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 № 8**

**по дисциплине**

**«Программирование компьютерной графики»**

**Тема: «Сплайны»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Жигалов И.Е.

Владимир, 2020

ЦЕЛЬ РАБОТЫ

Ознакомление с алгоритмическими основами применения сплайнов в компьютерной графике, освоение методов формирования изображений на основе сплайнов.

ЗАДАНИЕ

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

2. Выполнить действия, приведенные в разделе 8.2. При разработке программы имя проекта, создаваемого в MS Visual Studio, должно содержать фамилию студента и группу (например, Ivanov\_Ivan\_ISG\_105\_lab\_1).

3. Выполнить реализацию сплайна в соответствии с вариантом задания. Сплайн по варианту должен строиться близким к контуру заданного изображения и выводиться по дополнительной кнопке на панели.

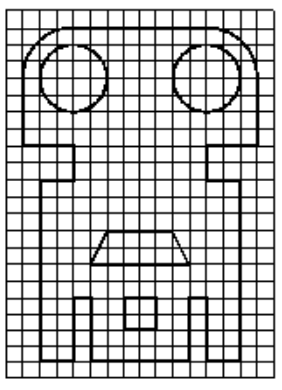


Рисунок 1. Вариант 15.

ХОД РАБОТЫ

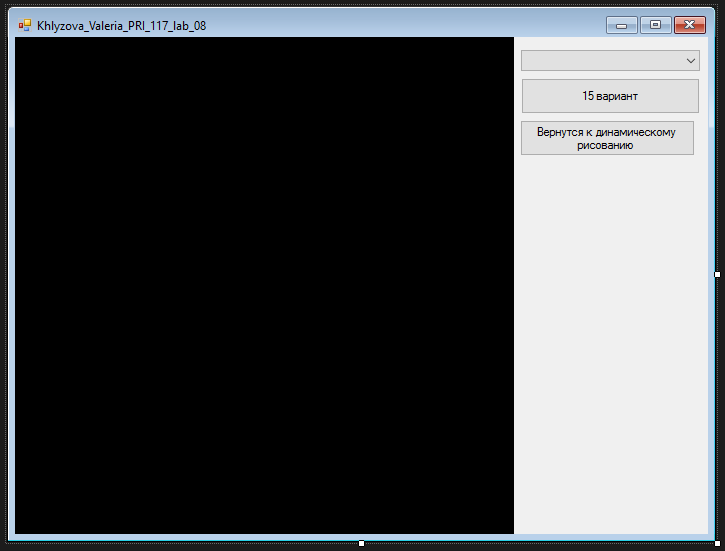


Рисунок 2. Конструктор формы.

Обработчик событий формы

using System;

using System.Windows.Forms;

using Tao.FreeGlut;

using Tao.OpenGl;

namespace Khlyzova\_Valeria\_PRI\_117\_lab\_08

{

public partial class Form1 : Form

{

private float[,] DrawingArray = new float[64, 2];

private int count\_points = 0;

double ScreenW, ScreenH;

private float devX;

private float devY;

float lineX, lineY;

float Mcoord\_X = 0, Mcoord\_Y = 0;

int captured = -1;

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

{

Glut.glutInit();

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

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

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

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

if ((float)AnT.Width <= (float)AnT.Height)

{

ScreenW = 500.0;

ScreenH = 500.0 \* (float)AnT.Height / (float)AnT.Width;

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

}

else

{

ScreenW = 500.0 \* (float)AnT.Width / (float)AnT.Height;

ScreenH = 500.0;

Glu.gluOrtho2D(0.0, 500.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 500.0);

}

devX = (float)ScreenW / (float)AnT.Width;

devY = (float)ScreenH / (float)AnT.Height;

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

RenderTimer.Start();

comboBox1.SelectedIndex = 0;

}

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

private void AnT\_MouseClick(object sender, MouseEventArgs e)

{

if (comboBox1.SelectedIndex == 0)

{

Mcoord\_X = e.X;

Mcoord\_Y = e.Y;

lineX = devX \* e.X;

lineY = (float)(ScreenH - devY \* e.Y);

DrawingArray[count\_points, 0] = lineX;

DrawingArray[count\_points, 1] = lineY;

count\_points++;

}

}

private void AnT\_MouseMove(object sender, MouseEventArgs e)

{

if (comboBox1.SelectedIndex == 0)

{

Mcoord\_X = e.X;

Mcoord\_Y = e.Y;

lineX = devX \* e.X;

lineY = (float)(ScreenH - devY \* e.Y);

DrawingArray[count\_points, 0] = lineX;

DrawingArray[count\_points, 1] = lineY;

}

else

{

Mcoord\_X = e.X;

Mcoord\_Y = e.Y;

float \_lastX = lineX;

float \_lastY = lineY;

lineX = devX \* e.X;

lineY = (float)(ScreenH - devY \* e.Y);

if (captured != -1)

{

DrawingArray[captured, 0] -= \_lastX - lineX;

DrawingArray[captured, 1] -= \_lastY - lineY;

}

}

}

private void AnT\_MouseDown(object sender, MouseEventArgs e)

{

if (comboBox1.SelectedIndex == 1)

{

Mcoord\_X = e.X;

Mcoord\_Y = e.Y;

lineX = devX \* e.X;

lineY = (float)(ScreenH - devY \* e.Y);

for (int ax = 0; ax < count\_points; ax++)

{

if (lineX < DrawingArray[ax, 0] + 5 && lineX > DrawingArray[ax, 0] - 5 && lineY < DrawingArray[ax, 1] + 5 && lineY > DrawingArray[ax, 1] - 5)

{

captured = ax;

break;

}

}

}

}

private void AnT\_MouseUp(object sender, MouseEventArgs e)

{

captured = -1;

}

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

{

RenderTimer.Stop();

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

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 255);

Gl.glLineWidth((float)1.1);

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

Gl.glVertex2d(230, 120);

Gl.glVertex2d(230, 160);

Gl.glVertex2d(270, 160);

Gl.glVertex2d(270, 120);

Gl.glEnd();

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

Gl.glVertex2d(190, 200);

Gl.glVertex2d(210, 240);

Gl.glVertex2d(290, 240);

Gl.glVertex2d(310, 200);

Gl.glEnd();

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

Gl.glVertex2d(170, 480);

float[,] PointArray = new float[6, 2];

PointArray[0, 0] = 170; PointArray[0, 1] = 480;

PointArray[1, 0] = 160; PointArray[1, 1] = 477;

PointArray[2, 0] = 145; PointArray[2, 1] = 470;

PointArray[3, 0] = 130; PointArray[3, 1] = 460;

PointArray[4, 0] = 120; PointArray[4, 1] = 447;

PointArray[5, 0] = 110; PointArray[5, 1] = 420;

RoundOff(PointArray);

Gl.glVertex2d(110, 420);

Gl.glVertex2d(110, 340);

Gl.glVertex2d(170, 340);

Gl.glVertex2d(170, 300);

Gl.glVertex2d(170, 300);

Gl.glVertex2d(130, 300);

Gl.glVertex2d(130, 80);

Gl.glVertex2d(170, 80);

Gl.glVertex2d(170, 160);

Gl.glVertex2d(190, 160);

Gl.glVertex2d(190, 80);

Gl.glVertex2d(310, 80);

Gl.glVertex2d(310, 160);

Gl.glVertex2d(330, 160);

Gl.glVertex2d(330, 80);

Gl.glVertex2d(370, 80);

Gl.glVertex2d(370, 300);

Gl.glVertex2d(330, 300);

Gl.glVertex2d(330, 340);

Gl.glVertex2d(390, 340);

Gl.glVertex2d(390, 420);

PointArray = new float[6, 2];

PointArray[0, 0] = 390; PointArray[0, 1] = 420;

PointArray[1, 0] = 380; PointArray[1, 1] = 447;

PointArray[2, 0] = 370; PointArray[2, 1] = 460;

PointArray[3, 0] = 350; PointArray[3, 1] = 470;

PointArray[4, 0] = 340; PointArray[4, 1] = 477;

PointArray[5, 0] = 330; PointArray[5, 1] = 480;

RoundOff(PointArray);

Gl.glVertex2d(330, 480);

Gl.glEnd();

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

float[,] CircleArray = new float[7, 2];

CircleArray[0, 0] = 170; CircleArray[0, 1] = 460;

CircleArray[1, 0] = 130; CircleArray[1, 1] = 420;

CircleArray[2, 0] = 170; CircleArray[2, 1] = 380;

CircleArray[3, 0] = 210; CircleArray[3, 1] = 420;

CircleArray[4, 0] = 170; CircleArray[4, 1] = 460;

CircleArray[5, 0] = 130; CircleArray[5, 1] = 420;

CircleArray[6, 0] = 170; CircleArray[6, 1] = 380;

RoundOff(CircleArray);

Gl.glEnd();

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

CircleArray = new float[7, 2];

CircleArray[0, 0] = 330; CircleArray[0, 1] = 460;

CircleArray[1, 0] = 290; CircleArray[1, 1] = 420;

CircleArray[2, 0] = 330; CircleArray[2, 1] = 380;

CircleArray[3, 0] = 370; CircleArray[3, 1] = 420;

CircleArray[4, 0] = 330; CircleArray[4, 1] = 460;

CircleArray[5, 0] = 290; CircleArray[5, 1] = 420;

CircleArray[6, 0] = 330; CircleArray[6, 1] = 380;

RoundOff(CircleArray);

Gl.glEnd();

AnT.Invalidate();

}

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

{

RenderTimer.Start();

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

DrawingArray = new float[64, 2];

}

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

{

Draw();

}

private void PrintText2D(float x, float y, string text)

{

Gl.glRasterPos2f(x, y);

foreach (char char\_for\_draw in text)

{

Glut.glutBitmapCharacter(Glut.GLUT\_BITMAP\_8\_BY\_13, char\_for\_draw);

}

}

private void Draw()

{

int N = 30;

double X, Y;

int eps = 4, i, j, n = count\_points + 1, first;

double xA, xB, xC, xD, yA, yB, yC, yD, t;

double a0, a1, a2, a3, b0, b1, b2, b3;

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

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

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 0);

Gl.glPushMatrix();

Gl.glPointSize(5.0f);

Gl.glBegin(Gl.GL\_POINTS);

Gl.glVertex2d(0, 0);

Gl.glEnd();

Gl.glPointSize(1.0f);

PrintText2D(devX \* Mcoord\_X + 0.2f, (float)ScreenH - devY \* Mcoord\_Y + 0.4f, "[ x: " + (devX \* Mcoord\_X).ToString() + " ; y: " + ((float)ScreenH - devY \* Mcoord\_Y).ToString() + "]");

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

{

X = DrawingArray[i, 0];

Y = DrawingArray[i, 1];

if (i == captured)

{

Gl.glLineWidth(3.0f);

}

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

Gl.glVertex2d(X - eps, Y - eps);

Gl.glVertex2d(X + eps, Y - eps);

Gl.glVertex2d(X + eps, Y + eps);

Gl.glVertex2d(X - eps, Y + eps);

Gl.glEnd();

if (i == captured)

{

Gl.glLineWidth(1.0f);

}

}

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

{

X = DrawingArray[i, 0];

Y = DrawingArray[i, 1];

PrintText2D((float)(X - 20), (float)(Y - 20), "P " + i.ToString() + ": " + X.ToString() + ", " + Y.ToString());

}

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

for (i = 1; i < n - 1; i++)

{

first = 1;

xA = DrawingArray[i - 1, 0];

xB = DrawingArray[i, 0];

xC = DrawingArray[i + 1, 0];

xD = DrawingArray[i + 2, 0];

yA = DrawingArray[i - 1, 1];

yB = DrawingArray[i, 1];

yC = DrawingArray[i + 1, 1];

yD = DrawingArray[i + 2, 1];

a3 = (-xA + 3 \* (xB - xC) + xD) / 6.0;

a2 = (xA - 2 \* xB + xC) / 2.0;

a1 = (xC - xA) / 2.0;

a0 = (xA + 4 \* xB + xC) / 6.0;

b3 = (-yA + 3 \* (yB - yC) + yD) / 6.0;

b2 = (yA - 2 \* yB + yC) / 2.0;

b1 = (yC - yA) / 2.0;

b0 = (yA + 4 \* yB + yC) / 6.0;

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

{

t = (double)j / (double)N;

X = (((a3 \* t + a2) \* t + a1) \* t + a0);

Y = (((b3 \* t + b2) \* t + b1) \* t + b0);

if (first == 1)

{

first = 0;

Gl.glVertex2d(X, Y);

}

else

Gl.glVertex2d(X, Y);

}

}

Gl.glEnd();

Gl.glFlush();

AnT.Invalidate();

}

private void RoundOff(float[,] PointArray)

{

int i, j, N = 4, end=PointArray.Length/2-2;

double xA, xB, xC, xD, yA, yB, yC, yD, t;

double a0, a1, a2, a3, b0, b1, b2, b3;

double X, Y;

for (i = 1; i < end; i++)

{

xA = PointArray[i - 1, 0];

xB = PointArray[i, 0];

xC = PointArray[i + 1, 0];

xD = PointArray[i + 2, 0];

yA = PointArray[i - 1, 1];

yB = PointArray[i, 1];

yC = PointArray[i + 1, 1];

yD = PointArray[i + 2, 1];

a3 = (-xA + 3 \* (xB - xC) + xD) / 6.0;

a2 = (xA - 2 \* xB + xC) / 2.0;

a1 = (xC - xA) / 2.0;

a0 = (xA + 4 \* xB + xC) / 6.0;

b3 = (-yA + 3 \* (yB - yC) + yD) / 6.0;

b2 = (yA - 2 \* yB + yC) / 2.0;

b1 = (yC - yA) / 2.0;

b0 = (yA + 4 \* yB + yC) / 6.0;

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

{

t = (double)j / (double)N;

X = (((a3 \* t + a2) \* t + a1) \* t + a0);

Y = (((b3 \* t + b2) \* t + b1) \* t + b0);

Gl.glVertex2d(X, Y);

}

}

}

}

}

Пример работы программы

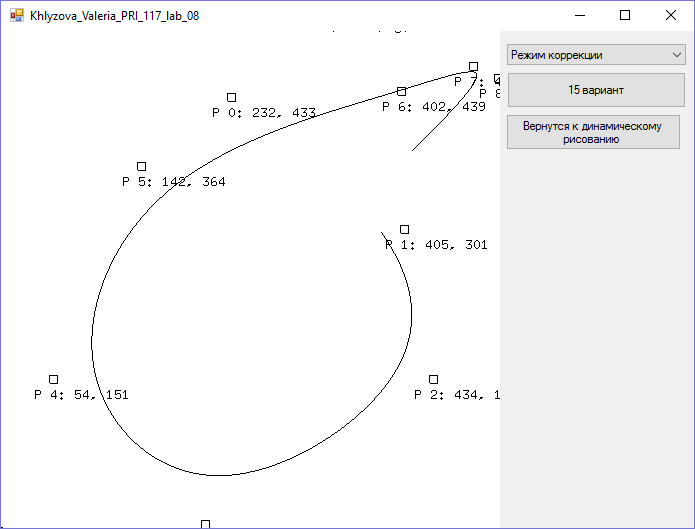


Рисунок 3. Рисование сплайна по точкам

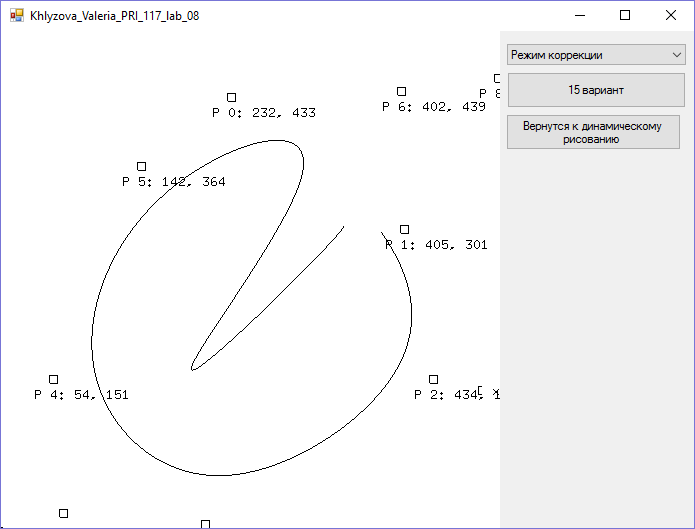


Рисунок 4. Редактирование сплайна.

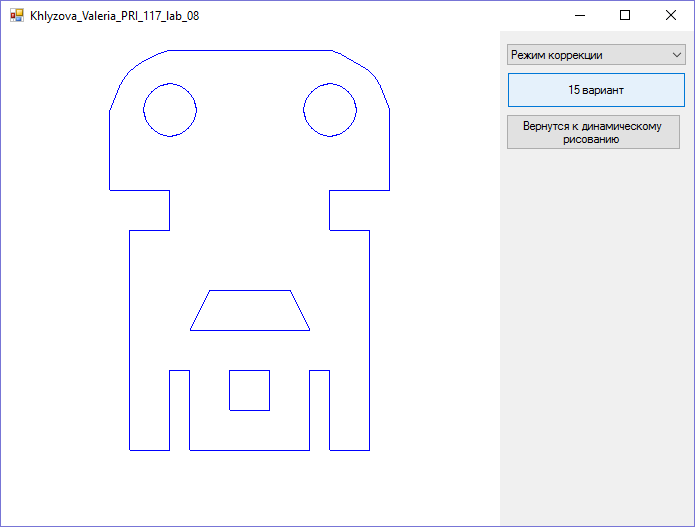


Рисунок 5. Вариант 15.

ВЫВОД

В ходе выполнения лабораторной работы были изучены основы применения сплайнов в компьютерной графике и освоены методы формирования изображений на основе сплайнов.