Folosirea noilor ferestre Windows 2000 în aplicațiile Delphi

– Marian Vețeanu

Deși creat ca succesor al sistemului *NT*, *Windows 2000* implementează pe lângă evidentele servicii administrative și de rețea, și cea mai productivă interfață grafică comparativ cu celelalte sisteme create de *Microsoft* până în prezent.

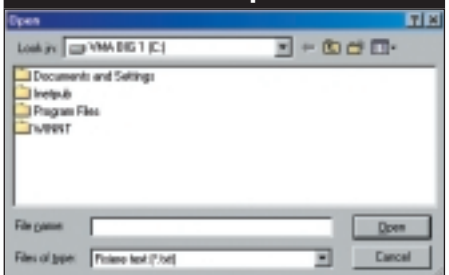
Cum era și normal, *Windows 2000* suportă toate elementele GUI introduse în sistemele precedente, la care adaugă o serie nouă de proprietăți. Dacă pentru utilizatori această nouă înfățișare a lui *W2K* nu le poate aduce decât o senzație mai plăcută în lucrul cu calculatorul, pentru programatori apare acum o adevărată problemă: portarea vechilor programe pe noul sistem în conformitate cu specificațiile *Microsoft*, și asta în lipsa unor noi medii de dezvoltare certificate *Windows 2000*.

Prin articolul curent nu se dorește oferirea de soluții la toate problemele legate de acest subiect, ci doar se încearcă prezentarea câtorva noutăți aduse de *W2K* în materie de interfețe utilizator, precum și implementarea acestora folosind mediul *Borland Delphi 5.0*.

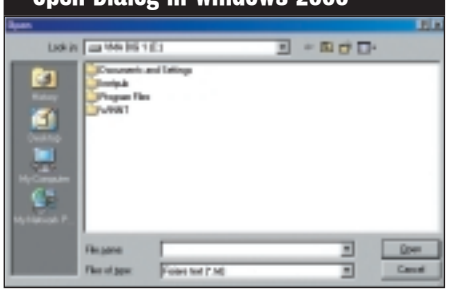
Subiectele tratate în cadrul acestui articol:

1. Noile ferestre *Open/Save Dialog Box*
2. Ferestre stratificate (*Layered Windows*) și Alpha Blending
3. Ferestre animate

Fereastra clasică Open



Open Dialog în Windows 2000



1. Noile dialoguri Open / Save din Windows 2000

Vom începe prezentarea cu noile ferestre de acces la fișiere : *Open* și *Save*. După cum se observă și din figura următoare există o oarecare deosebire între acestea și fereastrele clasice din *Windows 9x*.

Probabil că utilizatorii *Office 2000* sunt deja familiarizați cu noul tip de dialoguri, chiar dacă au un sistem de operare mai vechi.

Afișarea unei ferestre de tip *Open* se face prin apelul funcției API *GetOpenFileName* conținută de biblioteca *commdlg32.dll*, și care are următorul prototip :

```
BOOL GetOpenFileName(
  LPOPENFILENAME lpofn // pointer
   către o structură cu date de ini-
   țializare
);
```

Asemănătoare cu aceasta este și funcția *GetSaveFileName* care realizează afișarea dialogului *Save as…*

```
BOOL GetSaveFileName(
  LPOPENFILENAME lpofn // pointer
   către o structură cu date de ini-
   țializare
);
```

Ambele funcții solicită un singur parametru de tip pointer către o structură de date ce conține informații pentru inițializarea ferestrelor amintite și în care se întoarce de asemenea starea ferestrei după selecția utilizatorului (vezi figura „Structura OPENFILENAME”).

Structura de mai sus este aproape identică cu cea folosită de funcțiile *GetOpenFileName* și *GetSaveFileName* în versiunile mai vechi ale sistemului de operare cu excepția câtorva câmpuri, dintre care cel mai interesant fiind *FlagsEx*. Implicit noul tip de ferestre afișează în partea stângă bara cu butoane numită *Places Bar*, care însă poate fi ascunsă dacă în acest câmp se pune constanta `OFN_EX_NOPLACESBAR`.

Și acum să vedem modul de integrare în aplicațiile Delphi a noilor tipuri de ferestre. După cum probabil știți, Delphi implementează cu ajutorul clasei *TOpenDialog* fereastra din figura „Fereastra clasică Open”. Din păcate nu se poate folosi această clasă și în scopul obținerii de ferestre W2K datorită faptului că *TOpenDialog* ascunde structura de date ce se transmite funcției API.

Așadar va trebui să construim o nouă clasă, moștenită din *TOpenDialog*, pentru atingerea scopului propus. Dacă ne aruncăm o privire în sursa unitului *commdlg.pas* (în mod normal se găsește în C:\Program Files\Borland\Delphi5\Source\Rtl\Win\) observăm modul în care cei de la Borland trimit structura de mai sus către funcția API, și anume prin folosirea unei variabile de tipul **var** în locul folosirii unui pointer creat cu operatorul @. Acest lucru ne obligă să redefinim codul pascal ce îmbracă funcția *GetOpenFileName*, deoarece compilatorul nu ne dă voie să modificăm tipul variabilei **var**. Vom folosi în acest sens următoarea declarație:

```
function GetOpenFileNameEx(var Open-
   File: TOpenFilenameEx): Bool;
   stdcall;
external 'comdlg32.dll'  name
   'GetOpenFileNameA';
```

Structura OPENFILENAME

```
typedef struct tagOFN {
  DWORD          lStructSize;
  HWND           hwndOwner;
  HINSTANCE      hInstance;
  LPCTSTR        lpstrFilter;
  LPTSTR         lpstrCustomFilter;
  DWORD          nMaxCustFilter;
  DWORD          nFilterIndex;
  LPTSTR         lpstrFile;
  DWORD          nMaxFile;
  LPTSTR         lpstrFileTitle;
  DWORD          nMaxFileTitle;
  LPCTSTR        lpstrInitialDir;
  LPCTSTR        lpstrTitle;
  DWORD          Flags;
  WORD           nFileOffset;
  WORD           nFileExtension;
  LPCTSTR        lpstrDefExt;
  LPARAM         lCustData;
  LPOFNHOOKPROC lpfnHook;
  LPCTSTR        lpTemplateName;
#if (_WIN32_WINNT >= 0x0500)
  void *         pvReserved;
  DWORD          dwReserved;
  DWORD          FlagsEx;
#endif // (_WIN32_WINNT >= 0x0500)
} OPENFILENAME, *LPOPENFILENAME;
```

Acestea fiind stabilite trecem la implementarea propriu-zisă a noii clase, pe care o numim *TW2KOpenDialog* :

```
type TW2KOpenDialog =
   class(TOpenDialog)
  public
    ShowPlacesBar:boolean;
    constructor Create(AOwner:
   TComponent); override;
    function Execute: Boolean;
   override;
  protected
    FInterceptor : Pointer;
    function IsWin2000 : boolean;
  end;
```

ShowPlacesBar controlează afișarea sau ascunderea zonei *Places bar*. Această variabilă trebuie setată de utilizator înainte de apelarea metodei *Execute*. Pentru a fii siguri că se încercă afișarea unei ferestre de *Windows 2000* doar pe sisteme cu *Windows 2000,* vom verifica tipul și versiunea sistemului de operare cu ajutorul funcției *IsWin2000* care întoarce *true* în cazul prezenței acestui sistem.

Implementarea noii clase o vom face într-un unit numit *w2kdialogs.pas* a cărui sursă poate fi descărcată de la www.netreport.ro.

Suplimentar în acest unit am implementat și noua versiune a ferestrei *Save as…* prin intermediul unei clase moștenite din cea precedentă:

```
type TW2KSaveDialog =
   class(TW2KOpenDialog)
  public
    constructor Create(AOwner:
TComponent); override;
  end;
```

Tot ce ne mai rămâne de făcut este să creăm un mic program de test prin care să verificăm funcționarea corectă a celor două clase.

Acest mic program nu prezintă nimic spectaculos în realizare și ca urmare vom lista doar codul sursă al lui. Unit-ul principal (test w2kdialogs.pas) și definiția formei ferestrei principale (w2kdialogs.dfm) le puteți descărca de la www.netreport.ro.

Codul sursă proiect Delphi

```
program testdialogs;

uses
  Forms,
  testw2kdialogs in
   'testw2kdialogs.pas' {Form1},
  w2kdialogs in 'w2kdialogs.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1,
   Form1);
  Application.Run;
end.
```

2. Layered Windows & Alpha Blending

Un alt element de interfață utilizator introdus în *Windows 2000*, chiar mai spectaculos decât cele două ferestre prezentate mai sus, îl constituie posibilitatea creării de ferestre stratificate transparente și translucide.

Încă de la instalarea sistemului de operare se observă noile elemente GUI la aproape toate controalele : cursorul are o mică umbră translucidă, tool-tip-urile și meniurile apar și dispar cu fading, reprezentarea grafică a operației *drag and drop* este făcută cu alpha-blending, etc. Chiar și programele de la alte firme au început să profite de aceste îmbunătățiri vizuale ale interfeței.

Pentru folosirea acestor efecte programatorul trebuie să seteze bitul *WS_EX_LAYERED* la crearea ferestrei, sau mai târziu printr-un apel al funcției API *SetWindowLong*, în scopul creeri unei ferestre de tipul *Layered* la care îi va seta proprietățile cu funcția *SetLayeredWindowAttributes*.

Să le luăm pe rând :

Funcția *SetWindowLong* ce are prototipul

```
LONG SetWindowLong(
  HWND hWnd,        // handle to window
  int nIndex,       // offset of value
   to set
  LONG dwNewLong    // new value
);
```

se folosește pentru schimbarea atributelor unei ferestre specificate prin handlerul *hWnd*.

Parametru *nIndex* reprezintă indexul valorii ce se dorește a fii setată. De importanță pentru noi este valoarea *GWL_EXSTYLE* a acestui parametru ce permite setarea unui stil extins de fereastră.

A doua funcție este *SetLayeredWindowAttributes* și are următorul prototip:

```
BOOL SetLayeredWindowAttributes(
  HWND hwnd,            // handle to
   the layered window
  COLORREF crKey,       // specifies
   the color key
  BYTE bAlpha,          // value for
   the blend function
  DWORD dwFlags         // action
);
```

Ea se folosește pentru setarea transparenței sau gradului de opacitate al unei ferestre de tipul *Layered*. Cele două efecte sunt selectate prin intermediul parametrului *dwFlags*. Dacă acesta are valoare *LWA_COLORKEY*, atunci zona din cadrul ferestrei ce are culoarea specificată de *crKey* se va desena transparentă. Dacă *dwFlags = LWA_ALPHA*, atunci funcția stabilește gradul de opacitate al ferestrei folosind parametrul *bAlpha* care este un număr cuprins intre 255 – opacitate totală și 0 – transparență totală.

2.1. Crearea unei ferestre translucide

Și acum să facem un test al celor spuse mai sus. Creem în Delphi un nou proiect cu o singură formă pe care plasăm un control de tipul trackbar din care controlăm gradul de transparență al ferestrei, precum și alte controale pentru a le observa comportamentul pe timpul rulării programului.

Primul lucru care trebuie să-l facem este să declarăm constantele folosite cât și funcția *SetLayeredWindowAttributes*.

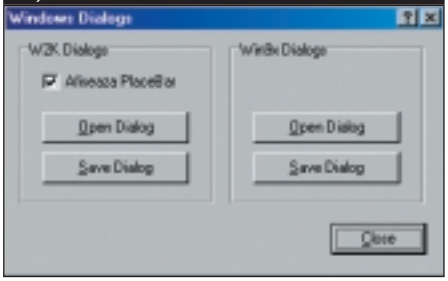
```
const
  lwa_Alpha      = 2; // constanta
   folosita de
   SetLayeredWindowAttributes
  ws_Ex_Layered = $80000; // Noul stil


function SetLayeredWindowAttributes
   (Wnd: hWnd; crKey: ColorRef;
   bAlpha: Byte; dwFlags: DWord):
   Bool; stdcall;
external 'user32.dll';
```

Apoi punem în secțiunea *Public* a clasei create de mediul IDE următoarea declarație

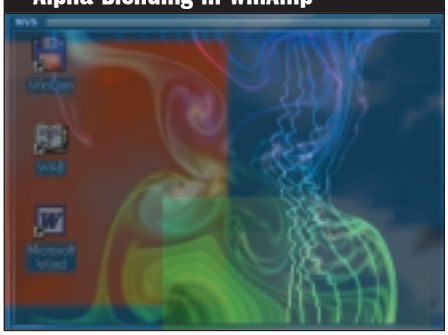
```
Procedure CreateParams(var Params:
   TCreateParams); override;
```

pe care o vom implementa în secțiunea *implementation* a unit-ului în felul următor:

Test clase: TW2KOpenDialog și TW2KSaveDialog



Alpha Blending în WinAmp



```
Procedure TMainForm.CreateParams(var
    Params: TCreateParams);
begin
  inherited CreateParams(Params);
  Params.ExStyle :=Params.ExStyle or
   ws_Ex_Layered;
end;
```

Aceasta determină aplicarea noului stil *Layered* asupra ferestrei. După cum am spus acest lucru se putea face și mai târziu printr-un apel al funcției *SetWindowLong* ca în exemplul:

```
SetWindowLong (Handle, gwl_ExStyle,
    GetWindowLong (Handle, gwl_ExStyle)
    or ws_Ex_Layered);
```

În subrutina de tratare a evenimentului ridicat de trackbar la mișcarea cursorului punem un apel către funcția *SetLayeredWindowAttributes* pentru a seta parametrii noii ferestre.

```
procedure TMainForm.track_translucid-
    Change(Sender: TObject);
begin
 SetLayeredWindowAttributes (Handle, 0,
    track_translucid.Position,
    lwa_Alpha);
end;
```

Mai trebuie să facem un singur lucru, și anume scrierea liniei `track_translucid.Position:=255;` în subrutina ce se execută la crearea formei. Aceasta determină schimbarea poziției cursorului trackbarului și deci apelarea metodei *TMainForm.track_translucidChange*. Astfel ne asigurăm că funcția *SetLayeredWindowAttributes* se execută și imediat după crearea ferestrei.

După compilarea și rularea programului, ne va apărea următoarea fereastra din figura „Fereastra de tip layered window".

Până acum nimic deosebit. Aceasta seamănă cu orice fereastră Windows clasică. Acționând însă cursorul trackbar-ului din partea de jos, observăm cum fereastra începe să se *degradeze*, prin ea putându-se observa obiectele aflate în spate.

Codul sursă al unit-ului acestui program se poate descărca de pe www.netreport.ro.

2.2. Crearea unei ferestre cu umbră

Folosind efectul de *Alpha Blending* vom crea în continuare un program ce are o umbră translucidă în partea dreaptă și de jos a ferestrei. Programul va arăta ca în figura „Fereastră cu umbră".

Pentru simularea efectului de umbră vom crea în subrutina de tratare a evenimentului ce apare la crearea ferestrei principale (*OnCreate*) încă două obiecte de tipul *TForm* pe care le vom transforma în ferestre de tipul layered cu gradul de opacitate 150.

Pentru ca întreaga fereastră compusă din trei formuri (cel principal și cele două pentru umbre) să funcționeze ca un tot, mai trebuie tratate și evenimentele care apar la mutarea și redimensionarea ferestrei. Fără această tratare umbra ar rămâne pe loc dacă utilizatorul ar muta de exemplu fereastra.

Întrucât nu sunt lucruri noi față de exemplul precedent nu vom mai da detalii suplimentare, codul sursă al unitului principal putând fi descărcat de la www.netreport.ro.

3. Ferestre animate

Din studiile efectuate s-a constatat că dacă informațiile sunt prezentate cu diverse efecte grafice și de animație experiența produsă de utilizarea calculatorului personal este mult mai plăcută decât dacă informația ar fi pur și simplu aruncată pe ecran.

În acest sens în Windows ferestrele pot fi făcute să apară și dispară cu efecte de animație. Pentru a anima o fereastră se folosește funcția API *AnimateWindow* ce are următorul prototip:

```
BOOL AnimateWindow(
  HWND hwnd,      // handle to window
  DWORD dwTime,   // duration of
   animation
  DWORD dwFlags   // animation type
);
```

Ea așteaptă ca parametrii de intrare handle-ul ferestrei, durata animației și tipul de animație.

Parametrul dwTime exprimă în milisecunde timpul în care trebuie făcută animația. O valoare normală este 200.

Pentru specificarea tipului de animație se folosește *dwFlags* cu una din următoarele constante sau combinații ale lor:

AW_SLIDE – Folosește animația de tip slide. Implicit se folosește cea de tip scroll. `AW_SLIDE` este ignorat când se folosește cu `AW_CENTER`

AW_ACTIVATE – Activează fereastra. Nu trebuie folosit cu `AW_HIDE`

AW_BLEND – Folosește efect de tip fadding. Acest efect poate fi folosit doar dacă fereastra este deasupra.

AW_HIDE – Ascunde fereastra. Implicit fereastra se afișează.

AW_CENTER – Face ca fereastra să apară și dispară din centrul ei.

AW_HOR_POSITIVE – Crează o animație de la stânga la dreapta. Se ignoră când se folosește cu `AW_CENTER` sau `AW_BLEND`.

AW_HOR_NEGATIVE – Crează o animație de la dreapta la stânga. Se ignoră când se folosește cu `AW_CENTER` sau `AW_BLEND`.

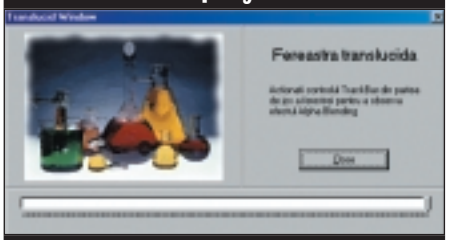
AW_VER_POSITIVE – Crează o animație de sus în jos. Se ignoră când se folosește cu `AW_CENTER` sau `AW_BLEND`.

AW_VER_NEGATIVE – Crează o animație de jos în sus. Se ignoră când se folosește cu `AW_CENTER` sau `AW_BLEND`.

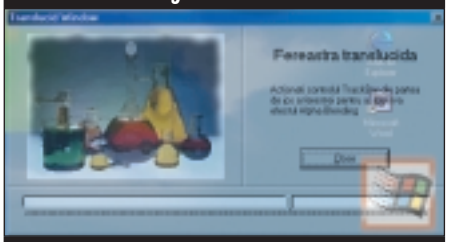
Pentru testarea acestei funcții vom crea în Delphi un program demonstrativ compus din două forme, una din care se selectează tipul de animația iar alta afișată sau ascunsă cu animația selectată.

Codul sursă al unit-urilor ce însoțesc cele două forme se poate descărca de la ww.netreport.ro.

Fereastra de tip layered window

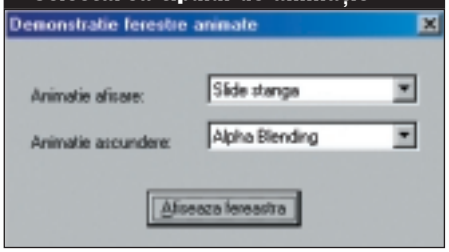


Fereastră layered translucidă



Fereastră cu umbră

Selectarea tipului de animație



Bibliografie

MSDN Online
http://msdn.microsoft.com

The Delphi Magazine
http://www.thedelphimagazine.com

Delphi Informant Magazine
http://www.delphizine.com

Marian Vețeanu este ... email: vmasoft@yahoo.com, Web: http://vmasoft.hypermart.net. ■ 89