**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительной математики и программирования»

**Лабораторные работы № 4-5**

по курсу «Компьютерная графика»

Тема: Ознакомление с технологией OpenGL.

Студент: Сорокин Д.М.

Группа: 80-304Б

Преподаватель: Филлипов Г.С.

Оценка:

Москва, 2017

Задание:

Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Код:

#define \_USE\_MATH\_DEFINES

#include <math.h>

#include <vector>

#include "core.h"

using namespace std;

GLvoid Engine::SetProjectionMatrix(GLvoid){

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glFrustum(-1, 1, -1, 1, 5, 30);

}

GLvoid Engine::SetModelviewMatrix(){

glMatrixMode(GL\_MODELVIEW); glLoadIdentity();

glTranslatef(0.0, 0.0, -24.0);

}

GLvoid Engine::Resize(GLsizei width, GLsizei height){

if (height == 0)

{

height = 1;

}

glViewport(0, 0, width, height); // Устанавливается область просмотра

Height = height;

Width = width;

SetProjectionMatrix();

SetModelviewMatrix();

}

GLvoid Engine::Init(GLvoid){

glEnable(GL\_AUTO\_NORMAL);

glEnable(GL\_LIGHTING);

glLightModelf(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE);

glEnable(GL\_NORMALIZE);

//glEnable(GL\_TEXTURE\_GEN\_S);

//glEnable(GL\_TEXTURE\_GEN\_T);

glClearColor(255.0f, 255.0f, 255.0f, 0.0f); // Устанавливается черный фон

glClearDepth(1.0f); glEnable(GL\_DEPTH\_TEST); glDepthFunc(GL\_LEQUAL);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_LINE\_SMOOTH);

glHint(GL\_LINE\_SMOOTH\_HINT, GL\_NICEST);

glEnable(GL\_BLEND); glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

}

void calcParaboloid(GLfloat size, int NumberOfCircles, int NumberOfParabols, bool isCapOn, bool isWithTex)

{

glBegin(GL\_TRIANGLES);

//glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_EYE\_LINEAR);

//glTexGeni(GL\_T, GL\_TEXTURE\_GEN\_MODE, GL\_EYE\_LINEAR);

vector<double> tempPoint;

double t0 = 0, tn =5 , dt = (tn - t0) / NumberOfCircles;

vector<vector<double>> points;

for (int i = 0; i < NumberOfCircles; ++i)

{

double x = t0 + i \* dt;

double z = sqrt(9 - x \* x);

vector<double> point;

point.push\_back(x);

point.push\_back(0);

point.push\_back(-z);

points.push\_back(point);

}

double u0 = 0, un = 2 \* M\_PI, du = (un - u0) / NumberOfParabols;

bool temp = true;

for (int j = 0; j < NumberOfParabols; ++j)

{

vector<vector<double>> curPoints;

vector<vector<double>> nextPoints;

double angle = u0 + j \* du;

double nextAngle = u0 + (j + 1) \* du;

double matrixR[3][3];

double matrixRR[3][3];

for(int n = 0; n < 3; ++n)

for(int m = 0; m < 3; ++m)

{

matrixR[n][m] = 0;

matrixRR[n][m] = 0;

}

matrixR[0][0] = matrixR[1][1] = cos(angle);

matrixR[1][0] = -1 \*(matrixR[0][1] = sin(angle));

matrixR[2][2] = 1;

matrixRR[0][0] = matrixRR[1][1] = cos(nextAngle);

matrixRR[1][0] = -1 \* (matrixRR[0][1] = sin(nextAngle));

matrixRR[2][2] = 1;

for (int i = 0; i < NumberOfCircles; ++i)

{

vector<double> t1;

t1.push\_back(0);

t1.push\_back(0);

t1.push\_back(0);

vector<double> tp;

tp.push\_back(points[i][0]);

tp.push\_back(points[i][1]);

tp.push\_back(points[i][2]);

for(int m = 0; m < 3; m++)

for(int n = 0; n < 3; n++)

t1[m] += matrixR[m][n] \* tp[n];

vector<double> t2;

t2.push\_back(0);

t2.push\_back(0);

t2.push\_back(0);

for(int m = 0; m < 3; m++)

for(int n = 0; n < 3; n++)

t2[m] += matrixRR[m][n] \* tp[n];

curPoints.push\_back(t1);

if (j == NumberOfParabols - 1)

{

nextPoints.push\_back(points[i]);

}

else

{

nextPoints.push\_back(t2);

}

}

int nn = curPoints.size();

for (int i = 0; i < nn - 1; ++i)

{

if (i != nn - 2) glColor3f(1.0f, 1.0f, 1.0f);

else glColor3f(1.0f, 0.0f, 0.0f);

glVertex3f(curPoints[i][0], curPoints[i][1], curPoints[i][2]);

//glNormal3f(1, 1, 1);

glVertex3f(nextPoints[i + 1][0], nextPoints[i + 1][1], nextPoints[i + 1][2]);

//glNormal3f(1, 1, 1);

glVertex3f(curPoints[i + 1][0], curPoints[i + 1][1], curPoints[i + 1][2]);

//glNormal3f(1, 1, 1);

if (i != nn - 2) glColor3f(0.1f, 0.0f, 0.0f);

glVertex3f(curPoints[i][0], curPoints[i][1], curPoints[i][2]);

glVertex3f(nextPoints[i][0], nextPoints[i][1], nextPoints[i][2]);

glVertex3f(nextPoints[i + 1][0], nextPoints[i + 1][1], nextPoints[i + 1][2]);

}

double center[3] = { 0, 0, 0 };

if(temp) { glColor3f(1.0f, 1.0f, 1.0f); temp = false; }

else { glColor3f(0.1f, 0.0f, 0.0f); temp = true; }

if(isCapOn)

{

double numberOfParabolsDouble = NumberOfParabols;

double jDouble = j;

if(isWithTex)

{

if(jDouble <= numberOfParabolsDouble/4)

glTexCoord2d( (4/numberOfParabolsDouble)\*jDouble ,0);

if(numberOfParabolsDouble/4 < jDouble && jDouble <= numberOfParabolsDouble/2)

glTexCoord2d(1,(4/numberOfParabolsDouble)\*(jDouble-numberOfParabolsDouble/4));

if(numberOfParabolsDouble/2 < jDouble && jDouble <= 3\*numberOfParabolsDouble/4)

glTexCoord2d(1-(4/numberOfParabolsDouble)\*(jDouble-numberOfParabolsDouble/2),1);

if(3\*numberOfParabolsDouble/4 < jDouble && jDouble < numberOfParabolsDouble)

glTexCoord2d(0, 1-(4/numberOfParabolsDouble)\*(jDouble-3\*numberOfParabolsDouble/4));

}

glVertex3f(curPoints[nn - 1][0], curPoints[nn - 1][1], curPoints[nn - 1][2]);

if(isWithTex)

{

if(jDouble < numberOfParabolsDouble/4)

glTexCoord2d( (4/numberOfParabolsDouble)\*(jDouble+1),0);

if(numberOfParabolsDouble/4 <= jDouble && jDouble < numberOfParabolsDouble/2)

glTexCoord2d(1,(4/numberOfParabolsDouble)\*(jDouble-numberOfParabolsDouble/4 + 1));

if(numberOfParabolsDouble/2 <= jDouble && jDouble < 3\*numberOfParabolsDouble/4)

glTexCoord2d(1-(4/numberOfParabolsDouble)\*(jDouble-numberOfParabolsDouble/2 + 1),1);

if(3\*numberOfParabolsDouble/4 <= jDouble && jDouble < numberOfParabolsDouble)

glTexCoord2d(0, 1-(4/numberOfParabolsDouble)\*(jDouble-3\*numberOfParabolsDouble/4 + 1));

}

glVertex3f(nextPoints[nn - 1][0], nextPoints[nn - 1][1], nextPoints[nn - 1][2]);

glTexCoord2d(0.5, 0.5); glVertex3f(center[0], center[1], center[2]);

//if(j > 0) {

//glVertex3f(tempPoint[0], tempPoint[1], tempPoint[2]);

//glVertex3f(curPoints[nn - 1][0], curPoints[nn - 1][1], curPoints[nn - 1][2]);

//glVertex3f(center[0], center[1], center[2]);

//}

}

}

//glDisable(GL\_TEXTURE\_GEN\_S);

//glDisable(GL\_TEXTURE\_GEN\_T);

//glDisable(GL\_TEXTURE\_2D);

glEnd();

}

GLvoid Engine::Draw(int NumberOfCircles, int NumberOfParabols, bool isCapOn, bool isWithTex, AUX\_RGBImageRec \*image)

{

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

if(isWithTex)

{

glEnable(GL\_TEXTURE\_2D);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP);

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3,

image->sizeX,

image->sizeY,

0, GL\_RGB, GL\_UNSIGNED\_BYTE,

image->data);

}

else glDisable(GL\_TEXTURE\_2D);

// свойства материала

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, materialDiffuse);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, materialAmbient);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, materialSpecular);

glEnable(GL\_LIGHT1);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, lightDiffuse);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, lightAmbient);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, lightSpecular);

glLightfv(GL\_LIGHT1, GL\_POSITION, lightPosition);

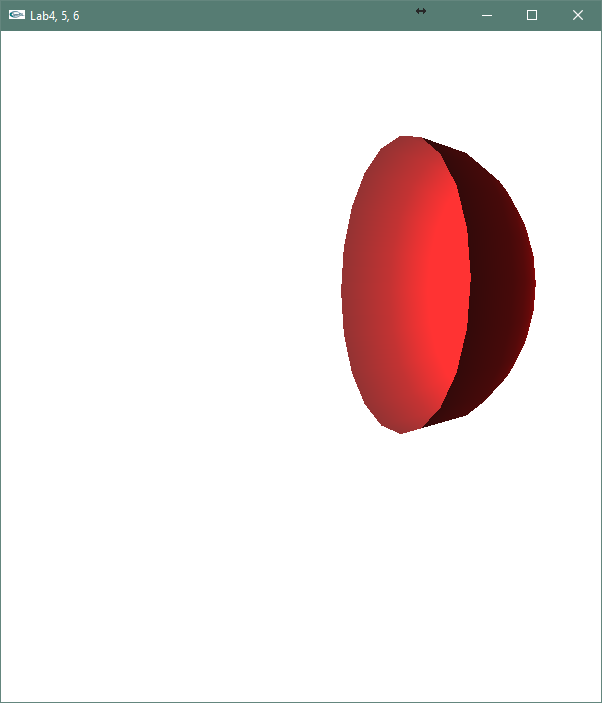
glLightf(GL\_LIGHT1, GL\_CONSTANT\_ATTENUATION, 0.0);

glLightf(GL\_LIGHT1, GL\_LINEAR\_ATTENUATION, 0.2);

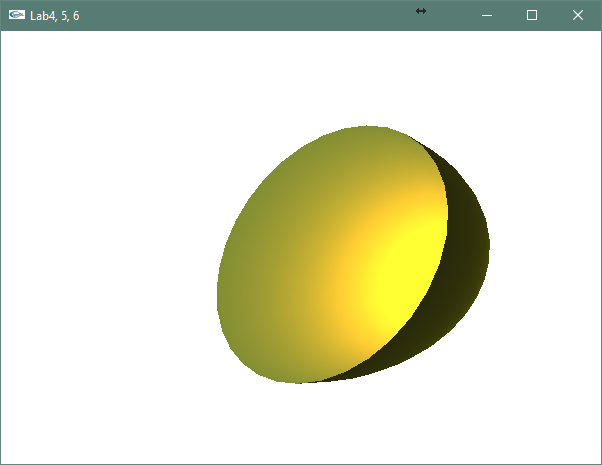
glLightf(GL\_LIGHT1, GL\_QUADRATIC\_ATTENUATION, 0.4);

glPushMatrix(); calcParaboloid(1.0f, NumberOfCircles, NumberOfParabols, isCapOn, isWithTex);

glPopMatrix(); }

Примеры работы программы:

После некоторых изменений (свет, положение, масштаб)



Выводы:

В данной лабораторной я познакомился с технологией OpenGL. Технология эта весьма полезна. Было очень интересно разобраться в ней, особенно после собственноручного написания некоторых методов, для реализации визуализации.