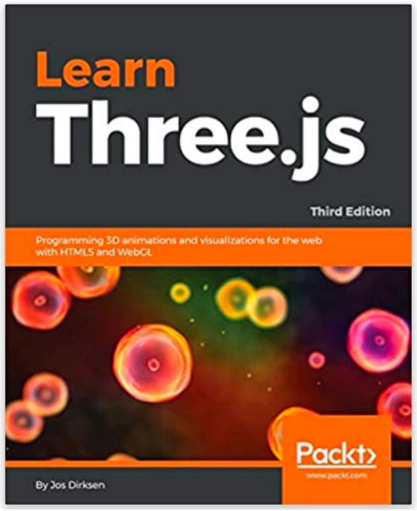
COMPUTER GRAPHICS


Learn
Three.js
Third Edition
Programming 3D animations and visualizations for the web with HTML5 and WebGL
Packt
www.packt.com
By Jos Dirksen

* Based on CISC 3620 material by Prof. Michael Mandel

JAVASCRIPT AND CANVAS DRAWING

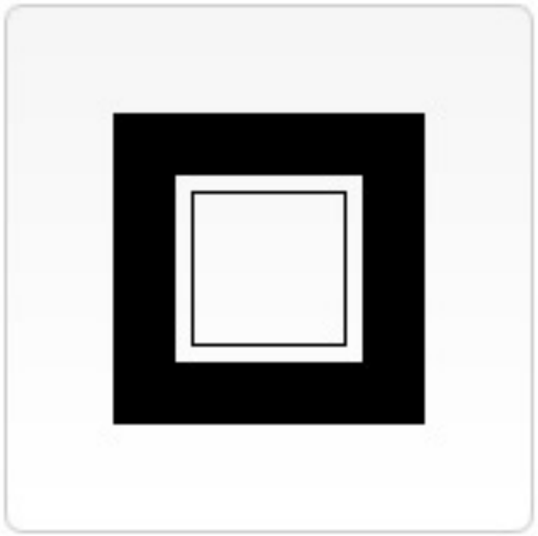
Based on this CS 307 reading and this CS 307 lecture*

Drawing rectangles - Exercise

- Your result may look like this

Example: Drawing rectangles

- Start from this codepen
- Try to draw this picture using the above rectangle functions



Questions?

Answer: Drawing rectangles

- function draw() {
- var canvas = document.getElementById('canvas’);
- if (canvas.getContext) {
    - var ctx = canvas.getContext('2d’);
    - ctx.fillRect(25, 25, 100, 100);
    - ctx.clearRect(45, 45, 60, 60);
    - ctx.strokeRect(50, 50, 50, 50);
  - }
- }

Drawing paths

- A path is a list of points, connected by segments of lines
- Segments can be different shapes, curved, straight, different colors

Drawing paths

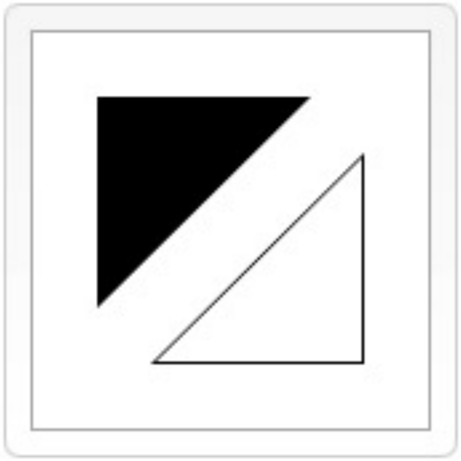
- To make shapes using paths:
  - Create the path using beginPath()
  - Use moveto(x,y) to go to the start point
  - Add segments to the path
    - Using lineto(x,y)
  - Call closePath();
  - Draw it using either:
    - fill() - solid shape, or
    - stroke() - just outline

Example: Drawing a triangle

- function draw() {
  - var canvas = document.getElementById('canvas’);
  - if (canvas.getContext) {
    - var ctx = canvas.getContext('2d’);
    - ctx.beginPath();
    - ctx.moveTo(75, 50);
    - ctx.lineTo(100, 75);
    - ctx.lineTo(100, 25);
    - ctx.closePath(); //closePath is optional when calling fill
    - ctx.fill();
    - }
  - }

Example 2: drawing 2 triangles

- Start from this CodePen
- Try to draw this picture using the above path functions



Questions?

Answer: Drawing two triangles

- Function draw() {
- var canvas = document.getElementById('canvas’);
- if (canvas.getContext) {
  - var ctx = canvas.getContext('2d'); // Filled triangle
  - ctx.beginPath();
  - ctx.moveTo(25, 25);
  - ctx.lineTo(105, 25);
  - ctx.lineTo(25, 105);
  - ctx.fill(); // Stroked triangle
  - ctx.beginPath();
  - ctx.moveTo(125, 125);
  - ctx.lineTo(125, 45);
  - ctx.lineTo(45, 125);
  - ctx.closePath();
  - ctx.stroke();
  - }
- }

Drawing arcs in paths

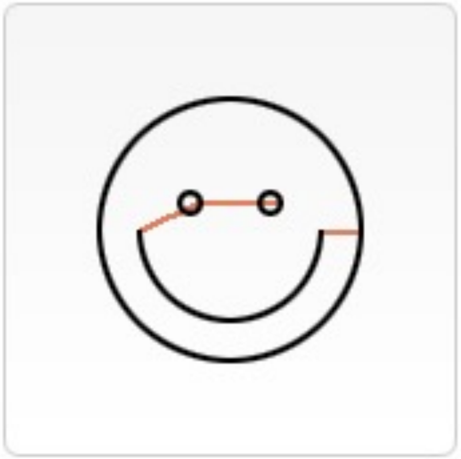
- There are two path functions to add arcs (portions of circles)
  - arc(x, y, radius, startAngle, endAngle, anticlockwise) draws an arc
    - centered at (x, y)
    - with radius r
    - starting at startAngle (in radians with 0 to the right)
    - ending at endAngle
    - going in the given direction indicated by anticlockwise (defaulting to clockwise)

Drawing arcs in paths

- arcTo(x1, y1, x2, y2, radius) draws an arc
  - starting at current point
  - going to (x1, y1) and then (x2, y2)
  - for a circle with radius radius
  - Angles in radians can be computed with
  - $radians = (\frac{\Pi}{180}) * degrees$

Example: **complete the smiley face**

- Start from this codepen
- Complete the smiley face to match this picture (don't worry about the orange lines):



Questions?

Answer: complete the smiley face

- Function draw() {
- var canvas = document.getElementById('canvas’);
- if (canvas.getContext) {
    - var ctx = canvas.getContext('2d’);
    - 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();
    - }
- }

Coloring shapes

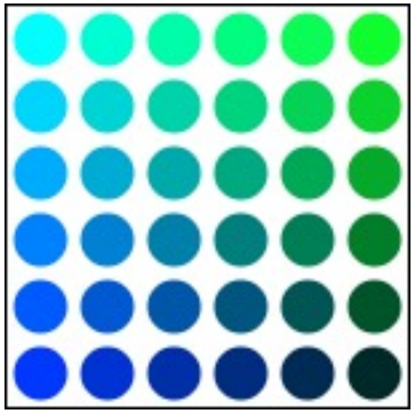
- Setting the context’s fillStyle property affects all future shapes
  - until the property is set again
  - Same for strokeStyle
- Can use any CSS color specification
  - ctx.fillStyle = 'orange’;
  - ctx.fillStyle = '#FFA500’;
  - ctx.fillStyle = 'rgb(255, 165, 0)’;
  - ctx.fillStyle = 'rgba(255, 165, 0, 1)’;

Coloring shapes - RGBA

- ctx.fillStyle = 'rgba(255, 165, 0, 1)’;
- Fourth parameter is alphe
  - Defines the opacity as a number between 0.0 (fully transparent) and 1.0 (fully opaque)

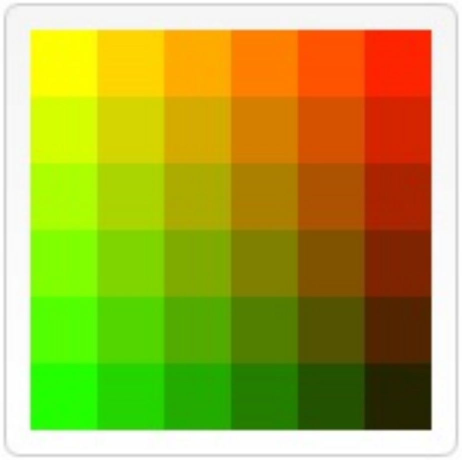
Example: coloring circles

```
Function draw() {
var ctx = document.getElementById('canvas').getContext('2d’);
for (var i = 0; i < 6; i++) {
    for (var j = 0; j < 6; j++) {
        ctx.fillStyle = 'rgb(0, ' + Math.floor(255 - 42.5 * i) + ', ‘ +
        Math.floor(255 - 42.5 * j) + ')’;
        ctx.beginPath();
        ctx.arc(12.5 + j * 25, 12.5 + i * 25, 10, 0, Math.PI * 2, true);
        ctx.fill();
        }
    }
}
```



Example: rectangle grid

- Start from this codepen
- Try to match this picture



Answer:

- function draw() {
- var ctx = document.getElementById('canvas').getContext('2d’);
- for (var i = 0; i < 6; i++) {
    - for (var j = 0; j < 6; j++) {
        - ctx.fillStyle = 'rgb(' + Math.floor(255 - 42.5 * i) + ', ' + Math.floor(255 - 42.5 * j) + ‘,’+ 0+’)’;
        - ctx.fillRect(j * 25, i * 25, 25, 25);
        - }
    - }
- }

Example: rectangle grid

- More info can be found here

Summary

- Get familiar with JavaScript syntax
- Data has a type
  - variables do not
- Functions are first-class objects
- Objects look like Java objects
- We can use the <canvas> element to draw many shapes

Questions?