**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)

*A new game will start when the player presses the "Play" button on the main menu.*

*Priority: High*

*This is essential for initiating gameplay.*

*The player can exit the game by pressing the "Quit" button on the main menu.*

*Priority: High*

*This is important for providing players with a quick way to exit.*

*The "world" scene will be loaded when the player presses the "Play" button.*

*Priority: High*

*Ensures that the game world is initiated properly.*

*The system shall allow the player to control a character (player.gd script) within the game world, including movement, jumping, and interacting with the environment.*

*Priority: High*

*Character movement and control are central to gameplay.*

*The system shall detect and respond to collisions between the player and enemies, such as frogs and opossums (frog.gd, oposum.gd scripts).*

*Priority: High*

*This is necessary for handling game mechanics such as health loss or game over events.*

*The system shall allow the player to collect items like cherries (cherry.gd script) and update their score or provide other benefits.*

*Priority: Medium*

*Item collection enhances the player’s progression and rewards exploration.*

*The system shall introduce dynamic obstacles and enemies that increase in difficulty as the player progresses (implemented in bg.gd and related enemy scripts).*

*Priority: Medium*

*Dynamic challenge is core to the game's increasing difficulty.*

*The system shall transition between different scenes, such as game levels and the main menu.*

*Priority: Medium*

*Scene transitions are essential for level progression and navigation between different parts of the game.*

*The system shall allow the player to return to the main menu from the game world.*

*Priority: Low*

*This is useful for user convenience but is less critical during gameplay.*

*Enemy ai ust work properly ie following the user*

*Priority high*

*Game must take you to the menue upon death*

*Priority: High*

*Killbox under level that resets player to start*

*Priority: High*

# Non-functional Requirements (10 points)

*The game needs to run efficiently on desktop platforms, with compatibility for different operating systems.*

*Priority: High*

*The game should maintain a stable frame rate of at least 60 FPS.*

*Priority: High*

*Ensures smooth gameplay, which is critical for platformers.*

*The game should load scenes quickly, especially when transitioning between the main menu, levels, and game over screens.*

*Priority: Medium*

*The game should efficiently use CPU and GPU resources, especially as new enemies and obstacles (frog.gd, oposum.gd, bg.gd) are introduced.*

*Priority: Medium*

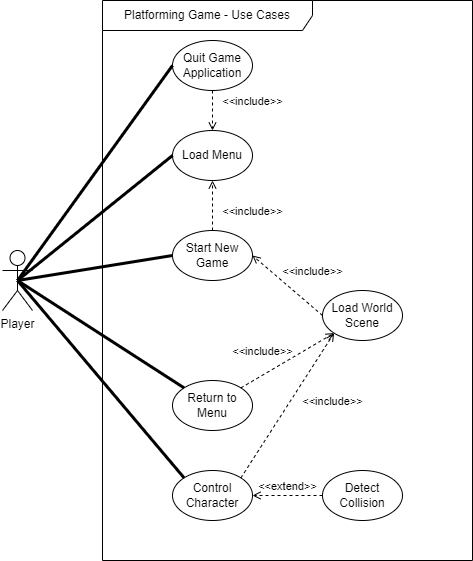
*The game will have a simple and intuitive user interface with responsive controls.*

*Priority: High*

*Essential for an engaging user experience.*

# 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.*

***1. Quit Game Application***

***Description****: Allows the player to exit the game entirely and close the application.****Actors****: Player****Preconditions****: The game application must be running.****Postconditions****: The application is closed.*

***2. Load Menu***

***Description****: Displays the main menu of the game, providing options like starting a new game, continuing a game, or quitting the application.****Actors****: Player****Preconditions****: The game is running, and the player is not currently in gameplay.****Postconditions****: The main menu is displayed.*

***3. Start New Game***

***Description****: Initializes a new game session by resetting all progress and loading the first level of the game.****Actors****: Player****Preconditions****: The main menu is displayed, and the player selects "Start New Game."****Postconditions****: The world scene of the first level is loaded, and gameplay begins.*

***4. Load World Scene***

***Description****: Loads the visual and interactive elements of a specific game level or world, including platforms, enemies, and collectibles.****Actors****: Game System****Preconditions****: The player has started a new game or transitioned to a different level.****Postconditions****: The current level is displayed, and the player can interact with it.*

***5. Return to Menu***

***Description****: Allows the player to navigate back to the main menu from gameplay or other submenus.****Actors****: Player****Preconditions****: The player is in the game or a submenu.****Postconditions****: The main menu is displayed.*

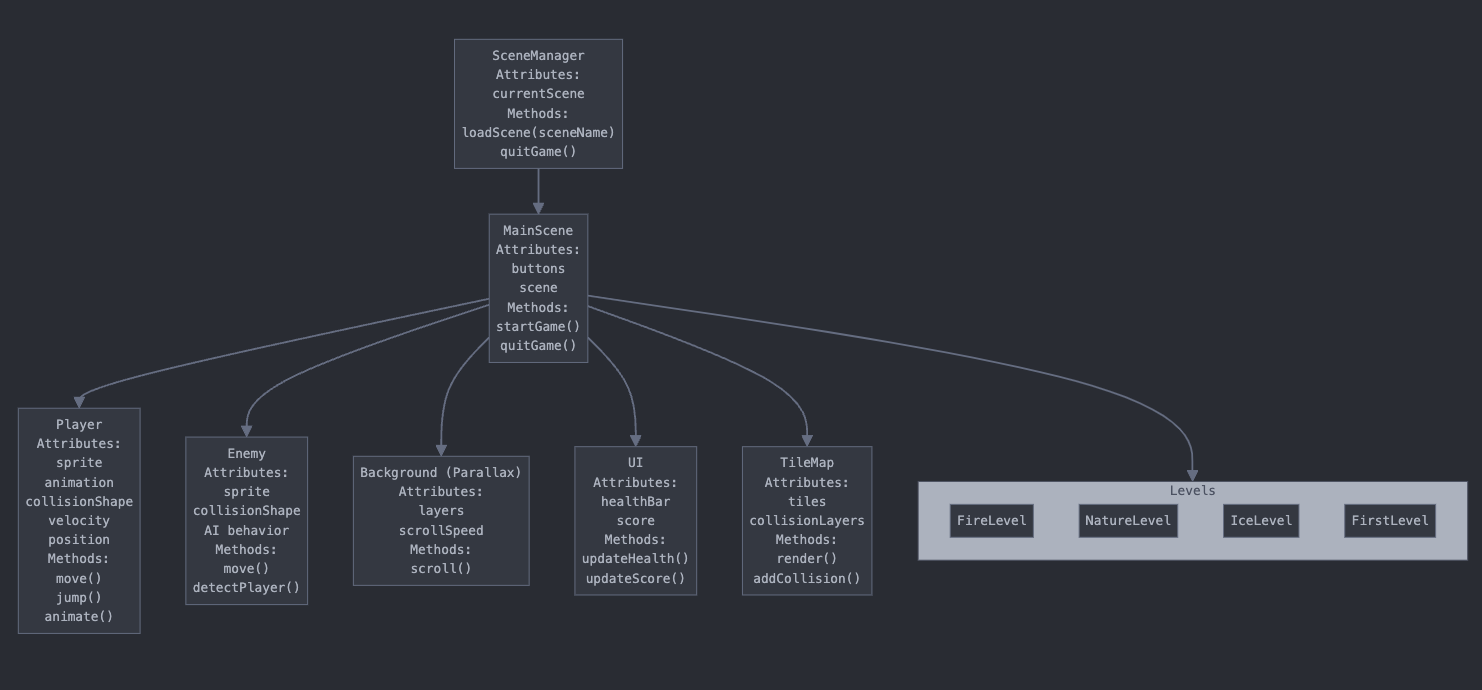
***6. Control Character***

***Description****: Enables the player to move the character within the game world using input devices such as a keyboard, mouse, or controller.****Actors****: Player****Preconditions****: The game world is loaded, and the player is in control.****Postconditions****: The character reacts to the player’s input and moves accordingly.*

***7. Detect Collision***

***Description****: Detects and handles interactions between the player character and other game objects (e.g., platforms, enemies, collectibles).****Actors****: Game System****Preconditions****: The game world and objects are loaded.****Postconditions****: The collision is processed, and appropriate responses (e.g., stopping movement, taking damage, or collecting an item) are triggered.*

# Class Diagram and/or Sequence Diagrams (15 points)



# 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)

*Assuming user has decent PC specs Assuming user is on windows or mac Assuming user has basic understanding of platform dependent on Godot being updated regularly and still being relevant dependent on user having godot installed. Assumes proper installation.*