

Specification for the Simple Vector Format (SVF) v1.1

- [Overview](#)
- [The file intro](#)
- [Drawing lines](#)
- [Defaults](#)
- [Image Extents](#)
- [Polylines](#)
- [Relative lines and polylines](#)
- [Circles and arcs](#)
- [Text](#)
- [Points and rectangles](#)
- [Bezier curves](#)
- [Colors](#)
- [Pen width](#)
- [Filled regions](#)
- [Hyperlinks](#)
- [Layers](#)
- [Notifications](#)
- [MIME type](#)
- [Miscellaneous](#)

Overview

SVF is designed to be a simple format for describing vector images. The basic drawing objects include [points](#), [lines](#), [circles](#), [arcs](#), [bezier curves](#) and [text](#). Features of the format include [layers](#) (for controlling the visibility of objects), [hyperlinks](#) (for allowing the user to click on a portion of the drawing to perform an action), [notifications](#) (for sending messages when the user has passed a certain zoom level), [fills](#), and the ability to override the default [colors](#).

The file is broken into 3 sections, the [intro](#), the header, and the main body. The intro is simply a text string identifying the file as SVF. The header includes general information useful for displaying or manipulating the image like layers, extents, and colors. The main body describes how to draw the image and hyperlinks.

Objects are drawn in the order they occur in the file, so if you wish to draw overlapping objects the latter object will be drawn on top of previous objects.

Command parameters are integers (coordinates, colors, etc.). Most command parameters can be different lengths (1 to 8 bytes with the 1 and 2 byte versions currently implemented and the 8 byte floating point version implemented for scale and base point), allowing the programmer to create more efficient files. The two most significant bits of the command byte determine the length (00 - 1, 01 - 2, 10 - 4, 11 - 8). So, for example, the 1 byte form of the Point command is hex 10 and the 2 byte form is hex 90. Numbers are stored bigendian.

As with most vector formats, the origin (0,0) is in the lower left corner with increasing x coordinates moving to the right and increasing y coordinates moving up. Coordinates are specified as positive integers; offsets can also be negative.

The programming examples reference code from [svfout.c](#) (with prototypes included in [svfout.h](#) and defines in [svfcom.h](#)) which handles outputting the commands as described in the [syntax](#) document. These functions automatically handle outputting the appropriate form of the command based on the required length. Note that 'fp' is a file pointer (or reference).

The file intro

The file starts out with text which describes the file as SVF and lists a version number. The following code accomplishes this and then closes the file when the file is finished. Note that SVFOpen writes a null-terminated text string (e.g. "SVFv1.1") to begin the file and SVFClose simply closes the file.

```
FILE *fp = SVFOpen("name.svf");  
...  
SVFClose(fp);
```

Drawing lines

The simplest way to draw lines is with the MoveTo and LineTo commands. MoveTo moves the current point to a new location without drawing. It is used to specify the starting point for most drawing commands. LineTo draws a line from the current point to the specified position, updating the current point in the process.

The following code draws a line from (10,10) to (50,30).

```
SVFOutputMoveTo(fp,10,10);  
SVFOutputLineTo(fp,50,30);
```

These two lines combined with the two lines listed in the file [intro section](#) would create a valid SVF file which simply drew a line.

Defaults

Most settings have defaults. The default drawing color is white (color index 255); the default background color is black (color index 0). The current point is initially at (0,0). The current layer is 0. Text height starts out at 10. Pen width defaults to 0. Fill mode is off. Default extents are (0,0)-(255,255).

Image extents

Since the size of the image can change, you may need to include the image's size (or extents) so the program viewing it will know how to display it initially. The default is (0,0)-(255,255). Extents information is required if your extents differ from the default. For example, if your image was entirely contained in the square (0,0)-(5000,5000), the following code would convey this.

```
SVFOutputExtents(fp,0,0,5000,5000);
```

Polylines

Polylines are a more efficient method for drawing a series of connected lines. Note that polylines start drawing from the current point. The value for the number of points allowed in the polyline takes up the same number of bytes as the coordinate values (limit 255 for 1 byte coordinate values, etc.).

Unlike the other convenience functions, polylines require that the programmer decide whether to use the short form (0-255) or long form (0-65535) of the command. Each full polyline command consists of the command describing how many points will be specified followed by each of the points. The final parameter specifies how many bytes each value requires. A polyline displayed with [fill mode](#) on is displayed as a filled polygon. The final

line segment is always an implicit line from the first point (current pen position) to the last point. Polyline [link regions](#) follow the same rules as drawing a polygon.

```
SVFOutputMoveTo(fp,100,200);
SVFOutputPolylineStart(fp,3,2); /* 3 pts in polyline */
SVFOutputPolyPoint(fp,200,100,2);
SVFOutputPolyPoint(fp,300,200,2);
SVFOutputPolyPoint(fp,200,300,2);
```

Relative lines and polylines

MoveTo, LineTo, and Polyline also have a relative form, where each point describes an offset from the current point. The offsets can be positive or negative. Thus the previous polyline example could be rewritten using the 1 byte form of relative polylines.

```
SVFOutputMoveTo(fp,100,200);
SVFOutputRelPolylineStart(fp,3,1);
SVFOutputPolyPoint(fp,100,-100,1);
SVFOutputPolyPoint(fp,100,100,1);
SVFOutputPolyPoint(fp,-100,100,1);
```

Circles and arcs

Circles and arcs are drawn with their center at the current point A [filled](#) arc looks like a piece of pie while a plain arc just consists of the curved arc segment. Arc [link regions](#) also behave like a piece of pie.

The following example draws a circle with radius 20 with a center at (10,10).

```
SVFOutputMoveTo(fp,10,10);
SVFOutputCircle(fp,20);
```

Arcs are specified by radius and start and end angles, the arc being drawn counter clockwise. The 1 byte form of the arc command describes the angles ranging from 0 to 255 in 255/360 degree increments. The 2 byte form angles range from 0 to 65535 in 65535/360 degree increments.

This code draws an arc centered at (10,10) with a radius of 20 starting at 30/360 degrees sweeping counterclockwise to 240/360 degrees.

```
SVFOutputMoveTo(fp,10,10);
SVFOutputArc(fp,20,30,240);
```

A problem with the arc convenience function comes from the fact that most of the functions automatically determine the number of bytes used by its parameters. If you wanted an arc of radius 200 starting at 10 out of 65535 angle units and continuing to 243 out of 65535 angle units, the command would assume that since 200, 10, and 243 are all under 256, the 1 byte form should be used when the 2 byte form was actually required. To circumvent this problem, you can use SVFOutputArcLength to force a different version of the command.

Text

Text can be displayed using a default system dependent courier-like font. Text is drawn using the current text height (default is 10). A width can also be specified with the text. If this value is 0, the text will be drawn using the font's default width. If a width is specified, the text will be scaled to fit. This is suggested if the width of the text is important because the font metrics may differ between systems. The current point is used for the baseline of the text. Descenders may go below this baseline. Text in the SVF file is null-terminated.

```
SVFOutputTextHeight(fp,100);
SVFOutputMoveTo(fp,10,10);
```

```
SVFOutputText(fp,0,"Hello world");      /* text width 0 */
```

Points and rectangles

You can draw points. You can also draw rectangle outlines or filled areas (depending on the fill mode) which start at the current point. Rectangles do not update the current point.

```
SVFOutputPoint(fp,50,50);
SVFOutputMoveTo(fp,10,10);
SVFOutputRectangle(fp,100,100);
```

Bezier curves

Cubic Bezier curves are supported. The first end point is the current point. The command specifies two control points followed by the second end point. This command updates the current point. Note that Bezier curves do not go through the control points. For [link regions](#) and [filled regions](#), a Bezier curve region is defined as the area between the curve and a line connecting its two end points.

The following code draws a cubic bezier curve from (10,10) to (240,20) using the control points (105,110) and (200,100).

```
SVFOutputMoveTo(fp,10,10);
SVFOutputBezier(fp,105,110,200,100,240,20);
```

Colors

There is a default color table consisting of 256 colors which will be reasonable for most images. Alternately, you can specify your own color table.

If you work with the default colors, the following constants can be used for the basic colors:

SVFC_Black	0
SVFC_White	255
SVFC_LightGray	219
SVFC_Gray	146
SVFC_DarkGray	73
SVFC_Red	224
SVFC_Green	28
SVFC_Blue	3
SVFC_Yellow	252
SVFC_Cyan	31
SVFC_Magenta	227

```
SVFOutputSetColor(fp,SVFC_Yellow);
```

The color definition can be thought of as the byte rrrgggbb where each color specification (red, green, or blue) is the most significant bits of the desired intensity. For example, if you want the RGB value 34,200,145 (00100010, 11001000, 10010001 in binary), where each value ranges from 0 (black) to 255 (pure color) the appropriate color index would be 00111010 in binary, or 58 in decimal.

You can use SVFGetClosestColor to turn an RGB value into an index referencing the default color table.

SVFOutputSetColor sets the current color for drawing objects until the next color change.

SVFOutputBackground defines which color is used for the background. The default background color is black (0) and the default drawing color is white (255). SVFOutputTransparent sets a color to be transparent (objects drawn with that color are drawn with the background color of the target window). This overrides the definition

for that color in the color table. Note that these commands use a color index into the image's color table. There can only be a single background color and a single transparent color.

See [svfxampl.c](#) for an example of how to output your own color table.

In order to define an object which doesn't appear (useful for hyperlink regions which don't display), you can set the current color to invisible. Objects are not drawn until a non-invisible color is set.

```
SVFOutputInvisible(fp);
```

Pen width

By default, the pen width is 0, which means that all lines will be drawn 1 pixel wide. Alternately a pen width (in SVF coordinates) can be specified. Note that filled regions and link regions are unaffected by pen width.

```
SVFOutputSetPenWidth(fp,100);
```

Filled regions

To display filled regions, you can turn the fill mode on (the default is off). This affects [polylines](#), [rectangles](#), [circles](#), [arcs](#), and [bezier curves](#). All regions are displayed filled until the fill mode is turned off.

```
SVFOutputFillMode(fp,SVF_ON);
SVFOutputMoveTo(fp,100,100);
SVFOutputCircle(fp,50);          /* circle is filled */
SVFOutputRectangle(fp,30,20);    /* rectangle is filled */
SVFOutputFillMode(fp,SVF_OFF);
SVFOutputRectangle(fp,20,30);    /* only outline is drawn */
```

Hyperlinks

Hyperlinks are regions defined in the image where the user can click in order to perform an action, similar to clicking on links in a Web document to bring up different information. Hyperlinks are enabled or disabled based on the layer they are on.

Links are defined in the main body. You set whether the following link regions will be 1 dimension or 2 dimension, the link action, and optional explanatory text. The link regions are displayed like any other entity unless the current color is invisible. On the Web, the action would be a URL which will be activated by the link. If this text is an empty string, the associated link is inactive. The explanatory text may be used by a browser to give the user more information about the link. If the associated layer is turned off, the link is deactivated.

A 1 dimensional link means that the user would select the line or curve. A 2 dimensional link is chosen by picking inside the region (as for a circle or polygon region). A point link is selected by picking close to the point. Text links are defined as the bounding box around the text.

Note that after a call to SVFOutput1dLink or SVFOutput2dLink, all following objects are links using the specified action until another link definition or an SVFOutputEndLink is found. SVFOutputEndLink is actually a call to SVFOutput1dLink with two NULL parameters.

```
SVFOutput2dLink(fp,"http://www.university.edu/",NULL);
SVFOutputMoveTo(fp,10,10);      /* triangle link region */
SVFOutputPolylineStart(fp,2,2);
SVFOutputPolyPoint(fp,500,500);
SVFOutputPolyPoint(fp,250,10);

SVFOutput1dLink(fp,"URL","select the circle outline");
SVFOutputMoveTo(fp,50,50);
```

```
SVFOutputCircle(fp,20);
```

```
SVFOutputEndLink(fp);
```

The [link syntax](#) allows links to activate a URL to bring up another document or toggle layers.

Link regions are checked in the reverse order they are defined so the image display matches how links are activated. If regions overlap, the first region selected will be the activated link.

Layers

Layers give you the ability to group objects together so the user can turn on or off the display of portions of the drawing. For example, if you drew a map with coastlines, rivers, and cities, you may want to allow the user to turn on and off the display of the city names so they can more easily view the information they are interested in. Each of these groups could be placed on separate layers. Hyperlinks are enabled or disabled based on the layer they are on.

The first step in using layers is to specify a layer table in the header section. This is simply a list of layer names and the initial display status (on or off) of each layer. The following example creates 3 layers called Coastline, Rivers, and Cities. If SVF_ON and SVF_OFF are constants representing 1 and 0 respectively, all objects on the first two layers would initially be displayed, but objects on the Cities layer would not be displayed. These settings can be changed when the user views the image.

```
SVFOutputLayerTable(fp,3);      /* 3 entries */
SVFOutputLayerEntry(fp,SVF_ON,"Coastline");
SVFOutputLayerEntry(fp,SVF_ON,"Rivers");
SVFOutputLayerEntry(fp,SVF_OFF,"Cities");
```

In the main body you can switch layers as often as you like. After you switch layers, all subsequent objects are part of that layer until you switch layers again.

```
SVFOutputSetLayer(fp,0);        /* Coastline layer... */
SVFOutputMoveTo(fp,10,10);
SVFOutputLineTo(fp,50,50);
SVFOutputSetLayer(fp,2);        /* Cities layer... */
SVFOutputLineTo(fp,30,50);
```

Note that the status of layers has no affect on commands which change settings: SetColor, SetLayer, TextHeight, and FillMode. The current point is updated by the appropriate commands regardless of whether the object is on a layer which is displayed or not.

Notifications

With notifications, actions can be performed when a certain magnification has been reached. These actions are the same as [hyperlink](#) actions and have the same [syntax](#). So, for example, when the user has zoomed in a ways, [layers](#) could be turned on so extra information could be displayed. Or information describing the current view can be passed back so a new image could be sent based on where the user was examining in the original image.

A notification can be triggered by zooming in or zooming out. This is specified by the width of the view. If, for example, a file was 1000 units on a side and you wanted to turn on the "Cities" layer after the current view was less than or equal to 300 units on a side, and have the layer turned back off if they zoomed out again, the following code would accomplish this:

```
SVFOutputNotifyTable(fp,1,1);   /* 1 zoomout and 1 zoomin */
/* first entry - notify when view <= 300 units wide */
SVFOutputNotifyEntry(fp,"layeron=Cities",300);
/* second entry - notify when view >= 300 units wide */
SVFOutputNotifyEntry(fp,"layeroff=Cities",300);
```

Note that when you initiate the notify table, the two numbers you pass indicate the number of entries in the zoomin portion of the table and the the zoomout portion respectively.

Notifications get executed before the image is redisplayed. If more than one notification gets triggered, the first one defined in the table is called first, then the others in order.

MIME type

The MIME type for SVF files is *vector/x-svf*. The standard file extension is .SVF or .svf.

Miscellaneous information

An image can contain a suggestion for how big it would like to appear. You can specify the suggested width of the image in millimeters. The display program is not required to honor the request. The SVF Plug-in for Netscape currently ignores this value.

```
SVFOutputWidth(fp,100); /* image should be 100 mm wide */
```

An SVF file can contain other data which isn't necessarily picked up by a viewer so extra information can be embedded in the file. A name can be added in the header section. Arbitrary data can be stored in the main body of the file.

```
SVFOutputName(fp,"some text");  
SVFOutputData(fp,20,ptr);      /* output 20 bytes of data */
```

Scale and base point can be added to the image. This information is not used for display. It would be used if the original coordinate system needed to be recreated from the integer values stored in the file. An SVF coordinate pair would first be added to the base point, then each value would be multiplied by the scale factor. The base point corresponds to the origin (0,0) of the SVF file. This command is the only one that currently uses the 8 byte floating point version of the commands.

```
SVFOutputTransform(fp,3.214,1.92,-4.3); /* scale, basex, basey */
```

A flag can be specified which indicates that a display program is free to rearrange entities within a layer. This is useful for certain optimizations. Note that it is still valid to display entities in the order they occur in the file. 0 is the default. 1 indicates you can rearrange within a layer.

```
SVFOutputSortByLayer(fp,1);
```

Related pages...

📄 [SVF Technical Information](#)

📄 [SVF Syntax](#)

📄 [HTML Syntax](#) for hyperlinks and embedding SVF images in HTML documents

📄 [Return to Softsource Home Page](#)

scotts@softsource.com