Billy Wei

100252633

December 1, 2017

**Term Project Report**

The term project that was made for this semester using the knowledge that we have learned thus far on programming with the JavaScript language was a space defender type game. The objective of the game for the player/user is to survive the onslaught of incoming yellow aliens while obtaining as high of a score as possible by destroying the yellow aliens. The destruction of each yellow alien will garner the player 20 score points. As the game progresses, the difficulty increases as more and more yellow aliens spawn at a faster and faster rate. The player is given 3000 health points at the start of the game. The player will lose 500 health points each time they make contact with a yellow alien. Once the player health points reaches zero, the player will be destroyed and their score will be displayed in the form of an alert message. However, to increase the player’s survivability, 70 health points are awarded to the player for the death/destruction of each yellow alien.

Steps were taken to first initialize the JavaScript programming of the game. The very first step taken for the making of the game was the creation of a canvas for a playing field of the game. The second step that was taken was the creation of a constructor for the game components (function GameComponent(x, y) at the beginning of the JavaScript codes) to define the x and y coordinates of any game component as well as vx and vy to define the speed of any game component in the x and y direction. Also included in the constructor was status that was set to true. The status element was made so that any game component can be set to a Boolean value of either true or false whenever needed further in the programming.

The player object was then created as a new game component (JavaScript code, line 23) that was initialized to the centre of the canvas. Additionally, the player object was then assigned an attribute of its movement speed to define a value for the speed that the player object will be moving at. The player object is also assigned a value of 30 for the number of pixels that will be used for its radius. This was done so that collision checks could be made easier later in the coding.

Since many of the same enemy will be used for the enemy component of the game, any array was initialized to hold each enemy object (code line 89). A for loop was then created to set the number of enemy objects to use, with the initialization of each enemy object that is stored in the array as a game component (coding lines 91 and 92). The status of each enemy object within array is initially set to false and the angle of each enemy object is initiated to 0. Furthermore, a function was also made for both the enemy’s speed and the enemy’s movement (coding lines 97 and 105). A final function was made regarding the each enemy’s collision with the player (coding line 115). Each enemy object is drawn within an update function that is executed every 10 milliseconds by a set interval.

An array was also initialized for the bullet component of the game (coding line 79). A variable was then set to define the speed that each bullet was to travel at. A for loop was then created to dictate how many bullet objects are to be stored within the bullet array (coding line 82). Within the created for loop, each bullet within the array were set as a game component that was to be drawn at the location of (0, 0) relative to the canvas (coding line 83). The default status of each bullet was then set to false. A function (coding line 129) was created to check for the status of each bullet object within the bullet array using a for-loop. When the bullet status becomes true after the space key is down (event listener at coding lines 213 – 228), a bullet will be drawn within an updating function that is executed every 10 milliseconds as dictated by and interval that was set (coding lines 254 and 452). Another function (coding line 146) was made to check for the collision between any bullet object and an enemy object. When collisions do exist, the statuses of both the enemy objects and the bullet objects are set to false.

The final function created was to kill the player’s object when its health points are equal to or lower than zero (coding lines 172 and 399). When the player’s health is at 0 or lower health points, the game will alert the player of their score and the end status of the game that they are playing. The game is then reloaded to the beginning where the player can either continue to play the game or not.