Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 16**

**по дисциплине**

**«Программирование компьютерной графики»**

**Тема: «Криволинейные поверхности»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Жигалов И.Е.

Владимир, 2021

ЦЕЛЬ РАБОТЫ

Изучение методов формирования моделей объемных объектов, ограниченных криволинейными поверхностями, при составлении графических программ.

Уравнения объектов с криволинейными границами можно выразить в параметрической или непараметрической форме. В число различных объектов, часто полезных в графических приложениях, входят поверхности второго порядка, полиномиальные и показательные функции, сплайновые поверхности. Данные входные описания объектов обычно представляются в мозаичной форме, что позволяет аппроксимировать поверхности многоугольными сетками.

ЗАДАНИЕ

1. Ознакомиться по методическим указаниям и литературе с теоретическим материалом.

2. Разработать программу построения следующих графических изображений:

- поверхностей второго порядка на основе их уравнений: тора и суперэллипсоида; параметры поверхностей (коэффициенты соответствующих уравнений) подобрать самостоятельно;

- тех же поверхностей (тора и суперэллипсоида) с использованием функций GLUT;

- сплайновых поверхностей с использованием функций OpenGL: поверхности Безье, би-сплайновой поверхности (с обрезкой поверхностей); параметры поверхностей подобрать самостоятельно;

ХОД РАБОТЫ

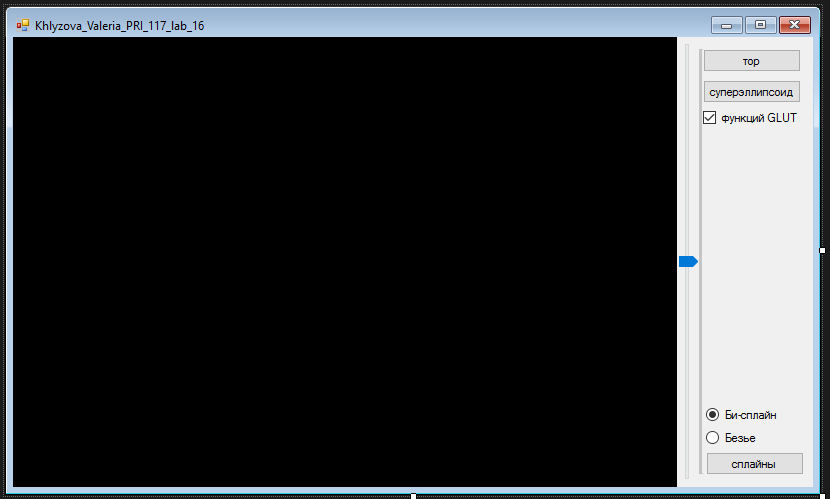


Рисунок 1 - Конструктор формы.

using System;

using System.Windows.Forms;

using Tao.FreeGlut;

using Tao.OpenGl;

using static Tao.OpenGl.Glu;

namespace Khlyzova\_Valeria\_PRI\_117\_lab\_16

{

public partial class Form1 : Form

{

unsafe GLUnurbs theNurb;

string figure = "tor";

string mode = "figure";

static float[,,] ctrlpoints = {

{{-1.5f, -1.5f, 4.0f}, {-0.5f, -1.5f, 2.0f},

{0.5f, -1.5f, -1.0f}, {1.5f, -1.5f, 2.0f}},

{{-1.5f, -0.5f, 1.0f}, {-0.5f, -0.5f, 3.0f},

{0.5f, -0.5f, 0.0f}, {1.5f, -0.5f, -1.0f}},

{{-1.5f, 0.5f, 4.0f}, {-0.5f, 0.5f, 0.0f},

{0.5f, 0.5f, 3.0f}, {1.5f, 0.5f, 4.0f}},

{{-1.5f, 1.5f, -2.0f}, {-0.5f, 1.5f, -2.0f},

{0.5f, 1.5f, 0.0f}, {1.5f, 1.5f, -1.0f}}

};

public Form1()

{

InitializeComponent();

simpleOpenGlControl.InitializeContexts();

}

private void Form1\_Load(object sender, EventArgs e)

{

Glut.glutInit();

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

Gl.glClearColor(0, 0, 0, 1);

Gl.glViewport(0, 0, simpleOpenGlControl.Width, simpleOpenGlControl.Height);

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

Glu.gluPerspective(45, (float)simpleOpenGlControl.Width / (float)simpleOpenGlControl.Height, 0.1, 200);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

Gl.glEnable(Gl.GL\_DEPTH\_TEST);

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glEnable(Gl.GL\_LIGHT0);

RenderTimer.Start();

}

private void RenderTimer\_Tick(object sender, EventArgs e)

{

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

Gl.glClearColor(255, 255, 255, 1);

Gl.glLoadIdentity();

Gl.glTranslated(0, 0, -17);

Gl.glRotated(trackBar1.Value, 0, 1, 0);

if (mode == "figure") Draw();

else DrawSpline();

Gl.glPopMatrix();

Gl.glFlush();

simpleOpenGlControl.Invalidate();

}

private void DrawSpline()

{

if (radioButton1.Checked)

{

// поверхности Безье

Gl.glMap2f(Gl.GL\_MAP2\_VERTEX\_3, 0, 1, 3, 4, 0, 1, 12, 4, ref ctrlpoints[0, 0, 0]);

Gl.glEnable(Gl.GL\_MAP2\_VERTEX\_3);

int i, j;

Gl.glPushMatrix();

for (j = 0; j <= 20; j++)

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

for (i = 0; i <= 50; i++)

Gl.glEvalCoord2f((float)i / 50.0f, (float)j / 20.0f);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_STRIP);

for (i = 0; i <= 50; i++)

Gl.glEvalCoord2f((float)j / 20.0f, (float)i / 50.0f);

Gl.glEnd();

}

Gl.glPopMatrix();

Gl.glFlush();

}

// би-сплайновая поверхность (с обрезкой поверхностей)

if (radioButton2.Checked)

{

float[] mat\_diffuse = { 0.6f, 0.6f, 0.6f, 1.0f };

float[] mat\_specular = { 0.9f, 0.9f, 0.9f, 1.0f };

float[] mat\_shininess = { 128.0f };

theNurb = gluNewNurbsRenderer();

float[] knots = { 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f, 1.0f };

float[,] edgePt = { { 0.0f, 0.0f}, { 1.0f, 0.0f}, { 1.0f, 1.0f}, { 0.0f, 1.0f}, { 0.0f, 0.0f}};

float[,] curvePt = { { 0.25f, 0.5f}, { 0.25f, 0.75f}, { 0.75f, 0.75f}, { 0.75f, 0.5f} };

float[] curveKnots = {0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f, 1.0f };

float[,] pwlPt = { { 0.75f, 0.5f }, { 0.5f, 0.25f}, { 0.25f, 0.5f } };

gluBeginSurface(theNurb);

gluNurbsSurface(theNurb, 8, knots, 8, knots, 12, 3, ctrlpoints, 4, 4, Gl.GL\_MAP2\_VERTEX\_3);

gluBeginTrim(theNurb);

gluPwlCurve(theNurb, 5, edgePt, 2, GLU\_MAP1\_TRIM\_2);

gluEndTrim(theNurb);

gluBeginTrim(theNurb);

gluNurbsCurve(theNurb, 8, curveKnots, 2,curvePt, 4, GLU\_MAP1\_TRIM\_2);

gluPwlCurve(theNurb, 3, pwlPt, 2, GLU\_MAP1\_TRIM\_2);

gluEndTrim(theNurb);

gluEndSurface(theNurb);

}

}

private void Draw()

{

if (checkBox1.Checked)

{

// отрисовка поверхностей с использованием функций GLUT

switch (figure)

{

case "tor": Glut.glutWireTorus(0.5, 4, 20, 50); break;

case "superellipsoid": Glut.glutWireSphere(5, 32, 32); break;

}

}

else

{

int numc = 20, numt = 50;

double Angle;

// отрисовка поверхностей на основе их уравнений

switch (figure)

{

case "tor":

Angle = 2 \* Math.PI;

Gl.glBegin(Gl.GL\_LINE\_STRIP);

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

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

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

{

for (int k = 1; k >= 0; k--)

{

double s = (i + k) % numc + 0.5;

double t = j % numt;

double x = (1 + 0.1 \* Math.Cos(s \* Angle / numc)) \* Math.Cos(t \* Angle / numt);

double y = (1 + 0.1 \* Math.Cos(s \* Angle / numc)) \* Math.Sin(t \* Angle / numt);

double z = 1 \* Math.Sin(s \* Angle / numc);

Gl.glVertex3d(4 \* x, 4 \* y, 0.5 \* z);

}

}

Gl.glEnd();

}

Gl.glEnd();

break;

case "superellipsoid":

double x1, y1, z1;

Angle = Math.PI;

for (double a = 0; a < Angle; a += Angle / numc)

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

for (double b = 0; b < 2.01 \* Angle; b += Angle / numc)

{

x1 = 5 \* Math.Cos(b) \* Math.Sin(a);

y1 = 5 \* Math.Sin(b) \* Math.Sin(a);

z1 = 5 \* Math.Cos(a);

Gl.glVertex3f((float)x1, (float)y1, (float)z1);

x1 = 5 \* Math.Cos(b) \* Math.Sin(a + Angle / numc);

y1 = 5 \* Math.Sin(b) \* Math.Sin(a + Angle / numc);

z1 = 5 \* Math.Cos(a + Angle / numc);

Gl.glVertex3f((float)x1, (float)y1, (float)z1);

}

Gl.glEnd();

}

break;

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

mode = "figure";

figure = "superellipsoid";

}

private void button1\_Click(object sender, EventArgs e)

{

mode = "figure";

figure = "tor";

}

private void button3\_Click(object sender, EventArgs e)

{

mode = "spline";

}

}

}

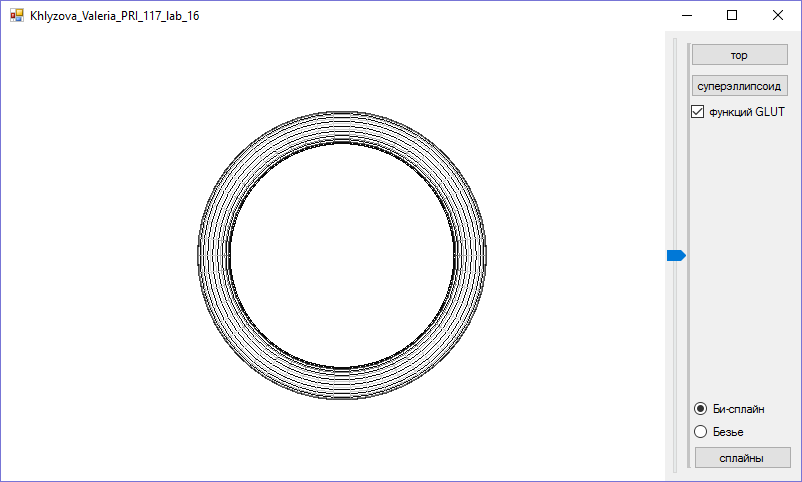


Рисунок 2 – Поверхность, выполненная с использованием функций GLUT.

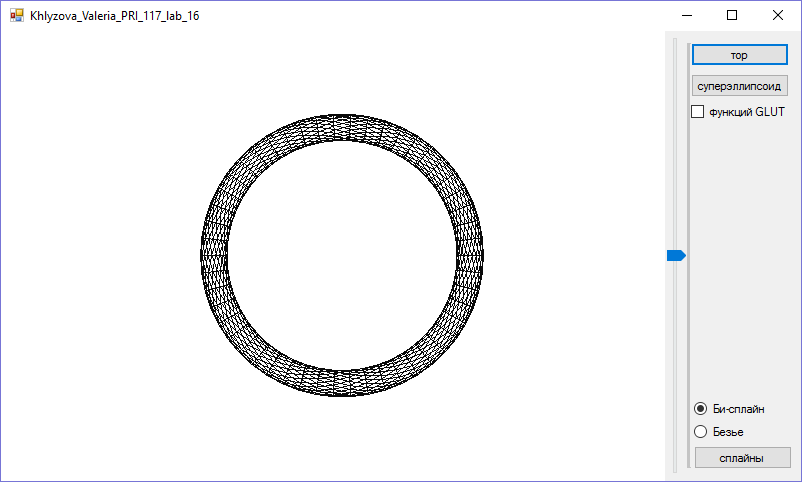


Рисунок 3 - Поверхность второго порядка на основе уравнений.

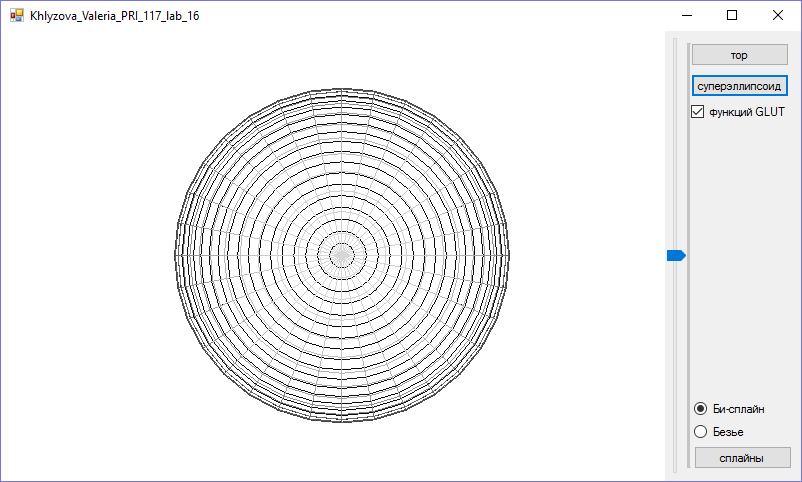


Рисунок 4 – Поверхность, выполненная с использованием функций GLUT.

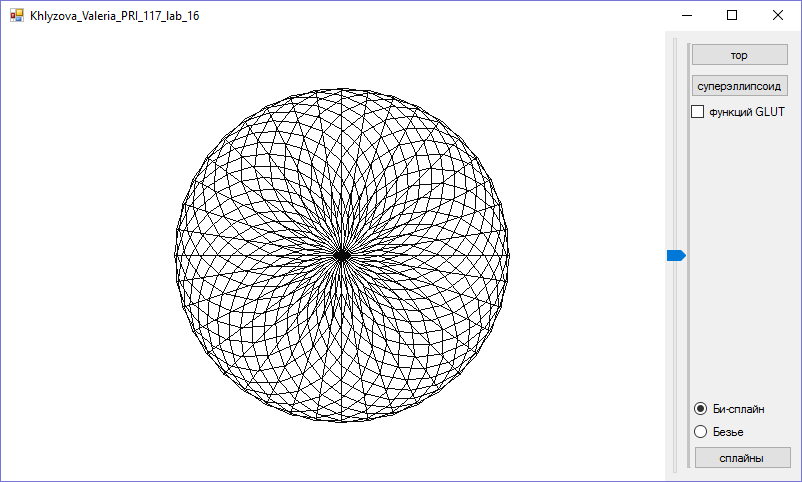


Рисунок 5 - Поверхность второго порядка на основе уравнений.

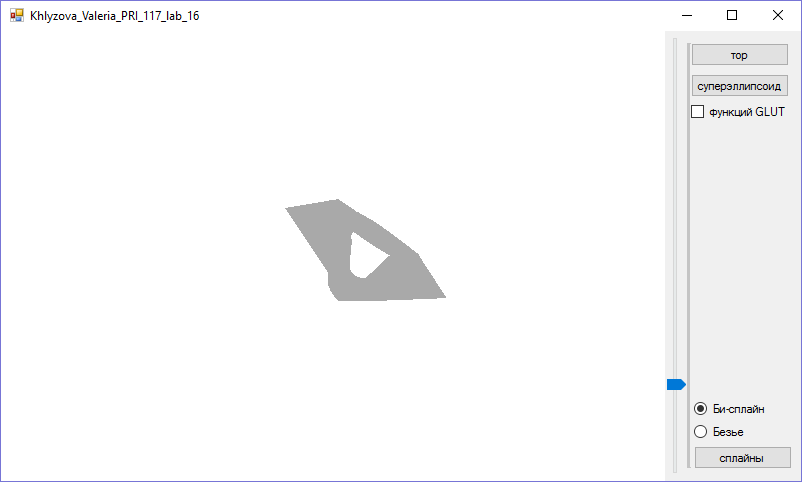


Рисунок 6 - Би-сплайновая поверхность (с обрезкой поверхностей).

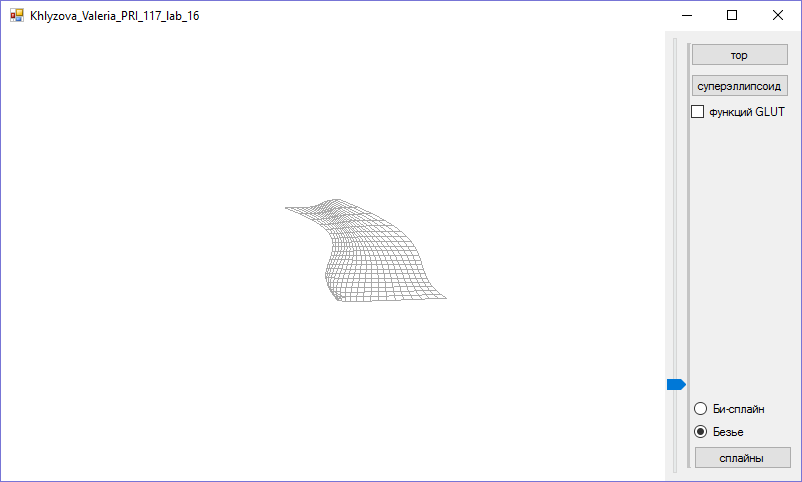


Рисунок 7 - Поверхность Безье.

ВЫВОД

В ходе выполнения лабораторной работы были изучены методы формирования моделей объемных объектов, ограниченных криволинейными поверхностями, при составлении графических программ.