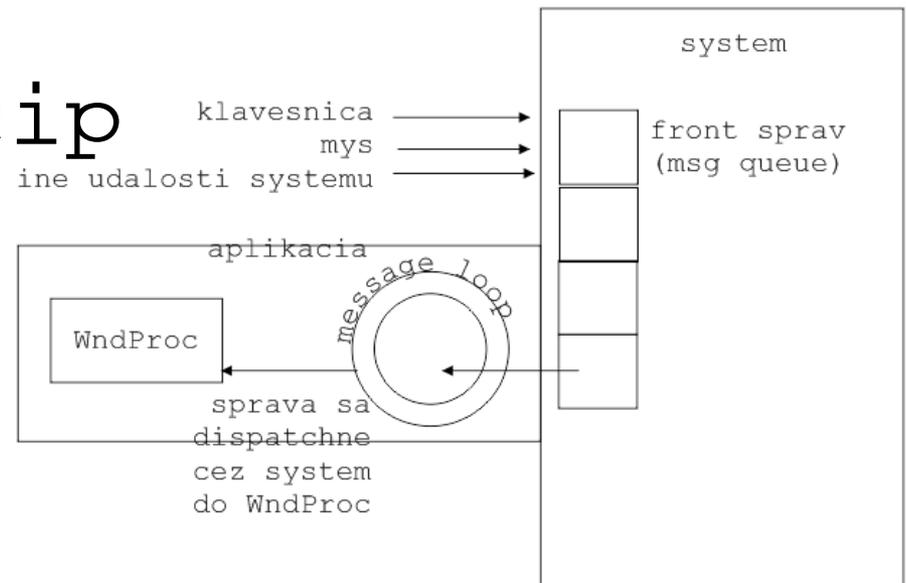


systemove programovanie
win32 programovanie

zakladny princip



- uzivatel interaguje so systemom
 - klavesnicou, mysou
- tym generuje udalosti, ktore sa radia do „message queue“ (front sprav)
- aplikacia vytahuje v cykle spravy z MQ a spracovava ich
 - nekonecny while() = *message loop*

zakladny princ

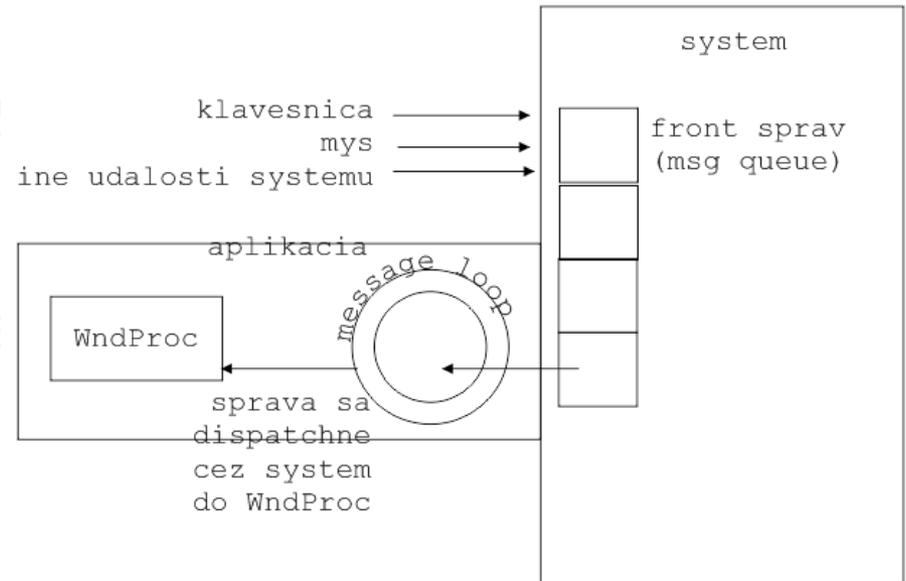
- sprava vytiahnuta z message queue je odoslana

na spracovanie do virtualnej funkcie

– dispatch message

- v aplikacii je teda cyklus:

1. vytiahni spravu
2. urci jej typ
3. posli do metody
4. goto 1



zakladny princip

- tento princip sa pouziva takmer vsade, kde je gui
 - java swing
 - MQ je schovana
 - EDT vlakno manazuje MQ
 - gtk
- jediny problem: message dispatch musi byt rychly
 - inak sa ,,kotvi" user interface



vykreslovane

- vykreslovane vo windowse riesi GDI (resp. GDI+)
 - (nie global defense initiative z C&C)
- GDI ma klasicke operacie
 - nakresli obdlznik
 - elipsu
 - vykresli text
- vo Viste a novsom je GDI na odchode
 - nemozno hw akcelerovat atd

programovanie v c

- zakladne kroky:
 - #include <windows.h>
- jemny rozdiel:
 - entry point je WINMAIN
 - nie main()

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE  
    hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
```

vytvorenie okna: WNDCLASS

- „trieda okna“
- vsetky okna s rovnakou WNDCLASS zdielaju *window procedure*
 - teda proceduru spracovavajucu spravy
- cez triedu mozno nastavit spolocne atributy okna
 - titulok, pozadie, ...

```
MSG msg ;
WNDCLASS wc = {0};
wc.lpszClassName = TEXT( "Center" );
wc.hInstance      = hInstance ;
wc.hbrBackground = GetSysColorBrush(COLOR_3DFACE);
wc.lpfnWndProc    = WndProc ;
wc.hCursor        = LoadCursor(0, IDC_ARROW);

RegisterClass(&wc);
CreateWindow( wc.lpszClassName, TEXT("Center"),
             WS_OVERLAPPEDWINDOW | WS_VISIBLE,
             100, 100, 250, 150, 0, 0, hInstance, 0);

while( GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return (int) msg.wParam;
```

ukazkovy
WINMAIN

vytvorenie okna: CreateWindow

- okno (window) je kazdy ovladaci prvok
- vytvorenie „instancie“:

```
HWND WINAPI CreateWindow(  
LPCTSTR lpClassName,  
LPCTSTR lpWindowName,  
DWORD dwStyle,  
int x, int y, int nWidth, int nHeight,  
HWND hWndParent,  
HMENU hMenu,  
HINSTANCE hInstance,  
LPVOID lpParam );
```

vytvorenie okna: CreateWindow

- nazov triedy okna
- titulok
- styl
- koordiny (x, y, sirka, vyska)
- identifikator
- a vlastnika (hInstance)

```
CreateWindow( wc.lpszClassName, TEXT("Center"),  
             WS_OVERLAPPEDWINDOW | WS_VISIBLE,  
             100, 100, 250, 150, 0, 0, hInstance, 0);
```

message loop

- MSG: window message
 - správa okna (prakticky int)
- Get: vytiahne z queue
- Translate: vyriesi kombinacie klaves
- Dispatch: odosle window procedure

```
MSG msg;

while( GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

window procedure

- procedura, ktora prijima message
- prakticky mega-switch, kde na zaklade typu spravy povie, co sa ma stat
- priklad: po sprave DESTROY zavrie okno

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

vytváranie komponentov formulara

- vo window procedure reagujeme na WM_CREATE
- povytvarame komponenty
 - CreateWindow
- kazdy komponent Windows ma svoj preddefinovany window class
 - priklad: label = "static"

vytvarame label

- vo WndProc, class: STATIC
- hwnd: handle rodica
- (HMENU) 1: identifikator konkrétného komponentu, ktorým sa môžeme riadiť v obsluhu udalosti

```
case WM_CREATE: {
    CreateWindow(TEXT("STATIC"),
                TEXT("Hello"),
                WS_CHILD | WS_VISIBLE | SS_LEFT,
                20, 20, 300, 230,
                hwnd,
                (HMENU) 1,
                NULL, NULL);

    return 0;
}
```

vytvarame button

- vo WndProc, class: button
- hwnd: handle rodica
- (HMENU) 2: identifikator konkrétneho komponentu, ktorým sa mozeme riadiť v obsluhu udalosti

```
CreateWindow(TEXT("button"), TEXT("Quit"),  
            WS_VISIBLE | WS_CHILD ,  
            120, 50, 80, 25,  
            hwnd, (HMENU) 2, NULL, NULL);
```

obsluha udalosti

- window procedure ma parametre suvisiace so spravou
- HWND: handle okna
- UINT msg: cislo spravy
 - existuje kilometrový zoznam preddefinovaných sprav
 - aplikacia moze definovat vlastne
 - vid WinAmp
- WPARAM, LPARAM: semantika zavisi od konkretnej spravy

obsluha udalosti

- reagujeme typicky na WM_COMMAND
- priklad: po kliknuti na tlacidlo s HMENU 2 koncime aplikaciu

```
case WM_COMMAND:
    {
        if (LOWORD(wParam) == 2) {
            PostQuitMessage(0);
        }

        break;
    }
```

poznámky

- window procedure musí obslužit všechny správy
- tie, ktoré nevie, musí delegovať na defaultnú window procedure

```
return DefWindowProc(hwnd, msg, wParam, lParam);
```

ako dalej, pan programator

- MSDN
- tutorialy:
 - husty, ale dostatočný:
 - <http://zetcode.com/tutorials/winapi/>
- jemne obsiahlejši
 - <http://www.winprog.org/tutorial>