**Software Implementation and Testing Document**

**For**

**Group <28>**

Version 2.0

**Authors**:

William White

Julian Schumacher

Ben Semenov

Reid McClellan

# Programming Languages (5 points)

*We are using the Godot game engine to develop and run our game. The entire engine uses GDScript, which is designed for the Godot environment. We enjoy Godot’s ease of use relative to other engines.*

# Platforms, APIs, Databases, and other technologies used (5 points)

*We used the Godot game engine as our platform.*

* *GDScript (Godot Scripting) - to manage gameplay logic*
* *main.gd - handles user interactions*
* *main.tscn - defines a menu scene with buttons for Play and Quit, connected to their methods in main.gd*
* *world.tscn - scene that contains static bodies and a player object*
* *forog.gd - Frog and player interactions aswell as frog intelligence*
* *utils.gd - handles save and loading features*
* *player.gd - Player movement and attributes*
* *iceworld.tscn - scene that contains static bodies and a player object themed for the ice level.*
* *Skeleton.tscn- enemy interactions, chase mechanics, and damage dealt, as well as score update upon kill.*
* *Icecrystal.tscn- collectable that updates player’s score*
* *fire\_level.tscn – scene that contains unique fire/hell-themed tile set, beginnings of lava interaction implementation*
* *Nature.tscn - nature themed level using cherries and frogs*
* *Cherry.gd - cherry collectible that increases coin count*
* *Gold/Health nodes - Gold and health count*

# Execution-based Functional Testing (10 points)

*We played through our game’s levels individually as they were constructed to ensure core components were functional and any bugs could be rectified at a cursory level.*

# Execution-based Non-Functional Testing (10 points)

*Tested cases in which the user would try to break the game, tested different collision interactions between enemies and player, observed the number of skips/ frame lag during gameplay and edited background and other components to try and improve FPS*

# Non-Execution-based Testing (10 points)

*During our game development project, we utilized non-execution-based testing methods—including code reviews, walkthroughs, and inspections—to enhance code quality and ensure a robust final product. We established a protocol where each team member's code was peer-reviewed before merging, fostering open communication and early detection of potential issues. Regular walkthrough sessions allowed us to collectively assess critical code sections, discuss improvements, and maintain consistency across modules.*