THE UNIVERSITY OF SYDNEY

SOFT2201 Assignment2 Report

Zhiyang(Alex) Zhang

SID: 470482647

Unikey: zzha6434

1. A discussion on how your design for assignment 1 helped or hindered your extensions made in this assignment

Honestly, I think my design for assignment 1 is too poor to help me in this assignment. Nevertheless, there are still two aspects of help resulting from my assignment 1's design. Firstly, at least by attending to make a design, I have a better understanding of the scope and overview towards the structure of the assignment, which gives me the fundamental of making better design this time. Secondly, a good design is to make all the objects including clouds, platform, hero, enemies etc. extends from entity because they all have similar attributes and behavior.

2. A discussion on each design pattern you have used including

- **Factory Pattern**

I have used factory pattern to create all the entities including hero, enemy, cloud, platform, flag and landscape where each of them has a corresponding factory class implementing the interface entity factory. This use of

design pattern is perfectly satisfying the open-close principle, for example in the future there would be adding entities like mushrooms, npcs, the only thing needs to do is to create another related factory and add their entry into the level builder. This also achieves high cohesion and low coupling.

- Strategy Pattern

I have also used strategy pattern for the enemies' moving regularity. This use gives low coupling between enemy entity and their moving strategy, which means if there would be incoming entities share the same moving strategy, it would be easy to achieve. And the moving strategies would be easy to extend simply by adding more classes implementing the moving strategy interface.

- Builder Pattern (In progress)

I also intend to use the builder pattern for the creation of levels, however currently it has not been fully implemented yet and only has a default builder.

- Overall benefits

By using these design patterns, the most significant benefit would be it is much easier to make extensions. By having a clear structure, to make an extension you only

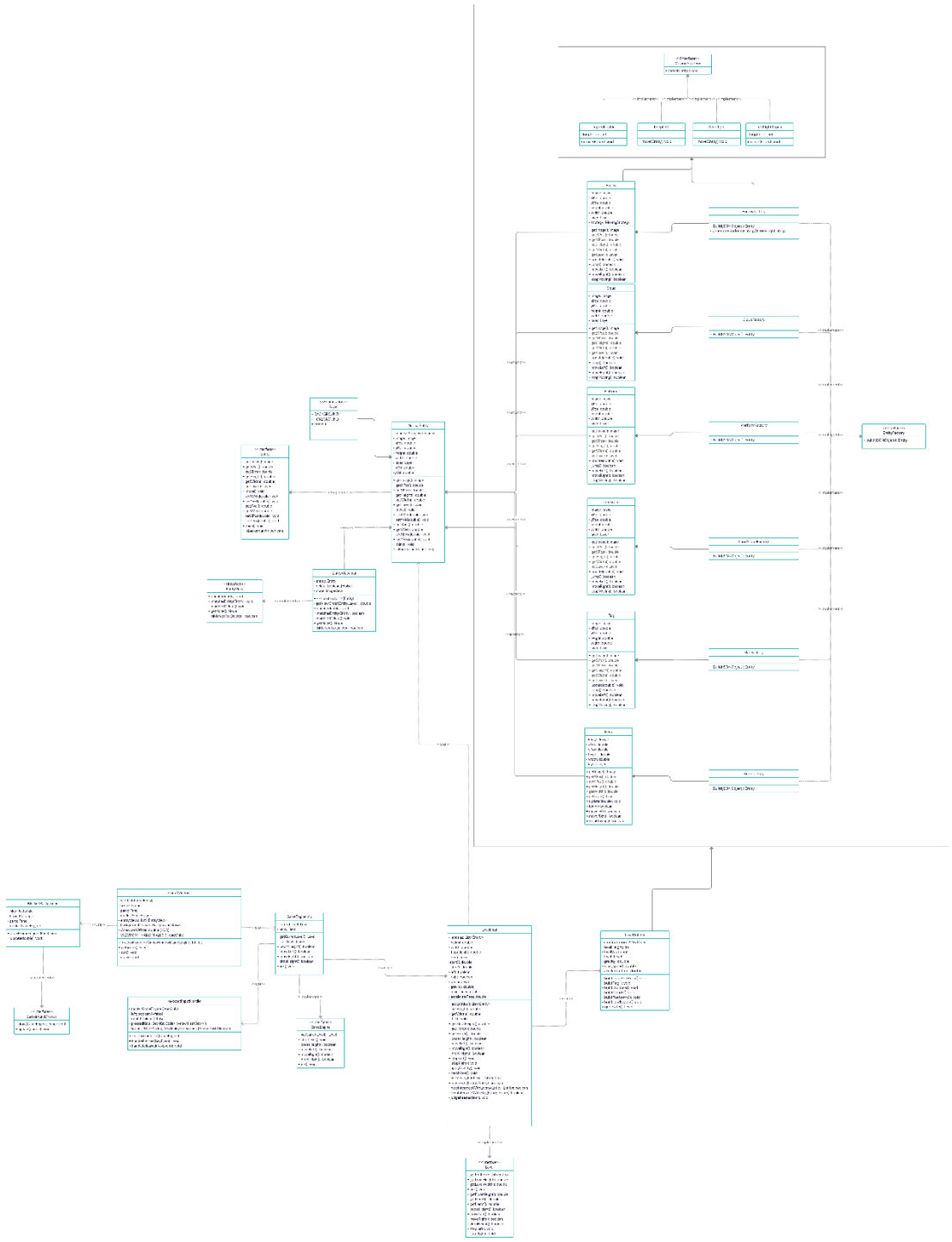
need to follow the structured procedure (Add a class -> Add corresponding factory -> Add the entry). Another benefit would be that the code has much more readability, there are much less if/else blocks or switches in the code. And when you made a change to somewhere, you do not have to make much more changes to all the related area, because all the related area now only uses the method based on the interface.

- Drawbacks

There are still some drawbacks using these design patterns. Formally, the number of classes have been significantly increased, especially for the factory pattern. Informally, it makes drawing UML diagram has much more work to do (:P). Additionally, for the strategy pattern, all the strategies must be public which might decrease the security.

3. UML diagram

https://app.creately.com/diagram/zW3uJjzi8A8/view



(Due to the largeness of the UML diagram, I will also add one png version into the src/main/resources folder in the code base)