vec3(1.0f, 1.0f, 1.0f);
P.diffuse = glm::vec3(1.0f, 1.0f, 1.0f);
```

```
P.specular = glm::vec3(1, 1, 1);
P.position = { -200, 200, -50 };
P.constant = 1;
P.linear = 0.01;
P.quadratic = 0.00001;
Lights.pointLights.push_back(P);
```

Свет, исходящий от Луны, представлен на рисунке 9.

Рисунок 9 – Белый свет Луны

Свет, исходящий от кита, представлен на рисунке 10.

Рисунок 10 – Белый свет кита

5 Анимация

Для достижения движения волн в сцене, две плоскости океана, с наложенной текстурой волн, должны осуществлять движение друг за другом: когда плоскость выходит за пределы сферы – достигает значения по оси x большего, чем 1000, она перемещается за другую плоскость. Происходит движение по кругу. Фрагмент кода, реализующий смену координат волн, приведён ниже.

```
    sea1Position.x += speed_2 * deltaTime; // Двигаем объект по X
    sea2Position.x += speed_2 * deltaTime; // Двигаем объект по X
    if (sea1Position.x >= 1000) {
        sea1Position.x = -1000;
    }
    if (sea2Position.x >= 1000) {
        sea2Position.x = -1000;
    }
```

Покачивание на волнах лодки, тигра и человека осуществляется из стороны в сторону на 15 градусов у лодки и на 5 градусов у тигра и человека по оси z. Параметры rotation и rotation2 передаются в функцию glm::rotate лодки, тигра и человека, соответственно. Фрагмент программы, в которой задаются параметры поворота представлен ниже.

```
    float rotation = 15.0f * sin(elapsed);
    float rotation2 = 5.0f * sin(elapsed);
```

Кит при нажатии клавиши «2» осуществляет следующее движение: опускается вертикально под воду, на середине пути от своего начального положения до лодки показывает спину над водой, около лодки поднимается вертикально вверх из воды и потом опускается обратно. Смена движения кита осуществляется при помощи флага. Значение флага меняется в каждом блоке if-else. Пример задания движения кита представлен во фрагменте кода ниже.

```
    if (glfwGetKey(window, GLFW_KEY_2) == GLFW_PRESS flag
= 1;
    if (flag == 1) {
        whalePosition.y += -speed3 * deltaTime;
        if (whalePosition.y < -20) {
            flag = 2;
        }
    }
```

6 Управление с клавиатуры

Для управления камерой в сцене задаются следующие основные параметры: скорость камеры speed, её чувствительность sensitivity, позиция камеры Position, ориентация камеры Orientation и вектор «вверх», взятый по оси y. Задание параметров показано во фрагменте программы ниже.

```
float speed = 0.4f;
float sensitivity = 100.0f;
glm::vec3 Position = { 9.23859,2.28816, 0.707238 };
glm::vec3 Orientation = { -0.997465, -0.00681765, -0.070843 };
glm::vec3 Up = glm::vec3(0.0f, 1.0f, 0.0f);
glm::mat4 cameraMatrix = glm::mat4(1.0f);
```

Движение камеры осуществляется с помощью клавиш:

– W – вперёд;
– A – влево;
– S – назад;
– D – вправо;
– Ctrl – вверх;
– Alt – вниз;
– Shift – увеличение скорости камеры.

Так же камеру можно поворачивать с помощью мыши. Пример цикла для управления камерой показан во фрагменте программы ниже.

```
if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
 {
     Position += speed * Orientation;
 }
```

Управление сценой с помощью клавиш осуществляется с помощью клавиш «1» и «2». При нажатии клавиши «1», камера возвращается на свою начальную позицию Position, а при нажатии клавиши «2», значение флага меняется, и кит начинает движение.

ЗАКЛЮЧЕНИЕ

Сцена «Жизнь Пи» смоделирована при помощи 3D-объектов, таких как лодка, человек, тигр, кит, загруженных при помощи функции LoadOBJ из файлов, сфер, смоделированных функцией genSphere и прямоугольников, созданных при помощи массива координат.

Для придания идентичности объектам были использованы текстуры поверхности неба, Луны, шерсти и кожи. Для загрузки текстур использовалась функция load_texture.

Для освещения сцены использовался фоновый свет и три источника точечного света, у которых были заданы параметры фонового, диффузного и отражённого светов для каждого объекта.

Для анимации в сцене «Жизнь Пи» изменялись координаты объектов, а также их расположение относительно осей координат. Изменённые параметры передавались в функции аффинных преобразований каждого движущегося объекта.

Управление камерой в сцене осуществлялось через клавиатуру и мышь при помощи передачи изменённых параметров в функцию lookAt, движение объектов осуществлялось через изменение параметров функций аффинных преобразований.

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

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Scream School. – URL: https://scream.school/blog/computergraphics (дата обращения 9.11.2024).

2. Академия TOP. – URL: https://arh.top-academy.ru/blog/computer-graphics-designer-the-subtleties-of-the-profession-and-scope (дата обращения 9.11.2024).

3. Кинопоиск. – URL: https://www.kinopoisk.ru/film/158786/?utm_referrer=www.google.com (дата обращения 1.11.2024).

4. Free3D. – URL: https://free3d.com/ru/3d-models/obj (дата обращения 3.11.2024).

5. Pixabay. – URL: https://pixabay.com/ru/images/search (дата обращения 3.11.2024).

6. Goodfon. – URL: https://www.goodfon.ru/textures/wallpaper-animal-texture-shkura-fur-tigr-mekh.html (дата обращения 3.11.2024).

7. Stock.liga.ws. – URL: https://stock.liga.ws/products/textures/%D0%BD%D0%B5%D0%B1%D0%BE-7 (дата обращения 3.11.2024).

8. OpenGL/VRML Materials. – URL: http://devernay.free.fr/cours/opengl/materials.html (дата обращения 5.11.2024).

ПРИЛОЖЕНИЕ А

Текст программы

Исходные тексты программы располагаются на прилагаемом электронном носителе.