



# **COMP 4021**

## **Internet Computing**

### **Dynamic SVG**

Dr. Kenneth LEUNG

Slides created by Dr. David ROSSITER

# Approaches to Dynamic SVG

- SVG can be dynamically changed *while* it is being displayed
- There are two different approaches:
  - 1) Use SVG commands to make changes:
    - There are SVG commands to make changes (transformations)
    - There are SVG commands to animate changes
    - Plugins may be need to access all SVG animation commands
  - 2) Use JavaScript to make change to DOM (SVG is just part of the DOM)
    - Should work in all browsers (No plugin is needed)
    - To be discussed in later presentation

# Transformations (without JavaScript)

- All SVG graphic elements have a "transform" attribute to make changes to the graphic elements
- The transformation commands available are
  - translate
  - rotate
  - scale
  - matrix - can be used to do all of the above operations, individually or all at the same time

# Translate

- **translate( <tx> [<ty>] )** will move the element <tx> units along the x-axis and <ty> units along the y-axis.

```
<image xlink:href="ust.jpg" transform="translate(50,50)"  
x="0" y="0" width="300" height="200"/>
```

trans1\_nothing.svg



trans2\_translate.svg



# Scale

- **scale( <sx> [<sy>] )** will scale the element by multiplying <sx> and <sy> to the x and y coordinates
  - If <sy> is not given, it is assumed to be the same as <sx>
  - <sx> or <sy> is 0 it means the corresponding dimension has no change in scale
  - Scaling is **relative to the origin (0,0)**

# Scale

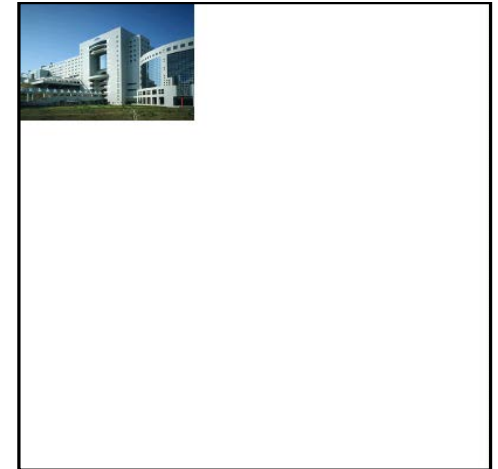
- Shrink the image to one half of its original size

```
<image xlink:href="ust.jpg" transform="scale(0.5 0.5)"  
x="0" y="0" width="300" height="200"/>
```

Demo – trans1\_nothing.svg



Demo – trans3\_scale.svg

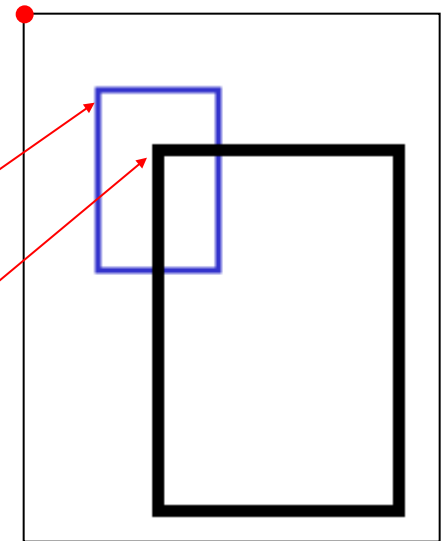


## Scale (Cont.)

- Scaling is relative to the origin (0,0)

```
<rect x="10" y="10" width="20" height="30"  
  transform="scale(2)" />
```

Original rect at 10,10  
Transformed rect at 20,20



To scale using a different center point, translate the element so that the center point becomes (0,0), perform scale, then translate the element back to its original location

# Rotate

- **rotate(<angle>, centre x, centre y)** rotates the element <angle> degrees around the point (centre x, centre y)

```
<image xlink:href="ust.jpg" transform="rotate(30,150,100)"  
x="0" y="0" width="300" height="200"/>
```

Rotate around  
the center of the  
photo

trans1\_nothing.svg



trans4\_combination.svg





## Rotate (Cont.)

- If rotation center is not given, assume the center is 0,0
- The following code has the same effect:

```
<image xlink:href="ust.jpg" transform="
  translate(150 100) rotate(30) translate(-150 -100) "
  x="0" y="0" width="300" height="200"/>
```

# Matrix

You need to understand the general idea of `matrix()` – but you won't be expected to build something using it, as it is too 'pure' computer graphics for COMP4021

- In computer graphics, matrices are commonly used
- They can be used to perform one single operation (i.e. one of translation/ rotation/ scaling)
- More powerfully, they can be used to combine a sequence of operations together into one equivalent set of numbers
- Then the set of numbers is applied to the object
- This is quicker than applying a lot of separate operations to the object
- SVG supports matrix operations with the *matrix()* command

# Matrix

- The matrix command uses a matrix of six numbers
- Simple example:

```
<image xlink:href="ust.jpg"
```

```
transform="matrix(1.5 0 0 1.5 0 0)"
```

*Multiply all y  
values by 1.5*

*Multiply all x values by 1.5*

```
x=" " y="0" width="300" height="200"/>
```

# Matrix

trans1\_nothing.svg



trans5\_matrix.svg



# Matrix

- What about a series of operations?
- For example, if you want to
  - move shape from (150, 100) to origin
  - rotate shape 30 degrees
  - move shape to new position (200, 100)
- First, work out the equivalent matrices
- Then multiply them all together to get an equivalent single matrix

# Matrix

$$\begin{bmatrix} 1 & 0 & x2 \\ 0 & 1 & y2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -x1 \\ 0 & 1 & -y1 \\ 0 & 0 & 1 \end{bmatrix}$$

Translate (x2,y2)

Translate (-x1,-y1)

Rotate (a) degrees

After multiplying all three matrices, the result is:

$$\begin{bmatrix} \cos(a) & -\sin(a) & -x1\cos(a) + y1\sin(a) + x2 \\ \sin(a) & \cos(a) & -x1\sin(a) - y1\cos(a) + y2 \\ 0 & 0 & 1 \end{bmatrix}$$

# Matrix

- The equivalent SVG matrix is:

`transform =`

```
"matrix(cos(a), sin(a), -sin(a), cos(a),  
-x1cos(a)+y1sin(a)+x2, -x1sin(a)-y1cos(a)+y2)"
```

- In this particular case:

```
transform="matrix(0.866 0.5 -0.5 0.866 175 -61.6)"
```

# Matrix

trans1\_nothing.svg



trans6\_matrix2.svg





# Animation (Without JavaScript)

- So far we have looked at SVG commands to change an SVG element (once)
- But how can we continually apply a change over time, to get some kind of **animation** effect?
- SVG has commands for this also, called:
  - animate
  - animateColor
  - animateMotion
  - animateTransform

# SVG Animation Commands

- `animate` - for animating any attribute
- `animateColor` - for animating color attributes only
  - The `animate` command can also do exactly the same thing that `animateColor` can do
- `animateMotion` - for animating any object in a motion path
- `animateTransform` - for animating any object by changing any transformation (i.e. animating translation/ scale/ rotation/ matrix parameters)

# animate

Anim01\_animate.svg

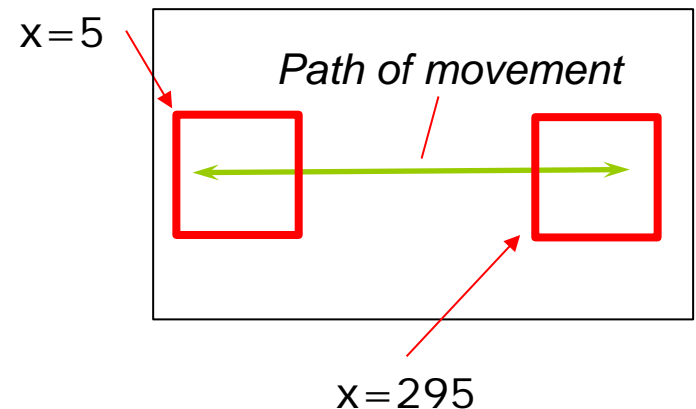
```
<rect x="5" y="150" width="100" height="100" style="fill:none; stroke:red;
stroke-width:5" >
```

```
  <animate attributeName="x" attributeType="XML"
    dur="5s" values="5; 295; 5"
    repeatCount="indefinite"/>
```

```
</rect>
```

The x position is changed over a period of five seconds, from x=5 to x=295, and then back to x=5

Values are interpolated between the three key values : 5, 295, 5



# Animate Two Parameters

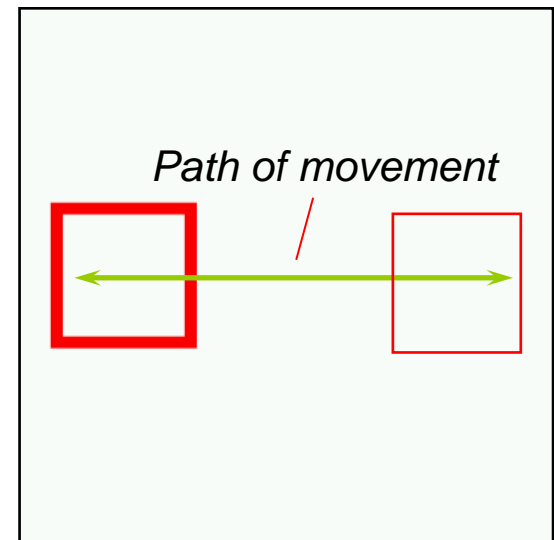
Anim02\_animate.svg

```
<rect x="5" y="150" width="100" height="100" style="fill:none; stroke:red; stroke-width:5" >
```

```
<animate attributeName="x" attributeType="XML"
  dur="5s" values="5; 295; 5"
  repeatCount="indefinite"/>
```

```
<animate attributeName="stroke-width"
  attributeType="CSS" dur="5s"
  values="10; 1; 10"
  repeatCount="indefinite"/>
```

```
</rect>
```



# colorAnimate

Anim03\_color.svg

```
<rect x="5" y="150" width="100" height="100"  
  style="fill:none;stroke:red;stroke-width:5" >  
  
  <animateColor attributeName="fill"  
    attributeType="CSS" from="rgb(255,255,255)"  
    to="rgb(255,0,0)" begin="0s" dur="5s"  
    fill="freeze" />  
  
</rect>
```

The fill colour is interpolated from white (255,255,255) to red (255,0,0) over five seconds

# colorAnimate

Anim04\_color.svg

```
<rect x="5" y="150" width="100" height="100"  
  style="fill:none;stroke:red;stroke-width:5" >
```

Colour is interpolated  
between these 7 key values

```
<animateColor attributeName="fill"  
  attributeType="CSS"  
  values="red;orange;yellow;green;blue;indigo;violet"  
  begin="0s" dur="8s" repeatCount="indefinite"/>  
</rect>
```

The fill colour shows all the colours of the rainbow, in a cycle lasting 8 seconds

# attributeType

- Each node can have a variety of attributes
- Some are from style sheet parameters; there are others such as those added by the programmer (these are called XML attributes)
- So the *attributeType* can be one of
  - "CSS" (if the attribute being controlled is a CSS property)
  - "XML" (if the attribute being controlled is an XML property)
  - or "auto" (this is the best value if you're not sure – the browser will search through all the attributes and use the right one)

# animateMotion

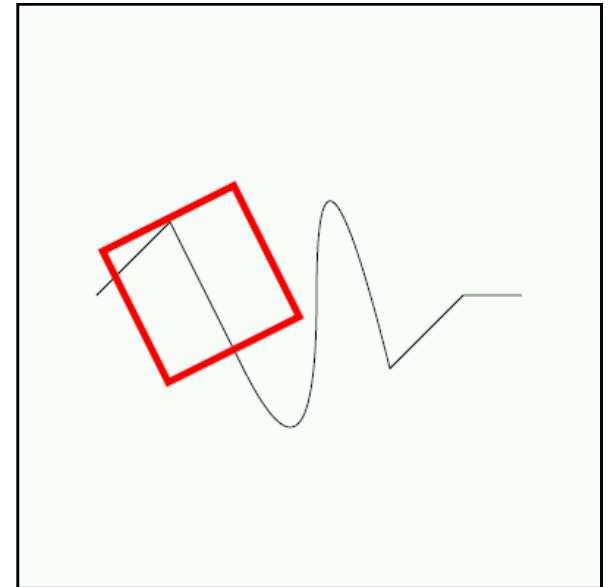
Anim05\_motion.svg

- SVG elements can be animated along a path specified by path data in the <animateMotion> element

```
<rect x="-50" y="-50" width="100" height="100"  
  style="fill:none;stroke:red;stroke-width:5" >
```

```
  <animateMotion  
    path="M55,200 l50,-50 t50,100  
          t50,-50 t50,50 l50,-50 L345,200"  
    dur="3s" fill="freeze" rotate="auto"/>
```

```
</rect>
```





# animateTransform

Anim06\_transform.svg

- animateTransform is for animating translation/ rotation/ scaling

```
<g transform="translate(200, 200)">  
<rect x="-50" y="-50" width="100" height="100"  
  style="fill:none;stroke:red;stroke-width:10">  
  <animateTransform type="scale" attributeName="transform"  
    attributeType="XML" dur="5s" values="1;2;1"  
    repeatCount="indefinite"/>  
</rect> </g>
```

The rectangle is made larger and smaller in a 5 sec period

# animateTransform

Anim07\_transform.svg

```
<g transform="translate(200, 200)">  
  <rect x="-50" y="-50" width="100" height="100"  
    style="fill:none;stroke:red;stroke-width:10">  
    <animateTransform type="rotate" attributeName="transform"  
      attributeType="XML" dur="5s" from="0" to="360"  
      repeatCount="indefinite"/>  
  </rect> </g>
```

The rectangle is constantly rotated