Project 2: Reflection

Object Oriented Programming: SWEN20003
Christopher Chapman 760 426

My changes:
When I started trying to understand how to do this project, I believe I went about the design the wrong way. In my initial implementation, I was using a map, a 2D arraylist of sprites containing tiles and the corresponding tiles at each tile, in order to check for blocked tiles, other sprites and for interacting between sprites. Being able to pass this in as a variable to all updates, making life a tiny bit easier. However, through this, I had immense problems with making sure that a change to map would lead to changes in world. It felt inefficient and maintaining both the sprites array and map lead to many inconsistencies.

In my previous UML, I also made some choices that lead an unsuitable design. I used inheritance with parent classes that added nothing to the code, having no common methods, had associations from loader to world, when as it was static, this wasn't required. I also could not for the life of me unlock the door with the switch implementation I had.

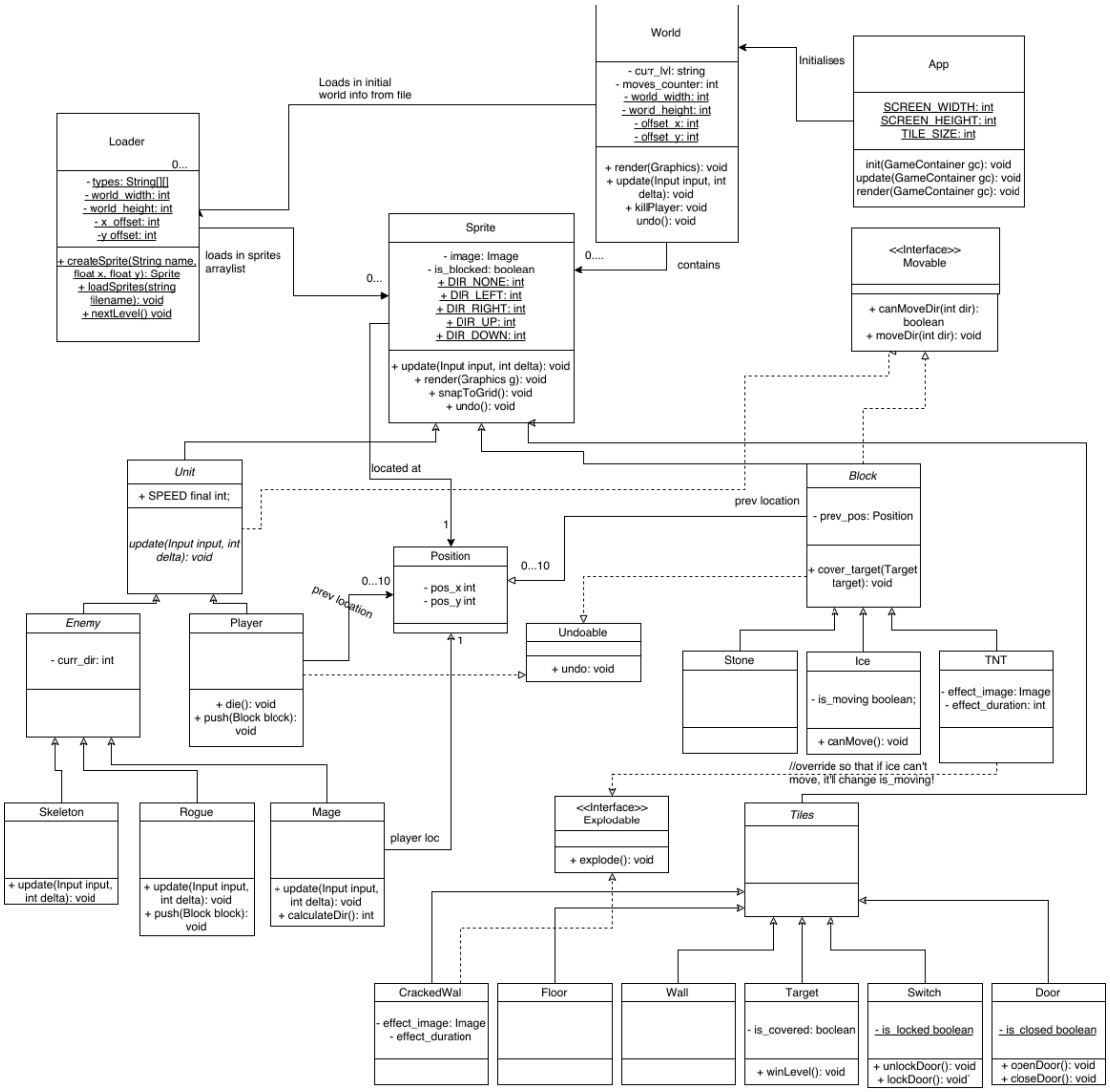
So, I chose to use Eleanor's diagram with only minor changes for added flexibility in some areas. I thought that movable and pushable were much cleaner relationships than in my previous design, being able to use common CanMove, MoveDir methods without having to use implementation, avoiding a lot of duplicate code that I was dealing with.

What I learnt:
I honestly learnt a lot from this project, in particular the importance of planning out how each of the classes would interact and what their delegated roles are. I found that by not planning project 1, I really struggled to implement the project. Yet, even though this project was much larger, the creation and reading of class diagrams really helped me get an understanding of the class relationships and using attribute classes, making what initially seemed to be an impossible task, to be quite reasonable

I am also proud of the way that I used class attributes, methods and constants to allow for greater flexibility. If I wanted to add a z axis, it would be relatively simple to update through position. I allowed for changeable numbers of undoable moves.

Difficulties:
Throughout my creation of the project, I was dealing with privacy issues. In particular, I struggled with accessing the World object from the Sprite classes for intractability between Sprites. My final solution was to use a public static class World, allowing for easy access to blocked locations, Sprites and directions. However, from a privacy standpoint, it left a lot of information readily accessible. If I were to do a similar project, I would place a greater focus on privacy and abstraction.

Previous UML



New, updated UML