

Mini Paint en Delphi 6

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls, Menus, ExtDlgs, Math, ComCtrls, Buttons,  
ColorGrd;
```

```
type
```

```
TForm1 = class(TForm)  
    Panel1: TPanel;  
    RadioGroup1: TRadioGroup;  
    ScrollBox1: TScrollBox;  
    Image1: TImage;  
    OpenPictureDialog1: TOpenPictureDialog;  
    MainMenu1: TMainMenu;  
    File1: TMenuItem;  
    New1: TMenuItem;  
    N1: TMenuItem;  
    Open1: TMenuItem;  
    SaveAs: TMenuItem;  
    Save: TMenuItem;
```

N2: TMenuItem;

Exit1: TMenuItem;

SavePictureDialog1: TSavePictureDialog;

Image2: TImage;

Edit1: TEdit;

Button1: TButton;

Label1: TLabel;

Edit2: TEdit;

Label2: TLabel;

UpDown2: TUpDown;

UpDown1: TUpDown;

ColorGrid1: TColorGrid;

Edit3: TEdit;

labelFuente: TLabel;

FontBox: TComboBox;

Panel2: TPanel;

ComboBox1: TComboBox;

Label3: TLabel;

Label4: TLabel;

FontDialog1: TFontDialog;

SpeedButton1: TSpeedButton;

BitBtn1: TBitBtn;

procedure Open1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormMouseDown(Sender: TObject);

```
        Button: TMouseButton;

        Shift: TShiftState;

        X, Y: Integer);

procedure FormMouseUp(Sender: TObject;

        Button: TMouseButton;

        Shift: TShiftState;

        X, Y: Integer);

procedure FormMouseMove(Sender: TObject;

        Shift: TShiftState;

        X, Y: Integer);

procedure SaveAsClick(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure UpDown2Click(Sender: TObject; Button: TUDBtnType);

procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);

procedure Exit1Click(Sender: TObject);

procedure ColorGrid1Change(Sender: TObject);

procedure New1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Edit3Exit(Sender: TObject);

procedure Edit3KeyPress(Sender: TObject; var Key: Char);

procedure FontBoxChange(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure ComboBox1Change(Sender: TObject);

private
```

```
{ Private declarations }

public
{ Public declarations }

end;

const
kPencil = 0;
kBrush = 1;
kLine = 2;
kEraser = 3;
kRectangle = 4;
kEllipse = 5;
kRellenar = 6;
kTexto = 7;
kFillRect = 8;

var
Form1: TForm1;
Drawing, buffered, bandera: boolean;
Startx, Starty, Lastx, Lasty, anchoBorde: Integer;
zoom: Real;
ImageBuffer: Tbitmap;
r1, r2: TRect;
textbox: TEdit;
fgColor, bgColor: Tcolor;
tam: integer;
```

fuelle: Tfont;

archivo : TFileName;

implementation

uses Types;

{SR *.dfm}

procedure TForm1.Open1Click(Sender: TObject);

begin

With OpenPictureDialog1 do

if Execute then

Image1.Picture.LoadFromFile(FileName);

end;

procedure TForm1.SaveAsClick(Sender: TObject);

begin

if bandera then

begin

With SavePictureDialog1 do

if Execute then

begin

Image1.Picture.SaveToFile(FileName);

archivo := FileName;

bandera := false;

```
        end;
    end
    else
        Image1.Picture.SaveToFile(archivo);
    end;

procedure TForm1.FormCreate(Sender: TObject);

var
    cnt : integer;

begin
    bandera := true;
    Image1.Canvas.Brush.Color := clWhite;
    Image1.Canvas.Pen.Color := clBlack;
    Form1.DoubleBuffered := true;
    form1.ComboBox1.ItemIndex := 1;
    for cnt := 0 to Screen.Fonts.Count-1 do
        FontBox.Items.Add(Screen.Fonts.Strings[cnt]);
    end;
```

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);

begin
    // Marca el inicio del trazo
    Drawing := True;
```

```
//Determina que herramienta se comenzará a utilizar
```

```
case RadioGroup1.ItemIndex of
```

```
  kPencil:
```

```
    begin
```

```
      Lastx:= x;
```

```
      Lasty:= y;
```

```
      Image1.Canvas.Pen.Width := tam;
```

```
      Image1.Canvas.MoveTo(x,y);
```

```
    end;
```

```
  kBrush:
```

```
    begin
```

```
      Lastx:= x;
```

```
      Lasty:= y;
```

```
      Image1.Canvas.Pen.Width := tam;
```

```
      Image1.Canvas.MoveTo(x,y);
```

```
    end;
```

```
  kLine:
```

```
    begin
```

```
      screen.Cursor := crCross;
```

```
      Startx:= x;
```

```
      Starty:= y;
```

```
      Lastx:= x;
```

```
      Lasty:= y;
```

```
      Image1.Canvas.Pen.Width := tam;
```

```
      Image1.Canvas.Pen.Style:= psSolid;
```

```
      Image1.Canvas.Pen.Mode:=pmNotXor;
```

```
    Image1.Canvas.MoveTo(x,y);

    Image1.Canvas.LineTo(x,y);

end;

kRectangle:

begin

    screen.Cursor := crCross;

    // Marca el comienzo del rectángulo que se dibujará

    Startx:= x;

    Starty:= y;

    Lastx:= x;

    Lasty:= y;

    Image1.Canvas.Pen.Width := tam;

    Image1.Canvas.Pen.Style := psSolid;

    Image1.Canvas.Brush.Style := bsClear;

    Image1.Canvas.Pen.Mode := pmCopy;

end;

kFillRect:

begin

    screen.Cursor := crCross;

    // Marca el comienzo del rectángulo que se dibujará

    Startx:= x;

    Starty:= y;

    Lastx:= x;

    Lasty:= y;

    Image1.Canvas.Pen.Width := tam;

    Image1.Canvas.Pen.Style := psSolid;
```

```
Image1.Canvas.Brush.Style := bsSolid;  
Image1.Canvas.Pen.Mode := pmCopy;  
end;
```

kEllipse:

```
begin  
    screen.Cursor := crCross;  
  
    Startx:= x;  
  
    Starty:= y;  
  
    Lastx:= x;  
  
    Lasty:= y;  
  
    Image1.Canvas.Pen.Width := tam;  
  
    Image1.Canvas.Pen.Style := psSolid;  
  
    Image1.Canvas.Brush.Style := bsClear;  
  
    Image1.Canvas.Pen.Mode := pmCopy;  
  
end;
```

kRellenar:

```
begin  
    with image1.Canvas do  
        begin  
            Brush.Color := fgColor;  
  
            Pen.Width := tam;  
  
            Pen.Style := psSolid;  
  
            Brush.Style := bsSolid;  
  
            Pen.Mode := pmCopy;  
  
            FloodFill(x, y, Pixels[x,y], fsSurface);  
        end  
    end  
end;
```

```
        Brush.Color := bgColor;

    end;

end;

kTexto:

begin

    Edit3.Text := 'Texto';

    Edit3.Font := fuente;

    Edit3.Visible := true;

    Edit3.Top := Panel2.Height+y;

    Edit3.Left := x;

    Edit3.SetFocus;

end;

end;

end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);

begin

    // Marca el fin del trazo

    Drawing := false;

    buffered := false;

    screen.Cursor := crDefault;

    Image1.Canvas.Pen.Color := fgColor;
```

```
case RadioGroup1.ItemIndex of
```

```
  kLine:
```

```
    begin
```

```
      Image1.Canvas.Pen.Width := tam;
```

```
      Image1.Canvas.Pen.Style:= psSolid;
```

```
      Image1.Canvas.Pen.Mode:=pmCopy;
```

```
      Image1.Canvas.MoveTo(StartX,StartY);
```

```
      Image1.Canvas.LineTo(x,y);
```

```
    end;
```

```
end;
```

```
end;
```

```
procedure Capturar(startx, starty, x, y: integer);
```

```
var
```

```
  ab : integer;
```

```
begin
```

```
  buffered := true;
```

```
  anchoBorde := Form1.Image1.Canvas.Pen.Width;
```

```
  ab := anchoBorde*2;
```

```
  ImageBuffer := Tbitmap.Create;
```

```
  ImageBuffer.width := abs(x-startx) + (ab*2);
```

```
  ImageBuffer.height := abs(y-starty) + (ab*2);
```

```
  with r1 do
```

```
    begin
```

```

        if startx < x then begin left:=startx-ab; right:=x+ab end
            else begin left:=x-ab; right:=startx+ab end;
        if starty < y then begin top:=starty-ab; bottom:=y+ab end
            else begin top:=y-ab; bottom:=starty+ab end;
    end;
with r2 do
    begin
        left:=0; top:=0; right:=r1.right-r1.left; bottom:=r1.Bottom-r1.Top;
    end;
    ImageBuffer.Canvas.CopyRect(r2, Form1.Image1.canvas, r1);
end;

```

```

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);

```

```

var

```

```

    SourceRect, DestRect: TRect;

```

```

begin

```

```

    // Toma las coordenadas de la imagen principal para hacer un zoom

```

```

    SourceRect.Left := x - Image2.Width div UpDown2.Position;

```

```

    SourceRect.Top := y - Image2.Height div UpDown2.Position;

```

```

    SourceRect.Right := x + Image2.Width div UpDown2.Position;

```

```

    SourceRect.Bottom := y + Image2.Height div UpDown2.Position;

```

```

    With Image2 do

```

```
DestRect := Rect(0, 0, Width, Height);

//Copia el fragmento de la imagen principal a la imagen de zoom
// la diferencia entre sourcerect y destrect provocan el efecto zoom
Image2.Canvas.CopyRect(DestRect, Image1.Canvas, SourceRect);

// Verifica si se está dibujando
if Drawing then

// Determina que herramienta se está utilizando
case RadioGroup1.ItemIndex of
kBrush:
begin
    // Utiliza elipses para dar efecto de brocha
    Image1.Canvas.Pen.Mode:= pmCopy; //pmCopy mezcla lo que se dibuja
    Image1.Canvas.Pen.Style := psClear;
    Image1.Canvas.Brush.Color := fgColor;

    Image1.Canvas.Ellipse(x-tam,
        y-tam,
        x+tam,
        y+tam);
end;
kEraser:
begin
    Image1.Canvas.Pen.Mode:= pmWhite;
```

```

Image1.Canvas.Pen.Style := psClear;

Image1.Canvas.Rectangle(x-tam,
                        y-tam,
                        x+tam,
                        y+tam);

end;

kPencil:
begin
    Image1.Canvas.Pen.Style := psDash;
    Image1.Canvas.LineTo(Lastx, Lasty);
    Lastx := x;
    Lasty := y;
end;

kLine:
begin
    Image1.Canvas.MoveTo(Startx, Starty);
    Image1.Canvas.LineTo(Lastx, Lasty);
    Image1.Canvas.MoveTo(Startx, Starty);
    Image1.Canvas.LineTo(x,y);
    Lastx :=X;
    Lasty :=Y;
end;

kRectangle:
begin
    // Arrastra el rectángulo mientras el botón del mouse sigue oprimido

```

```

// Va eliminando el rectángulo anterior
//ShowMessage(IntToStr(ImageBuffer.Width));
if buffered then
    begin
        Image1.Canvas.CopyRect(r1, ImageBuffer.Canvas, r2);
        ImageBuffer.Free;
        buffered := false;
    end;

Capturar(Startx, Starty, x, y);
// Dibuja el nuevo rectángulo
Image1.Canvas.Pen.Mode:=pmCopy;
Image1.Canvas.Rectangle(Startx, Starty, x,y);
Lastx :=X;
Lasty :=Y;
end;

kFillRect:
begin
    // Arrastra el rectángulo mientras el botón del mouse sigue oprimido
    // Va eliminando el rectángulo anterior
    //ShowMessage(IntToStr(ImageBuffer.Width));
    if buffered then
        begin
            Image1.Canvas.CopyRect(r1, ImageBuffer.Canvas, r2);
            ImageBuffer.Free;
            buffered := false;
        end;

```

```
Capturar(Startx, Starty, x, y);

// Dibuja el nuevo rectángulo

Image1.Canvas.Pen.Mode:=pmCopy;

Image1.Canvas.Rectangle(Startx, Starty, x,y);

Lastx :=X;

Lasty :=Y;

end;

kElipse:

begin

    if buffered then

        begin

            Image1.Canvas.CopyRect(r1, ImageBuffer.Canvas, r2);

            ImageBuffer.Free;

            buffered := false;

        end;

        Capturar(Startx, Starty, x, y);

        // Dibuja el nuevo rectángulo

        Image1.Canvas.Pen.Mode:=pmCopy;

        Image1.Canvas.Ellipse(Startx, Starty, x,y);

        Lastx :=X;

        Lasty :=Y;

    end;

end;

end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    //Muestra el cuadro de dialogo de selección de color y le asigna el color
    //  seleccionado a brush y pen
    with TColorDialog.Create(self) do
        if Execute then
            fgColor := Color;
            image1.Canvas.Pen.Color := fgColor;
    end;
```

```
procedure TForm1.UpDown2Click(Sender: TObject; Button: TUDBtnType);
begin
    Edit2.Text := IntToStr(UpDown2.Position);
end;
```

```
procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);
begin
    // Escribe en el cuadro de texto la posición de UpDown
    Edit1.Text := IntToStr(UpDown1.Position);
    tam := UpDown1.Position;
end;
```

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
```

```
//Cierra la ventana y termina el programa  
Close;  
Exit;  
end;  
  
procedure TForm1.ColorGrid1Change(Sender: TObject);  
begin  
    //Asigna el color seleccionado en ColorGrid a brush y pen  
    fgColor := ColorGrid1.ForegroundColor;  
    bgColor := ColorGrid1.BackgroundColor;  
    image1.Canvas.Brush.Color := ColorGrid1.BackgroundColor;  
    Image1.Canvas.Pen.Color := ColorGrid1.ForegroundColor;  
end;  
  
procedure TForm1.New1Click(Sender: TObject);  
begin  
    // Dibuja un rectángulo blanco en todo el fondo del lienzo  
    with image1, canvas do  
        begin  
            brush.color:=clwhite;  
            pen.Color := clWhite;  
            canvas.rectangle(clientrect);  
        end;  
end;
```

```

        ColorGrid1Change(Sender);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    //Pone el color de brush y pen al color de fondo seleccionado
    image1.Canvas.Brush.Color := ColorGrid1.BackgroundColor;
    Image1.Canvas.Pen.Color := Image1.Canvas.Brush.Color;
    //Pinta el fondo
    image1.Canvas.Rectangle(0,0,ClientWidth, ClientHeight);
    //Regresa brush y pen a su color
    image1.Canvas.Brush.Color := bgColor;
    Image1.Canvas.Pen.Color := fgColor;;
end;

procedure DibujarTexto;
var
    x, y : integer;
begin
    form1.Image1.Canvas.Font.Color := form1.Image1.Canvas.Pen.Color;
    form1.Image1.Canvas.Brush.Color := form1.Image1.Canvas.Brush.Color;
    form1.Image1.Canvas.Font := fuente;

    with Form1.Edit3 do
    begin
        x := Left;

```

```
    y := Top-form1.Panel2.Height;

    //y := Top;

    form1.Image1.Canvas.TextOut(x,y,Text);

    form1.Edit3.Visible := false;

end;

end;

procedure TForm1.Edit3Exit(Sender: TObject);

begin
    DibujarTexto;

end;

procedure TForm1.Edit3KeyPress(Sender: TObject; var Key: Char);

begin
    if key = chr(13) then
        begin
            DibujarTexto;

        end;

end;

procedure TForm1.FontBoxChange(Sender: TObject);

begin
    fuente.Name := form1.FontBox.Items[form1.FontBox.ItemIndex];
```

```
    form1.Edit3.Font.Name := fuente.Name;
end;

procedure TForm1.Edit1Change(Sender: TObject);
begin
    if not (Edit1.text = "") then
        tam := StrToInt(Edit1.Text);
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
    with FontDialog1 do
        if Execute then
            begin
                fuente := Font;
                FontBox.ItemIndex := FontBox.Items.IndexOf(fuente.Name);

                ComboBox1.Text := IntToStr(fuente.Size);
            end;
end;

procedure TForm1.ComboBox1Change(Sender: TObject);
begin

    fuente.Size := StrToInt(ComboBox1.Items[ComboBox1.ItemIndex]);
```

end;

initialization

buffered := false;

fuelle := TFont.Create;

finalization

ImageBuffer.free;

end.