ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»

Кафедра: ВТ

Факультет: ИВТ

Лабораторная работа №10

«Использование текстур»

Выполнил: студент

группы ИВТ-42-17

Алексеев Д.А.

Проверил: ст. преподаватель

Галибин С.В.

Чебоксары, 2019 г.

**Цель работы:** Целью лабораторной работы №11 является использование текстур с альфа-каналом для наложения декалей. Декали необходимы для локального изменения окраски поверхности при моделировании мелких, изменяемых деталей, например, следов от взрывов или дырок от пуль. В данной лабораторной работе декали будут спользоваться именно для наложения следов от взрывов на горизонтальную плоскость.

**Задание к лабораторной работе:**

1. Добавить класс Decal, предназначенный для работы с декалями.

2. Реализовать логику вывода декалей в месте взрыва бомбы.

3. Подключить библиотеку OpenAL и реализовать класс AudioManager

для работы со звуковыми эффектами.

4. Реализовать логику проигрывания звуковых эффектов.

**Decal.h**

#pragma once

#include <stdio.h>

#include "GraphicObject.h"

#include "Project1/PhongMaterialWithTexture.h"

#include "Texture.h"

#include "Mesh.h"

#include <gl/GL.h>

#include <gl/GLU.h>

#include "freeglut.h"

#include "glm.hpp"

using namespace std;

class Decal

{

public:

// Инициализация

static void init();

// Конструкторы

Decal();

Decal(glm::ivec2 position);

// Установить позицию

void setPosition(glm::ivec2 position);

// Получить позицию

glm::ivec2 getPosition() const;

// Вывести на экран

void draw();

private:

GraphicObject graphicObject; // Графический объект

glm::ivec2 position; // Позиция

static PhongMaterialWithTexture material;

static Mesh mesh;

static shared\_ptr<Texture> texture;

};

**Decal.cpp**

#include "Decal.h"

PhongMaterialWithTexture Decal::material;

Mesh Decal::mesh;

shared\_ptr<Texture> Decal::texture = make\_shared<Texture>();

// Инициализация

void Decal::init()

{

// Загрузить декаль из файла

mesh.load("../MESHES/Decal.obj");

texture->load("../TEXTURES/decal.png");

material.load("../MATERIALS/decal\_material.txt");

material.setTexture(texture);

}

// Конструктор

Decal::Decal()

{

this->position = glm::ivec2(0, 0);

graphicObject.setMaterial(&material);

graphicObject.setMesh(&mesh);

graphicObject.setPosition(glm::vec3(-10, 0, -10));

}

// Установить позицию

Decal::Decal(glm::ivec2 position)

/\*

Параметры:

position - позиция

\*/

{

this->position = position;

graphicObject.setMaterial(&material);

graphicObject.setMesh(&mesh);

graphicObject.setPosition(glm::vec3(position.x - 10, 0.001, position.y - 10));

}

// Установить координаты

void Decal::setPosition(glm::ivec2 position)

/\*

Параметры:

position - координаты

\*/

{

this->position = position;

}

// Получение координат

glm::ivec2 Decal::getPosition() const

/\*

Возвращаемое значение:

координаты декали

\*/

{

return this->position;

}

// Отобразить на экране

void Decal::draw()

{

glEnable(GL\_POLYGON\_OFFSET\_FILL);

glEnable(GL\_BLEND);

glPolygonOffset(-1, -3);

graphicObject.draw();

glDisable(GL\_BLEND);

glDisable(GL\_POLYGON\_OFFSET\_FILL);

}

**Simulation.cpp**

// Функционирование бомбы

void bombSimulation(float simulationTime)

{

if ( !hero->isMoving() // Установить бомбу

&& GetAsyncKeyState(VK\_SPACE)

&& !bomb->isInGame()

)

{

int x = hero->getPosition().x;

int y = hero->getPosition().y;

passabilityMap[x][y] = 9;

bomb->start(x, y, 2);

}

if (bomb->isInGame()) // Отследить состояние бомбы

{

bomb->simulate(simulationTime);

if (bomb->isExploded()) // Бомба взорвалась

{

audioManager.play(EXPLOSION);

int x = bomb->getPosition().x;

int y = bomb->getPosition().y;

Decal decal(glm::ivec2(x, y));

decals.push\_back(decal);

passabilityMap[x][y] = 0;

for (int i = -1; i <= 1; i++)

{

for (int j = -1; j <= 1; j++)

{

// Уничтожить объекты вокруг бомбы

if (passabilityMap[x + i][y + j] == 1) // Легкий ящик

{

passabilityMap[x + i][y + j] = 0;

gameObjects[x + i][y + j] = nullptr;

}

if (passabilityMap[x + i][y + j] == 8) // Монстр

{

for (int k = 0; k < monsters.size(); k++) // Все монстры

{

if ( monsters[k]->monster != nullptr // Убить монстра

& monsters[k]->position == ivec2(x + i, y + j)

)

{

monsters[k]->monster = nullptr;

passabilityMap[x + i][y + j] = 0;

gameObjects[x + i][y + j] = nullptr;

}

}

}

if (ivec2(x + i, y + j) == hero->getPosition()) // Игрок

{

killHero();

}

}

}

}

}

}

|  |  |  |
| --- | --- | --- |
| **Название модуля и его назначение** | **Кол-во строк** | |
| **\*.h** | **\*.cpp** |
| **Main – основная программа** | **-** | **83** |
| **Data - глобальные данные** | **83** | **145** |
| **Display – вывод объектов** | **13** | **121** |
| **Simulation – игровая логика** | **32** | **444** |
| **GraphicObject – класс для работы с трехмерным графическим объектом** | **67** | **118** |
| **Camera – класс для работы с камерой** | **70** | **148** |
| **Light – класс для работы со светом** | **42** | **25** |
| **Material – класс для работы с источником светом** | **55** | **102** |
| **Mesh – класс для работы с мешем** | **57** | **187** |
| **GameObject – класс для работы с игровым объектом** | **66** | **118** |
| **GameObjectFactory –класс для создания игровых объектов** | **65** | **102** |
| **Portal – класс для работы с порталами** | **35** | **61** |
| **Sprite – класс для работы со спрайтами** | **47** | **66** |
| **Bomb – класс для работы с бомбой** | **62** | **90** |
| **Texture – класс для работы с текстурой** | **38** | **90** |
| **AudioManager – класс для работы со звуками** | **65** | **164** |
|  | **797** | **2064** |