

## Re: mobile phone logo editor

**Source:** <http://coding.derkeiler.com/Archive/Delphi/alt.comp.lang.borland-delphi/2004-03/0491.html>

---

**From:** Supply and Demand (*\_at\_*)

**Date:** 03/23/04

Date: Tue, 23 Mar 2004 19:36:16 GMT

Hi!

I have some ideas on how you can implement some of the things you need :)

Test the code below before you try to understand it... See if it's ok!

I have written some code under the OnCreate, OnMouseUp and OnPaint events of the form, which is, by the way, called Form1 in this code...

You ONLY have to have a form – no buttons or similar...

So, all you need is to copy this code to the proper events, replace the Form1-class declaration with the one I wrote and finally add the constants as shown below!

Drawing on the screen is done by clicking on a pixel... You can't draw lines, circles, flood fill or anything like that!

If you need help with that – I'd love to help! I already have the code for flood fills of objects, line drawing, etc... So

write to me if you need that.

You can load and save a 200x100 pixel bitmap if you right-click on the form (as you can see in the code).

Try the code and write back to me! I'd like to know what you think of it! :)

Have fun!

```
// == THE CODE ==
```

```
unit Unit1;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
  Dialogs, StdCtrls;
```

```
const
```

```
  BOX_SIZE: Integer = 5; // Each pixel will be 5x5 pixels large on the  
  form.
```

```
  BOXES_X : Integer = 200; // Amount of horizontal pixels!
```

```
BOXES_Y : Integer = 100; // Vertical pixels.
```

```
PIXEL_ON: Integer = 1; // Used when drawing the grid.
```

```
PIXEL_OFF: Integer = -1; // To toggle a pixel between on or off,
```

```
    // you only have to multiply that value by -1!
```

```
    // I.e. if the pixel is PIXEL_ON then
```

```
PIXEL_ON*-1 will
```

```
    // produce the result PIXEL_OFF and vice versa.
```

```
type
```

```
  TForm1 = class(TForm)
```

```
    procedure FormMouseUp(Sender: TObject; Button: TMouseButton;
```

```
      Shift: TShiftState; X, Y: Integer);
```

```
    procedure FormCreate(Sender: TObject);
```

```
    procedure FormPaint(Sender: TObject);
```

```
  private
```

```
    screen: Array[1..200,1..100] of Integer;
```

```
    procedure DrawGrid; // Draw the pixel matrix!
```

```
    procedure ClearScreen; // Clear the matrix.
```

```
    procedure SaveGrid(FileName: String); // Save it as a bitmap file with a  
certain name.
```

```
    procedure LoadGrid(FileName: String); // Load a 200x100-pixel bitmap  
file.
```

```
  public
```

```
    { Public declarations }
```

```
  end;
```

```
var
```

```
  Form1: TForm1;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TForm1.ClearScreen;
```

```
var
```

```
  x,y: Integer;
```

```
begin
```

```
  for y := 1 to 100 do
```

```
    for x := 1 to 200 do
```

```
      screen[x,y] := PIXEL_OFF;
```

```
  DrawGrid;
```

```
end;
```

```
procedure TForm1.DrawGrid;
```

```
var
```

```
  x,y: Integer;
```

```
begin
```

```
  // Make sure that the form remains in the size that is necessary for
```

```
nicely
```

```
// drawing the grid... (Form1.Constraints could be used instead...).
if Form1.ClientWidth <> BOX_SIZE*BOXES_X then
  Form1.ClientWidth := BOX_SIZE*BOXES_X;
if Form1.ClientHeight <> BOX_SIZE*BOXES_Y then
  Form1.ClientHeight := BOX_SIZE*BOXES_Y;

// Draw the grid and the pixels!
For y := 1 to BOXES_Y do
  for x := 1 to BOXES_X do
  begin
    // Grid code.
    Form1.Canvas.MoveTo((x-1)*BOX_SIZE,0);
    Form1.Canvas.LineTo((x-1)*BOX_SIZE,Form1.ClientHeight-1);
    Form1.Canvas.MoveTo(0,(y-1)*BOX_SIZE);
    Form1.Canvas.LineTo(Form1.ClientWidth-1,(y-1)*BOX_SIZE);

    // The actual code for drawing the pixels.
    if screen[x,y] = PIXEL_ON then
      Form1.Canvas.Brush.Color := clBlack
    else // Pixel is off!
      Form1.Canvas.Brush.Color := clBtnFace;

    // Draw a "box" - the color decides whether or not the pixel is on or
    off.
    Form1.Canvas.FillRect(Rect(((x-1)*BOX_SIZE+1),
      ((y-1)*BOX_SIZE+1),
      (x*BOX_SIZE),
      (y*BOX_SIZE)));
  end;
end;

// OnMouseUp-event.
procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  selX,selY: Integer;
begin
  if Button = mbLeft then
  begin
    selX := X div BOX_SIZE + 1;
    selY := Y div BOX_SIZE + 1;

    screen[selX,selY] := screen[selX,selY]*-1; // Invert the pixel!

    DrawGrid;
  end
  else
  if InputBox('(S)ave or (L)oad the grid?','S/L','S') = 'S' then
    SaveGrid(InputBox('Save the grid as','File name','c:\test.bmp'))
  else
    LoadGrid(InputBox('Load the grid','File name','c:\test.bmp'));
```

```
end;

// OnCreate-event.
procedure TForm1.FormCreate(Sender: TObject);
begin
  ClearScreen;
end;

// The OnPaint-event!
procedure TForm1.FormPaint(Sender: TObject);
begin
  DrawGrid; // Redraw the grid when the form redraws itself...
end;

procedure TForm1.LoadGrid(FileName: String);
var
  x,y: Integer;
  pic: TPicture;
begin
  // This looks like the SaveGrid-procedure so take a look at it for further
  explanations!

  pic := TPicture.Create;

  pic.LoadFromFile(FileName);

  if (pic.Bitmap.Height = 100) and (pic.Bitmap.Width = 200) then
  begin
    pic.Bitmap.Monochrome := true;
    for y := 1 to 100 do
      for x := 1 to 200 do
        if pic.Bitmap.Canvas.Pixels[x-1,y-1] = clWhite then // Read all
pixels from the bitmap
          screen[x,y] := PIXEL_OFF // and put them
on our grid.
        else
          screen[x,y] := PIXEL_ON;
        end
      end
    else
      ShowMessage('You may only load 200x100 pixel bitmaps!');

    pic.Free;
    DrawGrid;
  end;

procedure TForm1.SaveGrid(FileName: String);
var
  x,y: Integer;
  pic: TPicture;
  bw: Array[-1..1] of TColor;
begin
```

```
// I have decided to create an instance of a TPicture and then I copy all
of the pixels
// from our grid to the TPicture's bitmap grid and finally I'll use
SaveToFile to save it!

bw[PIXEL_OFF] := clWhite; // Each pixel that is "off" will have this color
in the BMP-file.
bw[PIXEL_ON] := clBlack;

pic := TPicture.Create;

pic.Bitmap.Height := 100;
pic.Bitmap.Width := 200;

pic.Bitmap.Monochrome := true; // Monochrome bitmap (1 bit per pixel).

for y := 1 to 100 do
  for x := 1 to 200 do
    pic.Bitmap.Canvas.Pixels[x-1,y-1] := bw[screen[x,y]];
    // Note for the line above: if the value at screen[x,y] is PIXEL_OFF
then the
    // pixel at (x,y) will get the color bw[screen[x,y]] = bw[PIXEL_OFF] =
clWhite.
    // I hope it makes sense :)
    // Also - the canvas counts from the coordinate 0,0 to 199,99... and
we
    // counted from 1,1 to 200,100 which is why we need that [x-1,y-1].
    // You can change the screen-array to Array[0..199,0..99] of Integer
if you want to.

pic.SaveToFile(FileName);

pic.Free;
end;

end.
// == END OF CODE ==

"Jörg Schneider" <ng1.5.nekratog@spamgourmet.com> wrote in message
news:c3mo6g$2ahut9$1@ID-72028.news.uni-berlin.de...
> I'd like to write the following program:
> - a simple paint application like a mobile phone logo editor
> - display (canvas) size is about 200x100 Pixels and should be displayed
> as a grid
> - painting should be possible with different tools (brush, circle, square
> etc)
> - 1Bit-display (black/white) is enough
> - saving/loading of images, preferably in bmp, but any other format is ok.
>
> My first thought was to use one TPanel per pixel and keep them in an
array,
```

alt.comp.lang.borland-delphi: Re: mobile phone logo editor

- > *but the handling is pretty cumbersome (e.g. when you hold the mouse button*
- > *to draw a line from the upper left to the lower right)...*
- >
- > *Are there any components (especially for the display and/or the tools) or*
- > *complete applications with source for this task?*
- >
- > *Thanks in advance,*
- > *Jörg*
- >
- > *F'up2 alt.comp.lang.borland-delphi*
- >
- >