# Appendix SVG Profile (normative)

## Introduction

This appendix describes the subset of SVG elements, attributes, and properties which may be used within an S-100 compliant SVG document.

The goal is to limit the effort for an OEM to render the symbols on the one hand and give the symbol designers enough flexibility to use the common design techniques to create the symbols on the other hand. It should be possible to use SVG editors as they are available on the market.

The S-100 SVG profile is a subset of the SVG Tiny 1.2 profile

<http://www.w3.org/TR/SVGTiny12/>

## Document structure

### The top-level element

SVG documents that comply with this specification must have an <svg> element as the root element.

This is the only <svg> element allowed in the document.

The following attributes can be used on the <svg> elements to describe the properties of the SVG symbol.

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Description** | **Use** | **Remarks** |
| xmlns | The namespace URI of SVG | required | Fixed to:  http://www.w3.org/2000/svg |
| version | The language version of SVG | required | Fixed to: 1.2 |
| baseProfile | The base profile used | required | Fixed to: tiny |
| xml:space | Defines the space handling | optional | Default: preserve |
| width | The intrinsic width of the symbol | required | Must be given in mm |
| height | The intrinsic height of the symbol | required | Must be given in mm |
| viewBox | Defines the user coordinate system and the range of allowed coordinates | required | See the section on Coordinate System for more details |

The presentation attributes can be used at the <svg> element as well as they can be used on any element in an SVG file. See the section on styling properties for a list of possible attributes.

Example:

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny" xml:space="preserve" width="4.34mm" height="5.35mm" viewBox="-2.22 -2.79 4.34 5.35">

### Supported elements

The following elements are supported to comply with this specification:

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Multiplicity** | **Description** | **Remarks** |
| <title> | 0..1 | The name of the symbol | See the example below |
| <desc> | 0..1 | The brief textual description of the symbol | See the example below |
| <metadata> | 0..1 | Metadata according to the S-100 SVG Metadata schema | See the section on metadata. |
| <g> | 0..\* | Container element | Can have all the following elements and other <g> elements as children |
| <rect> | 0..\* | Shape element for rectangle | See the section on basic shapes for more details. |
| <circle> | 0..\* | Shape element for circle |
| <ellipse> | 0..\* | Shape element for ellipse |
| <line> | 0..\* | Shape element for a straight-line segment |
| <polyline> | 0..\* | Shape for a set of straight-line connected segments |
| <polygon> | 0..\* | Shape that describes a closed set of connected straight-line segments |
| <path> | 0..\* | Outline of a shape that can be filled or stroked. The outline can be curved | See the section on path commands for more details. |

Example of a <title> element:

<title>ACHARE02</title>

Example of a <desc> element:

<desc>anchorage area as a point at small scale, or anchor points of mooring trot at large scale</desc>

### Metadata

SVG has a metadata element which allows for the direct inclusion of metadata document fragments from other namespaces.

For this profile the element is defined in a schema assigned to the namespace:

<http://www.iho.int/SVGMetadata/5.1>

It defines an element <S100SVG> with the following attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Use | Type | Description |
| publisher | required | xs:string | The name of the publisher |
| creationDate | required | xs:date | The date of creation |
| source | optional | xs:string | The source of the symbol (e.g. S-52) |
| format | optional | xs:string | The format used |
| version | required | xs:decimal | The version of this symbol |

Example:

<metadata>

<iho:S100SVG xmlns:iho="http://www.iho.int/SVGMetadata/5.1">

<iho:Description iho:publisher="IHO" iho:creationDate="2014-06-09" iho:source="S52Preslib4.0" iho:format="S100SVG" iho:version="0.1"/>

</iho:S100SVG>

</metadata>

## Coordinates

### The initial user coordinate system

The initial user coordinate system is defined by the *width*, *height*, and *viewport* attributes of the <svg> element. All coordinates used in the document are based on this coordinate system. Note that within the document the user coordinate system can be modified with the transform attribute. See the section on transformation for the details.

The viewport of the symbol is defined with the attributes *width* and *height* at the <svg> element. The values must be given in millimetres.

Example:

<svg … width="4.34mm" height="5.35mm" …>

The *viewbox* attribute defines which range of coordinates the user can use in that viewport. It must be formatted as a list of four numbers (min-x min-y width height), separated by white space and/or a comma.

The coordinate system used for SVG has the x-axis pointing to the right and the y-axis pointing down.

The pivot point of the symbol is always at the 0,0 position.

Example:

<svg width="4mm" height="5.5mm" viewBox="0 0 4 5.5" …>

Defines a physical viewport of 4x5.5 millimetres and maps it to a coordinate space from 0.0 to 4.0 for the x-coordinates and from 0.0 to 5.5 for the y-coordinates. The coordinate (0, 0) – which defines the pivot point - is at the upper left corner of the symbol.

Each unit in the initial user coordinate system is equivalent to 1 millimetre.

Another example:

<svg width="10mm" height="10mm" viewBox="-50 -50 100 100" …>

Here the x-coordinates can range from -50 to +50 and the y-coordinates can range from -50 to +50 to span the viewport of 10mm for both dimensions. The pivot point is located in the centre of the area and each user defined unit corresponds to a tenth of a millimetre.

The mapping of that coordinate range to the physical dimension takes the aspect ratio into account. The *preserveAspectRatio* attribute is not supported by this profile; thus its default value must be used by any renderer. This value is: “xMidYMid”.

It is recommended that the ratio of the *width* and *height* components of the viewbox is identical to the ratio of the *width* and *height* of the viewport. Otherwise, the viewbox will not fill the entire viewport. Both view box and physical space will be aligned at their centres and the viewbox is uniformly scaled so that it fits completely inside the viewport.

To summarize:

* *The width* and *height* of the viewport must be defined in millimetres.
* It is recommended that the width and height of the viewbox have the same ratio as the width and height of the viewport.
* The min-x and min-y coordinates of the viewbox must be chosen in a way that the pivot point (0, 0) is situated at the corresponding part of the symbol’s graphic.

### Transformations

A new user coordinate system can be established by specifying transformations in the form of a 'transform' attribute on a container or a graphics element.

The transform attribute can have either the value ‘none’ or a list of affine transformations.

If ‘none’ is specified, the identity transformation is defined. This is identical to not specifying the attribute at all.

The transformation list consists of single transformation elements separated by white spaces and/or a comma.

The transformation commands are:

1. **Translation:**  translate(tx [ty])  
     
   A translation by tx and ty. If ty is not given it is assumed to be 0.  
   Matrix:   
   Examples: translate(20 30)
2. **Scaling:** scale(sx [sy])  
   A scale operation by sx and sy. If sy is not given it is assumed to be equal to sx.  
   Matrix:   
   Examples: scale(3 2) scale(0.5)
3. **Rotation:** rotate(α [cx cy])  
     
   A rotation by α degrees around the point (cx, cy)  
   Matrix:   
   Examples: rotate(30) rotate(-45 120 30)
4. **Skewing:** skewX(a) and skewY(a)  
     
   A skew transformation along the x-axis or the y-axis.Matrix: for skewing along the x-axis or along the y-axis  
   Examples: skewX(20) skewY(15)
5. **Matrix:** matrix(a b c d e f)  
     
   A generic affine transformation  
   Matrix:

The coordinate transformation is defined by:

Transformation can be nested by multiplying the corresponding matrices.

All defined transformations define the Current Transformation Matrix (CTM)

That leads to the following equation for transforming user coordinates to viewport coordinates.

## Styling

### Styling-properties

The following styling properties are included in this profile. Renderers must support them to be compliant with the profile.

For each styling property defined in this specification, there is a corresponding XML attribute (the presentation attribute) with the same name that is available on all relevant SVG elements.

For example, SVG has a 'fill' property that defines how to paint the interior of a shape. There is a corresponding presentation attribute with the same name (i.e., 'fill') that can be used to specify a value for the 'fill' property on a given element.

The following example shows how the 'fill' and 'stroke' properties can be assigned to a rectangle using the 'fill' and 'stroke' presentation attributes. The rectangle will be filled with red and outlined with blue:

<?xml version="1.0"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="10mm" height="5mm" viewBox="0 0 1000 500">

<rect x="200" y="100" width="600" height="300" fill="red" stroke="blue" stroke-width="3"/>

</svg>

All presentation attributes can be used at the <svg> element, the <g> element, and on any shape element used in the SVG file. If it is used at the <svg> element or the <g> element the values will be inherited to all child elements of these elements. If defined at a child element this value overrides the value from the container element.

The following table describes the style properties supported by this profile.

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Initial value | Description | Remarks |
| fill | black | Describes the colour that is used to fill the interior of a shape. The value ‘none’ indicates the shape should be not filled. | Applies to all shapes but has no effect for <line>  See notes on colours below. |
| fill-opacity | 1 | Specifies the opacity to be used for the filling of the interior of a shape. Values must be between 0 (fully transparent) and 1 (fully opaque) | Applies to all shapes but has no effect for <line> |
| fill-rule | nonzero | Describes the algorithm to identify which part of a shape must be filled.  Possible values are:   * nonzero * evenodd | See below. |
| stroke | none | Describes the colour to be used for the rendering of the outline of the graphical element where it is used. The value none indicates that the outline should not be rendered. | See notes on colours below. |
| stroke-opacity | 1 | Specifies the opacity to be used for the stroke. |  |
| stroke-width | 1 | The width of the stroke. | The width is related to the local user coordinate system applicable to the element. |
| stroke-linecap | butt | Specifies the shape to be used at the end of open subpaths when they are stroked.  Image showing stroke-linecap alternatives |  |
| stroke-linejoin | miter | Specifies the shape to be used at the corners of shapes when they are stroked.  Image showing stroke-linejoin alternatives |  |
| stroke-miterlimit | 4 | When the line join is miter and the ratio between the miter length and the stroke width exceeds this value the linejoin should be converted from miter to bevel.  The miter length is the distance between the outer tip and the inner corner of the miter. | The ratio of miter length to 'stroke-width' is directly related to the angle (θ) between the segments in user space by the formula: |
| stroke-dasharray | none | Specifies the pattern of dashes and gaps that shall be used to stroke paths. It is either a list of lengths or ‘none’. ‘none’ means that the stroke is drawn continuously. Line caps apply at both ends of each stroke. | If the number of lengths is odd the list should be internally duplicated.  e.g. 1,2,3 becomes 1,2,3,1,2,3 |
| stroke-dashoffset | 0 | Specifies the distance into the dash pattern that must be used to start the dash. |  |
| shape-rendering | auto | The creator of SVG content might want to provide a hint to the implementation about what trade-offs to make as it renders vector graphics elements such as 'path' elements and basic shapes such as circles and rectangles. The 'shape-rendering' property provides these hints.  Possible values are:   * auto * optimizeSpeed * crispEdges * geometricPrecision | See below. |
| display | inline | Defines whether an element is to be rendered or not.  Allowed values for this profile:   * inline : will be rendered * none: will not be rendered | Can be used to include elements that do not belong to the symbol itself but may be visible in engineering drawings (pivot point, view box, etc.) |

**Colours**:

The rendering of symbols in systems like ECDIS requires that colours are not specified explicitly. They will be defined by colour-tokens instead. The tokens will be translated to SRGB values by means of style-sheets. See the section on CSS for more details.

NOTE:  In converting CIE to sRGB, the rendering intent must follow an absolute colorimetry method. Due to the differences in colour and luminance performance between individual monitors, any “formula“  for conversion from CIE to sRGB must be based on measurements to characterize (calibrate) the monitor in order to meet the colour accuracy and separation specified for ECDIS. For interoperability with ECDIS, portrayal of other S-1xx products would need to follow the same rendering intent. This may mean that the assignment from colour token to RGB values might be overridden by an ECDIS system to the hardware dependent values and the values from the CSS file are only there as a fall back.

**Fill rule**:

For a simple, non-intersecting path, it is intuitively clear what region lies "inside"; however, for a more complex path, such as a path that intersects itself or where one sub-path encloses another, the interpretation of "inside" is not so obvious.

The two options are defined as follows:

|  |  |
| --- | --- |
| nonzero | evenodd |
| Image showing nonzero fill ruleDraw a ray from the point to infinity in any direction and then examine the places where a segment of the shape crosses the ray. Starting with a count of zero, add one each time a path segment crosses the ray from left to right and subtract one each time a path segment crosses the ray from right to left. After counting the crossings, if the result is zero then the point is *outside* the path. Otherwise, it is *inside*. | Draw a ray from the point to infinity in any direction and counting the number of path segments from the given shape that the ray crosses. If this number is odd, the point is inside; if even, the point is outside.  Image showing evenodd fill rule |

**Shape rendering**:

The hints for the property shape rendering are defined as follows:

|  |  |
| --- | --- |
| **auto** | Indicates that the user agent shall make appropriate trade-offs to balance speed, crisp edges and geometric precision, but with geometric precision given more importance than speed and crisp edges. |
| **optimizeSpeed** | Indicates that the user agent shall emphasize rendering speed over geometric precision and crisp edges. This option will sometimes cause the user agent to turn off shape anti-aliasing. |
| **crispEdges** | Indicates that the user agent shall attempt to emphasize the contrast between clean edges of artwork over rendering speed and geometric precision. To achieve crisp edges, the user agent might turn off anti-aliasing for all lines and curves or possibly just for straight lines which are close to vertical or horizontal. Also, the user agent might adjust line positions and line widths to align edges with device pixels. |
| **geometricPrecision** | Indicates that the user agent shall emphasize geometric precision over speed and crisp edges. |

### The ‘style’ attribute

The ‘style’ attribute is part of SVG 1.1, but is not supported by SVG tiny and therefore cannot be used by this profile. Nevertheless, it was allowed in previous versions of this standard.

Robust parsers and renderers may interpret the content of the style attribute. A style attribute uses CSS inline styling to specify the display properties.

### The ‘class’ attribute

The ‘class’ attribute is used to assign one or more class names to an element. In the S-100 SVG profile the class attribute is used to assign style information by way of a CSS stylesheet. It can also be used to filter or control which elements should be shown. Essentially the class tokens can be used as a key to find a set of style instructions in the corresponding Cascading Style Sheet (CSS). A processing instruction at the head of the SVG symbol indicates the corresponding CSS file.

Example:

<?xml-stylesheet href="MyStyle.css" type="text/css"?>

### CSS

The Cascading Style Sheet (CSS) mechanism allows for the possibility to change styling properties by means of a CSS file.

Though the mechanism allows modification of all styling-properties the mechanism must only be used for two use cases.

1. The colour properties ‘fill’ and ‘stroke’
2. The visibility of elements that do not belong to the symbol but may be used for engineering diagrams (e.g., the pivot point and the initial view box)

All symbols should be designed in a way that they can be viewed:

* by third party tools that do not support CSS
* in the absence of the appropriate CSS file.

How this can be achieved is demonstrated in the following example.

An excerpt of a CSS file (MyStyle.css) is defined as follows.

/\* Set to 'inline' to show viewbox and pivot point or to 'none' to hide them \*/

.layout {display: inline}

.svgBox {fill: none; stroke: blue;}

.pivotPoint {fill: none; stroke: red;}

.fNone {fill: none}

.sNone {stroke: none}

.fNODTA {fill: #93AEBB}

.sCURSR {stroke: #E38039}

.fCURSR {fill: #E38039}

.sCHBLK {stroke: #000000}

.fCHBLK {fill: #000000}

.sCHYLW {stroke: #E1E139}

.fCHYLW {fill: #E1E139}

…

For each colour token two classes are defined: one for the stroke property and the other for the fill property. In addition, there are classes for fill=”none” and stroke=”none” and classes that control the colours and the display for the pivotPoint and the svgbox.

The symbol is defined as:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<?xml-stylesheet href="style.css" type="text/css"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="70mm" height="70mm" viewBox="-350 -350 700 700" >

<g stroke="#000000" stroke-width="10" fill="#E1E139" class="fCHYLW sCHBLK">

<circle cx="0" cy="-50" r="75"/>

<circle cx="-35" cy="-70" r="10" stroke="none" fill="#000000" class="sNone fCHBLK"/>

<circle cx="35" cy="-70" r="10" stroke="none" fill="#000000" class="sNone fCHBLK"/>

<path d="M -40,-20 C -20,-40 20,-40 40,-20" fill="none" class="fNone" transform="rotate(180 0 -20)"/>

</g>

<g display="none" stroke-width="1" fill="none" class="layout">

<!-- view box-->

<rect x="-349.5" y="-349.5" height="699" width="699" stroke="blue" class="svgBox"/>

<!-- pivot point -->

<g stroke="red" class="pivotPoint">

<circle cx="0" cy="0" r="20"/>

<line x1="0" y1="-50" x2="0" y2="50"/>

<line x1="-50" y1="0" x2="50" y2="0"/>

</g>

</g>

</svg>

The first <g> element defines the stroke and fill property as fill="#E1E139" (yellow) and stroke="#000000" (black). In addition, it assigns the class to class="fCHYLW sCHBLK".

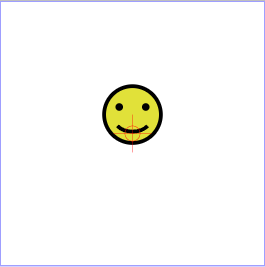
The latter assigns the fill and stroke properties to the values in the CSS file. Here the values are the same but in a different CSS file (e.g. for night view) different values may be defined.

The first <circle> element defines no a circle with these properties i.e. a circle with an black outline and a yellow fill.

The next two <circle> elements override the fill and stroke values to: stroke="none" fill="#000000" and the class to class="sNone fCHBLK". Both circles will be rendered with a black fill an no outline.

The <path> element overrides the fill value only to fill="none" and the class to class="fNone". The path will be rendered with a black stroke, and it will not be filled.

The second <g> element in the file defines the elements for the pivot point and the view box. If the symbol is rendered by using the CSS file the display property will be changed from none to inline, and in this case the elements are visible. If the symbol is rendered without the use of the CSS the elements will not be visible.

If the symbol is rendered using the style sheet the result will be:

If the CSS file is not used the pivot point and viewbox will not be visible.

## Drawing Elements

### Path

A path represents the outline of a shape which can be filled or stroked.

A path is described using the concept of a current point. In an analogy with drawing on paper, the current point can be thought of as the location of the pen. The position of the pen can be changed, and the outline of a shape (open or closed) can be traced by dragging the pen in either straight lines or curves.

The path element can carry all presentation attributes, the *class* attribute, the transform attribute and as a path specific attribute the *d* attribute.

The *d* attribute defines the outline of the shape and consists of moveTo, line, curve, and closePath commands.

Examples:

<path d="M -2.06,1.36 L -1,2.4 L 0.98,2.4 L 1.96,1.39"

class="sCHMGD" stroke="#c045d1" stroke-width="0.32"/>

<path d="M -5.88,-5.88 L 5.87,-5.88 L 5.87,5.87 L -5.88,5.87 L -5.88,-5.88 Z"

class="fDNGHL" fill="#ea5471" fill-opacity="0.25"/>

All commands are defined by a single letter following by a list of coordinates. All commands are available for absolute coordinates (uppercase letters) and for relative coordinates (lowercase letters). The relative coordinates are defined relative to the current point (cpx cpy) The absolute coordinates for relative coordinates (x y) are (cpx+x cpy+y)

All coordinates are coordinates of the local user space i.e., transformations must be taken into account.

#### MoveTo commands

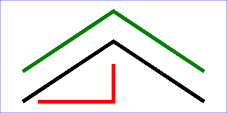
|  |  |  |
| --- | --- | --- |
| Command | Parameter | Description |
| **M** (absolute)  **m** (relative) | (x y)+ | Start a new sub-path at the given (x,y) coordinate. M (uppercase) indicates that absolute coordinates will follow; m (lowercase) indicates that relative coordinates will follow. If a moveto is followed by multiple pairs of coordinates, the subsequent pairs are treated as implicit lineto commands. Hence, implicit lineto commands will be relative if the moveto is relative, and absolute if the moveto is absolute. If a relative moveto (m) appears as the first element of the path, then it is treated as a pair of absolute coordinates. In this case, subsequent pairs of coordinates are treated as relative even though the initial moveto is interpreted as an absolute moveto. |

#### ClosePath commands

|  |  |  |
| --- | --- | --- |
| Command | Parameter | Description |
| **Z** (absolute)  z (relative) | (none) | The current sub-path shall be closed by drawing a straight line from the current point to current subpath's initial point, which then shall become the new current point. Since the Z and z commands take no parameters, they have an identical effect. |

#### LineTo commands

|  |  |  |
| --- | --- | --- |
| Command | Parameter | Description |
| **L** (absolute)  l (relative) | (x y)+ | Draws a straight line from the current point to (x, y) and set the current point to this position. If multiple coordinate pairs are given this is repeated.  **L x1 y1 x2 y2** is equivalent to **L x1 y1 L x2 y2 and**  **l x1 y1 x2 y2** is equivalent to **l x1 y1 l x2 y2** |
| **H** (absolute)  **h** (relative) | x+ | Draws a horizontal line to (x cpy) and set the current point to this position. |
| **V** (absolute)  **v** (relative) | y+ | Draws a vertical line to (cpx y) and set the current point to this position |

Example:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="60mm" height="30mm" viewBox="-300 -150 600 300" >

<path d="M-240 120 L 0 -40 240 120" fill="none" stroke="black" stroke-width="10"/>

<path d="M-240 40 l 240 -160 240 160" fill="none" stroke="green" stroke-width="10"/>

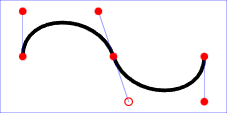
<path d="M-200 120 h 200 v-100" fill="none" stroke="red" stroke-width="10"/>

<rect x="-299.5" y="-149.5" height="299" width="599" stroke="blue" fill="none"/>

</svg>

#### Curve commands

|  |  |  |
| --- | --- | --- |
| Command | Parameter | Description |
| **C** (absolute)  c (relative) | (x1 y1 x2 y2 x y)+ | A cubic Bézier curve shall be drawn from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve.  The current point is then set to (x y). If multiple sets of coordinates are specified this is repeated. |
| **S** (abolute)  **s** (relative) | (x2 y2 x y)+ | A cubic Bézier curve shall be drawn from the current point to (x,y). The first control point shall be the reflection of the second control point on the previous command relative to the current point. (If there is no previous command or if the previous command was not an C, c, S or s, the first control point shall be coincident with the current point.) (x2,y2) shall be used as the second control point (i.e., the control point at the end of the curve). If multiple sets of coordinates are specified, a polybézier shall be drawn. At the end of the command, the new current point shall be the final (x,y) coordinate pair used in the polybézier. |
| **Q** (absolute)  **q** (relative) | (x1 y1 x y)+ | A quadratic Bézier curve is drawn from the current point to (x,y) using (x1,y1) as the control point. Then the current point is set to (x y). If multiple sets of coordinates are specified this is repeated. |
| **T** (absolute)  **t** (relative) | (x y)+ | The same as the Q, q command except the control point shall be the reflection of the control point on the previous command relative to the current point. If there is no previous command or if the previous command was not a Q, q, T or t, the control point shall be current point. |

Example for a path using cubic Bézier curves:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="60mm" height="30mm" viewBox="-300 -150 600 300" >

<path d="M-240 0 C -240 -120 -40 -120 0 0 S 240 120 240 0"

fill="none" stroke="black" stroke-width="10"/>

<g fill="red" stroke="none">

<line x1="-240" y1="0" x2="-240" y2="-120" stroke="blue"/>

<circle cx="-240" cy="0" r="10"/>

<circle cx="-240" cy="-120" r="10"/>

<line x1="-40" y1="-120" x2="0" y2="0" stroke="blue"/>

<circle cx="-40" cy="-120" r="10"/>

<circle cx="0" cy="0" r="10"/>

<line x1="0" y1="0" x2="40" y2="120" stroke="blue"/>

<circle cx="40" cy="120" r="10" fill="none" stroke="red" stroke-width="3"/>

<line x1="240" y1="0" x2="240" y2="120" stroke="blue"/>

<circle cx="240" cy="0" r="10"/>

<circle cx="240" cy="120" r="10"/>

</g>

<rect x="-299.5" y="-149.5" height="299" width="599" stroke="blue" fill="none"/>

</svg>

### Line

The <line> element defines a line segment that starts at one point and ends at another.

The <line> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| x1 | 0 | The x-coordinate of the start point of the line in user space coordinates |
| y1 | 0 | The y-coordinate of the start point of the line in user space coordinates |
| x2 | 0 | The x-coordinate of the end point of the line in user space coordinates |
| y2 | 0 | The y-coordinate of the end point of the line in user space coordinates |

Example:

<line x1="-240" y1="0" x2="-240" y2="-120" stroke="blue"/>

### Rectangle

The <rect> element defines a rectangle which is axis-aligned with the current user coordinate system. Rounded rectangles can be achieved by setting appropriate values for attributes *rx* and *ry*.

The <rect> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| x | 0 | The x-axis coordinate of the side of the rectangle which has the smaller x-axis coordinate value in the current user coordinate system. |
| y | 0 | The y-axis coordinate of the side of the rectangle which has the smaller y-axis coordinate value in the current user coordinate system. |
| width | 0 | The width of the rectangle |
| height | 0 | The height of the rectangle |
| rx | 0 | For rounded rectangles, the x-axis radius of the ellipse used to round off the corners of the rectangle. |
| ry | 0 | For rounded rectangles, the y-axis radius of the ellipse used to round off the corners of the rectangle. |

### Circle

The <circle> element defines a circle based on a centre point and a radius.

The <circle> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| cx | 0 | The x-axis coordinate of the centre of the circle in the current user coordinate system. |
| cy | 0 | The y-axis coordinate of the centre of the circle in the current user coordinate system. |
| r | 0 | The radius of the circle |

### Ellipse

The <ellipse> element defines an ellipse which is axis-aligned with the current user coordinate system based on a centre point and two radii.

The <ellipse> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| cx | 0 | The x-axis coordinate of the centre of the ellipse in the current user coordinate system. |
| cy | 0 | The y-axis coordinate of the centre of the ellipse in the current user coordinate system. |
| rx | 0 | The x-axis radius of the ellipse. |
| ry | 0 | The y-axis radius of the ellipse. |

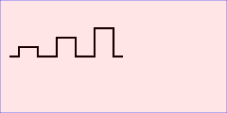
### Polyline

The <polyline> element defines a set of connected straight line segments. Typically, <polyline> elements define open shapes.

The <polyline> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| points | (empty) | A list of coordinates that defining pairwise the points that making the polyline. Coordinates are separated by whitespaces and/or a comma. The number of coordinates must be even, otherwise the attribute should be treated as if the attribute has not been specified. |

Note that the element can be filled but even if the polyline is closed (first and last point is identical) the line-join property will not been taken into account at this point. It is recommended that the element <polygon> or <path> is used for filled shapes.

Example:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="60mm" height="30mm" viewBox="-300 -150 600 300" >

<polyline points="-275,0 -250,0 -250,-25 -200,-25 -200,0

-150,0 -150,-50 -100,-50 -100,0 -50,0

-50,-75 0,-75 0,0 25,0" stroke="black"

fill="none" stroke-width="5"/>

<rect x="-299.5" y="-149.5" height="299" width="599"

stroke="blue" fill="red" fill-opacity="0.1"/>

</svg>

### Polygon

The <polygon> element defines a closed shape consisting of a set of connected straight line segments.

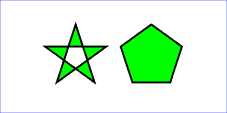
The <polygon> element can carry all presentation attributes, the *class* attribute, the transform attribute and the following specific attributes.

|  |  |  |
| --- | --- | --- |
| Attribute | Initial value | Description |
| points | (empty) | A list of coordinates that taken pairwise make up the polygon. Coordinates are separated by whitespaces and/or a comma. The number of coordinates must be even, otherwise the attribute should be treated as if the attribute has not been specified. |

A 'polygon' element must be rendered in a way that produces the same result as if the following path were specified instead:

1. Perform an absolute moveto operation to the first coordinate pair in the list of points.
2. For each subsequent coordinate pair, perform an absolute lineto operation to that coordinate pair.
3. Perform a closepath command.

Example:



<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"

width="60mm" height="30mm" viewBox="-300 -150 600 300" >

<g fill="#00ff00" fill-rule="evenodd" stroke="black" stroke-width="5">

<polygon points="0,-85.1 50,68.8 -80.9,-26.3 80.9,-26.3 -50,68.8"

transform="translate(-100 0)"/>

<polygon points="0,-85.1 80.9,-26.3 50,68.8 -50,68.8 -80.9,-26.3"

transform="translate(100 0)"/>

</g>

<rect x="-299.5" y="-149.5" height="299" width="599" stroke="blue" fill="none"/>

</svg>