CSCU9B2 Practical 7: HTML Canvas

Aims:

- **To use the HTML canvas tag.**
- **To use JavaScript to draw in the canvas.**

Please register your practical attendance: Go to the GROUPS\CSCU9B2 folder in your Computer folder and double-click on the Register icon. Ask a demonstrator if you need help or something goes wrong.

This sheet contains one checkpoint (see end of sheet).

This week's practical is about using the HTML5 <canvas> tag, and using JavaScript to draw inside the canvas. There is a good introduction at:

http://www.w3schools.com/html/html5_canvas.asp

And a whole load of examples at:

http://www.html5canvastutorials.com

What's on this practical sheet is just about enough, possibly, to do the lab work, but some things are probably better explained on the above tutorial pages!

Drawing lines in a canvas

The following code shows that basic recipe for drawing lines on a canvas:

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
// the next 2 lines provide access to the "context" of the canvas
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
// the following code draws a line from (x, y) to (x1, y1)
  context.beginPath(); // about to start a new path
  context.moveTo(x,y); // start the path at (x, y)
  context.lineTo(x1,y1); // draw a line from last location to new location
  context.stroke(); // make line visible
</script>
```

Lines can have as many `context.lineTo()` calls as required, allowing lines with many parts to be drawn. You can also join lines in different ways, by setting the property `context.lineJoin` to 'miter', 'round' or 'bevel'. Unless the lines are thick (and this is set using `context.lineWidth`), you may not be able to see the difference. The default is 'miter'.

Try it:

- Create a simple webpage containing a canvas.
- Call the canvas "myCanvas" and make it 600 by 300 pixels.
- Draw two lines, one from the top left to the bottom right, and the other from the top right to the bottom left.

Drawing a rectangle

This is achieved using `context.rect(x, y, width, height)`. The filling of the rectangle is set using context.fillStyle, and actual filling is done using context.fill. The line thickness is set using context.lineWidth, and the colour by context.strokeStyle. Finally, the rectangle is drawn using context.stroke. For example:

```
context.beginPath();
context.rect(180, 50, 200, 100);
context.fillStyle = "green";
context.fill();
context.lineWidth = 5;
context.strokeStyle = "red";
context.stroke();
```

(Note that there are also utility methods context.fillRect() and context.strokeRect() to do this, as per the examples in the lecture slides.)

Try it:

- Draw a box in the same canvas as before (myCanvas), centred in the canvas, of size 400 by 200 pixels, with a blue edge, and a yellow interior.
- Use context.lineWidth to make the lines drawn in the first part above, 10 pixels wide, and use context.strokeStyle to make the lines red in colour.

Putting some text in a canvas

Given a context for a canvas, the basic form of the program is

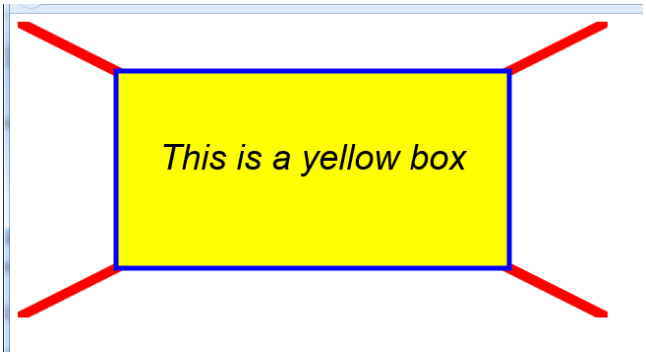
```
context.font = 'italic 40px Calibri'; // set the font style
context.fillText(str, x, y); // draw string starting at (x, y)
```

The string displayed by context.fillText consists of a font style (normal, italic or bold – the default is normal), a font size (a number of pixels = px), and a font family, in this case Calibri. You can set the text colour using context.fillStyle.

Try it:

- Put some text ("This is a yellow box") inside the yellow box you created above, in italic, 36 pixel size and Arial font.
    - If your text does not appear, it could be because you are still using yellow as your fill colour, so the text is the same colour as the box! So change it to another colour…
- Try to place it centred in the middle of the box, using the context.textAlign property.

If you have completed all the above steps, then your web page should look something like:



Drawing an image in a canvas

There is a very good tutorial on this at:

http://www.html5canvastutorials.com/tutorials/html5-canvas-images/

The one complexity is that the picture you want to display must be placed into an **Image** object before it can be used. In order to ensure that the image has been loaded before attempting to display it, one can draw the image using the **.onload** property of the Image object:

```
// create a new Image object, and access it as the var myImage
var myImage = new Image() ;
// call this function when myImage has finished loading
myImage.onload = function() {
        context.drawImage(myImage, 10, 5, 300, 150); // actually do
the drawing
      };
myImage.src = "myNiceImage.png" ;
```

Note that `context.drawImage(myImage, 10, 5, 300, 150);` will draw the image starting at the (x, y) coordinates (10, 5) with a width of 300 pixels and height of 150 pixels.

You will try this out for the **checkpoint** (see below). There are many variations on all of the above: the drawing facilities inside a canvas are quite sophisticated.

The exercise for today's checkpoint is given in the box on the next page.

CHECKPOINT [CANVAS]

Create a web page that displays the image of Stirling Castle (**Stirlingcastle.jpg** - which you will find in the **MediaExamples** folder from the previous practical) inside a canvas (drawing the image at a size of 800 by 600 pixels is sufficient). Annotate it with text and lines pointing to the trees, to the cliff, and to the castle wall, as follows: write some text describing each feature, and draw a line from the text to the area in the picture it refers to. The line should be 4 pixels wide in red, and the text also in red, size 36 pixels and Times font. This should all be done within the canvas, using the techniques described above.

- Note that all your drawing code should go inside the .onload() function, after the drawImage() statement (otherwise the image will be drawn last and overwrite your text and lines!)

For those of you for whom this is not sufficiently challenging, put arrowheads on the ends of the lines!

Your checkpoint page might look something like (exact arrangement is up to you):