

Software Implementation and Testing Document

For

Group <17>

Version 1.0

Authors:

Aidan S
Marat B
Zachary H
Olivia C
Zachary H

1. Programming Languages (5 points)

List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

Our game is built using the Godot game engine and we are using GDScript, used everywhere, and GDSherader which is used to color the sprites indicating if they are alive or dead.

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

List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

Godot Game Engine

Asprite - To create sprites

<https://www.piskelapp.com/> - To create sprites

3. Execution-based Functional Testing (10 points)

*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

- 1) We tested the ability to walk in four directions with 'wasd' and made sure that only one input is allowed at one time
- 2) We tested the ability to begin their run by play testing and walking towards the bunker to see if it gives us 3 options, and teleports us to the room.
- 3) We tested the systems ability to track all the stats by printing out to the log when the player gets new coins and their new total amount
- 4) We tested the ability to obtain new coins on the run and use it by play testing and checking our coin count at the start, go on a run and see that it has increased, and checked if we were able to buy a card in the shop we were previously unable to buy
- 5) We tested the ability to buy new cards by using the shop in the lobby scene, buying a card, and confirming it is in our inventory by looking into the chest in the lobby
- 6) We tested the ability for enemies to aggro on players by starting a run and walking towards an enemy and seeing that it begins chasing the player at a certain point
- 7) We tested the ability to end a run by being in the campfire scene after the first level and using the exit to return to the lobby, making sure all the items we obtained stayed in the inventory
- 8) We tested the ability to continue a run by just seeing if the player is teleported to a new run after they complete the first run and are in the campfire room
- 9) We tested the ability for the system to deal out 4 random cards each time by play testing a few times and seeing that the deck we get each battle is different and all part of the 8 that the player is holding.
- 10) We tested the combat system by play testing, seeing that the cards only work when the enemy is in a tile that is allowed by the used card, and made sure the movement crystal works and that the crystal refreshes each turn
- 11) We tested the systems ability to track health by making sure that the health done to an enemy is accurate to what cards were used, and making sure that the enemies die upon going under 0
- 12) We tested the ability to pick the next room from the campfire room as well by seeing if we are taken to a new room and not the lobby or anything else from the campfire room
- 13) We tested the ability to open chests during runs by play testing and finding a chest in the map, making sure it has the ability to open and add coins to the players inventory

- 14) We tested the ability to talk to the npc in the lobby and it giving the same dialogue every time by interacting with it multiple times over different runs
- 15) We tested the ability to get to the boss scene and start the combat scene by play testing and getting to the end and also play testing by starting our character in that room
- 16) We tested the ability for the help button to work by play testing the combat scene and checking if it explains every part of the system
- 17) We tested the systems ability to randomize enemies when taken to the combat scene by play testing and seeing the variety

4. Execution-based Non-Functional Testing (10 points)

Describe how/if you performed non-functional testing for your project (i.e., tested for the non-functional requirements listed in your RD).

1. We tested the ability for the system to allow easy creation of new cards, enemies, and rooms by using the presets, tools, and tilesets to easily create many new cards, enemy sprites, and rooms
2. We tested if the game gets 60 fps by play testing inside of Godot and using a “Debug Menu” free asset (150 fps)
3. We tested the games menu option to turn up and down volume by seeing if the slider actually adjusts volume in the settings page
4. We tested the system's ability to store what the player obtains in their runs by checking the output log when a player gets new coins to see what their new total is.
5. To test if each new scene is loaded in at 2-5 seconds max by play testing and going to every room possible to see that they all were less than 2 seconds load time
6. We tested if the system has an easy-to-use interface by having someone not in the group play test and see if they understood what was going on
7. We tested if the game has an easy to follow route by using the same out of the group play tester and seeing if they understood where they were supposed to go next.
8. We play tested the game on both Windows and Mac and they both worked
9. We tested how easy it is to update features by using the tileset to easily add in new rooms with templates
10. We tested the games ability to provide variety with the amount of different rooms per run by play testing and making sure the game runs use different runs over multiple runs
11. We tested the combat systems ability to randomize the quantity and type of enemy by play testing and initiating the combat scene many times

5. Non-Execution-based Testing (10 points)

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

Whenever the game has to grab a random resource for initializing an object in game we would test it by adding a test resource and loading it in too see if there are any issues where data is being overwritten that shouldn't or attempting to access elements that haven't been loaded in yet.

We also went through with a debugger if there are any redundant or unnecessary functions or properties to ensure cleaner code.

We looked through each of the rooms to make sure there was nothing left over from any now unused sprites/collisionShapes/tilesets

We made sure each room has no visible ways to accidentally exit the map and no ways to not leave that room

We went through the rooms to make sure there was no way the player could see off the edge of the map at the edges of the screen

We went through all the rooms to make sure all the tiles meshed at the color changes and every tree was above a darker tile

We went through every room and made sure every path was walkable, every path allowed enemies to pass through, and each room has a different pattern that gives the player different choices to make