Project Specification - Roguelike RPG

Ying Stokes

October 2020

1 Introduction

This project is a Rogue-like Role Playing Game implemented using the ncurses library for graphical output. It allows the user to control an adventurer to explore a dungeon. The dungeon should contain procedurally generated rooms which contain similarly procedurally generated entities of interest, such as monsters and items.

2 Design Description

2.1 Assessment Concepts

2.1.1 Memory Allocation from the Stack and the Heap

- **Arrays:** The game map will be implemented using a two-dimensional array.
- **Strings:** Most objects such as monsters, items, and parts of the environment will have textual information associated with them, such as descriptions of enemies.
- **Objects:** The project will rely heavily on object-oriented design, so most functionality of the game will use objects, such as entities, items, levels, the renderer, etc.

2.1.2 User Input and Output

- Input/output will be implemented in a blocking fashion (as the game is turn-based) to retreive user input commands in the form of particular keypresses (or combinations thereof).
- This will be achieved using ncurses, which will parse user input per-character, and alter the game state based on a number of factors (such as menu contexts etc.).

2.1.3 Object-oriented programming and design

- Objects will be used to model all aspects of the game environment, such as objects representing terrain tiles, and game entities.
- **Inheritance:** In order to implement various types of animated entities in the game, such as the player, as well as monsters that might oppose the player, a parent class to model a living creature will be used.
- **Inheritance:** Items in the game must all be able to do certain things, such as getting picked up by the player, or being dropped, or used. Thus, all items must inherit a base "item" class, while each individual item type may incorporate unique behaviours.

2.1.4 Testing

- The functionality of individual processes in the game will be tested manually by integration testing.
- In addition, unit testing will be used to verify that behaviours such as applying damage to creatures works properly.

2.2 Class Diagrams

2.3 Class Descriptions

2.3.1 Game

This will be the main object representing the game in its entirety, containing the instances for other objects, such as the UI, the renderer, and the levels.

2.3.2 UI

The UI object will contain the code which will control the renderer object

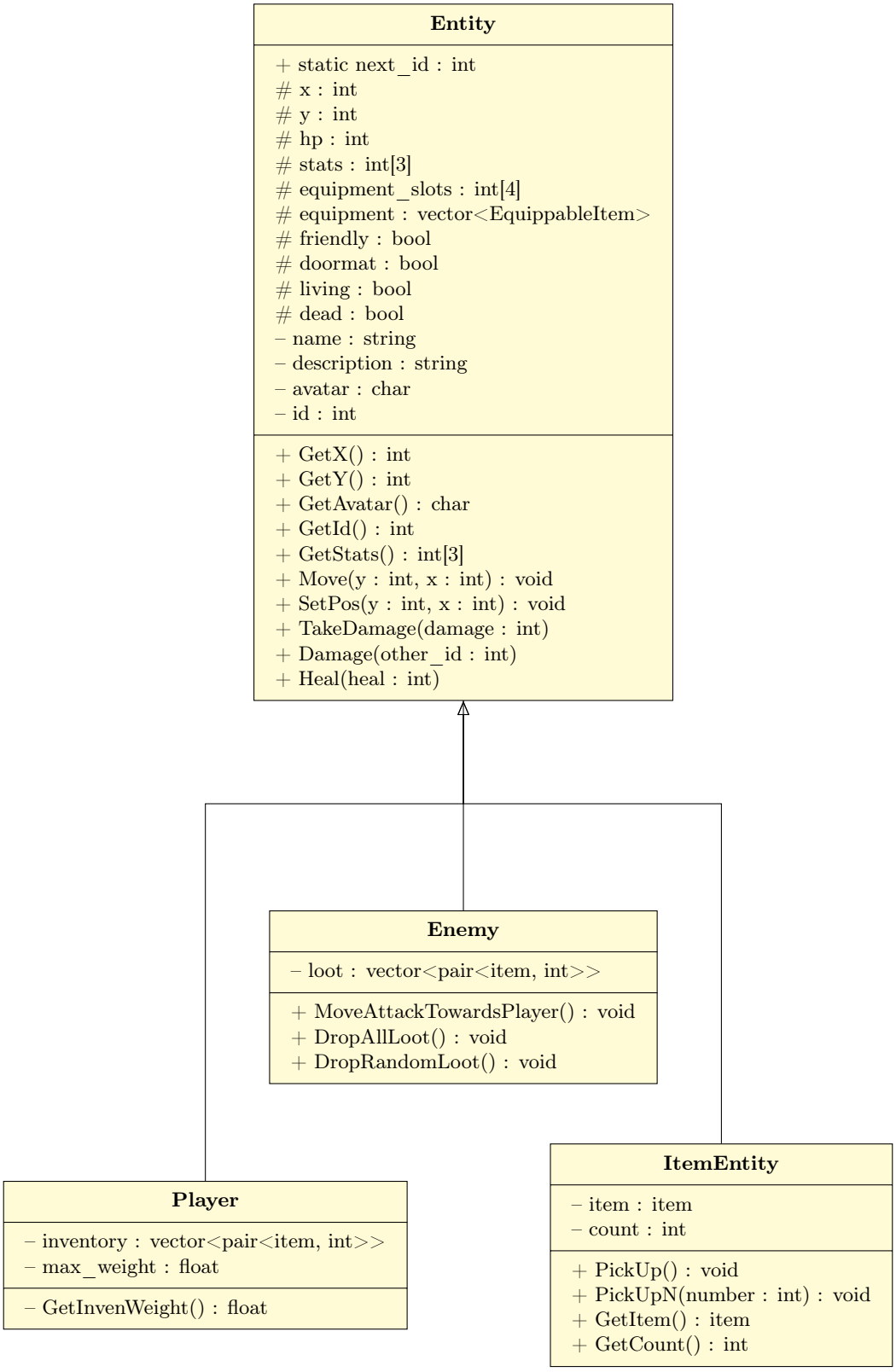

Figure 1: Entity Class Diagram

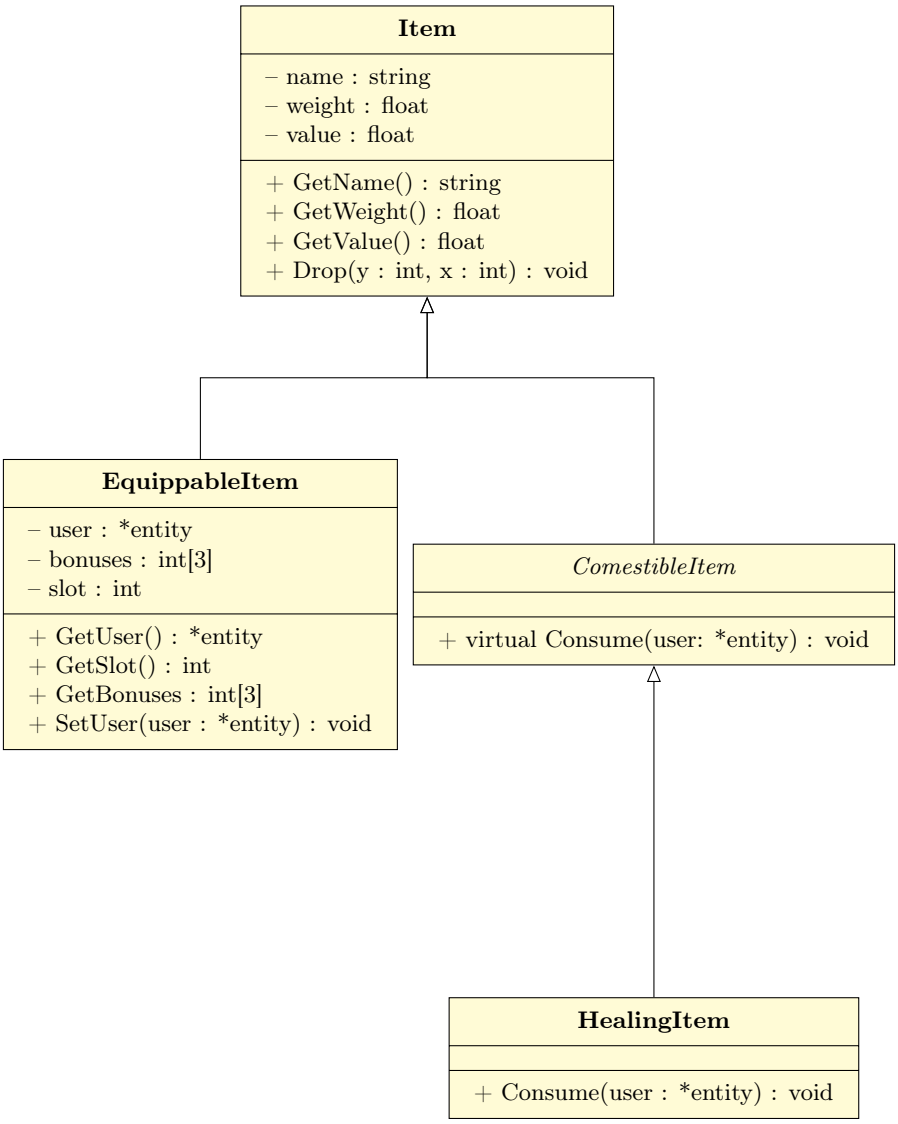

Figure 2: Item Class Diagram

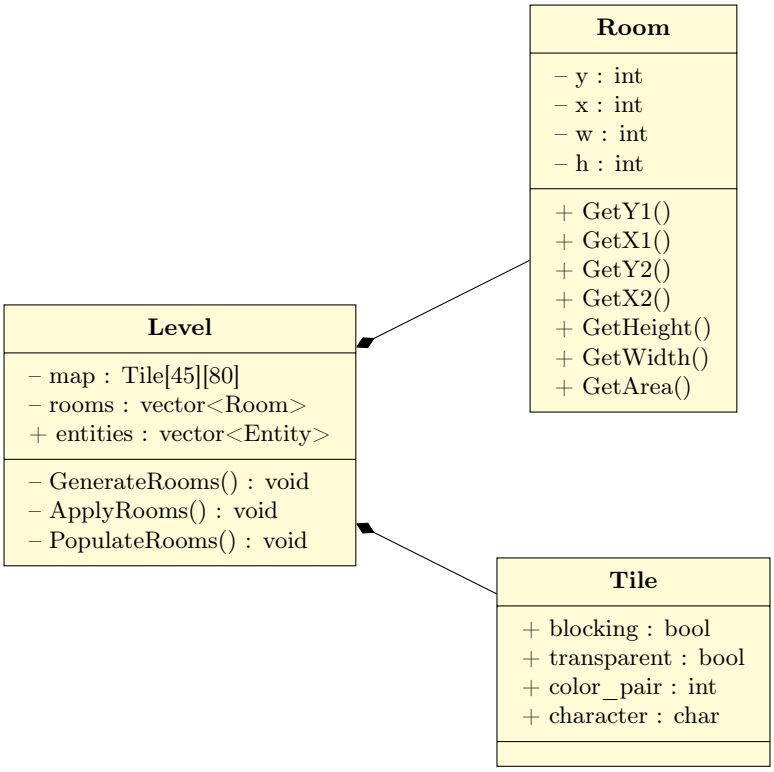

Figure 3: Level Class Diagram

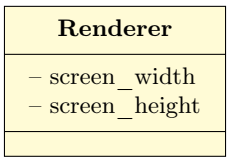

Figure 4: Renderer Class Diagram