Final Project Report

Zach Schrock

**Abstract**

The following report will detail what I have accomplished for my final project in Interactive Computer Graphics. I will talk about describe the game I have created that is similar to a 2D platforming game that is similar to that of an original Super Nintendo game, much like Super Mario World or Mega Man. I will now go on to explain the what, why, and the when of the video game created.

**Introduction**

The game, while not as in depth as them, was inspired by two of my favorite video games of all time, Super Mario World and Mega Man. The basic concept of the game is to defeat the boss so you can win the game, either avoiding or defeating other enemies while doing so. To defeat these enemies and the boss, the player must attack with one of three weapons, fire, ice, or electric. Each weapon is specifically effective against a different enemy, as the player will need to recognize the movements and attacks of each enemy to determine which weapon to use.

If the player loses all of their hit points, the game is over and will display a lose screen. If the player defeats the boss, however, the game results in a victory and will display a win screen. Whether either happens, the window will close after execution.

**Description**

The creation of the world is the first step in rendering the environment. This involved creating the blue background, which simply was a glClearColor function, and rendering a ‘floor’ for the player and enemies to land on when the y-positions of all of the were reaching too low. A clouds function was created using GL\_Polygons, and sent in a for loop to determine exactly how many clouds there are. Then the player and enemy models are created in the foreground of all of these elements. These characters were only created as rectangular shapes to make collision detection a bit easier (more on that later).

The character has the ability to move forward and backward, attack, and jump. These abilities are defined by the glut keyboard and special functions. The jumping mechanic increases the player’s y-axis until a certain point. Once that point is reached, the player’s y-axis will drop at that same rate. Moving forward will increment player’s x-axis, backward will decrement player’s x-axis. The attack function will execute one of the three attack functions, dependent upon what the player’s currentWeapon enumeration variable is.

The switching of weapons is also handled by the keyboard function. Pressing on the key of either one of the weapons will both change the currentWeapon value and the RGB color of the player. The three different weapons are unique in their abilities. Fire attack has a straight line ‘sword’ that ejects from the player, damaging enemies if landed. Ice weapon is similar to fire, but will instead lay on the ground in the same spot for the duration of the attack. These two attacks both make use of the player’s x-position, but only the fire attack needed the y-position, as it will follow the player even if they jump. The electric attack is a projectile based attack, and its position incremented after every glut timer function call to increasing x and y-positions.

Enemies have unique movement patterns and attacks. The bird enemy (white square) is much higher than the player and shoots projectiles at them, moving in the x and y directions simultaneously. The electric attack is the most effective attack against it. The lizard enemy (green rectangle) attacks much like the player’s fire attack, but its length of attack is longer than the player’s. It also moves back and forth in the x direction, grounded on the floor object. The ice attack is more effective than any against it. Then the robot enemy (gray rectangle) is a larger target than the other two, but its projectile is more of a straight-line approach than the bird’s, making it a tougher enemy to avoid its attacks. It moves up and down in the y-direction, but not as high as the bird enemy does. The fire attack is seen to be the most effective for this enemy.

The boss (black enemy model) is a larger opponent, that will shoot two projectiles at two different rates. The boss also will move back and forth in both the x and y-directions, making it a mobile target. The electric weapon is the most effective against the boss, but the fire weapon is also a good choice for close-range.

Both the player and the boss contain integer variables that keep track of hit points. The player will have three hit points, while the boss will have five hit points. Any time a player’s weapon creates a hit detection with the boss, the boss’s hit points will decrement by 1. Any time any enemy’s weapon creates a hit detection with the player, the player will lose a hit point. If the player loses all three of their hit points, a gameOver function will run, displaying the losing message, and then close the window. If the boss loses all of its hit points first, though, a win function will run, displaying the winning message, and also close the window.

Collision detection is implemented using the AABB technique. This collision detection is only applicable when using rectangular shapes that are not rotated amongst the x or y-axes. When a weapon’s ending position is larger than the target’s beginning position, and the target’s end position is larger than the weapon’s beginning position (note: this is for both axes) than collision is detected.

**Testing**

Testing is a bit difficult specifically for the game, as I had to manually test the game throughout every implementation of a feature to make sure it was working the way I envisioned. When something wasn’t working properly, I had a pretty decent of idea of where the issue was, but not specifically what the issue was. Hit detection was something I had to constantly check for, as there are many conditions and positions I had to make sure were exact, for each of the enemies, player and boss.

The only bug that I can think of as of now is that the player’s head (extra rectangle above the larger body) and the boss’s head and tail do not detect collision. This is something I am just now realizing, about 30 minutes from the due date.

**Future Implementations**

There are a few things that I would like to change in the future. Firstly, I would change the character and enemy model to be something other than the rectangular shapes they currently possess. I had these models from the beginning and fully intended to change them, but the collision detection was simple enough to create with rectangular shapes, that I did not think that I would have enough time to recreate all the models and discover a new collision detection algorithm. The next thing I would implement are shaders. I spent much more time working on the functionality of the game that I overlooked shaders and went with plain RGB colors. Lastly, I would create more enemies and a more complex world. I wanted to reach the requirements I set out in my original proposal, so I wanted to at least get the three different enemies and boss in. However, given more time I would put in multiple instances of these enemies and make the boss a little more difficult.

**Conclusions**

In all honesty, I really enjoyed this project. I did not get to implement everything that I wanted to implement, but even after this class is over, I will continue to work on this game and improve it as it was fun to work on something like this. One thing I learned is that collision detection is much harder than I ever thought it was to implement even in a basic rectangle shape game like mine. This project made me appreciate the game engineers and developers of AAA games infinitely more for what they do to create great products for us to enjoy when they come out. The only things I would do differently if I ever got a chance to do this project again is maybe try and create it in a 2.5D or 3D modeling system, to make the characters look at least a little better than they do now.

**References**

https://learnopengl.com

https://www.khronos.org/opengl/