NM2207 WRITE-UP

As per NM2207 requirements, this is my final write up on the game ‘Treetop Trouble’ which I’ve been able to create with knowledge garnered from this course. In doing so, I hope to reveal my motivations for coding out said game.

# **Overview and Description**

The original ball-and-paddle game was Breakout by Atari. A popular game format, the goals of a user is to use a paddle to bounce a ball into a set of bricks, which on collusion would demolish the brick, simultaneously they must ensure the ball does not fall beyond the paddle, otherwise, they would lose a life. Players win when they clear all the bricks and lose when they lose all their lives. The orientation of such games usually played vertically and the user controls the paddle via the arrow keys.

# **Detailed Description**

My final project includes both flipping the game horizontally while also allowing for an element of collaboratives, or competitiveness. Mashing the concepts from the popular game of “pong” into the ball-and-paddle format. I imagined that player(s) can control a right paddle via the ‘up’ and ‘down’ arrow keys and the left paddle via the ‘w’ and ‘s’ character keys. The bricks would be centred in the middle of the play area and player(s) must ensure that the ball(starting on the right paddle) does not fall beyond the x-axis of their respective paddles. The goal of the game varies on the number of balls in play and if the player(s) are playing competitively or collaboratively. There will be audio effects that accompany ball interactions with both bricks and paddle. Ultimately, the affordance of dual controls in the game ‘Treetop Trouble’ allows for multiplayer or a challenging single-player experience.

Assets:

-Ball

-Left paddle

-Right paddle

-Brick Array

-Game area + Border

# **Required Elements**

The following elements are incorporated into the game as per module requirements

Grid Layout in CSS Grid – HTML + CSS + Rapheal paper.

Arrays & Loops – Used for generating and re-generating bricks in a ‘multidirectional’ array, also used in ball and brick interactions.

SVG Graphics – Rapheal generated objects, i.e. paddle, ball, and bricks + interactions between these objects.

Javascript Timer Animation – Used in in-display time and to control game start and reset.

# **Computational Logic and Skills**

The game of Breakout appears simple at first glance but after mapping out the events such as interactions between balls and wall, balls and paddle, balls and bricks. After planning out the required conditional events, there was still a need to implement an ‘interval’, to control the game state, alongside an array for brick generation, as well as a means to control the logic behind ball collisions, of which combined proved more challenging than expected.

Besides these challenges, I’ve also found that I also needed to include the means to start/reload the game upon mouse click. I've nested various conditional functions and loops (if, else, for), controlling game state with an interval to do so.

It took me a while to wrap my head around how to handle the brick generation and various ball collisions. Hard coding the brick array was a futile effort that I spent too much time in, believing initially that I did not need arrays, thought they worked... in a way.

However, after being realizing that arrays are necessary I decided to look into how I could adopt it. With help from Google and Stack Overflow, I found the relevant code of a ‘multidimensional’ array which helps me form an initial idea of how to tackle this issue.

Next would be how should I program the model to know where the ball collides in the view? After some googling, I landed on this [page](https://developer.mozilla.org/en-US/docs/Games/Tutorials/2D_Breakout_game_pure_JavaScript/Collision_detection) from Mozilla that helpful laid out how to handle collisions and due to the relevancy of the example, the generation of the brick array as well.

At a point, the ball collisions became too predictable, and you could win by literally not moving your paddle. Hence, I thought that I can change the angle of the bounce the ball’s reverse X and Y rate and hence bounce angle by adding or subtracting from the ball’s reverse X and Y rate post-collision, I further experimented with the angle and speed off all Raphael objects to land on something that is pretty unpredictable and capable of keeping the player(s) on their toes.

**Shortfalls**

           This game is pretty hard to make responsive for various screen sizes due to the way the elements are generated if I could I would but I think it’s not within my levels of expertise now.

I have intended to use .png files as a substitute for the Raphael objects generated in my main.js file (I was so optimistic then). After being about 80% done I realized that to so do would be problematic at best due to how such graphics have to be generated as its own Raphael objects. Doing so would require a substantial re-thinking of the code and accounting in the differences in object interaction (i.e. a Raphael paper.dot object is different from a Raphael object of a paper.imgage from a png of a ball) of which is harder to control.

The completed project has two paddles and one ball. Most people I’ve shown my project suggested an ‘easy’ and ‘hard’ mode, in which the ‘easy’ mode is the original game’s config and the ‘hard’ mode would include two balls spawning on each side of the game background, of which I enthusiastically agreed knowing that that will prove a substantial challenge in coordination for my player(s). Hence, I tried to implement this. However, as I was doing so I faced numerous bugs (balls exiting the game area, balls not bouncing off bricks, and so on) and I unhappily had to remove this feature due in part to due to pressures from submissions from other mods.

# **Reflection and Conclusions**

This whole module was incredibly difficult when taken in context, but knowingly I took it up convinced that being able to code or at least being able to understand/read code is vital for my future career prospects.

 I always had a fear of coding, this impression was garnered from speaking to friends in computing and developers I’ve had the honour for working alongside with after which I garnered that I had not much of an aptitude for it.

Despite maintaining that one should invest in their strengths as it will be your strengths that carve out your niche, in my case it in the creative field; I surprised myself by deciding to read NM2207 for I was certain I’ve consigned myself to a passing grade. Regretfully I believed I should really have taken a module I was good at, but hey since I had S/Us left why not try something truly challenging (even now I regret how much time I spent) and infinitely useful.

I pre-studied some HTML CSS and started pretty confident but as the workload from other modules and JS is introduced, I felt myself drowning, just a little bit; since then I admit that I became more disheartened and really, really tired.

If it was not for the possibility offered by the final project I might have just ‘given up’ then. But the notion of being able to code a game to life really appealed to me and hence I decided to attempt to tackle the final project early as so to remain motivated enough to continuing threading water and get a headstart I'm sure I needed on it; I’m glad I did for this headstart proved exceedingly necessary to work out JS collision events of which is only briefly covered in course materials.

 In retrospect, I felt that I tried really hard to deliver a strong final project but during the whole process, I found myself lacking in how I approached computational problems, being constantly frustrated and taking longer and making less progress than I would like to have.

In a way, I’m glad that I had this experience as it taught me to read these three languages(HTML, CSS, JS), although code does not come intuitively to me, I'm still glad to be marginally better than completely illiterate.

Despite my frustrations, misgivings, and despair I’m certain that I have learned a skill that would be invariably useful in any field I chose to enter.