

#+K!

Manual for wxSVGFileDC

by Chris Elliott

Contents

Copyright notice
wxSVGFileDC

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

Copyright notice

Copyright notice
topic0
rowse00002
Copyright notice
isableButton("Up")

\$#+K!wxSVGFileDC

wxSVGFileDC

wxSVGFileDC
topic1
browse00003
K wxSVGFileDC
DisableButton("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>

Derived from

wxDCBase

Include files

<wx/dcsvg.h>

See also

Members

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

^wxSVGFileDC

^wxSVGFileDC

^browse00004

^K wxSVGFileDC

^EnableButton("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
wxSVGFileDC::GetBrush
wxSVGFileDC::GetCharHeight
wxSVGFileDC::GetCharWidth
wxSVGFileDC::GetClippingBox
wxSVGFileDC::GetFont
wxSVGFileDC::GetLogicalFunction
wxSVGFileDC::GetMapMode
wxSVGFileDC::GetPen
wxSVGFileDC::GetPixel
wxSVGFileDC::GetSize
wxSVGFileDC::GetTextBackground
wxSVGFileDC::GetTextExtent
wxSVGFileDC::GetTextForeground
wxSVGFileDC::GetUserScale
wxSVGFileDC::LogicalToDeviceX
wxSVGFileDC::LogicalToDeviceXRel
wxSVGFileDC::LogicalToDeviceY
wxSVGFileDC::LogicalToDeviceYRel
wxSVGFileDC::MaxX
wxSVGFileDC::MaxY
wxSVGFileDC::MinX
wxSVGFileDC::MinY
wxSVGFileDC::Ok
wxSVGFileDC::ResetBoundingBox
wxSVGFileDC::SetAxisOrientation
wxSVGFileDC::SetDeviceOrigin
wxSVGFileDC::SetBackground
wxSVGFileDC::SetBackgroundMode
wxSVGFileDC::SetClippingRegion

wxSVGFileDC::SetPalette
wxSVGFileDC::SetBrush
wxSVGFileDC::SetFont
wxSVGFileDC::SetLogicalFunction
wxSVGFileDC::SetMapMode
wxSVGFileDC::SetPen
wxSVGFileDC::SetTextBackground
wxSVGFileDC::SetTextForeground
wxSVGFileDC::SetUserScale
wxSVGFileDC::StartDoc
wxSVGFileDC::StartPage

`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`

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

`~wxSVGFileDC()`^K

Destructor.

`wxSVGFileDC::~~wxSVGFileDC`

`topic3`

`browse00006`

`K wxSVGFileDC ~wxSVGFileDC`

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

`K ~wxSVGFileDC`

wxSVGFileDC::BeginDrawing

Does nothing

wxSVGFileDC::BeginDrawing

wxdcbegindrawing

rowse00007

wxSVGFileDC BeginDrawing

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

\$#+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)^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.

^wxSVGFileDC::Blit

^wxdcbrit

^browse00008

^K wxSVGFileDC Blit

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

^K Blit

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

void CalcBoundingBox(wxCoord x, wxCoord y)^K

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

^wxSVGFileDC::CalcBoundingBox

^wxdccalcboundingbox

^browse00009

^K wxSVGFileDC CalcBoundingBox

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

^K CalcBoundingBox

`$#+K!wxSVGFileDC::Clear`

`void Clear()`^K

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)K`

Not Implemented

`wxSVGFileDC::CrossHair`

`wxdccrosshair`

`browse00011`

`KwxSVGFileDC CrossHair`

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

`K CrossHair`

\$#+K! **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.

^w`wxSVGFileDC::DeviceToLogicalX`

^w`xdcdevicetologicalx`

^b`rowse00013`

^K`wxSVGFileDC DeviceToLogicalX`

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

^K`DeviceToLogicalX`

\$#+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.

^wwxSVGFileDC::DeviceToLogicalXRel

^wxdcdevicetologicalxrel

^browse00014

^K wxSVGFileDC DeviceToLogicalXRel

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

^K DeviceToLogicalXRel

`$#+K! wxSVGFileDC::DeviceToLogicalY`

`wxCoord DeviceToLogicalY(wxCoord y)K`

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

^wxSVGFileDC::DeviceToLogicalY

^wxdcdevicetologicaly

^browse00015

^K wxSVGFileDC DeviceToLogicalY

^EnableButton("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.

^wxSVGFileDC::DeviceToLogicalYRel

^wxdcdevicetologicalyrel

^browse00016

^K wxSVGFileDC DeviceToLogicalYRel

^EnableButton("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 SetTextForeground, SetTextBackground and wxMemoryDC.

^wwxSVGFileDC::DrawBitmap

^wxcdrawbitmap

^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`wxSVGFileDC::DrawCheckMark`

^w`xcdcdrawcheckmark`

^b`rowse00019`

^K `wxSVGFileDC DrawCheckMark`

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

^K `DrawCheckMark`

^K `DrawCheckMark`

\$#+K! **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.

See also

[DrawEllipse](#)

^w wxSVGFileDC::DrawCircle

^w xdcdrawcircle

^b rowse00020

^K wxSVGFileDC DrawCircle

^E nableButton("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.

See also

[DrawCircle](#)

^wwxSVGFileDC::DrawEllipse

^wxdcdrawellipse

^browse00021

^K wxSVGFileDC DrawEllipse

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

^K DrawEllipse

^K DrawEllipse

^K 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.

^w`wxSVGFileDC::DrawEllipticArc`

^w`xcdcdrawellipticarc`

^b`rowse00022`

^K`wxSVGFileDC DrawEllipticArc`

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

^K`DrawEllipticArc`

^{\$#+K!}**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::DrawIcon

^wxdcdrawicon

^browse00023

^K wxSVGFileDC DrawIcon

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

^K DrawIcon

\$#+K! **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.

^wwxSVGFileDC::DrawLine

^wxdcdrawline

^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)^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.

^wwxSVGFileDC::DrawLines

^wxdcdrawlines

^browse00025

^K wxSVGFileDC DrawLines

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

^K DrawLines

^K DrawLines

^{\$#+K!}**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.

^wxSVGFileDC::DrawPolygon

^wxdcdrawpolygon

^browse00026

^K wxSVGFileDC DrawPolygon

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

^K DrawPolygon

^K DrawPolygon

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

\$#+K! **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

wxcdrawrectangle

browse00028

K wxSVGFileDC DrawRectangle

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

K DrawRectangle

^{\$#+K!}**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

^wxSVGFileDC::DrawRotatedText

^wxdcdrawrotatedtext

^browse00029

^K wxSVGFileDC DrawRotatedText

^EnableButton("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.

^wxSVGFileDC::DrawRoundedRectangle

^wxcdrawroundedrectangle

^browse00030

^K wxSVGFileDC DrawRoundedRectangle

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

^K DrawRoundedRectangle

\$#+K! **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::DrawSpline

Wxdcdrawspline

browse00031

K wxSVGFileDC DrawSpline

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

K DrawSpline

K DrawSpline

^{\$#+K!}**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.

^wxSVGFileDC::DrawText

^wxcdrawtext

^browse00032

^K wxSVGFileDC DrawText

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

^K DrawText

wxSVGFileDC::EndDoc

void EndDoc()

Does nothing

wxSVGFileDC::EndDoc

wxdcenddoc

rowse00033

wxSVGFileDC EndDoc

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

EndDoc

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()^K

const wxBrush& GetBackground() const

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

^wxSVGFileDC::GetBackground

^wxdcgetbackground

^browse00037

^K wxSVGFileDC GetBackground

^K GetBackground

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

^K GetBackground

`$#+KK!` **wxSVGFileDC::GetBackgroundMode**

int GetBackgroundMode() const

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

See also

[SetBackgroundMode](#)

`wxSVGFileDC::GetBackgroundMode`

`wxdcgetbackgroundmode`

`rowse00038`

`K wxSVGFileDC GetBackgroundMode`

`K GetBackgroundMode`

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

\$#+KK! **wxSVGFileDC::GetBrush**

wxBrush& GetBrush()^K

const wxBrush& GetBrush() const

Gets the current brush (see [wxSVGFileDC::SetBrush](#)).

wxSVGFileDC::GetBrush

wxdcgetbrush

browse00039

K wxSVGFileDC GetBrush

K GetBrush

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

K GetBrush

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

wxCoord GetCharHeight()^K

Gets the character height of the currently set font.

wxSVGFileDC::GetCharHeight

wxdcgetcharheight

browse00040

K wxSVGFileDC GetCharHeight

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

K GetCharHeight

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

wxCoord GetCharWidth()^K

Gets the average character width of the currently set font.

wxSVGFileDC::GetCharWidth

wxdcgetcharwidth

browse00041

K wxSVGFileDC GetCharWidth

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

K GetCharWidth

^{\$#+K!}**wxSVGFileDC::GetClippingBox**

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

Not implemented

^wxSVGFileDC::GetClippingBox

^wxdcgetclippingbox

^browse00042

^K wxSVGFileDC GetClippingBox

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

^K GetClippingBox

wxSVGFileDC::GetFont

wxFont& GetFont()

const wxFont& GetFont() const

Gets the current font (see [wxSVGFileDC::SetFont](#)).

wxSVGFileDC::GetFont

wxdcgetfont

rowse00043

wxSVGFileDC GetFont

GetFont

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

GetFont

wxSVGFileDC::GetLogicalFunction

int GetLogicalFunction()^K

Gets the current logical function (see [wxSVGFileDC::SetLogicalFunction](#)).

^wxSVGFileDC::GetLogicalFunction

^wxdcgetlogicalfunction

^browse00044

^K wxSVGFileDC GetLogicalFunction

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

^K GetLogicalFunction

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

int GetMapMode()^K

Gets the *mapping mode* for the device context (see [wxSVGFileDC::SetMapMode](#)).

wxSVGFileDC::GetMapMode

wxdcgetmapmode

browse00045

K wxSVGFileDC GetMapMode

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

K GetMapMode

wxSVGFileDC::GetPen

wxPen& GetPen()

const wxPen& GetPen() const

Gets the current pen (see [wxSVGFileDC::SetPen](#)).

wxSVGFileDC::GetPen

wxdcgetpen

rowse00046

wxSVGFileDC GetPen

GetPen

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

GetPen

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

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

Not implemented

^wxSVGFileDC::GetPixel

^wxdcgetpixel

^browse00047

^K wxSVGFileDC GetPixel

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

^K GetPixel

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

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

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

^wxSVGFileDC::GetSize

^wxdcgetsize

^browse00048

^K wxSVGFileDC GetSize

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

^K GetSize

`$#+KK! wxSVGFileDC::GetTextBackground`

`wxColour& GetTextBackground()`^K

`const wxColour& GetTextBackground() const`

Gets the current text background colour (see [wxSVGFileDC::SetTextBackground](#)).

^w`xSVGFileDC::GetTextBackground`

^w`xdcgettextbackground`

^b`rowse00049`

^K `wxSVGFileDC GetTextBackground`

^K `GetTextBackground`

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

^K `GetTextBackground`

^{\$#+K!}**wxSVGFileDC::GetTextExtent**

**void GetTextExtent(const wxString& *string*, wxCoord **w*, wxCoord **h*,
wxCoord **descent* = NULL, wxCoord **externalLeading* = NULL, wxFont **font* =
NULL)^K**

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](#).

^wxSVGFileDC::GetTextExtent

^wxdcgettexttext

^browse00050

^K wxSVGFileDC GetTextExtent

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

^K GetTextExtent

wxSVGFileDC::GetTextForeground

wxColour& GetTextForeground()

const wxColour& GetTextForeground() const

Gets the current text foreground colour (see [wxSVGFileDC::SetTextForeground](#)).

wxSVGFileDC::GetTextForeground

wxdcgettextforeground

rowse00051

wxSVGFileDC GetTextForeground

GetTextForeground

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

GetTextForeground

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

void GetUserScale(double *x, double *y)^K

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

wxSVGFileDC::GetUserScale

wxdcgetuserscale

browse00052

K wxSVGFileDC GetUserScale

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

K GetUserScale

`wxSVGFileDC::LogicalToDeviceX`

`wxCoord LogicalToDeviceX(wxCoord x)`^K

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

^w`xSVGFileDC::LogicalToDeviceX`

^w`xdclogicaltodevicex`

^b`rowse00053`

^K `wxSVGFileDC LogicalToDeviceX`

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

^K `LogicalToDeviceX`

^{\$#+K!}**wxSVGFileDC::LogicalToDeviceXRel**

wxCoord LogicalToDeviceXRel(wxCoord x)^K

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::LogicalToDeviceXRel

^wxdclogicaltodevicexrel

^browse00054

^K wxSVGFileDC LogicalToDeviceXRel

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

^K LogicalToDeviceXRel

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

wxCoord LogicalToDeviceY(wxCoord y)^K

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

^wxSVGFileDC::LogicalToDeviceY

^wxdclogicaltodevicey

^browse00055

^K wxSVGFileDC LogicalToDeviceY

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

^K LogicalToDeviceY

^{\$#+K!}**wxSVGFileDC::LogicalToDeviceYRel**

wxCoord LogicalToDeviceYRel(wxCoord y)^K

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::LogicalToDeviceYRel

^wxdclogicaltodeviceyrel

^browse00056

^K wxSVGFileDC LogicalToDeviceYRel

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

^K LogicalToDeviceYRel

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

wxCoord MaxX()^K

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

wxSVGFileDC::MaxX

wxdcmaxx

browse00057

K wxSVGFileDC MaxX

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

K MaxX

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

wxCoord MaxY()^K

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

wxSVGFileDC::MaxY

wxdcmaxy

browse00058

K wxSVGFileDC MaxY

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

K MaxY

`$#+K!wxSVGFileDC::MinX`

`wxCoord MinX()`^K

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

`wxSVGFileDC::MinX`

`wxdcminx`

`rowse00059`

`K wxSVGFileDC MinX`

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

`K MinX`

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

wxCoord MinY()^K

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

wxSVGFileDC::MinY

wxdcminy

browse00060

K wxSVGFileDC MinY

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

K MinY

`$#+K!wxSVGFileDC::Ok`

`bool Ok()`^K

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

`wxSVGFileDC::Ok`

`wxdcok`

`browse00061`

`K wxSVGFileDC Ok`

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

`K Ok`

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

void ResetBoundingBox()^K

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

See also

[CalcBoundingBox](#)

wxSVGFileDC::ResetBoundingBox

wxdcresetboundingbox

browse00062

K wxSVGFileDC ResetBoundingBox

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

K ResetBoundingBox

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

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

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.

^wwxSVGFileDC::SetAxisOrientation

^wwdcsetaxisorientation

^browse00063

^KwxSVGFileDC SetAxisOrientation

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

^KSetAxisOrientation

^{\$#+K!}**wxSVGFileDC::SetDeviceOrigin**

void SetDeviceOrigin(wxCoord x, wxCoord y)^K

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::SetDeviceOrigin

^wxdcsetdeviceorigin

^browse00064

^K wxSVGFileDC SetDeviceOrigin

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

^K SetDeviceOrigin

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

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

Sets the current background brush for the DC.

wxSVGFileDC::SetBackground

wxdcsetbackground

browse00065

K wxSVGFileDC SetBackground

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

K SetBackground

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

void SetBackgroundMode(int *mode*)^K

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

^wxSVGFileDC::SetBackgroundMode

^wxdcsetbackgroundmode

^browse00066

^K wxSVGFileDC SetBackgroundMode

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

^K SetBackgroundMode

`$#+K! wxSVGFileDC::SetClippingRegion`

`void SetClippingRegion(wxCoord x, wxCoord y, wxCoord width, wxCoord height)K`

`void SetClippingRegion(const wxPoint& pt, const wxSize& sz)K`

`void SetClippingRegion(const wxRect& rect)K`

`void SetClippingRegion(const wxRegion& region)K`

Not implemented

^w`wxSVGFileDC::SetClippingRegion`

^w`xdcsetclippingregion`

^b`rowse00067`

^K `wxSVGFileDC SetClippingRegion`

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

^K `SetClippingRegion`

^K `SetClippingRegion`

^K `SetClippingRegion`

^K `SetClippingRegion`

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

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

Not implemented

^wxSVGFileDC::SetPalette

^wxdcsetpalette

^browse00068

^K wxSVGFileDC SetPalette

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

^K SetPalette

^{\$#+K!}**wxSVGFileDC::SetBrush**

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

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::SetBrush

^wxdcsetbrush

^browse00069

^K wxSVGFileDC SetBrush

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

^K SetBrush

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

void SetFont(const wxFont& font)^K

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::SetFont

wxdcsetfont

browse00070

K wxSVGFileDC SetFont

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

K SetFont

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

void SetLogicalFunction(int *function*)^K

Only wxCOPY is available; trying to set one of the other values will fail

wxSVGFileDC::SetLogicalFunction

wxdcsetlogicalfunction

browse00071

K wxSVGFileDC SetLogicalFunction

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

K SetLogicalFunction

^{\$#+K!}**wxSVGFileDC::SetMapMode**

void SetMapMode(int *int*)^K

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](#)) 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::SetMapMode

^wxdcsetmapmode

^browse00072

^K wxSVGFileDC SetMapMode

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

^K SetMapMode

^{\$#+K!}**wxSVGFileDC::SetPen**

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

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::SetPen

^wxdcsetpen

^browse00073

^K wxSVGFileDC SetPen

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

^K SetPen

^{\$#+K!}**wxSVGFileDC::SetTextBackground**

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

Sets the current text background colour for the DC.

^wxSVGFileDC::SetTextBackground

^wxdcsettextbackground

^browse00074

^K wxSVGFileDC SetTextBackground

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

^K SetTextBackground

^{\$#+K!}**wxSVGFileDC::SetTextForeground**

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

Sets the current text foreground colour for the DC.

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

^wxSVGFileDC::SetTextForeground

^wxdcsettextforeground

^browse00075

^K wxSVGFileDC SetTextForeground

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

^K SetTextForeground

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

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

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

^wwxSVGFileDC::SetUserScale

^wxdcsetuserscale

^browse00076

^K wxSVGFileDC SetUserScale

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

^K SetUserScale

`$#+K!wxSVGFileDC::StartDoc`

`bool StartDoc(const wxString& message)K`

Does nothing

`wxSVGFileDC::StartDoc`

`wxdcstartdoc`

`browse00077`

`K wxSVGFileDC StartDoc`

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

`K StartDoc`

wxSVGFileDC::StartPage

bool StartPage()

Does nothing

wxSVGFileDC::StartPage

wxdcstartpage

rowse00078

wxSVGFileDC StartPage

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

StartPage

