**Software Requirements and Design Document**

**For**

**Group <28>**

Version 2.0

**Authors**:

William White

Julian Schumacher

Ben Semenov

Reid McClellan

# Overview (5 points)

*We are developing a 2D platforming game where the player must progress through 5 levels, while avoiding obstacles and mobs. The game will become more and more difficult as the player progresses through these levels, and mechanics unfold. Collectables, themed to the specific level the player is on are used to incentivize the player top move around, which will eventually lead to unlocked powers or abilities.*

# Functional Requirements (10 points)

# 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.*

# 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.*

# 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.*

# Operating Environment (5 points)

*The game is designed to run on general-purpose hardware including personal computers, gaming laptops, and potentially consoles. It requires a device with at least 4GB of RAM, a 64-bit processor, and a GPU capable of supporting OpenGL 3.0 or higher.*

*PC Platforms: Compatible with Windows 10/11, macOS (10.14 and above), and major Linux distributions (e.g., Ubuntu 20.04 or Fedora 33).*

*The game is built using Godot Engine version 4.0 or above, which must be installed on the development and build systems. The runtime export templates provided by Godot must match the targeted platforms. For running, it requires no external dependencies beyond the system's capability to execute the exported binary. The game is self-contained and does not require external applications to operate. It is designed to peacefully coexist with standard desktop environments and system services. It avoids conflicts with background processes by adhering to standard API usage and resource management.*

# 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.*