Main Project

This project is a game in which the player has to shoot the bird on the screen to earn points. You are first introduced to welcome screen, after clicking on this screen you are taken into the game. The player has 60 seconds to score as many points as possible, every miss costs the player 1 point and every hit earns the player 3 points.

Specifically, this game has the following parts

* ‘BirdHuntGame.js’
  + This is where the main game code resides.
* ‘welcome.js’
  + This is the js file for the welcome/start page
* ‘gameover.js’
  + This is the js file for Gameover page
* ‘Index.html’
  + This is the main html for the game
* ‘index2.html’
  + This the page that is called by Gameover.js as well as welcome.js
* ‘index3.html’
  + This page is called by BirdHuntGame.js to display Gameover.js
* ‘game.css’
  + This is the css for the canvas
* ‘Images folder’
  + This has all the images needed for the game
* Createjs library
  + This library is what I used for working with canvas. Specifically using Easejs

The main game algorithm in BirdHuntGame.js works as follows:

1. Declare global variables
2. Run loadpage function
   1. We first set up the canvas size
      1. Retrieve the canvas set up the size
      2. This is also where we bring in createjs library
      3. Stage gets the canvas object and creates a stage object, This gives us the advantage of being able to use the methods available for stage in createjs.
   2. Load page
      1. Set all the item that are going to be added given an Id which is paired with a source
      2. Load everything listed
   3. Update the timer
3. Load Queue
   1. Load the background image by creating a createjs.bitmap object and refreceing it to the background image
   2. Display background image
   3. Display Score by creating another object and add it to the stage
   4. Display Time by creating another object and add it to the stage
4. Create Sprite sheets
   1. Create bird sprite sheet from settings taken from createjs along with the animation
   2. Create death sprite sheet and animation
5. Create another Enemy once we hit the bird
   1. Create another bird sprite sheet from the name it was given
6. Create crosshair
7. Set how fast it will be animated
   1. Set it to 15 FPS and add that to the stage
8. Set callback functions for the user to control the game
9. Run death function for the animation after the bird is shot
10. Make sure the bird is inside the game
    1. make sure it is less then the width and greater than 0 on the x axis
    2. if it is off the screen lower the speed
    3. do the same thing on the y axis
    4. set the position using the x and y parameter
11. Handle the mouse when down
    1. Each time the player shoots increase the speed
    2. Figure out where they shot
    3. Use collision testing to see if they shot the bird
       1. Create a hit flag. If the shot was 20 less then the position and 20 more than the poison then they have hit
       2. If the player hits the bat then remove the bat with the death animation and create a new bird
       3. Increase score and make it harder by increasing the speed
       4. If they missed deduct points and update score
12. Finally, Update time
    1. Check to see if it has been more than 60 seconds

I implemented two of main features I thought were needed: a way to have score and time be converted to string so it can be displayed properly and a way to let the player click anywhere before the game automatically starts. A third feature I added is a check for the timer to run out to display game over page. One improvement I did not have time to implement was a way to display score and buttons on the game over page.

The main module for the project is the JavaScript (.js) file ‘BirdHuntGame.js’. It houses two 19 global variables and 8 functions. Because the 19 variables are referenced throughout the module, they are global variables. The functions *createBird()*, *deathfly*(), and tickevent are helper functions; they are called only after the augmented necessary arguments are created.

The code in this project is fairly extensible. For instance, since *load*() operates dynamically on the values queue and custom builds, it is easily scalable. And we can easily add more birds by adding more sprite sheets and functions similar to *createBird*(). In this project, the helper functions are deathfly(),createbird(),and *tickEvent*(), but other functions like *instructions*(…), *createmorebirds*(…), and *moreanimation*(…) could all be added later to extend its functionality.

One “external constructor” is *update*() function in *welcome*.js , because it takes the user to an external client webpage. One “internal constructor” is *updateTime() in BirdHutGame.js* , because it constructs an internal evaluation how long the player has been playing. These are the two “static” functions because they don’t change and are accessible by a client. The helper functions *tickEvent()* and *queueLoaded*(*)* are “dynamic” because they take parameters the client doesn’t have (event). These two helper functions can be reused in other modules if needed – they will work the same way with any matching inputs.