**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительной математики и программирования»

**Лабораторная работа № 6**

по курсу «Компьютерная графика»

Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1

Студент: Сорокин Д.М.

Группа: 80-304Б

Преподаватель: Филлипов Г.С.

Оценка:

Москва, 2017

Задание:

Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта

Вариант:

15. Анимация. Сдвиг вдоль нормали пропорционально времени

Код:

Функция, отвечающая за анимацию:

void TimerFunction(int value){

glTranslatef(0.02, 0.02, 0.02);

if(isWithAnimation) glutTimerFunc(50,TimerFunction,1);

glutPostRedisplay();

}

Main.cpp

#define por10 1

#define okr10 2

#define por100 3

#define okr100 4

using namespace std;

//AUX\_RGBImageRec \*image;

Engine \*engine;

GLfloat thisLightAmbient[3] = {3.0, 0.0, 0.0};

GLfloat thisLightDiffuse[3] = {1.0, 0.0, 0.0};

GLfloat thisLightSpecular[3] = {0.2, 0.0, 0.0};

GLfloat thisMaterialAmbient[4] = {1.0, 1.0, 1.0, 1.0};

GLfloat thisMaterialDiffuse[4] = {1.0, 1.0, 1.0, 1.0};

GLfloat thisMaterialSpecular[4] = {1.0, 1.0, 1.0, 1.0};

GLfloat thisLightPosition[4] = {5.0, 5.0, 0.0, 1.0};

int fotoNum = 1;

bool isCapOn = false;

bool isWithTex = true;

bool isWithAnimation = false;

double mx = 0, my = 0, px = 0, py = 0;

int NumberOfCircles = 20, NumberOfParabols = 20;

double actualAngle = 0;

std::wstring s2ws(const std::string& s)

{

int len;

int slength = (int)s.length() + 1;

len = MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, 0, 0);

wchar\_t\* buf = new wchar\_t[len];

MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, buf, len);

std::wstring r(buf);

delete[] buf;

return r;

}

GLvoid display(GLvoid)

{

engine->lightAmbient[0] = thisLightAmbient[0]; engine->lightAmbient[1] = thisLightAmbient[1]; engine->lightAmbient[2] = thisLightAmbient[2];

engine->lightDiffuse[0] = thisLightDiffuse[0]; engine->lightDiffuse[1] = thisLightDiffuse[1]; engine->lightDiffuse[2] = thisLightDiffuse[2];

engine->lightSpecular[0] = thisLightSpecular[0]; engine->lightSpecular[1] = thisLightSpecular[1]; engine->lightSpecular[2] = thisLightSpecular[2];

engine->materialAmbient[0] = thisMaterialAmbient[0]; engine->materialAmbient[1] = thisMaterialAmbient[1]; engine->materialAmbient[2] = thisMaterialAmbient[2]; engine->materialAmbient[3] = thisMaterialAmbient[3];

engine->materialDiffuse[0] = thisMaterialDiffuse[0]; engine->materialDiffuse[1] = thisMaterialDiffuse[1]; engine->materialDiffuse[2] = thisMaterialDiffuse[2]; engine->materialDiffuse[3] = thisMaterialDiffuse[3];

engine->materialSpecular[0] = thisMaterialSpecular[0]; engine->materialSpecular[1] = thisMaterialSpecular[1]; engine->materialSpecular[2] = thisMaterialSpecular[2]; engine->materialSpecular[3] = thisMaterialSpecular[3];

engine->lightPosition[0] = thisLightPosition[0]; engine->lightPosition[1] = thisLightPosition[1]; engine->lightPosition[2] = thisLightPosition[2]; engine->lightPosition[3] = thisLightPosition[3];

AUX\_RGBImageRec \*image;

std::stringstream ss;

ss<<fotoNum;

std::string str;

ss>>str;

string photoNum = "photo";

photoNum += str;

photoNum += ".bmp";

std::wstring stemp = s2ws(photoNum);

LPCWSTR result = stemp.c\_str();

image = auxDIBImageLoad(result);

engine->Draw(NumberOfCircles, NumberOfParabols, isCapOn, isWithTex, image);

glutSwapBuffers();

}

GLvoid reshape(GLsizei width, GLsizei height)

{

engine->Resize(width, height);

}

void MyKeyboard(int key, int a, int b)

{

switch(key)

{

case GLUT\_KEY\_UP: glRotatef(-5, 1.0, 0.0, 0.0); break;

case GLUT\_KEY\_LEFT: glRotatef(-5, 0.0, 1.0, 0.0); break;

case GLUT\_KEY\_DOWN: glRotatef(5, 1.0, 0.0, 0.0); break;

case GLUT\_KEY\_RIGHT: glRotatef(5, 0.0, 1.0, 0.0); break;

case GLUT\_KEY\_PAGE\_UP: glRotatef(5, 0.0, 0.0, 1.0); break;

case GLUT\_KEY\_PAGE\_DOWN: glRotatef(-5, 0.0, 0.0, 1.0); break;

case GLUT\_KEY\_F1: glScalef( 0.99f, 0.99f, 0.99f ); break; // MessageBox( NULL, L"Could Not Unregister Class.", L"SHUTDOWN ERROR", MB\_OK | MB\_ICONINFORMATION);

case GLUT\_KEY\_F2: glScalef( 1.01f, 1.01f, 1.01f ); break;

case GLUT\_KEY\_F3: thisLightPosition[0] -= 5; break;

case GLUT\_KEY\_F4: thisLightPosition[0] += 5; break;

case GLUT\_KEY\_F5: thisLightPosition[1] -= 5; break;

case GLUT\_KEY\_F6: thisLightPosition[1] += 5; break;

case GLUT\_KEY\_F7: thisLightPosition[2] -= 5; break;

case GLUT\_KEY\_F8: thisLightPosition[2] += 5; break;

}

glutPostRedisplay();

}

void TimerFunction(int value){

glTranslatef(0.02, 0.02, 0.02);

if(isWithAnimation) glutTimerFunc(50,TimerFunction,1);

glutPostRedisplay();

}

void processMenuEventsPlus(int option) {

switch (option) {

case por10 :

NumberOfParabols += 12; break;

case okr10 :

NumberOfCircles += 12; break;

case por100 :

NumberOfParabols += 100; break;

case okr100 :

NumberOfCircles += 100; break;

}

glutPostRedisplay();

}

void processMenuEventsMinus(int option) {

switch (option) {

case por10 :

if(NumberOfParabols > 12) NumberOfParabols -= 12; else NumberOfParabols = 2; break;

case okr10 :

if(NumberOfCircles > 12) NumberOfCircles -= 12; else NumberOfCircles = 2; break;

case por100 :

if(NumberOfParabols > 100) NumberOfParabols -= 100; else NumberOfParabols = 0; break;

case okr100 :

if(NumberOfCircles > 100) NumberOfCircles -= 100; else NumberOfCircles = 1; break;

}

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(600, 600);

glutInitWindowPosition(0, 0);

glutCreateWindow("Lab4, 5, 6");

engine = new Engine();

engine->Init();

glutSpecialFunc(MyKeyboard);

createGLUTMenus();

glutTimerFunc(5,TimerFunction,1);

//image = auxDIBImageLoad(L"photo.bmp");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

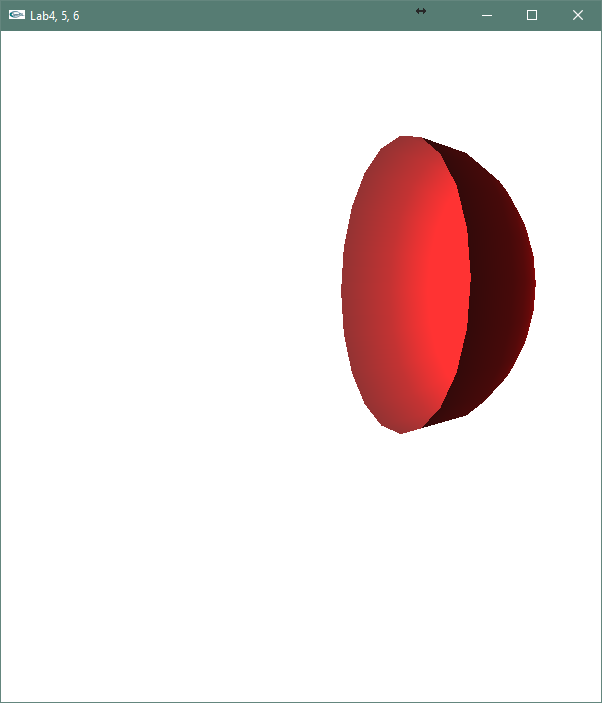
//glutMotionFunc(MouseFunc);

glutMainLoop();

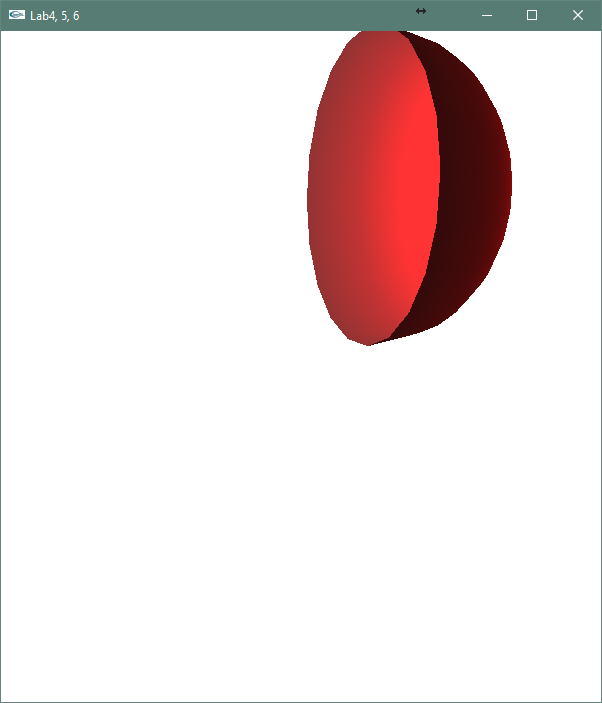
return 0;

}

Примеры работы программы:



После анимации:



Выводы:

В данной лабораторной я научился создавать анимацию с помощью технологии OpenGl. Все оказалось не так сложно, если разобраться! И в общем-то решается все практически добавлением пары функций, вот такой OpenGl удобный в этом плане.