Amara Tariq
GIMM 110
Final Rhetorical Analysis

The first part of the game that the viewer sees is my 30 second animation. This introduction should be a movie clip embedded into the game in order to allow for audio to function correctly. In order to export as a movie, we had to create an actionscript file with the 30 second intro and then add audio and export the video from there. I had a lot of issues with my audio lining up correctly, but I was finally able to figure it out. By the time I figured it out, I realized that a lot of my animations wouldn't export correctly due to all of my animations being nested. I realized this way too late and unfortunately had to make the tough decision to not include audio for a majority of my introduction animation. This is a major part that I wish I could've fixed because audio does play a huge role in the gameplay experience. If I had more time, I would go back and alter this to correctly function with audio. The audio that was included was all the parts that had lip sync.

In this introduction animation, I have nested animations of subtle idle animations, such as the wind blowing the grass blades, kids playing on playground equipment, etc. I think these were good choices, as they added more details to the animation and made the scene really come together. The colours I chose for the background characters helped create more interest in the scene. In this scene, I have my main character, in white, come walking out the bench. Included in this animation is his walk animation. This is a bit different than the bone structure that we learned in class, but it was hard to follow that method of creating the walk animation due to the fact that my character doesn't have any joints, such as knees.

The next scene is him on a bench rustling around and pulling a sandwich out to munch on. I really liked this scene, and I wished the audio for this would've worked out because it really did add to the scene a lot. This scene added a comedic value and also set up the tone for the game. The drawings I imported did end up becoming a bit pixelated in the process of importing to Animate, which is something I would change if I could have more time to make more revisions.

While the main character is eating his sandwich meteors start to fall. The view is at first showing the entire park, and then we zoom in to see one fall right next to the character, leaving behind a robot after the dust settles. I chose to add a dust cloud to help make the transition between the meteor and the robot feel smoother and create the sense of an impact. This next scene where the main character pushes up his "glasses" shows more of the lighthearted, comedic tone and style that I wanted to incorporate in the game. It also shows the transition and the reasoning as to why the main character decided to explore the robot. A walk animation shows him climbing the robot, where he then glances around and sees a note with keys to unlock the door that leads into the robot. Again this plays into the tone that I wanted to keep in the game. We also see the sandwich keychain on the keys that lead into the robot. I wanted to show a sense of continuity of events that were occurring in the animation and game. This continuity, as well as others sprinkled throughout the game, give a hint to what happens in the end of the game, during the victory animation.

Once inside, a movie clip plays for the lipsync portion. I had a few difficulties for this particular video, as it won't autoplay. I have two other videos with audio that autoplay fine, but

this one for some reason won't, even though the settings are set for it to autoplay. Due to this bug, I doubled the time for how long the animation would stay on this frame with the video. This gives time for people to click and play the video animation. This clip shows the owner of the robot telling the main character what the robot is and why it came and crashed on the planet. It sets up the first level of the game.

The concept of this first level is for the main character to reboot the robot to help save the planet from enemies. The video includes the visuals of what the "control room" for the robot looks like, with the owner's hologram showing up to explain to the character what is going on.

The first four interactions are a version of the drag and drop example we were given in class. I had to make some alterations to the code as the objects I am dragging and dropping do not match the end point in terms of shape and location. The objective is to drag the wires from the left to right, whilst matching their colours.

Looking to the code, I created the wires and their ends in the same way as the drag and drop example program showed. The slight differences start to occur in the root.start function, where I have to alter the scope of the positions, as my wires within their movie clips weren't aligned to be equivalent to the global scope, hence the switch by using localToGlobal. The rest of this function is simply creating the wires and wire ends as children to the stage. This level uses essentially the same restartHandler, alteration just made for what count that is being reset. In this case the count is for the completed wires, which controls when the player continues on to the next set of tasks. For the mouseDownHandler function, I removed (by commenting out) a lot of the original code that sets the targets and moves the targets to the front of the stage. I didn't choose to do this because I had a frame designed to go over the scene to hide the length of the wires. If the wires were moved to the top, it wouldn't look appealing. I also commented out where the target's x and y were reset to move with the mouse. This was taken out due to the localToGlobal differences. In order to accommodate for that, I grab the global coordinates of the mouse position in line 68. I also store which wire the user is interacting with in line 70. These will be used later to move the wires. I then set up the mouse move and mouse up handlers.

In the mouse move handler, I check to make sure that there is a target wire, and if there is, we create a variable for the mouse offset, which is the difference between the local and global positions. This offset is then added to the current location of the wire in a global scope. Line 85 then takes the information we gathered, and switches it to a local coordinate so that we can then set the object to that, which will line up correctly with where the mouse is on the stage.

The mouse up handler is next, which simply checks if the mouse is off of the stage, therefore not pressing and interacting with anything. This then checks if there is a target, and if so runs a check function on it. This check function determines if the point where the mouse was unclicked is the correct position for that specific wire. I have a large portion of code commented out, which were my attempts prior, which shows the process I went through to get to the logic I ended up using. I dave the point where the target was dropped, and then switch that to a local coordinate, so that I can check if the wire end that is correct was under it. The variable spot is a boolean, which checks if it was the right drop point, and if it is false, we call a onMiss function that resets the location of the object to where it originally was. If true, we compare the name of the target wire (the one picked up and dropped) to the name of the wire end. If they match, then the function onMatch is called which clicks the wire into its position. OnMatch also increases the count of the correct wires, which is then checked and if it equals the length of the array of the

children, then we call the onWin function which continues the user to the next task. Throughout this I also update a taskbar, showing the progress.

The next task is turning on four parts of the robot. This is a simple code that I created on my own. This replicates code that I used for the sunrise project. Each part is a button, that when pressed, goes and stops on a frame within the clip that is green, indicating that it is on. Again I also update a taskbar throughout the task to show the progress. In the update tracker function, I also add four to the count of the amount of shields turned on to accommodate for the last 4 wires that were completed. In between this last task and the final task I include a short animation that shows the character thinking if this is a good idea to follow through with.

This final task is a simple button. The first click raises the case to the “official” start button to the robot. This is essentially the same code as the previous task. I have another funny animation inbetween showing the robot turn on and cook a bird, which is another hint to the end animation. There is then another lip sync transitioning together the first half of the level which turns on the robot and where the player actually fights the enemies, which are aliens.

This level follows the setup of the zombie example game we were given. At first I was coding it from scratch, but with tutors’ suggestions I switched it to the class setup of the zombie game. The game class isn’t too different, the only part that really changed was where I randomised two values for the enemy class. One is the enemyTypeNum, which is on line 30. This generates a number 0-5 which determines what frame of the enemy movieclip we stop on, which changes the colour of the enemy. Line 31, enemyHitNum, generates a random number 0-3 which is the number of hits that are needed to kill that enemy. Due to these additions of properties for the enemy class, I had to add additional inputs for the creation of enemies on line 32. The last addition to the game class was adding a skip class, which creates a skip button for the user to be able to click and skip this level. The reason for this creation will be explained a bit later.

In the player class, I readjusted the sizing and location of the player, as well as added some movement that is different from the zombie game. I switched my player to move on the Y axis instead of x, as well as leaving the wasd functions that the zombie game had to allow for additional movement. For actions, I have two, one for a single mouse click, another for a double click. A single mouse click causes the player to throw hands, which punches enemies. Double clicking fires a laser, which is then stored into an array. This is so they can also be removed once going off screen, which is taken care of in the removeObject function.

In the enemy class, we see in the constructor the enemyType chosen and the frame stopped in lines 33 and 34. The other part added is the enemyHitNum. This changes the update function, as we need to check with every hit if it’s enough to kill the enemy. This portion is where I was unable to figure out what was going on. I was unable to get any sort of hit detection to be registered. If I had some time to sit with a tutor I might be able to see what is causing this. I built my movie clips the same way and structure as the zombie game, but it isn’t being registered the same for some reason. This is why I created the skip button, because I wasn’t able to get even one hit detection registered. I wanted the player to still be able to see and experience the victory animation and ending to the game, thus the creation of the skip class. This is created in the same way the player is. An event listener is added for the button to function on a mouse click, which then triggers the game to end.

The final victory animation had audio and is a nested video clip. This shows the hologram congratulating the player. A slow transition from the previous angel and devil characters shows that this entire game was just a dream. The main character is fast asleep in bed. We also see the bird, which was thought to have been cooked by the robot's lazers, is safe and alive in the characters room. This could've been figured out by the player, as the sandwich appears not only when the main character is eating, but also on the keys, when the keys were supposedly from a different planet.

Overall I think my animations flow well together, with minor problems of pixilation, which were mostly fixed by the end of the game. I think the only major issues are the audio and the second level.