Brian Williams

Professor Anthony Ellertson

GIMM 110

1 November 2017

Individual Game Rhetorical Analysis

I started my process by familiarizing myself with the game templates and how they worked by adding comments and changing variables. Next, I began brainstorming themes for my game. I thought of my alien character from the sunrise assignment and it spurred the idea for an alien invasion theme. After I had this idea, I dove straight into designing the graphics. I used tools in Animate to make all of the objects I would need in the game. This included a background, ground turrets, helicopters, an alien ship, fuel, bullets, health/fuel bars, and a giant robot boss.

I knew the Survive template would definitely be the best template for my idea because it had player movement with stationary turrets which is exactly what was needed in my game. I began getting myself very familiar with the survive code by studying each part and adding useful comments. After this I dove straight into editing the game. I started with the camera. The camera follows the player around the stage by assigning the background with the inverse x value of the player. I wanted to keep the player on the background so I added simple if statements to the camera methods that cause the background to not move if the player is past specified x values. Next, I moved to the player class and began adjusting the default values, removing rotation and adding gravity to make the player movement feel the way I wanted it to. I included gravity by adding a set value to the y of the player if the player direction isn’t up or down. I didn’t change a lot in the turret class besides modifying several of the physics values for the shot. I added a health value that would apply to each turret. I also included a destroy method similar to the one used in the shoot game.

Next, I moved to the main game document where I did a majority of my modifications. First, I added the turret health subtraction inside the hit test loop for player bullets on the turrets, as well as an if statement calling the turret destroy method if the turret health is below zero.

Next, I needed to modify the makeTurrets method so they weren't floating and spinning all over the sky. I changed the turret.y to a static value at the location of the road I made on the background. I also added a new array called turretBases that adds a new turret base at the same location of each turret. The idea was to have a stationary base that the turret is mounted on. I couldn’t simply add the base to the turret object because they would both rotate together.

The next problem I worked on was the player’s health and the health bar. After adding the playerHealth variable, I added the subtraction inside the hit test for the turret bullets on the player. Next I created the health bar with an object called hpColor that I would use as the red bar that changes with the player health. To accomplish this, I added a method called updateHealthBar that I can call every time I need the playerHealth to decrease. This method uses basic division to find the percent of hp left and uses scaleX to shrink the hpColor accordingly.

After testing the playerHealth, it gave me the idea of including a fuel system to the player’s ship that could work very similarly. I started by creating the fuel bar objects and the updateFuelBar method. I decided to only subtract and update the fuel bar when a player goes up(presses W). I added this in the keyDownHandler with if statements that cause the player lose vertical control if the fuel falls below zero.

After creating the fuel depletion system, I knew I would need a good system for player’s to acquire more fuel and keep fighting. Helicopters were the perfect fit for the theme and the purpose so I began designing the chopper and fuel tanks. I decided it would be best to create new class for the choppers since they would have unique physics and respawns. In this method I included an update method that subtracts the chopperspeed from the x value so it would fly across the stage from right to left. Next, I added a fall method that adds gravity to the y value of the chopper. This triggers when the choppershealth reaches zero. Next, I moved back to the main document and added a hit test for player bullets on choppers so they can be eliminated. I used if statements to trigger the fall method and a makeFuel method when the choppers health is below zero. Next I made the makeFuel method which adds the fuel tank object at the choppers location. It also uses greensock to make the fuel slowly drop straight down like a parachute. In order for the player to collect the fuel I went with a distanced based collision method called getDistance that applies the distance formula with object locations and returns the distance between them. In the update method, I included an if statement that removes the fuel and increases/updates the playerFuel value if the distance between the player and fuel is less than 75.

I also used the getdistance method in a similar way to cause the chopper to destroy if it collides with player while it is flying. This results in large damage to the playerHealth with the bonus of instantly acquiring the fuel.

At this point I felt like my game needed a boss that would come out after all of the turrets are destroyed. I felt like a boss would fit the story as Earth's last resort to fight off the alien. I designed the boss as a large robot-like tank with laser eyes. I added a method called makeBoss that gets called when turretDeaths=10. I wanted the boss to shoot out of his eyes so I would need two separate turrets with different shooting physics from the the standard turrets. I accomplished this by creating a Laser class very similar to the turret class with a different bullet object and more challenging shooting physics. I went back and added two lasers inside the makeBoss method as well as greensock animations so the boss would slowly roll into the center of the stage. Lastly, I set up the hit test for the boss so the player can destroy him for a big chunk of points.

Next, I created an explosion animation with classic tweens that I could add to the chopper, boss, turrets, and player objects. I used gotoAndPlay to play the explosion when they are destroyed.

Next, I needed to add the scoring system. I created a text field and added it to the stage with the health and fuel bars. The score variable increases when the player destroys enemies and updates the field.text every time.

The last thing I decided to include in my game was a system for restarting the game without needing to reboot the swf. I first created a death screen that includes a restart button. I added a restart game method that get’s called when a player dies. This method removes everything from the background and displays the death screen. When a player clicks the restart button, all of the variables are reset and objects are re-added to the background in the same manner as in makeLevelOne. I first tried to call the makeLevelOne method but ran into issues with it adding the background on top of the touch layer.

I fully understand all aspects of my game and how important each part is working together. I’m really happy with how it turned out and I love seeing the reactions of friends and family trying it out.