COMP 4021
Internet Computing

Dynamic SVG

Extended by Dik Lee from original slides
prepared by 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
     - Works in Chrome, Safari, etc, and also IE
     - Older versions of IE and Firefox may not support SVG animation
  2) Use JavaScript to make change to DOM (SVG is just part of the DOM)
     - Should work in all browsers
     - 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
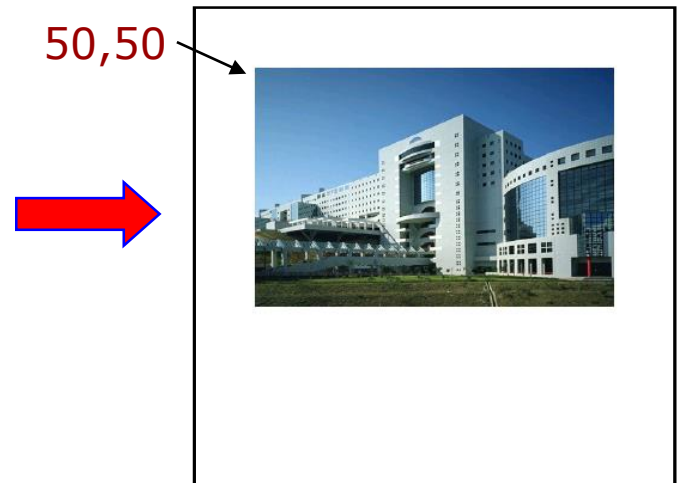
Translate

- **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

50,50

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 <xy> 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 href="ust.jpg" transform="**scale**(0.5 0.5)"
  x="0" y="0" width="300" height="200"/>
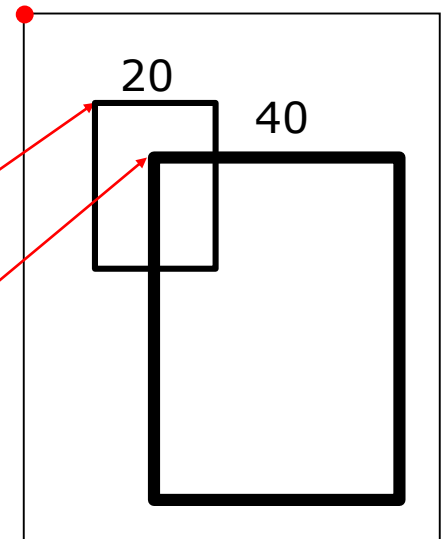
Demo – trans1_nothing.svg

Demo – trans3_scale.svg

Scale an Object not Located at (0,0)

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

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

20

40

Original rect at 10,10

Transformed rect at 20,20

Scale an Object at its own Center

- 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

- Scaling an object around the center:
  - Translate center to 0,0
  - Scale
  - Translate back to **original** center
    (hard-code center in translate command)

- `<rect x="10" y="10" width="20" height="30" fill="none" stroke="green" transform="translate(20,25)scale(3)translate(-20,-25)" />`   Demo
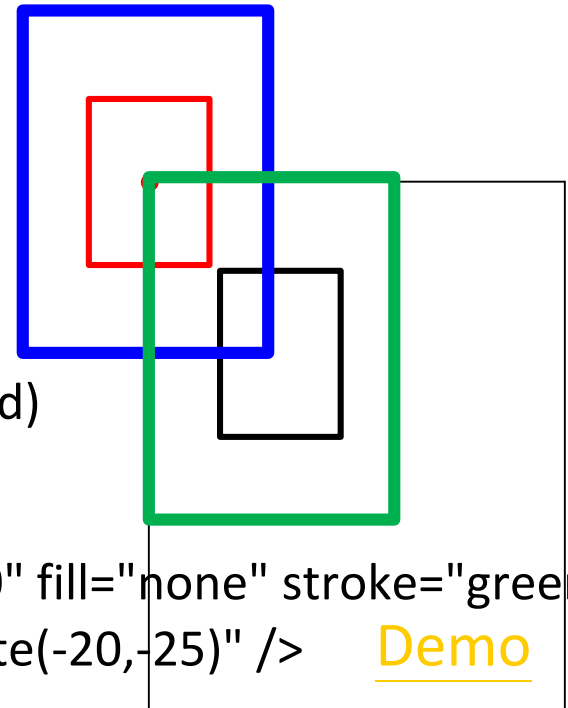
Scale with JavaScript

- With JavaScript:
  - Save original center in variables
  - Scale
  - Translate new center to original center

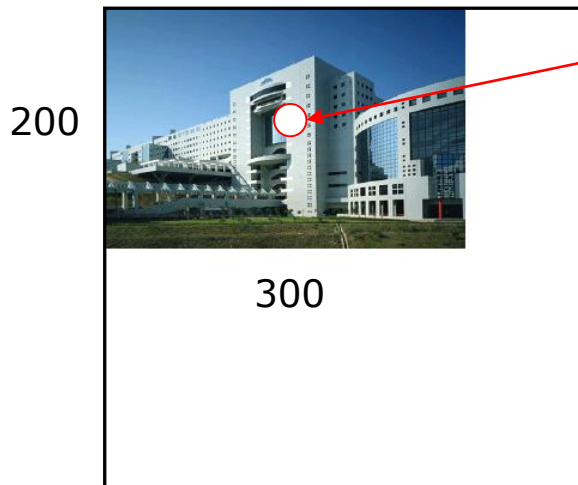# Rotate

- **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" />
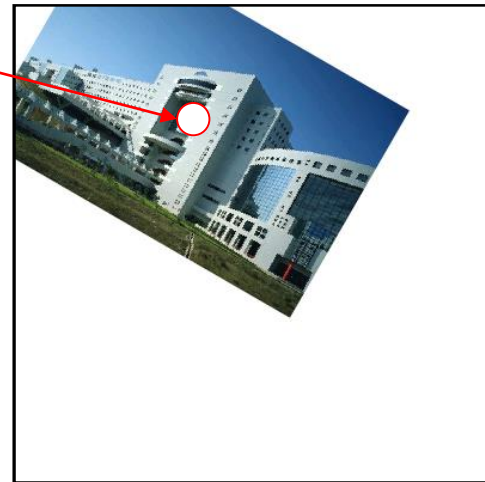
Rotate around the center of the photo

trans1_nothing.svg

150,100

200

300

trans4_combination.svg

Rotate (Cont.)

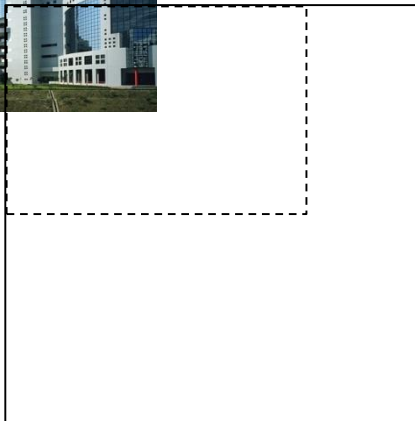□ If rotation center is not given, (0,0) is assumed
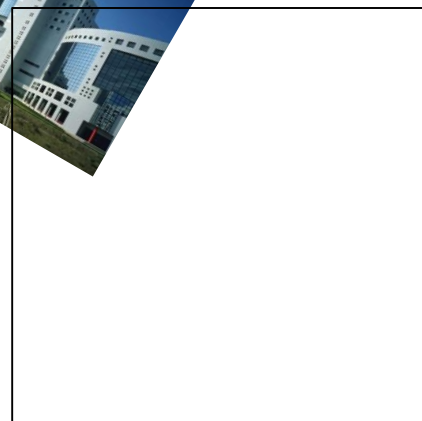
Rotate (Cont.)

- ☐ The following code has the same effect:

`<image href="ust.jpg" transform="`

`translate(150 100) rotate(30) translate(-150 -100) "`

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

Operations are performed from right to left

Multiple Operations in Transform (1)
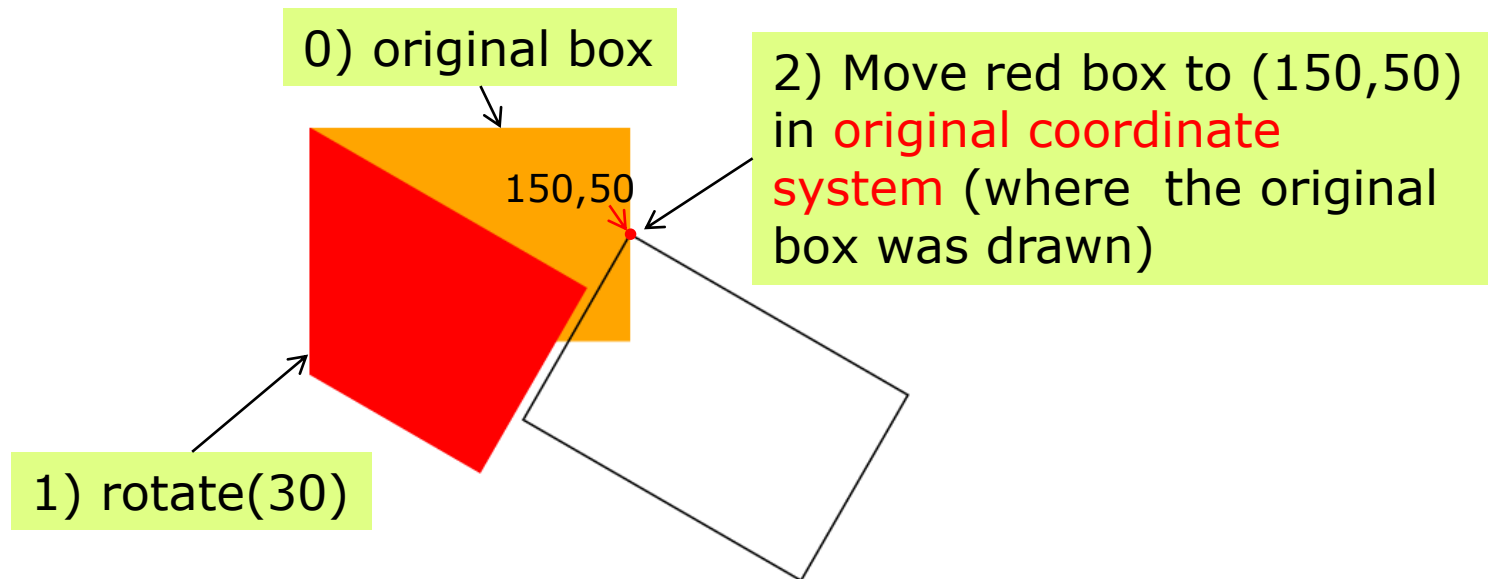
☐ Rotate a picture then translate it

<rect x="0" y="0" width="150" height="100" transform="**translate**(150,50) **rotate**(30)" />

☐ Transform operations are performed from right to left, i.e., first perform rotate(30) then translate(150,50)

0) original box

2) Move red box to (150,50) in original coordinate system (where the original box was drawn)

150,50

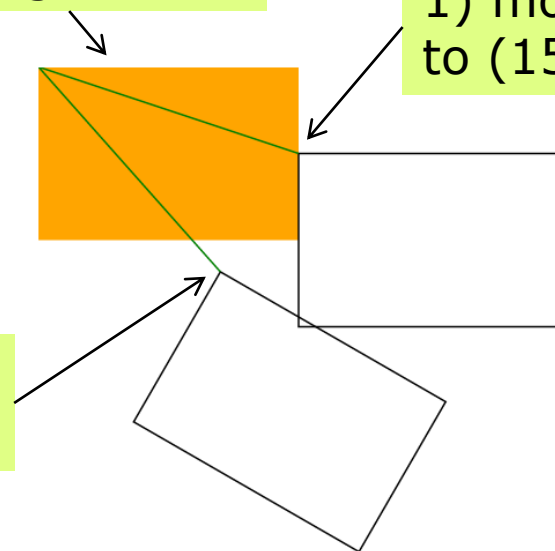1) rotate(30)

Multiple Operations in Transform (2)

- All transform operations perform in the original coordinate system; an operation does not change the coordinate system of subsequent operations

  - In previous example, coordinate system of translate(150,50) is not affected by rotate(30)

- <rect x="0" y="0" width="150" height="100" transform=" **rotate**(30) **translate**(150,50)" />

0) original box

1) move box to (150,50)

2) rotate(30) around (0,0), not around (150,50)

Codes for the Previous Examples

```
<svg width="300" height="400">
<rect x="0" y="0" width="150"
height="100" style="fill:orange"/>
<rect x="0" y="0" width="150"
height="100" style="fill:red"
transform="rotate(30)"/>
<rect x="0" y="0" width="150"
height="100"
style="fill:none;stroke:black"
transform="translate(150,50)
rotate(30)"/>
</svg>
```

Reordering the operations gives different results!

```
<svg width="300" height="400">
<rect x="0" y="0" width="150" height="100"
style="fill:orange"/>
<rect x="0" y="0" width="150"
height="100" style="fill:none;stroke:black"
transform="translate(150,50)"/>
<rect x="0" y="0" width="150" height="100"
style="fill:none;stroke:black"
transform="rotate(30) translate(150,50)"/>

<line x1=0 y1=0 x2=150 y2=50
style="stroke:green"/>
<line x1=0 y1=0 x2=150 y2=50
style="stroke:green"
transform="rotate(30)">
</svg>
```

Animation in 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: x, y, stroke color, fill color, etc.

- **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

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

        <animate attributeName="x"
            dur="5s" values="5; 295; 5"          From x=5 to
            repeatCount="indefinite"/>            x=295 inc 5

</rect>
```

Demo

x=5
y=150

Path of movement

x=295
y=150

- 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

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

   <animate attributeName="x"
      dur="5s" values="5; 295; 5"
      repeatCount="indefinite"/>
   <animate attributeName="stroke-width"
      dur="5s"
      values="10; 1; 10"
      repeatCount="indefinite"/>
</rect>
```
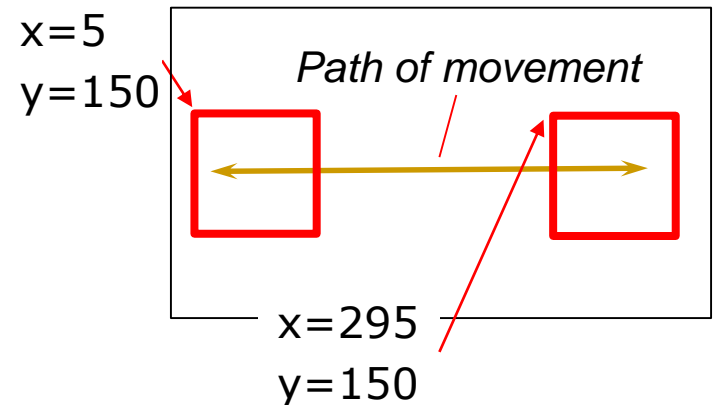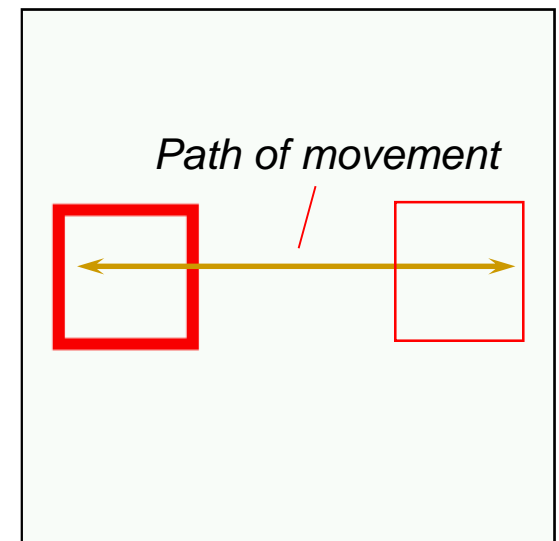
that if applied with care,

[Demo](#)

Path of movement

Combining Animation Effects

- When there are two <animate…/>, animating two parameters, say, x and y, how does the 2nd animate affect the first one?
  - additive="sum" 2nd animate will add to the first one (x and y both change)
  - additive="replace" 2nd animate will replace the first one
- When an animation has multiple iterations, how does the nth iteration affect the n-1th one?
  - accumulate="sum" the nth parameter value will add to the n-1th iteration
  - accumulate="none" the nth animate overwrites the n-1th one

Demo

Color Animation

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

  <animate attributeName="fill"
    from="rgb(255,255,255)"
    to="rgb(255,0,0)" begin="0s" dur="5s"        Demo
    fill="freeze" />
</rect>
```

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

- fill="freeze": display the last color at the end of animation

Color Animation

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

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

Colour is interpolated between these 7 key values

Demo

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

animateMotion

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

```
<path d="M55,200 l50,-50 t50,100 t50,-50 t50,50 l50,-50 L345,200"
    style="fill:none;stroke:black"/>
<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>
```

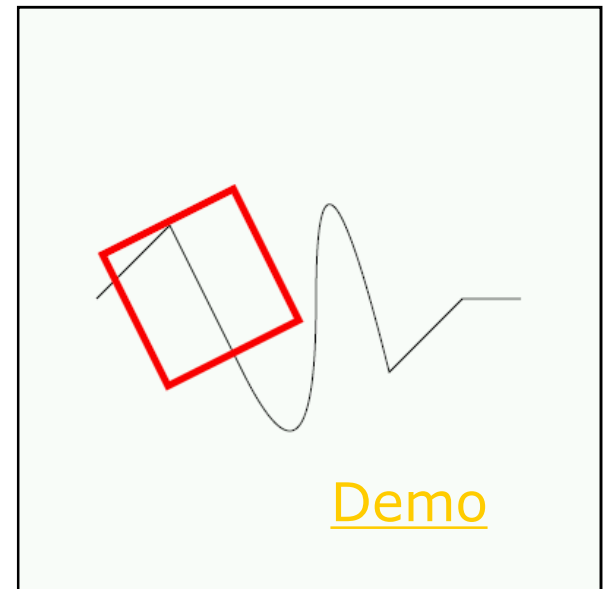Draw quadratic Bezier to the specific points with implicit control points

Recall:

Small letter: relative   Capital letter: absolute
t/T: smooth quadratic   q/Q: quadratic Bezier

Demo

animateTransform

❑ animateTransform is for animating the transform="…" command, i.e., translate(), rotate() and scale()

```
<g transform="translate(200, 200)">
```
What is this?
```
<rect x="-50" y="-50" width="100" height="100"
   style="fill:none;stroke:red;stroke-width:10">

   <animateTransform type="scale"
   attributeName="transform" dur="5s" values="1;2;1"
   repeatCount="indefinite"/>
</rect>  </g>
```

Demo

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

■ Note: How to make the box scale at its own center?

animateTransform

```
<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" dur="5s" from="0" to="360"
    repeatCount="indefinite"/>
</rect>   </g>
```

Demo

- The rectangle is constantly rotated
- Again, how to make the box rotate around its own center?

Take Home Message

- SVG does not just display simple graphics; it can transform and animate graphics
- All these are done within SVG without complex programming
  - The idea of markup languages is that non-programmers can do what they want without programmers' help
- Warning: There are many nicely written tutorial on SVG, but many are wrong in the execution order of multiple operations (right to left, not left to right!!!)
  - Understand the operations and their relation to the coordinate system
- JavaScript is not an essential requirement although it can further enhance interactivity