Report of AI game

Background:

The game we created is based on an arcade based action game. The game objective is to defeat each enemy boss one by one. The game adopts a similar structure to megaman arcade game which also the player choose to fight enemy boss one by one. To win the game, the player must fight the enemy using the tools we provided to the main character. When an enemy boss health reaches zero, then that boss is defeated. The player must avoid to have his health to reach zero, or he is defeated and loses the game. Defeating all bosses meaning the game is won.

The background of the game is that there is some evil monster that invaded the city, the hero must go and defeat the bosses one by one or the city will fall. Each boss will have its own fighting style that the hero must adapt and fight against the stragegy the boss uses.

The game is developed in javascript and can be run using a normal web browser. It is set up in a GitHub project and adapt the web broswing feature that can be accessed anywhere.

Game logic:

The framework of our game is in four layers. The first layer is the introduction menu layer. It allows the player to start the game and proceed to the second layer. The second layer is the enemy selection screen, which allows the player to choose the enemy to fight. After choosing the enemy, the game proceed to the third layer. The third layer is multiple parellel instance that contains the game AI. The player will fight according to the game AI mechanism. The result will proceed to layer 4, the result reporting layer. The game will go back to layer 2 if not all the enemies are defeated.

For our file structure, we have set up a main html file that will run all included javascripts. We have seperated most functions into several javascripts. There is a javascript to handle the game menu. For the player’s character and different enemies, we all have a different javascript containing functions to allow the game to run. This allow us to modify individual encounter easier without altering other finished functions.

Our setup of the game is shown as the diagram below:

<--insert a game logic diagram-->

For the game to run, we used several logic for calculation of status in the game. There are multiple variables set up to measure the current status of the game. As the setting of the game is on Earth, there is gravity. There are two variables to handle the physics of how the player drop because of the gravity. We used drop speed and drop accelearation to handle the gravity. Another logic for the environment is the player character moving status. We used several variables to determine how the player is moving. We set the maximum speed that the player can go and also the normal walking speed of the character. We set up a boolean value to determine if the player is on platform or in the air. We also have variables to set the player jumping distance. At last, an action status variable is set to record the player’s action. They include idle, walk, attack mode A, attackm mode B, defense and jump. This variable can help us to find the correct image for animation.

The image in this game is created by our own. We have drawn several characters and their corresponding models. They included idle, walking and attacking model. We have used image sheet to handle the image to be displayed. We included all the models belonging to the same character in an image sheet. This can reduce the loading time and memory usage.

The image background of our game is using a simple jpg file. The correct one will be chosen to display according to the enemy. However, the standing platforms are not included in the background image. We handle this with another javascript that draws simple rectangles. In this way, we can have a clearer obstacle for us to create an object on and detect collision.