2D Web Graphics: SVG

I. Herman
W3C

D.A. Duce, F.R.A. Hopgood
Oxford Brookes University

Slides available on-line at:

http://www.w3.org/Talks/2002/IH-Web3D/

- Introduction (15 minutes)
- Transformations, structuring (15 minutes)
- Geometry (15 minutes)
- Clipping, masking (10 minutes)
- Attributes, fonts (30 minutes)
- Styling (10 minutes)
- Filters (10 minutes)
- Animation (20 minutes)
- DOM (20 minutes)
- Miscellaneous (5 minutes)
- Implementations, examples (10 minutes)
- Future (10 minutes)

History

- 1989: Invention of the Web by Tim Berners-Lee
- 1993: Mosaic browser released
- 1994/95: Take off
- 1995: "Browser War"
- 1994: W3C Formed at MIT, Cambridge, USA
  - modeled after the X Consortium, but...
  - ... more international from the start!
- 1995: European Host at INRIA, France
- 1996: Asia-Pacific Host at Keio University, Japan
- 1997 onwards: Office program (local representation in 11 countries)

Membership Growth in the World

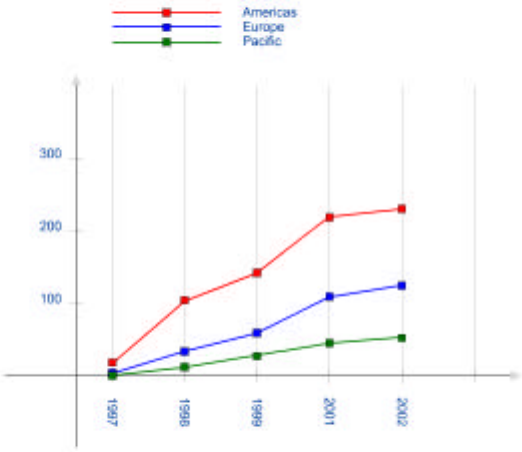
Type	Americas	Europe	Pacific	Total
Full	63	28	16	107
Affiliate	230	124	52	406
Hosts	1	1	1	3
Total	294	153	69	516

Americas
Adobe, Akamai, AOL, Apple, AT&T, Boeing, Corel, HP, IBM, IEEE CS, Library of Congress, Lucent, NASA Ames, Microsoft, MIT, OMG, Openwave, Sun, Waterloo Maple, Web3D,...

Asia
Academia Sinica, Fujitsu, Hitachi, Hong Kong Un. of S & T, IDA, ETRI, NEC, NTT DoCoMo, Mitsubishi, Sharp, Toshiba,...

Australia
CSIRO, DSTC, IPR, National Library, Un. of NSW,...

Europe
Agfa, BBC, BT, CERN, CWI, DaimlerChrysler, Elsevier, Ericsson, EUnet, Fraunhofer, INRIA, Nokia, Philips, Reuters, Siemens, Software AG, Thomson Multimedia, ...



W3C Affiliate Membership evolution 1997-2001

W3C delivers...

- Recommendations
  - Web "standards"
  - Developed by W3C working groups, with the participation of members
  - Members and public support for industry-wide adoption
- Sample Code
  - Jigsaw Web server
  - Amaya editor/browser
  - Validators
- Assistance, guidelines, outreach
  - Validators
  - Design guidelines
  - Information on, eg, accessibility issues

Why "Web Graphics"?

Bandwidth

Images are a major bottleneck in accessing web sites

Flexibility

It is unknown what the rendering device will be (PC, PDA, Phone,...)

Graphics should not be of fixed resolution

Hyperlinking

This is the Web: any graphics element should be potentially an anchor and/or target

Integration with the environment

Graphics is not used in isolation, but part of a Web infractucture!

Metadata

Graphics should be searchable, should be explorable for agents

Client side animation and interaction

Needed by large number of applications (eg, CAD, cartography)

Graphics should also be "sexy" (if necessary)

They should be able to represent company logos, publicity images, etc

What do we Need?

Vector graphics (of course...)

- Resolution independent
- Keeps "content" information
- In line with earlier developments (GKS, PHIGS, PostScript, ...)

Textual encoding (too)

- Let the client side interpret the content
- Searchable
- Easy to edit, add links, etc

Two lines of development:

- 2D Graphics: WebCGM, SVG
  - Developed by the World Wide Web Consortium
- X3D
  - Successor of VRML, also in XML
  - Developed by the Web3D Consortium

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

A Simple SVG Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="600" height="300">
  <g transform="translate(10 10)">
    <g stroke="none" fill="lime">
      <path d="M  0.0  112 L  20  124 L  40  129 L  60  126 . . .
                 . . .
      </path>
    </g>
  </g>
</svg>
```

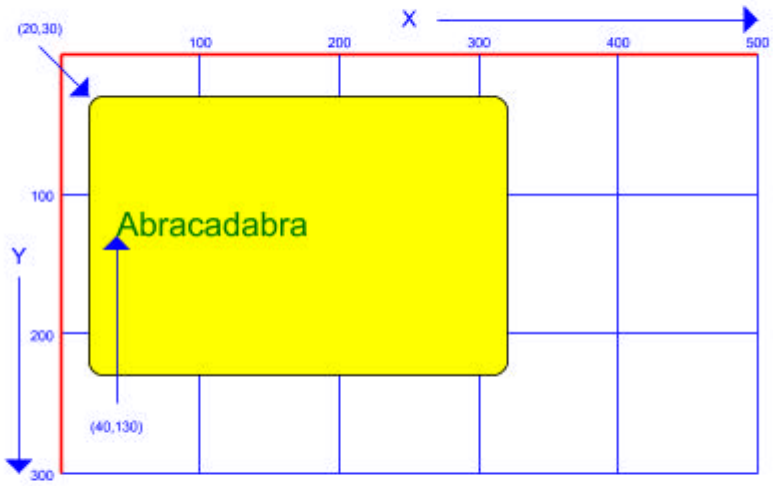
- svg is the top XML element, it establishes the (initial) coordinate system
- g is used for (hierarchical) grouping
- path is the general tool to define geometry
- paths can be filled, stroked,...
- paths can be transformed, animated, filtered,...

Coordinates, Transformations

- Initial coordinate system is set up in the svg element:
  - width and height attributes
    - the user agent should honour those values when displayed
  - viewBox attribute
    - full image is mapped onto the view canvas by the user agent
    - there is a preserveAspectRatio attribute to control the mapping
  - or both
- Large palette of units (px, pt, mm, cm, in, etc.)
- Transformations can be defined on all elements
  - translate, scale (both in X and Y), skewX and skewY, rotate
  - general 2x3 matrix
  - concatenation of all these (from right to left):

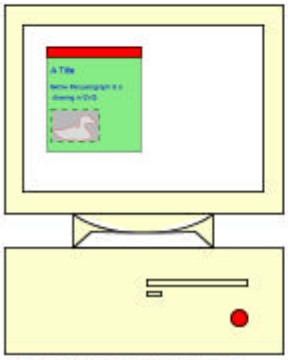
    ... transform="scale(2) rotate(20) translate(1,2) scale(4)" ...

Basic Coordinate System

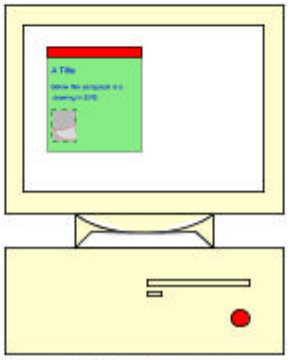



```
<rect x="20" y="30" width="300" height="200" rx="10" ry="10"/>
<text x="40" y="130">Abracadabra</text>
```


Establishing the Coordinate System



```
<object data="coord.svg"
   type="image/svg+xml"
   width="600" height="400">
</object>
```



```
<object data="coord.svg"
   type="image/svg+xml"
   width="300" height="400">
</object>
```


```
<object data="coord.svg"
   type="image/svg+xml"
   width="W" height="H">
</object>
```

```
<svg width="600" height="400">
```


```
<svg viewBox="0 0 600 400"
   preserveAspectRatio=
      "XMaxYMax slice ">
      "XMidYMid meet">
      "none">
```

Preservation of Aspect Ratio: Meet and Slice

Viewport A: <object data="duck.svg" type="image/svg+xml" width="600" height="400" />
Viewport B: <object data="duck.svg" type="image/svg+xml" width="400" height="600" />

<svg viewBox="0 0 1200 1200" preserveAspectRatio="XMinYMin slice " >

XMinYMin meet

XMidYMid meet

XMaxYMax meet

XMinYMin meet

XMidYMid meet

XMaxYMax meet

XMinYMin slice

XMidYMid slice

XMaxYMax slice

XMidYMid slice

XMaxYMax slice

Simple Transformations

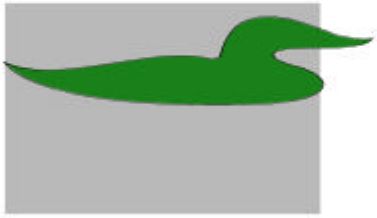
Defined on 300 x 200 coordinate system
x increases across the page, y down the page

Translate: 100 in x, -20 in y

```
<path transform="translate(100, -20)"
   d="M 0 112c40 48 120-32 160-6 ... z"/>
```

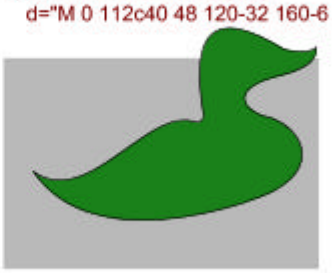
Scale 1.2 in x and 0.5 in y

```
<path transform="scale(1.2, 0.5)"
   d="M 0 112c40 48 120-32 160-6 ... z"/>
```



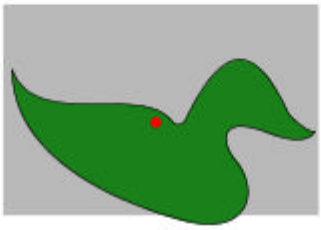
Rotate: -15 degrees about the origin

```
<path transform="rotate(-15)"
   d="M 0 112c40 48 120-32 160-6 ... z"/>
```

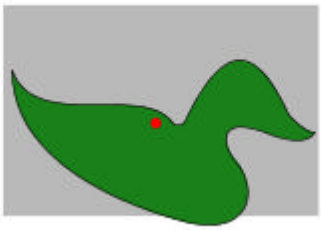


Rotations

Rotation about the point (145, 112)
transform="translate(145, 112) rotate(20) translate(-145, -112)"



Rotation about the point (145, 112)
transform="rotate(20,145,112)"



Transformation Examples

Grouping

- Elements can be "grouped" in a g element:
  - groups can be nested
  - groups can set transformations, attributes, etc, inherited by their constituents
  - groups can be named, can be referred to in, eg, URL-s
  - groups can make the SVG content more accessible for specialized browsers
    - eg, browsers for visually impaired
  - groups make SVG code more readable
- Well-known concept in graphics packages
- Typical usage of a group:

```
<g id="duck" transform="translate(100,345)">
  <desc>This is the duck></desc>
  <path d="...."/>
</g>
```


Structuring SVG Files: Templates

- Elements can be enclosed in a defs section (ie, they are not displayed!)
  - used repeatedly through a use element:

```
<defs>
   <g id="duck"> <path d="..." .../> </g>
</defs>
...
<use xlink:href="#duck">
```

  - transformation, style, etc, can also be defined on use
- symbol can also used instead of <defs><g id="...">...</g></defs>
  - it has its own view box and preserveAspectRatio attribute
- svg elements can also be nested:

```
<svg ....>
   ...
   <svg ....>...</svg>
   ...
</svg>
```

  - may be useful in setting up local coordinate systems, copying external files into content, ...
- use may also refer to external URL-s!

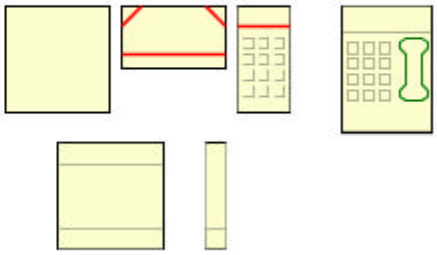
Grouping and Templates are Used a Lot!

Grouping is used a lot

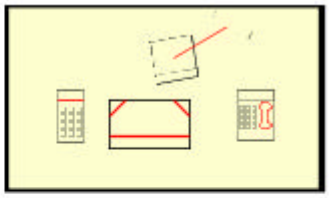

```
<g transform="translate(100,150)">
    <circle r="60" />
    <text>My Circle</text>

      <g transform="translate(0,30)">
          <g transform="scale(0.25)">
              <circle r="60" />
              <text>My Circle</text>
          </g>
      </g>

</g>
```



Over 20 levels used in this example!



- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Geometry: Path Expressions

- Define a general contour with the d attribute:
  - <path d="M 0.0 112 L 20 124 L 40 129 L 60 126...">
- A sequence of command characters and coordinates:
  - M: move
  - L, H, V: line (general, horizontal, vertical)
  - Q, C: Bézier curve (quadratic, cubic)
  - A: (elliptical) arc
- Contour can be closed or open
  - the character z or Z closes a path

Path Simplifications

Special features to reduce the size of path expressions:

Use relative coordinates

l instead of L, c instead of C, etc, define relative coordinates

Drop repeated command characters

"M 1 2 L 3 4 5 6 7 8 z"

is equivalent to:

"M 1 2 L 3 4 L 5 6 L 7 8 z"

Syntactic simplification

Whitespace can be omitted everywhere where possible:

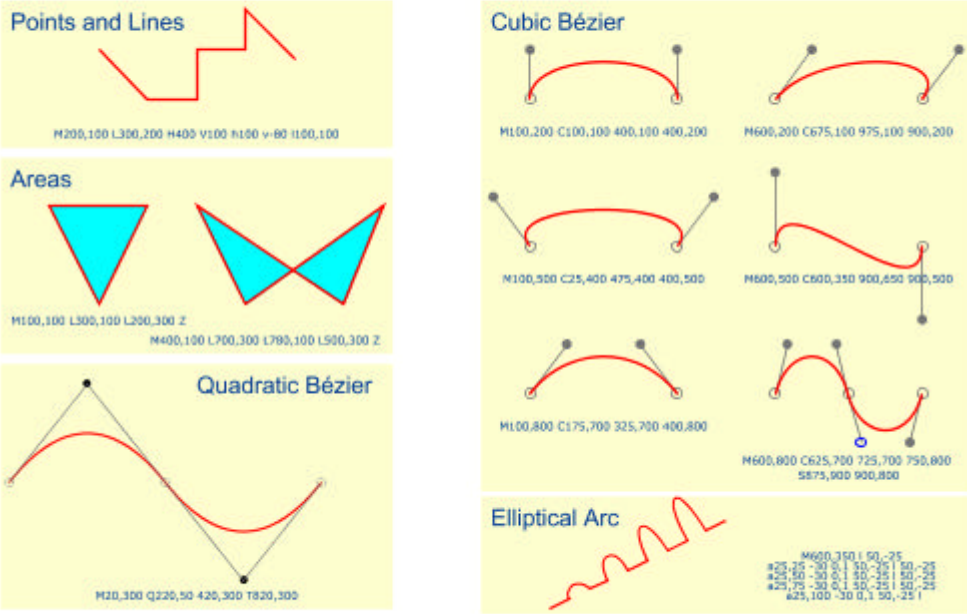
<path d="M0 312c40 48 120-32 160-6c0 0 5 4 10-3c10-103 50-83 90-42
c0 0 20 12 30 7c-2 12-18 17-40 17c-55-2-40 25-20 35c30 20 35 65-30 71
c-50 4-170 4-200-79z"/>

is a compressed version of the (smooth) duck (a reduction of over 60%!)

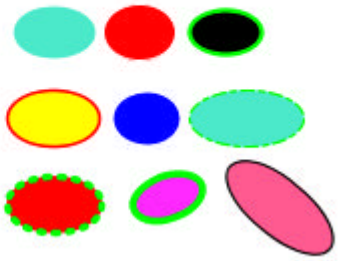
Smooth curves

Implicit definition of Bézier control points for adjacent curves

Path Examples

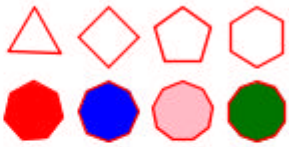


Path "Shortcuts"



<rect x=".." y=".." width=".." height=".."/>

<ellipse cx=".." cy=".." rx=".." ry=".."/>



<circle cx=".." cy=".." r=".."/>

<polyline points=".."/>
<line x1=".." x2=".." y1=".." y2=".."/>
<polygon points=".."/>

Text

- Is a separate element:

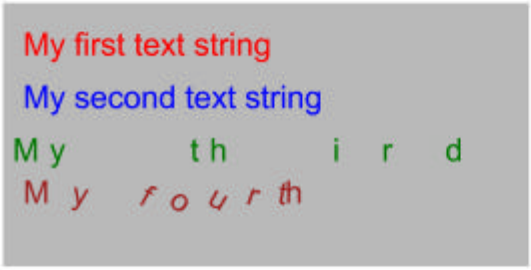
  <text x=".." y="..">A text</text>
- Has tons of attributes (most of come from CSS):
  - font, weight, font style, size
  - colour (both outline and inside a character)
  - writing mode, text anchor
  - ...
- Attributes, position, etc, can be changed with tspan
  - Most text on this slide is *one* text with lots of tspan-s!
- Text can also be drawn on a curve:

  <path id="path"> ...</path>

  <text><textPath xlink:href="#path"> ...</textPath></text>

Plain Text Examples

```
<text x="20" y="50">My first text string</text>
<text dx="20" dy="100">My second text string</text>
<text dx="10 10 100 10 10 100 40 50" y="150">My third</text>
<text dx="20 20 20 20 20 20 20 20" y="190" rotate="0 10 20 30 40 30 20 10 0">My fourth</text>
```



tspan Examples

```
<text x="10" y="50" >ABC <tspan y="30" DEF</tspan> this stays up!</text>
<text x="10" y="100">But it can
  <tspan dx="40" dy="-30" >easily</tspan>
  <tspan dy="30">be fixed</tspan>
</text>
<text > <tspan x="40 80 120 160 200 240 280 320 360" y="150"> Solid and </tspan>
<tspan x="60 100 140 180 220 260 300 340" y="200"> reliable </tspan> </text>
```

Text on Paths

```
<path id="MyPath" d="M 20 100
  C 100 20 180  -60 260 20
  C 340 100 420 180 500 100
  C 580 20 660 20 660 20" >

<text>
  <textPath xlink:href="#MyPath">
    We go up, then we go down, then up again
  </textPath>
</text>
```


```
<textPath xlink:href="#MyPath"  startOffset="40%">
```


```
We go <tspan dy="-30"> up</tspan>
<tspan dy="30">,</tspan>
then we go down, then up again
```

International Text

Polish: Mogę jeść szkło, i mi nie szkodzi.

Russian: Я могу есть стекло, это мне не вредит.

Greek: Μπορώ να φάω σπασμένα γυαλιά χωρίς να πάθω τίποτα.

Text "אני יכול לאכול זכוכית וזה לא מזיק לי" is in Hebrew

Yiddish: איך קען עסן גלאָז און עס טוט מיר נישט וויי.

Chinese: 我能吞下玻璃而不伤身体。

Japanese: 私はガラスを食べられます。それは私を傷つけません。

Arabic: إيفان هيرمان

unicode-bidi:bidi-override. First, direction:ltr, then direction:rtl.

Text "יל קיזמ אל הזו תיכוכז לוכאל לוכי ינא" is in Hebrew

werbeH ni si "אני יכול לאכול זכוכית וזה לא מזיק לי" txeT

Japanese: 私はガラスを食べられます。それは私を傷つけません。

Japanese: 私はガラスを食べられます。それは私を傷つけません。

This text "我能吞下玻璃而不伤身体。" is in Chinese

Inclusion of Images

Technology and Society domain

<image x="20" y="20" width="212" height="48" xlink:href="tands.png" />

Technology and Society domain

<image x="20" y="20" width="212" height="48" xlink:href="tands.svg" />

Architecture domain

<image x="20" y="40" width="848" height="48" xlink:href="arch.png" />

Web Accessibility initiative

<image x="20" y="40" width="212" height="192" xlink:href="wai.png" />

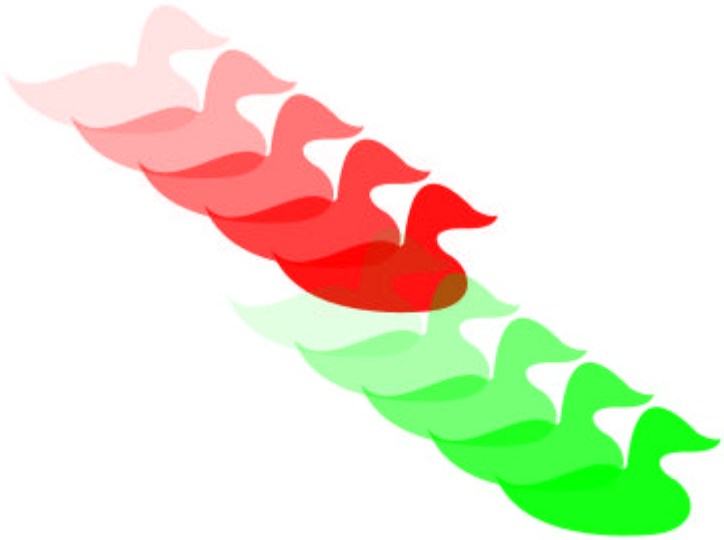
- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Rendering Model

- SVG uses the painters model for rendering
  - each operation "paints" on the output device successively
  - opacity level can influence final output ("alpha blending")
  - rendering order is implicit to the XML document
  - groups are rendered separately, then mapped on the output

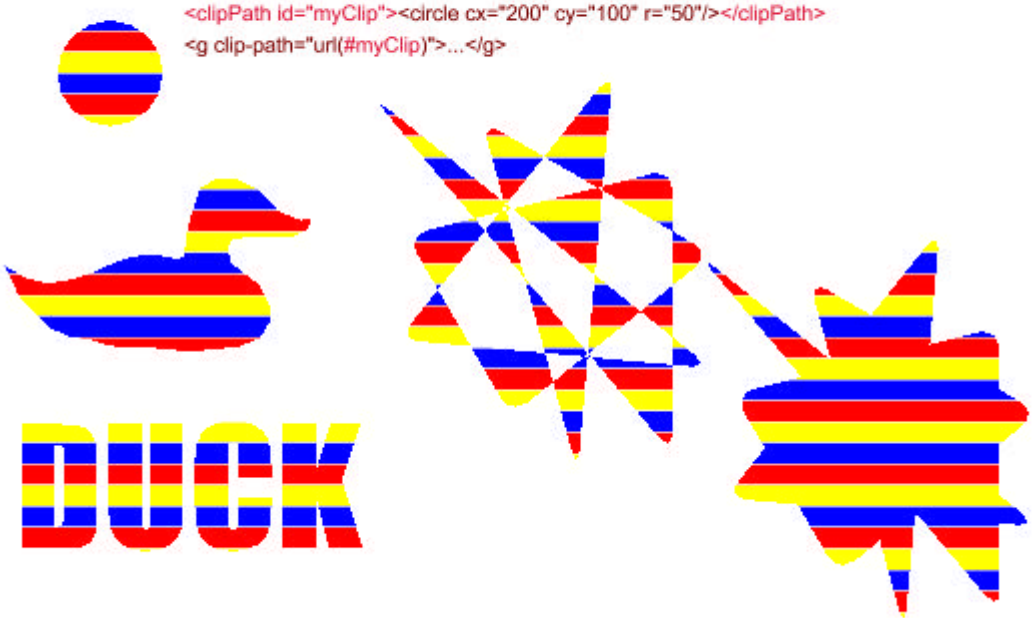
Opacity, blending

- opacity attribute can vary between 0 and 1
  - there are also separate fill-opacity and stroke-opacity attributes



Clipping

```
<clipPath id="myClip">
<circle cx="200" cy="100" r="50"/></clipPath>
<g clip-path="url(#myClip)">...</g>
```



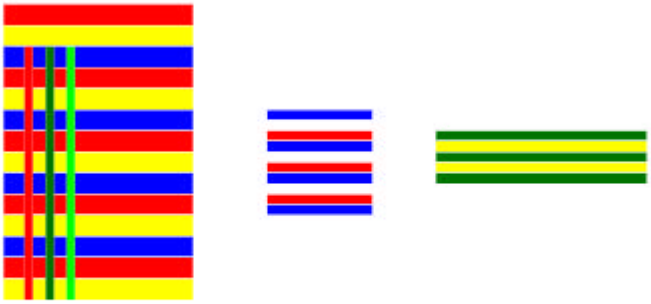
Clipping an Image

```
<clipPath id="duck"><path .../></clipPath>
<image xlink:href="wall.png" clip-path="url(#duck)"/>
```


The "objectBoundingBox" Attribute

- Clip path coordinate space adapted to the clipped object:

```
<clipPath id="myCliptxt" clipPathUnits="objectBoundingBox">
   <!-- the coordinates are in a (0,0,1,1) viewBox, set to the bounding box of the target -->
   <path d="M0 0.2 L0.5 1.0 L 1.0 0.8 L 0.5 0.0 z"/>
</clipPath>
<g clip-path="url(#myCliptxt)"><rect .... />...</g>
```



Masking

- Clipping draws with clip region transparent inside and opaque outside
- Masking draws *with opacity value taken from the mask*

Transparent rectangles with decreasing opacities:

Rectangles as Mask:


```
<mask id="Mask">
   <rect fill="white" fill-opacity="0.95" x="0"   y="30" width="100" height="80"/>
   <rect fill="white" fill-opacity="0.9"  x="100" y="30" width="100" height="80"/>
   . . .
</mask>
<text x="50" y="100" fill="blue" mask="url(#Mask)">Transparent rectangles in front of text</text>
```


Masking with an Image

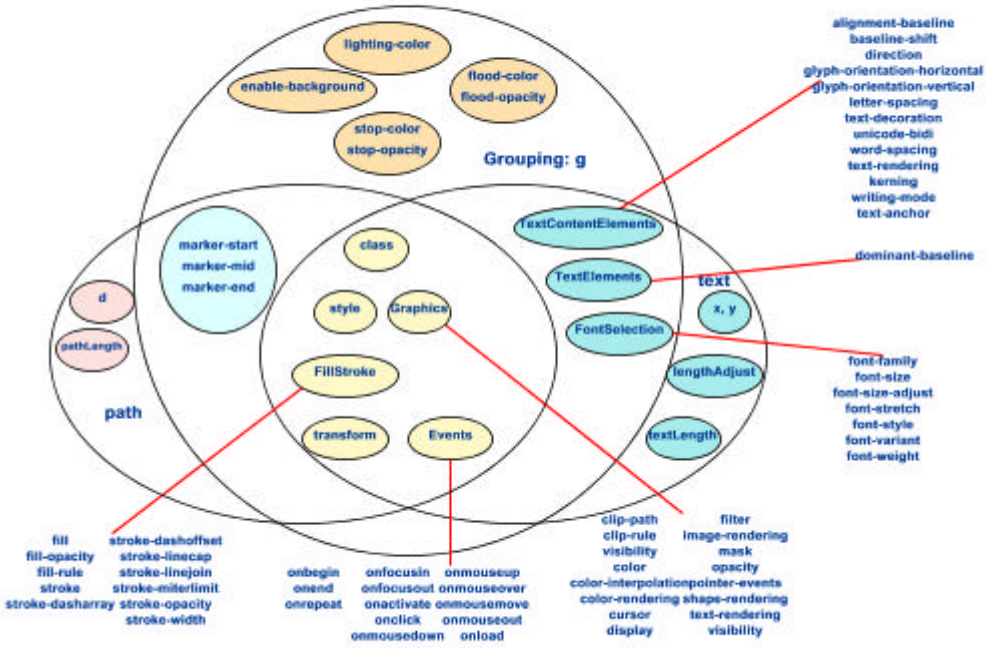
- A blue rectangle is drawn with the image as a mask
- Opacity (alpha) value is calculated for the image (data are in RGB):
  $\alpha = 0.2125R + 0.7154G + 0.0721B$

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

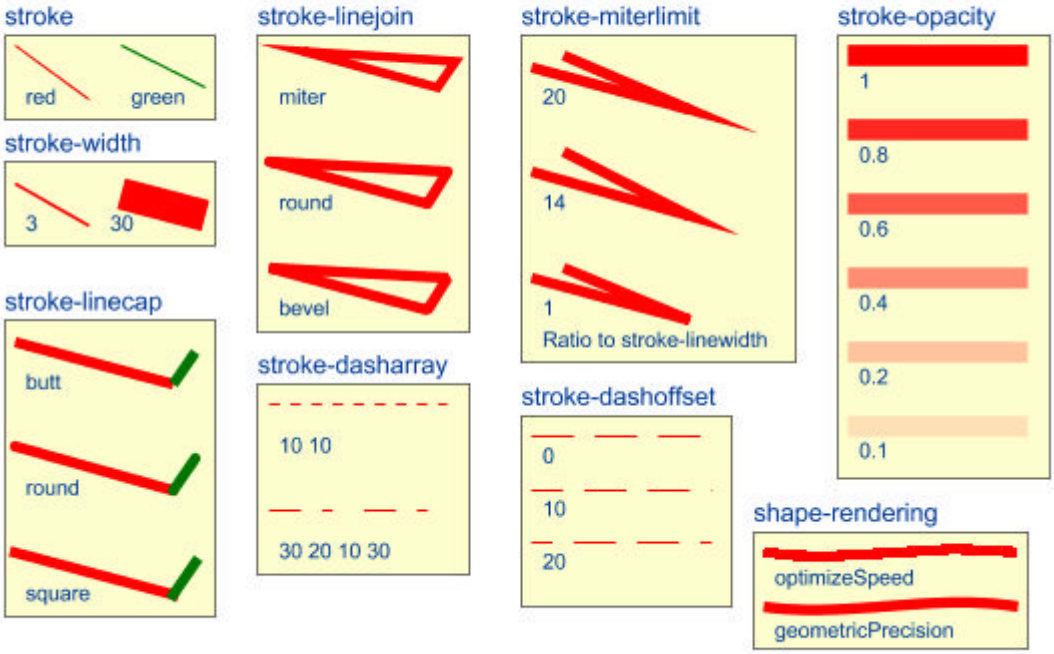
SVG Attributes

- The usual attributes set is available:
  - stroke attributes (line join, miter, dasharray, etc)
  - fill rule (even-odd, nonzero)
  - pattern fill (an image can also be a pattern!)
  - clipping, masking
- Colour
  - sRGB is the default
  - International Colour Consortium (ICC) colour profile can be chosen
- Contours can be filled with patterns or linear and radial colour gradients
- Opacity ("alpha value") is accessible separately

Overview of Attributes



Stroke Attributes

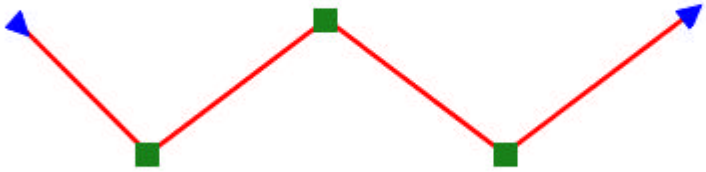


Dash line control

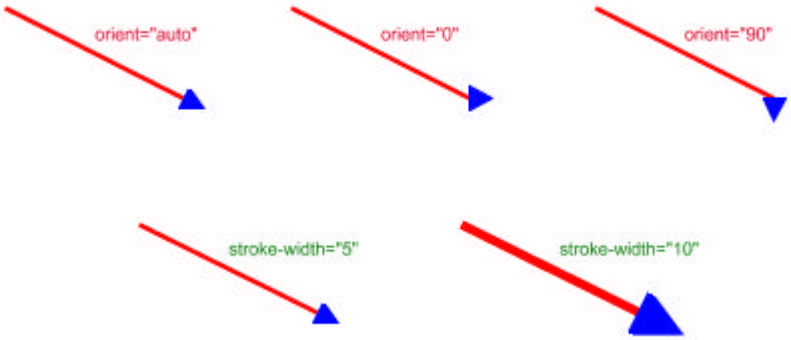
stroke-linecap applies to the individual dashes (round, square, butt)

```
<line x1="20" y1="20" x2="680" y2="20"
stroke-width="14" stroke="red" fill="none" stroke-linecap="round" stroke-dasharray="40 40" />
```

Marker Symbols



```
<marker id="Triangle" markerUnits="strokeWidth" viewBox="..." orient="auto"><path d="...."/></marker>
<marker id="Point" markerUnits="strokeWidth" viewBox="..." orient="0"><path d="...."/></marker>
<polyline marker-start="url(#Triangle)" marker-mid="url(#Point) marker-end="url(#Triangle) ... />
```

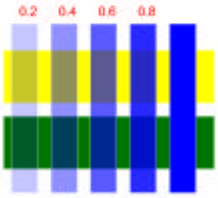


Area Attributes

Paint is used to fill and stroke

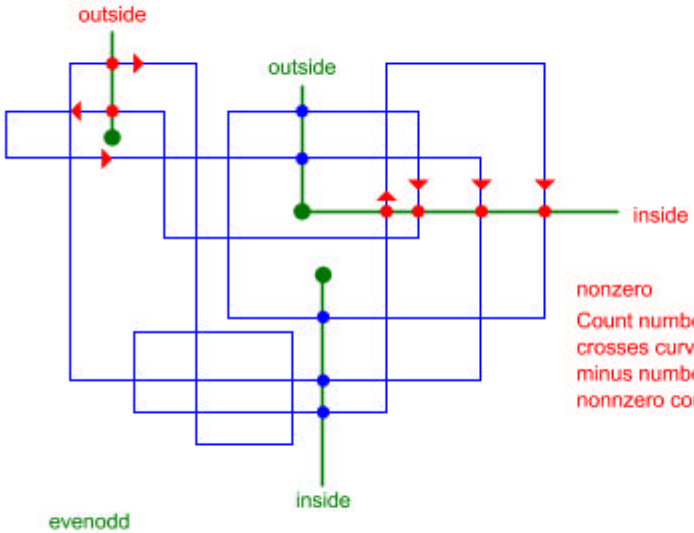
Paint can be a solid colour, a

gradient or a pattern



Fill properties include fill, fill-rule (evenodd, nonzero) and fill-opacity

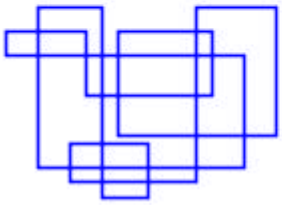
Fill Rules



nonzero
Count number of times line to infinity
crosses curve going clockwise
minus number going anticlockwise
nonnzero count means inside

evenodd
Point is inside if line to infinity
cuts the curve an odd number of times

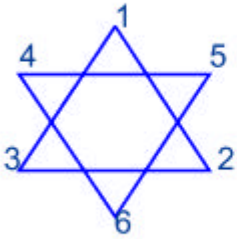
Fill Rule Examples



path

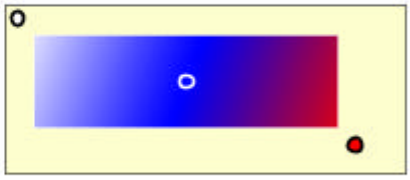
evenodd

nonzero

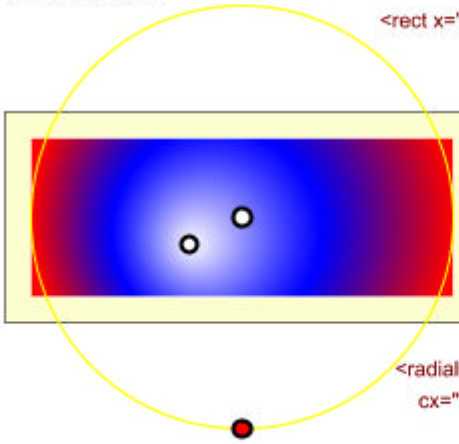


Gradient Fill Definitions

```
<linearGradient  id="Grad" gradientUnits="userSpaceOnUse"
   x1="0" y1="0" x2="40" y2="15">
      <stop offset="0" stop-color="white"/>
      <stop offset="0.5" stop-color="blue"/>
      <stop offset="1" stop-color="red"/>
</linearGradient>
```



```
<rect x="2" y="2" width="36" height="11" fill="url(#Grad)" />
```



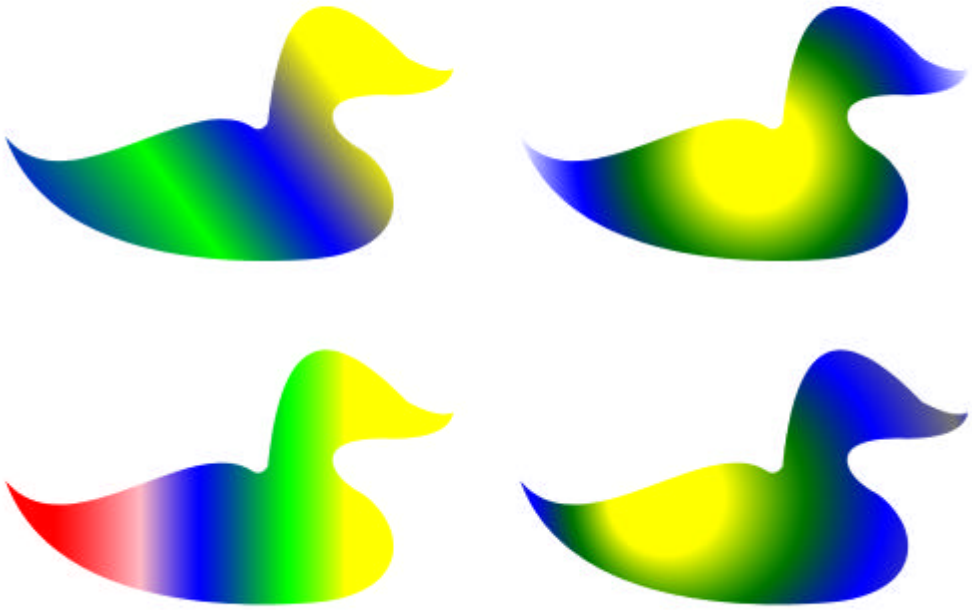
cx,cy,r define boundary (ie, offset=1)

fx,fy define focal point (ie, offset=0)

need at least 2 stops defined

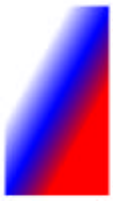
```
<radialGradient id="Grad" gradientUnits="userSpaceOnUse"
   cx="200" cy="75" r="200" fx="150" fy="100">
```

Gradient Fill Examples



Details of Gradient Specification

```
<linearGradient id="Gr" gradientUnits="objectBoundingBox" x1="0" y1="0" x2="1" y2="1">
   <stop offset="0" stop-color="white"/>
   <stop offset="0.5" stop-color="blue"/>
   <stop offset="1" stop-color="red"/></linearGradient>
<rect width="100" height="180" fill="url(#Gr)" />
```



```
<linearGradient  id="Gr2" gradientUnits="objectBoundingBox"
    spreadMethod="pad" x1="0.3" y1="0.3" x2="0.7" y2="0.7">
   <stop offset="0" stop-color="white"/>
   <stop offset="0.5" stop-color="blue"/>
   <stop offset="1" stop-color="red"/></linearGradient>
 <rect width="100" height="180" fill="url(#Gr2)" />
```

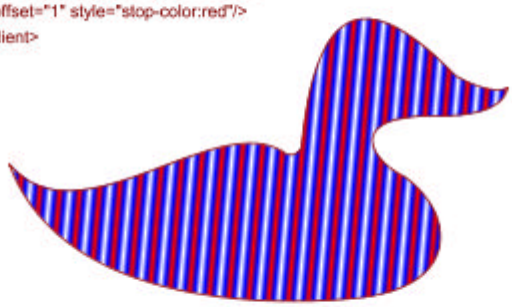
```
<linearGradient id="Gr3" gradientUnits="objectBoundingBox"
    spreadMethod="reflect" x1="0.3" y1="0.3" x2="0.7" y2="0.7">
   <stop offset="0" stop-color="white"/>
   <stop offset="0.5" stop-color="blue"/>
   <stop offset="1" stop-color="red"/></linearGradient>
<rect width="100" height="180" fill="url(#Gr3)"/>
```

spreadMethod="repeat"

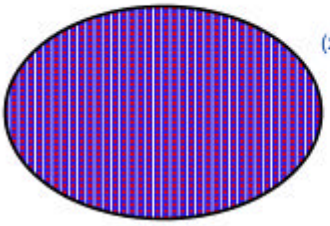
Transforming Gradients

An additional transformation can be applied on the gradient:

```
<linearGradient  gradientUnits="objectBoundingBox" gradientTransform="scale(0.25,4)"
  id="Gradient" spreadMethod="reflect"  x1="0.48" y1="0.48" x2="0.52" y2="0.52">
    <stop offset="0"  style="stop-color:white"/>
    <stop offset="0.5"  style="stop-color:blue"/>
    <stop offset="1" style="stop-color:red"/>
</linearGradient>
```



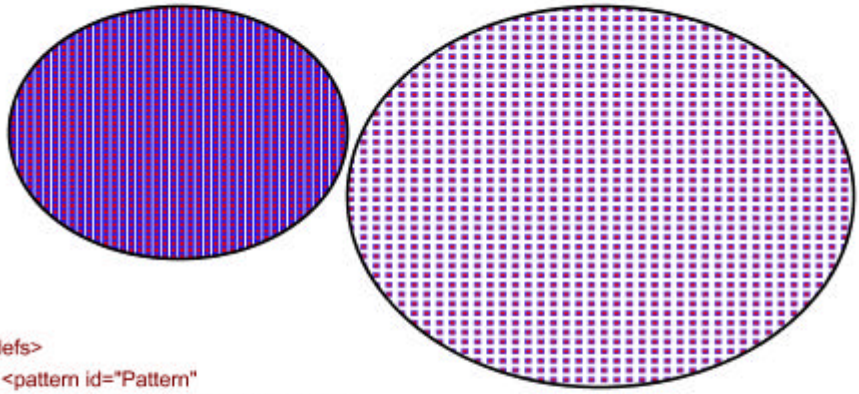
Pattern Fill




```
<defs>
  <pattern id="Pattern" patternUnits="userSpaceOnUse"
    x="0" y="0" width="8" height="6"  >
    <rect x="1" y="1" width="5" height="4" fill="red" stroke="blue"/>
  </pattern>
</defs>

<ellipse fill="url(#Pattern)" stroke="black" cx="200" cy="200" rx="150" ry="100"/>
```

Adaptation of a Pattern to an Area



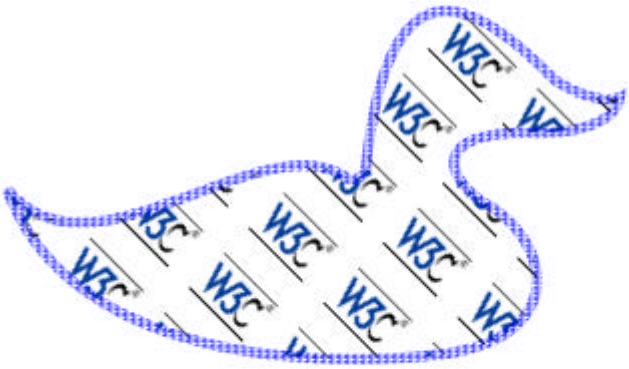
```
<defs>
  <pattern id="Pattern"
    patternUnits="objectBoundingBox"
    x="0%" y="0%" width="2.5%" height="2.5%">
      <rect x="1" y="1" width="5" height="4" fill="red" stroke="blue"/>
  </pattern>
</defs>
<ellipse fill="url(#Pattern)" stroke="black" cx="200" cy="120" rx="160" ry="120" />
<ellipse fill="url(#Pattern)" stroke="black" cx="600" cy="180" rx="240" ry="180" />
```


Image as a Pattern

```
<pattern id="P" patternUnits="userSpaceOnUse"
  x="0" y="0" width="100" height="80">
    <image x="0" y="0" width="72" height="48" xlink:href="w3chome.png"/>
</pattern>
```

Patterns can be Transformed

```
<pattern id="P" patternUnits="userSpaceOnUse" patternTransform="rotate(45)"
   x="0" y="0" width="100" height="80">
      <image x="0" y="0" width="72" height="48" xlink:href="w3chome.png"/>
</pattern>
```



Stroke/Fill Attributes Defaults

- For stroke
  - stroke: none (except for line, which is black)
  - stroke-width: 1
  - stroke-linecap: butt
  - stroke-linejoin: miter
  - stroke-dasharray: none
  - stroke-opacity: 1
- For fill
  - fill: black
  - fill-rule: nonzero
  - fill-opacity: 1

Text Attributes

font-family
monospace
serif

font-size
20
40

font-stretch
normal
wider
narrower
ultra-condensed
extra-condensed
expanded

font-style
normal
italic
oblique

text-decoration
underline
overline
~~line-through~~

font-weight
normal
bold
bolder
lighter
100
900

text-anchor
start
middle
end

alignment-baseline
The baseline
The middle
The top

baseline-shift
$x^{super}+y_{sub}+1^{100\%}$

fill
red
green
blue
none

stroke
red
green
blue

writing-mode
lr (left to right)
rl (right to left)
tb (top to bottom)

glyph-orientation-vertical (tb)
value:auto
value:270
value:90
value:0

glyph-orientation-horizontal (lr)
value:0
value:90
value:180
value:270

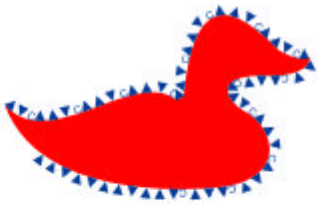
Font Selection

- Use font name
  - can be specific: font-family="Helvetica"
  - use generic name (defined in CSS): font-family="sans-serif"
  - list several names, let the user agent choose: font-family="Helvetica, sans-serif"
  - but... specific fonts are not always available!
- Refer to explicit, downloadable fonts ("Web fonts")
  - there is a detailed specification in CSS2
  - the CSS2 rule @font-face can be used to import a font
  - but... CSS2 does not require availability of a specific format
- SVG also defines its own *SVG fonts*
  - font specfication can be embedded in the SVG file
  - can be stored in a separate file, and referred to via @font-face (in CSS) or font-face-uri
  - very small fonts for 2-3 characters can also be defined (useful for, eg, company logos)

SVG Font Example

```
<defs>
  <font ... >
    <font-face font-family="MyOwn" units-per-em="1000"/>
    <missing-glyph horiz-adv-x="1300">
      <circle r="440" cx="500" cy="500"/>
    </missing-glyph>
    <glyph unicode="A" horiz-adv-x="1300">
      <path d="M0 0 L 500 1000 L1000 0 L 0 0"/>
    </glyph>
    <glyph unicode="B" horiz-adv-x="1300">
      <path d="M0 1000 L 500 0 L1000 1000 L 0 1000"/>
    </glyph>
  </font>
  <path id="duck" d="..."/>
</defs>
...
<text>This is "ABCA" using MyFont: <tspan font-family="MyOwn">ABCA</tspan></text>
...
<use xlink:href="#duck" fill="red"/>
<text font-family="MyOwn" ><textPath xlink:href="#duck">...</textPath></text>
```

This is "ABCA" using MyFont: ▲▼ C▲



SVG Fonts Example from the Web

Larabiefont Bold

a Ray Larabie For

Sample

The quick brown fox

The quick brown fox

The quick brown fox

Code Example

```
<font-face font-family="Larabiefont" font-weight="bold">
  <font-face-src>
   <font-face-uri xlink:href="http://nagoya.apache.org/svgfonts/larabieb.svg#fontDef"/>
  </font-face-src>
</font-face>

<text font-family="Larabiefont" font-weight="bold" >My Text using SVG Font</text>
```

Named Colours

aliceblue		darkgoldenrod		floralwhite		lightcyan		mediumslateblue		pink		thistle	
antiquewhite		darkgray		forestgreen		lightgoldenrodyellow		mediumspringgreen		plum		tomato	
aqua		darkgreen		fuchsia		lightgray		mediumturquoise		powderblue		turquoise	
aquamarine		darkgrey		gainsboro		lightgreen		mediumvioletred		purple		violet	
azure		darkkhaki		ghostwhite		lightgrey		midnightblue		red		wheat	
beige		darkmagenta		gold		lightpink		mintcream		rosybrown		white	
bisque		darkolivegreen		goldenrod		lightsalmon		mistyrose		royalblue		whitesmoke	
black		darkorange		gray		lightseagreen		moccasin		saddlebrown		yellow	
blanchedalmond		darkorchid		grey		lightskyblue		navajowhite		salmon		yellowgreen	
blue		darkred		green		lightslategray		navy		sandybrown	
blueviolet		darksalmon		greenyellow		lightslategrey		oldlace		seagreen	
brown		darkseagreen		honeydew		lightsteelblue		olive		seashell	
burlywood		darkslateblue		hotpink		lightyellow		olivedrab		sienna	
cadetblue		darkslategray		indianred		lime		orange		silver	
chartreuse		darkslategrey		indigo		limegreen		orangered		skyblue	
chocolate		darkturquoise		ivory		linen		orchid		slateblue	
coral		darkviolet		khaki		magenta		palegoldenrod		slategray	
cornflowerblue		deeppink		lavender		maroon		palegreen		slategrey	
cornsilk		deepskyblue		lavenderblush		mediumaquamarine		paleturquoise		snow	
crimson		dimgray		lawngreen		mediumblue		palevioletred		springgreen	
cyan		dimgrey		lemonchiffon		mediumorchid		papayawhip		steelblue	
darkblue		dodgerblue		lightblue		mediumpurple		peachpuff		tan	
darkcyan		firebrick		lightcoral		mediumseagreen		peru		teal	

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Styling in Text Processing

- Separation of *style* and *content*:
  - TeX/LaTeX style files
  - templates for MsWord files
  - styles for FrameMaker
  - *cascading style sheets* for HTML/XML
- Has great advantages:
  - ease of maintenance
    - changing the colour of an element should be made only once
  - house/corporate style control
  - clarity of structure
  - adaptation to the environment
    - different styles for different viewer engines
    - adaptation to users with disabilities
  - design control
    - styles are often made by design professionals

Styling in Graphics

- Similar ideas can be used for graphics
  - control the fonts for all the text on these slides through style
  - control colours in a set of schematic diagrams
- But...what is style in graphics?
  - colour, font style, dash pattern, etc, may be style, but...
  - font used in a logo is *not* style, it is *content*
  - colour showing a traffic light is *not* style, it is *content*

There is a need for a dual control over content and style

- It is not a new idea:
  - GKS:85 and PHIGS had the concept of "attribute bundling"
  - PHIGS and GKS:94 had a nameset mechanism for a finer control
- SVG uses both direct attribute setting and styling
  - Styling uses other W3C technologies, eg, CSS

SVG Styling

- Simple styling example:

```
<rect fill="yellow" stroke="black" x=".." y=".." width=".." height=".."/>
<text font="Helvetica" font-size="14" x=".." y="..">...</text>
```

- Equivalent CSS syntax:

```
<rect style="fill:yellow; stroke:black" x=".." y=".." width=".." height=".."/>
<text style="font:Helvetica; font-size:14" x=".." y="..">...</text>
```

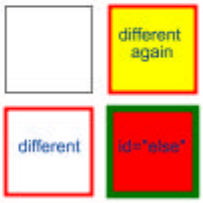
- Separation into separate style:

```
<svg ...>
  <style type="text/css"><![CDATA[
    rect.example {fill:yellow; stroke:black }
    text.example {font:Helvetica; font-size:14 }
  ]]><style>
  ...
  <rect class="example" x=".." y=".." width=".." height=".."/>
  ...
  <text class="example" x=".." y="..">...</text>
</svg>
```


SVG Styling 2

- Class selector mechanism is quite powerful:

```
<style type="text/css"> <![CDATA[
  rect { stroke:black; fill:white }
  rect.different { stroke:red; stroke-width:4; fill:none }
  rect.again { fill:yellow }
  rect[id ~= "else"] { stroke:green; stroke-width:8; fill:red}
]]> </style>
<rect x="20" y="20" ... />
<rect class="different" x="20" y="140" ... />
<rect class="different again" x="140" y="20" ... />
<rect id="else" x="140" y="140" width="100" height="100" />
```



  - The W3C CSS2 Recommendation contains all the rules
- Separation into separate style file:

style.css file contains:

```
rect.example {fill:yellow; stroke:black }
text.example {font:Helvetica; font-size:14 }
```

the svg file itself is:

```
<?xml-stylesheet type="text/css" href="slide.css/>
<svg ...>
  ...
  <rect class="example" x=".." y=".." />
  <text class="example" x=".." y="..">...</text>
</svg>
```

Styling Priorities

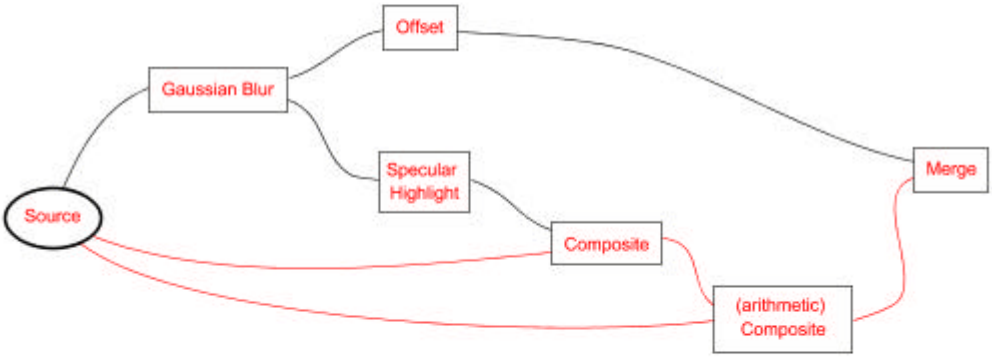
- The "cascade" of styling priorities (in increasing order):
  - Default values (eg, fill is black)
  - Attribute settings (eg, < ... fill="red" ... />)
  - CSS settings in the *included* style file
  - CSS settings in the local <style> section
    - if two rules refer to the same element, the latter specified wins
  - CSS settings on the element (eg, < ... style="fill:red" ... />)
- Practical advise: do not mix direct attribute settings and CSS!

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Filters

- Graphics on the Web is often used for "artwork"
  - Complex logos
  - "Sexy" presentations (shadow effects, pseudo 3D)
  - Advertisements...
- Images (jpeg, png, gif, etc) *can* be included in SVG
  - but this breaks the goals...
- *Filters* have been added to SVG to produce artwork:
  - a collection of *15 filter primitives:*
    - combined in a dataflow network
    - applied *after* all rendering operations, but *before* display
    - operating either on the alpha or the RGB channel
  - usually operate on a group of primitives

Filter Details



Coding Filters in SVG

```
<defs>
  <filter id="DropShadowFilter">
    <feGaussianBlur in="SourceAlpha" stdDeviation="2" result="BlurAlpha"/>
    ...
    <feMerge>
      <feMergeNode in="OffsetBlurAlpha"/>
      <feMergeNode in="SourceGraphic"/>
    </feMerge>
  </filter>
</defs>
...
<g filter="url(#DropShadowFilter)">
  <path d="...">
  ...
</g>
```


(Approximate) Overview of Filters

- Blending, compositing, merging
  - pixel-wise combination of two images
- Colour manipulation
  - brightness, contrast adjustment, colour thresholding
  - direct colour matrix manipulation
- Convolution (blurring, sharpening, etc)
- Diffuse and specular lighting
  - operate on the alpha channel as a bump map, result is an opaque RGB image
    - result colour depends on the light properties
  - separate elements to control light source elements and properties: distant light, point light, spot light
- Displacement map
  - displace pixels in one image under the control of another
- Offset
- Fattening/thinning (ie, dilation or erosion)
- Tiling
- Generation of artifical textures (turbulence functions)

Some Examples for Filtering

Full 3D Effect (like before):

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

SVG Animation

- SVG Animation = change object attributes dynamically
- Animation is controlled through a set of animation objects
  - Reusing parts of W3C's SMIL2.0 Specification
    (Synchronized Multimedia Integration Language Version 2.0)
- Declarative syntax, no real increase in file size
- Animation is performed on *client side*
- Aspects to describe:
  - *What* can be animated?
  - *How* does animation take place?
  - *When* does animation occur?

SVG Animation Example

Technology and Society domain

Web Accessibility initiative

W3C WORLD WIDE WEB consortium

Document Formats domain

Interaction domain

Architecture domain

What is animated?

- Practically all attributes can be changed
  - XML: coordinates, path expression, filter characteristics, ...
  - CSS: colour, visibility, opacity, ...
- Separate animation elements
  - animate
  - animateMotion: motion along a path
  - animateColor
  - animateTransform: controls transformation (rotate, translate, etc)
  - set: changes discrete values (eg, visibility)
- Examples:

```
<circle id="c1" r="50" style="fill:red">
  <animate attributeType="XML" attributeName="r" from="50" to="140" ...>
</circle>
```

```
<circle id="c1" r="50" style="fill:red; opacity:1">
  <animate attributeType="CSS" attributeName="opacity"
    from="1" to="0" ...>
</circle>
```

Path Deformation

```
<animate attributeName="d"
  from="M 20 100 c 40 48 120 -32 160 -6 c 0 0 5 4 10 -3 c..."
  to="M 80 100 c 40 48 120 -2 160 -36 c 0 0 5 -11 10 -18 c..."
</animate>
```

Only the coordinates can be changed!

(ie, one cannot change a line to a curve)

: DDLQJ 0 DQ

Animate Transformation

- Several animation elements can be applied concurrently
- The additive="sum" attribute ensures the accumulation of the effects

```
<circle . . .
   <animateTransform attributeName="transform" type="rotate"
      from="0" to="360" dur="1s" repeatDur="indefinite"/>
   <animateTransform attributeName="transform" type="scale"
      from="1,1" to="3,0.3" additive="sum" dur="60s" repeatDur="indefinite"/>
</circle . . .
```

Animate Motion

```
<path d="M0 0 v -5 h20 v-10 l10 15 l-10 15 v-10 h-20 v-5" ...
  <animateMotion path="M20 100 ..."
    rotate="auto"
    rotate="auto-reverse"
    rotate="45"
    rotate="90"
  </animateMotion>
</path>
```


Animate Colour

```
<rect x="10"  y="50"  width="150" height="75"  >
   <animateColor  attributeType="CSS" attributeName="fill"
      from="aqua" to="crimson"
      begin="0s" dur="5s" fill="freeze">
</rect>
```

Instanteneous Change: "set"

```
<rect x="10"  y="50"  width="150" height="75"  style="visibility:visible">
   <set attributeType="CSS" attributeName="visibility" to="hidden"
         begin="1s;2s;3s;4s;5s;6s"/>
   <set attributeType="CSS" attributeName="visibility" to="visible"
         begin="1.5s;2.5s;3.5s;4.5s;5.5s"/>
</rect>
```


How is animation performed?

- By default, animation is linear
  - the [from,to] interval is interpolated linearly over required duration
- Alternative possibilities:
  - Discrete jumps (no interpolation)
  - Specify intermediate (key) values and time
  - Replace linear interpolation by (cubic) splines (in [0,1])
  - Key times and spline interpolation can be combined

Key Values

```
<animate attributeName="cx" values="15;615" .../>
```

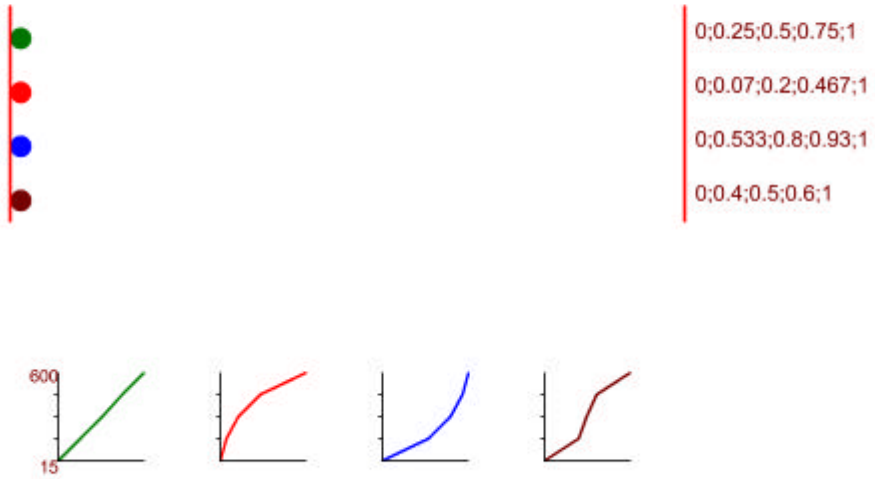
Key Values example

```
<circle  - - r="20" style="fill:blue">
   <animate attributeName="r" values="10;100;10;100;10" ... />
</circle>
<circle cx="0" cy="400" r="10" style="fill:green">
   <animate attributeName="cy" values="400;200;100;300;0;50;250;150;350;400" .../>
   <animate attributeName="cx" values="0;200;100;300;80;400;400;250;300;0"  .../>
</circle>
```

Key Values and Times

```
<animate attributeName="cx" values="15;165;315;465;615"
                            keyTimes="0;0.25;0.5;0.75;1 .../>
```

Time for each interval specified as a fraction of total time



Duck Race...

Fast acceleration, poor brakes, same top speed as red duck

Acceleration and brakes about the same for the red duck

Fast acceleration, poor brakes, higher top speed than red duck

```
values="0;2;8;18;32;96;224;352;384;440;480;508;512"
keyTimes="0;0.0238;0.0476;0.071;0.0952;0.19;0.381;0.571;0.619;0.714;0.8095;0.9048;1"
```

Time Interpolation Modes

```
<animate attributeName="cx" values="15;45;615" calcMode="linear" .../>
```

(for simplicity, key times are not used)

linear

discrete

paced

spline

linear: time split equally between intervals
discrete: jumps between defined values
paced: even pace of change across values
spline: gives finer control over the motion in each interval

Key Splines

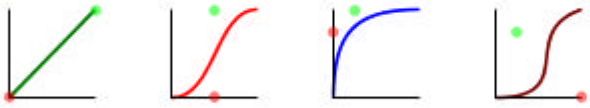
```
<animate attributeName="cx" values="15;615" keyTimes="0;1" keySplines="0 0 1 1".../>
```

0 0 1 1

.5 0 .5 1

0 .75 .25 1

1 0 .25 .25



Bouncing balls

linear speed

When is animation performed?

- Simple case: *time-based animation*
  - begin, dur, end attributes
  - values are in time (with "t = 0" value at loading)
  - all kinds of time units are available
- Complicated case: *event-based animation*
  - begin attribute can refer to *events*, eg:
    - obj.click, when a click occurs on obj
    - anim.end or anim.begin, when anim ends, resp. begins
    - anim.end; anim2.end, when anim *or* anim2 ends
    - anim.begin+4s 4 seconds after anim begins
    - obj.mouseover, obj.mouseout when mouse moves over/out obj
    - ...
- Animation can be repeated
- Animation effects can be "frozen"
- Detailed semantics is defined in a separate W3C document (SMIL Animation, usable for any XML application)

Chaining Animations

```
<animate attributeName="cx" from="50" to="200" fill="freeze"
        id="first" xlink:href="#c1" dur="1s" begin="1s" />
```

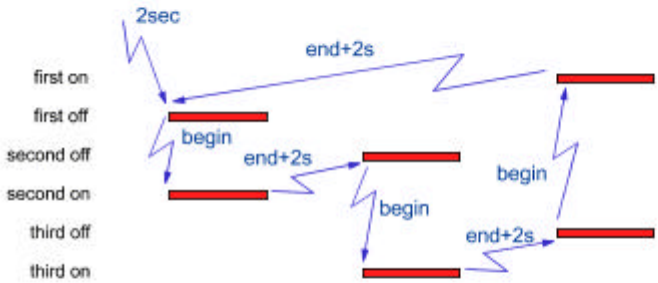
```
<animate .. id="second" xlink:href="#c2" dur="1s" begin="first.begin" />
```

```
<animate .. id="third" xlink:href="#c2" dur="1s" begin="first.end" />
```

```
<animate .. id="fourth" xlink:href="#c2" dur="1s" begin="first.end+1s" />
```

This has no repeatCount: it goes on forever!

Multimedia Show...

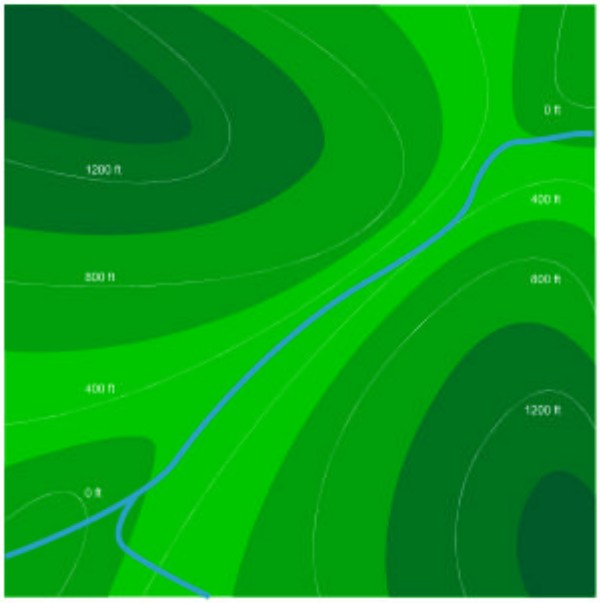


Control with Mouse Events

```
<circle id="click" . . style="fill:grey" />
<circle id="overout" . . style="fill:grey" />
<circle id="downup" . . style="fill:grey" />
<circle style="fill:red" id="c1" cx="50" cy="100" r="50" />
```


```
<animate xlink:href="#c1" begin="click.click" attributeName="cx" values="50;100;50" dur="2s"/>
<animate xlink:href="#c1" begin="overout.mouseover" attributeName="cx" from="50" to="150"/>
<animate xlink:href="#c1" begin="overout.mouseout" attributeName="cx" from="150" to="50" />
<animate xlink:href="#c1" begin="downup.mousedown" attributeName="cy" from="100" to="200"/>
<animate xlink:href="#c1" begin="downup.mouseup" attributeName="cy" from="200" to="100"/>
```


Cartography Example



Topography:

- Labels
- Streets
- Points of Interest
- Freeways
- Legend

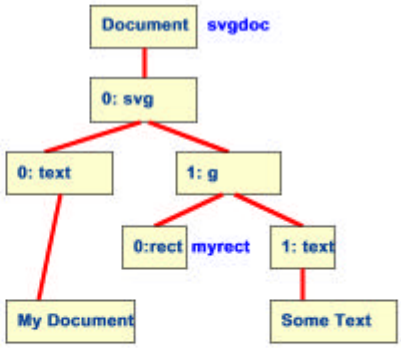
- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

SVG DOM

- DOM = Document Object Model
  - programmatic interface to the runtime XML tree
    - interface definition in OMG's IDL
    - language bindings in Javascript, Java, C++
  - a general W3C mechanism with an SVG specialization
- Scripts can be included in SVG (through a script node)
- Scripts can modify:
  - the XML tree (add and/or delete nodes)
  - the attributes of individual objects
- Scripts can be used for animation and interaction:
  - more powerful than animation objects, but...
  - more complicated and error prone to write
  - require additional interpreters on the client-side

DOM Tree

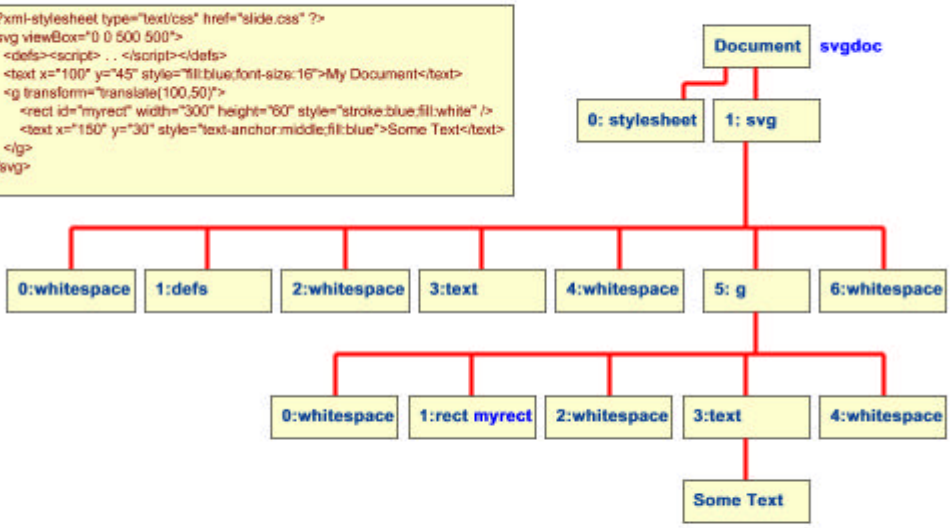
```
<svg viewBox="0 0 500 500">
   <text x="100" y="45" style="fill:blue;font-size:16">My Document</text>
   <g transform="translate(100,50)">
      <rect id="myrect" width="300" height="60" style="stroke:blue;fill:white" />
      <text x="150" y="30" style="text-anchor:middle;fill:blue">Some Text</text>
   </g>
</svg>
```




```
svgdoc.childNodes[0].childNodes[1].childNodes[0].getStyle().setProperty("fill", "pink");
svgdoc.getElementById("myrect").getStyle().setProperty("fill", "lightgreen");
```


But it is not that simple...

```
<?xml-stylesheet type="text/css" href="slide.css" ?>
<svg viewBox="0 0 500 500">
   <defs><script> . . </script></defs>
   <text x="100" y="45" style="fill:blue;font-size:16">My Document</text>
   <g transform="translate(100,50)">
      <rect id="myrect" width="300" height="60" style="stroke:blue;fill:white" />
      <text x="150" y="30" style="text-anchor:middle;fill:blue">Some Text</text>
   </g>
</svg>
```




```
svgdoc.childNodes[1].childNodes[5].childNodes[2].getStyle().setProperty("fill", "pink");
```

SVG DOM Objects and Methods

- General DOM methods (inherited by SVG):
  - getOwnerDocument(), getNodeName(), getNodeType(), getChildNodes(),...
  - getPreviousSiblings(), getAttributes(), getAttributes().item(0),...
  - getLength(), item()
  - getElementsByTagName(), getElementById(),...
  - getAttribute(), setAttribute(), hasAttribute(),...
- SVG Specific:
  - SVGDocument object (subcless of Document)
    - refers to title, URL, etc, has a reference to the root svg element
  - SVGTranform object
    - setMatrix(), setTranslate(), setRotate(), etc, methods
  - SVGPathElement object
  - etc

Interacting with an Element

- Interacting directly:

```
< id="myrect" onclick="interact(evt)"... >
  function interact(evt) {
    svgobj = evt.target;
    svgobj.setAttribute('fill','red');
    svgstyle = svgobj.getStyle();
    svgstyle.setProperty('opacity',0.5);
    ...
```

- Interacting elsewhere from a script:

```
svgdoc = evt.getCurrentNode.getOwnerDocument;
svgobj = svgdoc.getElementById('myrect');
```

Example: Interaction Through "onclick"

```
<script type="text/ecmascript">
  function changerect(evt)
  {  var v;
     var svgobj = evt.target;
     svgstyle = svgobj.getStyle();
     svgstyle.setProperty('opacity', 0.3);
     v = svgobj.getAttribute('x');
     v = v*1+50;
     svgobj.setAttribute('x',v);
  }
</script>
<rect onclick="changerect(evt)" style="fill:blue;opacity:1" x="10" y="0" width="50" height="30"/>
<rect onclick="changerect(evt)" style="fill:red;opacity:1" x="10" y="40" width="50" height="30"/>
```

svgobj set to the rect object that was the target (clicked)

svgstyle set to the style attribute

evt is the event object passed to the function

Reacting on "mouseover"

```
<g onmouseover="DoOnMouseOver(evt)" onmouseout="DoOnMouseOut(evt)"  - - -
   <circle cx="125" cy="325" r="50" id="circle" style="visibility: visible; fill: green"/>
   <text x="200" y="325" id="circletext" style="visibility: hidden; ">Mouse Over</text>
</g>
```

```
function DoOnMouseOver(event) {
   if( event.getTarget().getAttribute("id") == "circle" ) {
      el = doc.getElementById("circle");
      el.getStyle().setProperty("fill", "blue");
      el = doc.getElementById("circletext");
      el.getStyle().setProperty("visibility", "visible");
   }
}
```

The value of doc is established on load

Similar effect could be achieved without scripting, using animation objects...

Modifying the DOM Tree

```
function addrect(evt)
{
   var svgobj=evt.target;
   var svgdoc = svgobj.getOwnerDocument();
   var newnode = svgobj.cloneNode(false);
   svgstyle = newnode.getStyle();
   svgstyle.setProperty ('fill', 'red');
   newnode.setAttribute ('x', '50');
   var contents = svgdoc.getElementById ('contents');
   newnode = contents.appendChild(newnode);
}
function removerect(evt)
{
   var svgobj=evt.target;
   var g=svgobj.getParentNode();
   g.removeChild(svgobj);
}
```

```
<g id="contents">
   <rect onclick="addrect(evt)" style="fill:blue" x="10" y="100" width="20" height="20" />
   <rect onclick="removerect(evt)" style="fill:green" x="10" y="200" width="20" height="20" />
</g>
```


Event List

- Pointer events
  - onfocusin, onfocusout, onactivate, onclick, onmousedown, onmouseup
  - onmouseover, onmousemove, onmouseout
- Window events
  - onload, onunload, onabort, onresize, onscroll, onzoom
- Animation object events
  - onbegin, onend, onrepeat
- When using in animation objects' begin, the "on" should be dropped
  - eg, <animate ... begin="i1.begin; i2.click" ... />

Interaction by Modifying DOM Tree

Moving Box

```
svgobj.setAttribute ('x', target.getAttribute('x'));
svgobj.setAttribute ('y', target.getAttribute('y'));
```

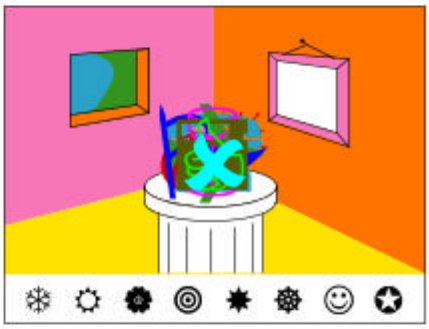
Rearranging Objects

```
var groupnode = target.getParentNode();
groupnode.removeChild (target);
groupnode.insertBefore (target, groupnode.getFirstChild());
```

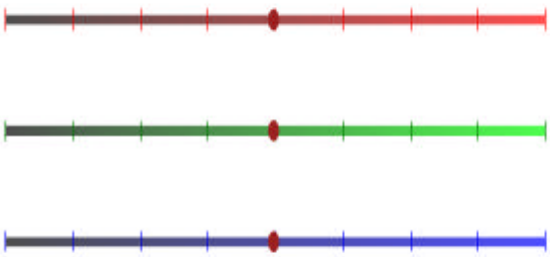
Cloning Object

Randomly choose location and colour

```
var newnode = target.cloneNode(false);
newnode.setAttribute ('x', x);
newnode.setAttribute ('y', y);
newnode.getStyle().setProperty ('fill', fill);
newnode = contents.appendChild (newnode);
```



Colour Selection Example



rgb(127,127,127)

Nearest SVG colour by name: gray

rgb(127,127,127)

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Miscellaneous

- Hyperlinking
  - *All* elements can be enclosed in a a element, much like in HTML
  - Through "fragment identifiers" one can refer to details of another SVG file:

```
<a xlink:href="Drawing.svg#details">...</a>
<a xlink:href="Drawing.svg#svgView(viewBox(10,10,100,100))">...</a>
```

- Control over zooming and panning
  - they can be disabled for a specific svg file
- Conditional Processing
  - A switch element can be used to separate branches, using:
    - requiredFeatures: describes features implemented (or not) by the client
    - requiredExtensions: describes extensions provided (or not) by the client
    - systemLanguage: inherited from xml
  - Examples:

```
<switch>
  <text systemLanguage="fr">Bonjour!</text>
  <text systemLanguage="en">Good morning!</text>
</switch>
...
<rect requiredFeatures="org.w3c.svg.dynamic" visibility="hidden" ... />
```

Metadata

- SVG can *include* information about itself:
  - title: provides a title for a graphics element
  - desc: textual description, may be added to *all* elements
- The metadata section can contain *any sort of metadata (eg, RDF)*
  - describe the image content for visually impaired users
  - include authentication, encrypted signatures, ...
  - describe library information (eg, using the Dublin Core)

```
<svg id="ThisSVG" ... >
  <metadata> <!-- Using the Dublic Core vocabulary -->
    <rdf:RDF xmlns:rdf="..." xmlns:dc="...">
      <rdf:Description rdf:about="#ThisSVG/">
        <dc:title>Presenting SVG</dc:title> <dc:creator>Ivan Herman</dc:creator>
        <dc:publisher>World Wide Web Consortium</dc:publisher>
        <dc:date> 2001-10-29</dc:date> <dc:format>image/xml+svg</dc:format>
      </rdf:Description>
    </rdf:RDF>
  </metadata>
  ...
</svg>
```


- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

SVG in Practice

- SVG files can be accessed directly
  - MIME Type is image/svg+xml
  - usual extensions are svg or svgz (the latter refers to a gzip compressed file)
    - servers have to be set up accordingly!
- Inclusion into HTML (with a plugin):

```
<object width="350" height="350" data="FileName.svg" type="image/svg+xml"></object>
```

  - embed also works, but is deprecated in HTML4 and XHTML
  - all kinds of units (mm, cm, inch, etc) can also be used, not only dimensionless size
- Access to the DOM from an HTML document:
  - the reference to the SVG content is, eg,:

```
<embed src="file.svg" name="SVGEmbed" ... >
```

  - in the script *in HTML*:

```
document.SVGEmbed.getSVGDocument().getElementById("myrect")....;
```

  - It should work properly with <object>, but it does not always...

Better (Future) Integration

- Inclusion into any XML application with namespaces (eg, XHTML)

```
<?xml version="1.0" standalone="yes"?>
<parent xmlns="http://example.org" xmlns:gr="http://www.w3.org/2000/svg">
  <parentElement>....</parentElement>
  <anotherParentElement>
    <gr:svg ... >
      <gr:rect>....</gr:rect>
    </gr:svg>
  </anotherParentElement>
</parent>
```

  - works in W3C's Amaya and in X-Smiles (but only partial implementations of SVG)
  - there is also a Mozilla sub-project to include SVG

Implementations

- Players, plugins:
  - Adobe's plugin, works with all major browsers on Win, Mac (Linux and Solaris in beta)
  - Various standalone players, mostly in Java (CSIRO and Batik toolkits, X-Smiles, Ipaq, ...)
  - Onging work for the inclusion of SVG into Mozilla

  None of these players implement 100% yet, but we are getting there!
- Authoring tools:
  - Export facilities from Adobe Illustrator, Corel Draw, Mayura, Quark Xpress, OpenOffice, ...
  - "Pure" SVG drawing tools (eg, Jasc)
  - Animation editors (eg, Sphinx, IMS Web Engine, ...)
  - Conversion tools (eg, Square1's PS to SVG, KK Software's or Celienas' raster to SVG, SVG Factory's WMF or BMP to SVG)
  - Subclasses to Java's Graphics2D for the generation of SVG from Java (Sun, CWI)

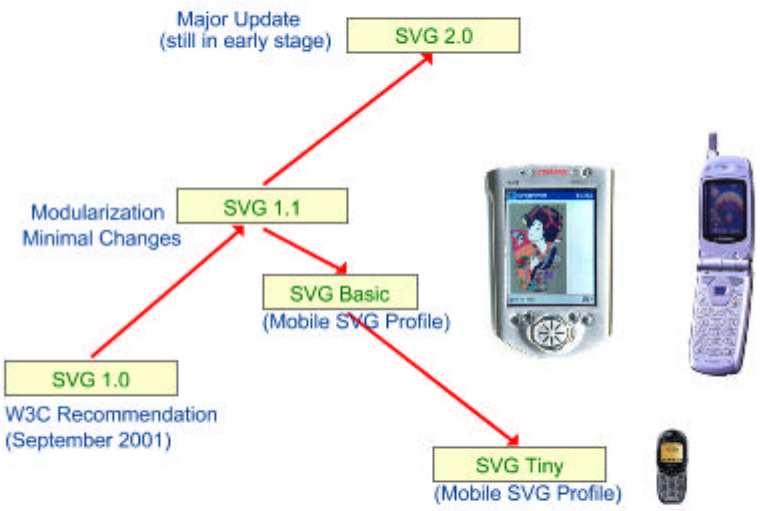
  Look at W3C's site: http://www.w3.org/Graphics/SVG/SVG-Implementations for further details!

Application Examples

- Raster to SVG Conversion: myself ;-) (using Celinea's conversion tool)
- Cartography: statistical data of Vienna or map of Europe (A. Neumann, University of Zurich)
- MathML: SchemaSoft Custard's conversion of MathML to SVG
- CML: Chemical Markup Language visualization (http://www.xml-cml.org/)
- CAD: CAD Standard Pro (http://www.cadstd.com)
- Business Graphics: Causeway (http://www.causeway.co.uk/demos/svg/)
- Web Site: Virtual Mechanics' web site (http://www.virtualmechanics.com), or svg Spider (http://www.svgspider.com)
- (Information) Visualization: GVF (http://gvf.sourceforge.net)
- Education: DVD Reading process (http://www.usbyte.com)

- Introduction
- Transformations, structuring
- Geometry
- Clipping, masking
- Attributes, fonts
- Styling
- Filters
- Animation
- DOM
- Miscellaneous
- Implementations, examples
- Future

Roadmap of SVG Evolution



SVG 1.1

- SVG 1.1: modularization
  - important for the definition of SVG Basic and Tiny ("Mobile Profiles")
- Minor extensions:
  - simple text wrapping

```
<div>
  <region>
    <rect .../></region>
    <rect .../></region>
  </region>
  <p>Bla bla bla...</p>
  <p>Bla bla bla with <br/>newline...</p>
</div>
```

  - Geographic Coordinate Systems: information added as metadata
- Latest public *draft* published middle of February 2002

SVG Basic

- JPEG, PNG, and SVG formats for images required
- Subset of CSS styling
  - subset of selectors
  - some font properties (or property values) missing
    - eg, no line-through or condensed
- No elliptical arc curve command in path
- Clipping against one rectangle element only
- A subset of the filter effects (remove, eg, turbulence, lighting support)
- A subset of the declarative animation functions
- Scripting is optional but, if supported, ECMAScript is required
- A subset of interactivity features (not yet specified)
- Subset of SVG fonts (restriction on glyph geometry); WebFonts
- "Lighter" conformance requirements (accuracy issues, for example)
- Latest public *draft* published middle of February 2002

SVG Tiny

- All the SVG Basic profile restrictions
- The value of the align parameter is restricted to XMidYMid
- No support for text on path, tspan, tref, or text wrapping
- Only solid colors for fill
- No opacity
- No symbols
- No filter effects
- For hyperlinking, eg, with use, all referenced object must be internal
  - this does not apply to the a element
- No linking to particular views of the target
- Only linear, paced, and discrete animation (ie, no splines)
- No WebFonts, only (restricted) SVG Fonts
- No requirement on ECMAScript even if scripting is supported
- Latest public *draft* published middle of February 2002

SVG 2.0

- Viewport coordinates (for toolbars, legends, etc)
- Extended set of path types (NURBS, general functions)
- Extended set of basic shapes (pieslice, spiral star, regular polygons)
- Perspective transformation (ie, full 3x3 matrices)
- User interface widget set
- Text justification (extending the SVG 1.1 specification on text flow)
- Print control
- Larger set of filter functions (eg, other compositing operations)
- Constraint features
  - "this line connects these two rectangles"
- Z-index (control over the basic painter model)
- Different timelines
- Level of detail support
- Encryption
- No public draft yet!

Advantages of XML

- Advantages of XML proper:
  - Clear text (easy editing, searching possibilities)
    - using compression reduces the size to acceptable level
  - lots of XML editors available; some can also be adapted to a particular DTD/Schema
  - validating parsers available in C++, Java, C, Python, ... (some for free)
- Synergy effect with other W3C recommendations, eg:
  - usage of Namespaces, XLink, XBase for linking, CSS styling
  - usage of XPointer for better hyperlinking *into* documents

    <use xlink:href="AnimalsInsline.svg#xpointer(//svg[contains(desc,"duckling")][position()=1])"/>
  - generation of SVG from other XML vocabularies on the fly using XSLT/XPath
  - clear inclusion of SVG images into, eg, XHTML
  - adding (RDF-based or not) metadata into SVG files, for example:
    - Dublin core metadata for digital libraries
    - signing (part of) an SVG file by adding a digital signature using XML Signature
    - the signature can be an embedded XML fragment in the metadata section
  - encoding (part of) an SVG file with XML Encryption
  - usage of XML Query to find a particular SVG file in a database
  - clear way of transferring SVG files in a Web services environment (using SOAP, WSDL, etc)

Further Information

These slides:
http://www.w3.org/Talks/2002/IH-Web3D/

SVG Home page:
http://www.w3.org/Graphics/SVG/

The SVG Recommendation itself:
http://www.w3.org/TR/SVG/

Some other sites on SVG:
http://www.kevlindev.com
http://wdvl.com/Authoring/Languages/XML/SVG/

More information about W3C:
http://www.w3.org/Consortium/

Contact information:
http://www.w3.org/Consortium/Contact

W3C home page:
http://www.w3.org

Mail me:
ivan@w3.org