Homework 5: Scripting New Games

Qiuyu Chen

Part 1: Scripting

In order to implement the scripting that supports different abilities. I integrated the ScriptManager from the example to my project. I updated the runScript function and added an additional runScript function which supports different types of parameters. Before the game loop starts, the method and class are registered. To meet the requirement of changing the game object using script, I registered the change size function of the platform. Even class was also registered to raise and handle the event. The script was stored in a js file that is located in the same directory as the project. For each game loop, the script will be loaded again to update the changes of the script. In my design, the script is able to change the size of the left platform in the middle air. When the game is running, I am able to use the script function trans to modify the size of the platform. Changing the number in the setSize method of the platform will immediately change the size of the platform of the running game. The handle event and raise event function supports raising and handling new events when the game is running. For example, I can raise the death event when the character hasn't entered the death zone. Having this gives more flexibility to modify the game when it is running.

However, the way I raised the event is not very efficient and does not support some of the events which need much information. I used the string to create a new event to raise, which the string is the type of event. The raised event was stored and can be handled after by using the event handler. In order to support more functionalities, I can register more classes to pass and create the information. I created three different runScript methods, one for the change size function which needs a platform as parameter. One for the raise event function which needs a string as parameter and another for event handler.

Part2: A Second Game

To implement the second game, I created two new projects based on the original client and server project. Since Space Invaders is not a 2D platform game, I removed the platform class, modified the move components to get rid of gravity and jump, the character(gun turret) can move in four directions. MovingPlatform was reused to create space ships since spaceships move in a regular pattern which is similar in the first game. Besides, I created a fire object which represents the fire shot by the character(gun turret), and a shoot component which the character has. Each time a fire is shot, a fire object will be created and stored in a list. When updating in each game loop, the fire will move upward. When the game starts, the character will locate with the middle low of the window. It can move forward, backward, left and right. When space is pressed, a fire will be shot. If the fire collides with the spaceship, the spaceship will be destroyed. There are two rows of spaceships.

The second game reuse most of the old design, such as multithreaded loop architecture, client-server, game object model, scripting and event management system. However, some improvements can also be done in the future design. New types of events can be added for Space Invaders and the server side can be updated to send and collect the information.