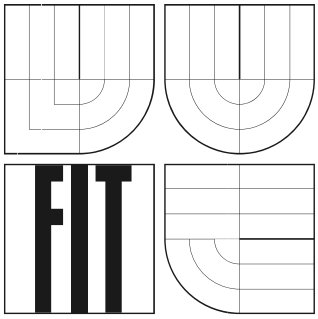
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Vývoj okenního systému ve WinAPI

Bakalářská práce

2006 Petr Dolejší

Vývoj okenního systému ve WinAPI

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pečivy.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Abstrakt

Tento text pojednává o souboru nástrojů sloužících pro tvorbu okenního systému s využitím WinAPI. Knihovna poskytuje programátorovi objektově orientované vysokoúrovňové programové rozhraní (API) napsané v C++. Pomocí tohoto rozhraní lze jednoduše vytvořit okenní aplikaci ve Windows ™, bez nutnosti jakékoliv znalosti WinAPI. Takto vytvořený okenní systém poskytuje programátorovi jeho řízení a naprostou kontrolu. Je primárně vytvářen pro jinak Unixovou utilitu IVView, sloužící k prohlížení 3D objektů ve formátu vrml a inventorovských souborů, avšak není s ním nijak svázán a může být tedy použit prakticky kdekoliv.

Klíčová slova

API, WinAPI, Inventor, IVView, SDI, MDI, smyčka zpráv.

Poděkování

Tímto bych chtěl poděkovat Ing. Janu Pečivovi za ochotu a spolupráci při řízení a pomoci na projektu.

Abstract

This text is about tool kit which is serving for creating window systems using WinAPI. This data library provides for programmer object-oriented multi-level program interface (API) written in C++. Creating window applications in Windows ™ is easily through using this interface moreover without any knowledge of WinAPI. This way created window system provides for programmer absolute control of this system. This interface has been primary created for UNIX utility IVView. This utility is serving for viewing 3D objects in vrml format and inventor files. Window system and IVView utility are not tied up so it means that window system can be used practically anywhere.

Keywords

API, WinAPI, Inventor, IVView, SDI, MDI, message loop.

Obsah

1 Úvod

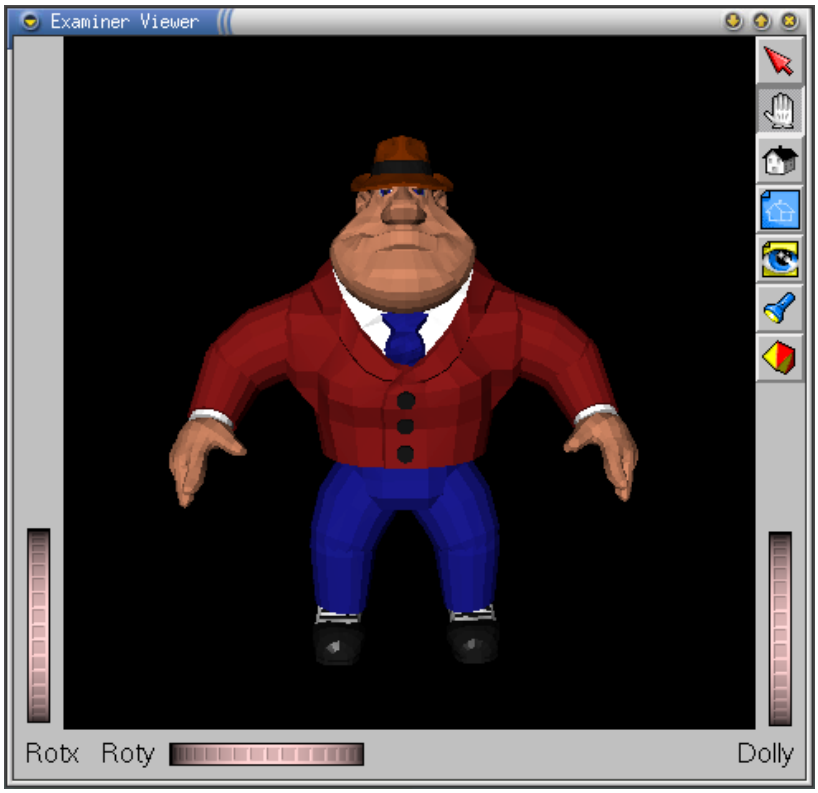
Tato práce vznikla na popud potřeby vytvořit pro programátora jednoduché aplikační rozhraní, které by mu co nejjednodušeji poskytlo možnost vytvářet aplikaci na systému Windows ™, bez nutnosti znalosti implementačních problémů při vytváření GUI. Hlavní myšlenkou projektu byla tedy jednoduchost použití rozhraní při zachování co největší funkčnosti a možností. Typickým příkladem je Inventorovská utilita IVView pro zobrazování 3D objektů. Tato utilita byla primárně vytvářena na systému Unix a vznikla potřeba ji jednoduše přepsat do systému Windows ™. Toto rozhraní to tedy umožňuje a samozřejmě nejenom této utilitě. Utilita IVView nám tak slouží jako názorný příklad pro použití této sady nástrojů.

1.1 Seznámení s utilitou IVView

Utilita IVView patří do množiny utilit Open Inventoru.

Open Inventor je populární, objektově orientovaná grafická knihovna postavená na OpenGL pro tvorbu realtimových 3D grafických aplikací. Poskytuje rozsáhlou množinu C++ tříd, využívajících vlastní OpenGL API a nabízí funkce pro vytváření a manipulaci s 3D scénou. Scény vytvářené v Inventoru jsou ukládány do souboru nazývaného graf scény, anglicky „*scene graph***“.** Tento soubor má pak koncovku .iv a může být zobrazený pomocí utilit IVView a Gview. IVView nám umožňuje zobrazovat .iv a .vrml soubory, zatímco GView nám dovoluje manipulovat se scénou pomocí grafického rozhraní. Blíže v seriálu [1].

IVView se používá jako prohlížeč pro Inventor a vrml soubory. Obsahuje možnosti pohybu s objektem a různá nastavení pohledů a zobrazení. V původní verzi je implementována pouze pod systémem Unix *viz obrázek 1.1* .a poslouží nám tedy k názornému vyzkoušení a použití vytvářené knihovny pro tvorbu okenního systému ve WinAPI a jeho implementaci pod systémem Windows ™ *viz obrázek 1.2.*



obrázek 1.1 Ukázka IVView pod Unixem

obrázek 1.2 Ukázka IVView pod Windows

2 Přehled možností a funkcí rozhraní

Vytvořené rozhraní pro vývoj okenního systému pod WinAPI umožňuje snad všechny základní funkce ohledně práce s okny aplikace a tvorby a nastavení menu oknu. Tyto dvě základní možnosti jsou logicky rozděleny do dvou implementačních souborů. Pro práci s okny a tvorbu základních dialogů je určena třída SoWindow jejíž implementace se nachází v souboru „SoWindow.cpp“ a jeho hlavičkovém souboru „SoWindow.h“. Pro práci s menu okna a odchyt událostí menu je to třída SoWinMENU umístěná v souboru „SoWinMenu.cpp“ s hlavičkovým souborem „SoWinMENU.h“.

2.1 Možnosti okenního rozhraní

Implementační část zabývající se **okenním systémem** programátorovi poskytuje:

a) Vytvoření a inicializace okna, což zahrnuje nastavení počátečních parametrů okna jako je například: počáteční velikost okna, poloha okna na obrazovce, titulek okna či celkový styl okna. Styl okna umožňuje například nastavení, kdy bude okno překreslováno, různý vzhled a chování okna (vždy navrchu, transparentnost, příslušnost k jinému oknu – dcerské okno, či povolení nebo zakázání zobrazení tlačítka pro maximalizaci a minimalizaci). Ohledně těchto možností bude pojednáno v kapitole *3.1 Okenní rozhraní*.

b) Specifikovat oknu příslušnost do tzv. skupiny MDI – „Multiple Document Interface“, nebo SDI - „Single Document Interface“. V MDI aplikaci máme hlavní okno, které obsahuje další dětská okna. MDI je označení aplikací, které umožňují uvnitř sebe sama otevřít a spravovat v jednom okamžiku větší množství dokumentů (oken). Typickým příkladem MDI aplikací jsou některé lepší textové editory, které umožňují pracovat s více dokumenty zároveň. SDI analogicky znamená rozhraní jednoho dokumentu (okna), aplikace tedy dokáže v jednom časovém okamžiku "obhospodařovat" pouze jeden jediný dokument. Problematika MDI je popsána v kapitole *3.1.3 Vytvoření MDI oken (Multiple Document Interface)*.

c) Zobrazení okna a v případě potřeby programátora následného nastavení (již za běhu) různých parametrů okna. Navržené rozhraní umožňuje například specifikovanému oknu změnu titulku, pozice, velikosti, případné omezení maximální a minimální

velkosti, popřípadě i velikosti při maximalizaci. Dále také omezení pohybu okna po obrazovce tj. statické okno či s omezeným pohybem určením maximálních a minimálních souřadnic x,y při pohybu – u SDI vzhledem k obrazovce u MDI vzhledem k rodičovskému oknu. Všechny možnosti budou blíže popsány v kapitole *3.1.4 Přehled metod okenního rozhraní*.

d) V případě potřeby jsou též programátorovi nabídnuty funkce jako je: zjištění handle okna, přiřazení vlastní funkce pro obsluhu fronty zpráv zasílaných systémem Windows, která zprávu z fronty zpráv daného procesu převezme a zpracuje ji. Mezi tyto zprávy patří například informace o vytvoření okna (zpráva WM_CREATE) , jeho ukončení (zpráva WM_CLOSE) atd. Více o frontě zpráv a jejich zpracování v kapitole *3.1.2 Zpracování zpráv Windows a smyčka zpráv*.

e) Uzavření okna.

f) Do tohoto rozhraní jsou přidány ještě funkce pro vyvolání standardních dialogů, jako je dialog pro otevření souboru, výběr barvy z palety, či změnu ukazatele myši. Tyto funkce byly přidány kvůli dosáhnutí stejné funkčnosti programu IVView pod Windows jako v Unixu.

2.2 Možnosti rozhraní pro tvorbu Menu

Implementační část zabývající se rozhraním pro **práci s menu** programátorovi poskytuje:

a) Načtení menu z resource souboru, který může být jednoduše vytvořen v tzv. WYSIWYG editoru například ve Visual Studiu. Avšak některé překladače mají problémy s podporou resource souborů, proto rozhraní nabízí funkci, která načte menu z textového souboru, který má stejnou strukturu jako resource soubor.

b) Připojení menu na vytvářené okno nebo dokonce připojit menu k již existujícímu oknu i cizí aplikace. Jelikož je rozhraní napsáno tak, že poskytuje obě možnosti, je tato funkce popsána dále v kapitole *3.2 Rozhraní pro správu menu*.

c) Odchyt události výběru položky menu a předání dále uživatelské funkci. Tyto možnosti jsou též popsány v kapitole *3.2 Rozhraní pro správu menu*.

d) Uvolnění prostředků menu ze systému.

3 Popis implementace

Postup při implementaci a zároveň i použití okenního rozhraní od vytvoření okna aplikace, přidání menu, až po vlastní obsluhu smyčky zpráv se dá logicky rozdělit do několika fází: nastavení vlastností okna, vytvoření vlastního okna, vytvoření, přiřazení a nastavení obsluhy menu, přepnutí do smyčky zpracovávání zpráv Windows, uvolnění prostředků systému.

3.1 Okenní rozhraní

3.1.1 Vytvoření a inicializace okna

Samotné vytvoření okna bez toho, aniž by uživatel požadoval nějaké speciální nastavení parametrů okna je velmi jednoduché a postačí na to pár řádků kódu. Celé rozhraní je koncipováno tak, aby jeho použití bylo co nejjednodušší a co nejvíce usnadňující práci. Z toho důvodu nemusí programátor pro vytvoření standardního systému oken téměř žádné parametry funkcí nastavovat a jsou použity standardní hodnoty. Ve srovnání s psaním čistě ve WinAPI je to nesrovnatelný rozdíl. Avšak pro pochopení a obeznámení se s možnostmi nastavovaných parametrů okna při inicializaci je vhodné, seznámit se se základními principy a postupy pro jeho vytvoření, protože kombinace možností a stylů okna jsou opravdu rozmanité.

Základní obecný postup při vytváření nového okna vypadá následovně:

1) **Registrace třídy okna:** Třída definuje určité vlastnosti, které jsou společné všem oknům, které k této třídě náleží. Některé z těchto vlastností můžeme později u jednotlivého okna změnit. Buď hned při jeho vytvoření nebo později za běhu programu. Existují některé systémové třídy, které nemusíme (a ani nemůžeme) v aplikaci registrovat a můžeme ihned vytvořit okno patřící k takové systémové třídě. Těmito třídami jsou například standardní prvky Windows jako button, list-box, edit-box apod. Každá třída, jak systémová tak registrovaná aplikací, má svoje jméno, což je řetězcová hodnota, na kterou se odkazujeme při vytváření okna patřícího této třídě.

2) **Vytvoření samotného okna a jeho zobrazení:** Jakmile máme zaregistrovanou třídu, můžeme vytvořit okno patřící této třídě. V této fázi se určují vlastnosti samotného okna jako například pozice na obrazovce či jeho velikost atd.

3) **Smyčka zpráv:** Systém Windows udržuje pro každý program, který právě běží takzvanou frontu zpráv. Při výskytu nějaké události (ať už vyvolané uživatelem nebo samotným systémem) Windows převede tuto událost na zprávu a uloží ji do fronty zpráv programu. Smyčka zpráv je pak cyklus, v němž program čeká na zprávu a po jejím přijetí ji předá do procedury okna ke zpracování, tj. jak bude na tuto událost aplikace reagovat.

Pro vytvoření okna objektu SoWindow je určena metoda

`HWND` SoWindow**::Init**(SoWindow_Class,SoWindow_Parameters)

vracející handle vytvořeného okna. Obsahuje 2 parametry určující vlastnosti vytvářeného okna.

SoWindow_Class: Což je struktura obsahující parametry potřebné pro Registrace třídy okna:.

DWORD mask	maska určující zadané parametry struktury. Pro jejich specifikaci jsou definovány konstanty WC_název_položky_struktury to celé velkými písmeny tj. například WC_LPCLASSNAME pro jejich kombinaci se používá logický součet | tzn. např. WC_LPCLASSNAME | WC_COLOR_BACKGROUND čímž řekneme, že zadáváme název třídy a barvu pozadí, které bude vyplňovat okno této třídy.
UINT style	kombinace stylů třídy. Obyčejně se používá kombinace CS_VREDRAW a CS_HREDRAW - obsah celého okna se překreslí, pokud dojde ke změně výšky či šířky okna, tedy k jeho roztažení. Přehled možností stylů třídy je obsažen v technické dokumentaci nebo v [2] *MSDN – Window Classes*.
WNDPROC lpfnWndProc	adresa procedury okna. Objekt SoWindow disponuje proměnnou SoWindow::UserWndProc, která má stejnou funkčnost, tudíž se nastavování procedury okna touto cestou nedoporučuje.
int cbClsExtra	ve speciálních případech: udává počet bytů, které chceme alokovat spolu s třídou, standardně 0.
int cbWndExtra	ve speciálních případech: alokace počtu bytů spolu s instancí programu, standardně 0.
HINSTANCE hInstance	handle instance programu.
LPSTR Id_Icon	handle ikony příslušné třídy okna, běžně získané pomocí funkce WinAPI LoadIcon či LoadImage. Můžeme také použít některou ze systémových ikon. Například takto: LoadIcon(NULL, IDI_APPLICATION)
LPSTR Id_SmallIcon	handle malé ikony (16x16) reprezentující třídu. Při NULL je použita zmenšená verze Id_Icon.
LPSTR Id_Cursor	handle kurzoru, který bude zobrazen, pokud se ukazatel myši přesune do oblasti okna patřícího k této třídě. Můžeme také použít některý ze systémových kurzorů. Například takto: LoadCursor(NULL, ICD_ARROW).

int Color_Background	definuje barvu pozadí, které bude vyplňovat okno této třídy. Například (COLOR_WINDOW+1)
LPCTSTR lpszMenuName	hlavní menu třídy okna. SoWinMENU umožňuje vytvořit menu na existující okno, tudíž tuto položku nebudeme využívat.
LPCTSTR lpszClassName	jméno identifikující naši třídu. Musí být jednoznačné v celém systému.

Pokud uživatel nechce zadávat jednotlivé parametry, má možnost využít předem nastavených standardních hodnot a to tak, že do položky struktury „mask“ přiřadí hodnotu 0. Tím se budou brát standardní hodnoty parametrů.
Bližší seznámení s možnostmi těchto parametrů viz. [3] *MSDN – WNDCLASSEX Structure*.

2 parametr metody pro vytvoření okna je:

SoWindow_Parameters	struktura obsahující parametry nově vytvářeného okna.
DWORD mask	maska určující zadané parametry struktury. Pro jejich specifikaci jsou definovány konstanty WP_název_položky_struktury to celé velkými písmeny tj. například WP_DWEXSTYLE pro jejich kombinaci se používá logický součet | tzn. např. WP_DWEXSTYLE | WP_LPWINDOWNAME čímž řekneme, že zadáváme název rozšířený styl a titulek okna.
DWORD dwExStyle	rozšířený styl. S příchodem Win32 přibylo několik nových stylů a z důvodu zpětné kompatibility kódu, kdy nebylo možné rozšířit počet parametrů funkce CreateWindow, byla vytvořena tato rozšířená funkce. Výčet možných hodnot je v nápovědě [4] *MSDN – CreateWindowEx Function*. Například hodnota WS_EX_TOPMOST udává, že okno bude vždy navrchu.
LPCTSTR lpClassName	jméno registrované třídy, ke které má okno náležet. Při tvorbě okna by měl být parametr lpClassName shodný s parametrem lpszClassName struktury SoWindow_Class.
LPCTSTR lpWindowName	titulek okna. Může být též prázdný. Pokud není zadán je nastaven na hodnotu „SoWindow“.
DWORD dwStyle	styl okna. Parametr určující vlastnosti okna, jako například zda jej lze roztahovat tažením za okraj, jakou má či nemá systémovou ikonu a pod. Standardně je nastavena hodnota na WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN, což představuje okno se všemi obecně

	používanými náležitostmi. Tedy roztažitelným okrajem, systémovými ikonami v titulkovém pruhu atd. Výčet možných hodnot je obsažen v technické dokumentaci nebo v nápovědě [4] *MSDN – CreateWindowEx Function*.
int x	x souřadnice levého horního rohu okna na obrazovce.
int y	y souřadnice levého horního rohu okna na obrazovce.
int nWidth	šířka okna
int nHeight	výška okna
HWND hWndParent	handle rodičovského okna tj. svého předka V případě hlavního okna aplikace se uvádí hodnotu NULL.
HMENU hMenu	handle hlavního menu okna. SoWinMENU umožňuje vytvořit menu na existující okno, tudíž tuto položku nebudeme využívat.

Pokud uživatel nechce zadávat jednotlivé parametry, má možnost využít předem nastavených standardních hodnot a to tak, že do položky struktury „mask“ přiřadí hodnotu 0. Tím se budou brát standardní hodnoty parametrů.
Bližší seznámení s možnostmi těchto parametrů viz. [4] *MSDN – CreateWindowEx Function*.

3.1.2 Zpracování zpráv Windows a smyčka zpráv

Po vytvoření okna pomocí metody objektu typu SoWindow pomocí

```
HWND SoWindow::Init(SoWindow_Class,SoWindow_Parameters);
```

je třeba uvést program do stavu zpracovávání fronty zpráv tj. do tzv. 11. Procedura okna starající se o zpracování fronty zpráv a reakce na ně se nastavuje přiřazením adresy procedury do proměnné objektu SoWindow::UserWndProc například takto

```
Okno.UserWndProc = WndProc;.
```

Kde Okno je objekt SoWindow a WndProc je procedura okna zpracovávající frontu zpráv definovaná takto:

WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

Obecná procedura okna vypadá následovně:

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

Jak lze vytušit, vlastní zpracování zpráv probíhá v příkazu switch, který porovnává příchozí zprávu (uMsg) se zpracovávanými (WM_DESTROY). V tomto případě zpracováváme pouze jednu zprávu a to WM_DESTROY, která udává, že okno je právě rušeno a v okamžiku doručení zprávy je již odstraněno z obrazovky. Naše reakce na ni bude nastavení návratového kódu aplikace pomocí Sinali funkce PostQuitMessage.

Měli bychom si uvědomit, že Windows berou jako okna i jednotlivé prvky jak hlavního okna tak dialogů, z toho vyplývá, že je možné posílat zprávy i těmto objektům. K účelům interakce mezi procesy jsou často využity i parametry zpráv lParam a wParam, což jsou parametry pro bližší specifikaci zprávy.

Zpráv zasílaných proceduře okna je nespočetně mnoho, proto zde uvedeme pouze ty nejdůležitější:

WM_CLOSE	je zaslána, když má být okno zavřeno.
WM_CREATE	slouží k informaci o vytvoření okna. Používá se pro inicializační operace. V parametru v lParam je ukazatel na strukturu CREATESTRUCT, která obsahuje kopie parametrů předaných funkcí CreateWindow při tvorbě okna.
WM_DESTROY	Je poslána proceduře okna poté, co bylo okno odstraněno z obrazovky.
WM_MOVE	zasílána při přesunu okna. lParam obsahuje novou pozici okna, kam má být přesunuto.
WM_SIZE	zaslána jakmile uživatel změní velikost okna. lParam obsahuje novou šířku a výšku okna.

WM_COMMAND Jedna z nejdůležitějších zpráv. Je odeslána, když uživatel zvolí položku z menu, klikne na tlačítko, stiskne akcelerátor apod. wParam specifikuje ID zvoleného objektu.

Více o zprávách, jejich zasílání a významu je k nalezení v nápovědě [5] *MSDN – Windows*.

Nyní již máme nastavenu reakci na příchozí zprávy a můžeme přistoupit k přepnutí do samotné 11. K tomu slouží metoda SoWindow::mainLoop().

Obecně tělo 11 vypadá přibližně následovně:

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

Funkce GetMessage vyjme zprávu z fronty zpráv aplikace a odešle jí k dalšímu zpracování. TranslateMessage překládá virtuální kódy zpráv klávesnice. Následující funkce DispatchMessage zprávu odešle do procedury příslušného okna ke zpracování. V případě zprávy **WM_QUIT** funkce GetMessage vrátí FALSE a smyčka zpráv je ukončena a posléze dojde k ukončení celé funkce WinMain, a tedy celé aplikace.

3.1.3 Vytvoření MDI oken (Multiple Document Interface)

Jak již bylo řečeno MDI je označení aplikací, které umožňují uvnitř sebe sama otevřít a spravovat v jednom okamžiku větší množství dokumentů (oken). MDI aplikace má hlavní okno, které obsahuje další dětská okna. Typickým příkladem MDI aplikací jsou některé lepší textové editory, které umožňují pracovat s více dokumenty zároveň.

Vytvářené rozhraní systému oken tuto vlastnost umožňují a není tedy problém vytvořit okno spravující více dceřiných oken. Problematika MDI oken je celkem složitá a zaslouží si tedy vlastní kapitolu pro osvětlení.

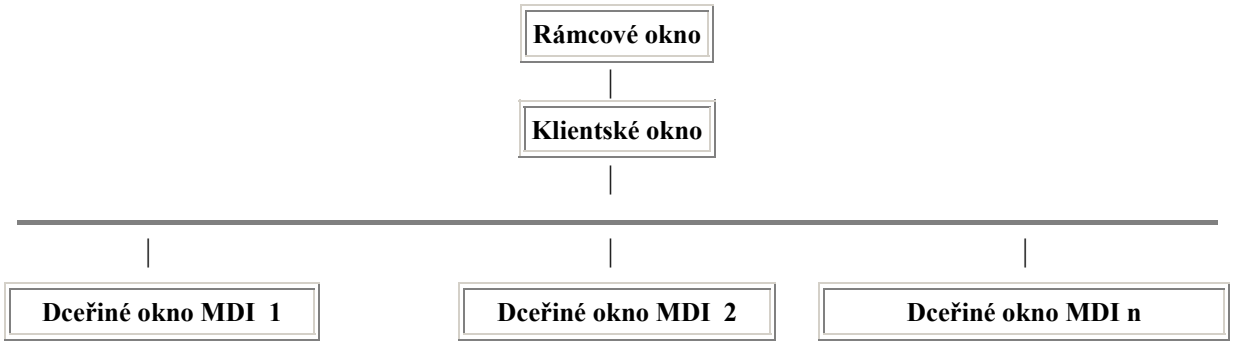
MDI aplikace má několik **charakteristických vlastností**:

- Uvnitř hlavního okna může být umístěno více dceřiných oken v klientské oblasti. Všechna tato okna jsou "zavěšena" na klientské oblasti hlavního okna.
- Při minimalizaci dceřiného okna se okno minimalizuje do levého dolního rohu klientské oblasti hlavního okna.
- Při maximalizaci hlavního okna se titulek dceřiného okna spojí s titulkem hlavního okna.

- Dceřiné okno můžeme zavřít stlačením klávesové zkratky Ctrl+F4 a přepínat se mezi dceřinými okny pomocí klávesové zkratky Ctrl+Tab.

Hlavní okno obsahující několik dceřiných oken nazýváme **rámcové okno** (frame window). Jeho klientská oblast slouží pro existenci dceřiných oken, proto název "rám". Jeho práce je o trochu komplikovanější než obyčejného okna, protože se musí zabývat koordinací rozhraní MDI.

Na řízení libovolného počtu dceřiných oken v klientské oblasti je třeba speciální okno, které se nazývá **klientské okno**. Můžeme si klientské okno představit jako průhledné okno umístěné nad celou klientskou oblastí rámcového okna.Toto klientské okno je aktuální rodič MDI dceřiných oken. Klientské okno je skutečný správce dceřiných oken MDI. Tuto strukturu demonstruje obrázek *Struktura MDI aplikace 3-1*.



obrázek 3.13-1 struktura MDI aplikace.

Pro vytvoření MDI okna poskytuje objektu SoWindow metodu

```
HWND SoWindow::CreateNewMDIChild (
        char szTitle[]="Untitled",
        BOOL AddChildMenu=TRUE,
        HMENU hChildMenu=NULL,
        void (*ChildWndProc)(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM
        lParam)=NULL,
        int style=MDIS_ALLCHILDSTYLES
        );
```

Jak je názvy parametrů napovídají možnosti nastavení MDI okna jsou velice podobné jako u obyčejného okna.

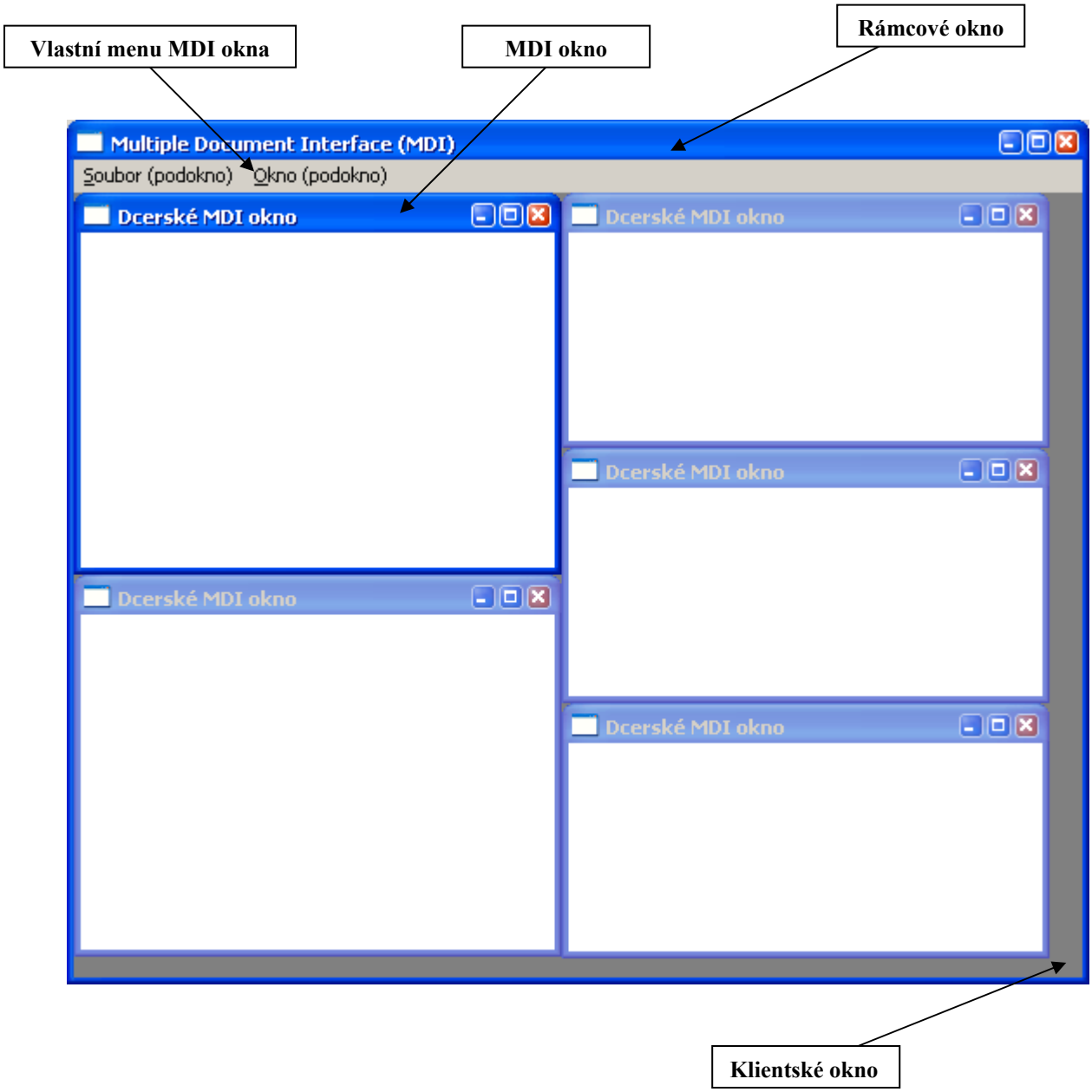
Vysvětlení parametrů:

char szTitle[]	nastavení titulku okna
BOOL AddChildMenu	TRUE => přidá do menu MDI okna položku Windows, obsahující nástroje pro usnadnění práce s MDI okny tj. kaskádování oken, rozdělení klientské oblasti rámce jednotlivým oknům, přepínání mezi okny atd.
HMENU hChildMenu	handle menu, které má být přiřazeno tomuto MDI oknu. Pro vytvoření menu MDI okna lze též použít metody objektu SoWinMENU: HMENU SoWinMENU::CreateMDIMenu(int,void (*)(int)).
ChildWndProc	procedura okna pro zpracování zpráv.
int style	parametr stylu okna. Stejné jako u vytváření obyčejného SDI okna.

Ucelený postup pro vytvoření MDI okna je tedy následující:

1) Vytvoříme obyčejné SDI okno viz. kapitola *3.1.1 Vytvoření a inicializace okna.*
2) Pomocí metody SoWindow::CreateNewMDIChild(..) vytvoříme MDI okno v klientské oblasti hlavního okna.

Celý postup je velice jednoduchý a neukládá uživateli žádné zvláštní požadavky.



obrázek 3.13-2 struktura MDI aplikace v reálu.

Pro vytvořená MDI okna jsou k dispozici metody stejné jako pro práci s oknem obyčejným, v jejich názvu se navíc objevuje zkratka MDI.

Každé takto vytvořené okno může mít vlastní přiřazené menu. U MDI oken je menu součástí rámcového okna, takže menu konkrétního MDI okna je jakoby skryto, dokud není toto okno aktivní. Na tento problém je v okenním rozhraní myšleno a přiřazení menu MDI oknu se provádí buď parametrem hChildMenu ve funkci SoWindow::CreateNewMDIChild. Potom by se odchyt událostí

menu prováděl v proceduře MDI okna specifikované parametrem ChildWndProc. Nebo může být použita jednodušší varianta a to metoda třídy SoWinMENU:
HMENU CreateMDIMenu(int,void (*)(int)), či její alternativa
HMENU CreateMDIMenufromFile(char *Menu_file,void (*CALL)(int)).
Zde je jako parametr udávána adresa uživatelské procedury zpracovávající výběr položky menu.

3.1.4 Přehled metod okenního rozhraní

Jak již bylo řečeno okenní rozhraní reprezentuje třída SoWindow, umožňující vytvoření jak SDI tak i MDI okna. Ovšem pokud by rozhraní poskytovalo pouze tyto možnosti, chyběla by možnost komplexnějšího dotváření celkového vzhledu a funkčnosti celé aplikace. Proto je doplněna možnost nastavení vlastní procedury okna a zpracování fronty zpráv u SDI i MDI oken, změna pozice okna, nastavení jeho zobrazení, vlastnosti při pohybu atd.

HWND **SoWindow::Init** (SoWindow_Class,SoWindow_Parameters)

- viz kapitola *3.1.1 Vytvoření a inicializace okna*

void **SoWindow::Show**(int)

- zobrazení okna. Parametr udává jakým způsobem má být okno zobrazeno.
  Standardní hodnota -1 tj. SW_SHOWNORMAL.
  viz [6] *MSDN – ShowWindow Function*.

int **SoWindow::mainLoop**()

- viz kapitola *3.1.2 Zpracování zpráv Windows a smyčka zpráv*.

void (***UserWndProc**)(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

- pokud si uživatel přeje vlastní obsluhu fronty zpráv, přiřadí do této proměnné adresu procedury SDI okna. Tato možnost je velice vhodná v případě, kdy chce mít uživatel naprostou kontrolu nad oknem, použít ho k jinému účelu, či například omezit manipulaci uživatele aplikace s oknem apod.

BOOL **SoWindow::SetMinMaxInfo**(int MaxSizeX=-1, int MaxSizeY=-1,
int MaxPositionX=-1, int MaxPositionY=-1,
int MaxTrackSizeX=-1,int MaxTrackSizeY=-1,
int MinTrackSizeX=-1,int MinTrackSizeY=-1)

- slouží k omezení velikosti okna.

- MaxSize určuje rozměry, které bude mít okno, když bude maximalizované.
- MaxPosition určuje polohu levého horního rohu takto omezeného maximalizovaného okna.
- TrackSize určuje minimální a maximální roztažitelné rozměry okna tj. odkud kam lze okno roztáhnout.

Pokud si uživatel nepřeje zadávat některou z těchto hodnot nastaví -1.

BOOL **SoWindow::SetWindowTitle**(LPCTSTR Title)

- slouží k nastavení titulku okna.

BOOL **SoWindow::SetWindowPos**(HWND hWndInsertAfter,int X,int Y,int cx,int cy,UINT uFlags)

- k nastavení pozice okna na obrazovce.

- HWND hWndInsertAfter určuje pořadí okna vůči okolí. Například HWND_TOPMOST je vždy navrchu.
- int X,int Y souřadnice levého horního rohu okna
- int cx,int cy specifikuje výšku a šířku
- UINT uFlags specifikuje příznak použití parametrů. Například při použití SWP_NOSIZE jsou ignorovány parametry šířky a výšky.

BOOL **SoWindow::SetStaticWindowPos**(int X=-1,int Y=-1)

- k nastavení statické polohy okna.

- int X,Y souřadnice levého horního rohu okna.

HWND **SoWindow::GetHWND**()

- vrací hodnotu handle okna

BOOL **SoWindow::Close**()

BOOL **SoWindow::Close**(HWND hWindow)

- uzavře okno aplikace.

BOOL **SoWindow::SetMinMaxWindowPos** (int max_x=-1,int max_y=-1,
int min_x=-1,int min_y=-1)

- nastaví maximální a minimální hodnotu polohy okna vzhledem k obrazovce.

int max_x,max_y,min_x,min_y hodnoty maximálních a minimálních souřadnic okna

Pokud si uživatel nepřeje zadávat některou z těchto hodnot nastaví -1.

HWND **SoWindow::CreateNewMDIChild**(char szTitle[]="Untitled",
BOOL AddChildMenu=TRUE,
HMENU hChildMenu=NULL,
void (*ChildWndProc)(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)=NULL,
int style=MDIS_ALLCHILDSTYLES);

- viz *kapitola 3.1.3 Vytvoření MDI oken (Multiple Document Interface)*

BOOL **SoWindow::SetMDIMinMaxInfo** (HWND hWindow,
int MaxSizeX=-1,int MaxSizeY=-1,
int MaxPositionX=-1,int MaxPositionY=-1,
int MaxTrackSizeX=-1,int MaxTrackSizeY=-1,
int MinTrackSizeX=-1,int MinTrackSizeY=-1)

- slouží k omezení velikosti MDI okna. Má stejné parametry jako předchozí metoda, je však doplněna o parametr hWindow, který specifikuje okno.

void **SoWindow::AddMDIFrameStandartdMenu**(void)

- připojí k již existujícímu (pokud neexistuje vytvoří jej) menu položku Windows, obsahující nástroje pro usnadnění práce s MDI okny tj. kaskádování oken, rozdělení klientské oblasti rámce jednotlivým oknům, přepínání mezi okny atd. Tato možnost má stejný efekt jako nastavení parametru hChildMenu v metodě SoWindow::CreateNewMDIChild.

BOOL **SoWindow::SetMDIWindowTitle**(HWND hWindow,LPCTSTR Title)

- slouží k nastavení titulku MDI okna, specifikovaného parametrem hWindow.

BOOL **SoWindow::SetMDIWindowPos**(HWND hWindow,
HWND hWndInsertAfter,
int X,int Y,
int cx,int cy,
UINT uFlags)

- obdoba SetWindowPos, okno je specifikováno parametrem hWindow.

BOOL **SoWindow::SetMDIMinMaxWindowPos**(HWND MDIhandle=NULL,
int max_x=-1,int max_y=-1,
int min_x=-1,int min_y=-1)

- obdoba SetMinMaxWindowPos, okno je specifikováno parametrem hWindow.
- Souřadnice jsou zde však brány vzhledem ke klientské oblasti rámcového okna.

BOOL **SoWindow::SetMDIStaticWindowPos**(HWND MDIhandle=NULL,int X=-1,int Y=-1)

- obdoba SetStaticWindowPos, okno je specifikováno parametrem hWindow.

HWND **SoWindow::GetMDIActiveHWND**();

- vrátí handle právě aktivního MDI okna. Vhodné pro použití v některých metodách MDI oken k získání parametru hWindow.

Součástí okenního rozhraní jsou také funkce pro vyvolání některých standardních dialogů, které byly potřebné pro kompletní funkčnost utility IVView pod systémem Windows.

BOOL **Open_Dialog**(LPTSTR lpFile, HWND hwndOwner,const char * files)

- otevře standardní dialog pro otevření souboru a vrátí TRUE při zdárném vybrání souboru. Pokud se nepodařilo soubor vybrat vrací FALSE.

- LPTSTR lpFile v parametru lpFile vrací název vybraného souboru.
- HWND hwndOwner handle okna, které má být rodičem dialogu pro otevření souboru.
- const char * files maska pro vybírané soubory.

BOOL **Save_Dialog**(LPTSTR lpFile, HWND hwndOwner,const char * files)

- otevře standardní dialog pro uložení souboru a vrátí TRUE při zdárném zadání. Má stejné parametry jako Open_Dialog, jen v parametru lpFile vrací název a cestu k zadanému souboru.

BOOL **Open_ColorDialog**(DWORD * col, HWND hwndOwner)

- otevře standardní dialog pro výběr barvy a vrátí TRUE při zdárném vybrání.

- DWORD * col ukazatel na hodnotu obsahující vybranou barvu typu DWORD ve formátu RRGGBB.
- HWND hwndOwner handle okna, které má být rodičem dialogu pro výběr barvy.

void **setBusyCursor**(BOOL showBusy)

- nastaví ukazatel myši na přesýpací hodiny či na normální ukazatel podle parametru BOOL showBusy.

  showBusy=TRUE – přesýpací hodiny.

  showBusy=FALSE – normální ukazatel.

void **OpenDokument**(const char * FullPath)

- Pracuje jako příkazový řádek, který provede příkaz zadaný parametrem FullPath.
- Vhodné pro otevírání různých typů dokumentů apod. Systém totiž sám vybere asociovaný program pro daný typ dokumentu.

3.2 Rozhraní pro správu menu

Pro plnohodnotnou práci bylo okenní rozhraní rozšířeno o práci s menu okna. Pro práci s menu okna je implementována třída SoWinMENU. Postup práce s menu se dá logicky rozčlenit do fáze načtení menu, jeho vytvoření, přiřazení k oknu a odchyt událostí menu.

3.2.1 Načtení menu

Typickým způsobem načítání menu bývá z resource souboru. Jeho vytvoření je např. ve Visual Studiu otázkou několika chvil a je tzv. WYSIWYG. Bohužel se některé překladače potýkají s problémy zpracování resource souborů a tak bylo rozhraní doplněno o možnost načítání menu z textového souboru.

Takovýto textový soubor má pevně danou strukturu, která je velice podobná jako struktura v resource souboru, použití v tomto případě s sebou však nenese potřebu kompilace resource souborů.

Popis struktury textového souboru s menu:

```
Ukázkové_Menu MENU
BEGIN
    POPUP "Soubor"
    BEGIN
        MENUITEM "Nový",        40001
        MENUITEM "Otevřít",     40002
        MENUITEM "Konec",       40003
    END
    POPUP "Nastavení"
    BEGIN
        MENUITEM "Barva",       40004
        POPUP "Filtr"
        BEGIN
          MENUITEM "Rozpustit",40005
          MENUITEM "Sépie",     40006
        END
        MENUITEM "O aplikaci",  40007
    END
END
```

Struktura tohoto menu je téměř shodná se strukturou menu, které by bylo vytvořeno pomocí resource editoru. Jediný rozdíl by byl v tom, že ID položky by nebyla číselná hodnota, ale název konstanty nesoucí tuto hodnotu.

Před klíčovým slovem MENU je uveden název menu. Začátek celého menu je uvozen klíčovým slovem BEGIN a ukončen END. Tato dvě slova se používají též k vymezení bloku položek menu patřících do stejné skupiny, která nese za název obsah řetězce následujícího po klíčovém slovu POPUP. POPUP tedy určuje rozbalitelnou položku menu, jejíž název je uveden bezprostředně za slovem POPUP. V bloku patřícím do jisté rozbalitelné položky může být uvedena další rozbalitelná položka menu. Takovéto zanoření může být do libovolné hloubky. V rozbalitelné položce se může též nacházet již konkrétní položka menu, která je uvozena klíčovým slovem MENUITEM. Její název je uveden bezprostředně za ním. Pro identifikaci položky se používá číselný identifikátor typu integer, který se uvádí po čárce za názvem položky viz. popis struktury menu uvedený výše.

Menu uvedené v popisu struktury by v praxi vypadalo následovně:

Nyní když je menu načteno je nutné ho přiřadit oknu. Jsou dva způsoby jak budeme nakládat s vytvořeným menu. Buď vytvořené menu připojíme k již existujícímu oknu. Tuto možnost používáme při přidávání menu samostatnému SDI oknu. MDI tuto možnost také využívá, ale nelze přiřazovat oknu úplně cizí aplikace. Nebo menu přiřadíme oknu již při jeho vytváření a tak si ušetříme práci při odchytu událostí menu.

Skutečnost, že přiřazujeme menu již vytvořenému oknu, má dvě s sebou nesoucí fakta.

1) Menu může být přiřazeno kterémukoliv již existujícímu oknu. Což je jistě velká výhoda.
2) Pro zpracování událostí menu se musí použít tzv. hookování tj. odchyt událostí, což velice znepříjemňuje implementaci. Například tím, že při odchytu zpráv se musí rozeznávat, od kterého okna a jeho menu tato zpráva přišla a kam tuto skutečnost předat. Tyto problémy však za uživatele nesou navržené metody třídy SoWinMENU.

U menu v MDI oknech je zase jiný problém způsobený tím, že menu MDI okna je zobrazeno v liště rámcového okna a tím pádem se musí při aktivaci jiného okna toto menu zrušit a překreslit odpovídajícím menu právě aktivního okna.O tyto problémy se však uživatel nestará.

3.2.2 Přehled metod rozhraní pro správu menu

Hlavní součástí jsou metody pro načtení menu, jeho přiřazení oknu a předání události menu programátorovi, na kterém již pak je zpracování a reakce aplikace na konkrétní výběr položky.

void **SoWinMENU::Init**(HWND Window)

- provede inicializaci menu na uvedené okno, zadané pomocí handle. Pro každé menu musí být tato metoda volána jako první.

HMENU **SoWinMENU::CreateMenu**(int,void (*)(int))

- Vytvoří menu z resource souboru, které je zadané identifikátorem menu, což je první parametr.
- Při události výběru položky menu je volána funkce, která je zadána jako druhý parametr. Tato funkce je uživatelská funkce, která má jako parametr integer, což je identifikátor položky tj. ID položky menu. Obsahem této funkce bývá switch testující hodnotu ID, podle níž si uživatel specifikuje akci na výběr položky.

HMENU **SoWinMENU::CreateMenufromFile**(char *Menu_file,void (*CALL)(int))

- Obdobná funkce jako SoWinMENU::CreateMenu, pouze je za zdroj menu brán soubor, který je uveden v prvním parametru. Tato metoda používá pro načtení menu privátní metodu HMENU ReadMenuFromFile(char *Menu_file), která po zdárném načtení vrací handle vytvořeného menu, jinak vrací NULL. Tento soubor má pevně danou strukturu viz kapitola *3.2.1Načtení menu.*

HMENU **SoWinMENU::CreateMDIMenu**(int,void (*)(int))

- Metoda velice podobná SoWinMENU::CreateMenu avšak aplikovatelná na MDI okno. Okno, na které má být aplikována, je určeno již v předchozím volání SoWinMENU::Init.

HMENU **SoWinMENU::CreateMDIMenufromFile**(char *Menu_file,void (*CALL)(int))

- Metoda aplikovatelná na MDI okno se stejným chováním jako SoWinMENU::CreateMenufromFile.

HMENU **SoWinMENU::GetHMENU**()

- Vrátí handle menu.

BOOL **SoWinMENU::ToggleMenuItemChecked**(int)

- Zamění zaškrtávací políčko u položky menu zadané pomocí ID v parametru metody. Pokud položka není zaškrtnuta, zaškrtne ji a pokud není, odškrtne ji.

void **SoWinMENU::Close**()

- Uvolní paměť po menu.

void **SoWinMENU::CloseAll**();

- Uvolní paměť po všech menu.

HMENU **SoWinMENU::ReadMenuFromFile**(char *Menu_file)

- Privátní metoda SoWinMENU sloužící při načítání menu z textových souborů. Tato metoda otevře pro čtení zadaný soubor (je zadáván i s cestou k souboru) a pomocí konečného automatu postupně načítá položky menu a nakonec jej vytvoří a vrátí jeho handle.

4 Ukázka praktického použití rozhraní

Nyní si konečně uvedeme praktické použití vytvářeného rozhraní. Pro tyto účely použijeme již zmiňovanou utilitu IVView jinak implementovanou na systému Unix. Ukážeme si vytvoření samostatného okna s menu i vytvoření MDI oken s vlastními menu.

4.1 SDI okno s IVView

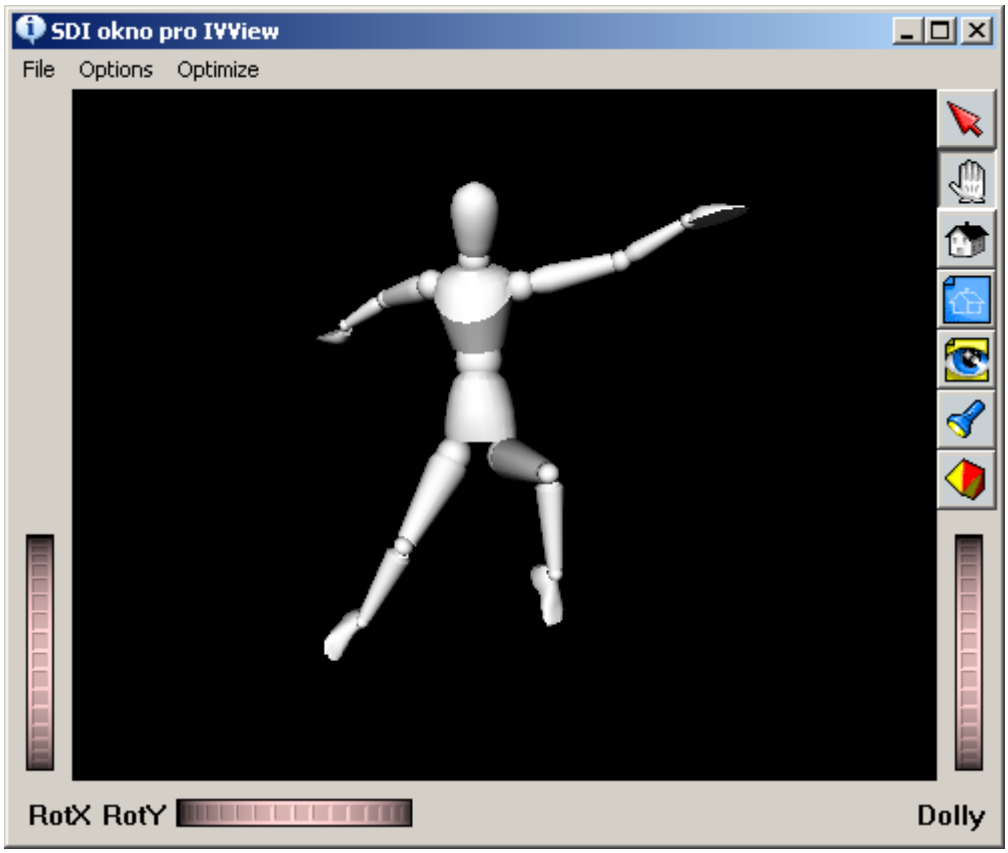
Při použití vytvořeného API rozhraní pro vytvoření standardního okna bez jakýchkoliv změn stylů apod. postačuje opravdu pár řádku kódu.

Ukázka kódu pro SDI okno s IVView:

```
SoWindow Okno;  //instance třídy SoWindow
SoWinMENU Menu; //instance třídy SoWinMENU

 SoWindow_Class WinClass;
  SoWindow_Parameters WinParam;
  WinClass.mask=0;      //ponechání standardních parametrů pro třídu okna
  WinParam.mask=0;      //ponechání standardních parametrů pro inicializaci okna

  //inicializace okna
  HWND handle = Okno.Init(WinClass,WinParam);
  //nastavení titulku okna (je možno docílit i nastavením lpWindowName ve struktuře
  SoWindow_Parameters a přiřazením do masky konstanty WP_LPWINDOWNAME)
  Okno.SetWindowTitle("SDI okno pro IVView");
```

```
  //inicializace menu na naše okno
  Menu.Init(handle);
  //načtení a přiřazení menu našemu oknu a nastavení procedury menu okna
  Menu.CreateMenufromFile("menu.txt",Obsluha_Menu);
  //zobrazení okna
….
…Inicializace samotného IVVIEW
..
  Okno.Show(-1);
  // smyčka zpráv
  Okno.mainLoop();
```

Ukázka výsledné aplikace:



obrázek 4.1-1 ukázka výsledné SDI aplikace

4.2 MDI okna s IVView

Nyní si ukážeme vytvoření MDI oken s IVView a přiřazení jim vlastních menu.

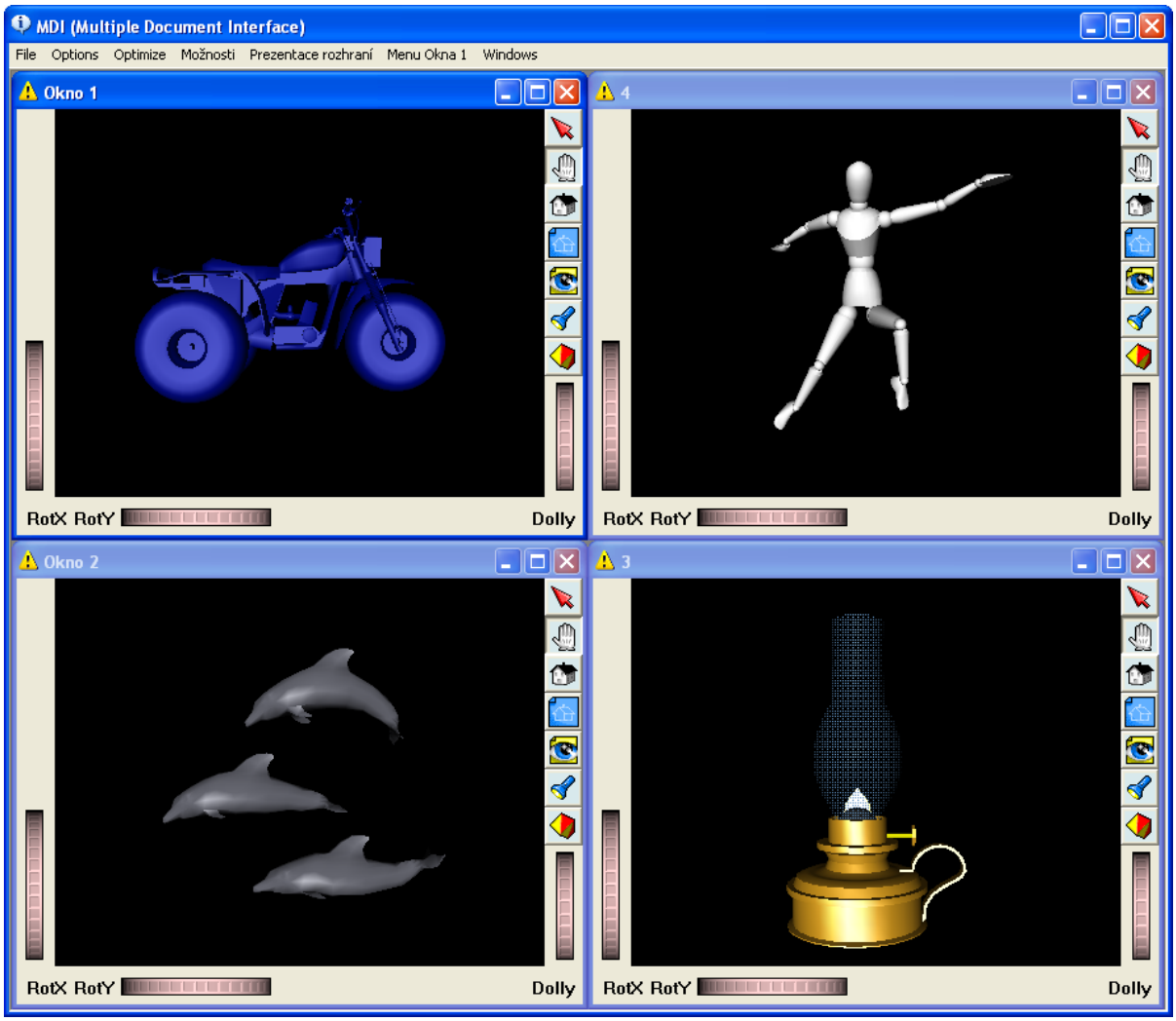
Ukázka kódu pro MDI okna s IVView:

```
SoWindow Frame; //instance třídy SoWindow
SoWinMENU Menu1, Menu2, Menu3, Menu4; //instance třídy SoWinMENU
SoWindow_Class WinClass;
SoWindow_Parameters WinParam;

 WinClass.lpszClassName="MDI class";  //nastavení vlastního názvu třídy okna
 WinClass.mask=WC_LPSZCLASSNAME;      //vymaskování parametru lpszClassName

 WinParam.lpClassName="MDI class";    //okno bude patřit do třídy MDI class
 WinParam.mask=WP_LPCLASSNAME;        // vymaskování parametru lpClassName

 HWND window = Frame.Init(WinClass,WinParam); //inicializace okna

 HWND MDIOkno_1= Frame.CreateNewMDIChild("1"); //vytvoření nového MDI okna
 HWND MDIOkno_2= Frame.CreateNewMDIChild("2"); //vytvoření nového MDI okna
 HWND MDIOkno_3= Frame.CreateNewMDIChild("3"); //vytvoření nového MDI okna

 /*vytvoření nového MDI okna s přidáním položky Windows do menu a nastavením vlastní
 procedury okna*/
 HWND MDIOkno_4= Frame.CreateNewMDIChild("4",true,NULL,MyChildProces);

 //změna titulků jednotlivých MDI oken
 Okno1.SetMDIWindowTitle(MDIOkno_1,"Okno 1");
 Okno1.SetMDIWindowTitle(MDIOkno_2,"Okno 2");
 //změna titulku Frame okna
 Frame.SetWindowTitle("MDI (Multiple Document Interface)");

 //inicializace menu na jednotlivá MDI okna
 Menu1.Init(MDIOkno_1);
 //načtení a přiřazení menu našemu MDI oknu a nastavení procedury menu okna
 HMENU hMenumain = Menu.CreateMDIMenufromFile("menu1.txt",processTopbarEvent1);

 Menu2.Init(MDIOkno_2);
 Menu2.CreateMDIMenufromFile("menu2.txt",processTopbarEvent2);

 Menu3.Init(MDIOkno_3);
 Menu3.CreateMDIMenufromFile("menu3.txt",processTopbarEvent3);

 Menu4.Init(MDIOkno_4);
 Menu4.CreateMDIMenufromFile("menu4.txt",processTopbarEvent4);
 ….
…Inicializace samotného IVVIEW
 Frame.Show(SW_MAXIMIZE);
 Frame.mainLoop(); // smyčka zpráv
```

Slovní popis kódu:

Tento příklad se od předchozího příliš neliší. Na začátku při inicializaci je přidána do parametrů tvorby okna změna třídy okna a provede se vytvoření okna stejně jako u předešlého příkladu. Nyní jsou vytvořena nová MDI okna přiřazená do již vytvořeného okna pomocí metody HWND SoWindow::CreateNewMDIChild. V prvních třech volání této metody je uveden jako parametr pouze titulek nového okna. V posledním případě je však parametrů více určujících nejen titulek okna, ale i přidání položky Windows do menu (hodnota TRUE je implicitní) a nastavením vlastní procedury okna (MyChildProces). Dále je zde uvedena opětovná změna titulků MDI oken a následně vytvoření menu MDI oken pomocí HMENU CreateMDIMenufromFile(char *Menu_file,void (*CALL)(int)). Následně pak opět inicializace IVViewerů na jednotlivá MDI okna, zobrazení okna (maximalizovaně) a přepnutí do smyčky zpráv.

Ukázka výsledné aplikace:



obrázek 4.2-1 ukázka výsledné MDI aplikace

5 Závěr

Výsledný celek poskytuje poměrně silný nástroj na vytvoření okenní aplikace v systému Windows a její naprostou programátorskou kontrolu.
Celkový koncept řešení okenního systému ve WinAPI byl implementován s hlavním důrazem na srozumitelnost a jednoduchost použití rozhraní bez zvláštních znalostí programování ve WinAPI. Většina metod poskytuje možnost ponechání standardních, již přednastavených, hodnot parametrů.

Ukázka implementace byla provedena na utilitě IVView, což ale neznamená, že nelze použít na jakoukoliv jinou aplikaci. Mezi první aplikace, které se nabízejí, patří zbylé utility inventoru jako například SceneViewer či GView.

Praktické využití tohoto rozhraní nabízí rozsáhlé možnosti. Mezi přední bych uvedl aplikace jako IVView, kdy došlo k situaci, že implementace samotné činnosti byla již známá a šlo tedy pouze zasadit tuto funkčnost do okenního systému pod Windows.

Jako další zásadní možné pokračování této práce by byla implementace podobného okenního rozhraní na systému Unix, což by dělalo z této sady opravdu šikovný nástroj pro tvorbu okenního systému.

Literatura

[1] PEČIVA, Jan. *Open Inventor* [online]. 2003 [cit. 2006-04-18]. Seriál na ROOT.CZ. Dostupný z WWW: <http://www.root.cz/clanky/open-inventor/>.

[2] *MSDN – Window Classes* [online]. © Microsoft Corporation , c2006 , poslední změna 01.01.2006 [cit. 2006-04-18]. Dostupný z WWW: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/windowclasses/aboutwindow.asp>.

[3] *MSDN – WNDCLASSEX Structure* [online]. © Microsoft Corporation , c2006 , poslední změna 01.01.2006 [cit. 2006-04-18]. Dostupný z WWW: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/windowclasses/windowclassreference/windowclassstructures/wndclassex.asp>.

[4] *MSDN – CreateWindowEx Function* [online]. © Microsoft Corporation , c2006 , poslední změna 01.01.2006 [cit. 2006-04-18]. Dostupný z WWW: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/windows/windowreference/windowfunctions/createwindowex.asp>.

[5] *MSDN – Windows* [online]. © Microsoft Corporation , c2006 , poslední změna 01.01.2006 [cit. 2006-04-18]. Dostupný z WWW: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/windows.asp>.

[6] *MSDN – ShowWindow Function* [online]. © Microsoft Corporation , c2006 , poslední změna 01.01.2006 [cit. 2006-04-18]. Dostupný z WWW: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/windowing/windows/windowreference/windowfunctions/showwindow.asp>.

Příloha

Popis obsahu CD

Na přiloženém CD je k dispozici elektronická podoba tohoto dokumentu v souborech Bakalářská práce.doc (Dokument Microsoft Word) a Bakalářská práce.pdf (Portable Document Format). Dále jsou přiloženy zdrojové texty vytvářeného rozhraní, prezentace obsahující ukázku jeho použití, aplikace na zmiňovanou utilitu IVView, již zkompilované soubory a knihovny potřebné ke spuštění aplikace a technická dokumentace v podobě webových stránek a ve formátu chm (Microsoft Compiled HTML).

Technická dokumentace

V technické dokumentaci jsou uvedeny a popsány všechny metody poskytované vytvořeným rozhraním. Na CD je ve formátu webových stránek (soubor index.html) nebo ve formátu Microsoft Compiled HTML (soubor dokumentace.chm).

Strom obsahu CD