**Vs.shader**

#version 400

layout(location = 0) in vec4 vertex\_position;

layout(location = 1) in vec4 vertex\_color;

uniform mat4 model\_mat\_shader;

uniform mat4 projection\_mat\_shader;

out vec4 fragment\_color;

void main(void)

{

gl\_Position = projection\_mat\_shader \* model\_mat\_shader \* vertex\_position;

fragment\_color = vertex\_color;

}

**Fs.shader**

#version 400

in vec4 fragment\_color;

out vec4 out\_Color;

void main(void)

{

out\_Color = fragment\_color;

}

**Main**

#include <iostream>

#include <fstream>

#include <string>

#include <GL/glew.h>

#include <GL/freeglut.h>

#include <Vector\_Matrix.h>

using namespace std;

typedef struct

{

float XYZW[4];

float RGBA[4];

} Vertex;

// v6----- v5

// /| /|

// v1------v0|

// | | | |

// | |v7---|-|v4

// |/ |/

// v2------v3

Vertex Vertices[] =

{

// v0-v1-v2 (front)

{ { 0.5f, 0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

{ { -0.5f, 0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

{ { -0.5f, -0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

// v2-v3-v0

{ { -0.5f, -0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

{ { 0.5f, -0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

{ { 0.5f, 0.5f, 0.5f, 1.0f },{ 1.0f, 0.0f, 0.0f, 1.0f } },

// v0-v3-v4 (right)

{ { 0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

{ { 0.5f, -0.5f, 0.5f,1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

{ { 0.5f, -0.5f, -0.5f,1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

// v4-v5-v0

{ { 0.5f, -0.5f, -0.5f, 1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

{ { 0.5f, 0.5f, -0.5f, 1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

{ { 0.5f, 0.5f, 0.5f,1.0f },{ 0.0f, 1.0f, 0.0f, 1.0f } },

// v0-v5-v6 (top)

{ { 0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { 0.5f, 0.5f, -0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, 0.5f, -0.5f,1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

// v6-v1-v0

{ { -0.5f, 0.5f, -0.5f,1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { 0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

// v1-v6-v7 (left)

{ { -0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, 0.5f, -0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, -0.5f, -0.5f,1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

// v7-v2-v1

{ { -0.5f, -0.5f, -0.5f,1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, -0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

{ { -0.5f, 0.5f, 0.5f, 1.0f },{ 0.0f, 0.0f, 1.0f, 1.0f } },

// v7-v4-v3 (bottom)

{ { -0.5f, -0.5f, -0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

{ { 0.5f, -0.5f, -0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

{ { 0.5f, -0.5f, 0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

// v3-v2-v7

{ { 0.5f, -0.5f, 0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

{ { -0.5f, -0.5f, 0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

{ { -0.5f, -0.5f, -0.5f, 1.0f },{ 0.1f, 0.2f, 0.3f, 1.0f } },

// v4-v7-v6 (back)

{ { 0.5f, -0.5f, -0.5f, 1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } },

{ { -0.5f, -0.5f, -0.5f, 1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } },

{ { -0.5f, 0.5f, -0.5f, 1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } },

// v6-v5-v4

{ { -0.5f, 0.5f, -0.5f,1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } },

{ { 0.5f, 0.5f, -0.5f,1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } },

{ { 0.5f, -0.5f, -0.5f,1.0f },{ 0.4f, 0.5f, 0.6f, 1.0f } }

};

const size\_t BufferSize = sizeof(Vertices);

const size\_t VertexSize = sizeof(Vertices[0]);

const size\_t RgbOffset = sizeof(Vertices[0].XYZW);

int CurrentWidth = 700,

CurrentHeight = 700;

GLuint

VaoId,

VboId,

VertexShaderId,

FragmentShaderId,

ProgramId;

mat4 model\_fan, model\_lamp;

int model\_mat\_location,

projection\_mat\_location;

bool battat = false;

GLfloat height = 0, xoay = 0, gap = 0;

enum { Base = 0, LowerArm = 1, UpperArm = 2, NumAngles = 3 };

int Axis = Base;

float Theta[NumAngles] = { 0.0 };

void block(mat4 mtg, mat4 mt) {

//vi tri mat nhin

vec3 eye(0.0, 0.2, 2.0);

//nhin vao vi tri goc toa do (0,0,0)

vec3 at(0.0, 0.0, 0.0);

// huong mat nhin theo chieu duong truc y

vec3 up(0.0, 1.0, 0.0);

//goi ham lookat va truyen vao cac tham so eye,at,up

mat4 m = lookat(eye, at, up);

mat4 model\_base = m \* mtg \* mt;

glUniformMatrix4fv(model\_mat\_location, 1, GL\_FALSE, model\_base.m);

//thiet lap chieu phoi canh

float l = -0.5, r = 0.5, b = -0.5, t = 0.5, znear = 0.5, zfar = 3.0;

mat4 p = frustum(l, r, b, t, znear, zfar);

glUniformMatrix4fv(projection\_mat\_location, 1, GL\_FALSE, p.m);

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

// ---------------------------------------------------------------------------

void block(mat4 mt, bool demo) {

//vi tri mat nhin

vec3 eye(0.0, 0.2, 2.0);

//nhin vao vi tri goc toa do (0,0,0)

vec3 at(0.0, 0.0, 0.0);

// huong mat nhin theo chieu duong truc y

vec3 up(0.0, 1.0, 0.0);

//goi ham lookat va truyen vao cac tham so eye,at,up

mat4 m = lookat(eye, at, up);

mat4 model\_base = m \* model\_lamp \* mt;

glUniformMatrix4fv(model\_mat\_location, 1, GL\_FALSE, model\_base.m);

//thiet lap chieu phoi canh

float l = -0.5, r = 0.5, b = -0.5, t = 0.5, znear = 0.5, zfar = 3.0;

mat4 p = frustum(l, r, b, t, znear, zfar);

glUniformMatrix4fv(projection\_mat\_location, 1, GL\_FALSE, p.m);

glDrawArrays(GL\_TRIANGLES, 0, 10);

}

string ReadShaderSourceFile(string fileName) {

fstream reader(fileName.c\_str());

string line;

string code = "";

while (getline(reader, line)) {

code += line + "\n";

}

reader.close();

return code;

}

// ---------------------------------------------------------------------------

void CreatVaoVbo()

{

glGenVertexArrays(1, &VaoId);

glBindVertexArray(VaoId);

glGenBuffers(1, &VboId);

glBindBuffer(GL\_ARRAY\_BUFFER, VboId);

glBufferData(GL\_ARRAY\_BUFFER, BufferSize, Vertices, GL\_STATIC\_DRAW);

glVertexAttribPointer(0, 4, GL\_FLOAT, GL\_FALSE, VertexSize, 0);

glVertexAttribPointer(1, 4, GL\_FLOAT, GL\_FALSE, VertexSize, (GLvoid\*)RgbOffset);

glEnableVertexAttribArray(0);

glEnableVertexAttribArray(1);

}

// ---------------------------------------------------------------------------

void CreatShaders()

{

string vertexSrc = ReadShaderSourceFile("./vs.shader");

string fragmentSrc = ReadShaderSourceFile("./fs.shader");

const GLchar\* VertexShader = vertexSrc.c\_str();

const GLchar\* FragmentShader = fragmentSrc.c\_str();

VertexShaderId = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(VertexShaderId, 1, &VertexShader, NULL);

glCompileShader(VertexShaderId);

FragmentShaderId = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(FragmentShaderId, 1, &FragmentShader, NULL);

glCompileShader(FragmentShaderId);

ProgramId = glCreateProgram();

glAttachShader(ProgramId, VertexShaderId);

glAttachShader(ProgramId, FragmentShaderId);

glLinkProgram(ProgramId);

glUseProgram(ProgramId);

}

// ---------------------------------------------------------------------------

void CloseFunc()

{

glUseProgram(0);

glDetachShader(ProgramId, VertexShaderId);

glDetachShader(ProgramId, FragmentShaderId);

glDeleteShader(FragmentShaderId);

glDeleteShader(VertexShaderId);

glDeleteProgram(ProgramId);

glDisableVertexAttribArray(1);

glDisableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, 0);

glDeleteBuffers(1, &VboId);

glBindVertexArray(0);

glDeleteVertexArrays(1, &VaoId);

}

// ------------------------------------------

mat4 model\_room;

void room() {

mat4 san = identity\_mat4() \* translate(vec3(0, -0.5, 0)) \* rotate\_z(180) \* scale(vec3(2.5, 0.01, 1.6));

block(model\_room, san);

mat4 tran = translate(vec3(0, 1, 0)) \* scale(vec3(2.5, 0.01, 1.6));

block(model\_room, tran);

mat4 lung = translate(vec3(0, 0.25, -0.8)) \* rotate\_y(180) \* scale(vec3(2.5, 1.5, 0.01));

block(model\_room, lung);

mat4 trai = translate(vec3(1.25, 0.25, 0)) \* scale(vec3(0.01, 1.5, 1.6));

block(model\_room, trai);

mat4 phai = translate(vec3(-1.25, 0.25, 0)) \* rotate\_y(180) \* scale(vec3(0.01, 1.5, 1.6));

block(model\_room, phai);

}

mat4 model\_table;

void table() {

mat4 foot = scale(vec3(0.01, 0.3, 0.3));

block(model\_table, translate(vec3(-0.0, -0.3, -0.4)) \*foot);

block(model\_table, translate(vec3(-0.6, -0.3, -0.4)) \* rotate\_z(180) \* foot);

mat4 matban = scale(vec3(0.6, 0.01, 0.3));

block(model\_table, translate(vec3(-0.3, -0.15, -0.4)) \* rotate\_z(180) \* matban);

}

GLfloat xoay2 = 0, rota = 0, quaydeuquaydeu = 0;

mat4 model;

// ---------------------------------------------------------------------------

void DisplayFunc(void)

{

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

model\_room = rotate\_y(Theta[Base]);

model\_table = rotate\_y(Theta[Base]);

model\_mat\_location = glGetUniformLocation(ProgramId, "model\_mat\_shader");

projection\_mat\_location = glGetUniformLocation(ProgramId, "projection\_mat\_shader");

room();

table();

glutSwapBuffers();

}

// ---------------------------------------------------------------------------

void ReshapeFunc(int Width, int Height)

{

CurrentWidth = Width;

CurrentHeight = Height;

}

bool xoaycanh = false, xoaythan = false, xoayxuoi = false;

GLfloat tmp = 0;

// ---------------------------------------------------------------------------

void IdleFunc(void)

{

glutPostRedisplay();

}

// ---------------------------------------------------------------------------

void KeyboardFunc(unsigned char key, int x, int y)

{

switch (key) {

case 27:

exit(EXIT\_SUCCESS); break;

}

}

// ------------------------------------------

void MouseFunc(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

Theta[Axis] += 5.0;

if (Theta[Axis] > 360.0) { Theta[Axis] -= 360.0; }

}

if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) {

Theta[Axis] -= 5.0;

if (Theta[Axis] < 0.0) { Theta[Axis] += 360.0; }

}

glutPostRedisplay();

}

// ---------------------------------------------------------------------------

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitContextVersion(4, 0);

glutInitContextFlags(GLUT\_FORWARD\_COMPATIBLE);

glutInitContextProfile(GLUT\_CORE\_PROFILE);

glutSetOption(GLUT\_ACTION\_ON\_WINDOW\_CLOSE, GLUT\_ACTION\_GLUTMAINLOOP\_RETURNS);

glutInitWindowSize(CurrentWidth, CurrentHeight);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutCreateWindow("Phong tro");

glewExperimental = GL\_TRUE;

glewInit();

CreatVaoVbo();

CreatShaders();

glutDisplayFunc(DisplayFunc);

glutReshapeFunc(ReshapeFunc);

glutIdleFunc(IdleFunc);

glutKeyboardFunc(KeyboardFunc);

glutMouseFunc(MouseFunc);

glutCloseFunc(CloseFunc);

glutMainLoop();

return 0;

}