Tutorial Report

Catherine Yim
cyim@oxy.edu
Occidental College

1 Introduction

This tutorial guides the reader through creating a simple game using HTML5 and Javascript. The purpose of this exercise is to get the user familiar with the concepts of catering to the multitude of viewports and optimize for mobile; looking briefly at using the canvas API to draw to the screen; capturing touch events; making a basic game loop; introducing sprites, or game "entities"; adding collision detection and some simple math to spice things up; and adding a bit of polish and some basic particle effects.

2 Methods

2.1 Set-Up

Using the tutorial, I have started to learn how to make a mobile app. The first step was to set the stage. First, I would have to choose resolution sizes and pixel dimensions which are highly reliant on the device and platform of the project. I am tentatively using an android platform. To fit the viewpoint of our canvas by making the canvas small and then scaling up to prevent the loss of quality and performance. I created an HTML file using the meta viewport tag. This allows the browser to disable user scaling, meaning that the page cannot be shrunken down and interfere with the game.

I chose to mainly focus on an Android platform for my project, but to follow the tutorial and all of its resources, I used the apple- prefixed meta tag. This tag allows the game to be bookmarked on the iPhone.

As mentioned in my comps proposal, it is important to create a name-space for our game, first and foremost to avoid polluting the global one. Since our game focuses on Sue the Dog, we create a namespace called Sue. As any other project, we start by declaring our variables. Most of these variables just set the dimensions of our application. In the Sue.init, I adjusted the element's dimensions to 480 x 320 by grabbing a reference to the canvas element and getting its context. I also created a resize function to easily adjust the sizing without disturbing the ratio, and scaled it up using CSS.

There is a mention of getting rid of the address bar, but I did not want to do that just yet. However, for future reference, one would go about this by adding a few more pixels to the documentation and scrolling down. In order to do this, we must also write a couple lines to detect what type of user agent we are facing to account for resizing. This portion of the code would be at the end of Sue.init, before we call the resizing function.

2.2 Draw Object

Next, I created a Draw object that simplifies a few things. Currently, I have the ability to draw simple shapes (circles and rectangles), clear the screen, and add text. This allows a faster coding experience for myself as I do not have to call the canvas API manually anymore.

2.3 User Action to Input

In order to make the tapping action from the user affect the actual game as an input, and ultimately provide output. Morse specifically, in this web app, for the user to be able to "jump", they must be able to tap the screen and have that action interact with the actual game. We call this a touch event. I will create an array for touch events, each element of which contains touch coordinates and other data. I'm not entirely sure on the specifics of that yet, but for now, to become more familiar with this action, I created a touchstart, touchmove, and touchend events using arrays. It is important to call a function that prevents the user from being able to disable actions that might interrupt the flow of the game such as zooming and scrolling at this stage. I then passed the event data into an Input object that I will define later. I, then, ran into a problem of the shapes not appearing on the screen. This was due to the fact that I scaled the canvas, and now I have to account for mapping by subtracting the offset from the coordinates. Instead of calculating each touch event per rescale, it would be more efficient to adjust it after the rescale, so within my rescale function, I added the current scale and offset to refer back to them in the set method.

2.4 Game Loop

Using a loop, I accounted for the user input to update characters and process collision and repeat. This essentially

connects the user to the game and progresses the game. Within my loop, I used the setInterval and requestAnimationFrame. The issue with requestAnimationFrame is that it does not consistent across browsers, but I did not account for that just yet. At the end of Sue.init, we call the loop which in turn calls our update and render methods, and then the loop is called again.

There's an issue that occurs at this stage. I did not have a way to store items that were drawn on the screen, therefore every touch was deleted when I advanced in steps. In order to solve this issue, I will create an array that keeps track of all touch actions and positions. Then, I will create a Tap class that makes the avatar move up on the screen, mimicking a jumping effect.

3 Evaluation Metrics

It is difficult to evaluate the effectiveness of what I currently have because it is just the set-up of the application. The only way that I can evaluate the whether it worked or not was to play it over and over again.

4 Results and Discussion

However, I've tested it multiple times, and the program runs the way that I want it to However, it is unclear whether this framework will work perfectly with my specific game design. This is because I'm still not entirely sure on how to display the code editing part of my project onto a web-app. However, I'm fairly certain that I can use the majority or recycle a big part of what I wrote into my project.

So far, I have obtained four objectives: set-up, draw object, user action input to output, and game loop. I'm fairly certain that the set-up can be used as it is because it is essentially creating the application and setting dimensions. This should be the same for any app. My Draw object will also be useful because this portion of the code is abstracting simple operations, making a short cut. Simple games almost always operate on loops, and the loop method that we created is very basic and can at least be modified to fit my project. The issue is the user input to output section of my code. The tutorial that I followed guided the reader through the making of a game that in which the user bursts floating bubbles before they reach the top of the screen. This is slightly different from my project where the player taps on the screen to have the bird jump. This means that the touch actions are different. Once I complete the loop portion of the code that I outlined above, I will be able to know if that portion of the code can be altered to fit my particular project.

This tutorial was very useful in helping me get started in understanding mobile application development. It provided a good framework to begin my project, however, there are a lot of components to my project that I'm still unsure on how to conduct.