Scalable Vector (Images and) Graphics: SVG

CO2016

Multimedia and Computer Graphics

University *of* Leicester

Overview

- What is SVG?
- Why SVG and not FLASH
- Creating simple shapes
- Introduction to attributes
- Grouping
- Animation

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- Key idea: change pixels (a colour with a fixed position) to *“simple scalable elements with a vector position”*
- SVG graphics do NOT lose any quality if they are zoomed or resized [SCALABLE]
- Every element (eg a square) and every attribute of an element (eg colour, position) can be animated [time dependent VECTOR]

What is SVG?

- SVG is coded using XML format
- SVG is a World Wide Web Consortium (W3C) recommendation
- SVG integrates with other W3C standards such as the DOM and XSL

More Advantages of Using SVG

- SVG files can be read and modified by any text editor
- SVG files are smaller and more compressible than JPEG and GIF images
- SVG images can be printed with high quality at any resolution
- Text in SVG is selectable and searchable
- SVG works with Java technology
- SVG is an open standard
- SVG files are pure XML

Why not Flash?

- Both Flash and SVG have similar features . . . , but
- SVG complies with other standards
- Flash is not open source

Template for <file-name>.svg

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">

...
...

</svg>
```

Embed SVG into HTML

There are various options for embedding SVG into HTML. The preferred option for this module is:

```
...
<iframe src="myFileName.svg" width="450" height="300">
</iframe>
...
```

Basic Shapes

The following are predefined *elements* in SVG with the following *tags*.

- Rectangle <rect>
- Circle <circle>
- Ellipse <ellipse>
- Line <line>
- Polyline <polyline>
- Polygon <polygon>
- Path <path>
- Text <text>

The Stroke

- Roughly speaking: *Stroke* refers to a line, text, or the outline of an element.
- A stroke has width, colour and other attributes.
- A blue rectangle might have a red stroke boundary with a (relatively) small width . . .

How to Use Simple Shapes

Each shape has a corresponding *tag*. Inside the tag the attributes for the shape can be set.
Example of a rectangle:

```
<rect x="0" y="30" width="40" height="20''>
</rect>
```

Further attributes are

- `fill` (color)
- `stroke-width` (rational number ie decimal)
- `stroke` (color) [NB `stroke`, not `stroke-color`!!]
- `fill-opacity` (rational between 0 and 1)
- ...

Setting Attributes

There are two ways to set attributes.
Either set each attribute separately:

```
stroke-width="3"
fill="red"
stroke="rgb(23,255,10)"
...
```

Or set the `style` attribute which “collects together” other attributes:

```
style="fill:blue stroke-width:2 opacity:0.3"
```

Cascading Style Sheets

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents.

SVG shares many of its styling properties with CSS.

See `http://www.w3.org/Style/CSS/` and `http://www.w3.org/TR/SVG/styling.html` for more details.

Path

A path in SVG is a sequence of commands, with each command typically “moving to a point”, “drawing a line”, “drawing a curve” and so on.

Classically this is used to draw (curved) lines and shapes.

We will revisit paths when creating animated images: a path is used to specify the “animation path”.

Path Commands

- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic Belzier curve
- T = smooth quadratic Belzier curveto
- A = elliptical Arc
- Z = closepath

Path Commands

All of the commands can also be expressed with lower case letters. Capital letters mean absolutely positioned, lower case letters mean relatively positioned.

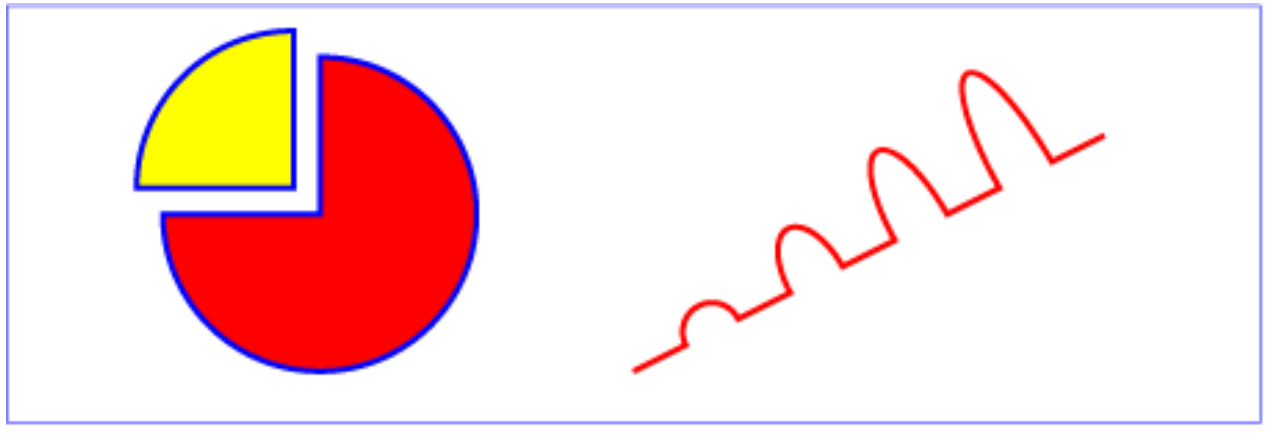
```
<path d="M150 0 L50 200 L250 200 Z" />
```

```
<path d="M150 0 l50 200 L250 200 Z" />
```

The documentation and examples for all Path commands can be found at
`http://www.w3.org/TR/2000/CR-SVG-20001102/paths.html`

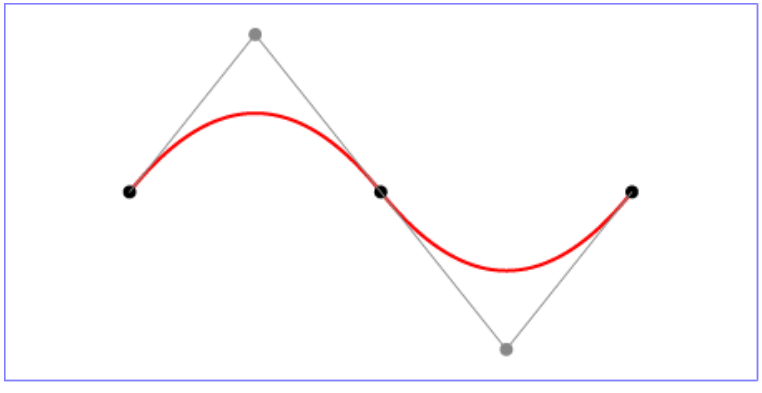
Path Examples

```
 <path d="M300,200 h-150 a150,150 0 1,0 150,-150 z"
       style="fill:red; stroke:blue; stroke-width:5"/>
<path d="M275,175 v-150 a150,150 0 0,0 -150,150 z"
       style="fill:yellow; stroke:blue; stroke-width:5"/>
<path d="M600,350 l 50,-25
          a25,25 -30 0,1 50,-25 l 50,-25
          a25,50 -30 0,1 50,-25 l 50,-25
          a25,75 -30 0,1 50,-25 l 50,-25
          a25,100 -30 0,1 50,-25 l 50,-25"
       style="fill:none; stroke:red; stroke-width:5" />
```



Path Examples

```
<path d="M200,300 Q200,50 600,300 T1000,300"
      style="fill:none; stroke:red; stroke-width:5" />
<g style="fill:#888888">
  <circle cx="400" cy="50" r="10"/>
  <circle cx="800" cy="550" r="10"/>
</g>
<path d="M200,300 L400,50 L600,300
         L800,550 L1000,300"
      style="fill:none; stroke:#888888; stroke-width:2"/>
```



What is wrong?

Grouping Elements

The g tag is a container element for grouping together related elements.

```
<g opacity=".9">
  <rect
     x="0"
     y="0"
     height="150"
     width="225"
     style="fill:rgb(0,0,156)">
  </rect>

  <circle
     cx="112.5"
     cy="75"
     r="0"
     fill="white">
  </circle>
</g>
```

Element Identifiers

The defs tag is a container and is used to define identifiers for elements. These elements can be referenced later in the image by quoting the identifier.

For each element to be referenced the id attribute must be set.

```
<defs>
   <g id="sp1" >
      <polygon
         points="−7.5, 0
                 7.5, 0
                 0,24"
         style="fill:white"/>
      <rect ....  />
   </g>
 </defs>
```

Element Identifiers

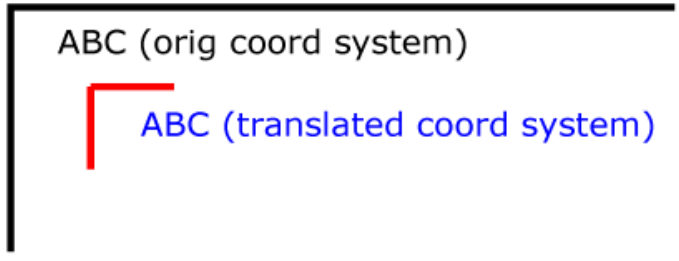
Depending on the type of element that is defined you can refer to it by the <use> tag or url.

```
<use xlink:href="#sp1" x="112.5" y="108"/>
```

```
<linearGradient id="MyGradient">...</linearGradient>
...
<rect style="fill:url(#MyGradient)"/>
```

Transform Examples

```
<g transform="translate(50,50)">
  <g style="fill:none; stroke:red; stroke-width:3">
    <!-- Draw lines of length 50 user units along
         the axes of the new coordinate system -->
    <line x1="0" y1="0" x2="50" y2="0" style="stroke:red"/>
    <line x1="0" y1="0" x2="0" y2="50" />
  </g>
  <text x="30" y="30" style="font-size:20 font-family:Verdana">
    ABC (translated coord system)
  </text>
</g>
```



Transform

A new user space (i.e., a new current coordinate system) can be established by specifying transformations in the form of a transform attribute on a container element or graphics element. The transform attribute transforms all user space coordinates and lengths on the given element. Transformations can be nested, in which case the effect of the transformations are cumulative.

- translate
- rotate
- skewX
- skewY
- matrix

Animate

SVG animation elements

- animate
- set
- animateMotion
- animateColor
- animateTransform
- path attribute
- mpath element
- keyPoints attribute
- rotate attribute

SVG's animation elements were developed in collaboration with the W3C Synchronized Multimedia (SYMM) Working Group, developers of the Synchronized Multimedia Integration Language (SMIL) 1.0 Specification [SMIL1].

Animate Example

The animate tag allows scalar attributes be assigned different values over time.

```
<rect x="0" y="0" height="40" width="60" style="fill:blue" >
<animate attributeType="XML" attributeName="height"
from="40" to="0" dur="5s" repeatCount="indefinite" />
</rect>
```

- width
- from
- to
- begin
- dur
- ...

Animate Motion Example

The animateMotion tag utilises a path that specifies the direction of motion for an element

```
<rect
   x="100"
   y="50"
   height="40"
   width="60"
   style="fill:blue" >

 <animateMotion path="M 0 0 L 100 0 L0 0"
                          dur="10s" fill="freeze" />


</rect>
```

Animate Color Example

```
<animateColor
        attributeName="fill"
        from="blue" to="white"
        dur="10s" />
```

Animate Transform Example

```
<rect x="0" ... >
 <animateTransform attributeName="transform" attributeType="XML"
   type="rotate" from="0" to="-30"
   begin="3s" dur="3s" fill="freeze" />
 <animateMotion path="M 0 0 L 200 80 L 0 200"
   begin="3s" dur="8s" fill="freeze" />
  </rect>
```

The types are

- translate
- scale
- rotate
- skewX
- skewY

Further Topics

- scripting
- event handling
- links
- additional variables