FIT2099 Sem2,2020 Assignment 3

Group Name: Shibalnu

Members: Liu Pei Lin, Gue Jia Xuan

Recommendations for extensions to the game engine

Problem 1

Ground only has allowableActions method that allows actions adjacent to its own location but not on its location. This reduces the flexibility of the Ground class usage. In some cases, Ground could have some interaction with the Player on its location, but it is not possible due to the engine. (like player should haverst grass when he stand on the grass not stand near by grass) The Ground should be treated as it can be interacted on its location and not be treated as it is can only interact with adjacent actor or item objects.

Proposed Design Change 1:
Ground should have a new method that returns an Action list when the player is right on the Ground itself. The action list returned should behave like allowableActions, just that the actions are on the Ground itself.

Advantages of Design Change 1:
This increases the flexibility and functionality of the game itself. The player can be on the Ground location and interact with it.

Problem 2:

World class instance variables are protected. Although it is not public, protected instance variables still can be modified by other classes in the same package.

Proposed Design Change 2:

Changing all the protected instance variables into private. If there is a need to get the instance variable, use getter methods. If there is a need to set the instance variable, use setters. For instance variables that are a list, an add method and remove method can be implemented.

Advantages of Design Change 2:

It provides encapsulation towards the data inside and we could implement a check to check if the set or added value is valid or not. We can control what is being added or set or removed instead of letting it be protected.

Problem 3:

Each tile in the world map, while running, processes it by tile. So tiles that are processed earlier will alter the actions performed by later tiles.
One example of such problem is that a male and female stegosaur might be apart, but the male stegosaur moves towards the female stegosaur when the female stegosaur has not made a move yet. This may result in the female stegosaur mating even though the male stegosaur already made a move(wandering behaviour). Our group did not have time to fix the bug because we did not know how to do so.

Proposed Design Change 3:
All tiles should make use of existing exits that the map currently has and only updating the map after all actions and ticks have been called on all tiles/items/actors.

Advantages of Design Change 3:
This allows for more consistency between ground/items/actors when interacting, while reducing potential bugs due to the interactivity between the different ground/item/actors.

Disadvantages of Design Change 3:
There could be a possibility that the reason why the game engine is processed this way after many semesters of improvement is that it was a less buggy option compared to my proposed design change mentioned above.

Problem 4:

World class is coded in a restrictive way that it is not possible to have different behaviours for every method inside the class. This causes an extension of the class that overrides all the methods at start and wastes all the code written in the World class as it is not used at all.

Proposed Design Change 4:

Instead of coding the behaviour of all the methods in the World class, the World class is made into an abstract class. Making all the methods abstract so that the methods can be defined inside the subclasses which the game would need, and there is no excess code.

Advantages of Design Change 4:

It provides more flexibility for the engine. This is because as all different games that use the same engine could have different behaviours in the methods. For example, the winning condition or the losing condition is different from game to game, the original World class inside the engine does not allow any of that to happen. We can define new methods that suit the game more inside the subclass of World and not repeat any code.

Problem 5:

In order for Eco Points to be implemented, I have to introduce a private static integer point (which is ecoPoints) and also introduced public static methods such as

gainEcoPoints, getEcoPoints and setEcoPoints. Having these static methods as public allows for user intervention, as users can call these methods without any problems. This is technically a huge problem as it technically results in very easy hacking of users, as they can just set or gain the number of ecopoints they wanted easily.

Proposed Design Change 5:
Having EcoPoints (or some sort of point system) in the game engine so there's no need to call for a public static method or creation of a private static integer in game.

Advantages of Design Change 5:
This can result with less intervention from user, as it will be more difficult for users to call these methods since they are private methods.

Disadvantages of Design Change 5:
The game engine will be less flexible, as all games that make use of engine will have to either implement a point system to make use of the code, or not use the code, and result in a longer unnecessary processing time, which is very wasted.