Programming Windows

Terry Marris Feb 2013

9 Edit Functions

We see how to implement the menu *Edit* options *Undo*, *Cut*, *Copy*, *Paste*, *Delete* and *Select All*.

9.1 Menu Edit Messages

The code for implementing the edit functions is refreshingly simple.

```
/* Edit Menu Messages */
case IDM_EDIT_UNDO:
   SendMessage(hwndEdit, WM_UNDO, 0, 0);
   return 0;

case IDM_EDIT_CUT:
   SendMessage(hwndEdit, WM_CUT, 0, 0);
   return 0;

case IDM_EDIT_COPY:
   SendMessage(hwndEdit, WM_COPY, 0, 0);
   return 0;

case IDM_EDIT_PASTE:
   SendMessage(hwndEdit, WM_PASTE, 0, 0);
   return 0;

case IDM_EDIT_CLEAR:
   SendMessage(hwndEdit, WM_CLEAR, 0, 0);
   return 0;

case IDM_EDIT_SELECT_ALL:
   SendMessage(hwndEdit, EM_SETSEL, 0, -1);
   return 0;
```

We send the appropriate message to the edit window.

EM_SETSEL selects a range of characters in an edit control. The range is from *wParam* up to *lParam*. If *wParam* is zero and *lParam* is -1, then all the text in the edit control is selected.

The constants *IDM_EDIT_UNDO*, *IDM_EDIT_CUT*, ... are defined in *pted3.h*.

9.2 Menu Keyboard Interface

You have probably selected a menu item by using the *Alt* key in combination with a letter in the menu name. Pressing the Alt key results in a letter of each menu item being underlined.

So, for example, *Alt F, O* selects the *Open* option in the *File* menu and starts the *Open File* dialog.

In the resource file we specify which character is to be used to select a menu item be preceding it with an ampersand.

```
POPUP "&File" {
   MENUITEM "&New",      IDM_FILE_NEW
   MENUITEM "&Open",     IDM_FILE_OPEN
   MENUITEM "&Save",     IDM_FILE_SAVE
   MENUITEM "Save &As", IDM_FILE_SAVE_AS
   MENUITEM SEPARATOR
   MENUITEM "&Print\",  IDM_FILE_PRINT
   MENUITEM SEPARATOR
   MENUITEM "E&xit",     IDM_APP_EXIT
}
```

9.3 Keyboard Accelerators

A keyboard accelerator is a combination of keys that launch an action. For example *Ctrl+Z* undoes the previous action, *Ctrl+X* cuts the selected text, and *Ctrl+V* pastes the text into the current cursor (or caret) position. Similarly, *Ctrl+O* launches the *Open* file dialolg, *Ctrl+S* saves the current text, and *Ctrl+P* launches the *Print* dialog.

First, we specify the keyboard accelerators in the resource file. For example:

```
...
POPUP "&Edit" {
   MENUITEM "&Undo\tCtrl+Z", IDM_EDIT_UNDO
   MENUITEM SEPARATOR
   MENUITEM "Cu&t\tCtrl+X",   IDM_EDIT_CUT
   MENUITEM "&Copy\tCtrl+C",  IDM_EDIT_COPY
   MENUITEM "&Paste\tCtrl+V", IDM_EDIT_PASTE
   MENUITEM "De&lete\tDel",   IDM_EDIT_CLEAR
   MENUITEM SEPARATOR
   MENUITEM "Select &All\tCtrl+A", IDM_EDIT_SELECT_ALL
}
...
```

The accelerator table is defined in the resource file.

```
PTED ACCELERATORS
{
   VK_BACK,   IDM_EDIT_UNDO,      VIRTKEY, ALT
   VK_DELETE, IDM_EDIT_CLEAR,     VIRTKEY
   VK_DELETE, IDM_EDIT_CUT,       VIRTKEY, SHIFT
   VK_F1,     IDM_HELP,           VIRTKEY
   VK_F3,     IDM_SEARCH_NEXT,    VIRTKEY
   VK_INSERT, IDM_EDIT_COPY,      VIRTKEY, CONTROL
   VK_INSERT, IDM_EDIT_PASTE,     VIRTKEY, SHIFT
   "^A",      IDM_EDIT_SELECT_ALL,ASCII
   "^C",      IDM_EDIT_COPY,      ASCII
   "^F",      IDM_SEARCH_FIND,    ASCII
   "^N",      IDM_FILE_NEW,       ASCII
   "^O",      IDM_FILE_OPEN,      ASCII
   "^P",      IDM_FILE_PRINT,     ASCII
   "^R",      IDM_SEARCH_REPLACE, ASCII
```

```
    "^S",       IDM_FILE_SAVE,      ASCII
    "^V",       IDM_EDIT_PASTE,     ASCII
    "^X",       IDM_EDIT_CUT,       ASCII
    "^Z",       IDM_EDIT_UNDO,      ASCII
}
```

PTED identifies the accelerators table. It must match the *appName* used in *WinMain*. *VK_* stands for virtual key. A virtual key code identifies a key in a device-independent way. *VIRTKEY* signifies a function key, such as the back-arrow key. *ALT*, *SHIFT* and *CTRL* means that the given key can be used along with the *Alt*, shift and control keys.

The ^ in *"^A"* specifies the *Ctrl +A* combination.

We load the accelerators in *WinMain*.

```
HACCEL accel;
...
accel = LoadAccelerators(hInst, appName);
```

And call *TranslateAccelerator* in the message loop.

```
while ((ret = GetMessage(&msg, NULL, 0, 0)) != 0) {
   ...
   if (!TranslateAccelerator(hwnd, accel, &msg)) {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
}
```

LoadAccerators returns *NULL* if an error occurs. *TranslateAccelerator* processes accelerator keys for menu commands. It returns zero if the function fails.

Shown below are the complete header and resource files, along with the main program module.

```
/* pted3.h */

#ifndef PTED1_H
#define PTED1_H

#define IDE_EDITABLE 101
#define IDB_STATUS_BAR 102
#define IDC_FILENAME 150

#define IDM_FILE_NEW 200
#define IDM_FILE_OPEN 201
#define IDM_FILE_SAVE 202
#define IDM_FILE_SAVE_AS 203
#define IDM_FILE_PRINT 204
#define IDM_APP_EXIT 205

#define IDM_EDIT_UNDO 210
#define IDM_EDIT_CUT 211
#define IDM_EDIT_COPY 212
#define IDM_EDIT_PASTE 213
#define IDM_EDIT_CLEAR 214
#define IDM_EDIT_SELECT_ALL 215
```

```
#define IDM_SEARCH_FIND 220
#define IDM_SEARCH_NEXT 221
#define IDM_SEARCH_REPLACE 222

#define IDM_FORMAT_FONT 230

#define IDM_HELP 240
#define IDM_APP_ABOUT 241

#define REP_FATAL 1
#define REP_WARNING 2
#define REP_DIAGNOSTIC 3

#define EDT_TEXTNOTCHANGED 0

#define PMAR_LINES 8
#define PMAR_PIXELS 450

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK EditProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);
int report(int, PSTR, PSTR, ...);

OPENFILENAME PTEDFileInit(HWND);
int fileNew(HWND, LPSTR fileName, LPSTR fileTitle);
int fileOpen(HWND, OPENFILENAME *, LPSTR, LPSTR);
int fileSaveAs(HWND, OPENFILENAME *, LPSTR, LPSTR);
int fileSaveOnExit(HWND, LPSTR, LPSTR);
int fileWrite(HWND, LPSTR);

BOOL PTEDFilePrint(HINSTANCE, HWND, HWND, LPSTR);

#endif
```

```
/* pted3res.rc - primitive text editor resources */

#include <windows.h>
#include <afxres.h>
#include "pted3.h"

ABOUTBOX DIALOG 32, 28, 180, 75
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
FONT 8, "Tahoma"
CAPTION "About PTED"
{
   DEFPUSHBUTTON "OK", IDOK, 66, 48, 50, 14
   CTEXT "PTED", IDC_STATIC, 40, 8, 100, 8
   CTEXT "Primitive Text Editor", IDC_STATIC, 7, 20, 166, 8
   CTEXT "Terry Marris Jan 2013", IDC_STATIC, 7, 32, 166, 8
}
```

```
PTED MENU
{
   POPUP "&File" {
      MENUITEM "&New\tCtrl+N",     IDM_FILE_NEW
      MENUITEM "&Open...\tCtrl+O", IDM_FILE_OPEN
      MENUITEM "&Save\tCtrl+S",    IDM_FILE_SAVE
      MENUITEM "Save &As...",      IDM_FILE_SAVE_AS
      MENUITEM SEPARATOR
      MENUITEM "&Print\tCtrl+P",   IDM_FILE_PRINT
      MENUITEM SEPARATOR
      MENUITEM "E&xit",            IDM_APP_EXIT
   }
   POPUP "&Edit" {
      MENUITEM "&Undo\tCtrl+Z", IDM_EDIT_UNDO
      MENUITEM SEPARATOR
      MENUITEM "Cu&t\tCtrl+X",   IDM_EDIT_CUT
      MENUITEM "&Copy\tCtrl+C",  IDM_EDIT_COPY
      MENUITEM "&Paste\tCtrl+V", IDM_EDIT_PASTE
      MENUITEM "De&lete\tDel",   IDM_EDIT_CLEAR
      MENUITEM SEPARATOR
      MENUITEM "Select &All\tCtrl+A", IDM_EDIT_SELECT_ALL
   }
   POPUP "&Search" {
      MENUITEM "&Find...\tCtrl+F",    IDM_SEARCH_FIND
      MENUITEM "Find &Next\tF3",       IDM_SEARCH_NEXT
      MENUITEM "&Replace...\tCtrl+R", IDM_SEARCH_REPLACE
   }

   POPUP "F&ormat" {
      MENUITEM "&Font...", IDM_FORMAT_FONT
   }

   POPUP "&Help" {
      MENUITEM "&Help Topics",   IDM_HELP
      MENUITEM "&About PTED...", IDM_APP_ABOUT
   }
}

PTED ACCELERATORS
{
   VK_BACK,   IDM_EDIT_UNDO,      VIRTKEY, ALT
   VK_DELETE, IDM_EDIT_CLEAR,     VIRTKEY
   VK_DELETE, IDM_EDIT_CUT,       VIRTKEY, SHIFT
   VK_F1,     IDM_HELP,           VIRTKEY
   VK_F3,     IDM_SEARCH_NEXT,    VIRTKEY
   VK_INSERT, IDM_EDIT_COPY,      VIRTKEY, CONTROL
   VK_INSERT, IDM_EDIT_PASTE,     VIRTKEY, SHIFT
   "^A",      IDM_EDIT_SELECT_ALL,ASCII
   "^C",      IDM_EDIT_COPY,      ASCII
   "^F",      IDM_SEARCH_FIND,    ASCII
   "^N",      IDM_FILE_NEW,       ASCII
   "^O",      IDM_FILE_OPEN,      ASCII
   "^P",      IDM_FILE_PRINT,     ASCII
   "^R",      IDM_SEARCH_REPLACE, ASCII
   "^S",      IDM_FILE_SAVE,      ASCII
   "^V",      IDM_EDIT_PASTE,     ASCII
   "^X",      IDM_EDIT_CUT,       ASCII
   "^Z",      IDM_EDIT_UNDO,      ASCII
}
```

```
/* pted3.c - primitive text editor */

#include <windows.h>
#include <commctrl.h>
#include <stdarg.h>
#include <stdio.h>

#include "pted3.h"

WNDPROC oldEdit;


int report(int repType, PSTR caption, PSTR format, ...)
{
   CHAR string[256];
   va_list args;

   va_start(args, format);
   vsprintf(string, format, args);
   va_end(args);

   MessageBox(NULL, string, caption, MB_OK);




   if (repType == REP_FATAL)
     PostQuitMessage(0);
   return 0;
}


int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst,
                   PSTR cmdLine, int cmdShow)
{
   static char appName[] = "PTED";
   HWND hwnd;
   MSG msg;
   WNDCLASS wc;
   HACCEL accel;
   BOOL ret;

   wc.style         = CS_HREDRAW | CS_VREDRAW;
   wc.lpfnWndProc   = WndProc;
   wc.cbClsExtra    = 0;
   wc.cbWndExtra    = 0;
   wc.hInstance     = hInst;
   wc.hIcon         = NULL;
   wc.hCursor       = LoadImage(NULL, IDC_ARROW, IMAGE_CURSOR,
                        0, 0, LR_SHARED);
   wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
   wc.lpszMenuName  = "PTED";
   wc.lpszClassName = appName;

   if (0 == (RegisterClass(&wc)))
      report(REP_FATAL, "WinMain", "%s", "RegisterClass failed");
```

```
   if (NULL == (hwnd = CreateWindow(appName,
                       "PTED3",
                       WS_OVERLAPPEDWINDOW,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                       CW_USEDEFAULT, CW_USEDEFAULT,
                       NULL,
                       NULL,
                       hInst,
                       NULL)))
      report(REP_FATAL, "WinMain", "%s", "CreateWindow failed");

   ShowWindow(hwnd, cmdShow);
   UpdateWindow(hwnd);

  accel = LoadAccelerators(hInst, appName);

   while ((ret = GetMessage(&msg, NULL, 0, 0)) != 0) {
      if (ret < 0)
         report(REP_FATAL, "WinMain", "%s", "GetMessage failed");
      if (!TranslateAccelerator(hwnd, accel, &msg)) {
         TranslateMessage(&msg);
         DispatchMessage(&msg);
      }
   }
   return msg.wParam ;
}


HWND newEditWindow(HWND hwnd, HINSTANCE hInstance)
{
   HWND hwndEdit;

   hwndEdit = CreateWindow("EDIT", NULL, WS_CHILD |
                           WS_VISIBLE | WS_HSCROLL |
                           WS_VSCROLL | WS_BORDER |
                           ES_LEFT | ES_MULTILINE |
                           ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                           0, 0, 0, 0,
                           hwnd, (HMENU)IDE_EDITABLE,
                           hInstance, NULL);
   SendMessage(hwndEdit, EM_LIMITTEXT, 32000, 0L);
   return hwndEdit;
}


void setInitialFont(HWND hwndEdit)
{
   HFONT hFont;
   HDC hdc;
   LOGFONT logFont;
   int pointSize = 10;
   char *faceName = "Lucida Console";
   HWND hwnd;

   hwnd = GetParent(hwndEdit);
   hdc = GetDC(hwnd);
   ZeroMemory(&logFont, sizeof(logFont));
   logFont.lfHeight = -MulDiv(pointSize,
                              GetDeviceCaps(hdc, LOGPIXELSY), 72);
```

```
   strcpy(logFont.lfFaceName, faceName);
   hFont = CreateFontIndirect(&logFont);
   SendMessage(hwndEdit, WM_SETFONT, (WPARAM)hFont, MAKELPARAM(TRUE,0));
}


HWND newStatusBar(HWND hwnd, HINSTANCE hInstance)
{
   HWND hwndStatusBar;
   int parts[] = { 150, -1 };

   hwndStatusBar = CreateWindow(STATUSCLASSNAME, NULL,
                           SBARS_SIZEGRIP | WS_CHILD | WS_VISIBLE,
                           0, 0, 0, 0,
                           hwnd, (HMENU)IDB_STATUS_BAR,
                           hInstance, NULL);

   SendMessage(hwndStatusBar, SB_SETPARTS,
               (WPARAM)sizeof(parts) / sizeof(int),
               (LPARAM)&parts);

   SendMessage(hwndStatusBar, SB_SETTEXT, 0,
               (LPARAM)"Line 0, Column 0");
   return hwndStatusBar;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg,
                     WPARAM wParam, LPARAM lParam)
{
   static HINSTANCE hInstance;
   static HWND hwndEdit;
   static HWND hwndStatusBar;
   int statusBarHeight = 20;

   static OPENFILENAME ofn;
   static char fileName[MAX_PATH];
   static char fileTitle[MAX_PATH];

   switch (msg) {
   case WM_CREATE:
      hInstance = ((LPCREATESTRUCT)lParam)->hInstance;
      hwndEdit = newEditWindow(hwnd, hInstance);
      setInitialFont(hwndEdit);
      hwndStatusBar = newStatusBar(hwnd, hInstance);

      oldEdit = (WNDPROC)SetWindowLong(hwndEdit, GWL_WNDPROC,
                                             (LONG)EditProc);
      ofn = PTEDFileInit(hwnd);
      return 0;

   case WM_SETFOCUS:
      SetFocus(hwndEdit);
      return 0;

   case WM_SIZE:
      MoveWindow(hwndEdit, 0, 0, LOWORD(lParam),
                 HIWORD(lParam) - statusBarHeight, TRUE);
      SendMessage(hwndStatusBar, WM_SIZE, 0, 0);
      return 0;
```

```
case WM_COMMAND:
   if (LOWORD(wParam) == IDE_EDITABLE)
      if (HIWORD(wParam) == EN_ERRSPACE ||
          HIWORD(wParam) == EN_MAXTEXT)
         report(REP_FATAL, "WndProc", "%s",
               "Edit control out of space");

   switch (LOWORD(wParam)) {
   /* File Menu Messages */
   case IDM_FILE_NEW:
      return fileNew(hwndEdit, fileName, fileTitle);

   case IDM_FILE_OPEN:
      return fileOpen(hwndEdit, &ofn, fileName, fileTitle);

   case IDM_FILE_SAVE:
      if (fileName[0]) {
         fileWrite(hwndEdit, fileName);
         return 1;
      }
      /* fall through */
   case IDM_FILE_SAVE_AS:
      return fileSaveAs(hwndEdit, &ofn, fileName, fileTitle);

   case IDM_APP_EXIT:
      fileSaveOnExit(hwndEdit, fileName, fileTitle);
      SendMessage(hwnd, WM_CLOSE, 0, 0);
      return 0;

   case IDM_FILE_PRINT:
      if (!PTEDFilePrint(hInstance, hwnd, hwndEdit, fileTitle))
         report(REP_WARNING, "WndProc", "%s could not be printed",
                fileTitle);
      return 0;

   /* Edit Menu Messages */
   case IDM_EDIT_UNDO:
      SendMessage(hwndEdit, WM_UNDO, 0, 0);
      return 0;

   case IDM_EDIT_CUT:
      SendMessage(hwndEdit, WM_CUT, 0, 0);
      return 0;

   case IDM_EDIT_COPY:
      SendMessage(hwndEdit, WM_COPY, 0, 0);
      return 0;

   case IDM_EDIT_PASTE:
      SendMessage(hwndEdit, WM_PASTE, 0, 0);
      return 0;

   case IDM_EDIT_CLEAR:
      SendMessage(hwndEdit, WM_CLEAR, 0, 0);
      return 0;

   case IDM_EDIT_SELECT_ALL:
      SendMessage(hwndEdit, EM_SETSEL, 0, -1);
      return 0;
```

```
      /* Search Menu Messages */
      case IDM_SEARCH_FIND:
      case IDM_SEARCH_NEXT:
      case IDM_SEARCH_REPLACE:
         report(REP_DIAGNOSTIC, "WndProc", "%s",
               "Search menu not implemented");
         return 0;

      /* Font Menu Messages */
      case IDM_FORMAT_FONT:
         report(REP_DIAGNOSTIC, "WndProc", "%s",
               "Format menu not implemented");
         return 0;

      /* Help Menu Messages */
      case IDM_HELP:
         report(REP_DIAGNOSTIC, "WndProc", "%s",
               "Help menu not implemented");
         return 0;

      case IDM_APP_ABOUT:
         DialogBox(hInstance, "ABOUTBOX" , hwnd,
                   (DLGPROC)AboutDlgProc);
         return 0;
      }
      return 0;

   case WM_DESTROY:
       PostQuitMessage(0);
       return 0;
   }
   return DefWindowProc(hwnd, msg, wParam, lParam);
}

LRESULT CALLBACK AboutDlgProc(HWND hDlg, UINT msg,
                              WPARAM wParam, LPARAM lParam)
{
   switch (msg) {
   case WM_INITDIALOG:
      return TRUE;

   case WM_COMMAND:
      switch (LOWORD(wParam)) {
      case IDOK:
         EndDialog(hDlg, 0);
         return TRUE;
      }
      break;
   }
   return FALSE;
}
```

```
/* EDitProc - extends hwndEdit */
LRESULT CALLBACK EditProc(HWND hwnd, UINT msg, WPARAM wParam,
                          LPARAM lParam)
{
   static long line, column;
   static long selection, start, lineIndex;
   static char buf[256];

   HWND hwndParent, hwndStatusBar;

   switch (msg) {
   case WM_KEYUP:
   case WM_LBUTTONUP:
      line = SendMessage(hwnd, EM_LINEFROMCHAR, -1, 0);

      selection = SendMessage(hwnd, EM_GETSEL, 0, 0);
      start = LOWORD(selection);
      lineIndex = SendMessage(hwnd, EM_LINEINDEX, -1, 0);
      column = start - lineIndex;

      sprintf(buf, "Line %ld, Column %ld", line, column);

      hwndParent = GetParent(hwnd);
      hwndStatusBar = GetDlgItem(hwndParent, IDB_STATUS_BAR);
      SendMessage(hwndStatusBar, SB_SETTEXT, 0, (LPARAM)buf);
      break;

   }
   return CallWindowProc(oldEdit, hwnd, msg, wParam, lParam);
}
```

Next, we work on the *Search* menu options.