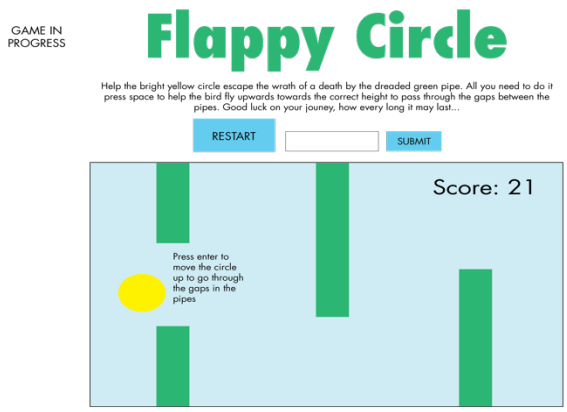
Code Critique Final Project- CJ Andrews

1. For my final project, I created a flappy bird clone game in which a player tries to fly their circle between pipe gaps in order to avoid hitting a pipe and losing. For each pipe they pass, they gain a point to their score. After the game is over, their score is recorded. If they decide not to enter a name to the scoreboard their player name is recorded as "Anonymous Circle". However, if they enter a name, the score board is updated to use their given name. I set up the name of the game and the instructions in html. Additionally, I wrote html code for a restart button and an input text field with a submit button for entering the player's name. I then styled these buttons and the text written with CSS to make the padding more normal and aesthetically pleasing. In order to put the buttons and fields on the same line, I used divs in the html to group these elements together. Also, I used CSS to create hover over color change animations for the buttons (lines 66- 72). From there, I wrote JavaScript to handle what was going on in the game canvas. First, I set variables that the game would run off of (lines 108-145). I then used setInterval to run the ticks for my game animation (line 122). For each tick, I run the main method for my program (lines 125-133). I check in the main if the game is over, and if it isn't currently, I check to see if the game has ended with my gameOver function (line 258). The gameOver function checks to see if there was a collision between the circle and any of the pipes in the game or if the circle has fallen off screen. If either of these conditions are true, I add a new entry to the score board with the most recently entered playerName and update the score board so that it sorts the new score record at the correct ranking and removes the lowest falling score from the scoreHistory record list if the list is already full. If the user submits a name using the input field and submit button, the entered name is added to the most recently added score entry using the event listener on line 100. If the game was not determined to be over, new pipes are generated when the newPipeCount reaches 20 via the setInterval ticks. The bird also falls once birdFallCount has reached a limit. These counters are returned to zero once they reach the target count. The pipes are also moved left based on the tick. If the user presses the space bar, the bird flies up as detected by the listener on line 154. When pipes go off screen to the left, I also remove them so as to not overload the game with unnecessary pipe data. Lastly, the vector graphics for the game are drawn within the canvas using D3. The pipes are drawn at their current coordinates with data joins and the bird is drawn at its current position.



2. These sketches align pretty closely with how my final project turned out. I changed the font used for the game and also added an instruction to “Enter Player Name” above the text field and the submit button so that it was clearer what those elements were used for. Additionally, I improved the graphics for the game by adding tops to the pipes and shading for where the light is hitting the pipes. These were done by creating. Rectangle with the same height as the pipe but with less width. Additionally, I added an inner circle to the circle to highlight that the circle is important to the game and improve the aesthetics. I also added a faded blue background to the game in order to improve the overall look.
3. One major issue is that the restart button remains selected once it is pressed and when the user goes to press the space bar to fly the circle up, if they don’t click on the canvas, the restart button is pressed again which is confusing without me explaining how to fix it. Additionally, the graphics for the bird are kind of jumpy and buggy because I have the circle falling and jumping at large intervals. This takes away from the game but doesn’t affect game play too much. On the bright side, I think the game is very intuitive to use besides these issues. The directions are very clearly marked and explain how play the game in a simplistic manner. Labeling has made the game more accessible to the user. The game is not overly complicated so there isn’t much to be confused about. I think that adding the enter player name section was good for game play as it allows the player to add stakes to the game and try to beat their friends in the game. Additionally, I like that the game saves the most recently entered player name for the next round so that they don’t have to reenter their name each time they lose.
4. I constructed my application is a way that was familiar to me because of my experience with OOD this past semester, I found it more familiar to separate the data updating for the pipes and birds in my javascript code from drawing the data in my code. This allowed me to focus on these two aspects with separate code blocks that interact minimally with each other. This made it simpler for me to debug as well as I could separate view problems from data model problems. The use of functions in my code was an important code pattern because I was able to call the functions when I needed to based on the timing. If statements were very important to my code as I needed to check for specific conditions in many instances throughout my functions. This is seen when I used an if statement to check for the two conditions that would result in the game ending (line 260). I also used for loops and data joins to apply code to the current

list of pipes in the game. This was necessary as the list could vary in length so the for loops and data joins took into account the length of the list as it changed with time.

5. Focusing on pipe movement, I manipulated the xCoord field in the Pipe data structure to change by 10 every tick (line 163 and 204). This was a very straightforward way of changing the pipe xcoord and made the pipes have a very steady moving animation that looked more realistic than the bird, which didn't fall with every tick. The use of setInterval for this data manipulation was very successful. When the pipes xCoord data was manipulated to the point that a pipe went off screen, the pipe was removed from the list of pipes in the game (line 242) so as to not make the list of pipes unnecessarily long. This was a smart move because if this was not done, the game would likely crash after some time because there would be too many pipes in the game to deal with. I don't think this part of the code could be made more efficient as the way I have done this was the most simplistic way possible in that the pipes move with the timer and efficient because I remove the pipes when they're too far off screen.
6. I think it may have been easier to use modulo to deal with the counting for making the bird fall (line 181). This would have made the counting for when to do certain changes to the data more simplistic and easier to understand from an outside perspective. Additionally, I would'nt need to set the birdFallCount back to zero as modulo looks for the remainder, which would be more elegant and efficient in reducing the number of lines of code. This is how I would have implemented this change:
    a. function fallDown() {
        birdFallCount++;
        if (birdFallCount % 3 == 0) {
            circleHeight = circleHeight + 30;
        }
        }
    b. This change reduces the number of lines and is easier to understand if you know what modulo is and how it functions.
7. Instead of separating the data from the view and just drawing the current location of the circle and the pipes with d3, I could changed the x position of the pipes using D3 and y position of the circle using d3. Using transitions would have minimized the amount of code I would need to do on my end and could have used the built-in shortcuts provided with D3. Should I have used D3 transitions for the movement, the pipes and circles may have moved more realistically and with less jumping as you can also have svg shapes move gradually or speed up to mimic acceleration and make thing more lifelike. Additionally, I could have used a random number to generate the pipes at a random coordinate and with a random height instead of having 5 set types of pipes in the game as I did with makeNewPipe() in my code. This would have introduced more variability to game play and made the game less predictable.
8. One code pattern that surprised me by how hard it was to work with was Arrays. I was having many issues with getting the scoreboard to update itself using insertion sort. I had implemented this in past CS classes but for some reason doing it in javascript was difficult for me and presented many obstacles. I had to work a lot with indexing for my while loop on line 282 so that the new score was placed in the correct high score slot

and previously recorded scores were displaced to the correct slot and deleted if the list was full with the new score added. I learned a lot about mutation as I needed to save the previous next entry in the high score list so that I could use it when dealing with what score to put in the following high score slot. Mutation requires complicated thinking and analysis to make sure it's working properly.

9. One thing that could be improved is the collision checking as when a circle goes through the pipe gap if the user presses space it is very easy for the circle to go too high and have the game ended even if the circle was just barely touching the pipe. If given more time I would try to make this more lenient and also improve this by lowering the height gained by pressing space as it seems to be too much. This leads into the other issue with how high the circle flies up when space is pressed. I had to make this value big so that it counteracted the amount the circle falls by. The amount the circle falls by had to be big because I was unable to implement acceleration to mimic gravity as the bird is falling due to my lack of physics knowledge. As these things were all connected, I tried my best to get the game to a point where it was playable but also challenging with my current skill level. If I had acceleration working, I could make the bird fall less fast at first and grow this falling as time goes on due to acceleration. This would allow me to make the fly up value to be less high and this improve pipe-circle collision checking.
10. As mentioned in my previous response, I wish I could have implemented acceleration in terms of how the bird fell to mimic gravity. This would have made the game feel more realistic and would have brought the game closer to feeling like a real flappy bird clone. In order to do this, I would have to do a brush up on my physics knowledge or reimplement the movement of the circles in game with D3, instead of regular JavaScript, as there is most likely a provided way of making a shape move with acceleration in the library. Due to time constraints I wasn't able to do either of these but doing so would have vastly improved the playability, quality, and realism of the game.