

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных технологий** | **Кафедра**  **Инженерной Графики** |

**Основная образовательная программа 09.03.02  
«Информационные системы и технологии»**

**Отчет по дисциплине «Компьютерная геометрия и графика»**

**по лабораторной работе № 5**

|  |  |
| --- | --- |
| Студент  Группы ИДБ-22-06 | Мустафаева П.М. |
|  |  |
| Преподаватель | к.т.н. , доц. Разумовский А.И. |

Москва, 2023

**ВВЕДЕНИЕ**

В этой лабораторной работе будут рассмотрены следующие необходимые аспекты:

1. Вращение объемных фигур

2. Построения сплайнов, виды сплайнов

Чтобы построить реалистичный трехмерный объект необходимо использовать перспективное преобразование координат. Для придания объекту большего реализма важно также настроить соотнесение его координат с источником света. Если объект также должен быть способен к вращению, то для этого необходимо соответствующим образом обрабатывать сообщения мыши WM\_MOUSEMOVE и WM\_MOUSEWHEEL.

**ЗАДАНИЕ 1**

Изобразить на экране вращающийся при помощи мыши куб с удаленными гранями и закраской.

Ниже представлен код для выполнения:

const int WIDTH = 400;

const int HEIGHT = 300;

float v11, v12, v13, v21, v22, v23, v31, v32, v33, v43;

float rho = 300., thetta = 75., phi = 30., ScreenDist = 500.;

float A, B, C, D, An, Bn, Cn;

float xt[3], yt[3], zt[3];

float Al, Bl, Cl;

float alpha;

float th, ph, costh, cosph, sinth, sinph;

float factor = atan(1.0) / 45.;

void VidMatCoeff(float rho, float thetta, float phi) { //определение компонентов направления наблюдения для сведения с лучом освещения

th = thetta \* factor; ph = phi \* factor;

costh = cos(th); sinth = sin(th); cosph = cos(ph); sinph = sin(ph);

v11 = -sinth; //элементы видовой матрицы

v12 = -cosph \* costh; v13 = -sinph \* costh;

v21 = costh; v22 = -cosph \* sinth; v23 = -sinph \* sinth;

v31 = 0.; v32 = sinph; v33 = -cosph; v43 = rho;

}

POINT Perspective(float x, float y, float z) {

POINT point;

float xe, ye, ze;

VidMatCoeff(rho, thetta, phi);

xe = v11 \* x + v21 \* y;

ye = v12 \* x + v22 \* y + v32 \* z;

ze = v13 \* x + v23 \* y + v33 \* z + v43;

point.x = ScreenDist \* xe / ze + 400.; //экранные координаты

point.y = ScreenDist \* ye / ze + 300.; //экранные координаты

return point;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM

wParam, LPARAM lParam)

{

PAINTSTRUCT ps; static HBRUSH hBrush; class TFPoint

{

public:

float X; float Y; float Z;

};

TFPoint CubePoints[] = { //задание координат точек куба

{-50,-50, -50},

{ 50,-50, -50},

{ 50, 50, -50},

{-50, 50, -50},

{-50, 50, 50},

{-50,-50, 50},

{ 50,-50, 50},

{ 50, 50, 50}

};

int Gran[6][4] = {

{0,3,4,5},

{0,5,6,1},

{2,7,4,3},

{7,6,5,4},

{0,1,2,3},

{2,1,6,7}

};

POINT point1[4];

HDC hdc;

int sx, sy, xPos, yPos, zDelta; switch (message)

{

case WM\_CREATE:

break; case WM\_SIZE:

sx = LOWORD(lParam); sy = HIWORD(lParam); break;

case WM\_MOUSEMOVE:

sx = LOWORD(lParam); //координаты мыши по оси x

sy = HIWORD(lParam); //координата мыши по оси y

thetta += ((sx % 180) - 90) / 10;

phi += ((sy % 180) - 90) / 10;

InvalidateRect(hWnd, NULL, TRUE); //обновление окна

break;

case WM\_MOUSEWHEEL:

zDelta = (int)wParam;

ScreenDist -= zDelta / 1000000.;

InvalidateRect(hWnd, NULL, TRUE);

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

th = thetta \* factor; ph = phi \* factor;

costh = cos(th); sinth = sin(th); cosph = cos(ph); sinph = sin(ph); A = rho

\* sinph \* costh; B = rho \* sinph \* sinth; C = rho \* cosph; Al = A / (sqrt(A \* A + B

\* B + C \* C));

Bl = B / (sqrt(A \* A + B \* B + C \* C));

Cl = C / (sqrt(A \* A + B \* B + C \* C));

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 3; j++) {

xt[j] = CubePoints[Gran[i][j]].X; yt[j] = CubePoints[Gran[i]

[j]].Y; zt[j] = CubePoints[Gran[i][j]].Z;

}

A = yt[0] \* (zt[1] - zt[2]) - yt[1] \* (zt[0] - zt[2]) + yt[2] \* (zt[0] -

zt[1]); B = -(xt[0] \* (zt[1] - zt[2]) - xt[1] \* (zt[0] - zt[2]) + xt[2] \* (zt[0] - zt[1]));

C = xt[0] \* (yt[1] - yt[2]) - xt[1] \* (yt[0] - yt[2]) + xt[2] \* (yt[0] - yt[1]); An = A / (sqrt(A \* A + B \* B + C \* C));

Bn = B / (sqrt(A \* A + B \* B + C \* C));

Cn = C / (sqrt(A \* A + B \* B + C \* C));

alpha = (An \* Al + Bn \* Bl + Cn \* Cl);

for (int j = 0; j < 4; j++) {

point1[j] = Perspective(CubePoints[Gran[i][j]].X,

CubePoints[Gran[i][j]].Y, CubePoints[Gran[i][j]].Z);

}

//определение направления обхода точек грани для выявления ориентации к наблюдателю

D = point1[0].x \* (point1[1].y - point1[2].y) -

point1[1].x \* (point1[0].y - point1[2].y) +

point1[2].x \* (point1[0].y - point1[1].y); if (D < 0)

{

hBrush = CreateSolidBrush(RGB((1 - alpha) \* 255,

(1 - alpha) \* 255, (1 - alpha) \* 255)); SelectObject(hdc,

hBrush);

Polygon(hdc, point1, 4);

}

}

EndPaint(hWnd, &ps); break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

Результат выполнения программы представлен ниже (рис. 1).

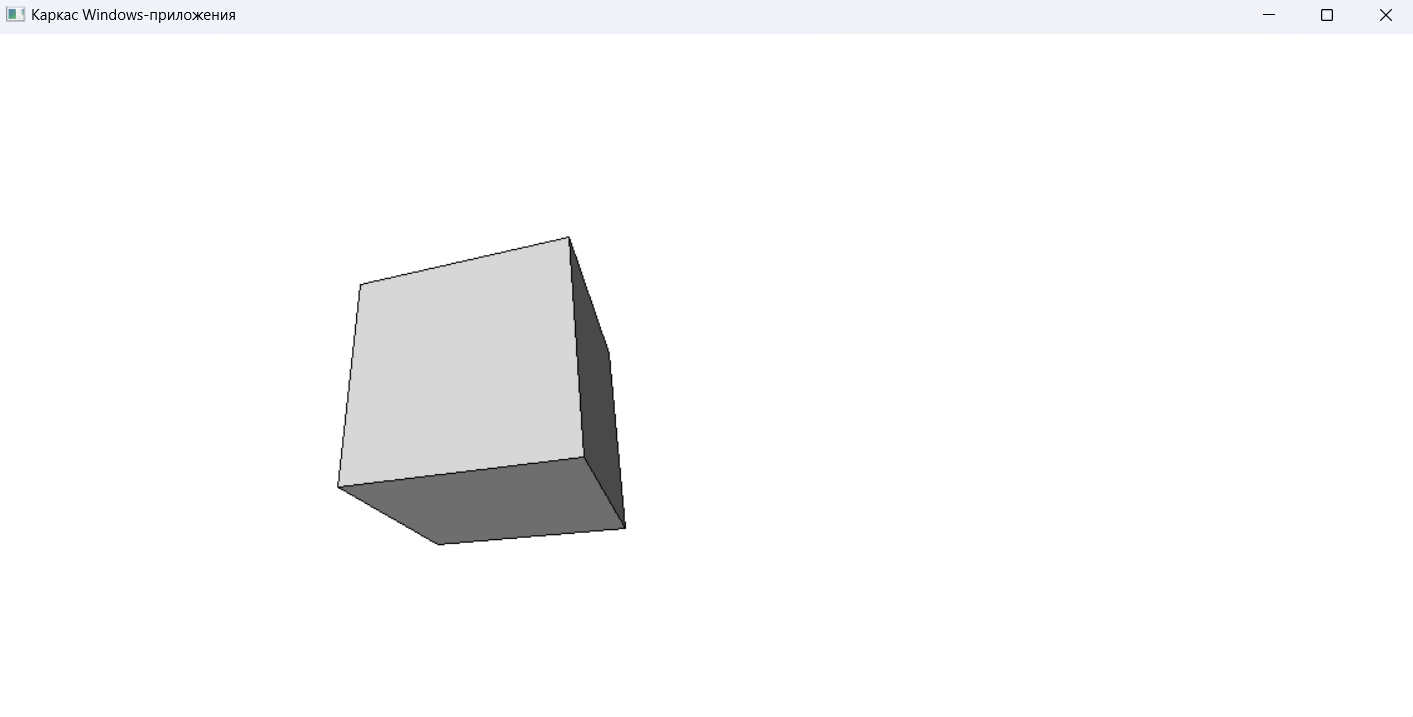


Рисунок 1. Вращающийся куб с удаленными гранями и закраской.

**ЗАДАНИЕ 2**

С помощью фрагментов предоставленного кода на основе приложения «Каркас» построить приложение «Кривая Безье».

Ниже представлена программа:

HDC hdc;

static int sx, sy;

const int SCALE = 1000;

const int MARK = 4;

void DcInLp(POINT& point)

{

point.x = point.x \* SCALE / sx;

point.y = SCALE - point.y \* SCALE / sy;

}

void transform(HDC& hdc)

{

SetMapMode(hdc, MM\_ANISOTROPIC);

SetWindowExtEx(hdc, SCALE, -SCALE, NULL);

SetViewportExtEx(hdc, sx, sy, NULL);

SetViewportOrgEx(hdc, 0, sy, NULL);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM

wParam, LPARAM lParam)

{

PAINTSTRUCT ps;

HDC hdc;

static HPEN hDash, hBezier; // два пера

static HBRUSH hRect, hSel; // две кисти

static POINT pt[20]; // массив хранения плоских точек

static POINT point; // структура под одну плоскую точку

RECT rt; // структура точек прямоугольника

static int count, index; // счётчик точек,

static bool capture; // логическая переменная для мыши

int i;

std::ifstream in; //класс файлового потокового ввода-вывода

std::ofstream out;

switch (message)

{

case WM\_CREATE:

in.open("dat.txt"); // открытие файлового потока

if (in.fail())

{

MessageBox(hWnd, \_T("Файл file.txt не найден"),

\_T("Открытие файла"), MB\_OK | MB\_ICONEXCLAMATION);

PostQuitMessage(0);

return 1;

}

// пока запоняется координата x - заполняем y

// в переменной count накапливается размер массива точек

for (count = 0; in >> pt[count].x; count++) in >> pt[count].y;

in.close(); // закрытие файлового потока

hDash = CreatePen(PS\_DASH, 1, 0); //пунктир

hBezier = CreatePen(PS\_SOLID, 4, RGB(0, 0, 255)); //сплошная линия

hRect = CreateSolidBrush(RGB(128, 0, 128)); //закраска прямоугольников

hSel = CreateSolidBrush(RGB(255, 0, 0)); //обводка прямоугольника

break;

case WM\_SIZE:

sx = LOWORD(lParam);

sy = HIWORD(lParam);

break;

case WM\_LBUTTONDOWN:

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

//Преобразование экранных координат мыши в логические

DcInLp(point);

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

{

SetRect(&rt, pt[i].x - MARK, pt[i].y - MARK,

pt[i].x + MARK, pt[i].y + MARK);

if (PtInRect(&rt, point))

{ // Курсор мыши попал в точку

index = i;

capture = true;

hdc = GetDC(hWnd);

transform(hdc); //Переход в логические координаты

FillRect(hdc, &rt, hSel);//Отметим прямоугольник цветом

ReleaseDC(hWnd, hdc);

SetCapture(hWnd);

return 0;

}

}

break;

case WM\_LBUTTONUP:

if (capture)

{

ReleaseCapture(); //Освобождение мыши

capture = false;

}

break;

case WM\_MOUSEMOVE:

if (capture)

{ //Мышь захвачена

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

DcInLp(point); //Преобразование экранных координат мыши

pt[index] = point; // в логические координаты

InvalidateRect(hWnd, NULL, TRUE);

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

transform(hdc); //Переход в логические координаты

SelectObject(hdc, hDash);

Polyline(hdc, pt, count); //Строим ломанную линию

SelectObject(hdc, hBezier);

PolyBezier(hdc, pt, count); //Строим кривую Безье

for (i = 0; i < count; i++) { //Закрашиваем точки графика прямоугольниками

SetRect(&rt, pt[i].x - MARK, pt[i].y - MARK, pt[i].x + MARK, pt[i].y + MARK);

FillRect(hdc, &rt, hRect);

}

} break;

case WM\_DESTROY:

PostQuitMessage(0);

DeleteObject(hDash);

DeleteObject(hBezier);

DeleteObject(hRect);

DeleteObject(hSel);

out.open("dat.txt"); //открыть файловый поток для записи даннах

for (i = 0; i < count; i++) out << pt[i].x << '\t' << pt[i].y << '\n';

out.close();

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

Результат выполнения программы представлен ниже (рис. 2).

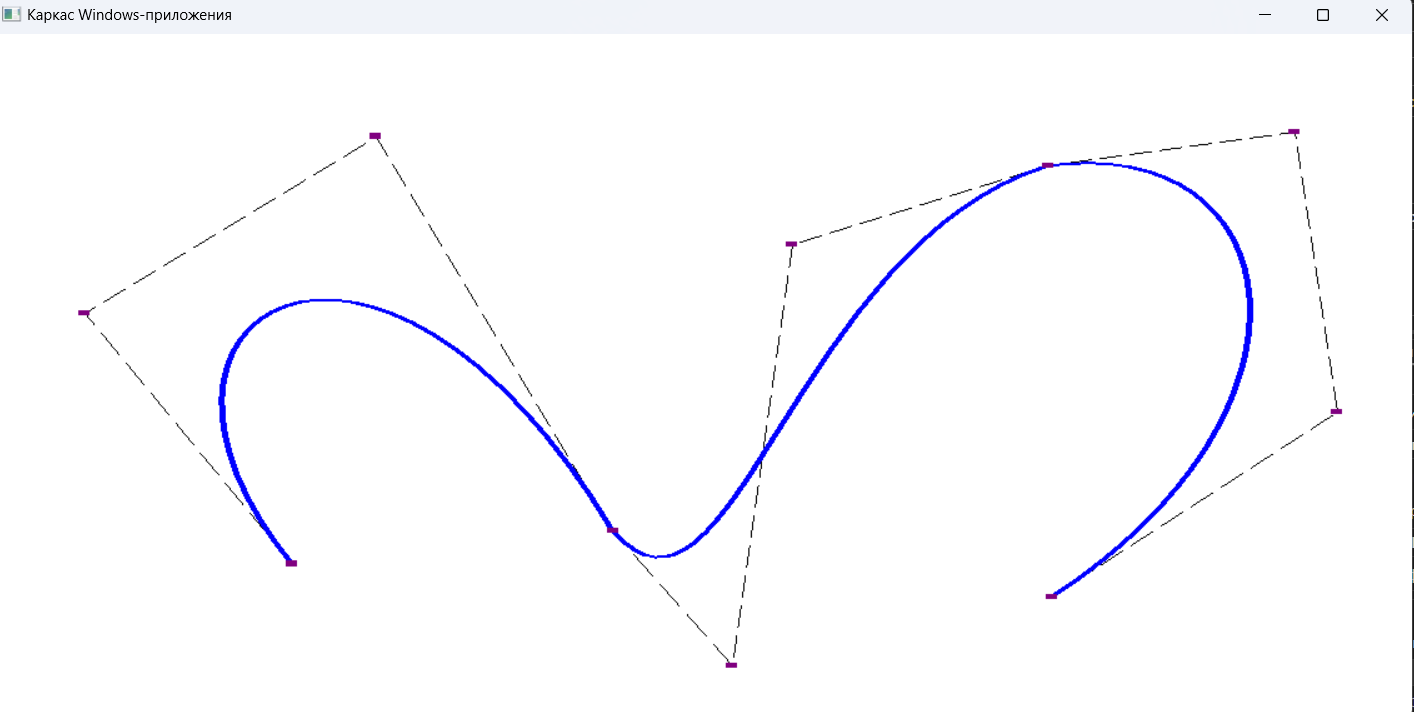


Рисунок 2. Приложение «Безье».

**ЗАДАНИЕ 3**

По примеру кода «Безье» адаптировать код программы, написанной в MS-DOS для построения кривой В-сплайна.

Ниже представлена программа:

HDC hdc;

PAINTSTRUCT ps;

static int sx, sy;

const int SCALE = 1000;

const int MARK = 4;

const int N = 30;

const int MAX = 100;

float x[MAX], y[MAX], eps = 6, X, Y, t, xA, xB, xC, xD,

yA, yB, yC, yD, a0, a1, a2, аЗ, b0, b1, b2, bЗ;

int n, i, j, first;

void DcInLp(POINT& point) {

point.x = point.x \* SCALE / sx;

point.y = SCALE - point.y \* SCALE / sy;

}

void transform(HDC& hdc) {

SetMapMode(hdc, MM\_ANISOTROPIC);

SetWindowExtEx(hdc, SCALE, -SCALE, NULL);

SetViewportExtEx(hdc, sx, sy, NULL);

SetViewportOrgEx(hdc, 0, sy, NULL);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message,

WPARAM wParam,

LPARAM lParam) {

static HPEN hDash, hPoint, hSpline;

static HBRUSH hRect, hSel;

static POINT pt[20];

static POINT point;

RECT rt;

static int count, index;

static bool capture;

int i;

std::ifstream in;

std::ofstream out;

switch (message) {

case WM\_CREATE:

in.open("data.txt");

if (in.fail()) {

MessageBox(hWnd, \_T("Файл не найден"),

\_T("Открытие файла"), MB\_OK |

MB\_ICONEXCLAMATION);

PostQuitMessage(0);

return 1;

}

for (count = 0; in >> pt[count].x; count++) in >>

pt[count].y;

in.close();

hDash = CreatePen(PS\_DASH, 1, 0);

hSpline = CreatePen(PS\_SOLID, 5, RGB(180, 180, 0));

hPoint = CreatePen(PS\_SOLID, 3, RGB(255, 0, 0));

hRect = CreateSolidBrush(RGB(128, 0, 128));

hSel = CreateSolidBrush(RGB(0, 255, 0));

break;

case WM\_SIZE:

sx = LOWORD(lParam);

sy = HIWORD(lParam);

break;

case WM\_LBUTTONDOWN:

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

DcInLp(point);

for (i = 0; i <= count; i++) {

SetRect(&rt, pt[i].x - MARK, pt[i].y - MARK,

pt[i].x + MARK, pt[i].y + MARK);

if (PtInRect(&rt, point)) {

index = i;

capture = true;

hdc = GetDC(hWnd);

transform(hdc);

FillRect(hdc, &rt, hSel);

ReleaseDC(hWnd, hdc);

SetCapture(hWnd);

return 0;

}

}

break;

case WM\_LBUTTONUP:

if (capture) {

ReleaseCapture();

capture = false;

}

break;

case WM\_MOUSEMOVE:

if (capture) {

point.x = LOWORD(lParam);

point.y = HIWORD(lParam);

DcInLp(point);

pt[index] = point;

InvalidateRect(hWnd, NULL, TRUE);

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

transform(hdc);

SelectObject(hdc, hDash);

Polyline(hdc, pt, count);

SelectObject(hdc, hPoint);

for (i = 0; i < count; i++) {

X = pt[i].x; Y = pt[i].y;

MoveToEx(hdc, pt[i].x - eps, pt[i].y - eps, 0);

LineTo(hdc, pt[i].x + eps, pt[i].y + eps);

MoveToEx(hdc, pt[i].x + eps, pt[i].y - eps, 0);

LineTo(hdc, pt[i].x - eps, pt[i].y + eps);

}

SelectObject(hdc, hSpline);

first = 1;

for (i = 1; i < count - 2; i++) {

xA = pt[i - 1].x;

xB = pt[i].x;

xC = pt[i + 1].x;

xD = pt[i + 2].x;

yA = pt[i - 1].y;

yB = pt[i].y;

yC = pt[i + 1].y;

yD = pt[i + 2].y;

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

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

a2 = (xA - 2 \* xB + xC) / 2.0; b2 = (yA - 2 \* yB +

yC) / 2.0;

a1 = (xC - xA) / 2.0; b1 = (yC - yA) / 2.0;

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

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

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

{

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

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

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

if (first) {

first = 0;

MoveToEx(hdc, X, Y, 0);

LineTo(hdc, X, Y);

}

else

LineTo(hdc, X, Y);

}

}

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

DeleteObject(hDash);

DeleteObject(hPoint);

DeleteObject(hSpline);

DeleteObject(hRect);

DeleteObject(hSel);

out.open("data.txt");

for (i = 0; i < count; i++) out << pt[i].x << '\t' <<

pt[i].y << '\n';

out.close();

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

Результат выполнения программы представлен ниже (рис. 3).

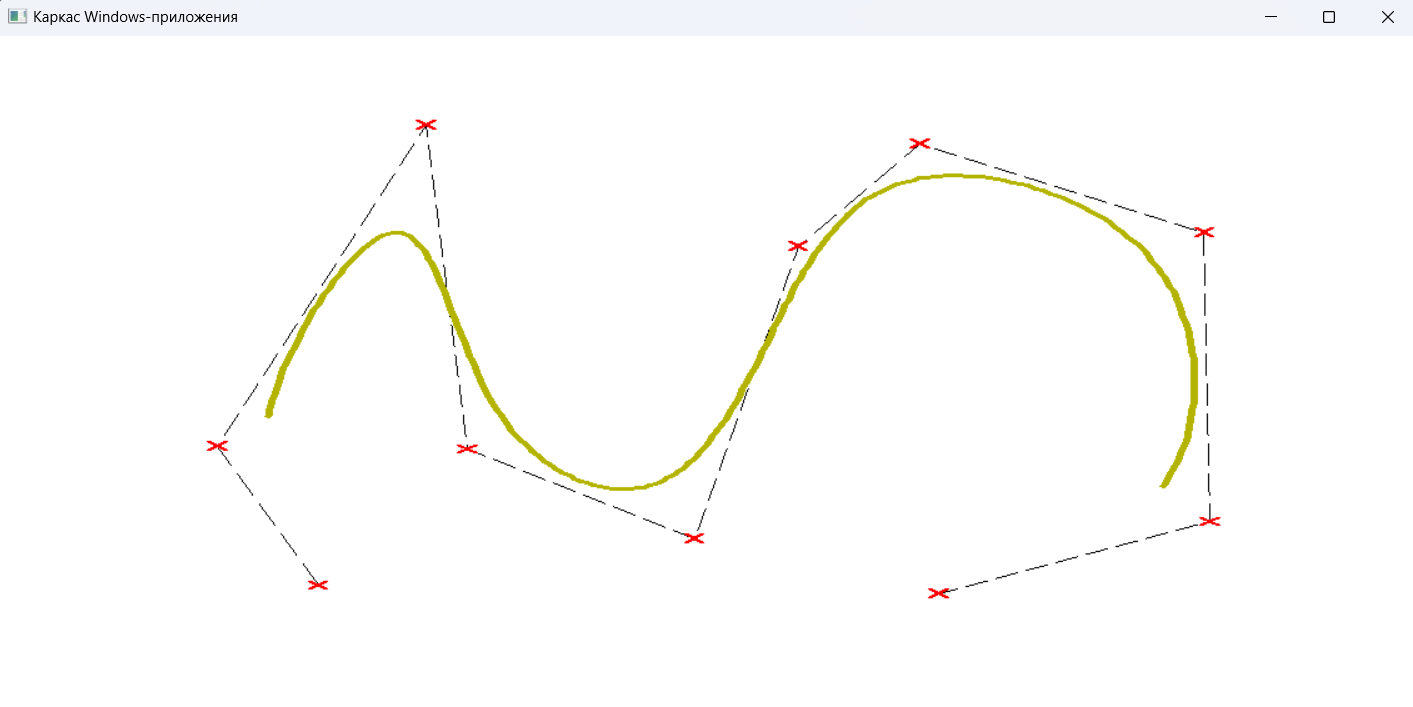


Рисунок 3. Приложение «Кривая В-сплайн».

**КООРДИНАТНЫЕ ДАННЫЕ**

225 199

154 402

301 871

330 398

490 267

563 694

649 844

849 714

853 292

662 187

**ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы было написаны 3 программы, одна из которых отрисовывает объемную, 3-х мерную, вращающуюся фигуру и 2 программы которые обрисовываю интерактивные кривые. Были освоены навыки вращения объемных фигур и построения сплайнов, вращающуюся фигуру.