TGradientFill Component V1.3

Copyright / License
How To Contact Us

Unit

GFVcl

Description

The TGradientFill component displays a gradient filled rectangle on the form. If you want to change the starting color of the fill, set the BeginColor property to the color you want the fill to start with. If you want to set the ending color of the fill, set the EndColor property to the color you want the fill to end with. If you want to change the direction of the fill, set the FillDirection property to the direction you want to fill. You can set the number of colors to use in the fill by setting the NumberOfColors property to the number of colors to be used. The HoldRedraw property allows multiple properties to be set without re-drawing the gradient between each setting. There are, also, properties to allow the TGradientFill to be used as a background to a MDI parent form. The TGradientFill has the ability to use a realized palette by setting the Realize property. This eliminates Windows dithering.

In addition to these properties, this component also has the properties, methods, and events that apply to all controls.

Properties

The additional properties supported by the TGradientFill component contain links. See the Delphi help for descriptions of the remaining properties.

Align	Height		PopupMenu	
Autosize	Hint		Realize	
BeginColor		HoldRedraw		ShowHint
Cursor	Left		Tag	
DragCursor	Name		Top	
DragMode		NumberOfColors		Visible
Enabled	PaletteHandle		Width	
EndColor		ParentShowHint		
FillDirection	Picture			

GFVcl Unit

The GFVcl unit contains the declaration for the TGradientFill component and its associated objects.

When you add a component declared in this unit to a form, the unit is automatically added to the uses clause of that form's unit.

The following items are declared in the GFVcl unit:

Types

TFillDirection

TNumberOfColors

Components

TGradientFill

To see a listing of items declared in this unit including their declarations, use the ObjectBrower.

BeginColor Property

Example

Applies to

TGradientFill component

Declaration

property BeginColor: TColor;

Description

The BeginColor property determines the starting color for the fill. This color is also used as the fill color when the NumberOfColors property is set to one. The default value of BeginColor is clBlue.

These are

the possible values of BeginColor:

Value	Meaning
clBlack	Black
clMaroon	Maroon
clGreen	Green
clOlive	Olive green
clNavy	Navy blue
clPurple	Purple
clTeal	Teal
clGray	Gray
clSilver	Silver
clRed	Red
clLime	Lime green
clBlue	Blue
clFuchsia	Fuchsia
clAqua	Aqua
clWhite	White
clBackground	Current color of your Windows background
clActiveCaption	Current color of the title bar of the active window
clInactiveCaption	Current color of the title bar of inactive windows
clMenu	Current background color of menus
clWindow	Current background color of windows
clWindowFrame	Current color of window frames
clMenuText	Current color of text on menus
clWindowText	Current color of text in windows
clCaptionText	Current color of the text on the title bar of the active window

clActiveBorder	Current border color of the active window
clInactiveBorder	Current border color of inactive windows
clAppWorkSpace	Current color of the application workspace
clHighlight	Current background color of selected text
clHightlightText	Current color of selected text
clBtnFace	Current color of a button face
clBtnShadow	Current color of a shadow cast by a button
clGrayText	Current color of text that is dimmed
clBtnText	Current color of text on a button
clInactiveCaptionText	Current color of the text on the title bar of an inactive window
clBtnHighlight	Current color of the highlighting on a button

The second half of the colors listed here are Windows system colors. The color that appears depends on the color scheme users are using for Windows. Users can change these colors using the Control Panel in Program Manager. The actual color that appears will vary from system to system. For example, the color fuchsia may appear more blue on one system than another. When you use the Color dialog box to select a color, you are assigning a new color value to the dialog box's BeginColor property. You can then use the value within the BeginColor property and assign it to the Color property of another control.

Example

This code sets the starting color of a gradient fill to red:

```
  TGradientFill1.BeginColor := clRed;
```

The following code changes the starting color of a gradient fill control using the Color dialog box. The example displays the Color dialog box when the Button1 button is clicked, allowing the user to select a color with the dialog box. The example then assigns the color value selected with the dialog box to the BeginColor property of the Gradient Fill control:

```
procedure TForm1.TButton1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    GradientFill1.BeginColor := ColorDialog1.Color;
end;
```

EndColor Property

Example

Applies to

TGradientFill component

Declaration

property EndColor: TColor;

Description

The EndColor property determines the ending color for the fill. The default value of EndColor is clBlack.

These are the possible values of EndColor:

Value	Meaning
clBlack	Black
clMaroon	Maroon
clGreen	Green
clOlive	Olive green
clNavy	Navy blue
clPurple	Purple
clTeal	Teal
clGray	Gray
clSilver	Silver
clRed	Red
clLime	Lime green
clBlue	Blue
clFuchsia	Fuchsia
clAqua	Aqua
clWhite	White
clBackground	Current color of your Windows background
clActiveCaption	Current color of the title bar of the active window
clInactiveCaption	Current color of the title bar of inactive windows
clMenu	Current background color of menus
clWindow	Current background color of windows
clWindowFrame	Current color of window frames
clMenuText	Current color of text on menus
clWindowText	Current color of text in windows
clCaptionText	Current color of the text on the title bar of the active window
clActiveBorder	Current border color of the active window
clInactiveBorder	Current border color of inactive windows

clAppWorkSpace	Current color of the application workspace
clHighlight	Current background color of selected text
clHightlightText	Current color of selected text
clBtnFace	Current color of a button face
clBtnShadow	Current color of a shadow cast by a button
clGrayText	Current color of text that is dimmed
clBtnText	Current color of text on a button
clInactiveCaptionText	Current color of the text on the title bar of an inactive window
clBtnHighlight	Current color of the highlighting on a button

The second half of the colors listed here are Windows system colors. The color that appears depends on the color scheme users are using for Windows. Users can change these colors using the Control Panel in Program Manager. The actual color that appears will vary from system to system. For example, the color fuchsia may appear more blue on one system than another. When you use the Color dialog box to select a color, you are assigning a new color value to the dialog box's EndColor property. You can then use the value within the EndColor property and assign it to the Color property of another control.

Example

This code sets the ending color of a gradient fill to green:

```
  TGradientFill1.EndColor := clGreen;
```

The following code changes the ending color of a gradient fill control using the Color dialog box. The example displays the Color dialog box when the Button1 button is clicked, allowing the user to select a color with the dialog box. The example then assigns the color value selected with the dialog box to the EndColor property of the Gradient Fill control:

```
procedure TForm1.TButton1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    GradientFill1.EndColor := ColorDialog1.Color;
end;
```

FillDirection Property

Example

Applies to

TGradientFill component

Declaration

property FillDirection: TFillDirection;

Description

The FillDirection property determines the direction of the fill. These are the possible values:

Value	Meaning
fdTopToBottom	Fill from BeginColor at the Top to EndColor at the Bottom.
fdBottomToTop	Fill from BeginColor at the Bottom to EndColor at the Top.
fdLeftToRight	Fill from BeginColor at the Left to EndColor at the Right.
fdRightToLeft	Fill from BeginColor at the Right to EndColor at the Left.

The default value of FillDirection is fdTopToBottom.

TFillDirection Type

Unit

GFVcl

Declaration

TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);

Description

The TFillDirection type contains the values the FillDirection property can assume.

Example

This example uses a gradient fill component and a button named ChangeFill on a form. The code changes the direction of the fill when the user clicks the button by changing the FillDirection property.

```
procedure TForm1.CreatFillClick(Sender: TObject);
begin
  with GradientFill1 do
    begin
      case FillDirection of
        fdTopToBottom:  FillDirection := fdBottomToTop;
        fdBottomToTop:  FillDirection := fdLeftToRight;
        fdLeftToRight:  FillDirection := fdRightToLeft;
        fdRightToLeft:  FillDirection := fdTopToBottom;
      end;
    end;
  end;
end;
```

HoldRedraw Property

Example

Applies to

TGradientFill component

Declaration

property HoldRedraw: Boolean;

Description

This property is used to turn on and off the re-draw capabilities of the Gradient Fill component. If it is set to True, then the gradient is not re-drawn. When it is set to False, re-drawing resumes. This allows properties to be set without a re-draw happening between each change. When all changes are complete, then the HoldRedraw property can be set to False and the gradient will be re-drawn with all the new settings in place. A normal use for this might be in setting both the begin and end colors. Both can be set, then the re-draw can be turned on and the color change takes place all at once.

Example

This example turns off the re-draw, sets the Gradient Fill begin and end colors, then turns the re-draw back on, when a button is clicked, so the gradient is painted in the new colors.

```
procedure TForm1.TButton1Click(Sender: TObject);
begin
  GradientFill1.HoldRedraw := True;
  GradientFill1.BeginColor := clRed;
  GradientFill1.EndColor := clWhite;
  GradientFill1.HoldRedraw := False;
end;
```

NumberOfColors Property

Example

Applies to

TGradientFill component

Declaration

property NumberOfColors: TNumberOfColors;

Description

The NumberOfColors property determines the number of colors to be used in the fill. The value can range from 1 to 255. The default is 16.

TNumberOfColors Type

Unit

GFVcl

Declaration

TNumberOfColors: 1..255;

Description

The TNumberOfColors type defines the possible values of the NumberOfColors property for a gradient fill component.

Example

This example sets the number of colors to use in a gradient fill component to the maximum number of colors the video driver allows (up to 255).

```
procedure TForm1.FormShow(Sender: TObject);
var
  MaxColors: Longint;
  ScreenHandle: HDC
begin
  ScreenHandle := GetDC(0);
  MaxColors := GetDeviceCaps(ScreenHandle, NUMCOLORS);
  if MaxColors < 256 then
    GradientFill1.NumberOfColors := MaxColors
  else
    GradientFill1.NumberOfColors := 255;
  ReleaseDC(0, ScreenHandle);
end;
```

PaletteHandle Property

Example

Applies to

TGradientFill component

Declaration

property PaletteHandle: HPalette;

Description

A handle to the palette that is created when the component property, Realize, is set to True. The primary use of this property is to realize the palette for the component that contains the Gradient Fill component. This was primarily set up to allow the Gradient Fill component to be used as a background to an MDI parent form. If Realize is set to False, the PaletteHandle property will be Nil.

Example

This example realizes the forms palette to that of the Gradient Fill component. It then tiles the Gradient Fill bitmap (in the Picture property) in the background of the MDI parent window. This should only need to be done if the form is an MDI parent.

```
procedure TForm1.ClientWndProc(VAR Message: TMessage);
var
  MyDC : hDC;
  Ro, Co : Word;
begin
  with Message do
    case Msg of
      WM_ERASEBKGND:
        begin
          MyDC := TWMEraseBkGnd(Message).DC;

          { Set Canvas's palette to the new one }
          SelectPalette(MyDC, GradientFill1.PaletteHandle, False);
          { Make Canvas recognize the palette }
          RealizePalette(MyDC);

          for Ro := 0 TO ClientHeight DIV GradientFill1.Picture.Height DO
            for Co := 0 TO ClientWIDTH DIV GradientFill1.Picture.Width DO
              BitBlt(MyDC, Co*GradientFill1.Picture.Width,
                     Ro*GradientFill1.Picture.Height,
                     GradientFill1.Picture.Width,
                     GradientFill1.Picture.Height,
                     GradientFill1.Picture.Canvas.Handle,
                     0, 0, SRCCOPY);

          Result := 1;
        end;
    else
      Result := CallWindowProc(FPrevClientProc, ClientHandle, Msg, wParam, lParam);
    end;
end;
```

This example is the same as the above example except it stretches the gradient to the new size instead of tiling it.

```
procedure TForm1.ClientWndProc(VAR Message: TMessage);
var
  MyDC : hDC;
```

```
  Ro, Co : Word;
begin
  with Message do
    case Msg of
      WM_ERASEBKGND:
        begin
          MyDC := TWMEraseBkGnd(Message).DC;

          { Set Canvas's palette to the new one }
          SelectPalette(MyDC, GradientFill1.PaletteHandle, False);
          { Make Canvas recognize the palette }
          RealizePalette(MyDC);

          StretchBlt(MyDC, 0, 0,
                     ClientWidth,
                     ClientHeight,
                     GradientFill1.Picture.Canvas.Handle,
                     0, 0,
                     GradientFill1.Picture.Width,
                     GradientFill1.Picture.Height,
                     SRCCOPY);

          Result := 1;
        end;
    else
      Result := CallWindowProc(FPrevClientProc, ClientHandle, Msg, wParam, lParam);
    end;
end;
```

Picture Property

Example

Applies to

TGradientFill component

Declaration

property Picture: TBitmap;

Description

This property contains the Bitmap that the Gradient Fill is drawn to. This was primarily set up to allow the Gradient Fill component to be used as a background to an MDI parent form. The bitmap can be copied to any object, though, just as any bitmap can.

Realize Property

Example

Applies to

TGradientFill component

Declaration

property Realize: Boolean;

Description

This property is used to determine whether the Gradient Fill uses a realized palette to draw with or not. If it is set to True, then a realized palette will be used. If set to False, the standard palette is used. When running in true-color mode, it makes no difference whether a realized palette is used or not. But if running in 256 color, or less, mode this property makes the Gradient Fill look much nicer. This is because Windows attempts to dither the colors if no realized palette is used. If this dithered effect is what you are looking for or you are running in true-color mode, then set this property to False. Otherwise it should be set to True. If this property is set to True, the handle to the palette is stored in the PaletteHandle property.

Example

This example sets the Gradient Fill to use the standard palette instead of a realized palette when a button is clicked.

```
procedure TForm1.TButton1Click(Sender: TObject);
begin
  GradientFill1.Realize := False;
end;
```

Methods

The following are the methods provided by the TGradientFill component.

ReRealize

ReRealize Method

Example

Applies to

TGradientFill component

Declaration

procedure ReRealize(OrigWnd: hWnd);

Description

The ReRealize method allows the GradientFill to realize it's palette as a background palette. The method is designed to be called by the owner application if it receives a WM_PALETTECHANGED message. This will attempt to keep the colors as close as possible on a 256 color system. When calling this method, you need to pass it the handle of the form or control that is the owner of the GradientFill component. This routine was contributed by Colin Messitt (74774.1347@compuserve.com).

Example

This example realizes the palette as a background palette.

```
procedure TForm1.ClientWndProc(VAR Message: TMessage);
begin
  with Message do
    case Msg of
      WM_PALETTECHANGED:
        begin
          TGradientFill1.ReRealize(ClientHandle);
          Result := 1;
        end;
    else
      Result := CallWindowProc(FPrevClientProc, ClientHandle, Msg, wParam, lParam);
    end;
end;
```

Events

See the Delphi help for descriptions of the events.

OnClick	OnEndDrag
OnDblClick	OnMouseDown
OnDragDrop	OnMouseMove
OnDragOver	OnMouseUp

Using the TGradientFill Component

TGradientFill Reference

Purpose

Use the TGradientFill component to fill a rectangular area with a gradient starting with one color and ending with another. If only one color is specified, then the entire rectangle will be filled with the starting color.

Tasks

Adding a GradientFill component to a Form

Setting the colors relative to FillDirection

Specifying the number of colors to be used in the gradient fill

Adding a GradientFill component to a Form

The GradientFill component allows you to specify an area on a form that will contain a filled gradient rectangle. The component can be aligned to the client, which will make it appear as though the entire Form being filled with the gradient.

Placing the Component

You can place a GradientFill component anywhere on a form just like any other visual component. It can be sized to any size within the form.

Setting the colors relative to FillDirection

You can set the begin and end colors of the GradientFill component. Set the BeginColor property to set the starting color. Set the EndColor property to set the ending color. These colors are relative to the direction of the fill. If the FillDirection is set to TopToBottom, the BeginColor will be the color at the top of the gradient fill and the EndColor will be at the Bottom. If the FillDirection is set to BottomToTop, the BeginColor will be the color at the bottom of the gradient fill and the EndColor will be at the Top. If the FillDirection is set to LeftToRight, the BeginColor will be the left most color and the EndColor will be the right most color of the gradient fill. Finally, if the FillDirection is set to RightToLeft, the BeginColor will be the right most color and the EndColor will be the left most color of the gradient fill.

Specifying the number of colors to be used in the gradient fill

The number of colors to be used in the gradient fill can be set with the NumberOfColors property. This allows the gradient to be used on systems that do not have the ability to display 256 colors or to get a banding effect, if the programmer so desires. Some of the older VGA graphics cards only allow 16 colors to be displayed on the screen simultaneously. The gradient looks best at 255 colors. The number of colors to be used should be determined at startup of the program (using GetDeviceCaps) and the NumberOfColors property set appropriately. If this is not done and the program is being run on a system that cannot display 255 colors and the NumberOfColors property is set to 255, a General Protection Fault could occur.

Copyright / License

TGradientFill Component for Delphi.

Disclaimer

This component and the associated help file are the Copyright of TechnoSoft, Inc. They are distributed as Freeware. They can be freely used and distributed in commercial and private environments, provided this notice is not modified in any way without my expressed written consent. The TGradientFill Component is released as is. No warranty is implied and I am not responsible for any problems that might arise from the use of this component. You use this component entirely at your own risk.

For information on how to contact TechnoSoft, Inc. see How To Contact Us.

Bugs (a.k.a. undocumented features)

If you find any, please let me know, and I will attempt to fix them. Note that bugs often depend on glitches in the system. Sometimes it helps to re-boot and try again...

Feel free to contact me if you have any questions, comments or suggestions at cwhite@teleport.com

---

How To Contact Us

You can contact us in any of the following ways. Address all inquiries to:

Curtis White

(President, TechnoSoft, Inc.)

Internet E-Mail:

cwhite@teleport.com

Compuserve Mail:

102116,2616

Snail Mail:

TechnoSoft, Inc.

3795 SW 194th Place

Beaverton, OR 97007

Also feel free to check out our internet World Wide Web page at:

http://www.teleport.com/~cwhite/index.html

You will find a lot of Freeware and Shareware components, examples, information, and tips for Delphi.