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 №10 является вывод объектов с наложенными на них текстурами, загруженными из внешних файлов графических форматов, изучение различных видов фильтрации текстур, а также реализация системы классов с использованием наследования и полиморфизма для представления различных материалов.

**Задание к лабораторной работе:**

***1. Загрузка текстурных координат.*** Прежде всего, необходимо внести изменения в класс Mesh так, чтобы при загрузке меша из файла также считывались и текстурные координаты. Далее необходимо передать их в видеокарту и убедиться, что даже с дополнительными атрибутами меши по-прежнему выводятся корректно.

***2. Создание класса Texture для работы с текстурой.*** Класс для работы с текстурой имеет метод для загрузки текстуры из внешнего файла. Загрузка из внешнего файла выполняется с помощью библиотеки DevIL, рассмотренной в одной из предыдущих лабораторных работ.

***3. Вывод плоскости с наложенной на неё текстурой.*** Прежде чем переходить к генерации сложной системы классов, представляющих материал, необходимо убедиться, что наложение текстуры на плоскость происходит корректно. Для этого необходимо выбрать текстурный блок, привязать к нему ранее созданную текстуру, разрешить текстурирование в данном текстурном блоке и установить режим наложения текстуры. Если текстурные координаты и текстурный объект были созданы правильно, на экране будет выведена плоскость с наложенной на неё текстурой.

***4. Реализация системы классов для представления материала.***

***5. Внести изменения в классы GraphicObject и GameObjectFactory.*** Класс GraphicObject должен ссылаться на базовый класс материала (класс Material), что даст возможность на этапе выполнения программы указать, какой именно из двух типов материалов используется для вывода именно этого графического объекта. В классе GameObjectFactory необходимо определить для каждого игрового объекта свой собственный материал с учетом описания, заданного в xml-файле.

***6. Реализация монстров.***

***7. Выполнение прочих требований.*** В завершении необходимо реализовать прочие требования, в частности: переключение режимов фильтрации по нажатию на клавишу «2».

**Simulation.cpp**

// Монстры

void monstersSimulation(float simulationTime)

{

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

{

if (monsters[i]->monster != nullptr) // Монстр существует

{

if (!monsters[i]->monster->isMoving()) // Монстр не двигается

{

// Выбрать направление движения монстра

MoveDirection selectedDir = selectMoveDirection(monsters[i]);

// Монстр начал движение в указанном направлении

monsterMove(monsters[i], selectedDir);

}

else // Монстр уже двигается

{

monsters[i]->monster->simulate(simulationTime);

}

}

}

}

// Выбрать направление движения мостра

MoveDirection selectMoveDirection(shared\_ptr<Monster> monster)

{

// Определить свободное нарпавление вперед

vector<MoveDirection> freeDir;

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

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

if (passabilityMap[x][y - 1] == 0 // Верх

&& monster->returnDirection != MoveDirection::up

)

{

freeDir.push\_back(MoveDirection::up);

}

if (passabilityMap[x + 1][y] == 0 // Право

&& monster->returnDirection != MoveDirection::right

)

{

freeDir.push\_back(MoveDirection::right);

}

if (passabilityMap[x][y + 1] == 0 // Низ

&& monster->returnDirection != MoveDirection::down

)

{

freeDir.push\_back(MoveDirection::down);

}

if (passabilityMap[x - 1][y] == 0 // Лево

&& monster->returnDirection != MoveDirection::left

)

{

freeDir.push\_back(MoveDirection::left);

}

MoveDirection selectedDir;

if (!freeDir.empty()) // Есть свободное направление вперед

{

// Выбрать направление вперед

uniform\_int\_distribution<int> distribution(0, freeDir.size() - 1);

selectedDir = freeDir[distribution(generator)];

}

else // Можно пойти только назад

{ selectedDir = monster->returnDirection;}

return selectedDir;

}

// Монстр начал движение в указанном направлении

void monsterMove(shared\_ptr<Monster> monster, MoveDirection selectedDir)

{

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

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

monster->monster->move(selectedDir, SPEED / 5);

passabilityMap[x][y] = 0;

switch (selectedDir) // Выбранное направление

{

case MoveDirection::up: // Верх

monster->returnDirection = MoveDirection::down;

if (hero->getPosition() == ivec2(x, y - 1))

{

killHero();

}

monster->position += ivec2(0, -1);

passabilityMap[x][y - 1] = 8;

gameObjects[x][y - 1] = gameObjects[x][y];

gameObjects[x][y] = nullptr;

break;

case MoveDirection::right: // Право

monster->returnDirection = MoveDirection::left;

if (hero->getPosition() == ivec2(x + 1, y))

{

killHero();

}

monster->position += ivec2(1, 0);

passabilityMap[x + 1][y] = 8;

gameObjects[x + 1][y] = gameObjects[x][y];

gameObjects[x][y] = nullptr;

break;

case MoveDirection::down: // Низ

monster->returnDirection = MoveDirection::up;

if (hero->getPosition() == ivec2(x, y + 1))

{

killHero();

}

monster->position += ivec2(0, 1);

passabilityMap[x][y + 1] = 8;

gameObjects[x][y + 1] = gameObjects[x][y];

gameObjects[x][y] = nullptr;

break;

case MoveDirection::left: // Лево

monster->returnDirection = MoveDirection::right;

if (hero->getPosition() == ivec2(x - 1, y))

{

killHero();

}

monster->position += ivec2(-1, 0);

passabilityMap[x - 1][y] = 8;

gameObjects[x - 1][y] = gameObjects[x][y];

gameObjects[x][y] = nullptr;

break;

}

}

|  |  |  |
| --- | --- | --- |
| **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** |
|  | **732** | **1900** |