Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 12 - Wednesday

Dr Nick Hayward

HTML Canvas - draw with a function

- we may abstract drawing required shapes to a custom function
- a custom function may then be called to create a shape
  - *e.g. any size circle*

```
// define custom function to draw circle
function circle() {
    ...
}
```

HTML Canvas - draw with a function

custom drawn circles - part 1

- create a function to draw a custom circle
  - *position, radius, and fill*
- function draws a standard circle of varying radius and fill
- e.g. we might start with the following initial function

```
// define custom function to draw circle
function circle(x, y, radius, fillCircle) {

}
```

- then call this function as follows

```
// outer circle for head
circle(100, 100, 50, false);
```

HTML Canvas - draw with a function

custom drawn circles - part 2

- fill out the logic for our working `circle` function as follows,

```
// define custom function to draw circle
function circle(x, y, radius, fillCircle) {
    // 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) {
        context.fill();
    } else {
        context.stroke();
    }
}
```

HTML Canvas - draw with a function

a certain well-known *mouse* - part I

- we might use this new custom `circle` function
  - *create a certain well-known mouse*
- start by defining the `canvas` element in out HTML

```
<!-- add canvas -->
<canvas id="drawing" width="800" height="800"></canvas>
```

- then define the canvas and context in our JavaScript logic
  - *required to start drawing our shapes*

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

HTML Canvas - draw with a function

a certain well-known *mouse* - part 2

- add the `circle` function to our JavaScript
  - *we may start drawing the required shapes for our drawing*

```
// define custom function to draw circle
function circle(x, y, radius, fillCircle) {
    // 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) {
        context.fill();
    } else {
        context.stroke();
    }
}
```

HTML Canvas - draw with a function

a certain well-known *mouse* - part 3

- for this particular drawing
  - *add necessary specifics for colour of each circle's fill style*

e.g.

```
context.fillStyle = 'DarkRed';
```

HTML Canvas - draw with a function

a certain well-known *mouse* - part 4

- to draw the required shape for our well-known *mouse*
  - *we can use three circles*
- each circle will define
  - *position - x and y coordinates*
  - *a radius*
  - *and fill colour or not*
- then draw our well-known *mouse*
  - *call the `circle` function three times*

```
// 2. a certain well-known mouse
// left ear
circle(400, 100, 35, true);
// right ear
circle(500, 100, 35, true);
// head
circle(450, 160, 57, true);
```

- Example - circle function
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic11-function-circles/*

HTML Canvas - basic animations

draw and move

- we've seen how to draw static shapes and composite images
  - *e.g. from a stepped pyramid to a certain well-known mouse*
- it's also possible to animate these shapes
- animations within the confines of the defined `canvas` element
- animate on single or multiple axes
- add interaction and control
- move shapes around the canvas...

HTML Canvas - basic animations

horizontal animation - part 1

- start with a basic drawing
  - *then animate this shape across the screen*
- e.g. draw a simple rectangle to a standard HTML5 `canvas` element
- we may use this shape in the animation
  - *move it gradually across the HTML page*
- define a start position for the `X` coordinate
  - *then draw the initial shape*

```
// initial start position X for shape
var pos = 0;

// define rect for shape
context.fillRect(pos, 0, 40, 40);
```

HTML Canvas - basic animations

horizontal animation - part 2

- initially, the drawn rectangle is still simply static on the page
- to add a sense of animation
  - *need to continually draw this shape at a given time interval*
- need to ensure each previously drawn shape is removed from the `canvas`
- if not, drawing is a growing horizontal rectangle
  - *expands along the x-axis*

HTML Canvas - basic animations

horizontal animation - part 3

- we might now update our JavaScript code with a timer, `setInterval`

```
// initial start position X for shape
var pos = 0;

setInterval(function() {

  ...

}, 15);
```

- in the call to `setInterval`
  - *define a timer of 15 milliseconds*
- each call of `setInterval()` will execute an anonymous function
  - *controls drawing of the shape*
  - *controls the animation rendering*

HTML Canvas - basic animations

horizontal animation - part 4

- to draw a moving shape
  - *we need to clear the `canvas` or part depending upon the animation requirements*

```
// clear rect - matches size of canvas
context.clearRect(0, 0, 400, 400);
```

- `clearRect()` method on the `context` object
  - *called before each shape is drawn*
  - *dimensions set to size of defined `canvas` element in the HTML*
- we have a clear `canvas` for each frame of the animation

HTML Canvas - basic animations

horizontal animation - part 5

- we may draw our shape as expected

```
// define rect for shape
context.fillRect(pos, 0, 40, 40);
```

- with this usage we're dynamically updating the value of the shape's position
  - *makes the shape appear to move across the `canvas`*

HTML Canvas - basic animations

horizontal animation - part 6

- update the shape's position
  - *add a simple increment operator to our earlier `pos` variable*

```
// increment position value
pos++;
```

- need to check position of shape relative to defined dimensions of canvas

```
// check position to stop shape leaving canvas
if (pos > 400) {
  pos = 0;
}
```

- Example - horizontal animation
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation1/*

HTML Canvas - basic animations

animate size - part 1

- we may also animate the size of a shape using a similar pattern
- start by defining an initial size for our shape

```
// initial size for shape
var size = 0;
```

- set initial size to zero to allow the shape to grow
- for each frame of the animation
  - *modify dimensions of* `width` *and* `height`

HTML Canvas - basic animations

animate size - part 2

- we may use `setInterval()` to control canvas
  - *controls drawing of shape to create effect of animation*

```
setInterval(function() {

  ...

}, 15);
```

HTML Canvas - basic animations

animate size - part 3

- need to clear `canvas` for each frame of the animation
  - *then draw the required shape*

```
// clear rect - matches size of canvas
context.clearRect(0, 0, 400, 400);
// define rect for shape
context.fillRect(0, 0, size, size);
```

HTML Canvas - basic animations

animate size - part 4

- for this specific animation example
  - *we may save on redraws to the `context` by calling*

```
// clear rect - matches size of canvas
context.clearRect(0, 0, 400, 400);
```

- only when the shape has reached the edge of the canvas

HTML Canvas - basic animations

animate size - part 5

- we may increment the size of the shape

```
// increment position value
size++;
```

- also check overall size
  - *creates a loop to the animation*
  - *i.e. once shape has reached edge of canvas*

```
// check position to stop shape leaving canvas
if (size > 400) {
  size = 0;
}
```

- Example - animate size
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation2/*

HTML Canvas - animations

fun demos

Some fun examples of animations with HTML5 Canvas API.

- Destroy things in a video - http://www.craftymind.com/factory/html5video/CanvasVideo.html
- Particles - https://codepen.io/eltonkamami/pen/ECrKd
- Curtain - https://codepen.io/dissimulate/pen/KrAwx
- Jelly - https://codepen.io/dissimulate/pen/dJgMaO
- Canvas cycle - http://www.effectgames.com/demos/canvascycle/

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`

HTML Canvas - basic animations

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/animation3.1/*