TImageFX Component

Description

TImageFX is a Delphi component which allows you to specify 1 of 20 special effects when displaying a bitmap file. These ***Effect*** properties can be grouped in 4 categories:

Explode: image starts in the center of the image area as a small point and grows in all directions until it fills the image area. Three explode effects: ViewExplode, ViewZoomIn, ViewZoomOut.

Implode: image starts as full sized and shrinks to a point in the center of the image then disappears. Three implode effects: HideImplode, HideZoomIn, HideZoomOut.

Wipe: image starts from one of its sides (top, bottom, left or right) and grows to the opposite side. Twelve wipe effects: Wipe____Roll, Wipe____Slide, Wipe____Stretch. Fill in the blanks with Left, Right, Up or Down.

Curtain: image starts from left and right or top and bottom and grows toward the center from two directions. Two curtain effects: SideCurtainRoll, VertCurtainRoll.

Run the example file FXDEM101.EXE or compile FXDEM101.DPR to see examples of these effects. If your ImageFX.DCU only shows effects when Delphi is running, you have the demo version. Get the full VCL from CompuServe—GO SWREG and register # 11822.

Properties

AutoSize

Busy

Effect

EffectDelay

EffectRate

For all other properties, see online help for TPaintBox.

Methods

LoadFromFile

ShowEffect

Events

OnComplete

Using the TImageFX component

To use TImageFX, place one on a form, select an Effect, set EffectRate and EffectDelay to non-zero values and add the following 2 lines to a relevant place in your application:

```
ImageFX1.LoadFromFile('filename.bmp');
ImageFX1.ShowEffect;
```

To make TImageFX automatically size itself to fit the bitmap it contains, set AutoSize := True. The upper left corner (Left and Top properties) determine the placement of the displayed bitmap. The displayed bitmap is buffered so there is no flicker in the image as it displays an effect.

If an effect does not seem to Show:

When you use ViewExplode, HideImplode and Wipe___Roll to paint a bitmap over itself, the effect will not be visible. This is because the bitmap is gradually being replaced by an exact copy of itself, so no change is visible. Use a Hide effect first (which leaves no bitmap showing) or show a different bitmap with no effect before using these effects.

Setting `TImageFX.Visible := False;` terminates an effect in progress and makes the component invisible.

See also:

AutoSize property

Busy property

Effects property

EffectDelay property

EffectRate property

LoadFromFile method

ShowEffect method

OnComplete event

 AutoSize property

Declaration:

```
property AutoSize: Boolean;
```

Description:

When AutoSize is True, the TImageFX grows or shrinks to fit the bitmap it contains.

When AutoSize is False, the control remains the same size regardless of the bitmap size. If the control is smaller than the bitmap, only the portion of the bitmap that fits in the TImageFX is visible.

The default value is False.

Note: You must set AutoSize property := True before loading the picture or it has no effect.

Effect property

Declaration

```
TEffect = (HideImplode, HideZoomIn, HideZoomOut,
           ViewExplode, ViewZoomIn, ViewZoomOut,
           WipeRightRoll, WipeRightSlide, WipeRightStretch,
           WipeLeftRoll, WipeLeftSlide, WipeLeftStretch,
           WipeUpRoll, WipeUpSlide, WipeUpStretch,
           WipeDownRoll, WipeDownSlide, WipeDownStretch,
           SideCurtainRoll
           VertCurtainRoll);
```

Example: `ImageFX1.Effect := ViewExplode;`

Description:

The Effect property determines how a bitmap is drawn when it is displayed. The selected effect is drawn flicker free at the size of the component (AutoSize = False) or at the size of the bitmap loaded via the LoadFromFile method (AutoSize = True). The Left Top corner of the TImageFX component determines bitmap placement.

HideImplode:	normal sized image gradually disappears from its outer perimeter to its center
HideZoomIn	gets smaller and smaller until gone with perimeter bands still there, then disappears
HideZoomOut	expands from center until center few pixels fill the image, then disappears
ViewExplode	normal sized image is exposed in increments from a point in center until entire image shows
ViewZoomIn	enlarged image of a few large pixels filling image area shrinks until normal sized image shows
ViewZoomOut	small sized image grows from a point in center of image area until normal sized image fills area
WipeRightRoll	normal sized image gradually appears from left of image area and fills to the right
WipeRightSlide	“squished” image expands from left to right of image area until full
WipeRightStretch	enlarged image gradually shrinks from left to right until normal image fills area
WipeLeftRoll	same as WipeRightRoll, opposite direction
WipeLeftSlide	same as WipeRightSlide, opposite direction
WipeLeftStretch	same as WipeRightStretch, opposite direction
WipeUpRoll	same as WipeRightRoll, bottom to top rather than left to right direction
WipeUpSlide	same as WipeRightSlide, bottom to top rather than left to right direction
WipeUpStretch	same as WipeRightStretch, bottom to top rather than left to right direction
WipeDownRoll	same as WipeUpRoll, opposite direction
WipeDownSlide	same as WipeUpSlide, opposite direction
WipeDownStretch	same as WipeUpStretch, opposite direction
SideCurtainRoll	normal sized image appears gradually from left and right sides, fills horizontally to center
VertCurtainRoll	normal sized image appears gradually from top and bottom, fills vertically to center

EffectRate property

Declaration

```
property EffectRate: TRateRange;

TRateRange: 0..100;
```

Description:

Sets the percent of the bitmap which is drawn on each redraw update. Valid values are any integer between 0 and 100; higher values make the effect happen faster, lower values give a smoother effect. Generally, a value of 1 to 20 gives a nice effect; 1 gives 100 redraws and 20 gives 5 redraws to complete the image.

A value of 0 halts the effect but the effect trigger continues to occur every EffectDelay period until EffectRate is set to a non-zero value and the effect is completed. This allows an application to ‘‘pause’’ an effect.

EffectDelay property

Declaration

```
property EffectDelay: TDelayRange;

TDelayRange: 0..6000;
```

Description:

Sets the time delay between each redraw update. Valid values are numbers between 0 and 6000. The higher the value the slower the effect. Generally, a value of 1 to 10 gives a nice effect; 1 gives 1/100 second between redraws and 10 gives 1/10 second between redraws to complete the image; 6000 gives 60 seconds between update redraws.

A value of 0 halts the effect, stops the redraw delay timer and displays the most recent image (if any). If the effect is halted with part of an image displayed, the partial image remains.

Busy property

Declaration

```
property Busy: Boolean;
```

Description:

Run-time and read only. True when an effect is being drawn, otherwise False. Allows an application to monitor when an effect is underway and when it is complete.

LoadFromFile method

Declaration

```
procedure LoadFromFile(FileName: string);
```

Description

Loads a Windows bitmap (.BMP) file into the component for display. The file is not displayed until EffectDelay and EffectRate are set to non-zero values and the ShowEffect method is called.

ShowEffect method

Declaration

```
procedure ShowEffect;
```

Description

Starts the process of displaying a bitmap with the selected Effect. Completion of the effect can be determined by monitoring the Busy property or assigning the OnComplete event.

ShowEffect has no effect if Visible = False;

OnComplete event

Applies to: TImageFX

Declaration:

```
property OnComplete: TNotifyEvent;
```

An assigned OnComplete event occurs when an effect is finished drawing (bitmap is completely drawn) or when an effect is terminated by setting EffectDelay := 0 or Visible := False;

 **TImageFX Component for Delphi ©1996 by Beond Technology Corp. All Rights Reserved**

Design Only version

If your TImageFX only works in design mode, you can get the fully functional version from CompuServe's shareware forum (GO SWREG and search for components by 76640,2664 or register number 11822) or from:

Beond Technology Corp.
15370 W. Cherrywood Lane
Libertyville, IL 60048-1435
(708) 918-7750 (V/F)
CompuServe: 76640,2664
Internet: brianlow@mcs.com

Limited Warranty

Because you can completely evaluate it before you buy it, this software has no warranty whatsoever. This non–warranty is in lieu of any other warranty, expressed or implied, including the implied warranties of merchantability and fitness for a particular purpose. In no event will Beond Technology Corp. be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out of your use of or inability to use the software.

Support

The best support is via e-mail…send a clear question and you get a clear answer. Vague questions will be answered as well as possible. If you find a bug, e-mail a description and an example app which shows what's going wrong. If you want a modified version and are willing to pay for development or want to buy source code, e-mail your requirements or call.