SWEN20003: Project 2 - Shadow Quest (Reflection)

Shreyash Patodia

October 14, 2016

Abstract

This document contains a brief reflection on the various stages of development of the game - Shadow Quest which was completed by me as part of project work for SWEN20003: Object Oriented Software Development. A modified final class diagram has also been attached at the end of the document.

1 Changes made to inital design

I spent quite a bit of time thinking about my initial design which translated into a UML diagram which turned out to be more or less complete . Apart from a couple of minor changes, there have not been many changes made to the Class Diagram that was submitted as part of Project-2A.

The first change was that my Camera does not derive from the Entity any more. I realised that although Entity is a useful general purpose class that allows me to set and get positions of Objects in the game, I don't need to do that for the Camera because the only relevant data we want from the camera is the minimum and maximum coordinates rendered on the screen and all of this data can be calculated relative to the Unit that the camera is following.

A second change that I made to the design was the functions of the Interface called Interactable, although functions like withinRange(), action() etc still exist. I now also have an overloaded method called update() and also a render() function, this makes sense because everything that interacts also has to be rendered.

I used the world to set-up my ArrayList of interactables, the methods for which were not as well-defined in my Project 2-A.

2 Difficulties during the project

The biggest difficulty that I faced with the project was implementing the interaction between different Game Objects wherein one Interactable may have different effects on other Interactables. I ended up generalising my solution such that any future extension to the game would not be very hard (for example, adding another AI to the game which aids the player considering he is so overwhelmed by the number of monsters). My solution means that I have

various interacting classes implementing an interface called Interactable, such that they share the property of interaction amongst themselves.

3 Key Piece of Information

Something that I learnt from this project was about how hard is to write good code as projects become larger. It also becomes very important to think ahead and then design the program rather than program and then think about how to use it.

I also got into the habit of using version control regularly during the project allowing me to add version control to my arsenal of skills.

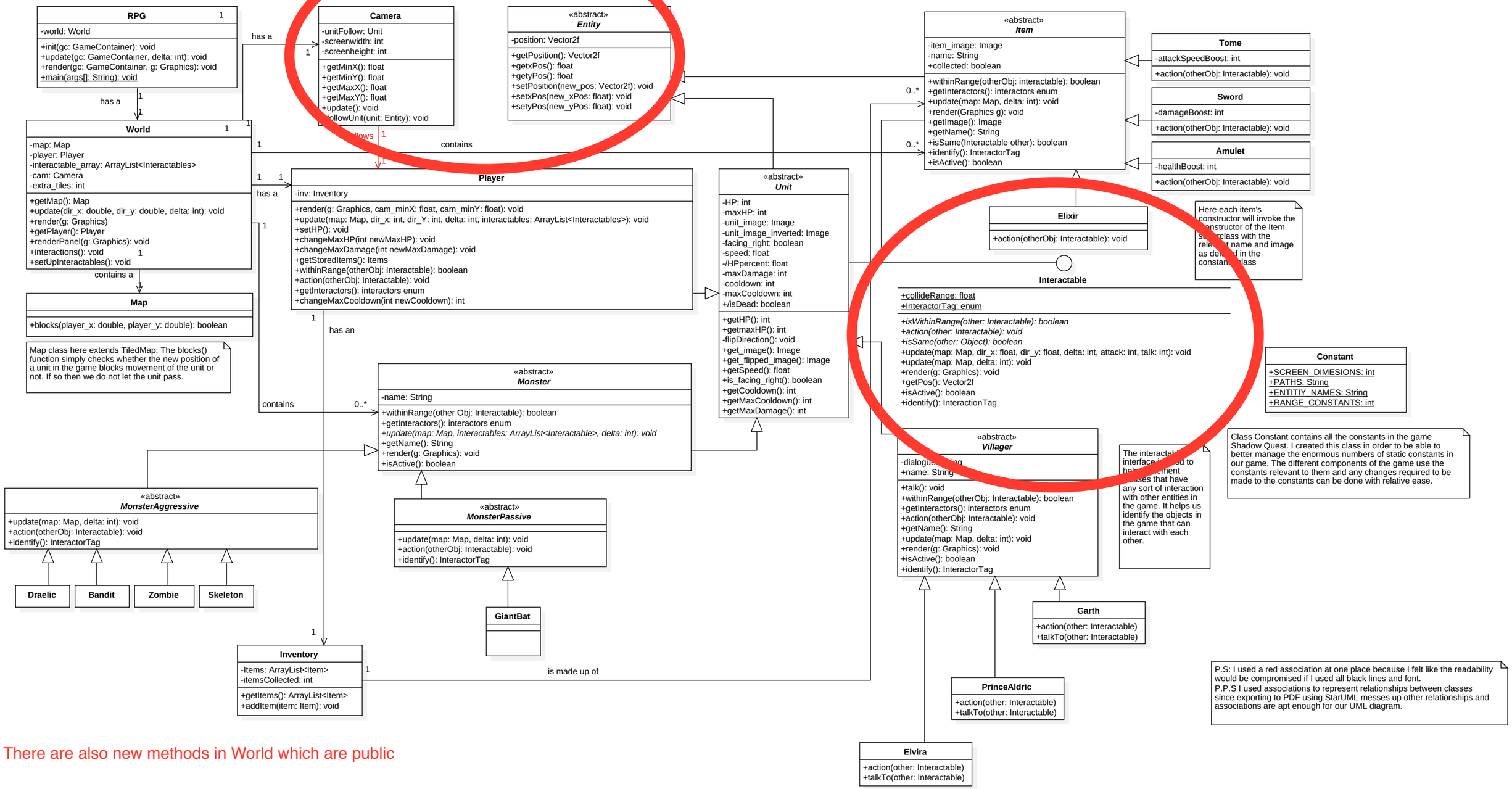
I also got a lot of course related knowledge in the form of usage of Abstract Classes, Interfaces etc. The game has allowed me to appreciate Object Oriented concepts more as I think game development is one of most shoe-in fields for application of Object Oriented Principles.

4 What I would do differently

I would probably think about design more in the beginning, for example I implemented a Player class in the beginning and ended up have a chain that looks like Entity - Unit - Player (where x - y represents x is a superclass of y), if I thought more about the design in the beginning I could have made the above segmentation earlier allowing me to expand my game in project 2B more easily. As it turned out I had to spend hours and hours in order to segment my code that was initially lumped together.

Model::Main

As we can see no relationship between Entity and Camera anymore



There are also new methods in World which are public

The changes in Interactable trickle down to a lot of the other classes like Player, Monster (basically all Interactables)

Major changes in Interactable interface on advice of the tutor