Arc

CreateArc (PictureHandle&, X1%, Y1%, X2%, Y2%, X3%, Y3%, X4%, Y4%) As Long
SCreateArc (PictureHandle&, X1#, Y1#, X2#, Y2#, X3#, Y3#, X4#, Y4#) As Long

X1: x-coordinate of the upper-left corner of the bounding rectangle.
Y1: y-coordinate of the upper-left corner of the bounding rectangle.
X2: x-coordinate of the lower-right corner of the bounding rectangle.
Y2: y-coordinate of the lower-right corner of the bounding rectangle.
X3: x-coordinate of the point that defines the arc's starting point.
Y3: y-coordinate of the point that defines the arc's starting point.
X4: x-coordinate of the point that defines the arc's endpoint.
Y4: y-coordinate of the point that defines the arc's endpoint.

Points 3 and 4 do not have to lie exactly on the arc.

Draws an elliptical arc. The arc drawn by using the function is a segment of the ellipse defined by the specified bounding rectangle. The actual starting point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified starting point intersects the ellipse. The actual ending point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified ending point intersects the ellipse. The arc is drawn in a counterclockwise direction using the current border attributes.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle for the arc object.
Scaling

Chord

CreateChord (PictureHandle&, X1%, Y1%, X2%, Y2%, X3%, Y3%, X4%, Y4%) As Long
SCreateChord (PictureHandle&, X1#, Y1#, X2#, Y2#, X3#, Y3#, X4#, Y4#) As Long

X1: Specifies the x-coordinate of the upper-left corner of the bounding rectangle.
Y1: Specifies the y-coordinate of the upper-left corner of the bounding rectangle.
X2: Specifies the x-coordinate of the lower-right corner of the bounding rectangle.
Y2: Specifies the y-coordinate of the lower-right corner of the bounding rectangle.
X3: Specifies the x-coordinate of the point that defines the chord's starting point.
Y3: Specifies the y-coordinate of the point that defines the chord's starting point.
X4: Specifies the x-coordinate of the point that defines the chord's endpoint.
Y4 Specifies the y-coordinate of the point that defines the chord's endpoint.

Draws a chord (a closed figure bounded by the intersection of an ellipse and a line segment). The (x1, y1) and (x2, y2) parameters specify the upper-left and lower-right corners, respectively, of a rectangle bounding the ellipse that is part of the chord. The (x3, y3) and (x4, y4) parameters specify the endpoints of a line that intersects the ellipse. The chord is drawn by using the current border, fill and background attributes. The figure drawn by the Chord function extends up to, but does not include the right and bottom coordinates. This means that the height of the figure is y2 - y1 and the width of the figure is x2 - x1.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle for the chord.
Scaling

DrawText

CreateDrawText (PictureHandle&, Text$, Left%, Top%, Width%, Height%, Fmt%) As Long
SCreateDrawText (PictureHandle&, Text$, Left#, Top#, Width#, Height#, Fmt%) As Long

Text: the string to be drawn.
Top, Left, Width, Height: the rectangle in which the text is to be formatted.
nFormat: the method of formatting the text.

Use OR to combine the following values to create the format:

DT_BOTTOM: bottom-justified text. This value must be combined with DT_SINGLELINE.
DT_CALCRECT: determines the width and height of the rectangle. If there are multiple lines of text, DrawText will use the width of the rectangle specified and extend the base of the rectangle to bound the last line of text. If there is only one line of text, DrawText will modify the right side of the rectangle so that it bounds the last character in the line. In either case, DrawText returns the height of the formatted text but does not draw the text.
DT_CENTER: centers text horizontally.
DT_EXPANDTABS: Expands tab characters. The default number of characters per tab is eight.
DT_EXTERNALLEADING: Includes the font's external leading in the line height. Normally, external leading is not included in the height of a line of text.
DT_LEFT: Aligns text flush-left.
DT_NOCLIP: Draws without clipping. DrawText is somewhat faster when DT_NOCLIP is used.
DT_NOPREFIX: Turns off processing of prefix characters. Normally, DrawText interprets the ampersand (&) mnemonic-prefix character as a directive to underscore the character that follows, and the two-ampersand (&&) mnemonic-prefix characters as a directive to print a single ampersand. By specifiying DT_NOPREFIX this processing is turned off.
DT_RIGHT: Aligns text flush-right.
DT_SINGLELINE: Specifies single line only. Carriage returns and linefeeds do not break the line.
DT_TABSTOP: Sets tab stops. The high-order byte of nFormat is the number of characters for each tab. The default number of characters per tab is eight.
DT_TOP: Specifies top-justified text (single line only).
DT_VCENTER: Specifies vertically centered text (single line only).
DT_WORDBREAK: Specifies word-breaking. Lines are automatically broken between words if a word would extend past the edge of the rectangle specified by lpRect. A carriage return-linefeed sequence will also break the line.

Note that the values DT_CALCRECT, DT_EXTERNALLEADING, DT_INTERNAL, DT_NOCLIP, and DT_NOPREFIX cannot be used with the DT_TABSTOP value.

Draws formatted text in the rectangle specified. It formats text by expanding tabs into appropriate spaces, aligning text to the left, right, or center of the given rectangle, and breaking text into lines that fit within the given rectangle. The type of formatting is specified by nFormat. This member function uses the device context's selected font, text color, and background color to draw the text. Unless the DT_NOCLIP format is used, DrawText clips the text so that the text does not appear outside the given rectangle. All formatting is assumed to have multiple lines unless the DT_SINGLELINE format is given. If the selected font is too large for the specified rectangle, the DrawText member function does not attempt to substitute a smaller font. If the DT_CALCRECT flag is specified, the rectangle specified will be updated to reflect the width and height needed to draw the text.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively. You can access Width and Height using these functions by treating them as X2, Y2.

Returns: The handle of the DrawText object.

Scaling

Ellipse

CreateEllipse (PictureHandle&, X1%, Y1%, X2%, Y2%) As Long
SCreateEllipse (PictureHandle&, X1#, Y1#, X2#, Y2#) As Long

X1: x-coordinate of the upper-left corner of the ellipse's bounding rectangle.
Y1: y-coordinate of the upper-left corner of the ellipse's bounding rectangle.
X2: x-coordinate of the lower-right corner of the ellipse's bounding rectangle.
Y2: y-coordinate of the lower-right corner of the ellipse's bounding rectangle.

Draws an ellipse. The center of the ellipse is the center of the bounding rectangle specified by x1, y1, x2, and y2. The ellipse is drawn with the current border, fill and background attributes. The figure drawn by this function extends up to but does not include the right and bottom coordinates. This means that the height of the figure is y2 - y1 and the width of the figure is x2 - x1. If either the width or the height of the bounding rectangle is 0, no ellipse is drawn.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle of the ellipse object.
Scaling

Line

CreateLine (PictureHandle&, X1%, Y1%, X2%, Y2%) As Long
SCreateLine (PictureHandle&, X1#, Y1#, X2#, Y2#) As Long

X1: x-coordinate of the startpoint for the line.
Y1: y-coordinate of the startpoint for the line.
X2: x-coordinate of the endpoint for the line.
Y2: y-coordinate of the endpoint for the line.

Draws a line from the startpoint up to, but not including, the endpoint. The line is drawn with the current border attributes.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle of the line object.
Scaling

Polygon (Empty)

CreatePolygon (PictureHandle&) As Long
SCreatePolygon (PictureHandle&) As Long

Creates an empty polygon.

Points may be added or deleted using AddPoint (SAddPoint), InsertPointAt (SInsertPointAt), RemovePointAt, RemoveAllPoints. Points may be changed by using SetX (SSetX) and SetY (SSetY).

Return Value: The handle of the polygon object.

Scaling
Setting & Getting Points
Other Editing

Polygon (VB Array)

BCreatePolygon (PictureHandle&, ThePoints() As PointAPI) As Long
SBCreatePolygon (PictureHandle&, ThePoints() As ScalePointApi) As Long

ThePoints: vertices of the polygon as a VB dynamic array.

Draws a polygon consisting of two or more points (vertices) connected by lines, using the current border, fill and background attributes. The system closes the polygon automatically, if necessary, by drawing a line from the last vertex to the first.

ThePoints is created in VB by

```
Dim ThePoints() as PointApi  ' or ScalePointApi
```

and later

```
Redim ThePoints(MaxIndex)
```

where MaxIndex is the number of points minus 1.

Changing the data in the VB array will change the appearance of the polygon the next time it is drawn. Note that you must call DoScale against the polygon, or a containing picture, after changing the data in order to reflect the changes.

Return Value: The handle of the polygon object.

Scaling

Pie

CreatePie (PictureHandle&, X1%, Y1%, X2%, Y2%, X3%, Y3%, X4%, Y4%) As Long
SCreatePie (PictureHandle&, X1#, Y1#, X2#, Y2#, X3#, Y3#, X4#, Y4#) As Long

X1: x-coordinate of the upper-left corner of the bounding rectangle.
Y1: y-coordinate of the upper-left corner of the bounding rectangle.
X2: x-coordinate of the lower-right corner of the bounding rectangle.
Y2: y-coordinate of the lower-right corner of the bounding rectangle.
X3: x-coordinate of the arc's starting point.
Y3: y-coordinate of the arc's starting point.
X4: x-coordinate of the arc's endpoint.
Y4: y-coordinate of the arc's endpoint.

Points 3 and 4 do not have to lie on the arc.

Draws a pie-shaped wedge by drawing an elliptical arc whose center and two endpoints are joined by lines. The center of the arc is the center of the bounding rectangle specified by x1, y1, x2, and y2 . The starting and ending points of the arc are specified by x3, y3, x4, and y4). The arc is drawn with the current border attributes, moving in a counterclockwise direction. Two additional lines are drawn from each endpoint to the arc's center. The pie-shaped area is filled with the fill and background attributes. If x3 equals x4 and y3 equals y4, the result is an ellipse with a single line from the center of the ellipse to the point (x3, y3) or (x4, y4).

The figure drawn by this function extends up to but does not include the right and bottom coordinates. This means that the height of the figure is y2 - y1 and the width of the figure is x2 - x1.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle of the Pie.
Scaling

PolyLine (Empty)

CreatePolyLine (PictureHandle&) As Long
SCreatePolyline (PictureHandle&) As Long

Creates an empty polygon.

Points may be added or deleted using AddPoint (SAddPoint), InsertPointAt (SInsertPointAt), RemovePointAt, RemoveAllPoints. Points may be changed by using SetX (SSetX) and SetY(SSetY).

Return Value: The handle of the polygon object.

Scaling
Setting & Getting Points
Other Editing

PolyLine (VB Array)

BCreatePolyLine (PictureHandle&, ThePoints() As PointAPI) As Long
SBCreatePolyline (PictureHandle&, ThePoints() As ScalePointApi) As Long

ThePoints: A Redim-ed VB array of points to be connected.

Draws a set of line segments connecting the points specified by ThePoints. The lines are drawn from the first point through subsequent points using the current border attributes.

ThePoints is created in VB by
    Dim ThePoints() as PointApi ' or ScalePointApi
and later
    Redim ThePoints(MaxIndex)
where MaxIndex is the number of points minus 1.

Changing the data in the VB array will change the appearance of the polyline the next time it is drawn. Note that you must call DoScale against the polyline, or a containing picture, after changing the data in order to reflect the changes.

Return Value: The handle of the polyline object.

Scaling

PolyPolygon (Empty)

CreatePolyPolygon (PictureHandle&) As Long
SCreatePolyPolygon (PictureHandle&) As Long

Creates an empty polypolygon. This will consist of a collection of points and a collection of vertex-counts (1 count for each polygon).

Points may be added or deleted using AddPoint (SAddPoint), InsertPointAt (SInsertPointAt), RemovePointAt, RemoveAllPoints. Points may be changed by using SetX (SSetX) and SetY.

Counts may be added or deleted using AddPolyCount, SAddPolyCount, InsertPolyCountAt, RemovePolyCountAt, RemoveAllPolyCounts. Points may be changed by using SetPolyCount.

There are symmetric "get" functions for the "set:" functions referred to above.

Return Value: The handle of the polypolygon object.

Scaling
Setting &Getting Points
Other Editing

PolyPolygon (VB Arrays)

BCreatePolyPolygon (PictureHandle&, ThePoints() As PointAPI, Polycounts%()) As Long
SBCreatePolyPolygon (PictureHandle&, ThePoints() As ScalePointApi, Polycounts%()) As Long

ThePoints: A Redim-ed VB array of points that define the vertices of the polygons.
PolyCounts: A Redim-ed VB array of integers, each of which specifies the number of points in one of the polygons in the ThePoints array.

Creates two or more polygons using the current border, fill and background attributes. The polygons may be disjoint or overlapping. Each polygon specified in a call to the PolyPolygon function must be closed in order to be filled. Unlike polygons created by the Polygon functions, the polygons created by PolyPolygon are not closed automatically.

ThePoints is created in VB by

```
Dim ThePoints() as PointApi  ' or ScalePointApi
```

and later

```
Redim ThePoints(MaxIndex)
```

where MaxIndex is the number of points minus 1.

PolyCounts is set up similarly (but as integer).

Changing the data in the VB arrays will change the appearance of the polyline the next time it is drawn. Note that you must call DoScale against the polyline, or a containing picture, after changing the data in order to reflect the changes.

Return Value: The handle of the polypolygon object.

Scaling

Rectangle

CreateRectangle (PictureHandle&, X1%, Y1%, X2%, Y2%) As Long
SCreateRectangle (PictureHandle&, X1#, Y1#, X2#, Y2#) As Long

X1: x-coordinate of the upper-left corner of the rectangle.
Y1: y-coordinate of the upper-left corner of the rectangle.
X2: x-coordinate of the lower-right corner of the rectangle.
Y2: y-coordinate of the lower-right corner of the rectangle.

Draws a rectangle using the current border, fill and background attributes. The rectangle extends up to, but does not include, the right and bottom coordinates. This means that the height of the rectangle is y2 - y1 and the width of the rectangle is x2 - x1.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle of the rectangle object.
Scaling

RoundRect

CreateRoundRect (PictureHandle&, X1%, Y1%, X2%, Y2%, X3%, Y3%) As Long
SCreateRoundRect (PictureHandle&, X1#, Y1#, X2#, Y2#, X3#, Y3#) As Long

X1: x-coordinate of the upper-left corner of the rectangle.
Y1: y-coordinate of the upper-left corner of the rectangle.
X2: x-coordinate of the lower-right corner of the rectangle.
Y2: y-coordinate of the lower-right corner of the rectangle.
X3: width of the ellipse used to draw the rounded corners.
Y3: height of the ellipse used to draw the rounded corners.

Draws a rectangle with rounded corners using the current border, fill and background attributes. The figure this function draws extends up to but does not include the right and bottom coordinates. This means that the height of the figure is y2 - y1 and the width of the figure is x2 - x1.

You can later set or get the coordinates using SetX/Y-GetX/Y or SSetX/Y-SGetX/Y respectively.

Return Value: The handle of the RoundRect object.
Scaling

TextOut

CreateTextOut (PictureHandle&, X1%, Y1%, NewText$) As Long
SCreateTextOut (PictureHandle&, X1#, Y1#, NewText$) As Long

X1: x-coordinate of the starting point of the text.
Y1: y-coordinate of the starting point of the text.
NewText: The character string to be drawn.

Writes a character string at the specified location using the current font. Character origins are at the upper-left corner of the character cell.

Return Value: The handle of the text object.
Scaling

TabbedTextOut

BCreateTabbedTextOut (PictureHandle&, X%, Y%, NewText$, TabStopPositions%(), TabOrigin%) As Long
SBCreateTabbedTextOut (PictureHandle&, X#, Y#, NewText$, , TabStopPositions#(), TabOrigin#) As Long

X: x-coordinate of the starting point of the string.
Y: y-coordinate of the starting point of the string.
NewText: the character string to draw.
TabStopPositions: Redim-ed VB array containing the tab-stop positions.
nTabOrigin: x-coordinate of the starting position from which tabs are expanded.

Writes a character string at the specified location, expanding tabs to the values specified in the array of tab-stop positions. Text is written in the current font.

The tab stops must be sorted in increasing order; the smallest x-value should be the first item in the array

Return Value: The handle of the TabbedText object.
Scaling

PolyTextOut (Empty)

CreatePolyTextOut (PictureHandle&) As Long
SCreatePolyTextOut (PictureHandle&) As Long

Creates an empty PolyTextOut object.

Items may be added or deleted using AddPText, SAddPText, InsertPTextAt, SInsertPTextAt, RemovePTextAt, RemoveAllPText. Points may be changed by using SetX/Y or SSetX/Y.

Return Value: the handle of the polytext object.

Scaling
Setting and Getting Points
Other Editing

PolyTextOut (VB Arrays)

BCreatePolyTextOut (PictureHandle&, ThePoints() As PointAPI, OutText$()) As Long
SBCreatePolyTextOut (PictureHandle&, ThePoints() As ScalePointApi, OutText$()) As Long

ThePoints: a redim-ed VB array of points
OutText: a redim-ed VB array of strings

Writes the strings at the points specified using the current font attributes.

Return Value: The handle of the polytext object.
Scaling

Shapes

Arc
Chord
Ellipse
Line
Pie
Polygon
Polygon (VB array)
Polyline
Polyline (VB array)
PolyPolygon
PolyPolygon (VB Arrays)
Rectangle
RoundRect

BorderWidth

SetBorderWidth GraphicHandle&, PenWidth%
GetBorderWidth (GraphicHandle&) as Integer

Sets (gets) the width for the border of shapes. In the case of a line/polyline, this is the thickness of the line. If positive, PenWidth is the width in pixels. If it is negative, then it is scaled according to the scale of the immediately containing picture.

You can set the border width of a picture, as opposed to a shape.

If you set this attribute, then the specified border width will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the border width of the drawing device will be set back to the value it had before drawing that object.

Drawing Sequence

Consider the code:

```
Picture2 = CreatePicture(Picture1)
Line1 = CreateLine(Picture1, 0, 0, 10, 10)
Line2 = CreateLine(Picture2, 0, 0, 10, 10)
```

The picture with handle Picture2 is created by and logically contained in Picture1 (although its physical placement is not necessarily within that of Picture1).

DoPaint commences drawing with the first global picture created (which is the default global picture automatically created for each instance of the VBX). DoDraw commences from the picture specified. Within a picture, objects are drawn in the order they were created, subject to the fact that if an object is in turn a picture, then all the objects contained in that picture are drawn before proceeding to the next object..

In the above example, the objects are drawn in the following sequence:

1. Picture1
2. Picture2
3. Line2
4. Line1

FastGraph v 0.9 (Beta)
(C) Decision Management Software CC 1994

Contents

Drawing Attributes

Drawing attributes consist of border, background, fill and mode attributes. They apply to all shapes.

Border

- BorderColor
- BorderStyle
- BorderWidth

Background

- BackColor
- BackStyle

Fill

- FillColor
- FillStyle

Mode

- DrawMode

BorderColor

SetBorderColor GraphicHandle&, crColor&
GetBorderColor (GraphicHandle&) as Long

Sets (gets) the border color. This is an RGB color such as returned by the VB functions RGB and QBColor.

You can set the border color of a picture, as opposed to a shape.

If you set this attribute, then the specified border color will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the border color of the drawing device will be set back to the value it had before drawing that object.

BorderStyle

Sub SetBorderStyle GraphicHandle&, nPenStyle%
GetBorderStyle (GraphicHandle&) as Integer

Sets (gets) the border style for an object. nPenstyle may have the following values:

0 - Solid
1 - Dash
2 - Dot
3 - Dash Dot
4 - Dash Dot Dot
5 - Transparent
6 - Inside Solid

You can set the border style of a picture, as opposed to a shape.

If you set this attribute, then the specified border style will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the border style of the drawing device will be set back to the value it had before drawing that object. If the border width is non-zero, the style is always drawn as Solid.

BackColor

SetBackColor GraphicHandle&, crColor&
GetBackColor (GraphicHandle&) as Long

Sets (gets) the background color. This is an RGB color such as returned by the VB functions RGB and QBColor.

You can set the background color of a picture, as opposed to a shape.

If you set this attribute, then the specified background color will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the background color of the drawing device will be set back to the value it had before drawing that object.

BackStyle

SetBackStyle GraphicHandle&, BackStyle%
GetBackStyle (GraphicHandle&) as Integer

Sets (gets) the background style for an object. BackStyle may have the following values:

1 - Transparent
2 - Opaque

If you set this attribute, then the specified background style will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the background style of the drawing device will be set back to the value it had before drawing that object.

FillColor

SetFillColor GraphicHandle&, crColor&
GetFillColor (GraphicHandle&) as Long

Sets (gets) the fill color. This is an RGB color such as returned by the VB functions RGB and QBColor.

You can set the fill color of a picture, as opposed to a shape.

If you set this attribute, then the specified fill color will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the fill color of the drawing device will be set back to the value it had before drawing that object.

FillStyle

SetFillStyle GraphicHandle&, FillStyle%
SetFillStyle (GraphicHandle&, FillStyle%) as Integer

Sets (gets) the fill style for an object. FillStyle may have the following values:

0 - Solid
1 - Hollow
2 - Horizontal Lines
3 - Vertical Lines
4 - Upward Diagonal
5 - Downward Diagonal
6 - Cross
7 - Diagonal Cross

If you set this attribute, then the specified fill style will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If the FillStyle is Hollow and the BackStyle is Opaque, objects are filled with the BackColor.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the fill style of the drawing device will be set back to the value it had before drawing that object.

DrawMode

SetDrawMode GraphicHandle&, DrawMode%
GetDrawMode (GraphicHandle&) as Integer

Sets (gets) the draw mode for an object. DrawMode may have the following values:

[Pen means the current border or fill color, depending on whether the border is being drawn or the shape is being filled]

1 Black Pen.
2 Inverse of setting 15 (Merge Pen) - NotMergePen.
3 Combination of the colors common to the background color and the inverse of the pen - MaskNot Pen.
4 Inverse of setting 13 (Copy Pen) - NotCopyPen.
5 Combination of the colors common to both the pen and the inverse of the display - MaskPenNot.
6 Inverse of the display - colorInvert.
7 Combination of the colors in the pen and in the display color, but not in both - XorPen.
8 Inverse of setting 9 (Mask Pen) - NotMaskPen.
9 Combination of the colors common to both the pen and the display - MaskPen.
10 Inverse of setting 7 (Xor Pen) - NotXorPen.
11 No operation - output remains unchanged. In effect, this setting turns drawing off - Nop.
12 Combination of the display color and the inverse of the pen color - MergeNotPen.
13 (Default) Color specified by the Border/Fill color - CopyPen.
14 Combination of the pen color and the inverse of the display color - MergePenNot.
15 Combination of the pen color and the display color - MergePen.
16 White Pen.

If you set this attribute, then the specified draw mode will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the draw mode of the drawing device will be set back to the value it had before drawing that object.

Text Objects

DrawText
PolyTextOut
PolyTextOut (VB Arrays)
TabbedTextOut (VB Arrays)
TextOut

Scaling Considerations

SCreateXXX, SSetXXX, SAddXXX and SInsertAtXXX functions take coordinates/dimensions in the scale of the picture in which the object is created; you must call DoScale against the handle of the object created, or a parent/ancestor picture handle, before the object is drawn by DoDraw or DoPaint. CreateXXX functions create pixel-based objects with coordinates given in pixels relative to the top-left of the device in which the object is drawn, and dimensions in pixels.

FontHeight

SetFontScaleHeight GraphicHandle&, dHeight#
GetFontScaleHeight (GraphicHandle&) as Double

SetFontPixelHeight GraphicHandle&, Height%
GetFontPixelHeight (GraphicHandle&) as Integer

SetFontPointHeight GraphicHandle&, Height%
GetFontPointHeight (GraphicHandle&) as Integer

Sets (gets) the height of the font in the scale of a picture, in pixels, or in points .

GraphicHandle is the handle of a picture or text object.

If ScaleHeight is used, the height of the font is expressed in the scale of the picture, if GraphicHandle is the handle of a picture, or the scale of the containing picture if it is not.

If the height is positive, then it specifies the height of the font including leading; if it is negative then it sets the actual height of the characters, excluding leading.

If you set this attribute, then the specified font height will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the font height of the drawing device will be set back to the value it had before drawing that object.

FontEscapement (Rotation)

SetFontEscapement GraphicHandle&, nEscapement%
GetFontEscapement (GraphicHandle&) as Integer

Sets (gets) the anti-clockwise rotation of the text from horizontal, in tenths of a degree.

GraphicHandle is the handle of a picture or text object.

If you set this attribute, then the specified escapement will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the font escapement of the drawing device will be set back to the value it had before drawing that object.

FontWeight

SetFontWeight GraphicHandle&, nWeight%
GetFontWeight (GraphicHandle&) as Integer

Sets (gets) the weigh of the font.

GraphicHandle is the handle of a picture or text object.

nWeight has one of the following values:

FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800
FW_BLACK	900
FW_HEAVY	900

The actual appearance of the font depends on the type face. Some fonts have only FW_NORMAL, FW_REGULAR, and FW_BOLD weights

If you set this attribute, then the specified font weight will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the font weight of the drawing device will be set back to the value it had before drawing that object.

FontItalic

SetFontItalic GraphicHandle&, nItalic%
GetFontItalic (GraphicHandle&)

Sets (gets) the italic attribute.

GraphicHandle is the handle of a text or picture object. nItalic should be set to True for italicized text, False otherwise.

If you set this attribute, then the specified italic attribute will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the italic attribute of the drawing device will be set back to the value it had before drawing that object.

FontUnderline

SetFontUnderline GraphicHandle&, nUnderline%
GetFontUnderline (GraphicHandle&) as Integer

Sets (gets) the underline attribute.

GraphicHandle is the handle of a text or picture object. nUnderline should be set to True for underlined text, False otherwise.

If you set this attribute, then the specified underline attribute will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the underline attribute of the drawing device will be set back to the value it had before drawing that object.

FontStrikeOut

SetFontStrikeOut GraphicHandle&, nStrikeOut%
SetFontStrikeOut (GraphicHandle&) as Integer

Sets (gets) the strikeout attribute.

GraphicHandle is the handle of a text or picture object. nStrikeOut should be set to True for struck-out text, False otherwise.

If you set this attribute, then the specified strikeout attribute will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, the strikeout attribute of the drawing device will be set back to the value it had before drawing that object.

FontFaceName

SetFontFaceName GraphicHandle&, FaceName$
GetFontFaceName (GraphicHandle&) as String

Sets (gets) the font's face name (corresponds to VB's FontName).

GraphicHandle is the handle of a text or picture object. FaceName is set to a value such as "Arial"..

RetStr must be long enough to contain the returned font name. The returned value is null-terminated, so WHY NOT JUST INCLUDE A COVER FUNCTION THAT DOES

```
Left(RetStr, Instr(RetStr, Chr(0)) - 1)
```

If you set FontFaceName for an object, then it will be set into the drawing device when that object is drawn.

If you do not explicitly set this attribute, then the text is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a text object or picture object, then after drawing that object, this attribute of the drawing device will be set back to the value it had before drawing that object.

FontColor

SetFontColor (GraphicHandle&, crColor&)
GetFontColor (GraphicHandle&) as Long

Sets (gets) the font color. crColor is an RGB color such as returned by the VB functions RGB and QBColor.

GraphicHandle is the handle of a text object or a picture.

If you set this attribute, then the specified font color will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the shape is drawn using the last value set in the drawing sequence.

If you set the PreserveAttributes property to True (see SetPreserveAttributes) for a shape object, text object or picture object, then after drawing that object, the font color of the drawing device will be set back to the value it had before drawing that object.

Font Attributes

FontColor
FontEscapement
FontFaceName
FontHeight
FontItalic
FontStrikeOut
FontUnderline
FontWeight

FastGraph v 0.9

What is FastGraph?

FastGraph is a Visual Basic custom control (VBX) that gives the VB developer much of the power of the Windows GDI API, while maintaining the simplicity of VB metaphors.

Its main advantages over calling the GDI directly are:

FastGraph uses an object-based metaphor with properties for graphic objects (such as FontItalic, Visible, FillStyle, BorderWidth) that are identical in name and semantics to those used in VB controls where possible. Where this is not posible, Fastgraph extends the familiar VB properties a natural way such as FontEscapement (rotation).

FastGraph provides support for scaling and positioning pictures using the Top, Left, Width, Height, ScaleTop, ScaleLeft, ScaleWidth, ScaleHeight attributes familiar to VB developers. A FastGraph picture can contain other FastGraph pictures, each with different coordinate and placement systems.

FastGraph pictures can be drawn to the screen or printer, create a metafile or copy themselves to the clipboard - each with a single, simple function call.

FastGraph is VB aware, so you can create objects such as PolyText, PolyLine, and PolyPolygon using native VB dynamic arrays for maximum speed and convenience.
In comparison with VB's Print, Line and Ellipse, FastGraph creates persistent objects that can be drawn to whatever target is required. Compute-intensive operations such as scaling are performed for you by highly optimized C code.

Set X, Y Properties

SetX GraphicHandle&, Idx%, X%
GetX (GraphicHandle&, Idx%) as Integer
SetY GraphicHandle&, Idx%, X%
GetY (GraphicHandle&, Idx%) as Integer
SSetX GraphicHandle&, Idx%, X#
SGetX (GraphicHandle&, Idx%) as Double
SSetY GraphicHandle&, Idx%, X#
SGetY (GraphicHandle&, Idx%) as Double

Sets (gets) points in Polygon, PolyLine, PolyTextOut etc. Idx specifies the index of the point required. Can also be used to get the coordinates for non-poly objects - Idx set to 0 will return X1/ Y1, to 1 will return X2/Y2 and so on.

Converting Between Pixel and Scale

ToScaleXY PicHandle&, X#, Y#
ToPhysicalXY PicHandle&, X#, Y#
ToScaleWH PicHandle&, W#, H#
ToPhysicalWH PicHandle&, W#, H#

Convert a pixel position to the scaled coordinates of the picture, and vice-versa.

Similarly for width and height.

Text Width/Height

GetTextWidth (pictureHandle&, hDC%, Text$) As Integer
GetTextHeight (pictureHandle&, hDC%, Text$) As Integer
GetScaleTextWidth (pictureHandle&, hDC%, Text$) As Double
GetScaleTextHeight (pictureHandle&, hDC%, Text$) As Double

hDC: handle to a device context
Text: the string to be measured

Returns the width/height of a string as it would be drawn in a particular device context in either pixels or the scale of the picture.

Be sure to set the required font attributes into the picture before calling this function.

Scale-Left, -Top, -Width, -Height

SetScale (pictureHandle&, WorldLeft#, WorldTop#, WorldWidth#, WorldHeight#

WorldLeft: x-coordinate that will map to Left in picture's placement.
WorldTop: y-coordinate that will map to Top in picture's placement.
WorldWidth: x-distance that will map to Width of picture's placement.
WorldHeight: y-distance that will map to Height of picture's placement.

Together with SetPlacement, this determines how scaled objects (those created using SCreate...) will be converted to pixels for drawing.

After calling this function, you must call DoScale against the the picture or an ancestor picture before calling DoPaint or DoDraw.

Left, Top, Width, and Height

SetPlacement pictureHandle&, Left#, Top#, Width#, Height#

Specifies the position, width and height corresponding to the scale values set by SetScale. Left, Top, Width and Height should be given in the scaling system of the containing picture if the picture being placed is not a global picture, or in pixels if it is a global picture.

Conversion & Measuring

Converting between pixel and scale coordinates/distances
Measuring text

Pictures

Scaling, Drawing, Printing Pictures

SetScale
SetPlacement
DoScale
DoPaint
DoDraw
IsScaleable

Realizing Scale Objects

DoScale GraphicHandle&

The pixel-based data required for actually drawing objects is not calculated until you call DoScale. DoScale creates the physical representation for all objects in a picture, recursing through any pictures contained in the one for which DoScale was called. It can also be called for a shape or text object, in which case just data for that object is computed.

After changing size/position attributes of an object, you must ensure that DoScale is called against that object, its parent picture, or an ancestor picture before calling DoPaint or DoDraw.

Painting to the Default Window

DoPaint (ByVal pictureHandle&)

Fastgraph controls have a property called DrawhWnd. By default this is the hWnd of the form on which the control is placed. Calling DoPaint causes the picture (and all its descendent pictures) to be drawn to the window specified by DrawhWnd. You can change the DrawhWnd property by assigning it the hWnd property of another VB control. If you wish to draw to an object that does not have an hWnd property, then you should use the DoDraw procedure.

DoPaint would normally be called in the Paint Event of the object in which the graphic is to be drawn

Drawing to an Arbitrary Device Context

DoDraw GraphicHandle&, TheHDC%

TheHDC: handle to a device context

Draws the specified shape, text or picture (and all its descendant pictures) to the specified device context.

This can be used to draw the picture on any VB object with an hDC property. Specifically, it is used to print a picture, as in this code segment:

```
Printer.ScaleMode = 3        ' Set to pixels so the next line does what's required
SetPlacement FG1 Printer.Left, Printer.Top, Printer.Width, Printer.Height
                             '...  Alternatively to whatever placement  is required
DoScale FG1          'Calculate pixel positions
Printer.Print ""             'NB:  Initialize printer
DoDraw FG1, Printer.hDC
Printer.EndDoc
SetPlacement FG1 Me.Left, Me.Top, Me.Width, Me.Height
                             'Assuming this is what it was originally
DoScale FG1          'Set pixel stuff back to what it was originally
```

Apart from printing, DoDraw would normally be called in the Paint Event of the object in which the graphic is to be drawn. It is also used during interactive drawing.

Creating Pictures

CreatePicture (pictureHandle&) As Long
CreateGlobalPicture () As Long

When an instance of the FastGraph control is created, a default global picture is automatically created, and the handle to this global picture is the default property of the instance of the control.

You can create pictures that are logically contained in the default picture by, for example:

SubPic1 = CreatePicture(FG1) 'FG1 is the Name of the FastGraph control

SubPic1 now contains the handle to a picture. You can add shapes and text to pictures by, for example:

Line1 = CreateLine(FG1, 0, 0, 100, 100) 'Creates line in global picture
Line2 = CreateLine(SubPic1, 0, 0, 100, 100) 'Creates line in sub-picture

Should you need additional global pictures, these can be created by calling CreateGlobalPicture.

Return Value: The handle of the picture created.

Scaling, Drawing, Printing

Deleting All Objects in a Picture

ClearAll pictureHandle&

Recursively removes all objects (including descendant pictures and their objects) from the specified picture. The specified picture itself is not deleted.

No drawing occurs.

Deleting a Graphic Object

RemoveObject TheHDC%, Mode%, RemoveObject%, Color&

TheHDC: handle to a DC
Mode: type of drawing required
RemoveObject: the object to be removed
Color: the color for drawing

If Mode is zero, no drawing occurs - you can pass any value for TheHDC.

If mode is 1, the object is drawn to TheHDC using the color specified. If most of the area is one colour (e.g. a line drawing is mostly the windows background colour) then it is visually effective to redraw the deleted object in that color, and subsequently redraw the global picture without clearing the window.

If mode is 2, the object is drawn with its current attributes. This visually erases the object if it was originally drawn with a DrawMode such as XOR.

RemoveObject is set to True or False - it specifies whether the object should be deleted from the picture, or just redrawn respectively. If True, recursively removes all objects (including descendant pictures and their objects) from the specified picture. The specified picture itself is ALSO deleted.

Color is the RGB color for drawing.

Editing PolyShapes

AddPoint GraphicHandle&, NewX%, NewY%
SAddPoint GraphicHandle&, NewX#, NewY#
GetNumPoints (GraphicHandle&) As Integer
RemoveAllPoints GraphicHandle&
InsertPointAt GraphicHandle&, nIndex%, NewX%, NewY%
RemovePointAt GraphicHandle&, nIndex%
SInsertPointAt GraphicHandle&, nIndex%, NewX#, NewY#
RemoveAllPolyCounts ByVal GraphicHandle&
SetPolyCount GraphicHandle&, nIndex%, nCount%
AddPolyCount GraphicHandle&, ByVal nNewCount%
InsertPolyCountAt GraphicHandle&, nIndex%, nNewCount%
GetNumPolyCounts (GraphicHandle&) As Integer
nIndex: Index into point or count array - 0 is the first element

These functions are used for editing polygons, polylines, and polypolygons which are NOT created with VB arrays. The PolyCount functions are for polypolygons, which have both a collection of points and a collection of PolyCounts.

SAddPoint and SInsertPointAt are for scaleable objects (created using SCreateXXX), while there counterparts without the "S" prefix are for pixel-based objects (CreateXXX).

Editing PolyText

AddPText PolyTextHandle&, nNewX%, nNewY%, pNewText$
SAddPText PolyTextHandle&, dNewX#, dNewY#, pNewText$
SetPTextAt PolyTextHandle&, nIdx%, pNewText$
GetPTextAt (PolyTextHandle&, nIdx%) as String
RemoveAllPText PolyTextHandle&
InsertPTextAt TextHandle&, nIdx%, X%, Y%, pNewText
SInsertPTextAt PolyTextHandle&, nIdx%, X#, Y#, pNewText$
RemovePTextAt ByVal PolyTextHandle&, ByVal nIdx%
nIdx: Index into polytext array - 0 is the first element

These functions are used for editing polytext objects which are NOT created with VB arrays.

SAddPText and SInsertPTextAt are for scaleable objects (created using SCreateXXX), while there counterparts without the "S" prefix are for pixel-based objects (CreateXXX).

Restoring Drawing & Font Attributes

SetPreserveAttribs GraphicHandle&, PreserveAttribs%

Use this procedure to set the PreserveAttribs property of a picture, shape or text objects.

When the object is drawn, any drawing or font attributes that you have explicitly set for that object are set into the drawing device.

If PreserveAttribs is true, then the status of the drawing device is saved before setting these attributes, and restored after drawing of the object completes.

FontWidth

SetFontScaleWidth GraphicHandle&, dHeight#
GetFontScaleWidth (GraphicHandle&) as Double

SetFontPixelWidth GraphicHandle&, Height%
GetFontPixelWidth (GraphicHandle&) as Integer

SetFontPointWidth GraphicHandle&, Height%
GetFontPointWidth (GraphicHandle&) as Integer

Sets (gets) the width of the font in the scale of a picture, in pixels, or in points .

GraphicHandle is the handle of a picture or text object.

If ScaleHeight is used, the width of the font is expressed in the scale of the picture, if GraphicHandle is the handle of a picture, or the scale of the containing picture if it is not.

If you set this attribute, then the specified font height will be set into the drawing device when the object is drawn.

If you do not explicitly set this attribute, then the system chooses an appropriate width for the height used.

Using FastGraph from VB

You install FastGraph simply by placing the distribution files on your hard disk in whatever directory is convenient. We recommend that you keep a single copy of the VBX file in your \Windows\System or equivalent directory.

To use FastGraph from VB you must use the File/Add File menu option in VB to add FG.VBX and FGDEF.BAS to your project.

Be sure to distribute FG.VBX with your application.

Technical Support

Decision Management Software CC

Tel: 27 11 802-8846
Fax: 27 11 802-8889

Compuserve: 70750,1776

Postal: **Box 998**
Kelvin
2054
South Africa

Is Object Scaleable?

IsScaleable (GraphicHandle&) as Integer

Returns True if the object was create by ScreateXXX or SBCreate XXX type functions, in which case it is scaled by DoScale according to the values set with SetScale and SetPlacement. Otherwise, returns False.