2021-10-26 Meeting notes

Date

26 Oct 2021 10pm - 11:30pm

Participants

- Avocado (@ Gabriel Ting @ Andrew Xie @ Prayag Rawat @ William Feng @ amandaliu120)

Goals

- More discussions about UML diagram after creating the initial (stub) functions
- Clarification regarding implementation of certain behaviour

Discussion topics

Item	Notes
Progress + UML diagram	• Refer to the UML diagram called 'Final' as it integrates the ideas of all our initial design plans • We need to actually add in the methods into the diagram before starting to code ◦ From pair programming, Gabriel and William realised that the static/collectable entities don't actually have many methods in those classes - but rather, the methods are primarily called from the player class ◦ Even though we have a general UML diagram, if we do not flesh it out further with the specific methods, we will run into a lot of problems since we may have different ideas about which methods/attributes in each class • Most of us have started writing some of the tests ◦ From the Git history, we have some conflicting tests (duplicated tests that test the same feature), so again, this outlines the importance of the UML diagram
Abstract class vs interface for certain entities	• Use abstract classes for different items due to simplicity of not having interfaces that don't serve much purpose • Interfaces may not contain any methods otherwise • Entity abstract class/interface with Item and MovingEntity abstract classes inheriting/implementing it ◦ LSP: Items, once collected, shouldn't have position properties, isInteractable, isPassable etc.
Movement implementation	• All the positions get updated via the Dungeon class • If we are trying to move Boulders, we would need to do a check within the Dungeon class, so that the boulder moves to the appropriate position, before the character catches up and moves into its original position
How to store items	• NoPositionItem and PositionItem both implement the Entity interface which contains an id • Code (`i` represents item, d represents dungeon, e represents entity) ◦ `i = e.createItem()` ◦ `d.remove(e)` ◦ `inventory.add(i)` • The problem with above code is that once we remove the entity from the dungeon, we no longer know what its class is (e.g. how do we know that it's wood?) ◦ Instead, we can have Dungeon "hiding" the Entity rather than removing

Action items

- [ ] Avocado (@ Gabriel Ting @ Andrew Xie @ Prayag Rawat @ William Feng @ amandaliu120) Complete UML Diagram by tomorrow so that we can properly do the testing/implementation over the next few days

Decisions

- Avocado (@ William Feng @ Andrew Xie @ Gabriel Ting @ Prayag Rawat @ amandaliu120) To add more detailed methods to UML by next meeting
- Avocado (@ William Feng @ Andrew Xie @ Gabriel Ting @ Prayag Rawat @ amandaliu120) Next meeting on Wednesday 27 October at 10:30pm