이미지를 내 맘대로 다뤄보자

#1. 간단한 이미지 툴 제작하기

2007-03-17

강사 : 조무영

www.delphischool.co.kr

이미지를 내맘대로 다뤄보자

#1 : 간단한 이미지 툴 제작하기

목차

1. 폼 UI 설계

1) UI 영역 구분 및 기능

① 메뉴 영역 (프로그램 상단)

☑ 파일 불러오기, 저장하기

② 툴바 영역 (프로그램 좌측)

☑ 편집 기능 선택

③ 레이어 목록 영역 (프로그램 우측)

☑ 레이어 추가, 삭제

☑ 편집 레이어 선택

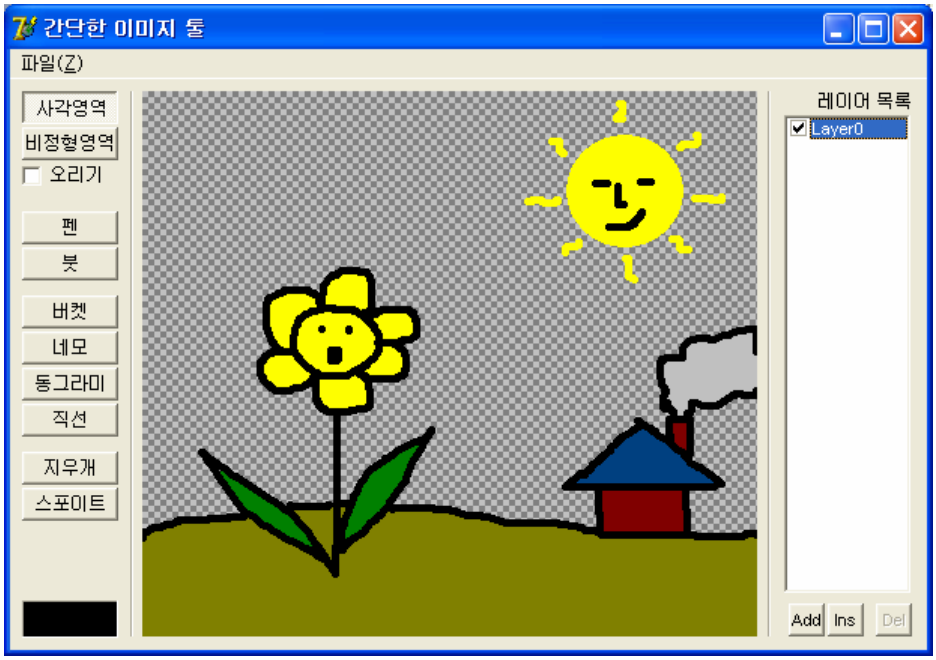
☑ 레이어 visible 속성 제어

④ 이미지 편집 화면 영역 (프로그램 중앙)

☑ 이미지 편집

2) UI 구성 예

① 이미지 툴 UI 구성 예



2. 이미지 레이어

1) TImageLayer 클래스

① 요구사항

☑ Bitmap 을 가지고 있어야 한다.

☑ Canvas 를 사용할 수 있어야 한다.

☑ Visible 속성을 가져야 한다.

☑ Bitmap 에 변경사항이 발생하면 이벤트 핸들러를 호출해야 한다.

☑ 크기 변경에 따라 Bitmap 의 크기를 자동으로 갱신해야 한다.

☑ 다른 Canvas 에 자신을 그리는 함수를 제공해야 한다.

☑ 투명색을 지원해야 한다.

② 설계

☑ 선언부

```
TImageLayer = class(TObject)
  private
    fBitmap: TBitmap;
    fVisible: Boolean;
    fOnUpdate: TNotifyEvent
    procedure BitmapChanged(Sender: TObject);
    function GetCanvas: TCanvas;
    procedure SetVisible(Value: Boolean);
    property OnUpdate: TNotifyEvent read fOnUpdate write fOnUpdate;
  protected
  public
    constructor Create(W, H: Integer);
    destructor Destroy; override;
    procedure Clear;
    procedure PaintTo(DstCanvas: TCanvas; SrcRect, DstRect: TRect);
    property Canvas: TCanvas read GetCanvas;
    property Bitmap: TBitmap read fBitmap;
    property Visible: Boolean read fVisible write SetVisible;
  end;
```

③ 구현

☑ 공통 상수

```
const
  ColorKey        = $00000001;
  ColorKeyConvert = $00000000;
```

☑ 초기화, 해제

```
constructor TImageLayer.Create(W, H: Integer);
begin
  fBitmap:=TBitmap.Create; // 비트맵을 만들고...
  with fBitmap do
  begin
```

```
      Width :=W;
      Height:=H;
      PixelFormat:=pf24bit;
Canvas.Brush.Color:=ColorKey;
      Canvas.Brush.Style:=bsSolid;
      Canvas.CopyMode:=cmSrcCopy;
      OnChange:=BitmapChanged;
   end;
   fVisible:=True;
   Clear;
end;

destructor TImageLayer.Destroy;
begin
```

☑ 프로퍼티

```
function TImageLayer.GetCanvas: TCanvas;
begin
   Result:=fBitmap.Canvas;
end;

procedure TImageLayer.SetVisible(Value: Boolean);
begin
   if fVisible<>Value then
   begin
      fVisible:=Value;
      BitmapChanged(Self);
   end;
end;
```

☑ 이벤트 핸들러

```
procedure TImageLayer.BitmapChanged(Sender: TObject);
begin
   if Assigned(fOnUpdate) then fOnUpdate(Sender);
end;
```

☑ 그리기

```
procedure TImageLayer.Clear; // 이미지 지우기
var
   PrevStyle: TBrushStyle;
   PrevColor: TColor;
begin
   with fBitmap, Canvas do
   begin
      PrevStyle:=Brush.Style;
      PrevColor:=Brush.Color;
      Brush.Style:=bsSolid;
      Brush.Color:=ColorKey;
      FillRect(Rect(0, 0, Width, Height));
```

```
      Brush.Color:=PrevColor;
      Brush.Style:=PrevStyle;
    end;
end;

// 어딘가에 이미지 그리기
procedure TImageLayer.PaintTo(DstCanvas: TCanvas; SrcRect, DstRect: TRect);
var
  PrevStyle: TBrushStyle;
begin
  with DstCanvas do
  begin
    PrevStyle:=Brush.Style;
    Brush.Style:=bsClear;
    BrushCopy(DstRect, fBitmap, SrcRect, ColorKey);
    Brush.Style:=PrevStyle;
  end;
end;
```

2) TImageLayerList 클래스

① 요구사항

- ☑ 여러 개의 레이어를 리스트로 가져야 한다.
- ☑ 편집중인 레이어의 인덱스를 가져야 한다.
- ☑ 레이어의 추가, 삭제 기능을 가져야 한다.
- ☑ 개별 레이어에 접근할 수 있어야 한다.
- ☑ 툴 편집에 활용하기 위해 별도의 레이어를 가져야 한다.
- ☑ 투명색을 나타내기 위해 바탕에 격자무늬를 그려야 한다
- ☑ 각 레이어 변경시에 전체를 업데이트할지 여부를 선택할 수 있어야 한다.
- ☑ 크기가 변경되면 모든 레이어의 크기를 자동으로 변경시켜야 한다.
- ☑ 전체 이미지에 해당하는 Canvas 를 이용할 수 있어야 한다.
- ☑ 레이어 전체를 저장하고 불러올 수 있어야 한다.
- ☑ 임의의 Canvas 에 이미지를 그리는 함수를 지원해야 한다.

② 설계

☑ 선언부

```
TImageLayerList = class(TCustomControl)
  private
    fBrushBitmap: TBitmap
    fLayerList: TList
    fToolLayer: TImageLayer
    fLayerIndex: Integer
    fAutoUpdate: Boolean;
    procedure ImageLayerUpdate(Sender: TObject);
    procedure ResetLayerListSize;
    function GetLayerCount: Integer;
    procedure SetLayerIndex(Value: Integer);
```

```
    procedure SetAutoUpdate(Value: Boolean);
    function GetLayer(Index: Integer): TImageLayer;
    procedure PaintBG;
    procedure PaintToCanvas(DstCanvas: TCanvas; SrcRect, DstRect: TRect);
    procedure WMSize(var Msg: TWMSize); message WM_SIZE;
  protected
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure DeleteAll;
    function Delete(Index: Integer): Boolean;
    function Add: Integer;
    function Insert(Index: Integer): Boolean;
    procedure Paint; override;
    function LoadFromFile(FileName: String): Boolean;
    procedure SaveToFile(FileName: String);
    procedure SaveToBitmapFile(FileName: String);
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property AutoUpdate: Boolean read fAutoUpdate write SetAutoUpdate;
    property Canvas;
    property LayerCount: Integer read GetLayerCount;
    property LayerIndex: Integer read fLayerIndex write SetLayerIndex;
    property Layer[Index: Integer]: TImageLayer read GetLayer;
    property ToolLayer: TImageLayer read fToolLayer;
  end;
```

③ 구현

☑ 초기화, 해제

```
constructor TImageLayerList.Create(AOwner: TComponent);
  procedure MakeCanvasBrushBitmap;
  const
    BrushBitmapR = 4;
    BrushBitmapW = BrushBitmapR*2;
    BrushBitmapColorF = clSilver;
    BrushBitmapColorB = clGray;
    BrushBitmapColor: array[0..1, 0..1]of TColor
          =((BrushBitmapColorF, BrushBitmapColorB),
            (BrushBitmapColorB, BrushBitmapColorF));
  var
    iX, iY: Integer;
  begin
    fBrushBitmap:=TBitmap.Create;
    with fBrushBitmap do
    begin
      Width :=BrushBitmapW;
      Height:=BrushBitmapW;
      for iX:=0 to BrushBitmapW-1 do for iY:=0 to BrushBitmapW-1 do
      begin
        Canvas.Pixels[iX, iY]:=BrushBitmapColor[(iX div BrushBitmapR) and $1,
                                                (iY div BrushBitmapR) and $1];
```

```
        end;
      end;
    end;
begin
  inherited Create(AOwner);

  DoubleBuffered:=True;

  // 레이어 리스트 객체를 만글고...
  fLayerList:=TList.Create;
  Add; // 최소한 하나 이상의 레이어는 갖고 있어야 한다...
  fLayerIndex:=0;

  // 툴 레이어도 만들어둔다.
  fToolLayer:=TImageLayer.Create(Width, Height);
  fToolLayer.OnUpdate:=ImageLayerUpdate;
  fToolLayer.Visible:=False;

  // 배경이미지를 그리기 위해서 미리 브러쉬 비트맵을 만들어둔다.
  MakeCanvasBrushBitmap;

  // 기타
  fAutoUpdate:=True;
end;

destructor TImageLayerList.Destroy;
begin
  // 이런저런 뒤처리...
  DeleteAll;
  if Assigned(fLayerList) then FreeAndNil(fLayerList);
  if Assigned(fToolLayer) then fToolLayer.Free;
  if Assigned(fBrushBitmap) then FreeAndNil(fBrushBitmap);

  inherited Destroy;
end;
```

☑ 프로퍼티

```
function TImageLayerList.GetLayerCount: Integer;
begin
  Result:=fLayerList.Count;
end;

procedure TImageLayerList.SetLayerIndex(Value: Integer);
begin
  if (Value>=0)and(Value<fLayerList.Count) then fLayerIndex:=Value;
end;

procedure TImageLayerList.SetAutoUpdate(Value: Boolean);
begin
  if fAutoUpdate<>Value then
  begin
    fAutoUpdate:=Value;
    if fAutoUpdate then Repaint;
```

```
  end;
end;

function TImageLayerList.GetLayer(Index: Integer): TImageLayer;
begin
  Result:=nil;
  if (Index>=0) and (Index<fLayerList.Count) then Result:=fLayerList.Items[Index];
end;
```

☑ 레이어 추가, 삭제, 끼워넣기

```
function TImageLayerList.Add: Integer;
begin
  Result:=-1;
  if Insert(fLayerList.Count) then
  begin
    Result:=fLayerList.Count-1;
  end;
end;

function TImageLayerList.Delete(Index: Integer): Boolean
begin
  Result:=False;
  with fLayerList do
  begin
    if (Index>=0)and(Index<Count) then
    begin
      TImageLayer(Items[Index]).Free;
      Delete(Index);
      Result:=True;
    end;
  end;
  Repaint;
end;

procedure TImageLayerList.DeleteAll;
var
  i: Integer;
begin
  for i:=fLayerList.Count-1 downto 0 do Delete(i);
end;

function TImageLayerList.Insert(Index: Integer): Boolean
var
  NewLayer: TImageLayer;
begin
  Result:=False;
  with fLayerList do
  begin
    if (Index>=0) and (Index<=Count) then
    begin
      NewLayer:=TImageLayer.Create(Width, Height);
      NewLayer.OnUpdate:=ImageLayerUpdate;
```

```
        fLayerList.Insert(Index, NewLayer);
        Result:=True;
      end;
  end;
end;
```

☑ 레이어 크기 변경

```
procedure TImageLayerList.WMSize(var Msg: TWMSize);
begin
  inherited;
  ResetLayerListSize;
end;

procedure TImageLayerList.ResetLayerListSize;
var
  i: Integer;
begin
  with fLayerList do
  begin
    for i:=0 to Count-1 do
    begin
      with TImageLayer(Items[i]).fBitmap do
      begin
        Width :=Self.Width;
        Height:=Self.Height;
      end;
    end;
  end;

  with fToolLayer.fBitmap do
  begin
    Width :=Self.Width;
    Height:=Self.Height;
  end;
end;
```

☑ 그리기

```
procedure TImageLayerList.ImageLayerUpdate(Sender: TObject);
begin
  if fAutoUpdate then Repaint;
end;

procedure TImageLayerList.Paint; // 레이어 전체를 콘트롤에 그려주는 함수
begin
  //Inherited;

  PaintBG;
  PaintToCanvas(Canvas, ClientRect, ClientRect);
end;

procedure TImageLayerList.PaintBG;
```

```
var
  PrevPenStyle: TPenStyle;
  PrevBitmap: TBitmap;
begin
  with Canvas do
  begin
    PrevPenStyle:=Pen.Style;
    PrevBitmap:=Brush.Bitmap;

    Pen.Style :=psClear;
    Brush.Bitmap:=fBrushBitmap;
    Rectangle(Rect(0, 0, Width+1, Height+1));

    Pen.Style:=PrevPenStyle;
    Brush.Bitmap:=PrevBitmap;
  end;
end;

procedure TImageLayerList.PaintToCanvas(DstCanvas: TCanvas;
                                        SrcRect, DstRect: TRect);
var
  i: Integer;
begin
  with fLayerList do
  begin
    for i:=0 to Count-1 do
    begin
      // 레이어를 그려주고
      with TImageLayer(Items[i]) do if Visible then
                                        PaintTo(DstCanvas, SrcRect, DstRect);
      // 툴 레이어를 그려주고
      if (i=fLayerIndex)and(fToolLayer.Visible) then
                              fToolLayer.PaintTo(DstCanvas, SrcRect, DstRect);
    end;
  end;
end;
```

☑ 불러오기, 저장하기

```
function TImageLayerList.LoadFromFile(FileName: String): Boolean;
var
  FS: TFileStream;
  iLayer, LCount: Integer;
  TempBool: Boolean;
  TempBMP: TBitmap;
begin
  Result:=False;
  if not FileExists(FileName) then Exit;

  DeleteAll;

  FS:=TFileStream.Create(FileName, fmOpenRead);
  FS.Read(LCount, SizeOf(LCount));
```

```
  for iLayer:=0 to LCount-1 do // 각 레이어별 불러오기
  begin
    Add; // 레이어를 만들고

    with Layer[iLayer] do
    begin
      // Visible 불러와서 적용
      FS.Read(TempBool, SizeOf(TempBool));
      Visible:=TempBool;

      // 비트맵 불러오기
      TempBMP:=TBitmap.Create;
      TempBMP.LoadFromStream(FS);
      Layer[iLayer].Canvas.CopyRect(ClientRect, TempBMP.Canvas, ClientRect);
      FreeAndNil(TempBMP);
    end;
  end;

  // 마무리
  FreeAndNil(FS);
  Result:=True;
end;

procedure TImageLayerList.SaveToFile(FileName: String);
var
  FS: TFileStream;
  BackupFileName: String;
  iLayer, TempInt: Integer;
  TempBool: Boolean;
begin
  // 기존 파일 백업
  if FileExists(FileName) then
  begin
    BackupFileName:=FileName+'~';
    if FileExists(BackupFileName) then DeleteFile(BackupFileName);
    RenameFile(FileName, BackupFileName);
  end;

  // 파일 스트림을 만들고
  FS:=TFileStream.Create(FileName, fmCreate);

  // 레이어 수 저장
  TempInt:=LayerCount;
  FS.Write(TempInt, SizeOf(TempInt));

  // 각 레이어별 저장
  for iLayer:=0 to LayerCount-1 do
  begin
    with Layer[iLayer] do
    begin
      TempBool:=Visible;
      FS.Write(TempBool, SizeOf(TempBool));

      Layer[iLayer].Bitmap.SaveToStream(FS);
```

```
    end;
  end;

  // 마무리
  FreeAndNil(FS);
end;

procedure TImageLayerList.SaveToBitmapFile(FileName: String);
var
  TempBitmap: TBitmap;
begin
  // 저장할 이미지를 만들고
  TempBitmap:=TBitmap.Create;
  TempBitmap.PixelFormat:=pf32bit;
  TempBitmap.Width :=Width;
  TempBitmap.Height:=Height;

  // 화면 전체를 그리고
  PaintToCanvas(TempBitmap.Canvas, ClientRect, ClientRect);

  TempBitmap.SaveToFile(FileName); // 저장

  FreeAndNil(TempBitmap); // 뒤처리
end;
```

3) 레이어 기능 구현

① 레이어 리스트 객체 생성, 해제

☑ 생성

```
procedure TMain.FormCreate(Sender: TObject);
begin
          :
  ImageLayerList:=TImageLayerList.Create(Self);
  with ImageLayerList do
  begin
    Parent:=ScreenPanel;
    Align :=alClient;
    OnMouseDown:=ImageLayerListMouseDown;
    OnMouseMove:=ImageLayerListMouseMove;
    OnMouseUp   :=ImageLayerListMouseUp   ;
  end;
          :
end;
```

☑ 해제

```
procedure TMain.FormDestroy(Sender: TObject);
begin
          :
  if Assigned(ImageLayerList) then FreeAndNil(ImageLayerList);
          :
end;
```

② 레이어 제어

☑ 리스트박스 목록 업데이트

```
procedure TMain.LayerListUpdate;
var
  i, PrevLayerIndex: Integer;
begin
  with ImageLayerList do
  begin
    PrevLayerIndex:=LayerIndex;

    for i:=0 to Max(LayerListBox.Items.Count, LayerCount)-1 do
    begin
      if i<ImageLayerList.LayerCount then
      begin
        if i>=LayerListBox.Items.Count then
                LayerListBox.Items.Add('Layer'+IntToStr(LayerCount-1-i))
        else
                LayerListBox.Items[i]:='Layer'+IntToStr(LayerCount-1-i);
        LayerListBox.Checked[i]:=ImageLayerList.Layer[LayerCount-1-i].Visible;
      end
      else
      begin
        LayerListBox.Items.Delete(LayerListBox.Items.Count-1);
      end;
    end;

    LayerListBox.ItemIndex:=LayerCount-1-PrevLayerIndex;

    DelLayerBtn.Enabled:=LayerCount>=2;
  end;
end;
```

☑ 레이어 인덱스 변경

```
procedure TMain.LayerListBoxClick(Sender: TObject);
begin
  with ImageLayerList do
  begin
    LayerIndex:=LayerCount-1-LayerListBox.ItemIndex;
  end;
end;
```

☑ 레이어 Visible 속성 변경

```
procedure TMain.LayerListBoxClickCheck(Sender: TObject);
var
  i: Integer;
begin
  with ImageLayerList, LayerListBox do
  begin
    for i:=0 to Items.Count-1 do
    begin
```

```
        Layer[i].Visible:=Checked[LayerCount-1-i];
      end;
    end;
end;
```

☑ 레이어 추가, 삭제, 끼워넣기

```
procedure TMain.AddLayerBtnClick(Sender: TObject);
begin
  if ImageLayerList.Add>=0 then LayerListUpdate;
end;

procedure TMain.InsLayerBtnClick(Sender: TObject);
begin
  if ImageLayerList.Insert(LayerListBox.ItemIndex) then LayerListUpdate;
end;

procedure TMain.DelLayerBtnClick(Sender: TObject);
begin
  if ImageLayerList.Delete(LayerListBox.ItemIndex) then LayerListUpdate;
end;
```

3. 편집 기능

1) 자료형과 변수

① 편집 모드

☑ 자료형, 변수 선언

```
type
  TEditMode = (emRectArea, emFreeArea, emPen, emBrush,
               emFill, emRect, emCircle, emLine, emErase, emSpoit);
Var
  EditMode: TEditMode;
```

☑ 편집 도구 버튼 Tag 프로퍼티 설정

편집 도구 버튼의 Tag 를 해당하는 EditMode 의 Ord 값으로 정해준다.

☑ 편집 도구 버튼 OnClick 이벤트 핸들러

```
procedure TMain.PenBtnClick(Sender: TObject);
begin
  EditMode:=TEditMode((Sender as TControl).Tag);
end;
```

※ 이벤트 핸들러를 공유할 것

② 편집 색상

☑ 변수 선언

```
Var
  ToolColor: TColor = clBlack;
```

☑ 변수 설정 함수

```
procedure TMain.SetToolColor(NewColor: TColor);
begin
  ColorPanel.Color:=NewColor;

  // 컬러키는 선택되지 않게 한다.
  if NewColor=ColorKey then NewColor:=ColorKeyConvert;

  ToolColor:=NewColor;
end;
```

☑ 색상 선택

```
procedure TMain.ColorPanelClick(Sender: TObject);
begin
  with ColorDialog do
  begin
    Color:=ColorPanel.Color;
    if Execute then
    begin
      SetToolColor(Color);
```

```
    end;
  end;
end;
```

2) 편집 기능 구현

① 기본적인 그리기 기능 구현 (펜, 붓, 지우개 모드)

☑ 상수

```
const
  BrushWidth =  5;
  EraseWidth = 20;
```

☑ ImageLayerListMouseDown

```
procedure TMain.ImageLayerListMouseDown
        (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
  begin
    Editing:=True; // 편집중 표시
          :
    PrevMousePos:=Point(X, Y); // 처음 눌려진 위치를 기억하고

    // 툴 레이어를 활성화하고 기본적인 설정을 해준다음
    with ImageLayerList, ToolLayer do
    begin
      Visible:=True;
      with Canvas do
      begin
        // 각 모드에 따라 각각의 처리를 하고
        case EditMode of
                  :
                  :
          emBrush:
          begin
            Pen.Width:=BrushWidth; // 붓은 모름지기 두꺼워야...
          end;
          emErase:
          begin
            // 지우개는 더더욱 두꺼워야...
            Pen.Width:=EraseWidth;
            // 편집할 레이어를 몽땅 툴레이어에 복사하고
            Layer[LayerIndex].PaintTo(Canvas, ClientRect, ClientRect);
            // 원본 레이어는 청소해준다...
            Layer[LayerIndex].Clear;
          end;
        end;

        case EditMode of
                  :
                  :
          emErase:
```

```
            begin
              Pen.Color   :=ColorKey;
              Pen.Style   :=psSolid;
              Brush.Color:=ColorKey;
              Brush.Style:=bsSolid;
            end;
            else
            begin
              Pen.Color   :=ToolColor;
              Pen.Style   :=psSolid;
              Brush.Color:=ToolColor;
              Brush.Style:=bsSolid;
            end;
          end;
        end;
      end;

      // 중복코드도 귀찮고 하니 이런 꼼수를...
      ImageLayerListMouseMove(Sender, Shift, X, Y);
    end;
end;
```

☑ ImageLayerListMouseMove

```
procedure TMain.ImageLayerListMouseMove
                                  (Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if Editing then
  begin
    // 좌표가 이미지 영역을 벗어나지 않도록 하고
    if X<0 then X:=0;
    if Y<0 then Y:=0;
    if X>ImageLayerList.Width   then X:=ImageLayerList.Width;
    if Y>ImageLayerList.Height then Y:=ImageLayerList.Height;

    with ImageLayerList.ToolLayer, Canvas do
    begin
      case EditMode of
                    :
                    :
        emPen, emBrush, emErase:
        begin
          // 선을 그리고
          MoveTo(PrevMousePos.X, PrevMousePos.Y);
          LineTo(X, Y);
          PrevMousePos:=Point(X, Y);
        end;
                    :
                    :
      end;
    end;
  end;
end;
```

☑ ImageLayerListMouseUp

```
procedure TMain.ImageLayerListMouseUp
        (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  SrcArea: TRect;
  i, TempInt: Integer;
begin
  if Editing then
  begin
    if X<0 then X:=0; // 좌표가 이미지 영역을 벗어나지 않도록 하고
    if Y<0 then Y:=0;
    if X>ImageLayerList.Width  then X:=ImageLayerList.Width;
    if Y>ImageLayerList.Height then Y:=ImageLayerList.Height;

    with ImageLayerList do
    begin
      case EditMode of
        emRectArea, emFreeArea:
        begin
                :
        end
        else
        begin
          // 툴 레이어에 그려진 것을 현재 레이어로 옮긴다
          ToolLayer.PaintTo(Layer[LayerIndex].Canvas, ClientRect, ClientRect);
        end;
      end;

      // 뒷정리
      with ToolLayer do
      begin
        case EditMode of
          emBrush, emErase:
          begin
            Canvas.Pen.Width:=1;
          end;
        end;
        Visible:=False;
        Clear;
      end;
    end;
    // 편집 끝
    Editing:=False;
  end;
end;
```

② 채우기 모드

☑ ImageLayerListMouseDown

```
procedure TMain.ImageLayerListMouseDown
        (Sender: TObject; Button: TMouseButton; Shift: TSh`iftState; X, Y:
Integer);
```

```
begin
  if Button = mbLeft then
  begin
          :
          :
    with ImageLayerList, ToolLayer do
    begin
          :
      Visible:=True;
      with Canvas do
      begin
        // 각 모드에 따라 각각의 처리를 하고
        case EditMode of
                  :
                  :
          emFill:
          begin
            // ToolLayer 에 편집중인 레이어를 한번은 복사해줘야 한다.
            Layer[LayerIndex].PaintTo(Canvas, ClientRect, ClientRect);
          end;
                  :
                  :
        end;
          :
          :
      end;
    end;
          :
          :
  end;
end;
```

☑ ImageLayerListMouseMove

```
procedure TMain.ImageLayerListMouseMove
                                  (Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if Editing then
  begin
          :
    with ImageLayerList.ToolLayer, Canvas do
    begin
      case EditMode of
          :
          :
        emFill:
        begin
          FloodFill(X, Y, Pixels[X,Y], fsSurface);
        end;
          :
          :
      end;
```

```
    end;
  end;
end;
```

③ 네모, 동그라미, 직선 모드

☑ ImageLayerListMouseMove

```
procedure TMain.ImageLayerListMouseMove
                                    (Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if Editing then
  begin
          :
          :
    with ImageLayerList.ToolLayer, Canvas do
    begin
      case EditMode of
                  :
                  :
        emRect, emCircle, emLine:
        begin
          // 이미지 리스트 객체의 자동 업데이트 기능을 잠시 끄고
          ImageLayerList.AutoUpdate:=False;

          Clear; // 청소를 하고

          // 다시 그림을 그린다.
          case EditMode of
            emRect:
            begin
              Rectangle(PrevMousePos.X, PrevMousePos.Y, X, Y);
            end;
            emCircle:
            begin
              Ellipse(PrevMousePos.X, PrevMousePos.Y, X, Y);
            end;
            else//emLine
            begin
              MoveTo(PrevMousePos.X, PrevMousePos.Y);
              LineTo(X, Y);
            end;
          end;

          // 자동 업데이트 기능을 원래대로 돌려놓는다.
          ImageLayerList.AutoUpdate:=True;
        end;
                  :
                  :
      end;
    end;
  end;
end;
```

④ 스포이트 모드 구현

☑ ImageLayerListMouseMove

```
procedure TMain.ImageLayerListMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
begin
  if Editing then
  begin
          :
          :
    with ImageLayerList.ToolLayer, Canvas do
    begin
      case EditMode of
                  :
                  :
        emSpoit:
        begin
          SetToolColor(ImageLayerList.Canvas.Pixels[X,Y]);
        end;
                  :
                  :
      end;
    end;
  end;
end;
```

4. 영역 선택

1) TImagePart 클래스

① 요구사항

☑ Bitmap 을 가지고 있어야 한다.

☑ 비정형 영역을 처리할 수 있어야 한다.

☑ 영역의 외곽선을 점선으로 그릴 수 있어야 한다.

☑ 마우스로 드래그할 수 있어야 한다.

☑ 드래그가 끝나면 이벤트 핸들러를 호출해야 한다.

☑ Bitmap 을 다른 Canvas 에 복사할 수 있어야 한다.

② 설계

☑ 선언부

```
TImagePart = class(TGraphicControl)
  private
    fBMP: TBitmap;
    fSrcPolygon: array of TPoint;
    fMoving: Boolean;
    fPrevPos: TPoint;
    fOnDrop: TNotifyEvent;
    procedure _SetPartImage(SrcCanvas: TCanvas; SrcRect: TRect);
    procedure WMLButtonDown(var SrcMsg: TWMLButtonDown);
                                              message WM_LBUTTONDOWN;
    procedure WMMouseMove(var SrcMsg: TWMMouseMove);
                                              message WM_MOUSEMOVE;
    procedure WMLButtonUp(var SrcMsg: TWMLButtonUp);
                                              message WM_LBUTTONUP;
  protected
    procedure Paint; override
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure SetPartImage(SrcCanvas: TCanvas;
                           SrcRect: TRect; Cut: Boolean); overload;
    procedure SetPartImage(SrcCanvas: TCanvas;
                           SrcPolygon: array of TPoint; Cut: Boolean); overload;
    procedure CopyTo(DstCanvas: TCanvas; DstRect: TRect);
    property OnDrop: TNotifyEvent read fOnDrop write fOnDrop;
  end;
```

③ 구현

☑ 초기화, 해제

```
constructor TImagePart.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);

  // 영역 표시 점선을 그리기 위해 Canvas 속성을 미리 설정해둔다.
```

```
  with Canvas do
  begin
    Pen.Color   :=clWhite;
    Pen.Style   :=psDot;
    //Pen.Mode    :=pmXor;
    Brush.Style:=bsClear;
  end;

  // 손 모양 커서를 쓰도록 한다.
  Cursor:=crHandPoint;

  // 복사할 이미지를 담을 Bitmap 객체를 미리 만들어둔다.
  fBMP:=TBitmap.Create;
  fBMP.PixelFormat:=pf24bit;
  fBMP.Width :=Width;
  fBMP.Height:=Height;
  fBMP.TransParent:=True;
  fBMP.TransParentColor:=ColorKey;
end;

destructor TImagePart.Destroy;
begin
  // 이런저런 뒤처리...
  FreeAndNil(fBMP);
  SetLength(fSrcPolygon, 0);
  inherited Destroy;
end;
```

☑ 이미지, 영역 설정

```
// 복사할 이미지 설정함수.
procedure TImagePart._SetPartImage(SrcCanvas: TCanvas; SrcRect: TRect);
var
  DstRect: TRect;
begin
  with DstRect do
  begin
    Left   :=0;
    Top    :=0;
    Right :=SrcRect.Right-SrcRect.Left;
    Bottom:=SrcRect.Bottom-SrcRect.Top;

    fBMP.Width :=Right;
    fBMP.Height:=Bottom;

    fBMP.Canvas.CopyRect(DstRect, SrcCanvas, SrcRect);

    Width :=Right;
    Height:=Bottom;
  end;
end;

// 사각 영역으로 복사할 이미지를 설정.
```

```
procedure TImagePart.SetPartImage
                              (SrcCanvas: TCanvas; SrcRect: TRect; Cut: Boolean);
begin
  // 사각 영역에 따라 이미지도 업뎃해주고 크기도 조절해주고...
  _SetPartImage(SrcCanvas, SrcRect);

  // 폴리곤을 쓰지 않는다고 표시해주고...
  SetLength(fSrcPolygon, 0);

  // 잘라내기
  if Cut then
  begin
    with SrcCanvas do
    begin
      Pen   .Color:=ColorKey;
      Pen   .Style:=psSolid;
      Brush.Color:=ColorKey;
      Brush.Style:=bsSolid;
      Rectangle(SrcRect);
    end;
  end;
end;

// 비정형 영역으로 복사할 이미지를 설정.
procedure TImagePart.SetPartImage
                   (SrcCanvas: TCanvas; SrcPolygon: array of TPoint; Cut: Boolean);
var
  fRgn: HRGN;
  SrcRect: TRect;
  i, ArrCount: Integer;
  iX, iY: Integer;
begin
  // 전체 사각 영역을 계산해준다. (비정형 영역에 외접하는 직사각형)
  SrcRect:=Rect(MaxInt, MaxInt,-MaxInt,-MaxInt);
  with SrcRect do
  begin
    for i:=0 to Length(SrcPolygon)-1 do with SrcPolygon[i] do
    begin
      if X<Left    then Left   :=X;
      if X>Right   then Right :=X;
      if Y<Top     then Top    :=Y;
      if Y>Bottom then Bottom:=Y;
    end;
    Inc(Right);
    Inc(Bottom);
  end;

  // 사각 영역에 따라 이미지도 업뎃해주고 크기도 조절해주고...
  _SetPartImage(SrcCanvas, SrcRect);

  // 영역을 자기 자신의 좌표 기준으로 변경
  ArrCount:=Length(SrcPolygon);
  SetLength(fSrcPolygon, ArrCount);
  for i:=0 to ArrCount-1 do with SrcPolygon[i] do
```

```
  begin
    Dec(x, SrcRect.Left);
    Dec(y, SrcRect.Top);
  end;

  // 영역을 만들고
  fRgn:=CreatePolygonRgn(SrcPolygon, ArrCount, ALTERNATE);

  // 컬러키를 설정하고
  fBMP.TransparentColor:=ColorKey;
  fBMP.Transparent:=True;

  // 이미지의 영역 외부를 컬러키로 칠해주자.
  for iX:=0 to fBMP.Width-1 do for iY:=0 to fBMP.Height-1 do
  begin
    if not PtInRegion(fRgn, iX, iY) then
    begin
      fBMP.Canvas.Pixels[iX, iY]:=ColorKey;
    end;
  end;

  // 잘라내기 (영역 내부를 투명색으로 바꿔주면 된다)
  if Cut then
  begin
    for iX:=0 to fBMP.Width-1 do for iY:=0 to fBMP.Height-1 do
    begin
      if PtInRegion(fRgn, iX, iY) then
      begin
        SrcCanvas.Pixels[SrcRect.Left+iX, SrcRect.Top+iY]:=ColorKey;
      end;
    end;
  end;

  DeleteObject(fRgn);

  // 영역의 외곽선을 보관하고...
  SetLength(fSrcPolygon, ArrCount);
  for i:=0 to ArrCount-1 do fSrcPolygon[i]:=SrcPolygon[i];
end;
```

☑ 마우스로 움직이기

```
procedure TImagePart.WMLButtonDown(var SrcMsg: TWMLButtonDown);
begin
  fMoving:=True; // 드래그 시작이라고 표시해주고
  with SrcMsg do
  begin
    fPrevPos:=Point(XPos, YPos); // 마우스 위치를 보관...
  end;

  Inherited;
end;
```

```
procedure TImagePart.WMMouseMove(var SrcMsg: TWMMouseMove);
var
  dX, dY: Integer;
begin
  if fMoving then
  begin
    with SrcMsg do
    begin
      dX:=XPos-fPrevPos.X; // 이동한 좌표 거리를 계산
      dY:=YPos-fPrevPos.Y;
    end;
    Left:=Left+dX; // 이동시켜준다.
    Top :=Top +dY;
  end;
  inherited;
end;

procedure TImagePart.WMLButtonUp(var SrcMsg: TWMLButtonUp);
begin
  Inherited;
  if fMoving then
  begin
    fMoving:=False; // 이동이 끝났다고 표시하고
    if Assigned(fOnDrop) then fOnDrop(Self); // 드롭 이벤트 핸들러를 호출한다.
  end;
end;
```

☑ 그리기

```
procedure TImagePart.Paint; // 화면에 그리기
begin
  with Canvas do
  begin
    Draw(0, 0, fBMP);

    // 선택 영역 표시를 점선으로 그려준다.
    if Length(fSrcPolygon)>0 then
    begin
      Polyline(fSrcPolygon);
    end
    else
    begin
      Rectangle(ClientRect);
    end;
  end;
end;

// 다른 Canvas 에 그리기
procedure TImagePart.CopyTo(DstCanvas: TCanvas; DstRect: TRect);
var
  PrevBrushStyle: TBrushStyle;
begin
  PrevBrushStyle:=DstCanvas.Brush.Style;
```

```
  DstCanvas.Brush.Style:=bsClear;
  DstCanvas.BrushCopy(DstRect, fBMP, ClientRect, ColorKey);

  DstCanvas.Brush.Style:=PrevBrushStyle;
end;
```

2) 영역 선택 기능 구현

① 영역 선택

☑ ImageLayerListMouseDown

```
procedure TMain.ImageLayerListMouseDown(Sender: TObject; Button:
TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
  begin
          :
          :
    with ImageLayerList, ToolLayer do
    begin
      Visible:=True;
      with Canvas do
      begin
        // 각 모드에 따라 각각의 처리를 하고
        case EditMode of
          emFreeArea:
          begin
            SetLength(SrcAreaPoint, 1); // 외곽선 점 하나 추가
            SrcAreaPoint[0]:=Point(X, Y);
          end;
                  :
                  :
        end;

        case EditMode of
          emRectArea, emFreeArea:
          begin
            Pen.Color   :=clWhite;
            Pen.Style   :=psDot;
            Brush.Color:=clBlack;
            Brush.Style:=bsClear;
          end;
                  :
                  :
        end;
      end;
    end;
          :
          :
  end;
end;
```

☑ ImageLayerListMouseMove

```
procedure TMain.ImageLayerListMouseMove
                                    (Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if Editing then
  begin
          :
          :
    with ImageLayerList.ToolLayer, Canvas do
    begin
      case EditMode of
        emRectArea, emFreeArea:
        begin
          ImageLayerList.AutoUpdate:=False;

          Clear; // 청소를 하고

          if EditMode = emRectArea then
          begin
            // 다시 사각형을 그린다.
            Rectangle(PrevMousePos.X, PrevMousePos.Y, X, Y);
          end
          else //emFreeArea
          begin
            // 점 배열 하나 추가
            SetLength(SrcAreaPoint, Length(SrcAreaPoint)+1);
            SrcAreaPoint[Length(SrcAreaPoint)-1]:=Point(X, Y);
            Polyline(SrcAreaPoint);
          end;

          ImageLayerList.AutoUpdate:=True;
        end;
                    :
                    :
      end;
    end;
  end;
end;
```

☑ ImageLayerListMouseUp

```
procedure TMain.ImageLayerListMouseUp
          (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  SrcArea: TRect;
  i, TempInt: Integer;
begin
  if Editing then
  begin
          :
    with ImageLayerList do
    begin
      case EditMode of
        emRectArea, emFreeArea:
```

```
begin
  if EditMode = emRectArea then
  begin
    // 영역 계산
    SrcArea:=Rect(PrevMousePos.X, PrevMousePos.Y, X, Y);
    with SrcArea do
    begin
      if Left>Right then
      begin
        TempInt:=Left;
        Left:=Right;
        Right:=TempInt;
      end;
      if Top>Bottom then
      begin
        TempInt:=Top;
        Top:=Bottom;
        Bottom:=TempInt;
      end;
    end;
  end
  else //emFreeArea
  begin
    // 닫힌 도형으로 만들기 위해 점 하나를 추가한다.
    SetLength(SrcAreaPoint, Length(SrcAreaPoint)+1);
    SrcAreaPoint[Length(SrcAreaPoint)-1]:=SrcAreaPoint[0];

    // 외접 사각 영역을 계산해준다.
    SrcArea:=Rect(MaxInt, MaxInt,-MaxInt,-MaxInt);
    with SrcArea do
    begin
      for i:=0 to Length(SrcAreaPoint)-1 do with SrcAreaPoint[i] do
      begin
        if X<Left   then Left  :=X;
        if X>Right  then Right :=X;
        if Y<Top    then Top   :=Y;
        if Y>Bottom then Bottom:=Y;
      end;
      Inc(Right);
      Inc(Bottom);
    end;
  end;

  // 영역의 크기가 0 이 아니면
  if (SrcArea.Left<SrcArea.Right)and(SrcArea.Top<SrcArea.Bottom) then
  begin
    // CopyPart 를 만든다.
    if EditMode = emRectArea then
    begin
      MakeCopyPart(Layer[LayerIndex].Canvas,
                   SrcArea,
                   ImageLayerList.Left+SrcArea.Left,
                   ImageLayerList.Top+SrcArea.Top,
                   AutoCutCheckBox.Checked);
```

```
            end
            else //emFreeArea
            begin
              MakeCopyPart(Layer[LayerIndex].Canvas,
                           SrcAreaPoint,
                           ImageLayerList.Left+SrcArea.Left,
                           ImageLayerList.Top+SrcArea.Top,
                           AutoCutCheckBox.Checked);
            end;
          end;
        end
        else
        begin
                :
        end;
      end;
        :
    end;
  end;
end;
```

② 영역 객체

☑ 생성

```
procedure TMain.MakeCopyPart
        (SrcCanvas: TCanvas; SrcRect: TRect; PosX, PosY: Integer; Cut: Boolean);
begin
  if not Assigned(CopyPart) then CopyPart:=TImagePart.Create(Self);
  with CopyPart do
  begin
    Parent:=ImageLayerList;
    Visible:=True;
    CopyPart.OnDrop:=CopyPartDrop;
  end;
  CopyPart.Left   :=PosX;
  CopyPart.Top    :=PosY;
  with ImageLayerList do
  begin
    AutoUpdate:=False;
    CopyPart.SetPartImage(SrcCanvas, SrcRect, Cut);
    AutoUpdate:=True;
  end;
end;

procedure TMain.MakeCopyPart(SrcCanvas: TCanvas;
                    SrcPointArr: array of TPoint; PosX, PosY: Integer; Cut: Boolean);
begin
  if not Assigned(CopyPart) then CopyPart:=TImagePart.Create(Self);
  with CopyPart do
  begin
    Parent:=ImageLayerList;
    Visible:=True;
    CopyPart.OnDrop:=CopyPartDrop;
```

```
  end;
  CopyPart.Left   :=PosX;
  CopyPart.Top    :=PosY;
  with ImageLayerList do
  begin
    AutoUpdate:=False;
    CopyPart.SetPartImage(SrcCanvas, SrcPointArr, Cut);
    AutoUpdate:=True;
  end;
end;
```

☑ 이미지 복사 → 해체

```
procedure TMain.CopyPartDrop(Sender: TObject);
var
  DstRect: TRect;
begin
  if not Visible then Exit;

  // 복사할 사각 영역 계산
  DstRect.Left   :=CopyPart.Left-ImageLayerList.Left;
  DstRect.Top    :=CopyPart.Top -ImageLayerList.Top;
  DstRect.Right :=DstRect.Left +CopyPart.Width;
  DstRect.Bottom:=DstRect.Top  +CopyPart.Height;

  // 영역 객체의 이미지를 복사
  with ImageLayerList do
  begin
    CopyPart.CopyTo(Layer[LayerIndex].Canvas, DstRect);
  end;

  // 영역 객체 해제
  FreeAndNil(CopyPart);
end;
```

5. 기타

1) 파일 불러오기, 저장하기

① 레이어 리스트

☑ 불러오기

```
procedure TMain.M_LoadClick(Sender: TObject);
begin
  with OpenDlg do
  begin
    Title:='레이어 파일 불러오기';
    DefaultExt:='ilf';
    Filter:='Image layer file(*.ilf)|*.ilf';
    if Execute then
    begin
      if not ImageLayerList.LoadFromFile(FileName) then
                          ShowMessage(FileName+' 파일을 불러올 수 없습니다.');
      LayerListUpdate;
    end;
  end;
end;
```

☑ 저장하기

```
procedure TMain.M_SaveClick(Sender: TObject);
begin
  with SaveDlg do
  begin
    Title:='레이어 파일 저장하기';
    DefaultExt:='ilf';
    Filter:='Image layer file(*.ilf)|*.ilf';
    if Execute then
    begin
      ImageLayerList.SaveToFile(FileName);
    end;
  end;
end;
```

② 비트맵 이미지

☑ 불러오기 → 영역 객체 생성

```
procedure TMain.M_LoadImageClick(Sender: TObject);
  function LoadBitmapFromFile(Bitmap: TBitmap; FileName: String): Boolean;
  var
    OrgBitmap: TBitmap;
    SrcRect: TRect;
  begin
    Result:=False;
    if not Assigned(Bitmap) then Exit;
    if not FileExists(FileName) then Exit;

    try
```

```
        // 일단 원본 이미지를 로드하고
        OrgBitmap:=TBitmap.Create;
        OrgBitmap.LoadFromFile(FileName);
        OrgBitmap.PixelFormat:=pf32bit;

        // 결과 이미지의 크기 등을 설정하고
        Bitmap.PixelFormat:=pf32bit;
        Bitmap.Width :=OrgBitmap.Width;
        Bitmap.Height:=OrgBitmap.Height;

        SrcRect:=Rect(0, 0, Bitmap.Width, Bitmap.Height);

        with Bitmap.Canvas do
        begin
          // 결과이미지를 ColorKeyConvert 로 채우고
          Pen.Color:=ColorKeyConvert;
          Pen.Style:=psSolid;
          Brush.Color:=ColorKeyConvert;
          Brush.Style:=bsSolid;
          FillRect(SrcRect);

          // 원본이미지에서 ColorKey 를 뺀 나머지를 결과이미지로 복사
          Brush.Color:=ColorKey;
          Brush.Style:=bsClear;
          BrushCopy(SrcRect, OrgBitmap, SrcRect, ColorKey);
        end;

        // 마무리
        FreeAndNil(OrgBitmap);

        Result:=True;
      finally

      end;
    end;
var
  TempBitmap: TBitmap;
begin
  with OpenDlg do
  begin
    Title:='그림 불러오기';
    DefaultExt:='bmp';
    Filter:='Bitmap image (*.bmp)|*.bmp';
    if Execute then
    begin
      TempBitmap:=TBitmap.Create;
      if not LoadBitmapFromFile(TempBitmap, FileName) then
        ShowMessage(ExtractFileName(FileName)+' 파일을 불러올 수 없습니다.');
      MakeCopyPart(TempBitmap.Canvas,
              Rect(0, 0, TempBitmap.Width, TempBitmap.Height), 0, 0, False);
      FreeAndNil(TempBitmap);
    end;
  end;
end;
```

☑ 저장하기

```
procedure TMain.M_SaveImageClick(Sender: TObject);
begin
  with SaveDlg do
  begin
    Title:='그림 저장하기';
    DefaultExt:='bmp';
    Filter:='Bitmap image (*.bmp)|*.bmp';
    if Execute then
    begin
      ImageLayerList.SaveToBitmapFile(FileName);
    end;
  end;
end;
```

수고하셨습니다...(^o^)/