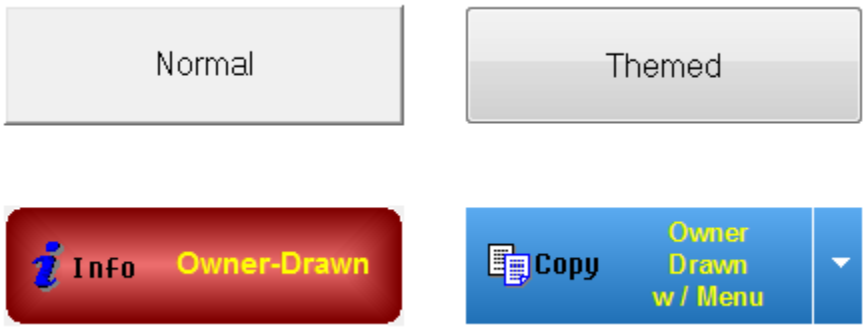
Xbase++ Owner-Drawing

Introduction

Xbase++ 1.9 has opened up some exciting new possibilities for incorporating new methods of control over the visual aspects of an application. In prior versions of Xbase++, drawing of Xbase Parts objects such as XbpPushButton() were accomplished by the operating system. Owner-drawing gives the programmer the option of drawing all or portions of the object using the Xbase++ graphics engine. For example, the top two buttons shown below are the styles available **without owner-draw** whereas the bottom two buttons shown below are an example of the styles available **with owner-draw**.



The purpose of this seminar is to provide a general overview of the basics of owner-drawing and to also provide a tutorial and source code for a powerful new pushbutton class and a new message box class.

Classes that support Owner-Drawing

Xbase++ does not support owner-drawing for all Xbase Parts classes, however it does support it for the following classes:

XbpPushButton, XbpStatic, XbpMenu, XbpMenuBar, XbpComboBox, XbpCellGroup, XbpListBox, XbpTabPage, XbpQuickBrowse, XbpMultiCellGroup, XbpColumn

These classes all support a new instance variable named **:drawMode.** This iVar specifies the mode used to display the object on the screen. In the default draw mode (XBP_DRAW_NORMAL), the operating system takes full control over the drawing process. In this mode, the object is displayed using default imagery and the object's appearance cannot be controlled by the application. However, this default behavior can be overridden. If the value XBP_DRAW_OWNER or XBP_DRAW_OWNERADVANCED is assigned to the **:drawMode** instance variable, no drawing is performed by the operating system.

Instead, the Xbase++ runtime system uses xbeP_Draw events to notify the application whenever the Xbase part needs to be redrawn. The application is responsible for processing this event and for creating the on-screen visualization. For this, the Graphics Engine commands can be used.

This protocol is called owner-drawing because displaying the object is the responsibility of the objects's owner. The drawing mode must be assigned to :drawMode before the method :create() is executed.

The Draw() method

The :draw() method is used to perform the actual drawing of the object and this method is called when an xbeP_Draw event is generated. The draw event is generated whenever an owner-drawn object must be redrawn. If a code block is assigned to the :draw instance variable, that code block is also evaluated.

Prior to drawing an object's current state, an Xbase++ application is expected to examine the information passed in the parameter <aInfo>. The elements in the array specify the object's current state and dimensions, as well as the operation which caused the redraw of the object. The data in the <aInfo> array can be identified using symbolic constants defined in the file "xbp.ch".

```
  Slot:       :draw := {| oPS, aInfo, self | ... }

Method:       :draw( oPS, aInfo ) --> self | lDefaultDraw
```

Parameter <oPS> contains a Micro Presentation Space object that must be used to draw the static object. It can be used with the GRA functions to define the static's imagery.

<aInfo> := {nItem, nAction, nState, aRect}

<aInfo> is an array with four elements with information about the operation which caused the redraw as

This method returns the object executing the method (self). The class also supports a logical value to be returned from method :draw() , or from a code block assigned to member :draw . Returning self or the value .T. (true) causes the object to perform default rendering. This depends on the drawing mode specified in :drawMode, as well as on the current state of the object

Elements of the array in parameter <aInfo>

Element	Constant	Description
nItem	XBP_DRAWINFO_ITEM	Not used with all owner-drawn classes
nAction	XBP_DRAWINFO_ACTION	Action or operation which caused xbeP_Draw event
nState	XBP_DRAWINFO_STATE	Current state of the object
aRect	XBP_DRAWINFO_RECT	Bounding rectangle for the drawing operation

Caveats

The xbeP_Draw event is sent synchronously by the :handleEvent() method of an object. This implies that xbeP_Draw events will not be retrieved by the AppEvent() function. The event is generated while the operating system redraws an application's user interface. Therefore, it is not advisable to perform lengthy operations during processing of xbeP_Draw. Otherwise, responsiveness of the application's user interface may be slow.

In Xbase++, Xbase Parts are created and maintained by a special system thread called the UI thread. Both the :draw() callback method and the :draw callback code block are executed on that thread. Therefore, during processing of an xbeP_Draw message, the application must assume all thread-local settings to contain default values. For instance, a work area opened in an application thread may not be available. It is recommended to prepare all the data required for rendering before the static object is created. This ensures fast processing of xbeP_Draw events and prevents thread-dependencies.

Debugging functions in the :draw() method must not create screen objects or the application will freeze.

The XbpPushButtonXP() class

It is not practical to discuss all the fine details of owner-drawing for each Xbase parts class in this seminar, so I will focus mostly on a single class, the **XbpPushButtonXP()** class, because it is typical of how owner-drawing is accomplished for other classes as well. Additionally, custom push buttons are the most commonly requested modification of Xbase classes. We will discuss a new class which inherits from the **XbpPushButton()** class and uses owner-drawing exclusively to render the buttons.

The source code for the XbpPushButtonXP() class is contained in XbpPushButton.Prg and the test program that uses the class is shown below. This new class supports the following features:

- Radiused corners
- Gradient options
- Multiple bitmaps, icons and caption strings
- A drop-down menu option
- MouseOver effects, such as change size, color, font and sounds
- Color options for normal, mouse-over, clicked and disabled states
- Shadowing of button and captions
- Bitmap tiling
- Caption and bitmap alignment options
- Focus rectangle options
- Border color
- Transparency
- Can be used as a set of Radio Buttons with one selected

The XbpPushButtonXPConfig() Class

When designing a new class that uses owner-drawing, it is important to also create a configuration class that is used to set all the default values. This configuration class can be used to establish a **theme** for a group of buttons or for an entire application. The **XbpPushButtonXPConfig()** class is used for this purpose.

A Test Program

```
#INCLUDE "Appevent.CH"

FUNCTION Main()

LOCAL oButton, nEvent, mp1, mp2, oXbp, oDlg, oButtonConfig, aMessage

oDlg := XbpDialog():new(,,{300,300},{800,500},,.f.)
oDlg:taskList := .t.
oDlg:title := 'PushButton Styles'
oDlg:drawingArea:colorBG := GRA_CLR_WHITE
oDlg:create()

oButtonConfig := XbpPushButtonXPConfig():new()
oButtonConfig:radius := 20
oButtonConfig:gradientStep := 4
oButtonConfig:gradientStyle := 1
oButtonConfig:mouseOverScale := 1.1
oButtonConfig:font := '12.Arial Bold'
oButtonConfig:mouseOverFont := '13.Arial Bold'
oButtonConfig:bgColor := GraMakeRGBColor({ 127,0,0 })
oButtonConfig:fgColor := GRA_CLR_YELLOW
oButtonConfig:bgColorMouse := GraMakeRGBColor({ 50,100,190 })
oButtonConfig:fgColorMouse := GRA_CLR_YELLOW
oButtonConfig:bgColorClick := GraMakeRGBColor({ 190,100,50 })
oButtonConfig:fgColorClick := GRA_CLR_DARKPINK

oButton := XbpPushButton():new(oDlg:drawingArea,,{100,300},{200,60})
oButton:caption := 'Normal'
oButton:useVisualStyle := .f.
oButton:setFontCompoundName('12.Helv')
oButton:activate := {||MsgBox('I am a "Normal" button')}
oButton:create()

oButton := XbpPushButton():new(oDlg:drawingArea,,{330,300},{200,60})
oButton:caption := 'Themed'
oButton:setFontCompoundName('12.Helv')
oButton:activate := {||MsgBox('I am a "Themed" button')}
oButton:useVisualStyle := .t.
oButton:create()

oButton := XbpPushButtonXP():new(oDlg:drawingArea,,{100,200},{200,60})
oButton:caption := 'Owner-Drawn'
oButton:config := oButtonConfig
oButton:bitmap := 'c:\expd19\bitmaps\info1.bmp'
oButton:bitmapOffset := 10
oButton:actionBlock := {||MsgBox('I am an "Owner-Drawn" button')}
oButton:create()

oButton := XbpPushButtonXP():new(oDlg:drawingArea,,{330,200},{200,60})
oButton:caption := 'Owner;Drawn;w / Menu'
oButton:bitmap := 'c:\expd19\bitmaps\copy1.bmp'
oButton:bitmapOffset := 10
oButton:mouseOverScale := 1.1
oButton:font := '11.Arial Bold'
oButton:gradientStep := 1
oButton:gradientReverse := .t.
```

```
oButton:mouseOverFont := '13.Arial Bold'
oButton:bgColor := GraMakeRGBColor({ 90,170,240 })
oButton:fgColor := GRA_CLR_YELLOW
oButton:bgColorMouse := GraMakeRGBColor({ 50,100,190 })
oButton:fgColorMouse := GRA_CLR_YELLOW
oButton:bgColorClick := GraMakeRGBColor({ 190,100,50 })
oButton:fgColorClick := GRA_CLR_YELLOW
oButton:actionBlock := {||MsgBox('I am an "Owner-Drawn with Menu" button')}
oButton:menuBlock := {||MsgboxCustom('Welcome Friends',aMessage)}
oButton:menuButtonColor := GRA_CLR_WHITE
oButton:menuButtonWidth := 25
oButton:menuButtonFont := '12.Marlett'
oButton:create()

aMessage := { 'Welcome to the Hanover Xbase++ Devcon.', ;
              'I hope you will enjoy this seminar', ;
              'about Owner-Drawing with Xbase++.', ;
              'as much as I enjoy German Beer.', ;
              '', ;
              'Danke'}

oDlg:show()

DO WHILE nEvent # xbeP_Close
  nEvent := AppEvent(@mp1,@mp2,@oXbp,.1)
  IF nEvent > 0
    oXbp:handleEvent(nEvent,mp1,mp2)
  ENDIF
ENDDO

oDlg:destroy()

RETURN nil
```

The Graphics Engine

The Xbase++ Graphics engine contains a set of functions that are used to render the object with the desired attributes and are used by the **:draw()** method. This is where the largest investment of time is expended when writing a new class that uses owner-draw features. Most Xbase++ programmers have developed an expertise in business applications and have seldom been exposed to graphics primitives. Don't be put off by your inexperience in this area because this is not as difficult as it may seem. The XbpPushButtonXP() class uses the following graphics functions:

- GraSetAttrArea() – Used to set the attributes, such as colors and border width, for subsequent area-fill operations using GraBox().

- GraSetAttrLine() – Used to set the attributes, such as colors and line width, for subsequent line drawing operations using GraLine().

- GraBox() – Used to draw a box, with optional fill and border characteristics, including radiuses. The class uses this function to draw the perimeter of the button, the concentric gradient and the focus rectangle.

- GraLine() – Used to draw a line, with optional characteristics. The class uses this function to draw gradients on buttons with no radius and to draw the line that separates the main button area from the
- optional menu area.

- GraSetColor() – Used to set the foreground and background color for subsequent Gra*() function calls.

- GraSetFont() – Used to set the font for subsequent GraStringAt() and GraCaptionStr() function calls.

- GraCaptionStr() – Used to draw caption strings.
- GraQueryTextBox() – Used to determine the exact rectangle required to draw a caption based on the current font. The class uses this function to determine where to draw text based on alignment options.

- GraGetRGBIntensity() – Converts a numeric color to an RGB three-element array. The class uses this function to increment each RGB value when painting the gradient lines or boxes.

- GraMakeRGBColor() – Converts an RGB three-element array into a numeric color. The class uses this function because only numeric color values can be used with drawing functions.

- XbpBitmap():draw() – Draws a bitmap. The class uses this method to draw all bitmaps on the button.

- XbpIcon():draw() – Draws an icon. The class uses this method to draw icons on the button.

The XbpPushButtonXP() class description

Instance Variables

- **:focusRectStyle** – This determines the style of the focus rectangle that is painted around the perimeter of the button when the button has focus. 0 – none, 1 – dotted, 2 – solid 1-pixel wide, 3 – solid 2 pixels wide. (N)umeric
- **:focusRectColor** – The color of the focus rectangle. (N)umeric.
- **:radius** – The radius of the corners of the button. (N)umeric.
- **:outline** – Determines if the button is outlined with a solid line. (L)ogical
- **:fgColor** - The foreground color for captions. (N)umeric
- **:bgColor** – The background color. (N)umeric
- **:fgColorMouse** – The foreground color for captions when the mouse is over the button. (N)umeric
- **:bgColorMouse** – The background color when the mouse is over the button. (N)umeric
- **:fgColorClick** – The foreground color for captions when the mouse has been clicked. (N)umeric
- **:bgColorClick** – The background color when the mouse has been clicked. (N)umeric
- :**fgColorSelected** – The foreground color for captions when the button has been selected. (N)umeric See :enableSelect
- :**bgColorSelected** – The background color when the button has been selected. (N)umeric See :enableSelect

- :**alignCaption** – Used to determine how the caption is aligned on the button. Use XBPALIGN_* constants. (N)umeric
- :**alignBitmap** – Used to determine how the bitmap is aligned on the button. Use XBPALIGN_* constants. (N)umeric
- :**bitmap** – The bitmap to be drawn. May be (N)umeric, (C)haracter, (O)bject
- :**tileBitmap** – The background tile bitmap to be drawn. May be (N)umeric, (C)haracter, (O)bject
- :**font** – The font for the caption. May be (C)haracter or (O)bject.
- :**mouseOverScale** – The amount to scale the button size when the mouse is over the object (N)umeric.
- :**mouseOverSound** – A wave file to play when the mouse is moved over the object. (C)haracter
- :**mouseOverFont** – The font for the captions when the mouse is over the button. (C)haracter, (O)bject
- :**captionArray** – An optional array of captions and bitmaps to display on the button. (A)rray
- :**gradientStep** – The amount of color increment to use for each gradient level. (N)umeric
- :**gradientStyle** – The style of the gradient. 1 for top to bottom, 2 for bottom to top, 3 for left to right, 4 for right to left. Applicable only for buttons with 0 :gradient. (N)umeric
- :**gradientLevel** – The level of the gradient. (N)umeric
- :**gradientBlock** – An optional code block to use for painting the gradient. Parameters are self, oPS, mp2, nMode. (B)lock
- :**actionBlock** – A code block to evaluate when the button is clicked. (B)lock
- :**menuBlock** – A code block to evaluate when the menu dropdown is clicked. (B)lock
- :**menuButtonFont** – The font to use for the menu button arrow. Example: 12.Marlett. (C)haracter or (O)bject
- :**menuButtonColor** – The foreground color for the menu button arraow. (N)umeric
- :**captionOffset** – The amount of pixels of the offset for the caption when the :alignCaption is set to XBPALIGN_LEFT. (N)umeric
- :**disabledBGColor** – The color of the background when the button is disabled. (N)umeric
- :**disabledFGColor** – The color of the captions when the button is disabled. (N)umeric
- :**disabledBMP** – The bitmap to display when the button is disabled. (N)umeric
- :**config** – An XbpPushButtonXPConfig() object used for the default configuration. (O)bject
- :**enableSelect** – Enable “Select” mode to simulate a set of radio buttons. (L)ogical
- :**selected** – Button has been selected as 1 of many. (L)ogical
- :**borderColor** – The color of the border around the button. (N)umeric
- :**isStatic** – Button is a Static object with no action. (L)ogical
- :**shadowType** – The type of shadow to paint around the button, (1-9). (N)umeric
- :**isTransparent** – If true then the background color of the button becomes the background color of its parent. (L)ogical
- :**isTextShadow** – If true then a shadow is placed around the caption text. (L)ogical
- :**textShadowColor** – The color of the text shadow. (N)umeric
- :**textShadowOffset** – The offset of the text shadow, in pixels. (N)umeric

- **:bitmapTransparentColor** – The color in bitmaps that will show as transparent. (N)umeric

Methods

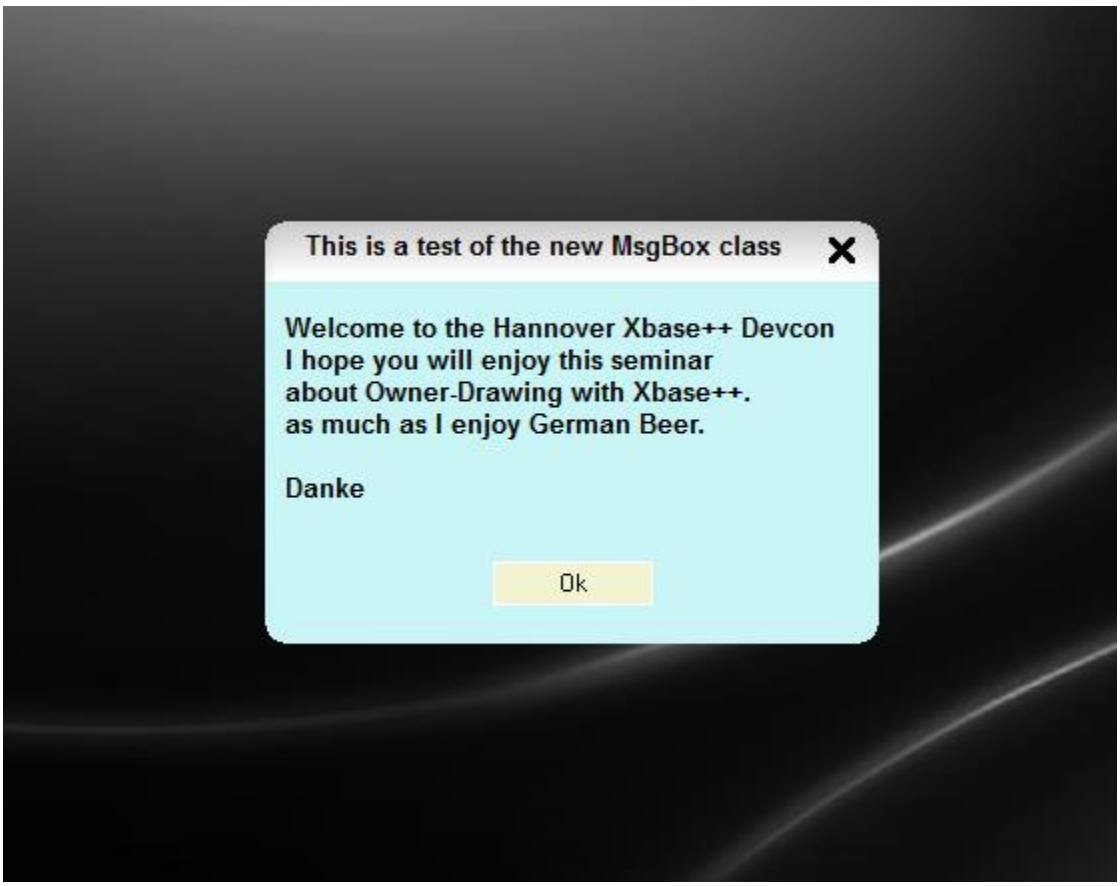
- :**init**() – The initialization routine.
- :**create**() – The creation routine.
- :**destroy**() – The destroy routine.
- :**setFontName**() – Used to change the font.
- :**setCaption**() – Used to change the caption.
- :**setBitmap**() – Used to change the bitmap.
- :**setBitmapTile**() – Used to change the background tile bitmap.
- :**setCaptionArray**() – Used to set a new array of captions, icons and bitmaps.
- :**setFontNameMouseOver**() – Used to set a new font for captions when mouse is over the button.
- :**refresh**() – Used to refresh the button.
- :**mouseEnter** – Called when the mouse is moved over the button.
- :**mouseLeave** – Called when the mouse is moved away from the button.
- :**draw**() – Called when the button need to be repainted.

The MsgBoxCustom() Function

The **XbpStatic()** class can be can be used, with its owner-draw features, to create a custom message box that has a better visual appeal than the MsgBox() function. The source code for the **MsgBoxCustom()** function and the **MsgBoxCustomStatic()** class is contained in **MsgBoxCustom.Prg**. This class was not intended to be a complete replacement for MsgBox() but instead is a starting point that can be used to create your own, more robust class.

Conclusion

A modified XbpPushButton class that uses owner-drawing can be used for many more functions than the generic class. A modified XbpStatic class that uses owner-drawing can be used to replace the Xbase++ MsgBox() function with a more appealing visual style. Here are a few examples.



f


XP Style Buttons
Window
Browse
Edit
Sort
Spin
Calculator
Calendar
<< Legacy Buttons
Click this Button to DISABLE the Button below
This is a Really Big Bug
This is a Button with a lot of text. It will word-wrap very nicely thank you!
Design
Gradient with Radius
Align Left VCenter
Align Right VCenter
Align Top HCenter
Align Bottom HCenter
Click this button to TURN OFF the display of source code on the Tooltip
This is Page 16
03/30/2012
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
Tab 6
Tab 7
Tab 8
This is Tab Page 3
Tab 9
Tab 10
Tab 11
Tab 12
Tab 13
Tab 14
Tab 15
Tab 16
Button with Halo
Button with Halo
Button with Halo