**Ministerul Educației, Culturii și Cercetării**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Raport**

Lucrarea de laborator nr.3

Disciplina: Programarea pilotată de evenimente.

Tema: Curbe Bezier. Principiile de lucru cu tastatura. Principiile de lucru cu mouse-ul.

**Efectuat**: st.gr. TI-207 Bunescu Gabriel.

**Verificat**: asist. univ. Gaidarji Alina

Chișinău 2022

**Scopul lucrării:**

* studierea primitivelor grafice ce permit afişarea curbelor Bezier ;
* afişarea acestor curbe cu ajutorul formulelor;
* studierea metodelor şi principiilor de lucru cu tastatura;
* însuşirea modului de prelucrare a mesajelor parvenite de la tastatura;
* studierea metodelor şi principiilor de lucru cu mouse-ul;
* însuşirea modului de prelucrare a mesajelor parvenite de la şi mouse.

**Sarcina:**

Scrieţi un program care afişază curba Bezier, utilizând funcţiile GDI standarde şi o curbă Bezier, afişată după calcule prin formule.  Una din curbe să  poată fi mişcată cu ajutorul tastelor ← (la stânga), ↑(în sus), →(la dreapta), ↓(în jos). De asemenea, trebuie să fie prevăzute taste pentru rotirea acestei curbe. Modificați  curba a doua astfel ca punctele de control  să poată fi deplasate, schimbînd forma curbei în timp real.

**Codul sursă:**

#include <windows.h>

//conține declarații pentru toate funcțiile din API-ul Windows, toate macrocomenzile comune utilizate in windows

#include <tchar.h> //operate matematice de bază

#define IDM\_EXIT 1

#define IDM\_CLEAR 4

#define IDM\_BEZ1 100

bool bezier1 = false;

HINSTANCE hInst;

POINT Pt[4] = { { 100, 48 }, { 88, 246 }, { 364, 192 }, { 250, 48 } };

int varvar = 0;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);//este un indicator de funcție care este apelat de

//fiecare dată când este îndeplinită o anumită condiție

void mybezier(HDC hdc, const POINT\* Pt) {

mybezier(hdc, Pt);

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

// este convenția de apelare . O convenție de apelare definește modul în

//care o funcție primește parametrii de la apelant.

{

static char szAppName[] = "HelloWin"; // setează numele

HWND hwndMain; // parametru manerul ferestrei este null

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

// Indicator la procedura ferestrei pentru această clasă de ferestre

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;// Handle la instanța aplicației (pe care am

//primit - o în primul parametru al WinMain()

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);// Pictogramă

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);//culoarea fundalului

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = L"MyWindowClass";// numele care identifica clasa

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwndMain = CreateWindowEx(

WS\_EX\_CLIENTEDGE, // este stilul ferestre extinse,

L"MyWindowClass",

L"Lab3",

WS\_OVERLAPPEDWINDOW, // stilul ferestrei

CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 500,

NULL, NULL, hInstance, NULL); // mânerul instanței aplicației și un

//pointer către datele de creare a ferestrei.

ShowWindow(hwndMain, iCmdShow);

UpdateWindow(hwndMain);

while (GetMessage(&msg, NULL, 0, 0))

// GetMessage()primește un mesaj din coada de mesaje a aplicației dvs

{

TranslateMessage(&msg); // face unele procesări suplimentare pe tastatură

DispatchMessage(&msg); ; // trimite mesajul către fereastră că mesajul a fost trimis către

}

return msg.wParam; //bucla de mesaj

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

// este un indicator de funcție care este apelat de fiecare dată când este îndeplinită o anumită condiție

{

int wmId, wmEvent;

HDC hdc;

PAINTSTRUCT ps;

//Structura PAINTSTRUCT conține informații pentru o aplicație.Aceste informații pot

//fi folosite pentru a picta zona client a unei ferestre deținute de acea aplicație.

RECT rect; //Rect reprezintă un dreptunghi abstract și poate fi folosit într-o varietate de situații

HMENU hMenubar = CreateMenu();

HMENU hFile = CreateMenu();

HMENU hOptions = CreateMenu();

switch (iMsg)

{

case WM\_CREATE: {

// este un mesaj trimis atunci când o aplicație solicită

//crearea unei ferestre prin apelarea funcției CreateWindowEx

HMENU hMenubar = CreateMenu();

HMENU hFile = CreateMenu();

HMENU hOptions = CreateMenu();

AppendMenu(hMenubar, MF\_POPUP, (UINT\_PTR)hFile, L"File");

AppendMenu(hMenubar, MF\_POPUP, NULL, L"Edit");

AppendMenu(hMenubar, MF\_POPUP, (UINT\_PTR)hOptions, L"Bezier");

AppendMenu(hFile, MF\_STRING, IDM\_EXIT, L"Exit");

AppendMenu(hOptions, MF\_STRING, IDM\_BEZ1, L"BEZ\_1");

AppendMenu(hOptions, MF\_STRING, IDM\_CLEAR, L"CLEAR");

SetMenu(hwnd, hMenubar);

break;

}

case WM\_COMMAND:

wmId = LOWORD(wParam); // lățimea spațiului de lucru

wmEvent = HIWORD(wParam); // inaltimea spațiului de lucru

// Analizați selecția meniului:

switch (LOWORD(wParam))

{

case IDM\_EXIT:

DestroyWindow(hwnd);

break;

case IDM\_BEZ1:

bezier1 = true;

InvalidateRect(hwnd, NULL, TRUE);

break;

case IDM\_CLEAR:

bezier1 = false;

InvalidateRect(hwnd, NULL, TRUE);

break;

default:

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

break;

case WM\_PAINT:

// este trimis atunci când sistemul sau o altă aplicație

//solicită să picteze o porțiune din fereastra unei aplicații.

hdc = BeginPaint(hwnd, &ps);

if (bezier1)

PolyBezier(hdc, Pt, 4); //functia de desenare a polybenzier

return 0;

case WM\_KEYDOWN:

switch (wParam)

{

case VK\_NUMPAD8:

Pt[3].y--;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_NUMPAD2:

Pt[3].y++;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_NUMPAD6:

Pt[3].x++;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_NUMPAD4:

Pt[3].x--;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_UP:

Pt[0].y--;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_DOWN:

Pt[0].y++;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_RIGHT:

Pt[0].x++;

InvalidateRect(hwnd, NULL, TRUE);

break;

case VK\_LEFT:

Pt[0].x--;

InvalidateRect(hwnd, NULL, TRUE);

break;

default:

break;

}

break;

case WM\_MOUSEMOVE:

if (GetKeyState(MK\_LBUTTON))

{

Pt[varvar].x = LOWORD(lParam);

Pt[varvar].y = HIWORD(lParam);

InvalidateRect(hwnd, NULL, TRUE);

}

break;

case WM\_LBUTTONDOWN:

if (varvar == 3) varvar = -1;

varvar++;

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

**Rezultatul execuției:**

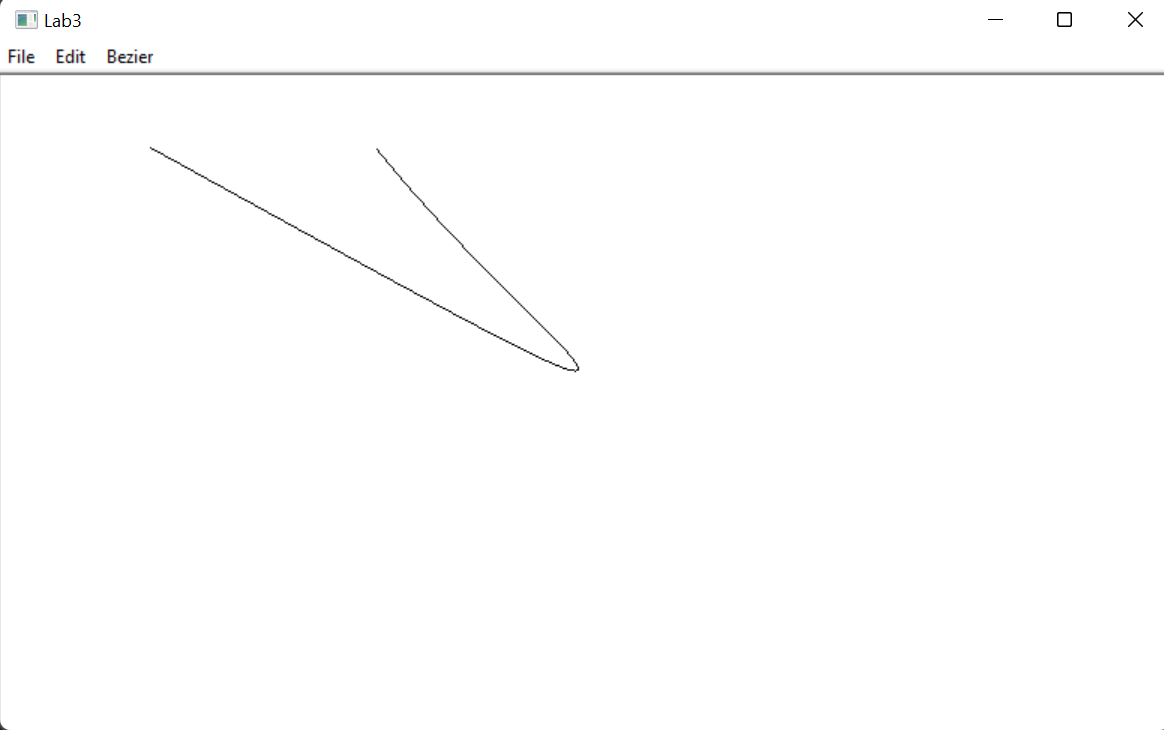


Figura 1. Rezultatul execuției

**Concluzie:**

În urma efectuării acestei lucrări de laborator am studiat bazele şi principiile de lucru cu tastatura și mouse-ul. De asemenea am învățat primitivele grafice ce  permit afișarea curbei Bezier, am învățat afișarea curbei cu ajutorul formulelor, am însușit modul de prelucrare a mesajelor parvenite de la tastatură si mouse.