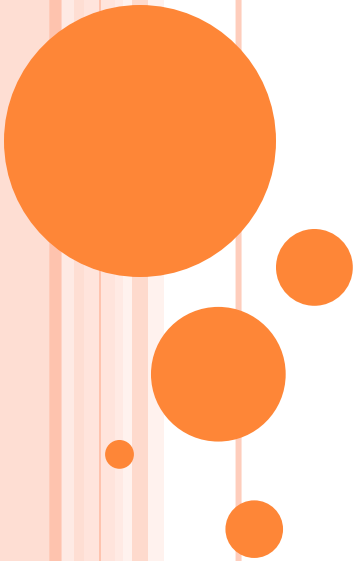


CS474 Multimedia Technology

Pr. G. Tziritas, *Spring 2018*

SVG – Animation Tutorial

G. Simantiris *(TA)*

Overview

- Introduction – Definitions
- SVG
- Creating SVGs
- SVG basics
- Examples
- Animation using software
- Examples
- References

Introduction - Definitions 1

- **Graphics**

  are visual images or designs on some surface (here: webpage) to inform, illustrate, or entertain – a pictorial representation of data.

- **Computer graphics**

  Two types:

  - **raster graphics**, where each pixel is separately defined, and
  - **vector graphics**, where mathematical formulas are used to draw lines and shapes, which are then interpreted at the viewer's end to produce the graphic.

Introduction - Definitions 2

- **Web graphics**
  - Websites began to use the GIF format to display small graphics (banners, advertisements, navigation buttons) on web pages.
  - Modern web browsers can now also display JPEG, PNG and SVG images.
  - Plugins expand the web browser functions to display animated, interactive and 3-D graphics contained within file formats such as SWF and X3D.

Introduction - Definitions 3

- **Animation**

  A dynamic medium in which images or objects are manipulated to appear as moving images.

  - Traditionally drawn/painted by hand.
  - Computer Generated Imagery (CGI): 2D/3D.
  - Other methods (paper cutouts, puppets, clay figures): stop motion technique.

  Commonly the effect of animation is achieved by a rapid succession of sequential images that minimally differ from each other.

Introduction – Definitions 4

- **XML** (**Ex**tensible **M**arkup **L**anguage)

  a markup language that defines a set of rules for encoding documents in a format that is both human- and machine-readable. Emphasis: simplicity, generality, and usability across the Internet. Focus is on documents but the language is widely used for the representation of arbitrary data structures such as those used in web services.

- **VML** (**V**ector **M**arkup **L**anguage)

  was an XML-based file format for two-dimensional vector graphics.

What is SVG?

SVG stands for **S**calable **V**ector **G**raphics. It is an XML language and file format, which allows you to code two-dimensional graphics that scale and can be manipulated via CSS or JavaScript.

- *The code tends to adhere to agreed upon standards of how SVG should be written and how client software should respond.*
- *It is written in text, and can generally be read not only by machines but also by humans.*
- *JavaScript can be used to manipulate both the objects and the Document Object Model, in ways quite similar to how JavaScript is used in conjunction with HTML.*

SVG

- Can be a static image (logo or illustration), or a complex animation*.
- All graphics are plotted on a coordinate system of at least an x and y axis. When authoring, we give the browser instructions on where to plot points on the coordinate system and connect them. By connecting the plotted points we can create shapes, lines, or paths.
- There isn't any degradation or loss of fidelity when the graphics are scaled up. They are simply redrawn to accommodate the larger size ➔ perfect for multi-context scenarios, like Responsive Web Design.

* see later on this tutorial

Why use SVG?

- **Scalable**

  SVG easily scales from small to big without loss of quality or fidelity. This is because vectors describe series of paths, shapes, and more that are created using connect points plotted on x-y axes.

- **File size**

  SVGs are typically smaller files. For example, a small logo in PNG can be 30 KB and the same SVG image only 6 KB. Big savings, especially for a site that requires a lean performance budget.

- **Modifiable**

  SVGs are easily modified using both JavaScript and CSS. This makes it simple to have a base SVG file and repurpose it in multiple locations on the site with a different treatment.

When to use SVG

- SVG images aren't perfect for everything!
- Ideally, you'd use SVGs for icons, simple line graphics, and, if possible, logos.
- SVGs are ideal for imagery that are used multiple times on a page (like interface icons).

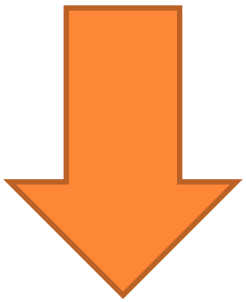
SVG History

- 1999: 1st draft
- 2000: W3C issues SSVG as a candidate recommendation.
- Multiple "working drafts" were published in just a couple years (see also SVG primer).
- Used by some designers and developers.
- **Now** that all modern browsers support SVG, the wide adoption of Responsive Web Design and increased focus on quality tools and practices for front-end development has only increased the use of SVG in web development projects.
- Current version of SVG: 1.1 (published as a W3C Recommendation in August 2011).

Creating SVGs with tools

- Adobe Illustrator
- Inkscape (free & open source)
- SVG-edit (free online)

} draw



save as **.svg**

Creating SVGs with code

- You first need a text editor that you are comfortable using ➔ Notepad++
- Then you'll need to know the syntax ➔ elements & attributes (remember, it's XML).

type

save as **.svg**

SVGs in HTML webpage 1

- Using the <img> tag

  Simplest way to add SVG to a web page:

  <img src="drawing.svg" >

  Advantages:

  - Very easy to do, markup that's simple and often used.
  - File can be cached for better performance.

  Disadvantages:

  - Unable to manipulate the SVG with JavaScript or CSS. For images like logos or icons that don't need manipulation, this option is a fine choice.

SVGs in HTML webpage 2

- Using the object element

  Most flexible way to add SVG to a web page:

  <object type="image/svg+xml“ data=“drawing.svg">
  ****

  </object>

  Advantages:

  - Full access to the SVG internal elements ➔ you can still manipulate the SVG with JavaScript and CSS (must access via JavaScript!).
  - The object element is cacheable.

  Disadvantages:

  - Implemented as-is, with a fallback image, the fallback image will always be retrieved by the client even if it is not needed!

SVGs in HTML webpage 3

- Plain old SVG XML

  Right in the HTML document. One less asset for the browser to retrieve from a remote location (everything needed to render the image is right in the HTML):

```
<svg width="400" height="400">
  <rect class="r1" x="0" y="0" width="400"
        height="400" fill="#56A0D3" />
</svg>
```

  Advantages:

  - Modify the SVG using CSS and JavaScript. Add a class to a one of the elements and then style it accordingly from your CSS.

  Disadvantages:

  - Not cacheable. May seem like a small affair but for high traffic sites this can be a big problem with scaling and performance.

Basic SVG elements

- **svg**

  Main element inside of which shapes, paths, lines etc. are nested.

  Defines the **viewport** or **canvas** for SVG content.

  Some attributes:

  - **height**, **width** : of the viewport
  - **viewBox** : set a new coordinate system
  - **preserveAspectRatio** : forces the aspect ratio so that the drawing isn't streched when defining a new coordinate system

Basic SVG elements

- **g**

  Used to create groups of SVG objects.

  Transformations applied to **g**, apply to everything inside the group.

  Accepts all global attributes (like **stroke**, **fill**, etc.)
- **circle**

  Creates circles.

  Three element specific attributes:
  - **cx** : x coordinate of the center
  - **cy** : y coordinate of the center
  - **r** : radius

Basic SVG elements

- **rect**

  Creates rectangles.

  Attributes:

  - **height**, **width** : height & width of the shape
  - **x**, **y** : (x,y) coordinate where the shape should start drawing

- **line**

  Line by connecting 2 points.

  Attributes:

  - **x1**, **y1** : (x,y) coordinate where the line starts
  - **x2**, **y2** : (x,y) coordinate where the line ends

Basic SVG elements

- **path**

  Any shape can be created with this element.

  Attributes:

  - **d** : definition attribute, which defines the path to follow to create the shape. The path is defined using "path descriptions:" *Moveto*, *Lineto*, *Curveto*, *Arcto*, *ClosePath*.
  - **pathLength** : total length of the path

- **polygon**

  Create any shape with many sides (e.g. octagon) using **points** (the only specific attribute for this element).

Basic SVG elements

- **text**

  Set text in SVG.

  Attributes:

  - **textLength** : exact value of the length of the text (used for very specific layouts).
  - **rotate** : list of numbers that will determine the rotation of the text.
  - **x**, **y** : (x,y) coordinate where the text begins

  Also uses global **style** attribute to set *size*, *font*, *stroke*, etc. of the text.

Basic SVG elements

- **textPath**

  Puts text along a **path** element so that it follows the shape (e.g. display text on a curve).

And many more which you can explore and try on W3C and other sources.

Also have in mind that you can create animation with SVG by using CSS or JavaScript…

Simple example (with CSS animation)

- Sample files:
  - helloSVG.html
  - drawing.svg
  - mystylesvg.css

SVG & JavaScipt

- JavaScript ➔ manipulate SVG images.
- Not different than for any other DOM (Document Object Model) element.
- Use JavaScript, jQuery, etc.
- There are also libraries which make it really simple to create and manipulate SVG. Popular choices:
  - Snap.svg
  - SVG.js
  - Velocity.js
  - D3.js
  - etc.

Example JavaScript controlled animation

- Sample files:
  - helloSVGjs.html
  - yellowHead.svg
  - mystylesvgjs.css
  - svgjs.js

Animation with Software

- Adobe Flash
- Vectorian Giotto
  - ➔ nice intro video + small tutorial (in greek) by Dr. Costas Panagiotakis, Associate Professor, Department of Business Administration (Agios Nikolaos), Technological Educational Institute of Crete.
- Bannersnack
- Synfig Studio

Synfig Studio Introduction

- Free & open source software for creating 2D animations (Windows, Linux, Mac).
- 3 types of animation:
  1. **Cut-out**
  2. **Morphing**
  3. **Frame by frame**
- Useful tutorials to start with:
  - Synfig Tutorial 1: Getting Started
  - Synfig Training Course – Lesson 1
  - Synfig Tutorial 2: Animation Basics
  - etc.

Synfig Studio Animation

- Examples shown in class
  - testanim.png
  - Synfig Animation 1.gif
- Sample files (.sifz)
  - testanim.sifz
  - testanim2.sifz
- Experiment and play around for yourself.

References

- w3schools.com
- HTML5
- HTML5 Tutorial
- CSS Tutorial
- SVG Tutorial
- JavaScript Tutorial
- Synfig Studio Tutorials
- How to use Synfig Studio