Prevent a Delphi form from being moved

There are various solutions to the problem of preventing a user from moving a form on screen.

Solution 1

First, make the form's *BorderStyle* something like *bsDialog*, so that the window can't be resized.

Next, add the following declaration to your form class:

```
procedure PosChange(var Msg: TWmWindowPosChanging);
  message WM_WINDOWPOSCHANGING;
```

Implement the message handler as:

```
procedure TForm1.PosChange(var Msg: TWmWindowPosChanging) ;
begin
  Msg.WindowPos.x := Left;
  Msg.WindowPos.y := Top;
  Msg.Result := 0;
end;
```

Thats it. Easy as can be. The only problem with this is that you can't move the form if you want your code to. To get around this, just set up a Boolean variable called *PosLocked*, set it to *True* when you want to lock the form's position, and to *False* when you need to move the form. (When your done, remember to set it back to true). Then to implement the proc above, just make it...

```
procedure TForm1.PosChange(var Msg: TWmWindowPosChanging) ;
begin
  if PosLocked then
  begin
    Msg.WindowPos.x := Left;
    Msg.WindowPos.y := Top;
    Msg.Result := 0;
  end
  else
    inherited;
end;
```

Solution 2

In this solution the form will be completely immovable, but the user can still resize, minimize or maximize it. How does it work? We will just remove the "move" command from system menu using simple code in the form's *OnCreate* event handler.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DeleteMenu(GetSystemMenu(Handle, False), SC_MOVE, MF_BYCOMMAND);
end;
```

Solution 3

In this solution both sizing and moving the form are inhibited by trapping the appropriate system commands and preventing further processing.

Add the following method to the form's class declaration:

```
private
  ...
  procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  ...
```

And implement it as follows:

```
procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if ((Msg.CmdType and $FFF0) = SC_MOVE) or
    ((Msg.CmdType and $FFF0) = SC_SIZE) then
  begin
    Msg.Result := 0;
    Exit;
  end;
  inherited;
end;
```

Solution 4

This final solution prevents the user from dragging the form with the mouse:

Add the following method to the form's class declaration:

```
private
  ...
  procedure WMNCHitTest(var Message: TWMNCHitTest); message WM_NCHITTEST;
  ...
```

And implement it as follows:

```
procedure TForm1.WMNCHitTest(var Message: TWMNCHitTest);
begin
  inherited;
  with Message do
    if Result = HTCAPTION then
      Result := HTNOWHERE;
end;
```

Author:	Unknown
Contributor:	Riccardo Faiella (Topellina)
Added:	2012-07-06
Last updated:	2012-07-06