Project 2 Reflection – Haydar Jerew 841583

**The default resource folder for this submission contains a texture pack (each sprite has the same name as the original). There is also a backup in the resource folder.**

Changes

One of the immediate apparent differences between my UML for 2a and 2b is the removal of the interface interactable. I took on feedback from 2a and turned it into a subclass of sprites, I also discovered that it was pointless to use an array of Interactables when I could filter out the type of sprites into two arrays in the loader. This works well for me as my loader also constructs the new world and returns it to app so everything is self-contained.

I also had to rely heavily on storing coordinates within world for crack, switch, door and ice. My main aim was to avoid multiple loops through the arrays in the world.update method and I couldn’t figure out how to contain these values within the update methods of the sprites without having to loop through the interacting sprite array to locate them.

Another obvious difference is static classes located within World and Sprite/InteractingSprite. The main objective of these is to reduce the messiness of code and avoid repeating code in Sprites that behave similarly (such as skeleton and rogue).

You will also notice App contains an array of worlds. I felt that this would be the most efficient way to load new worlds due to the nature of my loader class.

Difficulties

One of the most difficult parts of this project was NPC movement. My initial thought was to have 3 2d arrays (one for floor/target, one for interacting blocks and one for NPCs) however, I decided to hold the coordinates of NPCs in the interactable Array in an ArrayList so that I could include ice in the ArrayList when it is moving.

I also had an issue where my mage uses a targeting system based on player coordinates and therefore updates after the player. This causes the mage to act like a heat seeking missile and makes the final level impossible. The mage algorithm matches the one provided in the specification, however I fixed the issue by making the mage move on alternate turns. Whilst this sounds like a massive downgrade, it is still surprisingly difficult to escape the mage.

Another major difficulty was adding the functionality to undo moves. I learned late on that it was not working because I needed to make a deep copy of world and because of all the variables within it, I simply didn’t have time to work this out.

Key Piece of Knowledge Learnt

The main thing I learnt in this project was when I was attempting to maximise efficiency by creating and Image of a Sprite at render time. My friend explained how I was improving memory usage by creating less .png files but because these files are so small, it has no effect of improvement on a modern computer and what I should be focusing on instead was time efficiency. With one small change (creating the Image within the sprite constructor using the source), my game fps went from 400-1000+.

Things I Would Do Differently

The first thing I would do differently is utilise Slick’s coordinate system as it’s annoying to have to initialise an int array every time I want to return coordinates from a method.

I would also create a stack of previous coordinates within each sprite as this is the cleanest system I’ve seen for undoing moves.

Finally, I would have changed my interacting sprite system to incorporate the player into the array as my current system doesn’t allow much room for expansion. It also means that I wouldn’t have to have a separate move and push check for player and NPC within the InteractingSprite class.