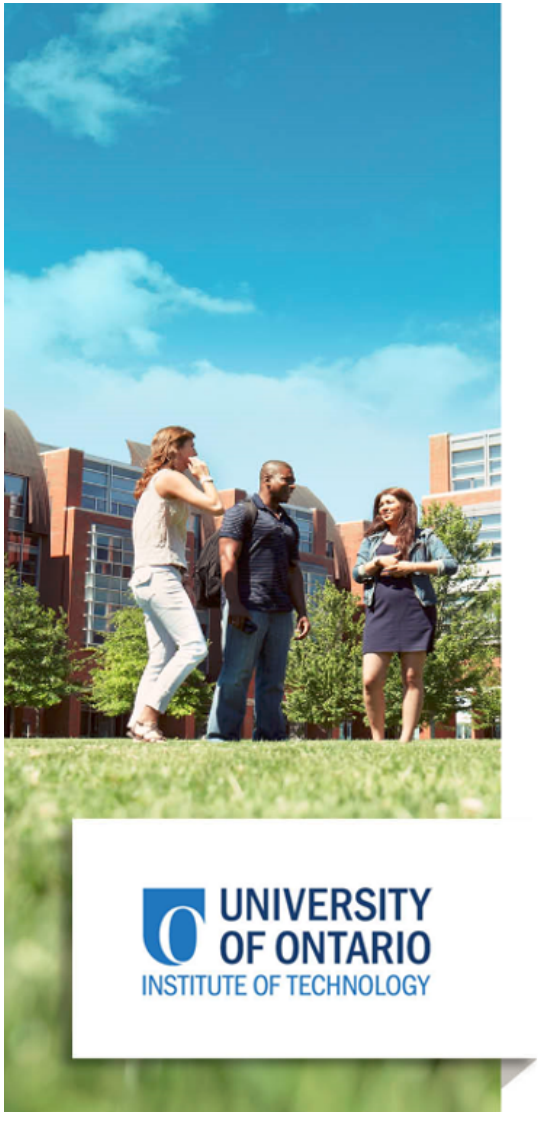
D3 AND SVG

DYNAMIC AND GRAPHICAL

Randy J. Fortier
randy.fortier@uoit.ca
@randy_fortier

OUTLINE

- Scalable Vector Graphics
- D3
  - Basics
  - Data loading and binding
  - Scales
  - Layouts
  - Animations



D3 AND SVG

SVG

SCALABLE VECTOR GRAPHICS (SVG)

- XML-based language for specifying vector graphics
  - Vector graphics scale with no loss in quality

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
          stroke="black" stroke-width="2" fill="red" />
</svg>
```

SVG SHAPES

- Primitives:
  - Circle
  - Ellipse
  - Rectangle
  - Line
  - Polyline
  - Polygon
  - Text
- Paths

SVG PRIMITIVES

- Circles:

```
<circle cx="0" cy="0" r="200"
        stroke="black" stroke-width="2" fill="yellow" />
```

SVG PRIMITIVES

- Circles
  - with CSS styling:

```
<circle cx="0" cy="0" r="200"
        style="fill:yellow; stroke:black; stroke-width:2" />
```

SVG PRIMITIVES

- Circles
  - SVG:

```
<circle cx="0" cy="0" r="200" id="face" />
```

  - CSS:

```
circle#face {
  fill: yellow;
  stroke: black;
  stroke-width: 2;
}
```

SVG PRIMITIVES

- Ellipses:

```
<ellipse cx="900" cy="125" rx="100" ry="75"
         style="fill:red; stroke:black; stroke-width:2" />
```

SVG PRIMITIVES

- Rectangle:

```
<rect x="550" y="50" width="200" height="150"
      style="fill:blue; stroke:black; stroke-width:5; opacity:0.5" />
```

SVG PRIMITIVES

- Line:

```
<line x1="550" y1="250" x2="750" y2="400"
      style="stroke:rgb(128,128,128); stroke-width:2" />
```

SVG PRIMITIVES

- Polyline:

```
<polyline points="550,450 650,575 750,490 850,560 950,515"
          style="fill:none; stroke:black; stroke-width:2" />
```

SVG PRIMITIVES

- Polygon:

```
<polygon points="800,325 950,270 1000,325 960,400 900,375"
         style="fill:blue; stroke:black; stroke-width:2" />
```

SVG PATHS

- A path allows for sequences of:
  - Lines
  - Elliptical arcs
  - Bezier curves
- Details can be found at Mozilla Developer Network

```
<path d="M90.92974268256818,41.6146836547142 5A100,100 0 0,1 -90.92974
```

SVG STROKE

- The stroke is the style of the lines
  - Colour
  - Thickness
  - Line cap/joins
  - Dashing
  - Opacity

```
<line x1="0"  y1="100" x2="10"  y2="20"
      stroke="black" stroke-width="20"
      stroke-linecap="butt" stroke-linejoin="bevel" />
<line x1="10" y1="90"  x2="110" y2="10" stroke-opacity="0.5"
      stroke="black" stroke-width="20" stroke-dasharray="5,5"
      stroke-linecap="round" stroke-linejoin="miter" />
<line x1="20" y1="80"  x2="120" y2="0"
      stroke="black" stroke-width="20" stroke-dasharray="5,3"
      stroke-linecap="square" stroke-linejoin="round" />
```

SVG FILL

- The fill is the style of the interior of the shape
  - Colour
  - Opacity

```
<rect x="0"  y="0" width="10"  height="50"
      fill="blue" fill-opacity="0.8" />
<rect x="10" y="10" width="10"  height="50"
      fill="red" fill-opacity="0.8" />
```

SVG GRADIENTS

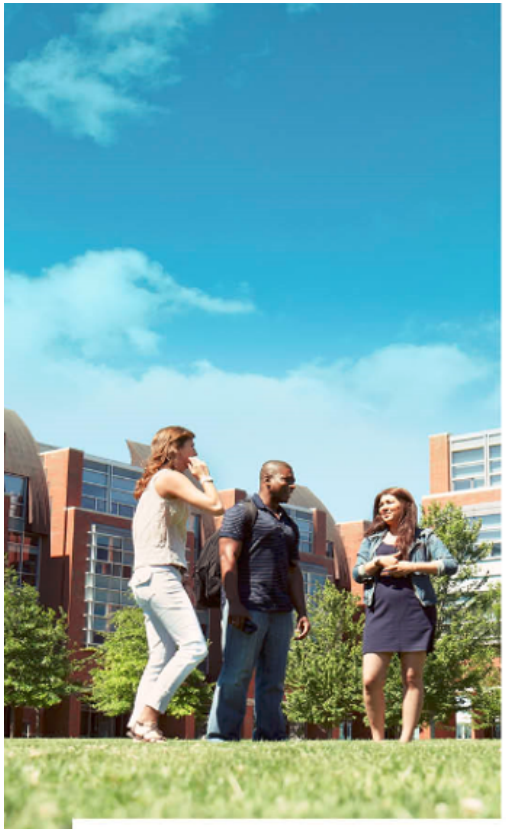
- A gradient allows colour blends:
  - Linear - left to right, top to bottom, across diagonal
  - Radial - from centre to outside

```
<linearGradient id="gradient" x1="0" x2="0" y1="0" y2="1">
  <stop offset="0%" stop-color="red" />
  <stop offset="50%" stop-color="orange" />
  <stop offset="100%" stop-color="yellow" />
</linearGradient>
<ellipse cx="200" cy="200" rx="75" ry="25"
         style="fill:url(#gradient); stroke:black; stroke-width:2" />
```

SVG TRANSFORMATIONS

- Transformations
  - Translation
  - Rotation
  - Scale
  - Skew
  - Arbitrary (with a matrix)

```
<g transform="translate(100,200),rotate(45)">
  <rect x="0" y="0" width="100" height="80" />
  <rect x="10" y="10" width="100" height="80" />
</g>
```



D3 AND SVG

D3.JS OVERVIEW

D3 OVERVIEW

- Data-Driven Documents (D3.js)
  - A JavaScript library quite similar to jQuery
  - D3.js, however, is structured around data
  - It is more declarative than jQuery

IMPORTING D3.JS

- Much like jQuery, you can:
  - Download and import locally
  - Import a specific version from a CDN
  - Import the latest version from a CDN:

```
<script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
```

SELECTION IN D3.JS

- Selection is very similar to queries in jQuery
- Select a single element:

```
d3.select("body").style("background-color", "black");
```

- Select multiple elements:

```
d3.selectAll("th").style("background-color", "lightGray"
```

METHOD CHAINING

- Like jQuery, D3.js supports method chaining
- Many methods return the object on which the method is operating

```
var canvas = d3.select("body")
               .append("svg")
               .attr("width", 500)
               .attr("height", 500);
```

DYNAMIC PROPERTY VALUES

- Properties can be static or dynamic values
  - Static - the actual value is hard coded
  - Dynamic - the value is determined by a function:

```
var colours = ['blue', 'red', 'green', 'yellow', 'orange'];
d3.selectAll("p")
  .style("color", function(data, index) { return colours[index]; });
```

ADDING NEW ELEMENTS

- You can add any elements that you like:

```
d3.select("body")
  .append("p")
  .attr("id", "dynamicParagraph")
  .text("This is a brand new paragraph")
  .style("color", "white");
```

DATA BINDING

- Data binding is the heart of D3
- D3 associates data elements with DOM elements

```
var dataArray = [1,2,3,4,5];
var circles = canvas.selectAll("circle")
                    .data(dataArray)
                    .attr("fill", "blue");
```

DATA BINDING

- When the selectAll() does not return enough elements, enter() is invoked

```
canvas.selectAll("circle")
      .data(dataValues)
      .enter()
        .append("circle")
        .attr("fill", "red")
        .attr("stroke", "black")
        .attr("stroke-width", "2")
        .attr("r", 50)
        .attr("cx", 600)
        .attr("cy", function(data, index) {
            return data;
         });
```

DATA BINDING

- When the selectAll() returns too many elements, exit() is invoked on the extras

```
canvas.selectAll("circle")
      .data(dataValues)
      .exit()
        .attr("fill", "gray");
```

DATA LOADING

- Loading from a CSV file is quite easy:

```
d3.csv('sales_data.csv', function(salesData) {
   canvas.selectAll("circle")
      .data(salesData)
      .enter()
         .append("circle")
         .attr("cx", function(data, index) {
            return index * 50 + 50;
         })
         .attr("cy", function(data, index) {
            return 300 - (data.sales / 10);
         })
         .attr("r", 10)
         .attr("fill", "red");
});
```

```
year,sales
2010,1076
2011,1214
2012,1107
2013,1520
2014,1712
2015,1606
2016,2188
```

DATA LOADING

- Loading from a JSON file works the same way:

```
d3.json('sales.json', function(salesData) {
   canvas.selectAll("circle")
      .data(salesData)
      .enter()
         .append("circle")
         .attr("cx", function(data, index) {
            return index * 50 + 50;
         })
         .attr("cy", function(data, index) {
            return 300 - (data.sales / 10);
         })
         .attr("r", 10)
         .attr("fill", "red");
});
```

LAYOUTS

- Layouts determine where components should go
  - Chord - generates arcs and chord shapes, based on relationships
  - Cluster - organizes components into a tree with all leafs in the same position
  - Force - uses physics (e.g. verlet, gravity) to position components
  - Histogram - bar chart with quantization
  - Pack - circles within circles, hierarchically
  - Partition - recursively subdivide pie/donut arcs
  - Pie - compute the angles for pie and donut charts
  - Stack - compute locations for stacked bar or area charts
  - Tree - organize components into a constant-sized tree
  - Treemap - organize components into a dynamically-sized tree

SCALES

- Scales map inputs to outputs
  - Linear - maps on a linear relationship (i.e. y = mx + b)

```
var colourScale = d3.scale.linear()
                          .domain([1000, 2500])
                          .range(["orange", "blue"]);
...
canvas.append("circle")
      .attr("cx", "100")
      .attr("cy", "100")
      .attr("r", "20")
      .attr("fill", function(data) { return colourScale(data.value);
```

ANIMATIONS

- Transitions and delays

```
canvas.selectAll("rect")
      .transition()
      .delay(2000)
      .duration(1000)
      .attr("y", function(data, index) {
         return 250;
      });
```

WRAP-UP

- In this section, we learned about:
  - SVG
  - D3
    - DOM manipulation
    - Data loading and binding
    - Layouts
    - Scales
    - Animation