Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедравычислительной техники

АГС.

Лабораторная работа №4.

«Реализация освещения по методу Фонга»

Выполнил: студент гр. ИВТ-41-15

Лазарев Дмитрий

Проверил:

Ст. преп. Галибин С.В.

Чебоксары – 2018

**Задание к лабораторной работе.**

В лабораторной работе №4 необходимо:

1. Реализовать классы CLight, CMaterial и CGraphicObject для представления источника света, материала и графического объекта соответственно. Классы должны, как минимум, иметь поля и методы, описанные выше. При необходимости разрешается добавлять дополнительные поля и методы, если это не противоречит архитектуре программы.

2. Реализовать класс CScene вышеописанной структуры. Класс должен иметь метод CreateGraphicObject, который позволяет создать новый объект по имени используемой модели. Сама модель состоит из меша, текстуры и прочих параметров, описанных в файле

«Resources.xml». Файл и сами ресурсы необходимо взять из прилагаемого примера.

3. Реализовать вывод нескольких (не менее 5) объектов, часть из которых является одинаковыми.

CGraphicObject.cpp

#include "CGraphicObject.h"

void CGraphicObject::setMesh(int meshID)

{

this->meshID = meshID;

}

void CGraphicObject::setMaterial(CMaterial & material)

{

this->material = material;

}

void CGraphicObject::setPosition(vec3 position)

{

modelMatrix= mat4(

vec4(1, 0, 0, 0), // 1‐ый столбец: направление оси ox

vec4(0, 1, 0, 0), // 2‐ой столбец: направление оси oy

vec4(0, 0, 1, 0), // 3‐ий столбец: направление оси oz

vec4(position, 1));

}

void CGraphicObject::setRotation(float degree)

{

mat4 Rotate = mat4

(vec4(cos(degree), 0.0, sin(degree), 0),

vec4(0.0, 1.0, 0.0, 0),

vec4(-sin(degree), 0.0, cos(degree), 0),

vec4(0, 0, 0, 1));

this->modelMatrix = modelMatrix \* Rotate;

}

int CGraphicObject::getMesh(void)

{

return meshID;

}

CMaterial & CGraphicObject::getMaterial(void)

{

// TODO: вставьте здесь оператор return

return material;

}

mat4 CGraphicObject::getModelMatrix(void)

{

return modelMatrix;

}

CScene.cpp

#include "CScene.h"

void CScene::init(void)

{

Shader.LoadVertexShader((char\*)"SHADER\\Example.vsh");

Shader.LoadFragmentShader((char\*)"SHADER\\Example.fsh");

Shader.Link();

Light = CreateLight();

// временная локальная переменная

CGraphicObject TempGraphicObject;

// первый объект

TempGraphicObject = CreateGraphicObject("house\_1\_bl");

TempGraphicObject.setPosition(vec3(0, 0, 0));

TempGraphicObject.setRotation(0);

GraphicObjects.push\_back(TempGraphicObject);

// второй объект

TempGraphicObject = CreateGraphicObject("light");

TempGraphicObject.setPosition(vec3(-6.5, -0.55, 3));

TempGraphicObject.setRotation(-90);

GraphicObjects.push\_back(TempGraphicObject);

// третий объект

TempGraphicObject = CreateGraphicObject("light");

TempGraphicObject.setPosition(vec3(+6.5, -0.55, 3));

TempGraphicObject.setRotation(-90);

GraphicObjects.push\_back(TempGraphicObject);

// четвертый объект

TempGraphicObject = CreateGraphicObject("ambul");

TempGraphicObject.setPosition(vec3(+2.5, -1.7, 5.2));

TempGraphicObject.setRotation(0);

GraphicObjects.push\_back(TempGraphicObject);

}

void CScene::simulate(float sec)

{

bool Forward = GetAsyncKeyState(VK\_UP);

bool Back = GetAsyncKeyState(VK\_DOWN);

bool Left = GetAsyncKeyState(VK\_LEFT);

bool Right = GetAsyncKeyState(VK\_RIGHT);

float dForward = (int(Forward) - int(Back))\*sec;

float dRigth = (int(Right) - int(Left))\*sec;

GetCursorPos(&newPossition);

float dHorizAngle = oldPossition.x - newPossition.x;

float dVertAngle = oldPossition.y - newPossition.y;

oldPossition = newPossition;

Camera.MoveOXZ(dForward, dRigth);

Camera.Rotate(dHorizAngle, dVertAngle);

}

CCamera \* CScene::getCamera(void)

{

return &Camera;

}

void CScene::draw()

{

Shader.Activate();

// получаем матрицу проекции

for (int i = 0; i < GraphicObjects.size(); i++)

{

mat4 ProjectionMatrix = Camera.GetProjectionMatrix();

Shader.SetUniform("ProjectionMatrix", ProjectionMatrix);

mat4 ViewMatrix = Camera.GetViewMatrix();

Shader.SetUniform("lAmbient", Light.getAmbient());

Shader.SetUniform("lDiffuse", Light.getDiffuse());

Shader.SetUniform("lSpecular", Light.getSpecular());

vec4 vec = ViewMatrix \* Light.getPosition();

Shader.SetUniform("lPosition", vec);

mat4 ModelViewMatrix1 = ViewMatrix \* GraphicObjects[i].getModelMatrix();

Shader.SetUniform("ModelViewMatrix", ModelViewMatrix1);

Shader.SetUniform("mAmbient", GraphicObjects[i].getMaterial().getAmbient());

Shader.SetUniform("mDiffuse", GraphicObjects[i].getMaterial().getDiffuse());

Shader.SetUniform("mSpecular", GraphicObjects[i].getMaterial().getSpecular());

CMesh\* mesh = CResourceManager::Instance().GetMesh(GraphicObjects[i].getMesh());

if (mesh != nullptr) mesh->Render();

}

}

CGraphicObject CScene::CreateGraphicObject(std::string name)

{

CGraphicObject tempGraphicObject;

CMaterial tempMaterial;

float temp[3];

pugi::xml\_parse\_result result = resources\_description.load\_file("Resources.xml");

pugi::xml\_node resources = resources\_description.child("Resources");

pugi::xml\_node models = resources.child("Models");

pugi::xml\_node model = models.find\_child\_by\_attribute("id", name.c\_str());

pugi::xml\_attribute mesh = model.child("Mesh").attribute("path");

char mesh\_name[128];

sprintf\_s(mesh\_name, "%s", mesh.value());

tempGraphicObject.setMesh(CResourceManager::Instance().LoadMesh(mesh\_name));

pugi::xml\_node material = model.child("Material");

pugi::xml\_node phong = material.child("PhongParameters");

pugi::xml\_attribute diffuse = phong.attribute("diffuse");

sscanf\_s(diffuse.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec4 vec = vec4(temp[0], temp[1], temp[2], 0);

tempMaterial.setDiffuse(vec);

pugi::xml\_attribute ambient = phong.attribute("ambient");

sscanf\_s(ambient.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec = vec4(temp[0], temp[1], temp[2], 0);

tempMaterial.setAmbient(vec);

pugi::xml\_attribute specular = phong.attribute("specular");

sscanf\_s(specular.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec = vec4(temp[0], temp[1], temp[2], 0);

tempMaterial.setSpecular(vec);

pugi::xml\_attribute shininess = phong.attribute("shininess");

sscanf\_s(shininess.value(), "%f", &temp[0]);

tempMaterial.setShininess(temp[0]);

tempGraphicObject.setMaterial(tempMaterial);

return tempGraphicObject;

}

CLight CScene::CreateLight()

{

CLight tempLight;

float temp[3];

pugi::xml\_parse\_result result = resources\_description.load\_file("Resources.xml");

pugi::xml\_node resources = resources\_description.child("Resources");

pugi::xml\_node light = resources.child("Light");

pugi::xml\_attribute direction = light.child("Direction").attribute("vector");

sscanf\_s(direction.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec4 vec = vec4(temp[0], temp[1], temp[2], 0);

tempLight.setPosition(vec);

pugi::xml\_node phong = light.child("PhongParameters");

pugi::xml\_attribute diffuse = phong.attribute("diffuse");

sscanf\_s(diffuse.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec = vec4(temp[0], temp[1], temp[2], 0);

tempLight.setDiffuse(vec);

pugi::xml\_attribute ambient = phong.attribute("ambient");

sscanf\_s(ambient.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec = vec4(temp[0], temp[1], temp[2], 0);

tempLight.setAmbient(vec);

pugi::xml\_attribute specular = phong.attribute("specular");

sscanf\_s(specular.value(), "%f %f %f", &temp[0], &temp[1], &temp[2]);

vec = vec4(temp[0], temp[1], temp[2], 0);

tempLight.setSpecular(vec);

return tempLight;

}