ITMD 441/541
Web Application Foundations
Week 10

FALL 2023 – OCTOBER 30, 2023

Reminders

- Attendance is 15% of your grade and is being taken each week for in person students.
- Quiz Week 9 lecture. **Due: Thursday, November 2**
- Lab 4 will come soon
- **Final exam will be posted on Sunday, December 3 and due by Thursday, December 7 by midnight. (Final Exam week Dec. 4-9th)**
- **Very similar to the midterm except it will cover all class material and assignments. No project. 60-75 questions.**

Weekly's Agenda

- Finish Intro to Web APIs from last weeks slides
- Canvas API

- Back to last weeks slides to finish Web APIs

Canvas API

Canvas 2D, Web GL 3D

- The HTML Canvas API gives the developer a way to draw graphics using JavaScript inside an HTML <canvas> element
- The Canvas API is primarily a 2D graphics API
  - Useful for animations, games, data visualization, video and photo processing, and more
  - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- The WebGL API is used to draw 2D and 3D graphics using hardware-acceleration but still uses the HTML <canvas> element
  - API is similar to OpenGL ES 2.0
  - https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
  - Go down the page until you see EXAMPLES
- We will probably do another lecture on Canvas

Canvas 2D, Web GL 3D

- These are all APIs used for drawing on a <canvas> element on the page using JavaScript
- Canvas is a raster-based where the canvas objects are drawn in immediate mode. Once the item is drawn to the canvas it is forgotten. If you are doing animation, you redraw the entire scene each frame. **You can not modify something once it has been drawn to the canvas.**
- This is different than Scalable Vector Graphics (svg) which is vector-based and the shapes are remembered as objects in the DOM and then rendered to a bitmap on the screen when the HTML renders. If a svg object attribute is changed then the browser can automatically re-render.
- https://developer.mozilla.org/en-US/docs/Web/SVG/Element/svg

- There are many libraries for canvas that add scene-graph capabilities to the canvas element.
- Very low level api. Much easier to use a library, especially for hit detection.

Animation

- Animation can be canvas, web gl, or just dom manipulation
- Animation requires rendering a scene with changes at a frame rate
  - Old way is with setInterval() – new way is with requestAnimationFrame()

- https://css-tricks.com/using-requestanimationframe/

Canvas Basics

- Tag is canvas.
- Should set width and height in html but you can also change it with css.
- Need an id to select it.
- Fall back content can be provided inside the canvas tag.
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations

```
<canvas id="stockGraph" width="150" height="150”>
    current stock price: $3.15 +0.15
</canvas>

<canvas id="clock" width="150" height="150">
    <img src="images/clock.png" width="150" height="150" alt=""/>
</canvas>
```

Rendering Context

- You need to get the rendering context to run the drawing commands on
- You can test for canvas support by seeing if the `getContext` function exists

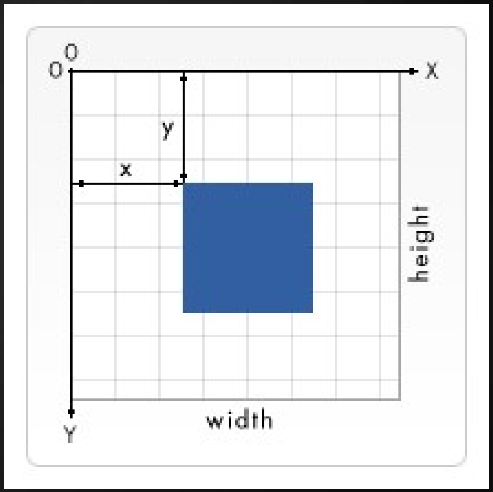
```
let canvas = document.getElementById('tutorial');
if (canvas.getContext){
        var ctx = canvas.getContext('2d');
        // drawing code here

} else {
        // canvas-unsupported code here
}
```

https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

The Grid

- The canvas is our coordinate space.
- The **top left is (0, 0)** and it is as wide and tall as the canvas element
- Normally one grid unit corresponds to one pixel
- Grid units can be fractional
- Be careful drawing lines that do not align with the grid lines. They can not cleanly convert to pixels.



Styles and colors

- In general these properties use css syntax
- Two basic ones
- fillStyle = color Sets the style used when filling shapes.
- strokeStyle = color Sets the style for shapes' outlines.

```
// these all set the fillStyle to 'orange’, ctx is the 2d context
ctx.fillStyle = "orange";
ctx.fillStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.fillStyle = "rgba(255,165,0,1)";
```

- Other properties described here.

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Applying_styles_and_colors

Rectangles

- Canvas only has one primitive shape, the rectangle
- These are functions on the context

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, making it fully transparent.

Vcode demo 1 and 2

1, 2

Paths

- The only other primitive is paths

1. Create the path
2. Use drawing commands to draw into the path
3. Close the path
4. Fill or stroke to render the path

Reminders

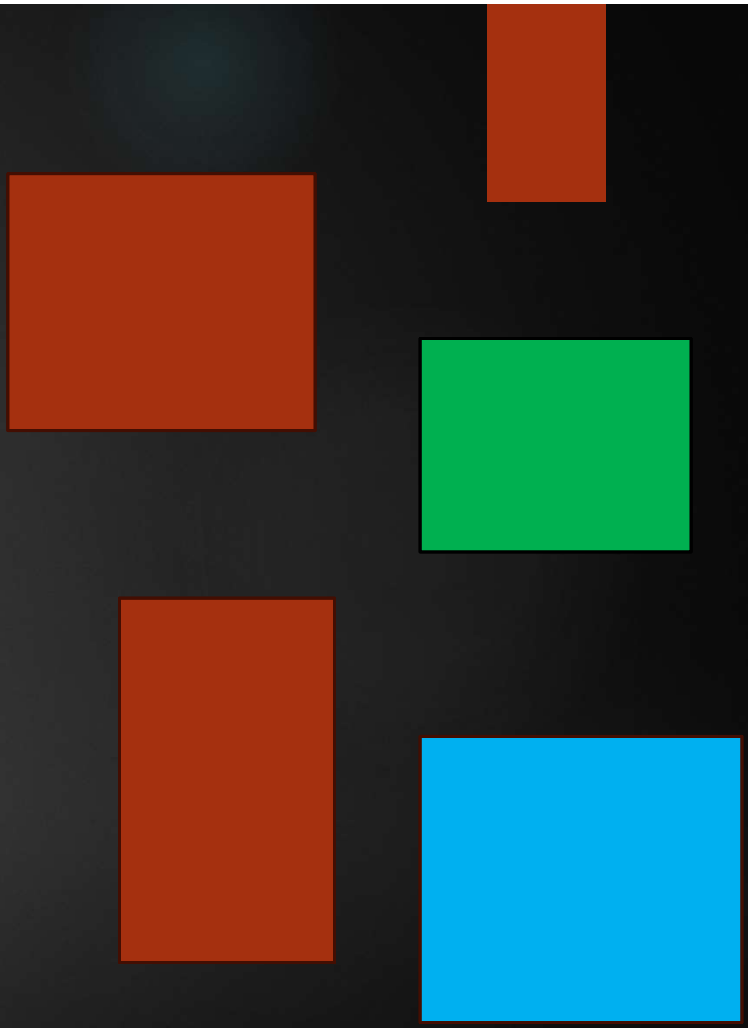
- Attendance is 15% of your grade and is being taken each week for in person students.
- Quiz Week 9 lecture. **Due: Thursday, November 2**
- Lab 4 will come soon
- **Final exam will be posted on Sunday, December 3 and due by Thursday, December 7 by midnight. (Final Exam week Dec. 4-9th)**
- **Very similar to the midterm except it will cover all class material and assignments. No project. 60-75 questions.**

Quick Review #1

What will the following code draw?

```
ctx.beginPath();
ctx.fillStyle = 'rgb(0,255,0)';
ctx.rect(45,45,100,100);
ctx.stroke();
```

A. Blue rectangle
B. Green rectangle
C. Red rectangle
D. Green square
E. None of the above

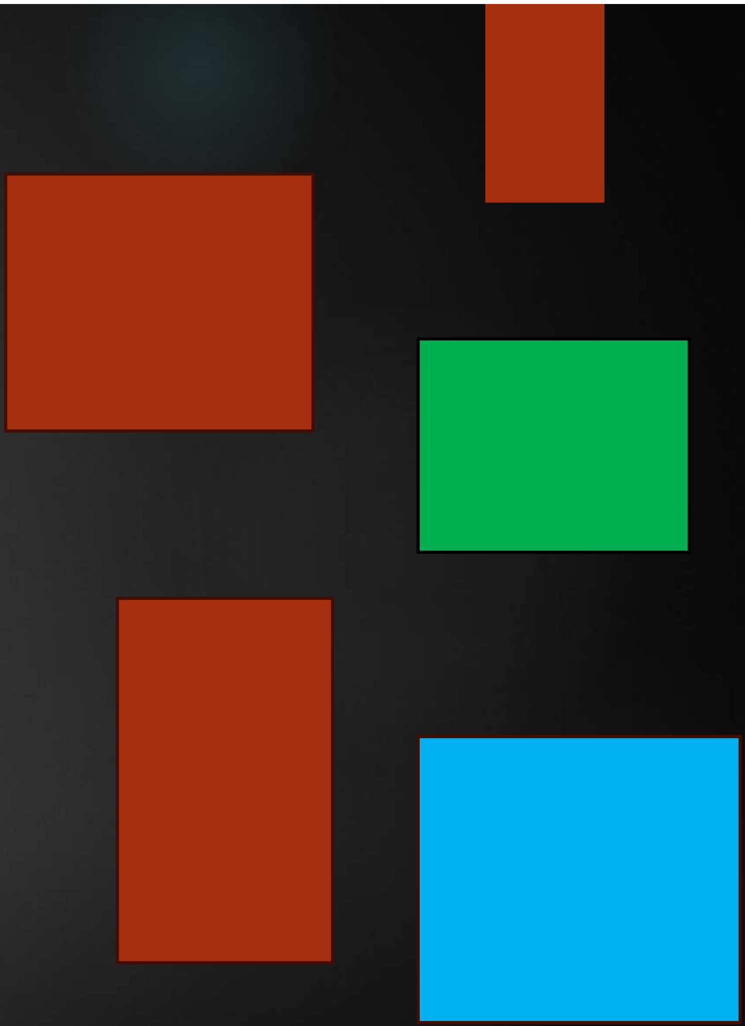


Quick Review #2

What will the following code draw?

```
ctx.beginPath();
ctx.fillStyle = 'rgb(255,0,0)';
ctx.rect(90,90,100,120);
ctx.fill();
```

A. Blue rectangle
B. Green rectangle
C. Red rectangle
D. Red square
E. None of the above



Paths

1. beginPath() Creates a new path. Once created, future drawing commands are directed into the path and used to build the path up.
2. Path methods Methods to set different paths for objects.
3. closePath() Closes the path so that future drawing commands are once again directed to the context.
4. stroke() Draws the shape by stroking its outline.

   fill() Draws a solid shape by filling the path's content area.

- **Vcode Demo 3**

Path Methods

- Move the pen without drawing
  moveTo(x, y)
- Drawing straight lines
  lineTo(x, y)
- Draw arcs or circles (measured clockwise from the positive x axis and expressed in radians)
  Convert to degrees with this formula: (Math.PI/180)*degrees
  arc(x, y, radius, startAngle, endAngle, anticlockwise)
  arcTo(x1, y1, x2, y2, radius)
- Bezier and quadratic curves
  quadraticCurveTo(cp1x, cp1y, x, y)
  - Change values to create an image for point and area calculations. Click on link to play in real time.
- bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
- Rectangles added to the current path vs rectangle primitive
  rect(x, y, width, height)

- **Vcode Demo 4,5 (3 versions), 6**
- **View code. Guess what it will look like before running it**

3,4,5,6

Path 2d

- The path 2d object allows you to cache or record drawing commands.
- All path methods like moveTo, rect, arc or quadraticCurveTo, etc., which we got to know above, are available on Path2D objects.
- Path2D() The **Path2D()** constructor returns a newly instantiated Path2D object, optionally with another path as an argument (creates a copy), or optionally with a string consisting of SVG path data.
- `new Path2D(); // empty path object`
- `new Path2D(path); // copy from another Path2D object`
- `new Path2D(d); // path from SVG path data`
- Can also take in svg data

Drawing Text

- Two methods for drawing text
- fillText(text, x, y [, maxWidth]) Fills a given text at the given (x,y) position. Optionally with a maximum width to draw.
- strokeText(text, x, y [, maxWidth]) Strokes a given text at the given (x,y) position. Optionally with a maximum width to draw.
- measureText() Returns a TextMetrics object containing the width, in pixels, that the specified text will be when drawn in the current text style.
- Text properties here
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_text
- **Vcode Demo 8**

Images

- Importing images into a canvas is basically a two-step process:
- Get a reference to an HTMLImageElement object or to another canvas element as a source. It is also possible to use images by providing a URL.
- Draw the image on the canvas using the `drawImage()` function.
- Be careful to wait until the image is loaded before using. Helpful to use the images onload attribute to trigger a callback function
- `drawImage` function has a few different signatures

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Using_images

- **Vcode Demo 9**

Transform

- To do transforms you must transform the context or grid and then draw into it. When you are done you need to undo that transform.
- The save and restore function are helpful there and can also be used to save other context states like colors.
- save() Saves the entire state of the canvas.
- restore() Restores the most recently saved canvas state.
- We will look at the transforms here

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations

- **Vcode Demo 10**

Basic Animation

1. Clear the canvas
2. Save the canvas state
3. Draw the animated shapes
4. Restore the state

- You also need a way to trigger your draw frame function on an interval
- setInterval(function, delay) Starts repeatedly executing the function specified by function every delay milliseconds.
- setTimeout(function, delay) Executes the function specified by function in delay milliseconds.
- requestAnimationFrame(callback) Tells the browser that you wish to perform an animation and requests that the browser call a specified function to update an animation before the next repaint.
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations
- **Vcode Demo 11**

Canvas Libraries

- There are canvas libraries that make this easier
- EaselJS is one that turns the canvas into a retained mode drawing context
- http://www.createjs.com/easeljs
- http://code.tutsplus.com/tutorials/using-createjs-easeljs--net-34840

WebGL

- WebGL is a JavaScript API that allows 2d and 3d rendering using an API very close to OpenGL ES 2.0 and usually uses hardware rendering in compatible browsers.
- Difficult to work with if you are not familiar with OpenGL
- https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
- There are libraries that make it easier but still much more complex than canvas 2d
- ThreeJS is one of the most popular
- http://threejs.org/

Canvas Resources

- https://en.wikipedia.org/wiki/Canvas_element
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shape
- https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D
- Many canvas libraries, some are listed on the Canvas_API page above, another is easeljs http://www.createjs.com/easeljs
- Web GL is a 3d implementation of canvas
  - Conforms closely to OpenGL ES 2.0
  - https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
  - Libraries to make it easier like three.js http://threejs.org/

Assignments

Reading/Assignments

- Read Eloquent JavaScript Ch 17 on Canvas
- Lab 4 will come soon
- Quiz 8 will come this weekend to cover tonight's materials