

#+K!

Manual for wxSVGFileDC

by Chris Elliott

Contents

Copyright notice  
wxSVGFileDC

---

C contents  
C contents  
browse00001  
K Contents  
DisableButton("Up")

<sup>\$#+K!</sup>**Copyright notice**

---

<sup>c</sup>opyright notice  
<sup>t</sup>opic0  
<sup>b</sup>rowse00002  
<sup>K</sup> Copyright notice  
<sup>D</sup>isableButton("Up")

**\$#+K!wxSVGFileDC**

wxSVGFileDC

---

<sup>w</sup>xSVGFileDC  
<sup>t</sup>opic1  
<sup>b</sup>rowse00003  
<sup>K</sup> wxSVGFileDC  
<sup>D</sup>isableButton("Up")



## \$#+K! 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.htm>8 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://www.xs4all.nl/~kholwerd/wxstuff/canvas/htmldocbook/aap.html>

wxheadingDerived from

wxDCBase

wxheadingInclude files

<wx/dcsvg.h>

wxheadingSee also

wxheadingMembers

wxSVGFileDC::wxSVGFileDC  
wxSVGFileDC::~wxSVGFileDC  
wxSVGFileDC::BeginDrawing  
wxSVGFileDC::Blit  
wxSVGFileDC::CalcBoundingBox  
wxSVGFileDC::Clear  
wxSVGFileDC::CrossHair  
wxSVGFileDC::DestroyClippingRegion  
wxSVGFileDC::DeviceToLogicalX  
wxSVGFileDC::DeviceToLogicalXRel  
wxSVGFileDC::DeviceToLogicalY  
wxSVGFileDC::DeviceToLogicalYRel

---

<sup>w</sup>xSVGFileDC

<sup>w</sup>xSVGFileDC

<sup>b</sup>rowse00004

<sup>K</sup> wxSVGFileDC

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId('svg.hlp', `topic1`)" )

wxSVGFileDC::DrawArc  
wxSVGFileDC::DrawBitmap  
wxSVGFileDC::DrawCheckMark  
wxSVGFileDC::DrawCircle  
wxSVGFileDC::DrawEllipse  
wxSVGFileDC::DrawEllipticArc  
wxSVGFileDC::DrawIcon  
wxSVGFileDC::DrawLine  
wxSVGFileDC::DrawLines  
wxSVGFileDC::DrawPolygon  
wxSVGFileDC::DrawPoint  
wxSVGFileDC::DrawRectangle  
wxSVGFileDC::DrawRotatedText  
wxSVGFileDC::DrawRoundedRectangle  
wxSVGFileDC::DrawSpline  
wxSVGFileDC::DrawText  
wxSVGFileDC::EndDoc  
wxSVGFileDC::EndDrawing  
wxSVGFileDC::EndPage  
wxSVGFileDC::FloodFill  
wxSVGFileDC::GetBackground  
wxSVGFileDC::GetBackgroundMode



<sup>\$#+K!</sup>**wxSVGFileDC::wxSVGFileDC**

**wxSVGFileDC(wxString *f*)<sup>K</sup>**    **wxSVGFileDC(wxString *f*, int *Width*,int *Height*)<sup>K</sup>**  
**wxSVGFileDC(wxString *f*, int *Width*,int *Height*,float *dpi*)<sup>K</sup>**

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.

---

<sup>w</sup>xSVGFileDC::wxSVGFileDC

<sup>t</sup>opic2

<sup>b</sup>rowse00005

<sup>K</sup> wxSVGFileDC wxSVGFileDC

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId( svg.hlp', `wxSVGFileDC')")

<sup>K</sup> wxSVGFileDC

<sup>K</sup> wxSVGFileDC

<sup>K</sup> wxSVGFileDC



**`$#+K!wxSVGFileDC::~~wxSVGFileDC`**

**`~wxSVGFileDC()`**<sup>K</sup>

Destructor.

---

`wxSVGFileDC::~~wxSVGFileDC`

`topic3`

`browse00006`

`KwxSVGFileDC ~wxSVGFileDC`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

`K ~wxSVGFileDC`

**wxSVGFileDC::BeginDrawing**

Does nothing

---

**wxSVGFileDC::BeginDrawing**

**w\_xdcbegindrawing**

**rowse00007**

**w\_xSVGFileDC BeginDrawing**

**enableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `w\_xSVGFileDC')")**

\$#+K! **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)<sup>K</sup>

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::Blit

Wxdcbliit

Browse00008

K wxSVGFileDC Blit

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

K Blit

<sup>\$#+K!</sup>**wxSVGFileDC::CalcBoundingBox**

**void CalcBoundingBox(wxCoord x, wxCoord y)<sup>K</sup>**

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

---

<sup>w</sup>xSVGFileDC::CalcBoundingBox

<sup>w</sup>xdccalcboundingbox

<sup>b</sup>rowse00009

<sup>K</sup> wxSVGFileDC CalcBoundingBox

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> CalcBoundingBox

`$#+K!wxSVGFileDC::Clear`

`void Clear()`<sup>K</sup>

This makes no sense in wxSVGFileDC and does nothing

---

`wxSVGFileDC::Clear`

`w_xdcclear`

`browse00010`

`K wxSVGFileDC Clear`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

`K Clear`

\$#+K! **wxSVGFileDC::CrossHair**

**void CrossHair(wxCoord *x*, wxCoord *y*)**<sup>K</sup>

Not Implemented

---

wxSVGFileDC::CrossHair

wxdccrosshair

browse00011

K wxSVGFileDC CrossHair

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

K CrossHair

\$#+K! **wxSVGFileDC::DestroyClippingRegion**

**void DestroyClippingRegion()**<sup>K</sup>

Not Implemented

---

wxSVGFileDC::DestroyClippingRegion

wxdcdestroyclippingregion

browse00012

K wxSVGFileDC DestroyClippingRegion

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp`, `wxSVGFileDC`)"

K DestroyClippingRegion

**`wxSVGFileDC::DeviceToLogicalX`**

**`wxCoord DeviceToLogicalX(wxCoord x)`**<sup>K</sup>

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

---

<sup>w</sup>`wxSVGFileDC::DeviceToLogicalX`

<sup>w</sup>`xdcdevicetologicalx`

<sup>b</sup>`rowse00013`

<sup>K</sup>`wxSVGFileDC DeviceToLogicalX`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

<sup>K</sup>`DeviceToLogicalX`



\$#+K! **wxSVGFileDC::DeviceToLogicalXRel**

**wxCoord DeviceToLogicalXRel(wxCoord x)<sup>K</sup>**

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.

---

<sup>w</sup>wxSVGFileDC::DeviceToLogicalXRel

<sup>w</sup>xdcdevicetologicalxrel

<sup>b</sup>rowse00014

<sup>K</sup> wxSVGFileDC DeviceToLogicalXRel

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DeviceToLogicalXRel

**$\$#+K!$  wxSVGFileDC::DeviceToLogicalY**

**wxCoord DeviceToLogicalY(wxCoord y)<sup>K</sup>**

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

---

<sup>w</sup>xSVGFileDC::DeviceToLogicalY

<sup>w</sup>xdcdevicetologicaly

<sup>b</sup>rowse00015

<sup>K</sup> wxSVGFileDC DeviceToLogicalY

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DeviceToLogicalY

<sup>\$#+K!</sup>**wxSVGFileDC::DeviceToLogicalYRel**

**wxCoord DeviceToLogicalYRel(wxCoord y)<sup>K</sup>**

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.

---

<sup>w</sup>xSVGFileDC::DeviceToLogicalYRel

<sup>w</sup>xdcdevicetologicalyrel

<sup>b</sup>rowse00016

<sup>K</sup> wxSVGFileDC DeviceToLogicalYRel

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DeviceToLogicalYRel

**`wxSVGFileDC::DrawArc`**

**`void DrawArc(wxCoord x1, wxCoord y1, wxCoord x2, wxCoord y2, wxCoord xc, wxCoord yc)`**<sup>K</sup>

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.

---

<sup>w</sup>`wxSVGFileDC::DrawArc`

<sup>w</sup>`xdcdrawarc`

<sup>b</sup>`rowse00017`

<sup>K</sup>`wxSVGFileDC DrawArc`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

<sup>K</sup>`DrawArc`

<sup>\$#+K!</sup>**wxSVGFileDC::DrawBitmap**

**void DrawBitmap(const wxBitmap& *bitmap*, wxCoord *x*, wxCoord *y*, bool *transparent*)**<sup>K</sup>

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, SetTextBackground and wxMemoryDC.

---

<sup>w</sup>xSVGFileDC::DrawBitmap

<sup>w</sup>xcdrawbitmap

<sup>b</sup>rowse00018

<sup>K</sup> wxSVGFileDC DrawBitmap

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawBitmap

**`wxSVGFileDC::DrawCheckMark`**

**`void DrawCheckMark(wxCoord x, wxCoord y, wxCoord width, wxCoord height)`**<sup>K</sup>

**`void DrawCheckMark(const wxRect &rect)`**<sup>K</sup>

Draws a check mark inside the given rectangle.

---

<sup>w</sup>`wxSVGFileDC::DrawCheckMark`

<sup>w</sup>`xcdcdrawcheckmark`

<sup>b</sup>`rowse00019`

<sup>K</sup> `wxSVGFileDC DrawCheckMark`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(^svg.hlp', `wxSVGFileDC')")`

<sup>K</sup> `DrawCheckMark`

<sup>K</sup> `DrawCheckMark`

\$#+K! **wxSVGFileDC::DrawCircle**

**void DrawCircle(wxCoord *x*, wxCoord *y*, wxCoord *radius*)**<sup>K</sup>

**void DrawCircle(const wxPoint& *pt*, wxCoord *radius*)**<sup>K</sup>

Draws a circle with the given centre and radius.

wxheadingSee also

[DrawEllipse](#)

---

<sup>w</sup>wxSVGFileDC::DrawCircle

<sup>w</sup>xdcdrawcircle

<sup>b</sup>rowse00020

<sup>K</sup> wxSVGFileDC DrawCircle

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawCircle

<sup>K</sup> DrawCircle

**wxSVGFileDC::DrawEllipse**

**void DrawEllipse(wxCoord *x*, wxCoord *y*, wxCoord *width*, wxCoord *height*)**<sup>K</sup>

**void DrawEllipse(const wxPoint& *pt*, const wxSize& *size*)**<sup>K</sup>

**void DrawEllipse(const wxRect& *rect*)**<sup>K</sup>

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](#)

---

<sup>w</sup>wxSVGFileDC::DrawEllipse

<sup>w</sup>xdcdrawellipse

<sup>b</sup>rowse00021

<sup>K</sup> wxSVGFileDC DrawEllipse

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId('svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawEllipse

<sup>K</sup> DrawEllipse

<sup>K</sup> DrawEllipse



**`$#+K! wxSVGFileDC::DrawEllipticArc`**

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

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.

---

<sup>w</sup>`wxSVGFileDC::DrawEllipticArc`

<sup>w</sup>`xcdrawellipticarc`

<sup>b</sup>`rowse00022`

<sup>K</sup>`wxSVGFileDC DrawEllipticArc`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

<sup>K</sup>`DrawEllipticArc`

<sup>\$#+K!</sup>**wxSVGFileDC::DrawIcon**

**void DrawIcon(const wxIcon& *icon*, wxCoord *x*, wxCoord *y*)**<sup>K</sup>

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.

---

<sup>w</sup>xSVGFileDC::DrawIcon

<sup>w</sup>xdcdrawicon

<sup>b</sup>rowse00023

<sup>K</sup> wxSVGFileDC DrawIcon

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawIcon

\$#+K! **wxSVGFileDC::DrawLine**

**void DrawLine(wxCoord x1, wxCoord y1, wxCoord x2, wxCoord y2)<sup>K</sup>**

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

---

wxSVGFileDC::DrawLine

wxcdrawline

browse00024

K wxSVGFileDC DrawLine

EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

K DrawLine

**`wxSVGFileDC::DrawLines`**

**`void DrawLines(int n, wxPoint points[], wxCoord xoffset = 0, wxCoord yoffset = 0)`**<sup>K</sup>

**`void DrawLines(wxList *points, wxCoord xoffset = 0, wxCoord yoffset = 0)`**<sup>K</sup>

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.

---

<sup>w</sup>`wxSVGFileDC::DrawLines`

<sup>w</sup>`wxcdrawlines`

<sup>b</sup>`rowse00025`

<sup>K</sup> `wxSVGFileDC DrawLines`

<sup>E</sup>`nableButton("Up");ChangeButtonBinding("Up", "JumpId(svg.hlp', `wxSVGFileDC')")`

<sup>K</sup> `DrawLines`

<sup>K</sup> `DrawLines`

<sup>\$#+K!</sup>**wxSVGFileDC::DrawPolygon**

**void DrawPolygon(int *n*, wxPoint *points*[], wxCoord *xoffset* = 0, wxCoord *yoffset* = 0, int *fill\_style* = wxODDEVEN\_RULE)<sup>K</sup>**

**void DrawPolygon(wxList \**points*, wxCoord *xoffset* = 0, wxCoord *yoffset* = 0, int *fill\_style* = wxODDEVEN\_RULE)<sup>K</sup>**

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.

---

<sup>w</sup>xSVGFileDC::DrawPolygon

<sup>w</sup>xdcdrawpolygon

<sup>b</sup>rowse00026

<sup>K</sup> wxSVGFileDC DrawPolygon

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawPolygon

<sup>K</sup> DrawPolygon

\$#+K! **wxSVGFileDC::DrawPoint**

**void DrawPoint(wxCoord x, wxCoord y)**<sup>K</sup>

Draws a point using the current pen.

---

<sup>w</sup>xSVGFileDC::DrawPoint

<sup>w</sup>xdcdrawpoint

<sup>b</sup>rowse00027

<sup>K</sup> wxSVGFileDC DrawPoint

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawPoint

\$#+K! **wxSVGFileDC::DrawRectangle**

**void DrawRectangle**(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)<sup>K</sup>

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.

---

<sup>w</sup>xSVGFileDC::DrawRectangle

<sup>w</sup>xdcdrawrectangle

<sup>b</sup>rowse00028

<sup>K</sup> wxSVGFileDC DrawRectangle

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawRectangle

<sup>\$#+K!</sup>**wxSVGFileDC::DrawRotatedText**

**void DrawRotatedText(const wxString& *text*, wxCoord *x*, wxCoord *y*, double *angle*)**<sup>K</sup>

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

---

<sup>w</sup>xSVGFileDC::DrawRotatedText

<sup>w</sup>xdcdrawrotatedtext

<sup>b</sup>rowse00029

<sup>K</sup> wxSVGFileDC DrawRotatedText

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawRotatedText



<sup>S#+K!</sup>**wxSVGFileDC::DrawRoundedRectangle**

**void DrawRoundedRectangle**(wxCoord *x*, wxCoord *y*, wxCoord *width*, wxCoord *height*, double *radius* = 20)<sup>K</sup>

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.

---

<sup>w</sup>xSVGFileDC::DrawRoundedRectangle

<sup>w</sup>xcdrawroundedrectangle

<sup>b</sup>rowse00030

<sup>K</sup> wxSVGFileDC DrawRoundedRectangle

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawRoundedRectangle

**wxSVGFileDC::DrawSpline**

**void DrawSpline(wxList \*points)<sup>K</sup>**

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)<sup>K</sup>**

Draws a three-point spline using the current pen.

---

<sup>w</sup>wxSVGFileDC::DrawSpline

<sup>w</sup>xdcdrawspline

<sup>b</sup>rowse00031

<sup>K</sup> wxSVGFileDC DrawSpline

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawSpline

<sup>K</sup> DrawSpline

<sup>\$#+K!</sup>**wxSVGFileDC::DrawText**

**void DrawText(const wxString& *text*, wxCoord *x*, wxCoord *y*)**<sup>K</sup>

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](#) for how to get the dimensions of a text string, which can be used to position the text more precisely.

---

<sup>w</sup>xSVGFileDC::DrawText

<sup>w</sup>xdcdrawtext

<sup>b</sup>rowse00032

<sup>K</sup> wxSVGFileDC DrawText

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> DrawText

**wxSVGFileDC::EndDoc**

**void EndDoc()**

Does nothing

---

**wxSVGFileDC::EndDoc**

**void EndDoc()**

Does nothing

**wxSVGFileDC::EndDoc**

**void EndDoc()**

Does nothing

**wxSVGFileDC::EndDrawing**

**void EndDrawing()**

Does nothing

---

**wxSVGFileDC::EndDrawing**

**void EndDrawing**

Does nothing

**wxSVGFileDC::EndDrawing**

**void EndDrawing()**

Does nothing

**wxSVGFileDC::EndPage**

**void EndPage()**

Does nothing

---

**wxSVGFileDC::EndPage**

**void EndPage()**

Does nothing

**wxSVGFileDC::EndPage**

**void EndPage()**

Does nothing

`$#+K! wxSVGFileDC::FloodFill`

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

Not implemented

---

`wxSVGFileDC::FloodFill`

`wxdcfloodfill`

`browse00036`

`K wxSVGFileDC FloodFill`

`EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")`

`K FloodFill`

**wxSVGFileDC::GetBackground**

**wxBrush& GetBackground()**<sup>K</sup>

constfunccconst wxBrush&GetBackground

Gets the brush used for painting the background (see [wxSVGFileDC::SetBackground](#)).

---

<sup>w</sup>xSVGFileDC::GetBackground

<sup>w</sup>xdcgetbackground

<sup>b</sup>rowse00037

<sup>K</sup> wxSVGFileDC GetBackground

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

<sup>K</sup> GetBackground



**\$#+K! wxSVGFileDC::GetBackgroundMode**

constfuncintGetBackgroundMode

---

<sup>w</sup>xSVGFileDC::GetBackgroundMode

<sup>w</sup>xdcgetbackgroundmode

<sup>b</sup>rowse00038

<sup>K</sup> wxSVGFileDC GetBackgroundMode

<sup>E</sup>nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")





