How can I add my own custom menu item to another application?

This tip is something that I've wanted to do for a while, but kept on forgetting to write the article for it. It involves adding a menu choice to the system menu of an application. For the most part, you'll never have a need to do this. But there are some things like setting a form style, or some other action that is more system oriented than application oriented that just belong in the system menu. Well, here it is folks, and as usual, it's pretty incredibly easy to implement.

If you've tried to do this before but couldn't, it's because there is no way to add a menu item with standard Delphi calls. You have to trap Windows the windows message *WM_SYSCOMMAND* and evaluate the *wParam* message element to see if your added menu item was selected. Really folks, it's not that hard, and a little digging in the API help was all I needed to do find out how to implement this in a program. Basically, what you have to do is this:

Create a new form. Override the *OnMessage* event by assigning a new event handler procedure for the *OnMessage* event. Create a constant that will be used as the ordinal identifier for your menu choice. In the *FormCreate*, make your menu choice with the *AppendMenu* API call. Here's the code to show you how to do it:

```
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, Menus;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { ... }
  public
    { This declaration is of the type TMessageEvent which is a pointer to a
      procedure that takes two variable arguments of type TMsg and Boolean,
      respectively }
    procedure WinMsgHandler(var Msg : TMsg; var Handled : Boolean);
  end;

var
  Form1: TForm1;

const
  MyItem = 100; // Here's the menu identifier. It can be any WORD value

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  { First, tell the application that its message handler is different from the
    default }
  Application.OnMessage := WinMsgHandler;
  { Add a separator }
  AppendMenu(GetSystemMenu(Self.Handle, False), MF_SEPARATOR, 0, '');
  { Add your menu choice. Since the Item ID is high, using the MF_BYPOSITION
    constant will place it last on the system menu }
  AppendMenu(GetSystemMenu(Self.Handle, False),
    MF_BYPOSITION, MyItem, 'My Men&u Choice');
end;

procedure TForm1.WinMsgHandler(var Msg: TMsg; var Handled: Boolean);
begin
  { If the message is a system one... }
  if Msg.Message=WM_SYSCOMMAND then
    if Msg.wParam = MyItem then
      { Put handling code here. I've opted for a ShowMessage for demonstration
        purposes }
```

```
    ShowMessage('You picked my menu!!!');
end;

end.
```

As you can see, this is fairly straight-forward. Granted, the tip is not very complicated. However, it does open up many doors to things you can do. In anticipation of some questions you might have later, The *AppendMenu* command can also be used with minimized apps. For instance, if you minimize your app, the icon represents the application, not your form. Therefore in order to make the system menu with your changes visible when in minimized form you would use *Application.Handle* instead of *Self.Handle* to deal with the application's system menu.

Author:	Unknown
Added:	2007-06-02
Last updated:	2007-06-02