Comp 125 - Visual Information Processing

Spring Semester 2018 - week 14 - monday

Dr Nick Hayward

HTML Canvas - basic animations

random movement - part 1

- create various shapes and then animate paths
  - *randomly move shape around the canvas*
- start by defining the `canvas` and the `context`

```
// define canvas
var canvas = document.getElementById('drawing');
// define context for drawing
var context = canvas.getContext('2d');
```

HTML Canvas - basic animations

random movement - part 2

- decide upon a shape to draw
  - *e.g. a circle...*
- we may slightly modify the `circle` function
  - *add option for variant colours*

```
// define circle function
function circle(x, y, radius, fillCircle, color) {
    // start recording
  context.beginPath();
    // define arc - x, y, radius, start posn, end posn, anticlockwise...
    context.arc(x, y, radius, 0, Math.PI * 2, false);
    // check fill or stroke
    if (fillCircle) {
        // colour for fill
      context.fillStyle = color;
        context.fill();
    } else {
        // set line width & line colour
        context.lineWidth = 2;
    context.strokeStyle = color;
        context.stroke();
    }
}
```

- abstract `color` usage for drawing a circle
  - *pass a parameter for the required colour*
  - *colour may be used for either a fill colour or stroke style*
- colour usage will be relative to `boolean` passed for `fillCircle`

random movement - part 3

- then call this updated `circle` function
  - *create our well-known mouse with variant colours*

```
// 1. a well-known mouse with variant colours
// left ear
circle(117, 100, 18, true, 'black');
// right ear
circle(183, 100, 18, true, 'black');
// head
circle(150, 130, 33, true, 'DarkRed');
```

- Example - variant mouse colours
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.11/*

abstracted function - draw a well-known mouse

- abstract the drawing of the well-known mouse
  - *e.g. vary size according to a relative scale for each circle*

```
// define function to draw mouse - x & y = centre of head, radius = head size, color1 = head colour
function mouse(x, y, radius, fill, color1, color2) {
  //draw left ear
  circle(Math.floor(x/1.28), Math.floor(y/1.3), Math.floor(radius/1.8), fill, color2);
    //draw right ear
    circle(Math.floor(x*1.22), Math.floor(y/1.3), Math.floor(radius/1.8), fill, color2);
    //draw head
  circle(x, y, radius, fill, color1);
}
```

abstracted function - draw a well-known mouse - part

- we might define the base mouse size as follows

```
// base small size for mouse
mouse(150, 130, 34, true, 'DarkRed', `black`);
```

- then scale by a factor of 2 for a large mouse size

```
// scale by 2 - x, y & radius
mouse(300, 260, 68, true, 'DarkBlue', `green`);
```

- we may also now specify colours for the mouse as well
  - `color1` *for the head*
  - `color2` *for the ears*
- Example - variant mouse colours
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.2/*

HTML Canvas - basic animations

random movement - part 4

- to animate a shape in a random direction and path
  - *create a custom function to `update` this position*
- function will randomly change the `x` and `y` coordinates
  - *create effect of shape moving around the canvas*

```
// update the x and y coordinates for shape animation
function update(coord) {
    var offset = Math.random()*5-2;
    coord += offset;

    if (coord > 400) {
        coord = 0;
    }
    if (coord < 0) {
        coord = 0;
    }

    return coord;
}
```

- check if coordinates go beyond the width or height of the canvas
  - *if yes, reset back to the top using a value of 0*

HTML Canvas - basic animations

random movement - part 5

- then use this `update` function to radomly animate the shape
  - *use standard `setInterval` timer*

```
// animate our well known mouse
setInterval(function() {
    context.clearRect(0, 0, 400, 400);

  // 1. base small size for mouse
  circle(x, y, 40, true, 'green');
    x = update(x);
    y = update(y);

}, 20);
```

- start by clearing `context` for defined size of `canvas`
- then draw required shape to animate per frame
- `x` and `y` coordinates will be random by calling the `update` function
  - *call function with previous frame's `x` and `y` coordinates*
- Example - random movement and animation
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.3/*

JavaScript - Object Prototype

intro

- a *prototype* object may be used to delegate the search for a particular property
- i.e. a *prototype* is a useful and convenient option
  - *used for defining properties and functionality accessible to other objects*
- useful option for replicating many concepts in object oriented programming

JavaScript - Object Prototype

understanding prototypes - part I

- in JS, we may create objects
  - *e.g. using object-literal notation*

```
let testObject = {
    property1: 1,
    property2: function() {},
    property3: {}
}
```

- in this object we have
  - *a simple value for the first property*
  - *a function assigned to the second property*
  - *and another object assigned to the third object*

JavaScript - Object Prototype

understanding prototypes - part 2

- as a dynamic language, JS will also allow us to
  - *modify these properties*
  - *delete any not required*
  - *or simply add a new one as necessary*
- this dynamic nature may change properties in a given object
- in traditional object-oriented programming languages
  - *this issue is often solved using inheritance*
- in JS
  - *we can use prototypes to implement inheritance*

References

- MDN - Prototype
- W3Schools - HTML5
  - *media elements*
  - *canvas element*
- W3Schools - Prototypes