Maegen Borden

CS 3160

Final Project

Space Invaders

1. This project is a recreation of the classic game Space Invaders using javascript and html. The game generates a ship object which rapidly shoots bullet objects at incoming enemies when the user hits the space bar. The user can avoid enemy attack by using the arrow keys to move the ship up, down, left, and right of the screen. Users go through up to 25 levels, trying to defeat the alien enemies. In order to play the game, user needs to simply download the zip file and launch the index file in their favorite browser. The most important test was a collision test. I had to test for the ship being hit by bullets and then taking appropriate action (game over 🡪 reset) and vice versa for the enemy. Also, score needed to increment. One important test case was making sure there were no dark areas like the edge of the screen that would prevent the ship from not ever being hit.
2. The project works and uses simple javascript and html to generate a canvas which holds multiple canvases for our game object and then renders it to the screen. You can test the project by launching it in the browser and playing the game. Developed components include: ship object, bullet, enemy, enemy bullets, and bosses. Also, a score is recorded. If the ship is hit by an enemy bullet, then the game is over and restarts. If the ship’s bullet hits an enemy, then the enemy disappears. The goal is to get through the levels and defeat increasingly, aggressive enemies.
3. The project may certainly be split into smaller modules in the future. With further implementation and work on the ship, enemy, ect.. these components could be split into separate javascript files. In that way, the code is separated from one very long file and it is easier to access code in the future for modifications without breaking other pieces. Key codes were used to get user input and an API for optimizing the animation loop.
4. All the datatypes and objects are dynamically bound. They can be accessed from other public functions and classes. Classes can be added to functions. In the future, the enemies should be able to break off from each other, rather than be in single rows. This would require adding different move patterns. It would make the game more interesting and harder, which is where levels come into play. Multiple boss objects would be added for fun. All the new implementations can be made in separate javascript files which will ensure existing code remains good if something breaks. I’d also like to add more weapons and power-ups and explosion effects in the future. Originally, I wanted to customize the game to a Harry Potter theme, but the pixel art (images) didn’t render correctly. That would have made the game fun and appealing.
5. A tool called a dirty rectangle was used in order to split the multiple canvases. By doing that, the dirty rectangle on the bullet canvas, clears only immediate data around the bullet and redraws it. Since we have many objects to clear and regenerate, it makes sense to use multiple canvases because it would be too expensive to clear the whole canvas at the rate it needs to be. Also, garbage collection was a factor for the bullet objects. Since objects were being created and deleted so quickly, it was necessary to implement the pool so the game isn’t slowed down from trying to free up memory. When creating more boss classes, you can extend the enemy class and copy the shoot method using multiple inheritance.