Programming Technology

3rd Assignment

Group 1

Nguyen Viet Minh Hieu

KU8RLA

Dec 6th 2021

Task

Red Light Green Light Maze

Inspired by the famous TV show Squid Game, the game is a version of the game Red Light Green Light. At the start of the game, a timer will start ticking down and by the time it reaches 0, if player does not reach to the doll in the middle, player will be executed. Player will have to collect as much money as possible and get to the doll before the timer hits 0. During that time, guard will be spawned in the map and looking for player. If player being touched by the guard, player will be executed. Throughout the duration, there will be two state, red light and green light. Red light is when the doll face is visible, if the player move during that time, player will get executed. During this time, guard can move around to find the player. On green light, the doll will turn around, player can move and guard can not move during this time. If player gets executed, player will lose all the money and game is over.

Analysis

There are different type of tiles to create the map. Some tiles will be passable, some tiles will block the movement.

There are different type of entity, player and enemy, both share the same logic of walking on tiles.

Entity will have hitbox so it can determine the behavior when it interact with different type of obstacles or entities.

Objects can be pick up by player but not enemy. These will gain player certain attributes or determine the state of the game.

Plan

To describe the tiles, a base class is introduce. The base class ‘Tile’ will present the general properties, and the class ‘Tile Manager’ is used to handle initiate array of Tile from map file and handle fetching the assets resources. The type of Tile will be presented in form of an array of Tile.

To identify the tile, there will properties such as image and collisionable, which determined whether the tile is passable for the entity or not.

The Entity base class represent the player and enemy. In Entity, there will be some general attributes such as facing directions, hitbox, collisionable, x and y coordinate and speed value. Moreover, to animate the moving animation of the entity, there will be 2 variants of each directions of a player, and

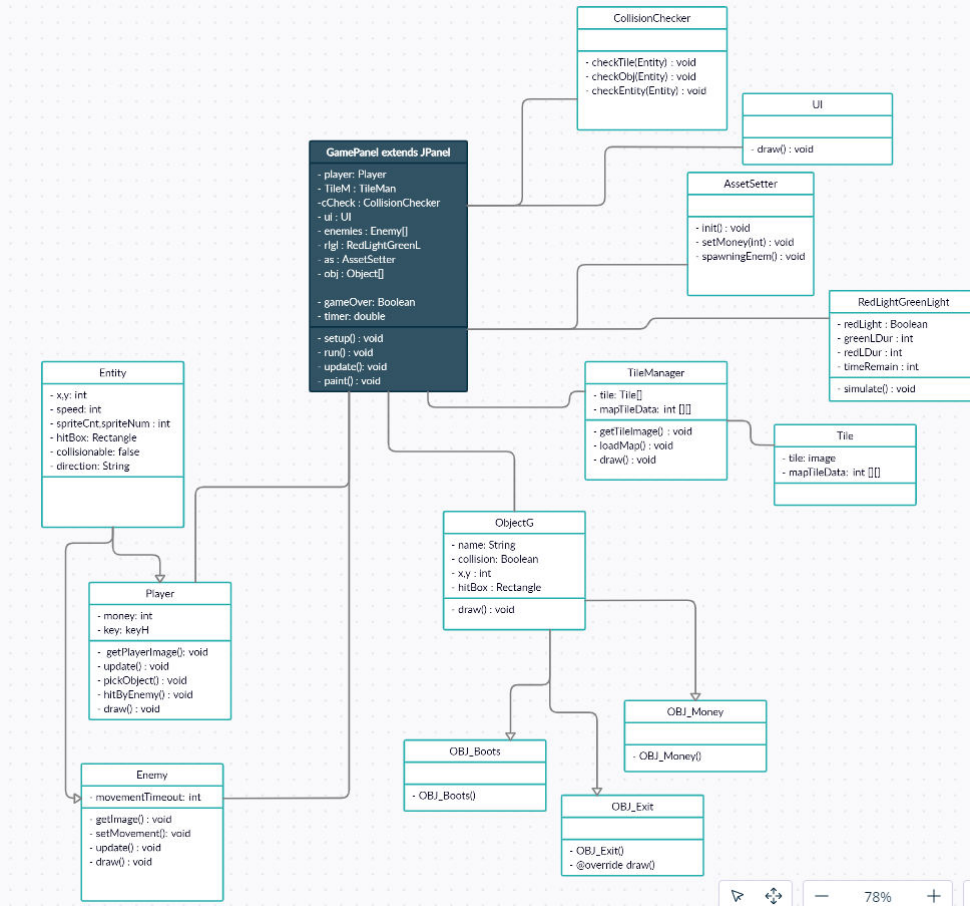
spriteNum value to determine which variant is being displayed. In Player class, it has a properties of money to store how much money the player has collected and KeyHandler for handling key pressed event.

There are multiple type of object in the map that serve different function. All object has image, x and y coordinate and a hitbox. They also have draw() function to render the object image on to the screen.

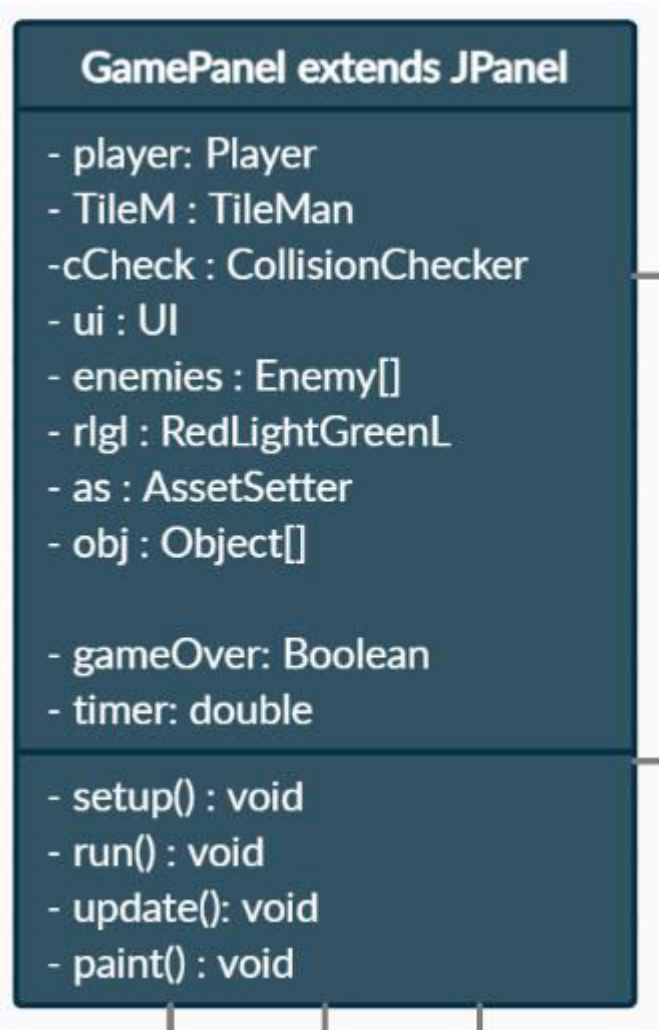
In term of game settings, there will be an AssetSetter for handling initiate object to an array, generate random position of money and spawning enemy rate. A CollisionChecker is used for checking the collision logic of the Entity with Tile, Entity with Entity and Entity with Object. An UI class is used to draw the data and state of the game on the screen. For game feature, to simulate the red light green light, a class is created with function to update the state of light. Lastly, the game is being handled with class GamePanel, which handle all the game functionalities, creating game loop, managing game state.

Classes Representation

Classes relationships can be illustrated through the use of UML diagram

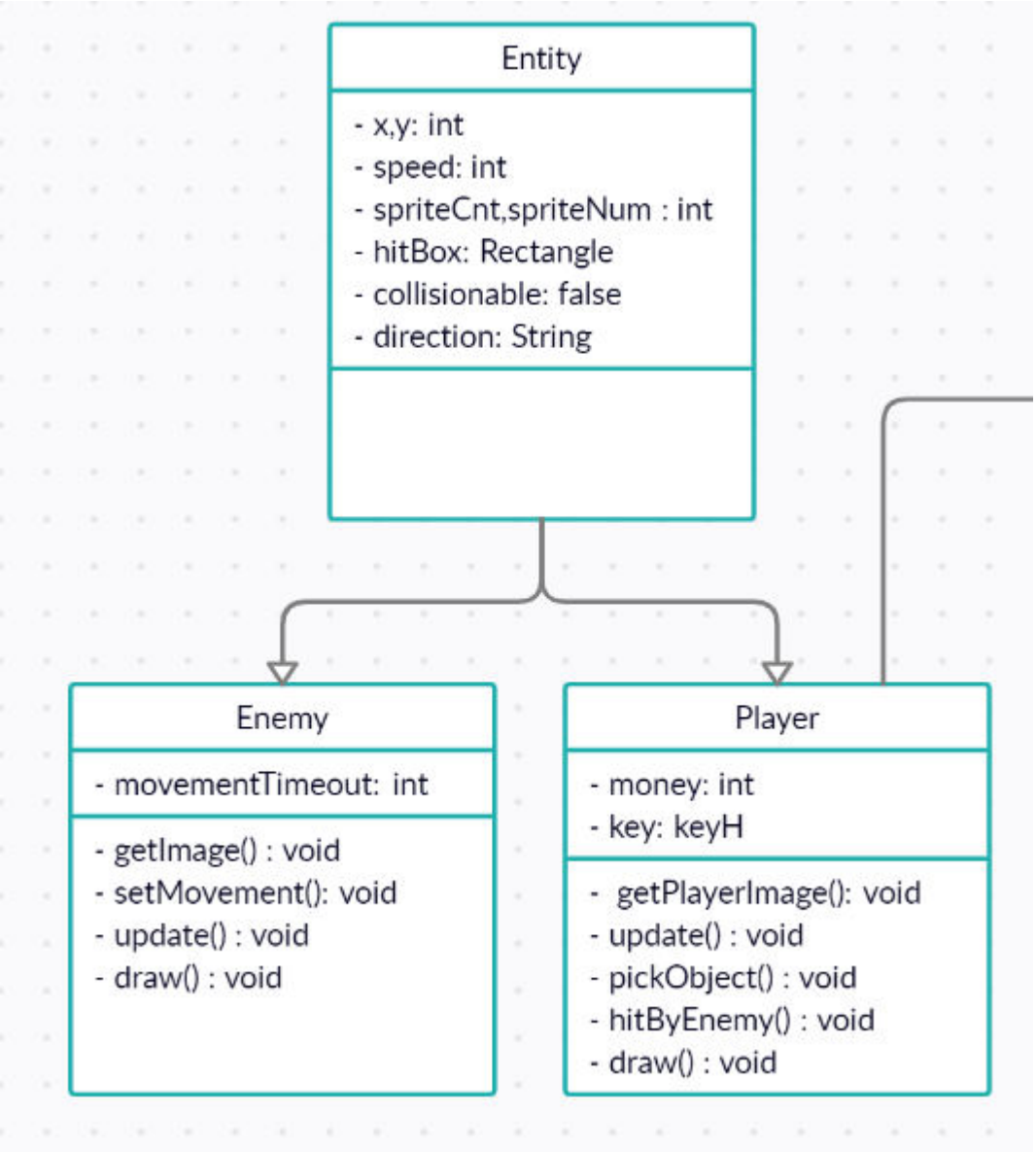


GamePanel



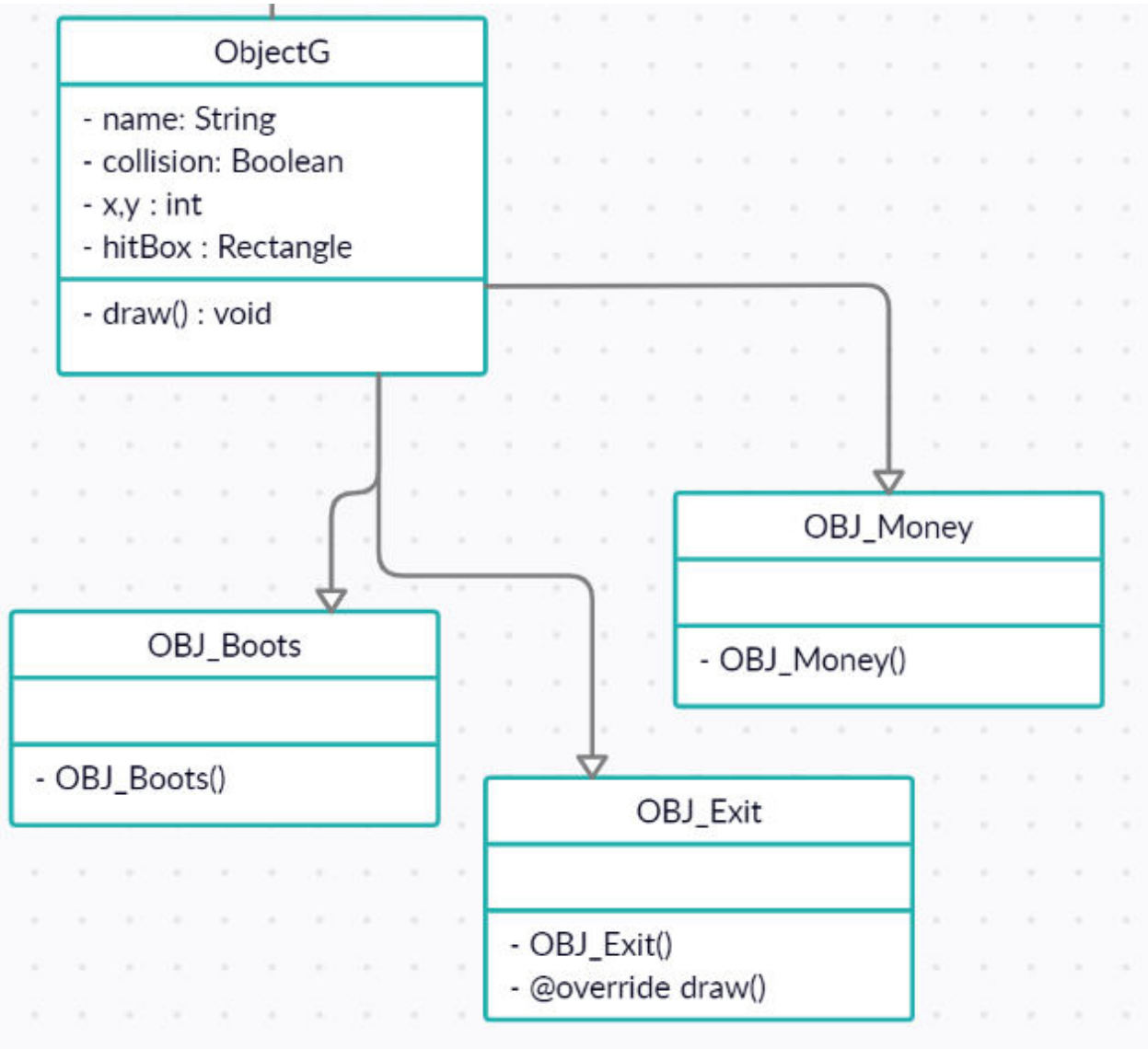
Game Panel responsible for setting up the game and connect every other part of the program. It's also responsible for creating the game loop that update every tick. Update() method is used for call every other class update() and paint() is used to call other class draw() method.

Enity



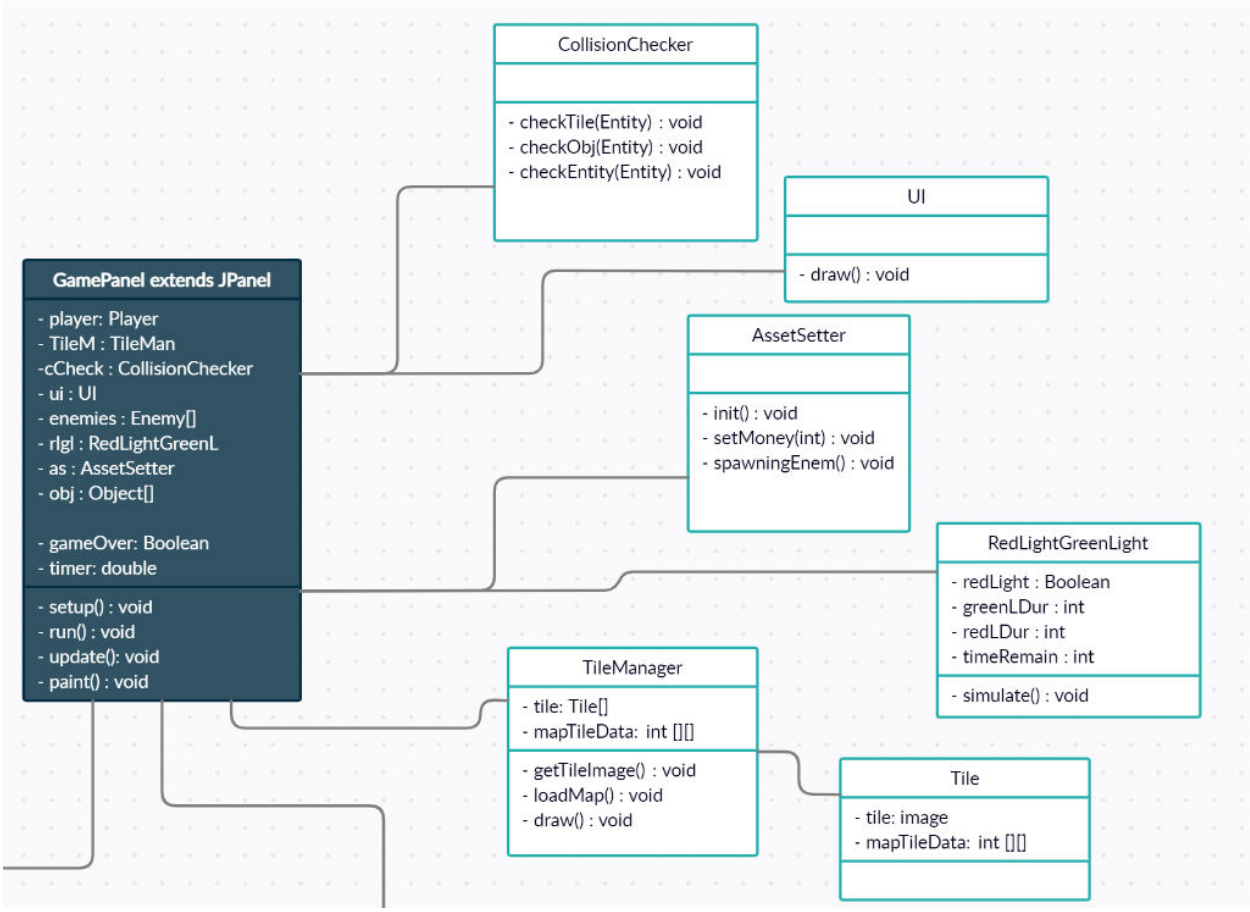
Enemy and Player class are inherited from the Entity class, which shares some common properties. However, since the interaction of Player and Enemy class are different, therefore, their method are not the same.

Object



In Object class, their inherited Object has the constructor to fetch the assets resource. All other properties hitbox, x,y are being initiated at AssetSetter class.

Other Classes



Other class are acted as the middleware class between the main GamePanel and other objects classes mention above. These class will take part in execute and update all the other object classes.

Descriptions

In GamePanel, as it implemented Runable, we override the run() and used it as a game loop. This game loop will update every nanosecond, and on each time, it will execute the update() and repaint() method. Repaint() is a method of JPanel as we inherited of GamePanel, and we are override it so it can call draw() function on other classes. The update() function is responsible for calling update() method for classes.

ColisionChecker is a class that hold method for collision checking. In the method CheckTile(), it uses an algorithm to predict the next move of the Entity in the current direction. Then use that coordinate to find the row and col of the two next blocks, then check the properties ‘collision’ of that block to determine if the entity can walk through that block or not. On another hand, the checkObject() method, used to check whether Entity has contacted with the Object or not, will get executed only when player hit the Object item. Therefore, we can simply check the collision by using the intersect() method of Rectangle to see if the entity hitbox collides with object hitbox.

Another method share the same idea is checkEntity method, which used to determine the collision between Entities, in the game, between Player and Enemy.

The AssetSetter class is used to initiate the assets for the classes. It will assign the position for Object on the map and their hitbox. Inside, there is also method setRandomMoney() to randomly put the coordinate of the money object on the map where it is non collision block.

The KeyHandler class is used to handle keyEvent by user. In the game, it handle the Key ‘W’, ‘S’, ‘A’ , ‘D’ as used for controlling movement. Inside the class, it’s override the keyPressed and keyReleased method.

The RedLightGreenLight class is used to simulate the game logic. It has the state of the redLight and fixed duration for redLightDuration. As the game go on, the time will be counting down and on simulate() method, it will randomize the greenlight duration on each interval after redlight, calculate the next time it switch back to redlight.

The UI class is used to represent the UI on the screen that shows the timer, current money and final end screen.