#$+K!

Manual for wxSVGFileDC

by Chris Elliott

Contents

---

^Contents
^Contents
^browse00001
^K Contents
^DisableButton("Up")

^{$#+K!}**Copyright notice**

---

wxSVGFileDC

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://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

---

[w]xSVGFileDC
[w]xSVGFileDC
[b]rowse00004
[K] wxSVGFileDC
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `topic1')")

wxSVGFileDC::wxSVGFileDC

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

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

^topic2

^browse00005

^K wxSVGFileDC wxSVGFileDC

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

^K wxSVGFileDC

^K wxSVGFileDC

^K wxSVGFileDC

wxSVGFileDC::~wxSVGFileDC

 ~wxSVGFileDC()^K

Destructor.

ᵂxSVGFileDC::~wxSVGFileDC
ᵗopic3
ᵇrowse00006
ᴷ wxSVGFileDC ~wxSVGFileDC
ᴱnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
ᴷ ~wxSVGFileDC

$^{\#+K!}$wxSVGFileDC::BeginDrawing

Does nothing

---

^{w}xSVGFileDC::BeginDrawing
^{w}xdcbegindrawing
^{b}rowse00007
^{K} wxSVGFileDC BeginDrawing
^{E}nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

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)*^K

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.

---

^{$#+K!}**wxSVGFileDC::CalcBoundingBox**

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

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

---

wxSVGFileDC::Clear

void Clear()^K

This makes no sense in wxSVGFileDC and does nothing

ʷxSVGFileDC::Clear

ʷxdcclear

ᵇrowse00010

ᴷ wxSVGFileDC Clear

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

ᴷ Clear

$#+K!wxSVGFileDC::CrossHair

void CrossHair(wxCoord *x*, **wxCoord** *y*)[K]

Not Implemented

---

[w]xSVGFileDC::CrossHair
[w]xdccrosshair
[b]rowse00011
[K] wxSVGFileDC CrossHair
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] CrossHair

wxSVGFileDC::DestroyClippingRegion

void DestroyClippingRegion()^K

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*)^K

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

[$#+K!]wxSVGFileDC::DeviceToLogicalXRel

wxCoord DeviceToLogicalXRel(wxCoord *x*)[K]

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.

---

[w]xSVGFileDC::DeviceToLogicalXRel

[w]xdcdevicetologicalxrel

[b]rowse00014

[K] wxSVGFileDC DeviceToLogicalXRel

[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

[K] DeviceToLogicalXRel

wxSVGFileDC::DeviceToLogicalY

wxCoord DeviceToLogicalY(**wxCoord** *y*)^K

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

---

^{w}xSVGFileDC::DeviceToLogicalY
^{w}xdcdevicetologicaly
^{b}rowse00015
^{K} wxSVGFileDC DeviceToLogicalY
^{E}nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^{K} DeviceToLogicalY

$#+K!wxSVGFileDC::DeviceToLogicalYRel

wxCoord DeviceToLogicalYRel(wxCoord *y*)[K]

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.

---

[w]xSVGFileDC::DeviceToLogicalYRel
[w]dcdevicetologicalyrel
[b]rowse00016
[K] wxSVGFileDC DeviceToLogicalYRel
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DeviceToLogicalYRel

wxSVGFileDC::DrawArc

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

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::DrawArc
^wxdcdrawarc
^browse00017
^K wxSVGFileDC DrawArc
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawArc

^{$#+K!}**wxSVGFileDC::DrawBitmap**

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

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 <u>SetTextForeground</u>, <u>SetTextBackground</u> and <u>wxMemoryDC</u>.

---

^wxSVGFileDC::DrawBitmap
^wxdcdrawbitmap
^browse00018
^K wxSVGFileDC DrawBitmap
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawBitmap

wxSVGFileDC::DrawCheckMark

void DrawCheckMark(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)[K]

void DrawCheckMark(**const wxRect &***rect*)[K]

Draws a check mark inside the given rectangle.

---

[w]xSVGFileDC::DrawCheckMark

[w]xdcdrawcheckmark

[b]rowse00019

[K] wxSVGFileDC DrawCheckMark

[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

[K] DrawCheckMark

[K] DrawCheckMark

wxSVGFileDC::DrawCircle

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

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

Draws a circle with the given centre and radius.

wxheadingSee also

<u>DrawEllipse</u>

---

^wxSVGFileDC::DrawCircle
^wxdcdrawcircle
^browse00020
^K wxSVGFileDC DrawCircle
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawCircle
^K DrawCircle

wxSVGFileDC::DrawEllipse

void DrawEllipse(**wxCoord** *x*, **wxCoord** *y*, **wxCoord** *width*, **wxCoord** *height*)[K]

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

void DrawEllipse(**const wxRect&** *rect*)[K]

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.

wxheadingSee also

DrawCircle

---

[w]xSVGFileDC::DrawEllipse
[w]xdcdrawellipse
[b]rowse00021
[K] wxSVGFileDC DrawEllipse
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawEllipse
[K] DrawEllipse
[K] DrawEllipse

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.

---

[w]xSVGFileDC::DrawEllipticArc
[w]xdcdrawellipticarc
[b]rowse00022
[K] wxSVGFileDC DrawEllipticArc
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawEllipticArc

wxSVGFileDC::DrawIcon

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

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*)[K]

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

---

[w]xSVGFileDC::DrawLine

[w]xdcdrawline

[b]rowse00024

[K] wxSVGFileDC DrawLine

[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")

[K] DrawLine

wxSVGFileDC::DrawLines

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

void DrawLines(**wxList** **points,* **wxCoord** *xoffset = 0,* **wxCoord** *yoffset = 0*)^K

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::DrawLines
^wxdcdrawlines
^browse00025
^K wxSVGFileDC DrawLines
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawLines
^K DrawLines

wxSVGFileDC::DrawPolygon

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

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

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.

---

[w]xSVGFileDC::DrawPolygon
[w]xdcdrawpolygon
[b]rowse00026
[K] wxSVGFileDC DrawPolygon
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawPolygon
[K] DrawPolygon

^{$#+K!}**wxSVGFileDC::DrawPoint**

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

Draws a point using the current pen.

---

^wxSVGFileDC::DrawPoint
^wxdcdrawpoint
^browse00027
^K wxSVGFileDC DrawPoint
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawPoint

wxSVGFileDC::DrawRectangle

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

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::DrawRectangle
^wxdcdrawrectangle
^browse00028
^K wxSVGFileDC DrawRectangle
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K DrawRectangle

wxSVGFileDC::DrawRotatedText

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

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

---

[w]xSVGFileDC::DrawRotatedText
[w]xdcdrawrotatedtext
[b]rowse00029
[K] wxSVGFileDC DrawRotatedText
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawRotatedText

$#+K!wxSVGFileDC::DrawRoundedRectangle

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

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.

---

[w]xSVGFileDC::DrawRoundedRectangle
[w]xdcdrawroundedrectangle
[b]rowse00030
[K] wxSVGFileDC DrawRoundedRectangle
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawRoundedRectangle

wxSVGFileDC::DrawSpline

void DrawSpline(**wxList ****points*)^K

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*)^K

Draws a three-point spline using the current pen.

---

wxSVGFileDC::DrawText

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

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.

---

[w]xSVGFileDC::DrawText
[w]xdcdrawtext
[b]rowse00032
[K] wxSVGFileDC DrawText
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] DrawText

wxSVGFileDC::EndDoc

void EndDoc()^K

Does nothing

---

^wxSVGFileDC::EndDoc
^wxdcenddoc
^browse00033
^K wxSVGFileDC EndDoc
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K EndDoc

wxSVGFileDC::EndDrawing

void EndDrawing()^K

Does nothing

---

^wxSVGFileDC::EndDrawing
^wxdcenddrawing
^browse00034
^K wxSVGFileDC EndDrawing
^EnableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^K EndDrawing

wxSVGFileDC::EndPage

void EndPage()[K]

Does nothing

---

[w]xSVGFileDC::EndPage
[w]xdcendpage
[b]rowse00035
[K] wxSVGFileDC EndPage
[E]nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
[K] EndPage

$#+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()^K

constfuncconst wxBrush&GetBackground

Gets the brush used for painting the background (see <u>wxSVGFileDC::SetBackground</u>).

---

^{w}xSVGFileDC::GetBackground
^{w}xdcgetbackground
^{b}rowse00037
^{K} wxSVGFileDC GetBackground
^{E}nableButton("Up");ChangeButtonBinding("Up", "JumpId(`svg.hlp', `wxSVGFileDC')")
^{K} GetBackground

wxSVGFileDC::GetBackgroundMode

constfuncintGetBackgroundMode