

A Collection of Delphi Tips & Tricks

March 11, 2005

The code has been tested in Delphi 6/7 and might not work in previous releases.

Contents

Application

Minimize application to taskbar.....	5
Prevent a screensaver to kick in as long as your application is running.....	22

Forms

Show/hide titlebar of a form.....	4
Set a form to stay on top of all other (non-topmost) windows.....	6
Capture Maximize/Minimize/Close button clicks.....	7
Disable (gray out) Close button of a form.....	7
Detect the movement of a form.....	13
Disabling the movement of a form.....	14
Create form with rounded corners.....	20
Make transparent form.....	20
Detect when a form has been minimized, maximized or restored.....	22

Dialogs

Modify the controls in common dialogs.....	6
Repositioning common dialogs.....	14

Controls

Customized tooltips.....	4
Show controls' hints on statusbar.....	5
Disable the popup menu of TEdit and TMemo.....	7
Hot-tracking controls.....	8
Implementing a control array.....	11
Using windows API in textbox programming.....	16

Menus

Disable the popup menu of TEdit and TMemo.....	7
--	---

Append items into system menu	9
Mouse	
Check whether a mouse button is being pressed	4
Capture mouse clicks at statusbar	19
Graphics	
Drawing rotated text	11
Avoid flickering in graphics programming	15
Changing text alignment	16
Drawing transparent text	17
Save graphics to a bitmap file or a Windows Metafile	21
Installing a font on the fly	23
Fonts	
Get a list of system fonts	7
Files and folders	
Drag'n'drop files from Windows Explorer	9
Open a file using its associated application	10
Determine the size of a file without opening it	10
Send a file to Recycle bin	10
Retrieve the path of the Windows directory	15
Windows API	
Check whether a mouse button is being pressed	4
Show/hide titlebar of a form	4
Set a form to stay on top of all other (non-topmost) windows	6
Modify the controls in common dialogs	6
Capture Maximize/Minimize/Close button clicks	7
Disable (gray out) Close button of a form	7
Hot-tracking controls	8
Append items into system menu	9
Drag'n'drop files from Windows Explorer	9
Send a file to Recycle bin	10
Open a file using its associated application	10
Drawing rotated text	11
Monitor the changes of clipboard's content	13
Detect the movement of a form	13
Disabling the movement of a form	14
Repositioning common dialogs	14
Retrieve the path of the Windows directory	15
Using windows API in textbox programming	16
Changing text alignment	16

Capture mouse clicks at statusbar	19
Create form with rounded corners.....	20
Make transparent form	20
Detect when a form has been minimized, maximized or restored.....	22
Prevent a screensaver to kick in as long as your application is running.....	22
Installing a font on the fly	23
Using WinHelp API	23
Listing system's drives in a listbox	23
Algorithms	
Converting between decimal, binary, and hexadecimal representation of a number	12
Miscellaneous	
Monitor the changes of clipboard's content	13
Delphi's equivalent of the VB's App object.....	17
Find if you are connected to the Internet.....	20

1. Customized tooltips

```
...
type
  TMyHintWindow = class(THintWindow)
    constructor Create(AOwner: TComponent); override;
  end;
...

constructor TMyHintWindow.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Canvas.Font.Name := 'Arial';
  Canvas.Font.Size := 14;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.ShowHint := false;
  HintWindowClass := TMyHintWindow;
  Application.ShowHint := true;
end;
```

2. Check whether a mouse button is being pressed

```
...
uses
  Windows;
...
function IsBtnPressed(ABtn: integer): boolean;
//ABtn can be either VK_LBUTTON, VK_MBUTTON, or VK_RBUTTON
begin
  result := (GetAsyncKeyState(ABtn) and $8000) = $8000;
end
```

3. Show/hide titlebar of a form

```
...
uses
  Forms, Windows;
...
procedure ShowTitlebar(AForm: TForm; AShow: boolean);
var
  style: longint;
begin
  with AForm do begin
    if BorderStyle = bsNone then exit;
```

```

style := GetWindowLong(Handle, GWL_STYLE);
if AShow then begin
  if (style and WS_CAPTION) = WS_CAPTION then exit;
  case BorderStyle of
    bsSingle, bsSizeable:
      SetWindowLong(Handle, GWL_STYLE, style or WS_CAPTION or WS_BORDER);
    bsDialog:
      SetWindowLong(Handle, GWL_STYLE, style or WS_CAPTION or DS_MODALFRAME or
        WS_DLGFRAME);
  end;
end else begin
  if (style and WS_CAPTION) = 0 then exit;
  case BorderStyle of
    bsSingle, bsSizeable:
      SetWindowLong(Handle, GWL_STYLE, style and (not(WS_CAPTION)) or WS_BORDER);
    bsDialog:
      SetWindowLong(Handle, GWL_STYLE, style and (not(WS_CAPTION)) or
        DS_MODALFRAME or WS_DLGFRAME);
  end;
end;
SetWindowPos(Handle, 0, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE or SWP_NOZORDER or
  SWP_FRAMECHANGED or SWP_NOSENDCHANGING);
end;
end;

```

4. Minimize application to taskbar

A common error is attempting to minimize the form (`WindowState := wsMinimized`), rather than the application itself.

```

...
procedure TForm1.Button1Click(Sender: TObject);
begin
  Application.Minimize;
end;

```

5. Show controls' hints on statusbar

First method:

```

Button1.ShowHint := true;
StatusBar1.AutoHint := true;

```

Second method:

```

type
  TForm1 = class(TForm)
...
private

```

```

    procedure ShowHint(Sender: TObject);
...

procedure TForm1.ShowHint(Sender: TObject);
begin
    StatusBar1.SimpleText := Application.Hint;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    Application.OnHint := ShowHint;
end;

```

6. Modify the controls in common dialogs

```

...
uses
    Windows, Forms, Dialogs, Dlg;
...
//create handler for OnShow event of common dialog
procedure TForm1.OpenDialog1Show(Sender: TObject) ;
var
    hwnd: THandle;
begin
    hwnd := GetParent(OpenDialog1.Handle);
    //change title of OK button, "Open" —> "Do Open"
    SetDlgItemText(hwnd, IDOK, PChar('&Do Open'));
    //hide the "Open as read-only" check box
    ShowWindow(GetDlgItem(hwnd, chx1), SW_HIDE);
    //these are the IDs of remaining controls:
    //IDCANCEL: the CANCEL button
    //stc3, stc2: the two label controls, "File name" and "Files of type"
    //edt1, cmb1: the edit control and combo box next to the labels
    //cmb2: combo box that allows to select the drive or folder
    //stc4: label for the cmb2 combo
    //lst1: list box that displays the contents of selected drive or folder
end;

```

7. Set a form to stay on top of all other (non-topmost) windows

```

...
uses
    Forms, Windows;
...
procedure SetTopmost(AForm: TForm; ATop: boolean);
begin

```

```

if ATop then
    SetWindowPos(AForm.Handle, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE)
else
    SetWindowPos(AForm.Handle, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE)
    ;
end;

```

8. Capture Maximize/Minimize/Close button clicks

```

...
type
    TForm1 = class(Form)
    ...
    public
        procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
    ...

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
    if Msg.CmdType = SC_MINIMIZE) or (Msg.CmdType = SC_MAXIMIZE) or (Msg.CmdType =
        SC_CLOSE) then
        ...
    else
        DefaultHandler(Msg);
end;

```

9. Disable (gray out) Close button of a form

```

procedure TForm1.FormCreate(Sender: TObject);
var
    hSysMenu: HMENU;
begin
    hSysMenu := GetSystemMenu(Handle, false);
    EnableMenuItem(hSysMenu, SC_CLOSE, MF_BYCOMMAND or MF_GRAYED);
end;

```

10. Get a list of system fonts

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    ComboBox1.Items.Assign(Screen.Fonts);
end;

```

11. Disable the popup menu of TEdit and TMemo

```

//create a handler for OnContextPopup event of TMemo
procedure TForm1.Memo1ContextPopup(Sender: TObject; MousePos: TPoint; var Handled:
    Boolean);

```

```

begin
    Handled := true;
end;

```

12. Hot-tracking controls

The purpose is to highlight the control which receives the focus.

First method:

```

type
    TMyEdit = class(TEdit)
    protected
        procedure DoEnter; override;
        procedure DoExit; override;
    end;
...

procedure TMyEdit.DoEnter;
begin
    Color := clAqua;
    inherited;
end;

procedure TMyEdit.DoExit;
begin
    Color := clWindow;
    inherited;
end;

```

Second method:

```

type
    TMyLabel = class(TLabel)
    protected
        procedure WndProc(var AMessage : TMessage); override;
    end;
...

procedure TMyLabel.WndProc(var AMessage: TMessage);
begin
    if (AMessage.Msg = CM_MOUSEENTER) then
        Color := clAqua
    else if (AMessage.Msg = CM_MOUSELEAVE) then
        Color := clWindow;
    inherited;
end;

```

13. Append items into system menu

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  end;
...
const
  ID_ABOUT = WM_USER + 1;

procedure TForm1.FormCreate(Sender: TObject);
var
  hSysMenu: HMENU;
begin
  hSysMenu := GetSystemMenu(Handle, false);
  AppendMenu(hSysMenu, MF_SEPARATOR, 0, nil);
  AppendMenu(hSysMenu, MF_STRING, ID_ABOUT, PChar('&About...'));
end;

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if Msg.CmdType = ID_ABOUT then
    AboutForm.ShowModal
  else
    inherited;
end;
```

14. Drag'n'drop files from Windows Explorer

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure WMDropFiles(var Msg: TWMDROPPFILES); message WM_DROPPFILES;
  end;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
  DragAcceptFiles(Handle, true);
end;

procedure TForm1.WMDropFiles(var Msg: TWMDROPPFILES);
var
  buf: array[0..MAX_PATH] of char;
```

```

    filename: string;
begin
    DragQueryFile(Msg.Drop, 0, @buf, sizeof(buf));
    DragFinish(Msg.Drop);
    filename := string(buf);
    ...
end;

```

15. Determine the size of a file without opening it

```

uses
    SysUtils;
    ...
function FileSizeByName(const AFile: string): integer;
var
    sr: TSearchRec;
begin
    if (Pos(AFile, '*') <> 0) or (Pos(AFile, '?') <> 0) or (FindFirst(AFile, faAnyFile,
        sr) <> 0) then
        result := -1 //file was not found
    else
        result := sr.Size;
    end;

```

16. Send a file to Recycle bin

```

uses
    ShellApi.pas;
    ...
function RecycleFile(const AFile: string): boolean;
var
    foStruct: TSHFileOpStruct;
begin
    with foStruct do begin
        wnd := 0;
        wFunc := FO_DELETE;
        pFrom := PChar(AFile+#0#0);
        pTo := nil;
        fFlags:= FOF_ALLOWUNDO or FOF_NOCONFIRMATION or FOF_SILENT;
        fAnyOperationsAborted := false;
        hNameMappings := nil;
    end;
    Result := SHFileOperation(foStruct) = 0;
end;

```

17. Open a file using its associated application

The result is the same as if the file were double-clicked in Windows Explorer.

```

function OpenFile(AFile: string; ADir: string = nil; AParams: string = nil): boolean
;
begin
    result := ShellExecute(Application.Handle, 'open', PChar(AFile), ADir, AParams,
        SW_SHOWNORMAL) >= 32;
end;

```

The **ADir** parameter specifies the default directory for the shell operation. If **nil**, your application's current directory is used. If **AFile** specifies an executable file then **AParams** should contain the command line parameters to be passed to the application.

18. Drawing rotated text

```

uses
    Windows, Graphics;
...
procedure AngleTextOut(ACanvas: TCanvas; Angle, X, Y: integer; AStr: string);
var
    LogFont: TLogFont;
    hOldFont, hNewFont: HFONT;
begin
    GetObject(ACanvas.Font.Handle, SizeOf(LogFont), Addr(LogFont));
    LogFont.lfEscapement := Angle * 10;
    LogFont.lfOrientation := Angle * 10;
    hNewFont := CreateFontIndirect(LogFont);
    hOldFont := SelectObject(ACanvas.Handle, hNewFont);
    ACanvas.TextOut(X, Y, AStr);
    hNewFont := SelectObject(ACanvas.Handle, hOldFont);
    DeleteObject(hNewFont);
end;

```

19. Implementing a control array

```

...
const
    Count = 5;

type
    TForm1 = class(TForm)
        procedure FormCreate(Sender: TObject);
    public
        Edits: array[0..Count-1] of TEdit;
        procedure EditChange(Sender: TObject);
    end;
...

procedure TForm1.FormCreate(Sender: TObject);

```

```

var
  i: integer;
begin
  for i := 0 to Count-1 do begin
    Edits[i] := TEdit.Create(Self);
    with Edits[i] do begin
      Parent := Self;
      SetBounds(10, 10 + i*50, 200, 40);
      Tag := i; //each control should remember its index
      OnChange := EditChange; //same event handler for all controls
    end;
  end;
end;

procedure TForm1.EditChange(Sender: TObject);
var
  i: integer;
begin
  i := TEdit(Sender).Tag;
  ShowMessage(IntToStr(i));
end;

```

20. Converting between decimal, binary, and hexadecimal representation of a number

```

uses
  StrUtils, SysUtils;

function DecToBin(N: int64): string;
var
  i: integer;
  neg: boolean;
begin
  if N = 0 then begin
    result := '0';
    exit
  end;

  SetLength(result, SizeOf(N)*8);
  neg := N < 0;
  N := Abs(N);
  i := 1;
  while N <> 0 do begin
    if N and 1 = 1 then
      result[i] := '1'
    else
      result[i] := '0';
  end;

```

```

    N := N shr 1;
    Inc(i);
end;
if neg then begin
    result[i] := '—';
    Inc(i);
end;
Delete(result, i, length(result));
result := ReverseString(result);
end;

function DecToHex(N: int64): string;
begin
    if N < 0 then
        result := '—' + Format('%0x', [Abs(N)])
    else
        result := Format('%0x', [N]);
    end;
end;

```

21. Monitor the changes of clipboard's content

```

uses
    Windows, Forms, Clipbrd;

type
    TForm1 = class(TForm)
        Image1: TImage;
        procedure FormCreate(Sender: TObject);
    public
        procedure WMDrawClipboard(var Msg: TWMDrawClipBoard); message WM_DRAWCLIPBOARD;
    end;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
    SetClipboardViewer(Handle);
end;

procedure TForm1.WMDrawClipboard(var Msg: TWMDrawClipBoard);
begin
    if Clipboard.HasFormat(CF_BITMAP) then
        Image1.Picture.Assign(Clipboard);
    end;
end;

```

22. Detect the movement of a form

```

uses

```

```

Windows, Forms;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
  public
    procedure WMMove(var Msg: TWMMove); message WM_MOVE;
  end;
...

procedure TForm1.WMMove(var Msg: TWMMove);
begin
  Edit1.Text := IntToStr(Left);
  Edit2.Text := IntToStr(Top);
end;

```

23. Disabling the movement of a form

```

uses
  Windows, Forms;

type
  TForm1 = class(TForm)
  public
    procedure WMNHitTest(var Msg: TWMNHitTest); message WM_NCHITTEST;
  end;
...

procedure TForm1.WMNHitTest(var Msg: TWMNHitTest);
begin
  inherited;
  with Msg do
    if Result = HTCAPTION then Result := HTNOWHERE;
  end;
end;

```

24. Repositioning common dialogs

```

//create handler for OnShow event of common dialog
procedure TForm1.OpenDialog1Show(Sender: TObject);
var
  hwnd: THandle;
  rect: TRect;
  dlgWidth, dlgHeight: integer;
begin
  hwnd := GetParent(OpenDialog1.Handle);
  GetWindowRect(hwnd, rect);

```

```

    dlgWidth := rect.Right-rect.Left;
    dlgHeight := rect.Bottom-rect.Top;
    MoveWindow(hwnd, Left+(Width-dlgWidth) div 2, Top+(Height-dlgHeight) div 2,
        dlgWidth, dlgHeight, true);
    Abort;
end;

```

25. Avoid flickering in graphics programming

There are four ways to reduce flickering:

1. Use the `DoubleBuffered` property of `TWinControl` descendants: set `DoubleBuffered := true;`.
2. If your control is not transparent, include `csOpaque` in `ControlStyle`: `ControlStyle := ControlStyle + [csOpaque];`.
3. Handle the `WM_ERASEBKGD` Windows message and set `Msg.Result := 1;` in the handler.
4. Use off-screen bitmaps (like double-buffering, but works for any control)

Let's see an example of the last method:

```

uses
    Graphics;
...

procedure TForm1.Paint(Sender: TObject);
var
    bmp: TBitmap;
begin
    bmp := TBitmap.Create;
    bmp.Width := ClientWidth;
    bmp.Height := ClientHeight;
    //now draw on bmp.Canvas as you would do on TForm1.Canvas
    with bmp.Canvas do begin
        ...
    end;
    Canvas.Draw(0, 0, bmp);
    bmp.Free;
end;

```

26. Retrieve the path of the Windows directory

```

uses
    Windows, StrUtils;
...

function GetWinDir: string;

```

```

var
  buffer: array[0..MAX_PATH] of char;;
begin
  GetWindowsDirectory(buffer, MAX_PATH);
  result := string(buffer);
  if RightStr(result,1) <> '\' then result := result + '\';
end;

```

27. Using windows API in textbox programming

`Memo1.Perform(EM_LINEFROMCHAR, nCaret, 0)` returns zero-based index of the line which contains zero-based caret position `nCaret`. If `nCaret` is `-1` then the index of the current line is returned.

`Memo1.Perform(EM_LINEINDEX, nLine, 0)` returns caret position at the beginning of the line `nLine`. If `nLine` is `-1` then the beginning of the current line is returned.

`Memo1.Perform(EM_LINELENGTH, nCaret, 0)` returns the length of the line which contains caret position `nCaret`.

`Memo1.Perform(EM_SETMARGINS, EC_LEFTMARGIN or EC_RIGHTMARGIN, MAKELONG(wLeft,wRight))` sets the left and right margin of memo to `wLeft` and `wRight` pixels, respectively.

`Memo1.Perform(EM_CHARFROMPOS, 0, MAKELPARAM(x,y))` returns character index in the low-order word and the line index in the high-order word corresponding to the pixel position `(x,y)` in memo's client area.

28. Changing text alignment

By default, `TCanvas.TextOut(X,Y,Text)` puts the upper left corner of `Text` at `(X,Y)`.

First method:

```

uses
  Windows;
...
TForm1.Paint(Sender: TObject);
begin
  SetTextAlign(Canvas.Handle, TA_CENTER or TA_BOTTOM);
  Canvas.TextOut(100,100,'Hello World');
  //other possibilities are:
  //TA_BASELINE, TA_BOTTOM, TA_TOP,
  //TA_LEFT, TA_RIGHT
end;

```

Second method:

```

...
TForm1.Paint(Sender: TObject);
var
  Text: string;
begin
  Text := 'Hello World';
  with Canvas do

```

```

    TextOut(100 - TextWidth(Text) div 2, 100 - TextHeight(Text), Text);
end;

```

29. Drawing transparent text

The background of the text drawn onto a canvas is filled with the current brush.

```

Canvas.Brush.Style := bsClear;
Canvas.TextOut(100,100,'Hello World');

```

30. Delphi's equivalent of the VB's App object

Go to Project | Options and fill in the fields in Application page and Version info page.

```

unit AppUnit;

interface

type
    TApp = class
    protected
        //if the constructor is declared as protected then the
        //class can be instantiated only within this unit
        constructor Create;
    public
        Title: string;
        Path: string;
        ExeName: string;
        HelpFile: string;
        FileDescription: string;
        FileVersion: string;
        ProductName: string;
        ...
    end;

var
    App: TApp;

implementation

uses SysUtils, StrUtils, Forms, Windows, Dialogs;

type
    PLongInt = ^Longint;

constructor TApp.Create;

```

```

var
  nBytes, nInfoLen: DWORD;
  psBuffer: PChar;
  pntValue: Pointer;
  iLangID, iCharSetID: Word;
  strID: string;
begin
  ExeName := Application.ExeName;
  Path := ExtractFilePath(ExeName);
  if RightStr(Path,1) <> '\' then Path := Path + '\';
  Title := Application.Title;
  HelpFile := Application.HelpFile;

  nBytes := GetFileVersionInfoSize(PChar(ExeName), nBytes);
  if nBytes = 0 then begin
    ShowMessage('No version information available!');
    exit;
  end;
  psBuffer := AllocMem(nBytes + 1);
  GetFileVersionInfo(PChar(ExeName), 0, nBytes, psBuffer)
  VerQueryValue(psBuffer, '\VarFileInfo\Translation', pntValue, nInfoLen)
  iLangID := LoWord(PLongint(pntValue)^);
  iCharSetID := HiWord(PLongint(pntValue)^);
  strID := Format('\StringFileInfo\%.4x%.4x\',[iLangID, iCharSetID]);

  if VerQueryValue(psBuffer, PChar(strID + 'FileDescription'), pntValue, nInfoLen)
    then
    FileDescription := AnsiReplaceStr(PChar(pntValue), '\n', #13#10);
  if VerQueryValue(psBuffer, PChar(strID + 'FileVersion'), pntValue, nInfoLen) then
    FileVersion := PChar(pntValue);
  if VerQueryValue(psBuffer, PChar(strID + 'ProductName'), pntValue, nInfoLen) then
    ProductName := PChar(pntValue);
  FreeMem(psBuffer, nBytes);
end;

initialization

App := TApp.Create;

finalization

App.Free;

end.

```

31. Capture mouse clicks at statusbar

uses

Windows, Messages, Forms, Classes, SysUtils, Controls, ComCtrls, Commctrl;

type

```
TForm1 = class(TForm)
  StatusBar1: TStatusBar;
  procedure FormCreate(Sender: TObject);
private
  procedure WMNotify(var AMsg: TWMNotify); message WM_NOTIFY;
end;
```

...

procedure TForm1.FormCreate(Sender: TObject);

var

i: integer;

begin

for i := 0 to 3 do

with StatusBar1.Panels.Add do **begin**

Width := 100;

Text := Format('Panel %d',[i]);

end;

end;

procedure TForm1.WMNotify(**var** AMsg: TWMNotify);

var

pNMM: PNMMouse;

ctrl: TWinControl;

hWindow: HWND;

iPanel: integer;

begin

inherited;

pNMM := PNMMouse(AMsg.NMHdr);

hWindow := pNMM^.hdr.hwndFrom;

ctrl := FindControl(hWindow);

iPanel := pNMM^.dwItemSpec;

case pNMM^.hdr.code **of**

NM_CLICK:

Caption := Format('%s was clicked at panel %d',[ctrl.Name,iPanel]);

NM_DBLCLK:

Caption := Format('%s was double-clicked at panel %d',[ctrl.Name,iPanel]);

NM_RCLICK:

Caption := Format('%s was right-clicked at panel %d',[ctrl.Name,iPanel]);

NM_RDBLCLK:

```

        Caption := Format('%s was right-double-clicked at panel %d',[ctrl.Name,iPanel
    ]);
end;
end;

```

32. Find if you are connected to the Internet

```

uses IWinInet;
...

procedure CheckConnection;
var
    dwFlags: DWORD;
begin
    if InternetGetConnectedState(@dwFlags, 0) then begin
        if (dwFlags and INTERNET_CONNECTION_MODEM)=INTERNET_CONNECTION_MODEM then
            ShowMessage('Connected through modem')
        else if (dwFlags and INTERNET_CONNECTION_LAN) = INTERNET_CONNECTION_LAN then
            ShowMessage('Connected through LAN')
        else if (dwFlags and INTERNET_CONNECTION_PROXY) = INTERNET_CONNECTION_PROXY then
            ShowMessage('Connected through Proxy')
        else if (dwFlags and INTERNET_CONNECTION_MODEM_BUSY) =
            INTERNET_CONNECTION_MODEM_BUSY then
            ShowMessage('Modem is busy');
        end else
            ShowMessage('Offline');
    end;

```

33. Create form with rounded corners

```

procedure TForm1.FormCreate(Sender: TObject);
var
    rgn: HRGN;
begin
    GetWindowRgn(Handle, rgn);
    DeleteObject(rgn);
    // set the radius of corners to 100px
    rgn:= CreateRoundRectRgn(0, 0, Width, Height, 100, 100);
    SetWindowRgn(Handle, rgn, TRUE);
end;

```

34. Make transparent form

```

uses
    Windows, Forms, Classes, Controls, ComCtrls;

type
    TForm1 = class(TForm)

```

```

    TrackBar1: TTrackBar;
    procedure TrackBar1Change(Sender: TObject);
end;
...

procedure SetTransparent(hWnd: longint; value: Byte);
// opaque: value=255; fully transparent: value=0
var
    iExStyle: Integer;
begin
    iExStyle := GetWindowLong(hWnd, GWL_EXSTYLE);
    if value < 255 then begin
        iExStyle := iExStyle Or WS_EX_LAYERED;
        SetWindowLong(hWnd, GWL_EXSTYLE, iExStyle);
        SetLayeredWindowAttributes(hWnd, 0, value, LWA_ALPHA);
    end else begin
        iExStyle := iExStyle xor WS_EX_LAYERED;
        SetWindowLong(hWnd, GWL_EXSTYLE, iExStyle);
    end;
end;

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
    SetTransparent(Handle, TrackBar1.Position);
end;

```

35. Save graphics to a bitmap file or a Windows Metafile

```

uses
    Graphics;
...

procedure SaveToBitmapFile(AFile: TFilename);
var
    bmp: TBitmap;
begin
    bmp := TBitmap.Create;
    bmp.Width := 400;
    bmp.Height := 400;
    with bmp.Canvas do begin
        //drawing commands
    end;
    bmp.SaveToFile(AFile);
    bmp.Free;
end;

procedure SaveToMetafile(AFile: TFilename);

```

```

var
    emf: TMetafile;
    mfc: TMetafileCanvas;
begin
    emf := TMetafile.Create;
    emf.Enhanced := true; // save as Enhanced Metafile
    emf.Width := 400;
    emf.Height := 400;
    mfc := TMetafileCanvas.Create(emf, 0);
    with mfc do begin
        //drawing commands
    end;
    mfc.Free;
    emf.SaveToFile(AFile);
    emf.Free;
end;

```

36. Detect when a form has been minimized, maximized or restored

```

TForm1 = class(TForm)
...
procedure WMSize(var Msg: TWMSize); message WM_SIZE;
end;
...

procedure TForm1.WMSize(var Msg: TWMSize);
begin
    inherited; // otherwise TForm1.Resize is not fired
    if Msg.SizeType = SIZE_MAXIMIZED then
        ...
    else if Msg.SizeType = SIZE_RESTORED then
        ...
    else if Msg.SizeType = SIZE_MINIMIZED then
        ...
end;

```

37. Prevent a screensaver to kick in as long as your application is running

```

uses
    Windows;
...

procedure AppMessage(var Msg: TMsg; var Handled: Boolean);
begin
    if (Msg.Message = WM_SYSCOMMAND) and ((Msg.wParam = SC_SCREENSAVE) or (Msg.wParam
        = SC_MONITORPOWER)) then
        Handled := true;

```

```

end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    Application.OnMessage := AppMessage;
end;

```

38. Installing a font on the fly

Assume that a font file `MyFont.ttf` resides in application directory.

```

uses
    Windows, Forms, SysUtils;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
    AddFontResource(PChar(ExtractFilePath(Application.ExeName) + '\MyFont.ttf'));
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
    RemoveFontResource(PChar(ExtractFilePath(Application.ExeName) + '\MyFont.ttf'));
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;

```

39. Using WinHelp API

Implementing your own “What’s this?” button

```

uses
    Windows;
...

procedure TForm1.btnHelpClick(Sender: TObject);
begin
    PostMessage(Handle, WM_SYSCOMMAND, SC_CONTEXTHELP, 0);
end;

```

40. Listing system’s drives in a listbox

```

uses
    Windows;

function GetVolumeName(aRoot: string): string;
var

```

```

VolumeSerialNumber : DWORD;
MaximumComponentLength : DWORD;
FileSystemFlags : DWORD;
VolumeNameBuffer : array[0..255] of char;
begin
  GetVolumeInformation(PChar(aRoot),
    VolumeNameBuffer,
    sizeof(VolumeNameBuffer),
    @VolumeSerialNumber,
    MaximumComponentLength,
    FileSystemFlags,
    nil,
    0);
  result := string(VolumeNameBuffer);
end;

function DiskInDrive(aDrive: char): boolean;
var
  vErrMode: word;
begin
  if aDrive in ['a'..'z'] then Dec(aDrive, $20);
  if not (aDrive in ['A'..'Z']) then
    raise Exception.Create('Not a valid drive letter');
  vErrMode := SetErrorMode(SEM_FAILCRITICALERRORS);
  try
    result := (DiskSize(Ord(aDrive) - $40) <> -1)
  finally
    SetErrorMode(vErrMode);
  end;
end;

procedure TMainForm.FormCreate(Sender: TObject);
var
  drv: char;
begin
  for drv := 'a' to 'z' do
    case GetDriveType(PChar(drv + ':\')) of
      DRIVE_REMOVABLE, DRIVE_FIXED, DRIVE_CDROM:
        if DiskInDrive(drv) then
          ListBox1.Items.Add(UpperCase(drv) + ': (' + GetVolumeName(drv + ':\') + ')')
        else
          ListBox1.Items.Add(UpperCase(drv) + ':');
        end;
    end;
end;

```