CSCE 315
README 1

The first README provides a textual overview and inventory of source code modules. Think of this as explaining what is going on as if a new member just joined your team, as well as to help the TA know what to look for where in your source code.

An explanation of your game mechanics and how your prototype, including interface and backend, are designed to support meaningful play.

EaselJS-release_v0.7.1
This directory contains all of the files that we downloaded from easeljs and make up the library framework that we are implementing in our game.

Images
This directory contains all of the images that we are using in our game from players, to objects, and backgrounds.

Index.html
This file contains script commands that link to the easeljs files , the rip.js file, and our css file. The file itself contains some styling to help organize our webpage and label is appropriately. This file also creates a canvas for our own implementation, some text for credits, and a video!

RIP.js
This is the file that contains the source code for our game. This file implements the libraries from easeljs.

In the very beginning we initialize our variables that will be used later.
Up next we create some handlers for key presses and begin creating functions.

- Init(): This function grabs the canvas, creates a stage, sets the backgrounds, and creates some event handlers for the mouse click. This function also creates a player.
- Tick(): This function is used when an event takes place to update the stage. This essentially provides feedback to the player so they know that their actions are being received.
- Player(): This function is used to set a player. When new Player is called, a new player is initialized on the canvas. This function grabs the appropriate image for the player as well.
- Bullet(): this function is used to load a bullet. The image is grabbed and the initialization of the points is also determined based on the player's position.
- Powerup(): creates a initializes a powerup to be drawn on the canvas at a pre-determined location. For now we are just implementing an ability that increases shooting speed and spawns in the middle of the map.
- Movep;ayer(): this function will move the image of the player on the canvas in alignment with keyboard commands. The function also limits where the player can move by setting the players location back to the max width/height

of the canvas once the value of the players location becomes larger than that of the canvas.

- Movebullets(): implements an array to move the image of the bullet across the screen. This will also implement a limitation on how far the bullet can travel. Ideally the bullet will be “deleted” once it goes out of the bounds of the canvas. This function also does some work by determining the position of the bullet with some math and the current bullet speed value.
- Rotateplayer(): this function will rotate the image of the player to follow the mouse. The current implementation will make the player track (look at) the mouse no matter where it is on the canvas and even if you stop moving it.
- Determinepowerup(): This function creates a powerup, decrements how long the powerup lasts, and removes the powerup after the time limit. Right now it just resets the firing rate back to normal and turns the player image back to normal.
- Checkpowerupcollision(): this function set the threshold required for the player to be within to pick up the powerup. If the player has collided with the powerup, the original player image is removed and the new “powered up” image is loaded as the player. The powerup image is then removed, the countdown of the player powerup is triggered, and the firerate of the player is increased.
- Mouseclick() & mouseunclick(): these functions are event handlers that will return booleans based on whether or not the mouse is clicked. Should it be clicked on the canvas, the player will shoot. Once the event of the mouse not being clicked is registered, the player stops shooting because the boolean is set to false.
- Handlekeydown() & handlekeyup(): are similar to the mouse functions above in that they handle the event of a key being pressed down. Should the appropriate key be pressed (WASD) the value of the corresponding variable will be set to true. This triggers player movement. Once the key is released, the value is set to false and the player stops moving.

<u>**Videos**</u>

This file will contain the video files that we implement for our game. Currently there is only the “song of the day” videos being uploaded.