Positive opinion on the Engine Class

Jeng Yang Kong

29903823

I can't think of any good recommendation changes for the engine package. Instead I found this engine package is sorted out quite well for us.

Throughout this entire Assignment 2 and 3 working, I have found that there are many pre-functional code which I can reuse to avoid duplicate code with the help of **inheritance**. In Assignment 2, I implement the zombie part. In default, every actor calls the attack action class to perform an attack. As I only need to change the zombie attack, I can create a subclass called ZombieAttackAction which extends from attackAction class. So that I only need to override the method which I need to change and focus on modifying it, all other methods that remain the same functionality no need to duplicate again. Actors other than zombies such as players will still call attack action.

Zombie is dumb and can't pickup up item. I then create a pickupBehavior which is implemented from Behaviour interface from Engine Class. Interface helps break up the complex designs and clear the **dependencies** between objects.

For the Lost limbs part, I can add my created arm and leg into Zombie inventory with the method AddItemInventory which is very useful and defined in engine Item Class. When the limbs drop to the floor. Player pick it up can directly use it as a weapon because it is a type of weapon item.

Besides, I found that the tick method from Ground class makes me implement the corpse easier. As the tick method itself is called every game turn, I can keep track of the corpse turn and make the dead human rise from dead after 10 turns.

In assignment 3, I was assigned to the town map task. With the help of the run method in the engine World and the GameMap class method, I am able to run the new town map while processing all the actors on the compound map.

Speaking of the Mambo Marie, with the use of engine gameMap getXRange and getYRange method, I can get the width and height of the map which Mambo Marie can respawn on the edge of the compound map using some Math to find the random edge position.

One the last part which ends the game option, I extend the world class and create a GameWorld class. I modify the protected stillRunning method from World class and add “player loses or win” conditions on it according to the problem task. In addition, I found that getAllowableAction from engine Actor and Item class will return the newly created list which is unmodifiable so that calling method can’t modify the list itself .This helps avoid the **privacy leak** which we learnt in the lecture.

Overall, I found the code in Engine class itself helped me have more understanding about the object orientated programming concept like **inheritance** and **encapsulation**. I stuck quite long for this assignment at first fews weeks but now I am more understanding of how this zombie game was made behind and realized that making a game is not easy!