**Technical Specification for Ethan’s Bird Game**

**Start:** The start screen just presents the start screen of the game and the phrase “Press enter to start single player or press Q for multiplayer mode”, which, when pressed, calls the start () method, which then begins the movement of the obstacles toward the sprite(s)

**Bird** (Main Character)

Generated by createBird () method, triggered by pressing a start key

Controlled by WASD keys on keyboard

W key: flyUp ()

Method moves bird a small distance up the screen to avoid obstacles

S key: flyDown ()

Method moves bird a small distance back down the screen to avoid obstacles

D key: moveForward ()

While key is held, bird continues to move forward

A key: flip ()

When A key is pressed, bird does a “trick”, which is essentially just flying around in a small circle; if the bird can perform this trick while also avoiding obstacles, then it acquires extra points in the form of 5 coins

WWW command: diveDown ()

When the W key is pressed three times in a row, the bird dives down much quicker than flyDown () would allow, which could be necessary as the game progresses and gets increasingly difficult

**NOTE:** this may be difficult to program and it may be much more simple to just use the Z or X key to trigger these movements

SSS command: soar ()

When the S key is pressed three times in a row, the bird moves up much quicker than flyUp () would allow, which could be necessary as the game progresses and gets increasingly difficult

**NOTE:** this may be difficult to program and it may be much more simple to just use the Z or X key to trigger these movements

**Various Obstacles**

**Car:** Small object on the road surface, which the sprite has to use the flyUp () method through the W key to avoid

Generated randomly at a set interval using createCar () method, which also slowly moves the car up the screen towards the sprite

If there is a collision, collide () method calls the death () method, which ends the game and displays the distance covered by the bird and the coins picked up while it was alive

**Truck:** Larger object on road, which approaches slower than a car, but has the same affect, using a createTruck () method to generate randomly

Employs same collision sequence as a car if it comes into contact with the sprite

**Signs**: Signs are randomly generated by sign () method, which uses Math.Random to create different height signs each time; sprite must use the flyUp () method to avoid

Employs same collision sequence as a truck if it comes into contact with the sprite

**Tunnel:** circular tube generated by tunnel () method, which approaches slowly, but is the only way through, so the sprite must use flyUp () or flyDown () to survive

Employs same collision sequence as a truck if it comes into contact with the sprite

**Hunter** (Experimental Feature)**:** Figure randomly generated by hunter () method, which sits on the ground level and shoots bullets up at a fixed angle at a set interval towards the sprite

This obstacle would only appear in the more difficult levels of the game and would employ the same collide () and death () sequence as the other obstacles if it is run into

**Coins:** Secondary objective(collected as game progresses)

Continuously as the sprite moves along, coin objects will be generated using the method coin (); sprite will have to maneuver using the aforementioned methods to collect as many as possible

If the sprite reaches a coin, the collect () method is called and the coin object disappears and the coin count is increased by 1

Score () method keeps track of the points and progress during the game in the top, far right corner of the screen and if the death () method is called, the data from score () is transferred to the final score presentation

**Levels**

When the sprite covers a certain set distance, it will automatically pause the game using a pause () method and will display the following options:

Quit (): this method just saves the current score of the player and quits the game

levelUp (): this method allows the sprite to continue, but the speed with which the objects approach is increased, thus making the game more difficult

**NOTE:** this could potentially be difficult to code because the levelUp () method would have to edit the time integer in all of the obstacle methods, which controls the speed with which they approach

**Multiplayer**

If the player presses the Q key on the start screen, it calls the multiplayer () method, which creates a secondary sprite with the same function as createBird (), but is called newBird () and generates a different colored sprite controlled by the arrow keys in the same configuration as the controls for the initial, single player bird

Both sprites move on the same screen and try to outlast each other by avoiding obstacles

If one of the birds death () method is called because of a collision, the other one proceeds as if it were in single player mode

**NOTE:** this may be difficult to code using the single player methods, so it might be necessary to recode certain parts of the controls into a seperate multiplayer part

**Details**

* No libraries or pre-written code necessary
* **Feasibility Analysis:** other than the various “NOTE” comments found above, most of these features seem relatively easy to code using Javascript
* This program is relatively simple and would probably require a week or two to complete

**Prioritization:**

1. Bird: Main character
2. Various obstacles (detailed above): main objective
3. Coins: Secondary objective
4. Score: Keeps track of points and distance
5. Multiplayer Mode: most difficult aspect to code

**Major Variables**

**Coins** (local variable)**:** An int variable, which keeps track of the coins picked up in the game

**Distance** (local variable)**:** An int variable, which increases every time the sprite moves forward to keep track of progress

**Speed** (global variable)**:** An int variable, which increases each time the sprite moves up a level and choses to continue; this variable factors into the Math.Random formula, which generates the obstacles, therefore increasing their appearances and making them approach faster

**Tentative Schedule**

Days 1: Generating main character

Day 2-3: Main character controls

Day 4: Drawing cars, trucks, and signs

Day 5: Drawing signs and tunnels

Day 6-7: Programming hunters

Days 8: Programming collisions/death coding

Days 9: Coins

Days 10-12: Work on making each level increasingly difficult by changing the speed int