

УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ
И КОМПЬЮТЕРНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КОМПЬЮТЕРНАЯ ГРАФИКА
ЛАБОРАТОРНАЯ РАБОТА

Выполнили:

Милосердов А. О.

Калугин Ф. И.

Группа Р3410

Преподаватель:

Бессмертный И. А.

Санкт-Петербург

2017 г.

Описание

Был реализован кроссплатформенный движок для FPS игр.

Особенности:

- Сборка и установка игровых файлов в игровой каталог (шейдеры, текстуры, карты, бинарь) на Windows и Linux.
- Высокоуровневый объектно-ориентированный Modern C++ API для работы с графикой.
- FPS камера.
- Использование различных текстур, модель освещения Phong, динамические тени (Shadow mapping).
- Обработка коллизий игрока и объектов мира.
- Физика гравитации.
- Загрузка карт мира в формате BMP.

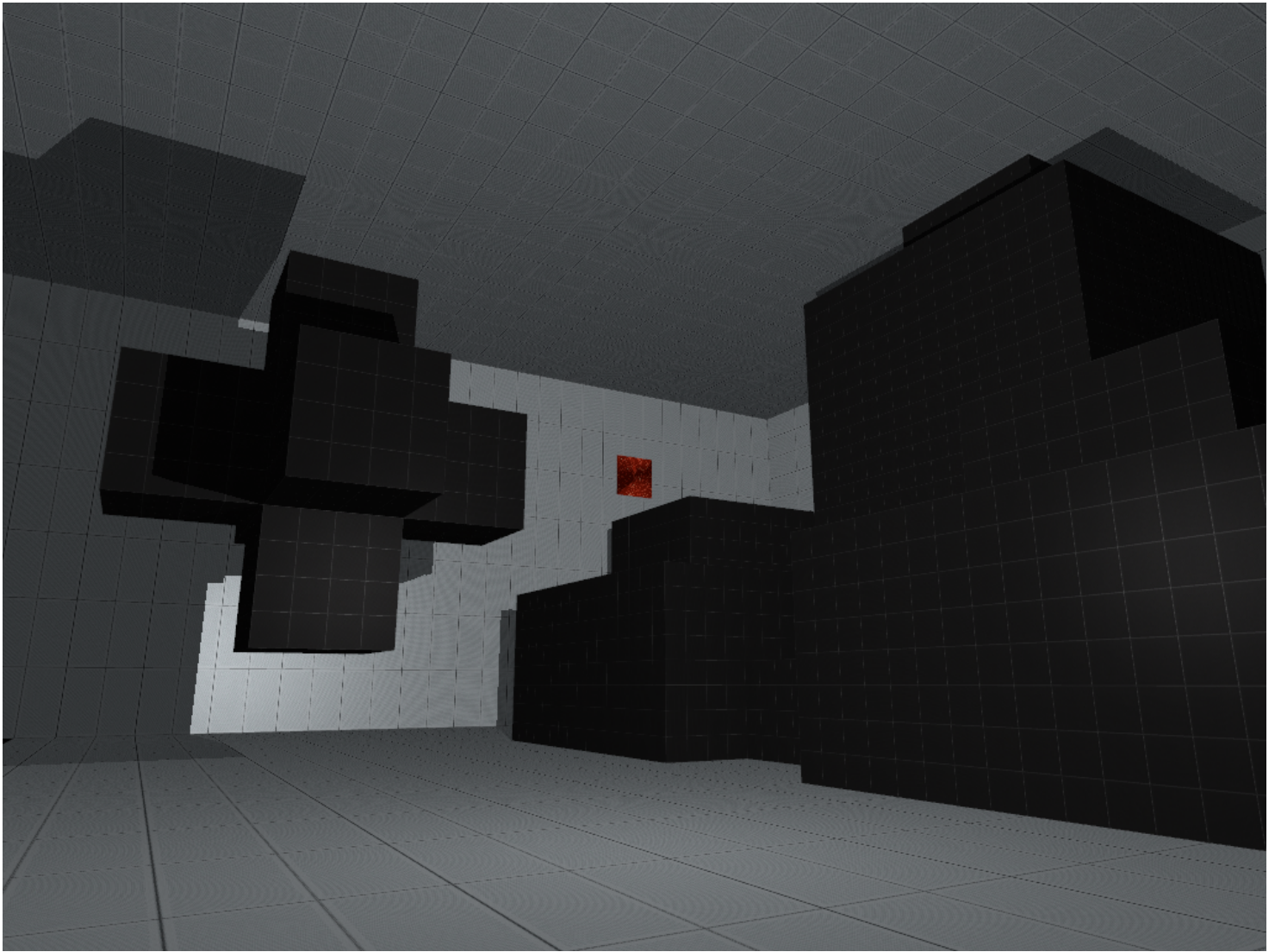


Рис. 1: Скриншот демонстрационного проекта (игрок выстрелил красным октаэдром который в полете испускает свет)

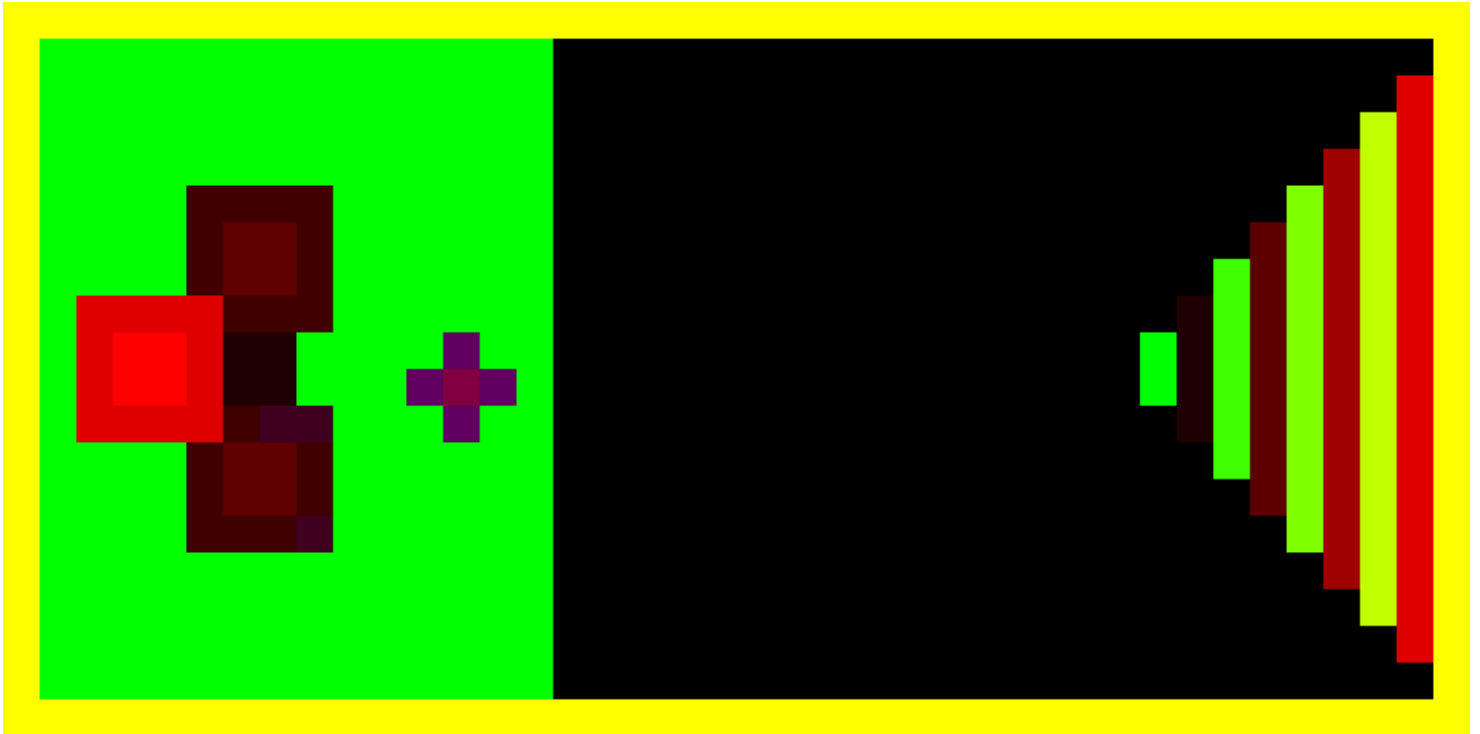


Рис. 2: Карта мира в формате bmp

Карта мира представляет собой продвинутую карту высот/текстур:

- Синий канал - стартовый уровень высоты данного участка.
- Красный канал - конечный уровень высоты данного участка.
- Зеленый канал - номер используемой текстуры участка.

Каждый пиксель занимает примерно 2 на 2 метра реального мира.

Modern C++ API

```

1  std::vector<std::shared_ptr<Texture>> textures;
2  textures.push_back(std::make_shared<Texture>("textures/portal_black.png"));
3  textures.push_back(std::make_shared<Texture>("textures/portal_white.jpg"));
4  ...
5  std::list<std::unique_ptr<OpenGLModel>> objects = Map::load("maps/map01.bmp", textures);
6  ...
7  Shader screenShader;
8  screenShader.attach("shaders/shadows.frag");
9  screenShader.attach("shaders/shadows.vert");
10 screenShader.link();
11 ...
12 for (auto &obj : objects) {
13     obj->draw(screenShader.getUniformLocation("model"));
14 }

```

Листинг 1: Пример использования

Листинги

```
1  #include "model.hpp"
2
3  #include <glad/glad.h>
4  #include <glm/gtc/matrix_transform.hpp>
5  #include <vector>
6  #include <iostream>
7
8  OpenGLModel::OpenGLModel(std::shared_ptr<Texture> texture) {
9      mTexture = texture;
10     glGenVertexArrays(1, &mID);
11 }
12
13 void OpenGLModel::initVBO(std::vector<float> &coordinates) {
14     glBindVertexArray(mID);
15
16     glGenBuffers(1, &mVBO);
17     glBindBuffer(GL_ARRAY_BUFFER, mVBO);
18     glBufferData(GL_ARRAY_BUFFER, coordinates.size() * sizeof(float), coordinates.data(), GL_STATIC_DRAW);
19     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), 0);
20     glEnableVertexAttribArray(0);
21
22     mVertexCount = coordinates.size() / 8 * 3;
23
24     glEnableVertexAttribArray(0);
25     glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(float), (void*)0);
26     glEnableVertexAttribArray(1);
27     glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(float), (void*)(3 * sizeof(float)));
28     glEnableVertexAttribArray(2);
29     glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 8 * sizeof(float), (void*)(6 * sizeof(float)));
30 }
31
32 OpenGLModel::~OpenGLModel() {
33     glDeleteVertexArrays(1, &mID);
34     glDeleteBuffers(1, &mVBO);
35 }
36
37 void OpenGLModel::draw(GLuint modelLocation) {
38     mTexture->activate();
39     glBindVertexArray(mID);
40     glm::mat4 model;
41     model = glm::translate(model, mPosition);
42     model = glm::rotate(model, glm::radians(mRotation.x), glm::vec3(1.0f, 0.0f, 0.0f));
43     model = glm::rotate(model, glm::radians(mRotation.y), glm::vec3(0.0f, 1.0f, 0.0f));
44     model = glm::rotate(model, glm::radians(mRotation.z), glm::vec3(0.0f, 0.0f, 1.0f));
45     glUniformMatrix4fv(modelLocation, 1, GL_FALSE, &model[0][0]);
46     glDrawArrays(GL_TRIANGLES, 0, mVertexCount);
47 }
48
49 void OpenGLModel::move(glm::vec3 position) {
50     mPosition = position;
51 }
52
53 void OpenGLModel::rotate(glm::vec3 rotation) {
54     mRotation = rotation;
55 }
56
57 glm::vec3 OpenGLModel::getPosition() {
58     return mPosition;
59 }
```

Листинг 2: Класс Модели игрового мира

Чтобы добавить новую модель достаточно лишь унаследовать класс OpenGLModel и определить конструктор с координатами:

```
1 Cube::Cube(std::shared_ptr<Texture> texture): OpenGLModel(texture) {
2     std::vector<float> coordinates = {
3         -1.0f, -1.0f, -1.0f,  0.0f,  0.0f, -1.0f,  0.0f,  0.0f,
4     ...
5     };
6     initVBO(coordinates);
7 }
```