Graphics Programming
Lecture 3

Recap

- Covered Raster vs. Vector images
- Introduction to image compression
- Computer Graphics Application
- HTML5 Canvas

HTML5 Canvas

- Checking for support
  - Not all browsers support HTML5 standard

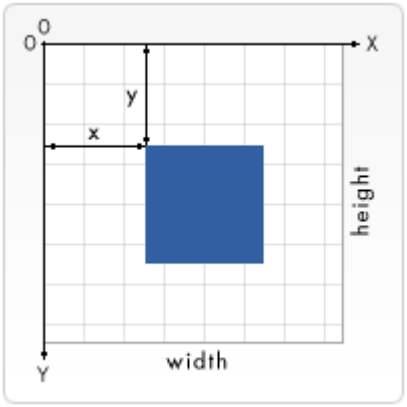
Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

- Can check for support programmatically by testing the operation of the getContext() method.

```
var canvas = document.getElementById('tutorial');

if (canvas.getContext){
  var ctx = canvas.getContext('2d');
  // drawing code here
} else {
  // canvas-unsupported code here
}
```

The HTML5 Grid

- Coordinate space
  - Normally 1 unit in the grid corresponds to 1 pixel on the canvas.
  - The origin of the grid is in the top left corner at coordinate (0,0).
  - All elements are placed relative to this location.
  - The position of the **top left corner** of the blue square is x pixels from the left and y pixels from the top (coordinate (x,y)).



HTML5 Rectangles and Paths

- <canvas> has only 2 primitive shapes: rectangles and paths
- All other shapes are created by combining one or more paths/polygons.
- However, there is a wide variety of path drawing functions thereby making it possible to create complex shapes.
- 3 functions to draw rectangles
  - fillRect(x, y, width, height) - ?
  - strokeRect(x, y, width, height) - ?
  - clearRect(x, y, width, height) - ?

HTML5 Rectangles and Paths

- <canvas> has only 2 primitive shapes: rectangles and paths
- All other shapes are created by combining one or more paths/polygons.
- However, there is a wide variety of path drawing functions thereby making it possible to create complex shapes.
- 3 functions to draw rectangles
  - fillRect(x, y, width, height) - Draws a filled rectangle.
  - strokeRect(x, y, width, height) - Draws a rectangular outline.
  - clearRect(x, y, width, height) - Clears the specified rectangular area - transparent.

HTML5 Rectangles and Paths

- A path is a list of points, connected by segments of lines (curved or straight)
- To make shapes using paths:
  - Create the path.
  - Draw the path.
  - Close the path (optional)
  - Once the path has been created, you can stroke or fill the path to render it.

HTML5 Rectangles and Paths

- A path is a list of points, connected by segments of lines (curved or straight)
- To make shapes using paths:
  - Create the path.
    - beginPath()
  - Draw the path.
    - Next slide
  - Close the path (optional)
    - closePath() - Closes the shape by drawing a straight line from the current point to the start.
  - Once the path has been created, you can stroke or fill the path to render it.
    - stroke() – draws outline
    - fill() – fills content area

HTML5 Rectangles and Paths

- A path is a list of points, connected by segments of lines (curved or straight)
- To make shapes using paths:
  - Create the path.
  - Draw the path.
    - ctx.moveTo(x,y);
    - ctx.lineTo(x,y);
    - ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise)
  - Close the path (optional)
  - Once the path has been created, you can stroke or fill the path to render it.

HTML5 Paths

- Moving the pen - ctx.moveTo(x,y);
  - Typically called after beginPath();
  - Doesn't draw anything – similar to lifting a pen from one spot to the next.
  - Can use to draw unconnected paths

```
ctx.beginPath();
ctx.arc(75,75,50,0,Math.PI*2,true); // Outer circle
ctx.moveTo(110,75);
ctx.arc(75,75,35,0,Math.PI,false);  // Mouth (clockwise)
ctx.moveTo(65,65);
ctx.arc(60,65,5,0,Math.PI*2,true);  // Left eye
ctx.moveTo(95,65);
ctx.arc(90,65,5,0,Math.PI*2,true);  // Right eye
ctx.stroke();
```

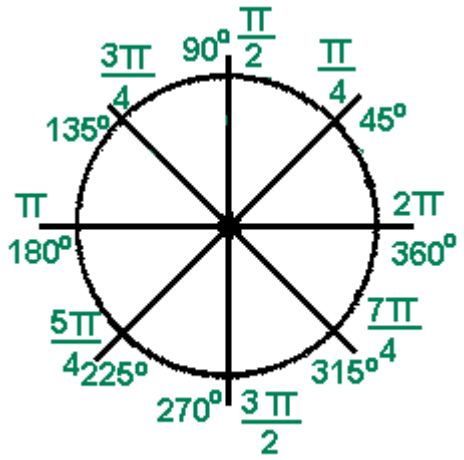
HTML5 Paths - Lines

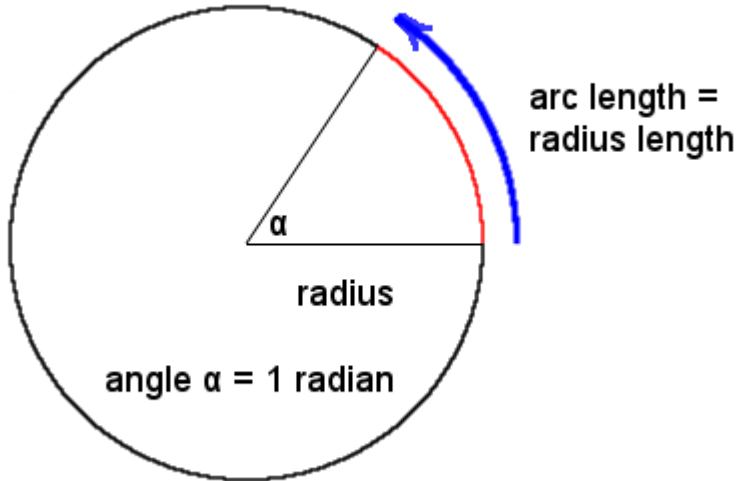
- Drawing straight lines- ctx.lineTo(x,y);
  - x,y are the coordinates of the line's end point.
  - Starting point is dependent on previously drawn path/pen position

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```

HTML Paths – Arcs and Circles

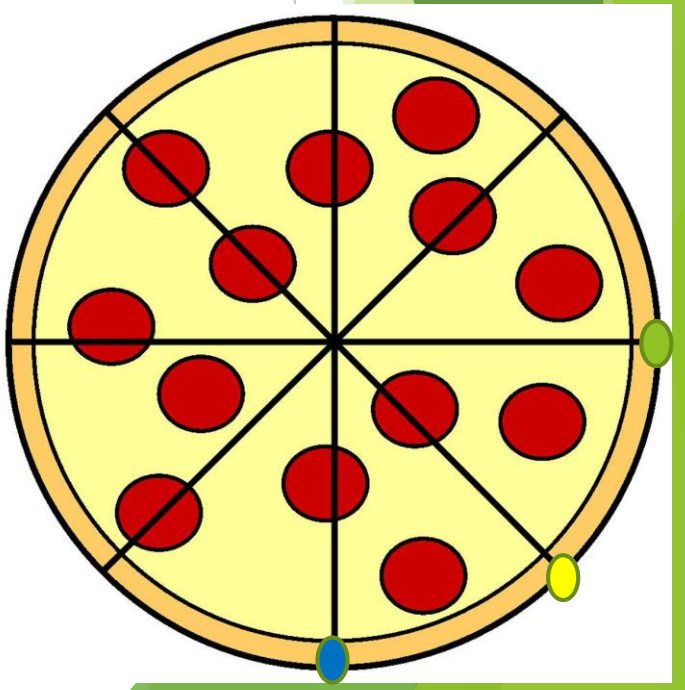
- arc(x, y, radius, startAngle, endAngle, anticlockwise)
  - The anticlockwise parameter draws the arc anticlockwise (true); otherwise, clockwise (false)
  - The startAngle and endAngle define the start and end points of the arc in radians
  - Radians:
    - JavaScript to convert degrees to radians: radians = (Math.PI/180)*degrees.





Polar to Cartesian Coordinates

- Locating x,y points on the curve, given the angle and the radius
  - $x = r \times \cos(\theta)$
  - $y = r \times \sin(\theta)$
- If radius is 40 and centre is at (50,50), what is x,y location of , ,
- Lab assignment - Javascript
  - Use variables
  - Use objects
  - Use functions



Javascript Recap

- Lab assignment - Javascript
  - Variables
    - var xPosition = 1;
  - Objects

```
var dog = {
  name: "Toby"
  , breed: "Labrador"
  , age: 5
};
```

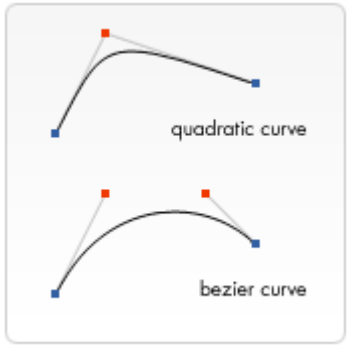
  - Functions

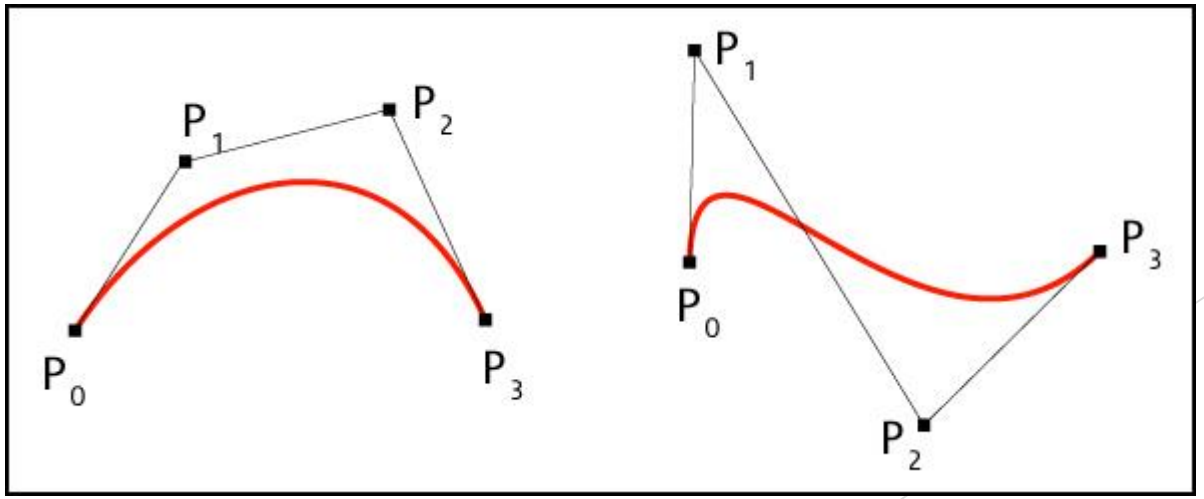
```
var no_dogs = 1;

function increasedogs(byhowmany) {
  no_dogs = no_dogs + byhowmany;
}
```

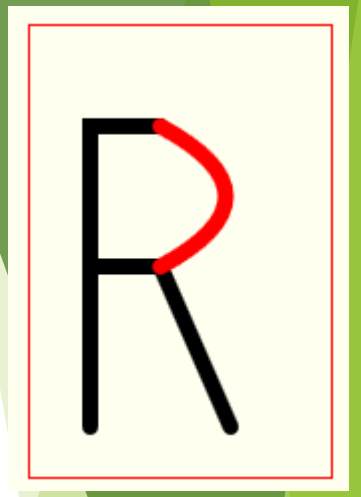
```
var dog = {
  name: "Toby"
  , breed: "Labrador"
  , age: 5
  , add_to_age: function(years) {
      this.age = this.age + years;
  }
};
```

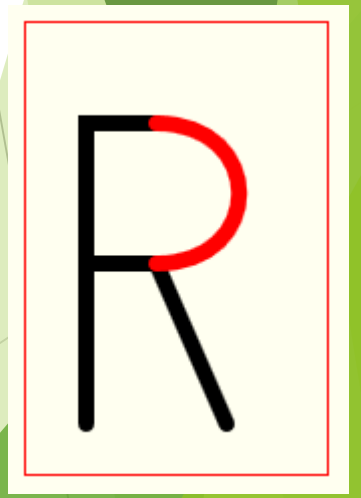
Cubic and Quadratic Bezier Curves

- Bezier curves used to draw complex shapes
  - Quadratic Bezier has only 1 control point
    - quadraticCurveTo(cp1x, cp1y, x, y)
  - Cubic Bezier has 2 control points – therefore more flexible
    - bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)



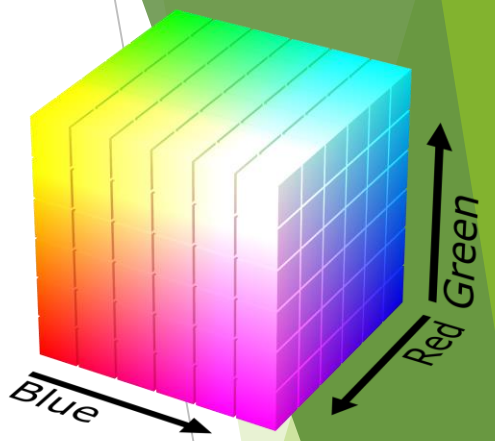




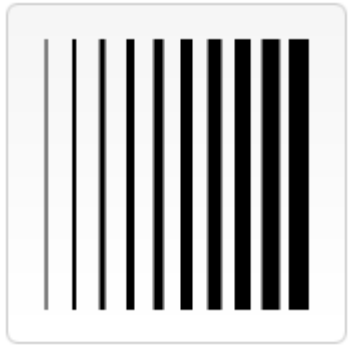


Colours

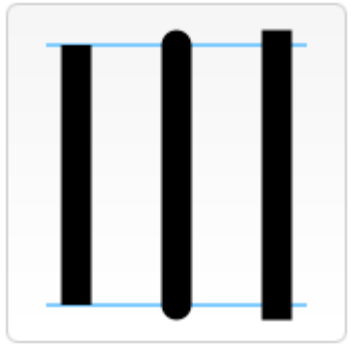
- Two properties for applying colours
  - ctx.fillStyle = colour
  - ctx.strokeStyle = colour
  - Colour represents a CSS <color>
  - Set to black as default = "rgb(0, 0, 0)";
- Transparency
  - For drawing opaque shapes to the canvas
  - CSS RGBA colour values
    - Between 0.0 (fully transparent) to 1.0 (fully opaque)



Line styles



- Line thickness
  - ctx.linewidth = 1;
- Line ending
  - ctx.lineCap = butt/round/square;
  - butt
    - The ends are squared off at the endpoints.
  - round
    - The ends are rounded.
  - square
    - The ends are squared off by adding a box with an equal width and half the height of the line's thickness.



Linear and Radial Gradients

- createLinearGradient(x1, y1, x2, y2)
  - Creates a linear gradient object with a starting point of (x1, y1) and an end point of (x2, y2).
- createRadialGradient(x1, y1, r1, x2, y2, r2)
  - Creates a radial gradient. The parameters represent two circles, one with its center at (x1, y1) and a radius of r1, and the other with its center at (x2, y2) with a radius of r2.
- Assign colors to CanvasGradient by using addColorStop() method.
  - gradient.addColorStop(position, color)
  - Position is a number between 0.0 and 1.0 - defines the relative position of the color in the gradient
  - Color is a CSS <color>, indicating the color the gradient should reach

HTML5 Canvas Text

- Two methods to render text

HTML5 Canvas Text

- Two methods to render text
  - fillText(text, x, y [, maxWidth])
    - Fills text at the given (x,y) position. Optionally with a maximum width to draw.
  - strokeText(text, x, y [, maxWidth])
    - Strokes at the given (x,y) position. Optionally with a maximum width to draw.

```
1  function draw() {
2    var ctx = document.getElementById('canvas').getContext('2d');
3    ctx.font = "48px serif";
4    ctx.fillText("Hello world", 10, 50);
5  }
```

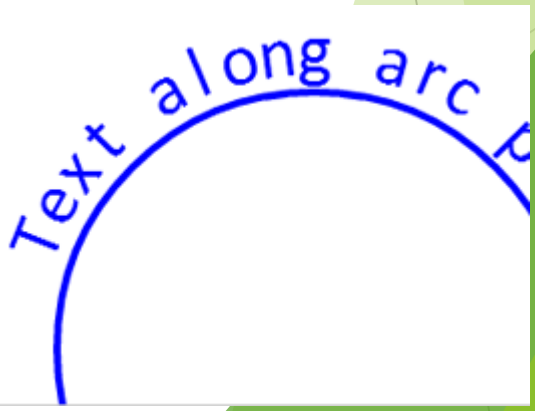
Hello world

```
1  function draw() {
2    var ctx = document.getElementById('canvas').getContext('2d');
3    ctx.font = "48px serif";
4    ctx.strokeText("Hello world", 10, 50);
5  }
```

Hello world

HTML5 Canvas Text

- Two methods to render text
  - fillText(text, x, y [, maxWidth])
    - Fills text at the given (x,y) position. Optionally with a maximum width to draw.
  - strokeText(text, x, y [, maxWidth])
    - Strokes at the given (x,y) position. Optionally with a maximum width to draw.
- Controlling font
  - ctx.font = value
    - The current text style used to draw text.
    - Same syntax as the CSS font property.
    - The default font is 10px sans-serif.
- More functionality offered in API



HTML5 Canvas – Using External Images

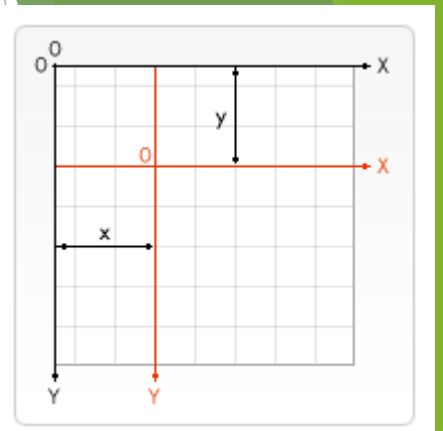
- Can be used as backdrops for graphs/games
  - PNG, GIF, JPEG supported
  - Frames from videos can also be captured
- Importing images into a canvas a two step process:
  - Get a reference to an HTMLImageElement object
  - Draw the image on the canvas using the drawImage() function.

```
1 var img = new Image();   // Create new img element
2 img.src = 'myImage.png'; // Set source path
```

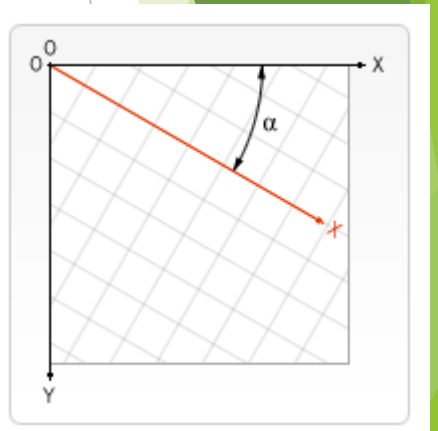
HTML5 Canvas – Using External Images

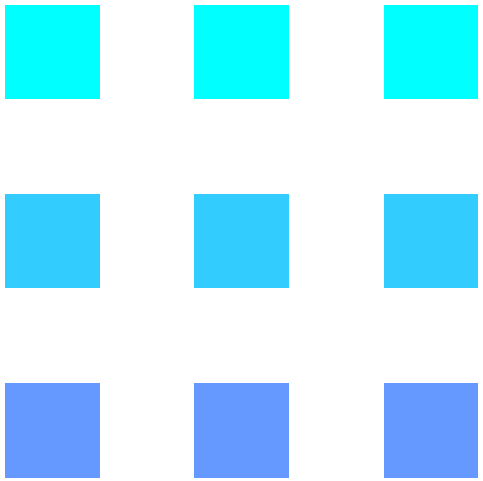
```
function draw() {
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext('2d');

  // Draw slice
  ctx.drawImage(document.getElementById('source'),
                33, 71, 104, 124, 21, 20, 87, 104);

  // Draw frame
  ctx.drawImage(document.getElementById('frame'),0,0);
}
```

Transformations



- Allows moving of the origin to a different position, grid rotation and scaling.
- Translation
- Rotation





Excellent Tutorials

- JavaScript refresher tutorial:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript
- HTML5 Canvas:
  - http://www.w3schools.com/html/html5_canvas.asp
  - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

HTML5 Animation

- Since we're using JavaScript to control <canvas> elements, it's also very easy to make animations.