Událostmi řízené programování ve Windows

Cvičení VI

1 První program a příprava projektu

V tomto cvičení si ukážeme, jak vytvořit aplikaci pro OS Windows od úplného začátku. Vytvoříme si proto prázdný projekt pro desktopové aplikace. Ve Visual Studiu pro nový projekt použijeme *Windows Desktop Wizard*, zvolíme *Application type: Desktop Application (.exe)* a *Empty Project*.[1] Aby se předešlo problémům se zobrazením národních znaků, je v případě grafických aplikací vhodné pro reprezentaci textu používat výhradně široké (dvoubytové) znaky, tj. typ `wchar_t` nebo `WCHAR`.[2] To lze vynutit buď nastavením projektu (*Advanced — Use Unicode Character Set*) nebo definicí symbolu `#define UNICODE`, před vložením hlavičkového souboru `windows.h`.

Vstupním bodem grafických desktopových aplikací je funkce `wWinMain`,[3] které je předán handle na aktuální aplikaci (`hInstance`), argumenty (`lpCmdLine`) a příznaky, jak má být zobrazeno hlavní okno aplikace. Aplikaci, která zobrazí jednoduché okno pomocí funkce `MessageBox`,[4] ukazuje následující kód.

```
#ifndef UNICODE
#define UNICODE
#endif

#include <windows.h>

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nCmdShow)
{
    MessageBox(NULL, L"Hello world", L"My First Window", MB_ICONEXCLAMATION);
}
```

2 Složitější aplikace

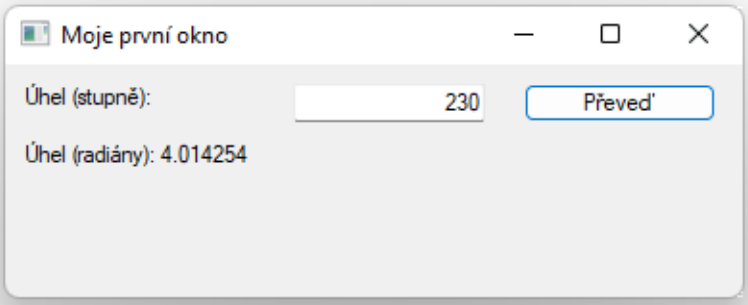
Nyní si ukážeme, jak vytvořit složitější aplikaci, která se bude skládat z jednoho okna, které bude sloužit k převodu úhlových jednotek, konkrétně k převodu stupňů na radiány. Předpokládaný výsledek můžete

[1] Tyto volby zajistí, že nebudeme mít projekt zaplněn dalšími předchystanými věcmi, které bychom stejně v tomto cvičení nevyužili.

[2] Použití tohoto typu je vhodné i z toho důvodu, že některé funkce jsou dostupné jen pro tento typ řetězců.

[3] `https://learn.microsoft.com/en-us/windows/win32/learnwin32/winmain--the-application-entry-point`

[4] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox`



Obrázek 1: Zamýšlený vzhled aplikace

vidět na Obrázku 1.

2.1 Vytvoření okna

K práci s okny, stejně jako práci s uživatelskými ovládacími prvky, používá operační systém Windows specifický objektový model. Chceme-li vytvořit okno, musíme vytvořit a zaregistrovat jeho třídu, například následovně.

```
1 WCHAR* WND_CLASS_NAME = L"First Wnd Class";
2 WNDCLASS wc = { .hInstance = hInstance,
3                 .lpfnWndProc = WindowProc,
4                 .lpszClassName = WND_CLASS_NAME };
5 RegisterClass(&wc);
```

Na řádku č. 1 definujeme název třídy námi vytvořeného okna, tento název by měl být v rámci aplikace unikátní. Na řádku č. 2 definujeme třídu okna, handle na běžící program, určíme funkci, která bude obsluhovat události spojené s oknem (řádek č. 3), a určíme název třídy (řádek č. 4). Nakonec třídu okna zaregistrujeme (řádek č. 5).

K samotnému vytvoření okna[5] slouží funkce `CreateWindow`,[6] případně `CreateWindowsEx`,[7] která umožňuje nastavit některé dodatečné vlastnosti.

```
1 HWND hwnd = CreateWindow(
2     WND_CLASS_NAME,      // trida okna
3     L"Moje první okno",  // nadpis
4     WS_OVERLAPPEDWINDOW, // styl okna
5     CW_USEDEFAULT,       // pozice x
```

[5] Vytvoření instance okna v terminologii OOP.
[6] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-createwindoww`
[7] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-createwindowexw`

```
6       CW_USEDEFAULT,      // pozice y
7       400,                // sirka
8       160,                // vyska
9       NULL,               // rodicovkseho okno
10      NULL,               // menu
11      hInstance,          // handle aplikace
12      NULL                // dodatecna data
13  );
```

Všimněme si zejména řádku č. 2, kde k identifikaci třídy okna používáme řetězec, který jsme určili při registraci třídy. Dále si všimněme, že funkce vrací hodnotu typu `HWND`, což je handle, jehož prostřednictvím budeme s oknem pracovat. Jednou z operací, kterou s oknem musíme provést, je jeho zobrazení pomocí funkce `ShowWindow`:

```
ShowWindow(hwnd, nCmdShow);
```

2.2 Smyčka událostí

Windows z pohledu grafických aplikací jsou událostmi řízený systém. To znamená, že běžící program získává od operačního systému zprávy o vzniklých událostech, na které pak definovaným způsobem reaguje. Typickými příkladem událostí jsou například stisknutí klávesy nebo kliknutí kurzorem myši na okno programu. Dále pomocí událostí operační systém sděluje, že se okno má překreslit nebo že došlo ke změně rozměrů okna.

Aby program mohl reagovat na jednotlivé příchozí události, obsahuje obvykle funkce `wWinMain` smyčku následujícího formátu:

```
1  MSG msg;
2  while (GetMessage(&msg, NULL, 0, 0) > 0) {
3      TranslateMessage(&msg);
4      DispatchMessage(&msg);
5  }
```

Na řádku č. 2 funkcí `GetMessage`[8] získáme zprávu, na kterou má program zareagovat, např. kliknutí myší nebo požadavek na překreslení okna. Tyto zprávy mohou být dalé transformovány, např. stisk kláves může být převeden na konkrétní událost, viz řádek č. 3 a následně je zpráva předána ke zprácování (řádek č. 4).

2.3 Reakce na příchozí zprávy

Funkce `DispatchMessage`[9] předá zprávu příslušnému oknu, které na ni zareaguje funkcí, kterou jsme uvedli při vytváření třídy okna, viz atribut `lpfnWndProc` a hodnota `WindowProc`.[10] Tato funkce je typu:

[8] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getmessage`
[9] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-dispatchmessage`
[10] https://learn.microsoft.com/en-us/windows/win32/learnwin32/writing-the-window-procedure

```
LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);
```

První argument udáva handle okna, kterého se událost týká, druhý argument představuje identifikátor zprávy, který okno obdrželo a zbývající dva argumenty představují informace přidružené ke zprávě, např. souřadnice, příznaky, textové informace apod. Význam těchto dvou argumentů je specifický pro každý typ zprávy a je potřeba jej vyčíst z dokumentace.

Funkce `WindowProc` typicky obsahuje jeden `switch`, který se stará o zpracování událostí, na které umí zareagovat, minimálně by to měla být událost `WM_PAINT`, která se stará o překreslení okna. V následujícím ukázkovém příkladě při obdržení zprávy `WM_PAINT` bude obdélníková oblast okna, která má být překreslena, vyplněna barvou okna. Dále při zrušení okna (zpráva `WM_DESTROY`) bude odeslána zpráva `WM_QUIT`, která ukončí smyčku událostí. Pokud nemáme způsob, jak na zprávu zareagovat, použijeme implicitní zpracování pomocí funkce `DefWindowProc`.[11]

```
LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg) {
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hwnd, &ps);
        FillRect(hdc, &ps.rcPaint, (HBRUSH)(COLOR_WINDOW));
        EndPaint(hwnd, &ps);
        return 0;
    }
    }
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}
```

2.4 Ovládací prvky

Pokud přeložíme a spustíme dosud popsaný kód, mělo by se nám zobrazit prázdné okno. Do tohoto okna je nutné vložit jednotlivé ovládací prvky. Ovládací prvky, jako jsou tlačítka, textová pole, popisky apod. jsou reprezentovány taktéž jako okna. K jejich vytvoření se proto používá funkce `CreateWindow`. Ukažme si použití na příkladu vytvoření tlačítka:

```
1 HWND hwndConvertBtn = CreateWindow(
2     L"BUTTON",   // trida ovladaciho prvku: tlacitko
3     L"Převeď",   // popisek tlačítka
4     WS_TABSTOP | WS_VISIBLE | WS_CHILD | BS_DEFPUSHBUTTON,  // styl
```

[11] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-defwindowprocw`

```
5       270,          // pozice x
6       10,           // pozice y
7       100,          // sirka
8       20,           // vyska
9       hwnd,         // rodicovske okno
10      (HMENU)IDBTNCONVERT,  // identifikator vstupu
11      hInstance,
12      NULL);
```

V tomto případě jsme jako třídu okna použili řetězec `"BUTTON"`. Kdybychom chtěli vytvořit textový vstup, použili bychom `"EDIT"`, pro statický popisek `"STATIC"`. Na řádku č. 4 jsou uvedeny vlastnosti tlačítka. Ty zahrnují vlastnosti obecného okna (`WS_*`) a vlastnosti specifické pro konkrétní třídu okna (`BS_*`, `ES_*` apod.)

Umístění jednotlivých ovládacích prvků a jejich rozměry se uvádí jako absolutní hodnoty v pixelech. Na řádcí č. 9 a 11 předáváme odkazy na rodičovské okno a aktuální program.

Důležitý je řádek č. 10, kde uvádíme číselný identifikátor konkrétního ovládacího prvku. Měla by to být číselná hodnota větší nebo rovna 100, která je přetypovaná na typ `HMENU`. Význam této hodnoty bude zřejmý z následujícího textu.

Poznámka: Přiložené zdrojové kódy ukazují vytvoření dalších ovládacích prvků.

2.5 Reakce na události spojené s ovládacími prvky

Pokud dojde ke stisku tlačítka, volbě položky z menu, změně textu v textovém vstupu, je rodičovskému oknu zaslána zpráva `WM_COMMAND` a to má možnost na tuto zprávu zareagovat. Abychom mohli odlišit, od kterého ovládacího prvku zpráva přišla, používá se jako identifikátor hodnota, která je předána funkci `CreateWindow` (viz řádek č. 10 v předchozím výpisu kódu) a tato hodnota se objeví ve `WindowProc` jako parametr `wParam`. Použití v konstrukci `switch` vypadá následovně.

```
case WM_COMMAND:
{
    if (wParam == IDBTNCONVERT) {
        onConvertButtonClick(hwnd);
        return 0;
    }
}
```

2.6 Další využití zásílání zpráv

Zasílání zpráv oknům (ať už klasickým oknům nebo ovládacím prvkům) je obecný mechanismus, kterým s okny manipulujeme. Chceme-li například změnit text, slouží k tomu zpráva `WM_SETTEXT`.[12] Tu můžeme

[12] https://learn.microsoft.com/en-us/windows/win32/winmsg/wm-settext

oknu explicitně zaslat pomocí funkce `SendMessage`,[13] kdy řetězec předáme pomocí parametru `lParam`, viz dokumentace.

```
SendMessage(hwndResult, WM_SETTEXT, 0, (LPARAM) L"Novy text");
```

Protože manipulace s objekty pomocí explicitního zasílání zpráv není zcela komfortní, existuje řada funkcí, která tento způsob práce s objekty zakrývají. Například pro změnu textu okna existuje funkce `SetWindowText`, která obstará zaslání zprávy v odpovídajícím formátu.

Poznámka: Přiložené zdrojové kódy ukazují, jak lze pomocí zasílání zpráv získat textovou hodnotu. Je to nutné udělat ve třech krocích, nejdříve musíme zjistit velikost textu, pak pro něj alokovat buffer, a nakonec získat text. To se děje zasláním zprávy `WM_GETTEXT`, kde jako `lParam` uvedeme cílový buffer.

3 Závěrečné poznámky a úkoly

3.1 Zkrocení zpětné kompatibility

Pokud aplikaci spustíme v této podobě, její vzhled bude silně „retro", tj. bude vypadat jako něco z dávné počítačové historie, čti z osmdesátých let. Tento vzhled odpovídá přibližně Windows 3.x nebo 95, viz Obrázek 2. Toto chování je důsledkem zachování zpětné kompatibility s předchozími verzemi Windows. Aby aplikace vypadalo soudobě, je nutné načíst novější verzi knihovny s uživatelskými prvky, tj. knihovnu `comctl32` ve verzi minimálně 6.0. K tomu můžeme použít buď nastavení projektu nebo následující direktivy překladače:

```
#pragma comment(lib, "comctl32.lib")
#pragma comment(linker,"\"/manifestdependency:type='win32' \
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \
processorArchitecture='*' publicKeyToken='6595b64144ccf1df' language='*'\"")
```

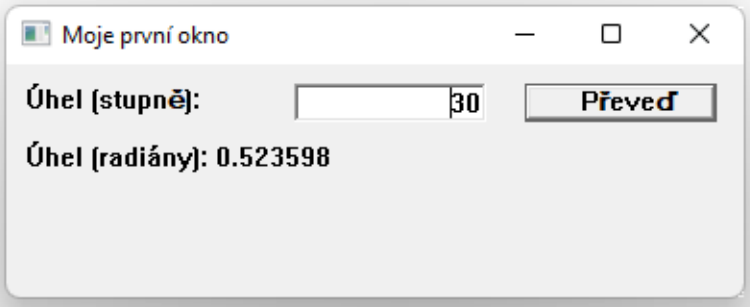
Dále je nutné nastavit estetičtější font:[14]

```
HFONT font = CreateFont(12, 0, 0, 0, FW_NORMAL, FALSE, FALSE, FALSE, DEFAULT_CHARSET,
                        OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                        DEFAULT_PITCH | FF_DONTCARE, L"MS Shell Dlg");
SendMessage(hwndInputCaption, WM_SETFONT, (WPARAM)font, TRUE);
SendMessage(hwndInput, WM_SETFONT, (WPARAM)font, TRUE);
SendMessage(hwndConvertBtn, WM_SETFONT, (WPARAM)font, TRUE);
SendMessage(hwndResult, WM_SETFONT, (WPARAM)font, TRUE);
```

Všimněme si, že ke změně fontu je použita zpráva `WM_SETFONT`.

[13] `https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-sendmessage`

[14] Ten původní je bitmapový a již v devadesatých letech vypadal hodně archaicky.



Obrázek 2: Nemoderní vzhled aplikace

3.2 Tvorba skutečných programů

Cílem tohoto cvičení bylo primárně představit mechanismus zasílání zpráv, jak je realizován v operačním systému Windows. Skutečné programy pro Windows se většinou tímto způsobem nevytváří. Definice vzhledu oken, menu apod. se provádí v samostatných souborech, které se označují jako `resources` a tyto soubory jsou vytvářeny typicky pomocí grafických nástrojů jako je Visual Studio, což značně zjednodušuje vývoj.

Poznámka: Vytvořte ve Visual Studiu vzorovou desktopovou aplikaci a podívejte se, jak jsou definovány jednotlivé „resources" a jak se s nimi pracuje z kódu programu.

Protože nativní rozhraní Windows není příliš komfortní, často se nad ním staví samostatné vrstvy, např. v jazyce C++, které programátora odstiňují od spousty technických detailů včetně toho, jak je realizována smyčka událostí a reakce na jednotlivé události. V praxi se dá běžně setkat i s tím, že si aplikace vykreslují obsah okna zcela ve vlastní režii, včetně jednotlivých ovládacích prvků, a z OS si berou jen mechanismus pro zasílání zpráv.

3.3 Úkoly

Úkol č. 1: Upravte aplikaci tak, aby se počet radiánů zobrazoval v textovém poli a bylo tam další tlačítko, které umožní převod z radiánů na stupně.

Úkol č. 2: Upravte aplikaci tak, aby testovala, jestli je na vstupu validní hodnota (číslo), a pokud ne, objeví se okno (MessageBox) s chybovou zprávou.