How to create a transparent TPanel

Answer 1

Tip originally posted for Delphi 2

Particularly note the *SetParent* bit. It works even with movement. It should even work in *Delphi 1*, as it doesn't use the Win32 non-rectangular-window method for creating transparency. The code is simple so can be easily retro-fitted to any control that you wished were transparent. I put this together in ten minutes, so it needs proper testing to make sure it doesn't cause any problems, but here it is.

Create one on a form, and drag it about over some edits, combo boxes etc. (and *TImages* and you'll get major flicker).

```
type

private
  procedure SetParent(AParent: TWinControl); override;
  procedure WMEraseBkGnd(Var Message: TWMEraseBkGnd); message WM_EraseBkGnd;
protected
  procedure CreateParams(Var Params: TCreateParams); override;
  procedure Paint; override;
public
  constructor Create(AOwner: TComponent); override;
  procedure Invalidate; override;
end;

constructor TTransparentPanel.Create(AOwner: TComponent);
begin
  Inherited Create(AOwner);
  ControlStyle := ControlStyle - [csOpaque];
end;

procedure TTransparentPanel.CreateParams(Var Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  Params.ExStyle := Params.ExStyle or WS_EX_TRANSPARENT;
end;

procedure TTransparentPanel.Paint;
begin
  Canvas.Brush.Style := bsClear;
  Canvas.Rectangle(0, 0, Width, Height);
  Canvas.TextOut(Width div 2, Height div 2, 'Transparent');
end;

procedure TTransparentPanel.WMEraseBkGnd(Var Message: TWMEraseBkGnd);
begin
  {Do Nothing}
  Message.Result := 1;
end;

procedure TTransparentPanel.SetParent(AParent: TWinControl);
begin
  Inherited SetParent(AParent);
  {The trick needed to make it all work! I don't know if changing the parent's
  style is a good idea, but it only removes the WS_CLIPCHILDREN style which
  shouldn't cause any problems.}
  if Parent <> Nil then
    SetWindowLong(
      Parent.Handle,
      GWL_STYLE,
      GetWindowLong(Parent.Handle, GWL_STYLE) And Not WS_ClipChildren
    );
end;

procedure TTransparentPanel.Invalidate;
var
```

```
  Rect:TRect;
begin
  Rect := BoundsRect;
  if (Parent <> Nil) and Parent.HandleAllocated then
    InvalidateRect(Parent.Handle, @Rect, True)
  else
    Inherited Invalidate;
end;
```

Tip author unknown

Answer 2

```
unit TransparentPanel;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TTransparentPanel = class(TPanel)
  private
    FBackground: TBitmap;
    procedure WMEraseBkGnd( Var msg: TWMEraseBkGnd ); message WM_ERASEBKGND;
  protected
    procedure CaptureBackground;
    procedure Paint; override;
  public
    procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer); override;
    property Canvas;
    constructor Create( aOwner: TComponent ); override;
    destructor Destroy; override;
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('PBGoodies', [TTransparentPanel]);
end;

procedure TTransparentPanel.CaptureBackground;
var
  canvas: TCanvas;
  dc: HDC;
  sourcerect: TRect;
begin
  FBackground := TBitmap.Create;
  with Fbackground do
  begin
    width := clientwidth;
    height := clientheight;
  end;
  sourcerect.TopLeft := ClientToScreen(clientrect.TopLeft);
  sourcerect.BottomRight := ClientToScreen( clientrect.BottomRight );
  dc:= CreateDC( 'DISPLAY', nil, nil, nil );
  try
    canvas:= TCanvas.Create;
    try
      canvas.handle:= dc;
      Fbackground.Canvas.CopyRect( clientrect, canvas, sourcerect );
    finally
      canvas.handle := 0;
      canvas.free;
    end;
  finally
```

```
    DeleteDC( dc );
  end;
end;

constructor TTransparentPanel.Create(aOwner: TComponent);
begin
  inherited;
  ControlStyle := controlStyle - [csSetCaption];
end;

destructor TTransparentPanel.Destroy;
begin
  FBackground.free;
  inherited;
end;

procedure TTransparentPanel.Paint;
begin
  if csDesigning In ComponentState then
    inherited
    {would need to draw frame and optional caption here do not call
    inherited, the control fills its client area if you do}
end;

procedure TTransparentPanel.SetBounds(ALeft, ATop, AWidth, AHeight: Integer);
begin
  if Visible and HandleAllocated and not (csDesigning In ComponentState) then
  begin
    Fbackground.Free;
    Fbackground := Nil;
    Hide;
    inherited;
    Parent.Update;
    Show;
  end
  else
    inherited;
end;

procedure TTransparentPanel.WMEraseBkGnd(var msg: TWMEraseBkGnd);
var
  canvas: TCanvas;
begin
  if csDesigning In ComponentState then
    inherited
  else
  begin
    if not Assigned( FBackground ) then
      Capturebackground;
    canvas := TCanvas.create;
    try
      canvas.handle := msg.DC;
      canvas.draw( 0, 0, FBackground );
    finally
      canvas.handle := 0;
      canvas.free;
    end;
    msg.result := 1;
  end;
end;

end.
```

Tip by Peter Below

Answer 3

This panel will be transparent only at runtime.

```
{ ... }

type
```

```
  TMyPopUpTransPanel = class(TPanel)
  protected
    procedure CMHitTest(var Message: TCMHitTest); message CM_HITTEST;
    procedure WndProc(var Message: TMessage); override;
    procedure CreateParams(var Params: TCreateParams); override;
    procedure Paint; override;
  end;

{ ... }

procedure TMyPopUpTransPanel.CMHitTest(var Message: TCMHitTest);
begin
  Message.Result:=Windows.HTNOWHERE;
end;

procedure TMyPopUpTransPanel.WndProc(var Message: TMessage);
var
  XControl: TControl;
  XPos: TPoint;
begin
  if not (csDesigning in ComponentState) and ((Message.Msg >= WM_MOUSEFIRST)
    and (Message.Msg <= WM_MOUSELAST)) then
  begin
    XPos := ClientToScreen(
      POINT(TWMMouse(Message).XPos, TWMMouse(Message).YPos)
    );
    XControl := Parent.ControlAtPos(
      POINT(TWMMouse(Message).XPos + Left,
      TWMMouse(Message).YPos + Top),
      true, true
    );
    if Assigned(XControl) and (XControl is TWinControl) then
    begin
      XPos := TWinControl(XControl).ScreenToClient(XPos);
      TWMMouse(Message).XPos := XPos.X;
      TWMMouse(Message).YPos := XPos.Y;
      PostMessage(
        TWinControl(XControl).Handle, Message.Msg,
        Message.WParam, Message.LParam
      );
    end
    else
    begin
      XPos := Parent.ScreenToClient(XPos);
      TWMMouse(Message).XPos := XPos.X;
      TWMMouse(Message).YPos := XPos.Y;
      PostMessage(Parent.Handle, Message.Msg, Message.WParam, Message.LParam);
    end;
    Message.Result := 0;
  end
  else
    inherited WndProc(Message);
end;

procedure TMyPopUpTransPanel.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);
  if not (csDesigning in ComponentState) then
    Params.ExStyle := Params.ExStyle or WS_EX_TRANSPARENT;
end;

procedure TMyPopUpTransPanel.Paint;
var
  XBitMap: TBitMap;
  XOldDC: HDC;
  XRect: TRect;
begin
  if (csDesigning in ComponentState) then
    inherited Paint
  else
  begin
    XRect := ClientRect;
    XOldDC := Canvas.Handle;
    XBitMap := TBitMap.Create;
```

```
    try
      XBitMap.Height := Height;
      XBitMap.Width := Width;
      Canvas.Handle := XBitMap.Canvas.Handle;
      inherited Paint;
      RedrawWindow(
        Parent.Handle,
        @XRect, 0,
        RDW_ERASE or RDW_INVALIDATE or RDW_NOCHILDREN or RDW_UPDATENOW
      );
    finally
      Canvas.Handle := XOldDC;
      Canvas.BrushCopy(XRect, XBitMap, XRect, Color);
      XBitMap.Free;
    end;
  end;
end;
```

Tip by Serge Gubenko

Original resource:	The Delphi Pool
Author:	Various
Added:	2013-01-27
Last updated:	2013-01-27