Rysowanie w obszarze roboczym

1. Tworzymy aplikację okienkową.
2. Definiujemy wielkość obszaru roboczego na 640 x 480 px.

```
   hwnd = CreateWindowEx (
        0,              /* Extended possibilites for variation */
        szClassName,        /* Classname */
        "Program02",      /* Title Text */
        WS_OVERLAPPEDWINDOW, /* default window */
        CW_USEDEFAULT,      /* Windows decides the position */
        CW_USEDEFAULT,      /* where the window ends up on the screen */
        648,            /* The programs width */
        514,            /* and height in pixels */
        HWND_DESKTOP,       /* The window is a child-window to desktop */
        NULL,            /* No menu */
        hThisInstance,      /* Program Instance handler */
        NULL            /* No Window Creation data */
        );
```

3. Odszukujemy w kodzie programu instrukcję SWITCH:

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 case WM_DESTROY:
     PostQuitMessage (0);
     break;
 default:
     return DefWindowProc (hwnd, message, wParam, lParam);
}

   return 0;
}
```

- Funkcja ta zajmuje się obsługą komunikatów pochodzących z systemu Windows.
- Komunikatem może być ruch myszą lub wciśnięty klawisz.
- Możemy też umieścić tutaj instrukcje, które narysują coś na obszarze roboczym.
- Wystarczy wykorzystać komunikat WM_PAINT – odpowiedzialny za rysowanie okna.
- Jest uruchamiany automatycznie w chwili tworzenia okna oraz wtedy, gdy okno zostanie odsłonięte.
- Komunikat WM_DESTROY odpowiedzialny jest za usunięcie okna.
- Standardowe komunikaty przetwarzane są przez funkcję: *DefWindowProc (hwnd, message, wParam, lParam);*

4. Dopisujemy obsługę komunikatu WM_PAINT:

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 case WM_PAINT:

    break;
 case WM_DESTROY:
    PostQuitMessage (0);
    break;
 default:
    return DefWindowProc (hwnd, message, wParam, lParam);
}

   return 0;
}
```

5. Definiujemy dwie zmienne typu: PAINTSTRUCT i HDC:

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 PAINTSTRUCT ps;
 HDC hdc;
 case WM_PAINT:

    break;
 case WM_DESTROY:
    PostQuitMessage (0);
    break;
 default:
    return DefWindowProc (hwnd, message, wParam, lParam);
}

   return 0;
}
```

HDC – to identyfikator kontekstowy/uchwyt kontekstu urządzenia
PAINTSTRUCT – struktura rysunku

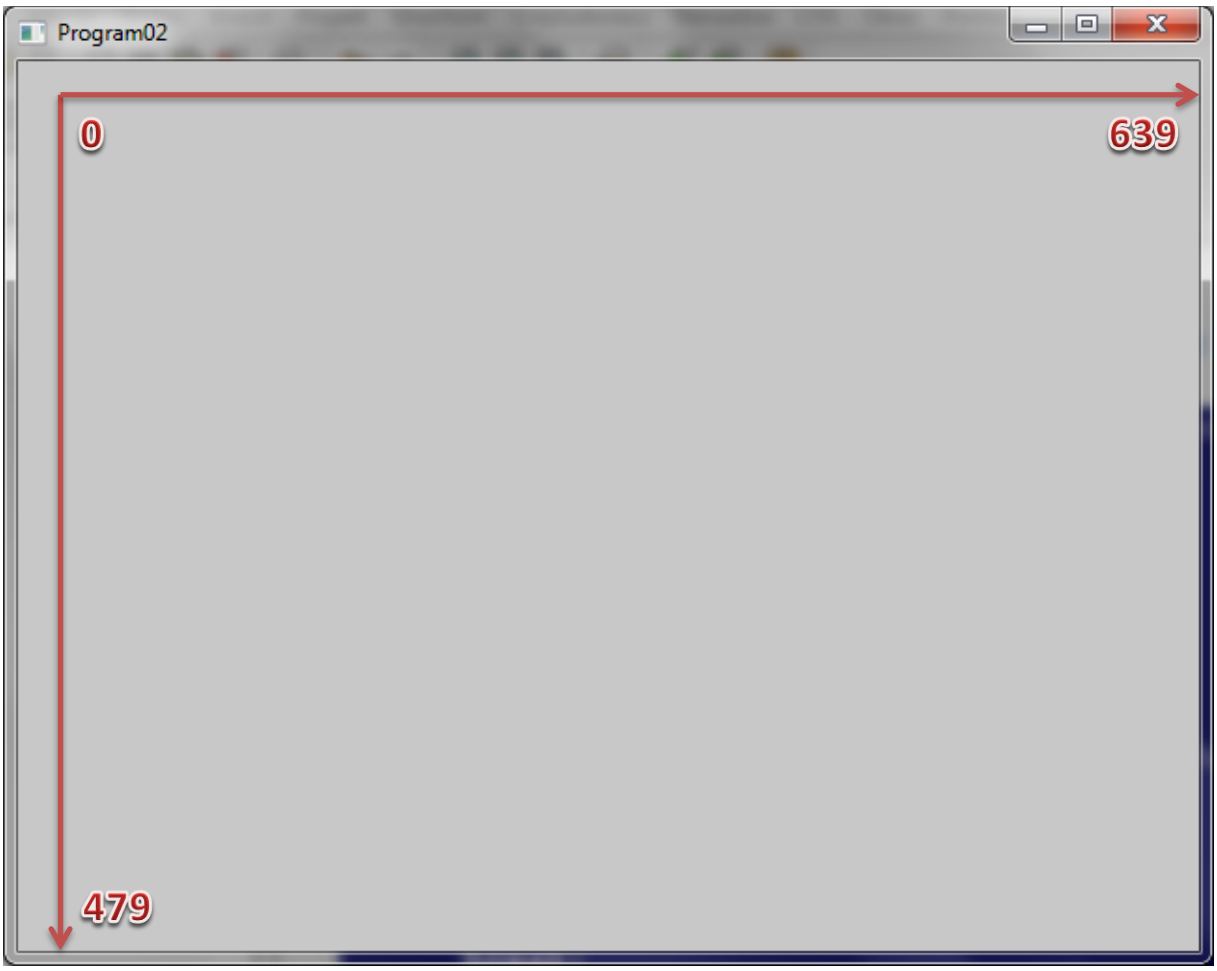
6. Dopisujemy wywołania fukcji BeginPaint() i EndPaint().

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 PAINTSTRUCT ps;
 HDC hdc;
```

```
 case WM_PAINT:
    hdc = BeginPaint(hwnd, &ps);
    /*Tu możemy umieszczać instrukcje obsługi grafiki*/
    EndPaint(hwnd, &ps);
    break;
 case WM_DESTROY:
    PostQuitMessage (0);
    break;
 default:
    return DefWindowProc (hwnd, message, wParam, lParam);
}

   return 0;
}
```

Program jest gotowy do rysowania w obszarze roboczym.



Zestawienie funkcji rysujących

BOOL Ellipse(HDC hdc, int x_lewy, int y_gorny, int x_prawy, int y_dolny);

```
Ellipse(hdc, 100, 20, 540, 460);
```

BOOL LineTo(HDC hdc, int x_koniec, int y_koniec);

```
LineTo(hdc, 639, 479);
```

BOOL MoveToEx(HDC hdc, int x_piora, int y_piora, LPPOINT poz);

```
MoveToEx(hdc, 0, 0, NULL);
```

COLORREF SetPixel(HDC hdc, int x, int y, COLORREF kolor);

```
SetPixel(hdc, 10, 10, RGB(50,170,255));
```

BOOL Rectangle(HDC hdc, int x_lewy, int y_gorny, int x_prawy, int y_dolny);

```
Rectangle(hdc, 400, 400, 450, 450);
```

BOOL RoundRect(HDC hdc, int x_lewy, int y_gorny, int x_prawy, int y_dolny, int x_e, int y_e);

```
RoundRect(hdc, 300, 300, 380, 380, 20, 20);
```

Zestawienie funkcji do tekstu

int SetBkMode(HDC hdc, int styl);

```
SetBkMode(hdc, TRANSPARENT);
SetBkMode(hdc, OPAQUE);
```

COLORREF SetBkColor(HDC hdc, COLORREF kolor);

```
SetBkColor(hdc, RGB(50,170,255));
```

COLORREF SetTextColor(HDC hdc, COLORREF kolor);

```
SetTextColor(hdc, RGB(255,0,0));
```

BOOL TextOut(HDC hdc, int x, int y, LPCTSTR napis, int dlugosc_napisu);

```
TextOut(hdc, 120, 38, "Ala ma kota, sierotka ma rysia!", 31);
```

Zaawansowane formatowanie tekstu

Aby zdefiniować dodatkowe atrybuty czcionki należy posłużyć się odpowiednią strukturą:

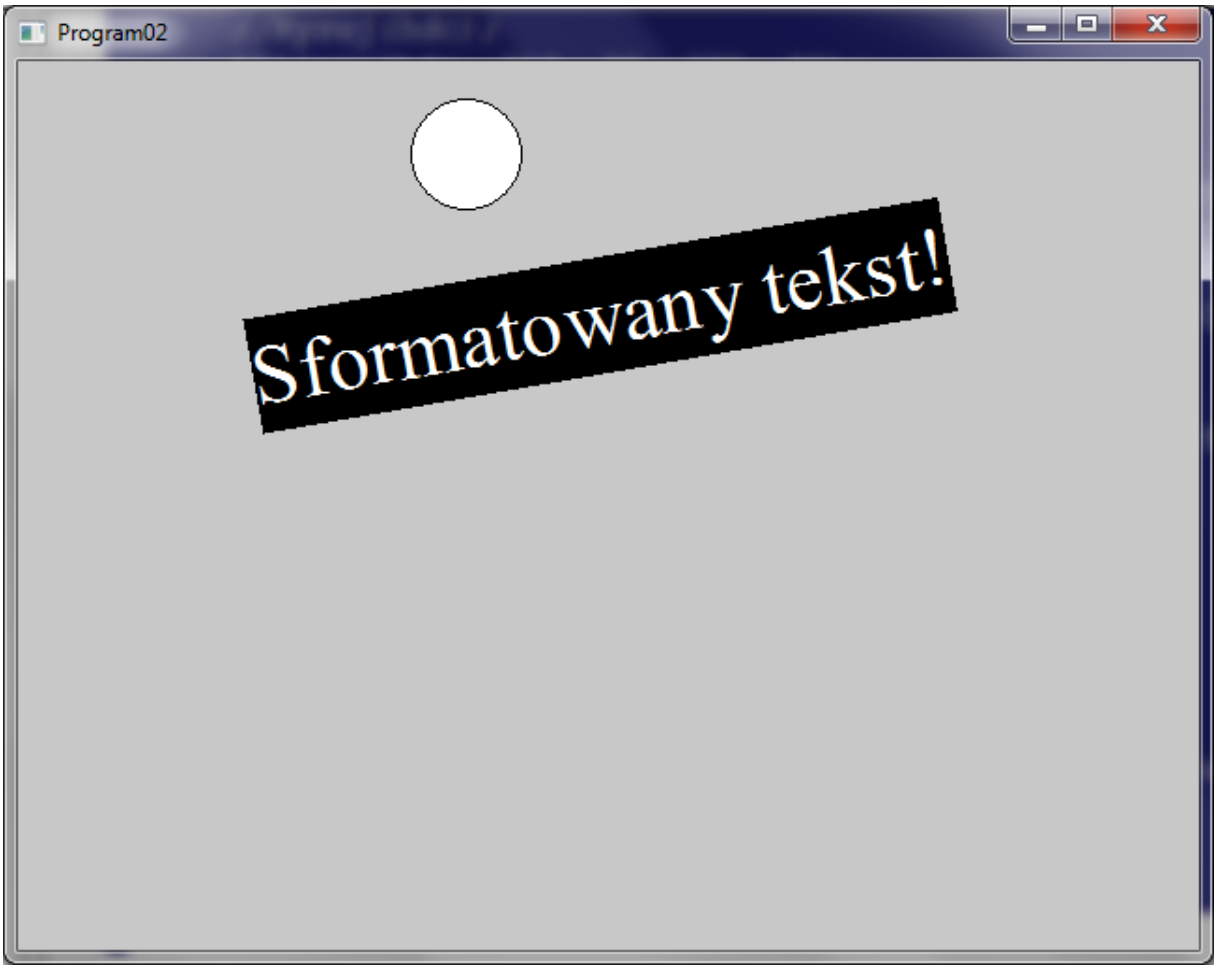
typedef struct tagLOGFONT {
LONG lfHeight; - wysokość czcionki
LONG lfWidth; - szerokość czcionki
LONG lfEscapement; - kąt wyświetlania tekstu
LONG lfOrientation; - kąt tekstu w stosunku do x
LONG lfWeight; - grubuść znaku
BYTE lfItalic; - kursywa
BYTE lfUnderline; - podkreślenie
BYTE lfStrikeOut; - przekreślenie
BYTE lfCharSet; - określenie zbioru znaków
BYTE lfOutPrecision;
BYTE lfClipPrecision;
BYTE lfQuality;
BYTE lfPitchAndFamily;
TCHAR lfFaceName[LF_FACESIZE]; - krój czcionki
} LOGFONT;

Jak ją wykorzystać?

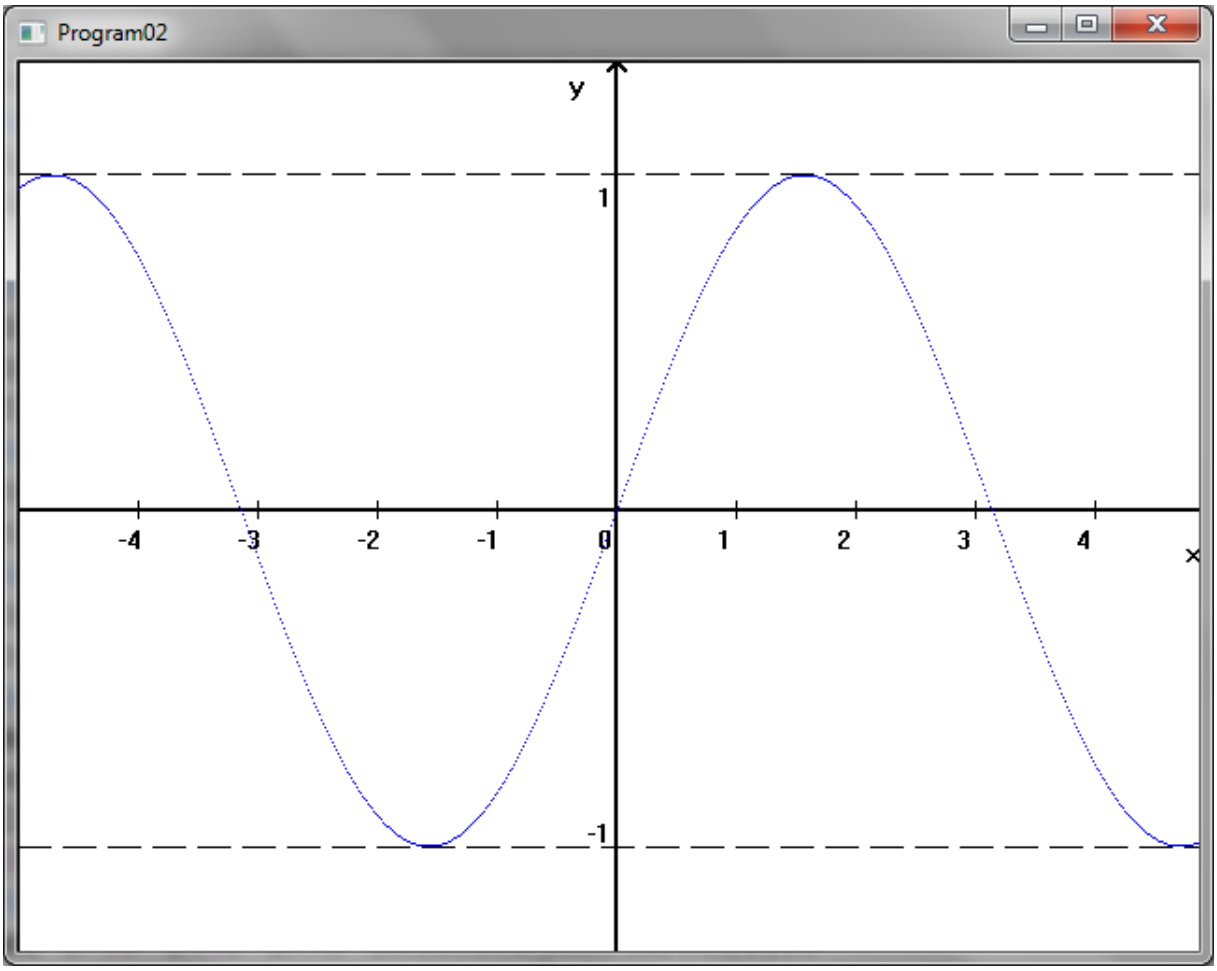
```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 PAINTSTRUCT ps;
 HDC hdc;
 HFONT hFont;
 case WM_PAINT:
    hdc = BeginPaint(hwnd, &ps);
    //Rysuj(hdc);
    Ellipse(hdc, 210, 20, 270, 80);
    SetBkColor(hdc, 0);
    SetTextColor(hdc, 0xFFFFFF);
    static LOGFONT lf; /* Deklaracja zmiennej struktury LOGFONT*/
    lf.lfHeight = 50;
    lf.lfEscapement = 100;
    lstrcpy (lf.lfFaceName, TEXT ("Times New Roman CE"));
    hFont = CreateFontIndirect(&lf);
    SelectObject(hdc, hFont);
    TextOut(hdc, 120, 138, "Sformatowany tekst!", 19);
    DeleteObject(hFont);
    EndPaint(hwnd, &ps);
    break;
 case WM_DESTROY:
    PostQuitMessage (0);
    break;
 default:
    return DefWindowProc (hwnd, message, wParam, lParam);
```

```
}

   return 0;
}
```



Przykład programu szkicującego wykres funkcji sinus



w main.cpp:

```
#include "okno2.h"

(...)

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
switch (message)
{
 PAINTSTRUCT ps;
 HDC hdc;
 case WM_PAINT:
     hdc = BeginPaint(hwnd, &ps);
     Rysuj(hdc);
     EndPaint(hwnd, &ps);
     break;
```

```
 case WM_DESTROY:
     PostQuitMessage (0);
     break;
 default:
     return DefWindowProc (hwnd, message, wParam, lParam);
}

   return 0;
}
```

w okno2.h:

```
#include <math.h>
const int MaxX = 640;
const int MaxY = 480;
void Rysuj(HDC hdc)
{
double x_lewy = -5.0;
double x_prawy = 5.0;
double krok_x = (x_prawy - x_lewy)/double(MaxX);
int MaxXd2 = MaxX/2;
int MaxYd2 = MaxY/2;
int MaxYd75 = int(MaxYd2*0.75);
//tworzenie piór
HPEN hPen[3];
hPen[0] = CreatePen(0, 0, 0);
hPen[1] = CreatePen(PS_DASH, 0, 0);
hPen[2] = CreatePen(0, 2, 0);
Rectangle(hdc, 0, 0, MaxX, MaxY);
SelectObject(hdc, hPen[2]);
//rysuj głowną oś x
MoveToEx(hdc, 0, MaxYd2, NULL);
LineTo(hdc, MaxX, MaxYd2);
//rysuj główną oś y
MoveToEx(hdc, MaxXd2, 0, NULL);
LineTo(hdc, MaxXd2, MaxY);
//rysuj strzałkę osi x
MoveToEx(hdc, MaxX, MaxYd2, NULL);
LineTo(hdc, MaxX-5, MaxYd2-5);
MoveToEx(hdc, MaxX, MaxYd2, NULL);
LineTo(hdc, MaxX-5, MaxYd2+5);
//rysuj strzałkę osi y
MoveToEx(hdc, MaxXd2, 0, NULL);
LineTo(hdc, MaxXd2-5, 5);
MoveToEx(hdc, MaxXd2, 0, NULL);
LineTo(hdc, MaxXd2+5, 5);
//rysuj nazwy osi
TextOut(hdc, MaxX-15, MaxYd2+15, "x", 1);
TextOut(hdc, MaxXd2-25, 5, "y", 1);
char tekst[8];
int MaxI = int(x_prawy-x_lewy);
int krok_i = MaxX/MaxI;
SelectObject(hdc, hPen[0]);
```

```
for(int i=int(x_lewy)+1; i<x_prawy; i++)
{
//rysuj współrzędne osi x
MoveToEx(hdc, MaxXd2+i*krok_i, MaxYd2-5, NULL);
LineTo(hdc, MaxXd2+i*krok_i, MaxYd2+5);
//rysuj wartości osi x
itoa(i, tekst, 10);
TextOut(hdc, MaxXd2+i*krok_i-10, MaxYd2+8, tekst, strlen(tekst));
}
//rysuj wartości osi y
TextOut(hdc, MaxXd2-10, MaxYd2-MaxYd75+5, "1", 1);
TextOut(hdc, MaxXd2-15, MaxYd2+MaxYd75-15, "-1", 2);
SelectObject(hdc, hPen[1]);
//rysuj granicę wartości y = 1
MoveToEx(hdc, 0, MaxYd2-MaxYd75, NULL);
LineTo(hdc, MaxX, MaxYd2-MaxYd75);
//rysuj granicę wartości y = -1
MoveToEx(hdc, 0, MaxYd2+MaxYd75, NULL);
LineTo(hdc, MaxX, MaxYd2+MaxYd75);
//rysuj wykres funkcji sinus
for(int x=0; x<MaxX; x++)
SetPixel(hdc, x,
MaxYd2 - int(MaxYd75*sin(x_lewy+double(x)*krok_x)),
0xFF0000);
//usuń utworzone piora
for(int i=0; i<3; i++) DeleteObject(hPen[i]);
}
```

Zadanie do wykonania

1. Napisz program, który narysuje domek ☺
2. Napisz program, który narysuje koła olimpijskie.
3. (*) Napisz program, który narysuje wykres funkcji kwadratowej $f(x)=x^2$