Manual for wxSVGFileDC

Chris Elliott

June 2002

Contents

Copyright notice

wxSVGFileDC

wxSVGFileDC

A wxSVGFileDC is a *device context* onto which graphics and text can be drawn, and the output produced as a vector file, in the SVG format (see http://www.w3.org/TR/2001/REC-SVG-20010904/). This format can be read by a range of programs, including a Netscape plugin (Adobe), full details at http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8 Vector formats may often be smaller than raster formats.

The intention behind wxSVGFileDC is that it can be used to produce a file corresponding to the screen display context, wxSVGFileDC, by passing the wxSVGFileDC as a parameter instead of a wxSVGFileDC. Thus the wxSVGFileDC is a write-only class.

As the wxSVGFileDC is a vector format, raster operations like GetPixel are unlikely to be supported. However, the SVG specification allows for PNG format raster files to be embedded in the SVG, and so bitmaps, icons and blit operations into the wxSVGFileDC are supported.

A more substantial SVG library (for reading and writing) is available at http://wxart2d.sourceforge.net/

Derived from

wxDCBase

Include files

<wx/dcsvg.h>

See also

wxSVGFileDC::wxSVGFileDC

wxSVGFileDC(**wxString** *f*) **wxSVGFileDC**(**wxString** *f*, **int** *Width*,**int** *Height*)
wxSVGFileDC(**wxString** *f*, **int** *Width*,**int** *Height*,**float** *dpi*)

Constructors: a filename *f* with default size 340x240 at 72.0 dots per inch (a frequent screen resolution). a filename *f* with size *Width* by *Height* at 72.0 dots per inch a filename *f* with size *Width* by *Height* at *dpi* resolution.

wxSVGFileDC::~wxSVGFileDC

~wxSVGFileDC()

Destructor.

wxSVGFileDC::BeginDrawing

Does nothing

wxSVGFileDC::Blit

bool Blit(**wxCoord** *xdest*, **wxCoord** *ydest*, **wxCoord** *width*, **wxCoord** *height*, **wxSVGFileDC*** *source*, **wxCoord** *xsrc*, **wxCoord** *ysrc*, **int** *logicalFunc = wxCOPY*, **bool** *useMask = FALSE*, **wxCoord** *xsrcMask = -1*, **wxCoord** *ysrcMask = -1*)

As wxDC: Copy from a source DC to this DC, specifying the destination coordinates, size of area to copy, source DC, source coordinates, logical function, whether to use a bitmap mask, and mask source position.

wxSVGFileDC::CalcBoundingBox

void CalcBoundingBox(**wxCoord** *x*, **wxCoord** *y*)

Adds the specified point to the bounding box which can be retrieved with *MinX* (p. xii), *MaxX* (p. xii) and *MinY* (p. xii), *MaxY* (p. xii) functions.

wxSVGFileDC::Clear

void Clear()

This makes no sense in wxSVGFileDC and does nothing

wxSVGFileDC::CrossHair

void CrossHair(**wxCoord** *x*, **wxCoord** *y*)

Not Implemented

wxSVGFileDC::DestroyClippingRegion

void DestroyClippingRegion()

Not Implemented

wxSVGFileDC::DeviceToLogicalX

wxCoord DeviceToLogicalX(**wxCoord** *x*)

Convert device X coordinate to logical coordinate, using the current mapping mode.

wxSVGFileDC::DeviceToLogicalXRel

wxCoord DeviceToLogicalXRel(**wxCoord** *x*)

Convert device X coordinate to relative logical coordinate, using the current mapping

mode but ignoring the x axis orientation. Use this function for converting a width, for example.

wxSVGFileDC::DeviceToLogicalY

wxCoord DeviceToLogicalY(**wxCoord** *y*)

Converts device Y coordinate to logical coordinate, using the current mapping mode.

wxSVGFileDC::DeviceToLogicalYRel

wxCoord DeviceToLogicalYRel(**wxCoord** *y*)

Convert device Y coordinate to relative logical coordinate, using the current mapping mode but ignoring the y axis orientation. Use this function for converting a height, for example.

wxSVGFileDC::DrawArc

void DrawArc(**wxCoord** *x1*, **wxCoord** *y1*, **wxCoord** *x2*, **wxCoord** *y2*, **wxCoord** *xc*, **wxCoord** *yc*)

Draws an arc of a circle, centred on (*xc, yc*), with starting point (*x1, y1*) and ending at (*x2, y2*). The current pen is used for the outline and the current brush for filling the shape.

The arc is drawn in an anticlockwise direction from the start point to the end point.

wxSVGFileDC::DrawBitmap

void DrawBitmap(**const wxBitmap&** *bitmap*, **wxCoord** *x*, **wxCoord** *y*, **bool** *transparent*)

Draw a bitmap on the device context at the specified point. If *transparent* is true and the bitmap has a transparency mask, the bitmap will be drawn transparently.

When drawing a mono-bitmap, the current text foreground colour will be used to draw the foreground of the bitmap (all bits set to 1), and the current text background colour to draw the background (all bits set to 0). See also *SetTextForeground* (p. xv), *SetTextBackground* (p. xv) and *wxMemoryDC*.

wxSVGFileDC::DrawCheckMark

void DrawCheckMark(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)

void DrawCheckMark(**const wxRect &***rect*)

Draws a check mark inside the given rectangle.

wxSVGFileDC::DrawCircle

void DrawCircle(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *radius*)

void DrawCircle(**const wxPoint&** *pt*, **wxCoord** *radius*)

Draws a circle with the given centre and radius.

See also

DrawEllipse (p. vi)

wxSVGFileDC::DrawEllipse

void DrawEllipse(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)

void DrawEllipse(**const wxPoint&** *pt*, **const wxSize&** *size*)

void DrawEllipse(**const wxRect&** *rect*)

Draws an ellipse contained in the rectangle specified either with the given top left corner and the given size or directly. The current pen is used for the outline and the current brush for filling the shape.

See also

DrawCircle (p. v)

wxSVGFileDC::DrawEllipticArc

void DrawEllipticArc(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*, **double** *start*, **double** *end*)

Draws an arc of an ellipse. The current pen is used for drawing the arc and the current brush is used for drawing the pie.

x and *y* specify the x and y coordinates of the upper-left corner of the rectangle that contains the ellipse.

width and *height* specify the width and height of the rectangle that contains the ellipse.

start and *end* specify the start and end of the arc relative to the three-o'clock position from the center of the rectangle. Angles are specified in degrees (360 is a complete circle). Positive values mean counter-clockwise motion. If *start* is equal to *end*, a complete ellipse will be drawn.

wxSVGFileDC::DrawIcon

void DrawIcon(**const wxIcon&** *icon*, **wxCoord** *x*, **wxCoord** *y*)

Draw an icon on the display (does nothing if the device context is PostScript). This can be the simplest way of drawing bitmaps on a window.

wxSVGFileDC::DrawLine

void DrawLine(**wxCoord** *x1*, **wxCoord** *y1*, **wxCoord** *x2*, **wxCoord** *y2*)

Draws a line from the first point to the second. The current pen is used for drawing the line.

wxSVGFileDC::DrawLines

void DrawLines(**int** *n*, **wxPoint** *points[]*, **wxCoord** *xoffset = 0*, **wxCoord** *yoffset = 0*)

void DrawLines(**wxList ***_points_, **wxCoord** *xoffset = 0*, **wxCoord** *yoffset = 0*)

Draws lines using an array of *points* of size *n*, or list of pointers to points, adding the optional offset coordinate. The current pen is used for drawing the lines. The programmer is responsible for deleting the list of points.

wxSVGFileDC::DrawPolygon

void DrawPolygon(**int** *n*, **wxPoint** *points[]*, **wxCoord** *xoffset = 0*, **wxCoord** *yoffset = 0*, **int** *fill_style = wxODDEVEN_RULE*)

void DrawPolygon(**wxList ***_points_, **wxCoord** *xoffset = 0*, **wxCoord** *yoffset = 0*, **int** *fill_style = wxODDEVEN_RULE*)

Draws a filled polygon using an array of *points* of size *n*, or list of pointers to points, adding the optional offset coordinate.

The last argument specifies the fill rule: **wxODDEVEN_RULE** (the default) or **wxWINDING_RULE**.

The current pen is used for drawing the outline, and the current brush for filling the shape. Using a transparent brush suppresses filling. The programmer is responsible for deleting the list of points.

Note that wxWindows automatically closes the first and last points.

wxSVGFileDC::DrawPoint

void DrawPoint(**wxCoord** *x*, **wxCoord** *y*)

Draws a point using the current pen.

wxSVGFileDC::DrawRectangle

void DrawRectangle(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)

Draws a rectangle with the given top left corner, and with the given size. The current pen is used for the outline and the current brush for filling the shape.

wxSVGFileDC::DrawRotatedText

void DrawRotatedText(**const wxString&** *text*, **wxCoord** *x*, **wxCoord** *y*, **double** *angle*)

Draws the text rotated by *angle* degrees.

The wxMSW wxDC and wxSVGFileDC rotate the text around slightly different points, depending on the size of the font

wxSVGFileDC::DrawRoundedRectangle

void DrawRoundedRectangle(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*, **double** *radius = 20*)

Draws a rectangle with the given top left corner, and with the given size. The corners are quarter-circles using the given radius. The current pen is used for the outline and the current brush for filling the shape.

If *radius* is positive, the value is assumed to be the radius of the rounded corner. If *radius* is negative, the absolute value is assumed to be the *proportion* of the smallest dimension of the rectangle. This means that the corner can be a sensible size relative to the size of the rectangle, and also avoids the strange effects X produces when the corners are too big for the rectangle.

wxSVGFileDC::DrawSpline

void DrawSpline(**wxList ***_points_)

Draws a spline between all given control points, using the current pen. Doesn't delete the wxList and contents. The spline is drawn using a series of lines, using an algorithm taken from the X drawing program 'XFIG'.

void DrawSpline(**wxCoord** *x1*, **wxCoord** *y1*, **wxCoord** *x2*, **wxCoord** *y2*, **wxCoord** *x3*, **wxCoord** *y3*)

Draws a three-point spline using the current pen.

wxSVGFileDC::DrawText

void DrawText(**const wxString&** *text*, **wxCoord** *x*, **wxCoord** *y*)

Draws a text string at the specified point, using the current text font, and the current text foreground and background colours.

The coordinates refer to the top-left corner of the rectangle bounding the string. See *wxSVGFileDC::GetTextExtent* (p. xi) for how to get the dimensions of a text string, which can be used to position the text more precisely.

wxSVGFileDC::EndDoc

void EndDoc()

Does nothing

wxSVGFileDC::EndDrawing

void EndDrawing()

Does nothing

wxSVGFileDC::EndPage

void EndPage()

Does nothing

wxSVGFileDC::FloodFill

void FloodFill(**wxCoord** *x*, **wxCoord** *y*, **const wxColour&** *colour*, **int** *style=wxFLOOD_SURFACE*)

Not implemented

wxSVGFileDC::GetBackground

wxBrush& GetBackground()

const wxBrush& GetBackground() **const**

Gets the brush used for painting the background (see *wxSVGFileDC::SetBackground* (p. xiii)).

wxSVGFileDC::GetBackgroundMode

int GetBackgroundMode() **const**

Returns the current background mode: `wxSOLID` or `wxTRANSPARENT`.

See also

SetBackgroundMode (p. xiii)

wxSVGFileDC::GetBrush

wxBrush& GetBrush()

const wxBrush& GetBrush() **const**

Gets the current brush (see *wxSVGFileDC::SetBrush* (p. xiv)).

wxSVGFileDC::GetCharHeight

wxCoord GetCharHeight()

Gets the character height of the currently set font.

wxSVGFileDC::GetCharWidth

wxCoord GetCharWidth()

Gets the average character width of the currently set font.

wxSVGFileDC::GetClippingBox

void GetClippingBox(**wxCoord** *x, **wxCoord** *y, **wxCoord** *width, **wxCoord** *height)

Not implemented

wxSVGFileDC::GetFont

wxFont& GetFont()

const wxFont& GetFont() **const**

Gets the current font (see *wxSVGFileDC::SetFont* (p. xiv)).

wxSVGFileDC::GetLogicalFunction

int GetLogicalFunction()

Gets the current logical function (see *wxSVGFileDC::SetLogicalFunction* (p. xiv)).

wxSVGFileDC::GetMapMode

int GetMapMode()

Gets the *mapping mode* for the device context (see *wxSVGFileDC::SetMapMode* (p. xiv)).

wxSVGFileDC::GetPen

wxPen& GetPen()

const wxPen& GetPen() **const**

Gets the current pen (see *wxSVGFileDC::SetPen* (p. xv)).

wxSVGFileDC::GetPixel

bool GetPixel(**wxCoord** *x*, **wxCoord** *y*, **wxColour ***colour*)

Not implemented

wxSVGFileDC::GetSize

void GetSize(**wxCoord ***width*, **wxCoord ***height*)

For a Windows printer device context, this gets the horizontal and vertical resolution.

wxSVGFileDC::GetTextBackground

wxColour& GetTextBackground()

const wxColour& GetTextBackground() **const**

Gets the current text background colour (see *wxSVGFileDC::SetTextBackground* (p. xv)).

wxSVGFileDC::GetTextExtent

void GetTextExtent(**const wxString&** *string*, **wxCoord ***_w_, **wxCoord ***_h_, **wxCoord ***_descent_ = *NULL*, **wxCoord ***_externalLeading_ = *NULL*, **wxFont ***_font_ = *NULL*)

Gets the dimensions of the string using the currently selected font. *string* is the text string to measure, *w* and *h* are the total width and height respectively, *descent* is the dimension from the baseline of the font to the bottom of the descender, and *externalLeading* is any extra vertical space added to the font by the font designer (usually is zero).

The optional parameter *font* specifies an alternative to the currently selected font: but note that this does not yet work under Windows, so you need to set a font for the device context first.

See also *wxFont*, *wxSVGFileDC::SetFont* (p. xiv).

wxSVGFileDC::GetTextForeground

wxColour& GetTextForeground()

const wxColour& GetTextForeground() **const**

Gets the current text foreground colour (see *wxSVGFileDC::SetTextForeground* (p. xv)).

wxSVGFileDC::GetUserScale

void GetUserScale(**double** *_x_, **double** *_y_)

Gets the current user scale factor (set by *SetUserScale* (p. xv)).

wxSVGFileDC::LogicalToDeviceX

wxCoord LogicalToDeviceX(**wxCoord** _x_)

Converts logical X coordinate to device coordinate, using the current mapping mode.

wxSVGFileDC::LogicalToDeviceXRel

wxCoord LogicalToDeviceXRel(**wxCoord** _x_)

Converts logical X coordinate to relative device coordinate, using the current mapping mode but ignoring the x axis orientation. Use this for converting a width, for example.

wxSVGFileDC::LogicalToDeviceY

wxCoord LogicalToDeviceY(**wxCoord** *y*)

Converts logical Y coordinate to device coordinate, using the current mapping mode.

wxSVGFileDC::LogicalToDeviceYRel

wxCoord LogicalToDeviceYRel(**wxCoord** *y*)

Converts logical Y coordinate to relative device coordinate, using the current mapping mode but ignoring the y axis orientation. Use this for converting a height, for example.

wxSVGFileDC::MaxX

wxCoord MaxX()

Gets the maximum horizontal extent used in drawing commands so far.

wxSVGFileDC::MaxY

wxCoord MaxY()

Gets the maximum vertical extent used in drawing commands so far.

wxSVGFileDC::MinX

wxCoord MinX()

Gets the minimum horizontal extent used in drawing commands so far.

wxSVGFileDC::MinY

wxCoord MinY()

Gets the minimum vertical extent used in drawing commands so far.

wxSVGFileDC::Ok

bool Ok()

Returns true if the DC is ok to use; False values arise from being unable to write the file

wxSVGFileDC::ResetBoundingBox

void ResetBoundingBox()

Resets the bounding box: after a call to this function, the bounding box doesn't contain anything.

See also

CalcBoundingBox (p. iv)

wxSVGFileDC::SetAxisOrientation

void SetAxisOrientation(**bool** *xLeftRight*, **bool** *yBottomUp*)

Sets the x and y axis orientation (i.e., the direction from lowest to highest values on the axis). The default orientation is the natural orientation, e.g. x axis from left to right and y axis from bottom up.

Parameters

xLeftRight

True to set the x axis orientation to the natural left to right orientation, false to invert it.

yBottomUp

True to set the y axis orientation to the natural bottom up orientation, false to invert it.

wxSVGFileDC::SetDeviceOrigin

void SetDeviceOrigin(**wxCoord** *x*, **wxCoord** *y*)

Sets the device origin (i.e., the origin in pixels after scaling has been applied).

This function may be useful in Windows printing operations for placing a graphic on a page.

wxSVGFileDC::SetBackground

void SetBackground(**const wxBrush&** *brush*)

Sets the current background brush for the DC.

wxSVGFileDC::SetBackgroundMode

void SetBackgroundMode(**int** *mode*)

mode may be one of wxSOLID and wxTRANSPARENT. This setting determines whether text will be drawn with a background colour or not.

wxSVGFileDC::SetClippingRegion

void SetClippingRegion(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)

void SetClippingRegion(**const wxPoint&** *pt*, **const wxSize&** *sz*)

void SetClippingRegion(**const wxRect&** *rect*)

void SetClippingRegion(**const wxRegion&** *region*)

Not implemented

wxSVGFileDC::SetPalette

void SetPalette(**const wxPalette&** *palette*)

Not implemented

wxSVGFileDC::SetBrush

void SetBrush(**const wxBrush&** *brush*)

Sets the current brush for the DC.

If the argument is wxNullBrush, the current brush is selected out of the device context, and the original brush restored, allowing the current brush to be destroyed safely.

See also *wxBrush*.

See also *wxMemoryDC* for the interpretation of colours when drawing into a monochrome bitmap.

wxSVGFileDC::SetFont

void SetFont(**const wxFont&** *font*)

Sets the current font for the DC. It must be a valid font, in particular you should not pass `wxNullFont` to this method.

See also *wxFont*.

wxSVGFileDC::SetLogicalFunction

void SetLogicalFunction(**int** *function*)

Only wxCOPY is avalaible; trying to set one of the othe values will fail

wxSVGFileDC::SetMapMode

void SetMapMode(**int** *int*)

The *mapping mode* of the device context defines the unit of measurement used to convert logical units to device units. Note that in X, text drawing isn't handled consistently with the mapping mode; a font is always specified in point size. However, setting the *user scale* (see *wxSVGFileDC::SetUserScale* (p. xv)) scales the text appropriately. In Windows, scaleable TrueType fonts are always used; in X, results

depend on availability of fonts, but usually a reasonable match is found.

Note that the coordinate origin should ideally be selectable, but for now is always at the top left of the screen/printer.

Drawing to a Windows printer device context under UNIX uses the current mapping mode, but mapping mode is currently ignored for PostScript output.

The mapping mode can be one of the following:

wxMM_TWIPS Each logical unit is 1/20 of a point, or 1/1440 of an inch.

wxMM_POINTS Each logical unit is a point, or 1/72 of an inch.

wxMM_METRIC Each logical unit is 1 mm.

wxMM_LOMETRIC Each logical unit is 1/10 of a mm.

wxMM_TEXT Each logical unit is 1 pixel.

wxSVGFileDC::SetPen

void SetPen(**const wxPen&** *pen*)

Sets the current pen for the DC.

If the argument is wxNullPen, the current pen is selected out of the device context, and the original pen restored.

See also *wxMemoryDC* for the interpretation of colours when drawing into a monochrome bitmap.

wxSVGFileDC::SetTextBackground

void SetTextBackground(**const wxColour&** *colour*)

Sets the current text background colour for the DC.

wxSVGFileDC::SetTextForeground

void SetTextForeground(**const wxColour&** *colour*)

Sets the current text foreground colour for the DC.

See also *wxMemoryDC* for the interpretation of colours when drawing into a monochrome bitmap.

wxSVGFileDC::SetUserScale

void SetUserScale(**double** *xScale*, **double** *yScale*)

Sets the user scaling factor, useful for applications which require 'zooming'.

wxSVGFileDC::StartDoc

bool StartDoc(**const wxString&** *message*)

Does nothing

wxSVGFileDC::StartPage

bool StartPage()

Does nothing