Comp 125 - Visual Information Processing

Spring Semester 2018 - week 13 - monday

Dr Nick Hayward

Final Demo and Presentation

Dates

- Week 15 - Monday 23rd, Wednesday 25th, & Friday 27th April 2018
- Final report due 3rd May 2018 by 1pm

Final Demo and Presentation

- group project - maximum 2 persons per group
- develop an app concept and prototype
- working app (as close as possible…)
  - *must use technologies outlined during the course*
- show and explain code used to develop the app
- explain design decisions
  - *describe patterns used in design of app*
  - *layout choices…*
- show and explain implemented differences
  - *where and why did you update the app?*
  - *perceived benefits of the updates?*

Further details on course website - https://csteach125.github.io/coursework/#assessment2

Video - Design

Digital Prototyping

Rapid Prototyping 2 of 3: Digital Prototyping
Source: YouTube - Google

HTML Canvas - draw arcs and circles

- not restricted to simply drawing shapes with straight lines or rectangles
- we might also need to draw a circle, or a custom arc
- to draw a circle or arc, start by specifying
  - *the centre point for the circle*
  - *its radius*
  - *extent of the circumference*
- to draw an arc we provide a value
  - *for the starting angle and end angle*
  - *use to define the arc to draw*

HTML Canvas - draw arcs and circles

radians

- required start and end angles for drawing an arc are defined in **radians**
- to measure a circle using radians, we begin at 0
  - ***0*** *is equivalent to* ***3*** *on a clock*
- relative to a standard circle as a clock
  - *12pm = 270° or (π x 3 / 2 radians)*
  - *3pm = 0° (0 radians) & 360° (π x 2 radians)*
  - *6pm = 90° (π / 2 radians)*
  - *9pm = 180° (π radians)*

HTML Canvas - draw arcs and circles

`arc()` method

- expected parameters for the arc method is as follows

```
arc(x, y, radius, startAngle, endAngle, anticlockwise);
```

- where `anticlockwise` is set to `false` by default

HTML Canvas - draw arcs and circles

full circle - part 1

- using this pattern to draw a full circle
  - *start at 3pm and continue back round to 3pm*
- i.e. start at 0 radians and continue to `(π x 2 radians)`
- in JS, this may be represented as follows

```
// draw a full circle
context.beginPath();
context.arc(50, 100, 25, 0, Math.PI * 2, false);
context.stroke();
```

HTML Canvas - draw arcs and circles

full circle - part 2

```
// draw a full circle
context.beginPath();
context.arc(50, 100, 25, 0, Math.PI * 2, false);
context.stroke();
```

- call `arc()` method on the `context` object passing required arguments
  - *`50, 100` = the centre of the circle as x and y coordinates*
  - *`25` = radius of circle*
  - *`0` = 0 radians for the start position of the circle (0°)*
  - *`Math.PI * 2` = (π x 2 radians) for the end position for the end of the circle (360°)*

HTML Canvas - draw arcs and circles

arcs - part 1

- we can then create various arcs, including a semi-circle

```
// draw a semi-circle
context.beginPath();
context.arc(125, 100, 25, 0, Math.PI, false);
context.stroke();
```

- call `arc()` method on the `context` object passing required arguments
  - *`125, 100` = x & y centre of the circle*
  - *`25` = radius of circle*
  - *`0` = start position of arc (0°)*
  - *`Math.PI` = end position of arc (180°)*

HTML Canvas - draw arcs and circles

arcs - part 2

- we might also draw a quarter circle

```
// draw a quarter circle
context.beginPath();
context.arc(175, 100, 25, 0, Math.PI / 2, false);
context.stroke();
```

- **n.b.** `false` value in `arc()` method refers to anticlockwise parameter
  - *by default, an arc will follow a clockwise path*
- Example - arcs and circles
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic8/*

HTML Canvas - draw arcs and circles

Bézier curves

- we can also draw more fluid, or organic, shapes using bézier curves
- use a couple of default methods
- support for **cubic** or **quadratic** varieties of bézier curves

Bézier curves - Wikipedia

HTML Canvas - draw arcs and circles

quadratic - part 1

- we can draw a quadratic bézier curve from a defined start point
  - *i.e. current pen position on the canvas, using the following method*

```
quadraticCurveTo(cp1x, cp1y, x, y)
```

- `cp1x` & `cp1y` = controls points for curve
- `x` & `y` = standard `x` and `y` coordinates on the canvas
  - *defines end point from the current pen position*
- this type of curve has a defined start and end point with a single control point

HTML Canvas - draw arcs and circles

quadratic - part 2

- for example

```
// draw a quadratic bézier curve
context.beginPath();
context.moveTo(75, 25);
context.quadraticCurveTo(25, 25, 25, 62.5);
context.quadraticCurveTo(25, 100, 50, 100);
context.quadraticCurveTo(50, 120, 30, 125);
context.quadraticCurveTo(60, 120, 65, 100);
context.quadraticCurveTo(125, 100, 125, 62.5);
context.quadraticCurveTo(125, 25, 75, 25);
context.fill();
```

- Example - Bézier curves - quadratic
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-quadratic/*
  - *W3Schools - quadraticCurveTo()*

HTML Canvas - draw arcs and circles

cubic - part 1

- a cubic bézier curve, by contrast, has the following method and usage

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```

- pattern is similar to a quadratic curve
  - *primary difference is use of two control points*
  - *potentially offers finer control over extent and nature of curve*

HTML Canvas - draw arcs and circles

cubic - part 2

- for example

```
// draw a cubic bézier curve
context.beginPath();
context.moveTo(75, 40);
context.bezierCurveTo(75, 37, 70, 25, 50, 25);
context.bezierCurveTo(20, 25, 20, 62.5, 20, 62.5);
context.bezierCurveTo(20, 80, 40, 102, 75, 120);
context.fill();
```

- Example - Bézier curves - cubic
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-cubic/*
  - *W3Schools - bezierCurveTo()*

References

- W3Schools - HTML5
- canvas element