Software Requirements and Design Document

For

Group 16

Version 1.0

Authors:
Kevin Sturge
Robert Romero
Valeria Torres
Asher Adams

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

We are proposing a video game made in Godot, using GDScript and C# as the main languages. The game will be a 2D platformer rogue-like with bosses and original characters. The game will be able to be played on Mac and Windows OS.

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

High Priority Requirements: Our video game requires a procedurally-generated dungeon (1) so that every run is unique. The dungeon system needs premade tile set rooms to be pulled and selected by the program when the dungeon is generated (2). The game needs to have a player that is moved by user input (3), receives and deals damage (4), and has movement logic fit to the nature of the dungeon (5). An enemy system will be in place which will have various enemies generated into rooms (6), with their own movement (7) and capabilities of attacking the player (8). At the end of each dungeon level, there will be a boss fight (9).

Medium Priority Requirements: All entities will have their own sprites with full animations (10). A HUD system will be in place that details player health, location, stats, and items (11). There will be a variety of enemies and bosses that are pulled in the same way a dungeon room is pulled (12). An item and coin system will exist that allows players to receive items within the run that have varying effects (14).

Low Priority Requirements: A shop system will be implemented in which players can purchase things like items, health, shields, arrows, etc. (15). Interactable NPCs will exist in the game where the player can speak to them to receive advice, items, and/or currency (16).

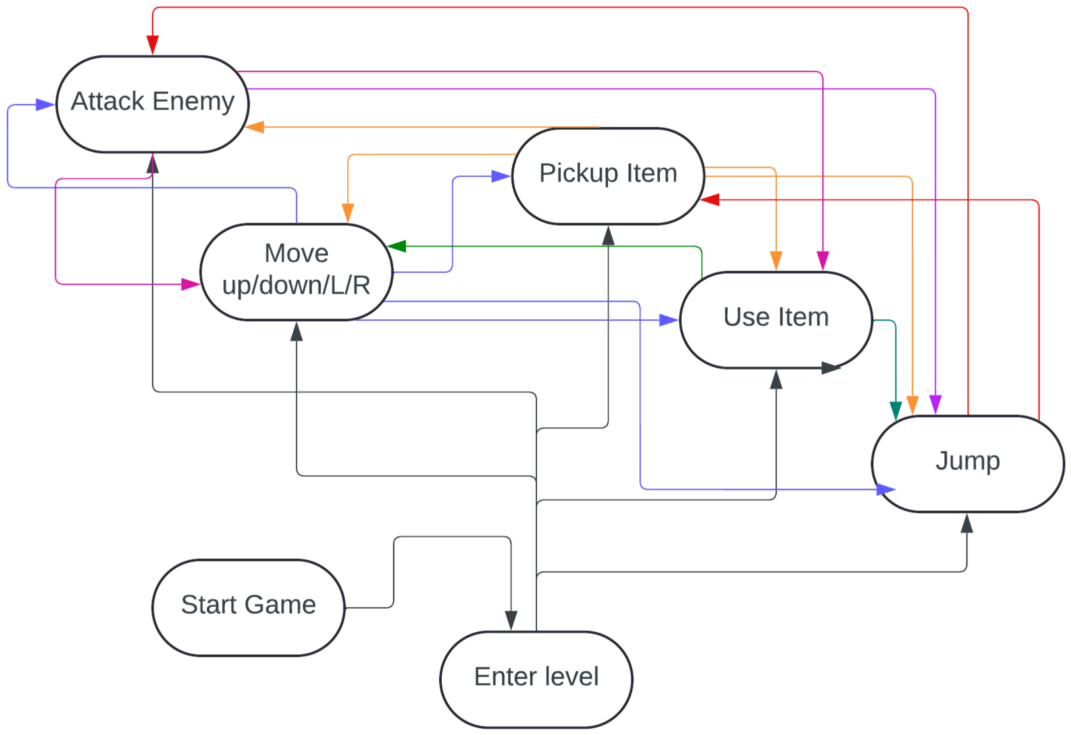
3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

The game should run consistently at 60+ frames per second to ensure smooth gameplay, with low load times of about 2-3 seconds. The room and level generation algorithms should be scalable to increasing complexity without performance degradation and the architecture should be designed to allow addition of new entities without requiring major codebase overhauls. The game should be able to run on Windows or macOS. The code should be easily readable and concise. Overall, the game needs to be enjoyable for players.

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



5. Class Diagram and/or Sequence Diagrams (15 points)

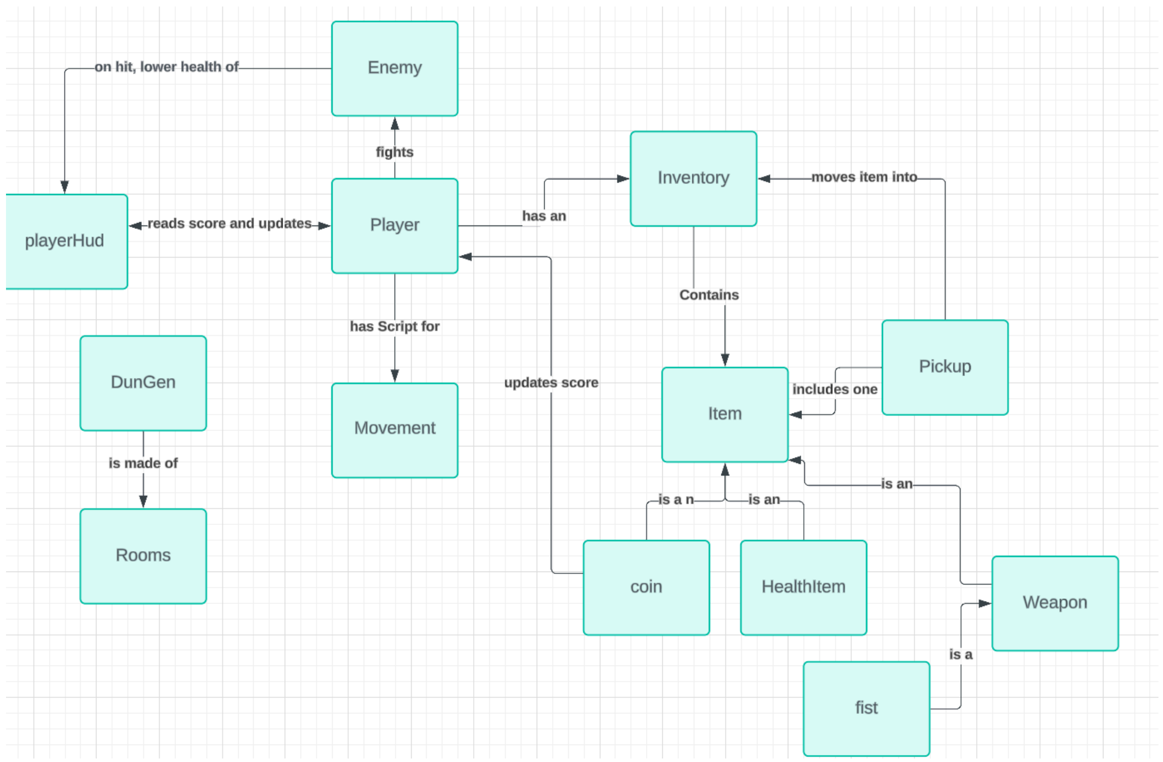
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*



6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

The software will operate on Windows or Macintosh machines. The game will be compatible with windows 10, windows 11, and mac os (no linux). It has to be able to coexist with the .NET architecture and C# due to the Godot engine being compatible with those languages/frameworks.

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

The main factor will be whether or not the game is fun or enjoyable to play. This is not something that can be determined from this early stage, and as time goes on, things are bound to change. What game dev lacks in technical complexity(all we need is Godot) it makes up for in attention to the user experience and iterative development which could make certain work obsolete..