



Website: www.chestysoft.com

Email: info@chestysoft.com

csXImage - Version 1.2

OCX Control for Display and Manipulation of Images

Introduction

This is an OCX control that enables processing of graphic images. A range of methods is provided for enhancing and manipulating the image. The image can be displayed in the control, saved to file, copied to the Windows clipboard for pasting into another application, or printed. The image can also be exported as binary data in the form of a variant array, making it suitable for use in a server side application such as an ASP script, or for saving to a database.

Exported images can be in BMP, PNG, JPG, GIF, WBMP, PCX, PSD or TIFF format. PNG and GIF images support a transparent colour. GIF images are uncompressed to avoid using the patented LZW algorithm. This does result in larger file sizes than the standard compressed GIF.

A similar range of options is available for import of images into the control. The options are: reading from a file, import of binary data from a variant array, reading an image from a URL and pasting from the Windows clipboard. BMP, PNG, JPG, WBMP, PCX, PSD and TIFF formats are supported. Unfortunately, it is not possible to provide the capability to read GIF images, due to patent restrictions on the LZW compression algorithm. This restriction also applies to some TIFF files in which the LZW algorithm has been used for image compression. Reading of these files is also not supported.

A full range of events are provided for programmers to respond to mouse clicks and movement of the mouse over the control.

The OCX file, csXImage.ocx, must be registered on the computer running the application. This should be done during the installation, but if it is not the command line utility Regsvr32.exe can be used. This is usually found in the Windows system folder and runs using the syntax:

```
regsvr32 ocxname
```

where *ocxname* is the path and name of the ocx file to register. When deploying an application that uses csXImage, the ocx file will also need to be registered on the target machine. Note that the licence file, csXImage.lic, is needed to use this control in a design environment and this file must not be distributed with an application, as is confirmed in your licence agreement.

To use this control in a Visual Basic project, select Project and Components from the pull down menu. This gives a list of available controls. Select csXImage Library and click Apply. This adds the control to the component palette. The class name is "ImageBox" and this will appear in the Object Browser where the properties and methods are listed. For other design environments, consult the documentation for importing ActiveX controls.

The Programme ID for the control is "csXImage.ImageBox" and this is used inside the CreateObject command when creating an instance of the object from code. In the trial version this ID is "csXImageTrial.ImageBox".

As an ActiveX control, csXImage can be used in a range of Windows based environments and languages. There will be slight differences in syntax especially with object creation, the use of parentheses/brackets surrounding method parameters, and the use of constants for enumerated properties.

The Trial Version

The trial version of csXImage is supplied as a different OCX file, called csXImageTrial.ocx, and has a separate installation programme. Images processed using the trial version have a line of text added at the top, to indicate that an unregistered version is being used. Apart from this it is fully functional and has no missing properties or methods and no time limit. Visit the Chestysoft website for details of how to buy the full version. - www.chestysoft.com

Getting Started

In these instructions, the properties and methods have been grouped into a number of categories depending on their general function. These categories are:

- Import and Export of Images
- Colour Formats
- Image Enhancement
- Selecting a region of an image
- Drawing
- Printing
- Miscellaneous
- Events
- File Info (JPEG Meta Data)

For Visual Basic users, a demonstration project, *csXImageDemo.vbp*, is provided with the csXImage installation. This demonstration project shows example code for using some of the key methods and properties available in csXImage.

Finally there is a summary of the enumerated properties used by the control and a note on deploying applications.

Import and Export of Images

Images can be imported into the control and exported from the control either as files on disk, variant arrays, or using the Windows clipboard.

Import/Export Methods

LoadFromFile (*FileName* As String) - Reads the file from disk, loads the image into the control and displays the image. *FileName* must be a complete path to the file, including the file extension. The extension must be .bmp, .jpg, .jpeg, .png, .wbmp, .wbm, .pcx, .psd, .tif or .tiff. If it is .png, transparency information will be loaded into the *Transparent* and *BGColor* properties.

ReadBinary (*Format* As TxGraphicsFormat, *OLEInput*) - Reads an image as binary data from *OLEInput*, which is an OLE Variant array. The graphics format is defined by *Format*, which can take one of the following values:

gfBMP:	Bitmap format
gfGIF:	GIF format (only available for <i>WriteBinary</i>)
gfJPG:	JPG format
gfPCX:	PCX format
gfPNG:	PNG format
gfWBMP:	Wireless Bitmap format
gfPSD:	Adobe Photoshop format
gfTIF:	TIFF format

LoadFromURL (*URL* As String) - Reads a file from a URL. URL must be a complete HTTP reference to the file, including a valid file extension. It is not necessary to include 'http://' at the start

of the string. Note that on some operating systems used by web servers, e.g. UNIX, file names are case-sensitive, so incorrect use of upper/lower case may result in no image being loaded.

SaveToFile (*FileName* As String) - Saves the file to disk. *FileName* must be a complete path to the file, including the file extension. The extension must be .bmp, .gif, .jpg, .jpeg, .png, .wbmp, .wbm, .pcx, .psd, .tif or .tiff. If it is .gif or .png, transparency will be defined by the *Transparent* and *BGColor* properties. If it is .jpg or .jpeg, the compression quality will be defined by the *JPEGQuality* property. If the file already exists, it will be overwritten without warning.

If *UseSelection* is True, *SaveToFile* will save only the selected region of the image.

WriteBinary (*Format* As TxGraphicsFormat) As Variant - Writes an image as binary data to an OLE Variant array. The graphics format is defined by *Format*, which can take any of the values listed under *ReadBinary* above. This method is a function and must be called by assigning to a variable. For example, in VB, the syntax would be of the form:

```
VariantName = WriteBinary (gfBMP)
```

If *UseSelection* is True, *WriteBinary* will write only the selected region of the image to the OLE Variant array.

NewFileSize (*Format* As TxGraphicsFormat) As Integer - Returns the file size in bytes if the current image is saved to disk in the graphics format given in *Format*.

If *UseSelection* is True, *NewFileSize* will return the file size if only the selected region of the image is saved to disk.

Copy () - Copies the currently loaded image to the Windows clipboard, in Bitmap format.

If *UseSelection* is True, *Copy* will only copy the selected region of the image.

Paste () - Retrieves an image from the Windows clipboard, if one is available in an appropriate format. *Paste* replaces an image already loaded into the control, without warning.

Import/Export Properties

JPEGQuality As Integer - When an image is saved in JPG format (file extensions .jpg or .jpeg), the quality can be varied between a high quality image, which gives a large file, or a lower quality image, which gives a smaller file. This property can take a value from 1 to 100. 100 is the highest quality, 1 is very low quality. (Default = 90).

Transparent As Boolean - Indicates that an image includes transparency information. This is only used by images to be saved in GIF or PNG formats. (Default = False).

BGColor As OLE_COLOR - The background, or transparent colour to be used in images with transparency. This is used by images to be saved in GIF or PNG formats. This colour is also used to fill in the background of rotated images using the *Rotate* method. (Default = White, &H00FFFFFF).

BMPHandle As Long - Returns a Windows handle to a copy of the bitmap image. Setting this property copies the image referenced by the new handle value into the control. This property can be used, for example, to copy an image between two instances of the control, in which case, the syntax would be in the form:

```
ImageBox2.BMPHandle = ImageBox1.BMPHandle
```

PictureData As Variant - Read-only property. Provides the image data in a format that can be transferred to an Image object in MS Access. For example, the following code copies an image from a csXImage object called 'ImageBox1' to a MS Access Image object called 'Image1':

```
Image1.PictureData = ImageBox1.PictureData
```

Colour Formats

When an image is loaded into the control, the colour format is automatically detected and properties of the control are set accordingly.

ColorFormat As *TxColorFormat* - The *ColorFormat* property defines the depth of colour in the image. The value of *ColorFormat* can be changed in order to convert the image to a different format.

ColorFormat can take any of the following values:

cf24bit - 24-bit colour, in which each pixel is represented by 3 bytes, one for the intensity of red, one for green and one for blue.

cf256Color - 256 colour (8-bit) palettised image. A table (palette) is stored which lists the red, green and blue intensities for each available colour. Each pixel is represented by 1 byte to indicate the colour entry in the palette.

cf256Grayscale - This image is stored in the same way as a 256 colour image, but every colour in the palette is a shade of grey, i.e., red, green and blue values are equal.

cf16Color - 16 colour (4-bit) palettised image. A table (palette) is stored which lists the red, green and blue intensities for each available colour. Each pixel is represented by 4 bits to indicate the colour entry in the palette.

cf16Grayscale - This image is stored in the same way as a 16 colour image, but every colour in the palette is a shade of grey, i.e., red, green and blue values are equal.

cfMono - Monochrome (1-bit) palettised image. A table (palette) is stored which lists the red, green and blue intensities for the two available colours. Each pixel is represented by 1 bit to indicate the colour entry in the palette. Note that the two colours can be any colours, it is not necessary to use black and white. However, some commonly used image processing applications always interpret monochrome images as being black and white, and may not display the colours of a *cfMono* image created by *csXImage* correctly.

cfMonoBW - Monochrome (1-bit) palettised image (see above) in which the two colours are pure black and pure white.

cfNone - This is a special value for the *ColorFormat* property which indicates that no image is loaded into the control. It is not possible for the user to set *ColorFormat* equal to *cfNone*.

Image Enhancement

Image Enhancement Methods

Brightness (*Value* As Long, *DoRed* As Boolean, *DoGreen* As Boolean, *DoBlue* As Boolean)
- Adjusts the brightness of the loaded image. *Value* must be an integer between 0 (minimum) and 100 (maximum).

0 will reduce brightness to the minimum extent.
100 will increase brightness to the maximum extent.
50 gives no change.

DoRed, *DoGreen*, *DoBlue*, are Boolean values indicating which of the three colours are adjusted. For normal brightness adjustment, set all three parameters to "True".

If the image is in a grayscale format, i.e., it has *ColorFormat* of *cfMonoBW*, *cf16Grayscale* or *cf256Grayscale*, the values entered for *DoRed*, *DoGreen* and *DoBlue* are ignored. In this case, these parameters will default to True.

If *UseSelection* is True and the *ColorFormat* is cf24bit, the brightness is only adjusted in the selected region of the image.

Contrast (*Value* As Long, *DoRed* As Boolean, *DoGreen* As Boolean, *DoBlue* As Boolean) - Adjusts the contrast of the loaded image. *Value* must be an integer between 0 (minimum) and 100 (maximum).

0 will reduce contrast to the minimum extent.
100 will increase contrast to the maximum extent.
50 gives no change.

DoRed, *DoGreen*, *DoBlue*, are Boolean values indicating which of the three colours are adjusted. For normal contrast adjustment, set all three parameters to "True".

If the image is in a grayscale format, i.e., it has *ColorFormat* of *cfMonoBW*, *cf16Grayscale* or *cf256Grayscale*, the values entered for *DoRed*, *DoGreen* and *DoBlue* are ignored. In this case, these parameters will default to True.

If *UseSelection* is True and the *ColorFormat* is cf24bit, the contrast is only adjusted in the selected region of the image.

Flip (*FlipMode* As TxFlipMode) - Flips the loaded image in either the horizontal or vertical plane (produces mirror image).

FlipMode can be either:

<i>fmHoriz</i> :	Flips in horizontal plane
<i>fmVert</i> :	Flips in vertical plane

Crop (*X1* As Long, *Y1* As Long, *X2* As Long, *Y2* As Long) - Crops the image to a rectangle bounded by the points (X1, Y1) and (X2, Y2).

Rotate (*Angle* As Single) - Rotates the image counter-clockwise. The amount of rotation in degrees is specified by *Angle*. If the image has *ColorFormat* of cf24bit, or the *Resample* property is set to True, an anti-aliasing technique is used to improve the quality of the rotated image. In that case, the resulting image always has a *ColorFormat* of cf24bit, whatever the *ColorFormat* of the original image. When an image is rotated by an angle that is not a multiple of 90 degrees, there will be an area around the image that must be filled with colour. The colour specified in the *BGColor* property is used.

ResizeImage (*NewWidth* As Long, *NewHeight* As Long) - Resizes the loaded image to the sizes given in *NewWidth* and *NewHeight*. The sizes are in pixels. If either *NewWidth* or *NewHeight* are set to zero, this value will be ignored, and the image will be resized using the other given measurement, and maintaining the current aspect ratio.

ScaleImage (*ScalePercent* As Single) - Resizes the loaded image, keeping the same aspect ratio. The image is scaled by a percentage value, *ScalePercent*. For example, if *ScalePercent* = 50, the image is reduced to half size.

Sharpen () - Sharpens the loaded image. This method requires no parameters.

Sharpen is exactly equivalent to the command *SharpenBy* (5).

SharpenBy (*Value* As Long) - Sharpens the loaded image. *Value* is an integer between 1 and 10 to specify the amount of sharpening. 1 gives a small sharpening effect, 10 gives a large effect.

Blur () - Blurs the loaded image. This method requires no parameters.

Blur is exactly equivalent to the command *BlurBy* (5).

BlurBy (*Value As Long*) - Blurs the loaded image. Value is an integer between 1 and 10 to specify the amount of blurring. 1 gives a small blurring effect, 10 gives a large effect.

If *UseSelection* is True, *Sharpen*, *SharpenBy*, *Blur* and *BlurBy* only affect the selected region of the image.

ReduceRedEye () - Reduces the red intensity for pixels in which the red intensity is significantly higher than the blue and green intensity. This method is intended for use on images where flash photography has resulted in a “red-eye” effect on the subject. It should be used in conjunction with selection of a region of the image, usually by using the *SelectEllipse* method to select the affected area of the eye. This method only operates on 24-bit images.

Merging Images and Watermarking

The following methods and associated properties are used for merging two images together. A wide range of options is provided allowing many different effects. The files can be simply merged with one smaller image copied over the top of a larger image, or a watermark effect can be obtained by using transparency in one of the images.

MergeFile (*MergeImageFile As String*) - Merges a second image (the watermark) with the current loaded image. The two images can have different sizes and different colour formats. The resulting image retains the size and colour format of the current image, not the watermark.

MergeImageFile is a String that must contain a complete path to the file containing the watermark image.

MergeBinary (*Format As TxGraphicsFormat*, *MergeImageBin As OLEVariant*) - This is identical to the *MergeFile* method, except that the second image is read from a variant array instead of from a file on disk.

MergeImageBin is the name of the variant array. *Format* defines the graphics format used in the variant array.

The following properties can be set to modify the behaviour of the WatermarkFile and WatermarkBinary methods:

MergeTransparent As Boolean - If *MergeTransparent* is True, one colour in the watermark image will be ignored in the merge process, i.e., it will be treated as if it is transparent. This can typically be used where the watermark file contains text or an image on a white background, and the background is to be treated as transparent (Default = False).

MergeTransparentColor As OLE_COLOR - Specifies the colour that is to be treated as transparent if *MergeTransparent* is True. (Default = White, &H00FFFFFF).

MergeTransparency As Double - A value between 0 and 100 indicating the percent overall transparency of the watermark image. A value of 100 means that the watermark is 100% transparent and the main image will be unchanged. A value of 0 means that the watermark is opaque and will completely overstamp the main image. (Default = 0).

MergeStyle As TxWMStyle - *MergeStyle* provides several options for the way that the watermark will be applied. The options are:

wmSingle: (Default). A single copy of the watermark image is applied, positioned at *MergeLeft*, *MergeTop*.

wmCentre: A single copy of the watermark image is applied, in the centre of the main image.

wmTile: The watermark is applied at the top left of the main image and repeated across the whole of the main image.

wmWrap: The watermark is applied first at the top left of the main image, then repeated to cover the whole image. Where the watermark is incomplete at the right hand side of the image, it is “wrapped”, i.e., starts on the next line at the point it had finished on the previous line.

MergeLeft As Long - The distance in pixels between the left side of the main image and the left side of the watermark image. Used when *MergeStyle* is *wmSingle* only. (Default = 0).

MergeTop As Long - The distance in pixels between the top of the main image and the top of the watermark image. Used when *MergeStyle* is *wmSingle* only. (Default = 0).

MergeReverse As Boolean - If set to True, the images are reversed, i.e., the current image is used as the watermark, and the second image becomes the main image. The size and colour format is then taken from the second image. (Default = False).

Image Enhancement Properties

Resample As Boolean - If set to True, this determines that an anti-aliasing technique is used to improve image quality of rotated images (see *Rotate* method). (Default = False).

Selecting a Region of an Image

The following methods and properties enable a part of an image to be selected, so that this part can be processed, saved or copied without affecting the remainder of the image.

Selecting a Region - Methods

SelectRectangle (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Selects a rectangular region bounded by X1, Y1 and X2, Y2.

SelectEllipse (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Selects an elliptical region. The ellipse is bounded by the rectangle defined by the points X1, Y1 and X2, Y2.

MouseSelectRectangle () - Selects a rectangular region of the image by using the mouse. The selection is started by pressing and holding the left mouse button. This defines one corner of the region. The mouse is then moved to the opposite corner of the region and the mouse button released. Code execution does not continue to the next line after the *MouseSelectRectangle* call until the mouse actions are completed, unless the call is cancelled using *CancelSelection*.

MouseSelectEllipse () - Selects an elliptical region of the image by using the mouse. This method operates in exactly the same way as *MouseSelectRectangle*.

GetSelectionCoords (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Retrieves the values of the X1, Y1 and X2, Y2 co-ordinates of a currently selected rectangle or ellipse.

CancelSelection () - Cancels the current selection, so no region is selected. The property *UseSelection* is set to False. *CancelSelection* can also be used to end a *MouseSelectRectangle* or *MouseSelectEllipse* call without receiving input from mouse actions.

CropToSelection () - Crops the image to a rectangle containing the selected region. If the selected region is non-rectangular, empty pixels are filled with the background colour *BGColor*.

Selecting a Region - Properties

UseSelection As Boolean - If this property is set to True, certain methods when used will be applied to the selected region, not to the whole image. This applies to the following methods:

- SaveToFile
- WriteBinary
- NewFileSize
- Copy
- Brightness
- Contrast
- Sharpen & SharpenBy
- Blur & BlurBy
- ReduceRedEye
- PrintImage

(Default = False).

SelectionVisible As Boolean - Determines whether the outline of the selected region is displayed on the image. (Default = True).

SelectionType As TxSelectionType - Read-only property to identify which shape of region is currently selected.

SelectionType can be one of:

<i>seNone:</i>	No region currently selected.
<i>seRectangle:</i>	A rectangle is currently selected.
<i>seEllipse:</i>	An ellipse is currently selected.

Drawing

A number of simple methods and properties are provided to enable drawing of lines, basic shapes and text on the image. Lines and the perimeters of shapes are drawn by the Pen. Shapes are filled by the Brush.

Drawing Properties

PenColor As OLE_COLOR - The colour of the Pen which will be used for drawing lines or the perimeters of shapes. (Default = Black, &H00000000).

PenMode As TxPenMode - Defines the way that the Pen will draw. Possible values are:

<i>pmBlack:</i>	Always black
<i>pmWhite:</i>	Always white
<i>pmNop:</i>	Unchanged
<i>pmNot:</i>	Inverse of image background colour
<i>pmCopy:</i> (Default)	Pen colour specified in <i>PenColor</i> property
<i>pmNotCopy:</i>	Inverse of pen colour
<i>pmMergePenNot:</i>	Combination of pen colour and inverse of image background
<i>pmMaskPenNot:</i>	Combination of colours common to both pen and inverse of image background
<i>pmMergeNotPen:</i>	Combination of image background colour and inverse of pen colour
<i>pmMaskNotPen:</i>	Combination of colours common to both image background and inverse of pen
<i>pmMerge:</i>	Combination of pen colour and image background colour
<i>pmNotMerge:</i>	Inverse of <i>pmMerge</i> : combination of pen colour and image background colour

<i>pmMask:</i>	Combination of colours common to both pen and image background
<i>pmNotMask:</i>	Inverse of <i>pmMask</i> : combination of colours common to both pen and image background
<i>pmXor:</i>	Combination of colours in either pen or image background, but not both
<i>pmNotXor:</i>	Inverse of <i>pmXor</i> : combination of colours in either pen or image background, but not both

PenStyle As TxPenStyle - Defines the style in which the Pen draws lines. Possible values are:

<i>psSolid:</i> (Default)	A solid line
<i>psDash:</i>	A line made up of a series of dashes
<i>psDot:</i>	A line made up of a series of dots
<i>psDashDot:</i>	A line made up of alternating dashes and dots
<i>psDashDotDot:</i>	A line made up of a series of dash-dot-dot combinations
<i>psClear:</i>	No line is drawn (used to omit the line around shapes that draw an outline using the current pen)

Note: Dotted or dashed pen styles are only available when the *PenWidth* property is 1.

PenWidth As Long - Defines the maximum width of the Pen in pixels. (Default = 1).

BrushColor As OLE_COLOR - The colour of the Pen which will be used for filling shapes. (Default = White, &H00FFFFFF).

BrushStyle As TxBrushStyle - Defines the pattern for the Brush. Possible values are:

<i>bsSolid:</i> (Default)	Fills the shape with <i>BrushColor</i>
<i>bsClear:</i>	No fill
<i>bsBDiagonal:</i>	Diagonal lines sloping bottom-left to top-right
<i>bsFDiagonal:</i>	Diagonal lines sloping top-left to bottom-right
<i>bsCross:</i>	Cross-hatching with vertical and horizontal lines
<i>bsDiagCross:</i>	Cross-hatching with diagonal lines
<i>bsHorizontal:</i>	Horizontal lines
<i>bsVertical:</i>	Vertical lines

TextFont As IFontDisp - Defines the font to be used for the *DrawText* method.

TextAngle As Single - The angle in degrees, measures counter-clockwise from the horizontal, of text written by the *DrawText* method. (Default = 0).

TextTransparent As Boolean - If *TextTransparent* is True, text written by the *DrawText* method will have a transparent background. (Default = False).

PixelColor (X As Long, Y As Long) As OLE_COLOR - The colour of the pixel at co-ordinates X, Y.

FloodFillColor As OLE_COLOR - The colour that defines the boundary of the *FloodFill* method. (Default = Black, &H00000000).

FloodFillStyle As TxFloodFillStyle - Defines the behaviour of the *FloodFill* method. Possible values are:

<i>fsBorder:</i>	A region bounded by pixels of colour <i>FloodFillColor</i> is filled.
<i>fsSurface:</i> (Default).	A region including only pixels of colour <i>FloodFillColor</i> and bounded by other colours is filled.

Drawing Methods

DrawLine (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Draws a straight line from *X1, Y1* up to but not including *X2, Y2* using the current Pen settings.

Rectangle (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Draws a rectangle using the current Pen settings and filled using the current Brush settings. The rectangle is bounded by the points (*X1, Y1*) and (*X2, Y2*).

Ellipse (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long*) - Draws an ellipse using the current Pen settings and filled using the current Brush settings. The ellipse is bounded by the rectangle defined by the points *X1, Y1* and *X2, Y2*.

RoundRect (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, X3 As Long, Y3 As Long*) - Draws a rectangle with rounded corners using the current Pen settings and filled using the current Brush settings. The rectangle is bounded by the points (*X1, Y1*) and (*X2, Y2*). The curve of the rounded corners matches the curvature of an ellipse with width *X3* and height *Y3*.

Pie (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, X3 As Long, Y3 As Long, X4 As Long, Y4 As Long*) - Draws a pie shaped section of an ellipse using the current Pen settings and filled using the current Brush settings. The ellipse is formed from a rectangle bounded by the points (*X1, Y1*) and (*X2, Y2*). The sides of the pie are defined by the intersections between the centre of the ellipse and the points (*X3, Y3*) and (*X4, Y4*).

Arc (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, X3 As Long, Y3 As Long, X4 As Long, Y4 As Long*) - Draws an arc. The arc traverses the perimeter of an ellipse that is bounded by the points (*X1, Y1*) and (*X2, Y2*). The arc is drawn following the perimeter of the ellipse, counter-clockwise, from the starting point to the ending point. The starting point is defined by the intersection of the ellipse and a line defined by the centre of the ellipse and (*X3, Y3*). The ending point is defined by the intersection of the ellipse and a line defined by the centre of the ellipse and (*X4, Y4*).

Chord (*X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, X3 As Long, Y3 As Long, X4 As Long, Y4 As Long*) - Draws a shape that is defined by an arc and a line that joins the endpoints of the arc. The chord consists of a portion of an ellipse that is bounded by the points (*X1, Y1*) and (*X2, Y2*). The ellipse is bisected by a line that runs between the points (*X3, Y3*) and (*X4, Y4*).

The perimeter of the chord runs counter-clockwise from (*X3, Y3*), counter-clockwise along the ellipse to (*X4, Y4*), and straight back to (*X3, Y3*). If (*X3, Y3*) and (*X4, Y4*) are not on the surface of the ellipse, the corresponding corners on the chord are the closest points on the perimeter that intersect the line.

FloodFill (*X As Long, Y As Long*) - Fills a region of the image with *BrushColor*. The fill starts at co-ordinates *X, Y* and moves outwards until a boundary is reached. The boundary is defined by *FloodFillColor* and *FloodFillStyle*. *FloodFillStyle* can be *fsBorder*, in which case, the fill continues to a boundary of pixels of *FloodFillColor*, or can be *fsSurface*, in which case pixels of *FloodFillColor* are filled.

DrawText (*X As Long, Y As Long, TextString As String*) - Writes the text contained in *TextString* onto the image, starting at the point (*X, Y*) and using the font defined by property *TextFont*. The text can be written at an angle specified in the property *TextAngle*. The colour of the text is *PenColor* and the colour of the background to the text is *BrushColor*. The property *TextTransparent* can be set to True to give a transparent background.

Printing

Functions are provided in `csXImage` to enable printing, either through a print dialogue, which includes a print preview, or by sending images directly to a printer, controlled by a number of properties. If it is desired to write an application with a customised print dialogue, then normal print functions can be used from most programming environments, independent of the functions described here.

Printing Methods

PrintImage () - Starts printing, either through a dialogue, or immediately, depending on the value of the boolean property *UsePrintDialog*.

Printing Properties

PrinterIndex As Long - The index number of the selected printer. Setting this property to -1 selects the default printer. (Default = -1).

PrintUnits As TxPrintUnits - The units of measurement to be used for the properties *PrintLeft* & *PrintTop* which determine the position of the printed image. Possible values are:

<i>puCm</i> :	(Default)	Centimetres (cm)
<i>puInch</i> :		Inches (in)

PrintLeft As Double - The distance from the left side of the page to the left side of the image, in units specified by *PrintUnits*. Note that this property is not used if *PrintFit* or *PrintCentre* are set to True. (Default = 0).

PrintTop As Double - The distance from the top of the page to the top of the image, in units specified by *PrintUnits*. Note that this property is not used if *PrintFit* or *PrintCentre* are set to True. (Default = 0).

UsePrintDialog As Boolean - If set to True, a dialogue is displayed when *PrintImage* is called. This dialogue allows selection of printer and adjustment of various print properties. It also provides a print preview. If set to False, the image prints immediately when *PrintImage* is called. (Default = True).

PrintCopies As Long - The number of copies to be printed. (Default = 1).

PrintScale As Double - The scaling factor of the printed image. The normal size of a printed image is determined by the size of the image in pixels and the resolution of the image in pixels per inch, or per meter (see properties *XDPI* etc.). If *PrintScale* is set to a value different from 100%, the image size will be adjusted accordingly. Note that this property is not used if *PrintFit* is set to True. (Default = 100).

PrintOrientation As TxPrinterOrientation - The orientation of the printed image. Possible values are:

<i>poPortrait</i> :	(Default)	Portrait
<i>poLandscape</i> :		Landscape

PrintFit As Boolean - If set to True, the size of the printed image is adjusted to fit the page. (Default = False).

PrintCentre As Boolean - If set to True, the printed image is positioned at the centre of the page. (Default = False).

Miscellaneous

Miscellaneous Properties

ImageHeight As Long - The height of the loaded image, in pixels.

This is a read-only property. To change the height of an image, an appropriate method should be used, e.g., *ResizeImage*.

ImageWidth As Long - The width of the loaded image, in pixels.

This is a read-only property. To change the width of an image, an appropriate method should be used, e.g., *ResizeImage*.

ImageLoaded As Boolean - Indicates whether an image is currently loaded into the control.

XDPI As Long - The horizontal pixel density of the image, in pixels per inch (dots per inch or DPI).

YDPI As Long - The vertical pixel density of the image, in pixels per inch (dots per inch or DPI).

XPelsPerMeter As Long - The horizontal pixel density of the image, in pixels per meter.

YPelsPerMeter As Long - The vertical pixel density of the image, in pixels per meter.

Note: The values of XDPI and XPelsPerMeter will remain consistent with each other if one is changed by the user. For example, if the user sets the value of XDPI, XPelsPerMeter will also be updated. The same applies to YDPI and YPelsPerMeter.

HasScrollBarHoriz As Boolean - This will be True if the image is currently displaying a horizontal scroll bar.

HasScrollBarVert As Boolean - This will be True if the image is currently displaying a vertical scroll bar.

ScrollBarHorizPos As Long - The position of the horizontal scroll bar in pixels. This value is equal to the distance in pixels between the left side of the image and the left-most pixel currently displayed.

ScrollBarVertPos As Long - The position of the vertical scroll bar in pixels. This value is equal to the distance in pixels between the top edge of the image and the top-most pixel currently displayed.

ScrollBarHorizWidth As Long - The width of the horizontal scroll bar in pixels. This value is also equal to the width of the displayed portion of the image.

ScrollBarVertHeight As Long - The height of the vertical scroll bar in pixels. This value is also equal to the height of the displayed portion of the image.

Miscellaneous Methods

Clear () - Deletes the current image from memory and redraws the empty control.

NewImage (*NewWidth* As Long, *NewHeight* As Long) - Creates a new image of size *NewWidth* x *NewHeight* pixels. The image is filled with the colour specified in *BGColor*. Any existing image in the control is cleared from memory without warning. By default, the image has *ColorFormat* cf24bit.

Events

The following mouse events are raised by the control:

OnMouseMove (*ShiftState* As Integer, *X* As Long, *Y* As Long) - This event occurs when the mouse is moved over any part of the ImageBox control, including the image area, the scrollbars if present, or the empty part of the control when an image smaller than the control is loaded. It is also fired by the mouse moving over the empty control with no image loaded.

OnMouseDown (*Button* As Integer, *ShiftState* As Integer, *X* As Long, *Y* As Long) - This event occurs when a mouse button is pressed over any part of the control.

OnMouseUp (*Button* As Integer, *ShiftState* As Integer, *X* As Long, *Y* As Long) - This event occurs when a mouse button is released over any part of the control.

Button is an integer which identifies the mouse button that was pressed or released to fire an *OnMouseDown* or *OnMouseUp* event. The value of *Button* corresponds to the values in *ShiftState* below, i.e., Left = 1, Right = 2, Middle = 4.

ShiftState is an integer which indicates the status of the mouse buttons and the SHIFT, CTRL and ALT keys on the keyboard at the time the event was fired. *ShiftState* is a bit field where each bit corresponds to a button or key and is set if the button or key is down. The bits are defined as follows:

Bit no.	Value	Button or Key
0	1	Left Mouse Button
1	2	Right Mouse Button
2	4	Middle Mouse Button
3	8	SHIFT Key
4	16	CTRL Key
5	32	ALT Key

For example, if the left mouse button plus the SHIFT key were pressed, the value of *ShiftState* would be 9 (=1+8).

X and *Y* give the co-ordinates of the mouse position over the control, measured in pixels from the top left corner of the control. Note that these co-ordinates may not correspond to the co-ordinates of the pixel in an image under the mouse pointer due to the image being in a scrolled position. The read-only properties of the control which give information about the scroll bar status can be used to work out the actual pixel co-ordinates. (For VB users, the demo VB application provided with csXImage contains an example of how this can be achieved. This is in the procedure 'mnuToolsPixelInfo').

OnClick () - This event occurs when a mouse button is clicked (pressed and then released) over any part of the control.

OnDbClick () - This event occurs when a mouse button is double-clicked over any part of the control.

File Info (JPEG Meta Data)

The JPEG and PSD file formats allow for text describing the image to be embedded in the file header. This is used in a number of applications including journalism (IPTC text). If a JPEG or PSD is loaded that contains File Info readable by csXImage the property HasFileInfo is set to True and the relevant properties are set. To save a JPEG or PSD with File Info the relevant data properties must be set. The HasFileInfo property will be set to True when any of the properties are written to. File Info can be read from or written to a separate file which uses the extension .FFO. Note that File Info is not supported for TIFF files.

File Info Properties

HasFileInfo As Boolean - When an image is loaded that contains File Info this property is set to True. It also determines whether File Info is exported with a JPEG or PSD. Setting any File Info property sets HasFileInfo to True. (Default = False).

FFO_Caption As String.

FFO_CaptionWriter As String.

FFO_Headline As String.

FFO_SpecialInstructions As String.

FFO_Category As String.

FFO_Byline As String.

FFO_BylineTitle As String.

FFO_Credit As String.

FFO_Source As String.

FFO_ObjectName As String.

FFO_City As String.

FFO_ProvinceState As String.

FFO_CountryName As String.

FFO_OTR As String - Original Transmission Reference.

FFO_CopyrightNotice As String.

FFO_ImageURL As String.

FFO_Urgency As Integer - Not saved/empty if set to 0.

FFO_DateCreated As Date.

FFO_CopyrightFlag As Boolean.

The Keywords and Supplemental Categories properties are zero based arrays of strings. Some additional properties and methods are needed to read and write them.

FFO_Keywords (*Index* As Long) As String - The keyword defined by the integer *Index*.

FFO_KeywordsCount As Integer - The number of items in the list. (Read-only).

FFO_KeywordsAdd (*Keyword* As String) As Long - Adds the string *Keyword* to the end of the list. Returns the new number of items in the list as an integer.

FFO_KeywordsDelete (*Index* As Long) - Deletes the keyword defined by the integer *Index*.

FFO_KeywordsClear - Deletes all the keywords.

FFO_KeywordsInsert (*Index* As Long, *Keyword* As String) - Inserts the string *Keyword* at position *Index* in the list. *Index* is an integer.

FFO_SuppCat (*Index* As Long) As String - The category defined by the integer *Index*.

FFO_SuppCatCount As Integer - The number of items in the list. (Read-only).

FFO_SuppCatAdd (*Cat* As String) As Long - Adds the string *Cat* to the end of the list. Returns the new number of items in the list as an integer.

FFO_SuppCatDelete (*Index* As Long) - Deletes the category defined by the integer *Index*.

FFO_SuppCatClear - Deletes all the categories.

FFO_SuppCatInsert (*Index* As Long, *Cat* As String) - Inserts the string *Cat* at position *Index* in the list. *Index* is an integer.

File Info Methods

FFO_Clear - Deletes all File Info values and sets *HasFileInfo* to False.

FFO_Save (*FileName* As String) - Writes the File Info data to a file (.FFO file). *FileName* is the physical path and file name, complete with extension.

FFO_Load (*FileName* As String) - Loads File Info data from a file. *FileName* is the physical path and file name, complete with extension.

Enumerations

For reference, the enumerated constants used in csXImage correspond to the following numerical values:

TxBrushStyle		TxGraphicsFormat		TxPenStyle	
bsSolid:	0	gfBMP:	0	psSolid:	0
bsClear:	1	gfGIF:	1	psDash:	1
bsHorizontal:	2	gfJPG:	2	psDot:	2
bsVertical:	3	gfPCX:	3	psDashDot:	3
bsBDiagonal:	4	gfPNG:	4	psDashDotDot:	4
bsFDiagonal:	5	gfWBMP:	5	psClear:	5
bsCross:	6	gfPSD:	6		
bsDiagCross:	7	gfTIF:	7	TxPrinterOrientation	
				poPortrait:	0
TxColorFormat		TxPenMode		poLandscape:	1
cf24bit:	0	pmBlack:	0		
cf256Color:	1	pmWhite:	1	TxPrintUnits	
cf256Grayscale:	2	pmNop:	2	pucm:	0
cf16Color:	3	pmNot:	3	puInch:	1
cf16Grayscale:	4	pmCopy:	4		
cfMono:	5	pmNotCopy:	5	TxSelectionType	
cfMonoBW:	6	pmMergePenNot:	6	seNone:	0
cfNone:	7	pmMaskPenNot:	7	seRectangle:	1
		pmMergeNotPen:	8	seEllipse:	2
TxFlipMode		pmMaskNotPen:	9		
fmHoriz:	0	pmMerge:	10	TxWMStyle	
fmVert:	1	pmNotMerge:	11	wmCentre:	0
		pmMask:	12	wmSingle:	1
TxFloodFillStyle		pmNotMask:	13	wmTile:	2
fsBorder:	0	pmXor:	14	wmWrap:	3
fsSurface:	1	pmNotXor:	15		

Deploying an Application

In order to deploy an application that uses csXImage you will need to distribute the OCX file, csXImage.ocx, together with the files that make up your application. This file will need to be registered on the machine running your application and you may wish to use a proprietary installer to do this. When csXImage was installed on your system our installer will have copied the OCX file to the directory "Program Files\Chestysoft\csXImage\", assuming you used the installer and accepted the defaults.

The number of copies of the OCX file that may be distributed is not limited by the licence. In order to use the control in a design environment the licence file, csXImage.lic, is also required. The number of machines this may be installed on is governed by your licence agreement and it must not be copied to any more machines than permitted by the licence. Note that this file is required if the control is used on a web server for an ASP or similar application and this would count as one of the installations.

Revision History

The current version of csXImage is 1.2.

New in Version 1.1

Methods for selecting and processing a region of the image.

Mouse events.

Properties giving information about scroll bar status.

Red-eye reduction.

File Info (JPEG Meta Data)

New in Version 1.2

Support for PSD and TIFF file formats.

Print methods and properties.

PictureData property.

Other Products From Chestysoft

Visit the Chestysoft web site for details of other COM objects.

ActiveX Controls

[csXGraph](#)

- An OCX control to draw pie charts, bar charts and line graphs.

ASP Components

[csImageFile](#)

[csDrawGraph](#)

[csWAPDraw](#)

[csIniFile](#)

[csASPUpload](#)

[csASPZipFile](#)

[csFileDownload](#)

- Similar functionality to csXImage but in an ASP component.
- Component to draw pie charts, bar charts and line graphs.
- Create and edit wireless bitmaps for WAP devices.
- Read and Edit Windows style inifiles.
- Process file uploads through a browser.
- Create zip files and control binary file downloads.
- Control file downloads with an ASP script.

Chestysoft, May 2003.

www.chestysoft.com