Software Implementation and Testing Document

For

Group 12

Version 2.0

Authors:

Bria W

Nathan O

Amisaday S

Matthew G

Isabela T

1. Programming Languages

The only language we used is GDScript. This is seen in the scripts made for the environment (fireflies, traps), Menu, Heart, Luna (Main Character) and enemies. We chose this language since it was very similar to Python and it is the most compatible with GoDot.

2. Platforms, API, Database, and other technologies used

For the platform, we used the Godot Engine to create a game which involves animation, movement, and scripting. For the API we used the Godot API that includes AnimatedSprite2D to create the main character and the enemy. This also contains the Area2D to make the traps for when the main characters interact with the traps. Along with the Canvas Modulate to help create the light source to help create the theme and game mechanic for the second incrementation. We used a free open source website for the sound effects named “pixabay” where we were able to find sounds for items and our main character.

3. Execution-Based Functional Testing

For us to perform functional testing we would do manual playtesting to see if the function we just implemented works right. Finding ways, we could break in the game or spamming certain inputs to make sure that it would work in the correct way.

This was done in the case of Luna’s Jump and Luna’s attack to make sure it had a cooldown, so that it was not spammable. Likewise, testing to make sure that when Luna is inputted to move in a certain direction she would not go in a different direction.

Another area tested was the Menu and making sure that when the game is paused nothing is affected like the coins and key collected and that the enemies would not move. This was important as losing the key would softlock the player and not be able to continue to the next level. Testing this we would die and see when Luna would respawn on the map and despawn if the player had it.

However, there were certain situations where the Debug button that Godot provides helped improve the testing. From being able to toggle the collision boxes and the visible paths giving us a better representation of whether they work instead of just guessing if we were in the boxes. This helped for when Luna would run into enemies and make sure that she would take damage and change the Player’s Healths on the top left. Then we would be able to handle the case for when Luna’s Health would reach 0.

4. Execution-based Non-Functional Testing

For non-functional testing we input many enemies to see how much the game can load in. To see if the game would start to lag and know how many enemies we can have for a certain stage. This stress test also helped in understanding if certain devices can handle a certain amount of items on the scene. As we all use different devices and different OS systems.

We also organized the project to know where certain things are for a certain stage. Having the Image folder for the sprites, Audio folder for the audio used, Scene's folder for the scenes, and the Scripts' folder for the scripts with comments to help group members.

To help maintain the Consistent Art Style we got outside help, to help us get in the right mood to match the art style that Luna was in. This would be used for when we create the different stages that would be in different environments. We managed to replace all the previous placeholder art for the traps and rats. We frequently talk to our artist to update where our design plan is leading to and what changes we need to already made sprites to make sure our game is flowing correctly.

5. Non-Execution-Based Testing

For non-execution-based testing we would do code reviews with one another. When one of us were ready to make a push and merge we would let everyone know that we were planning to do so via the discord we're all in. This lets us know to check their implementation to see if they missed something. If anything was caught, we would then create issues for any bugs or logic errors. We realized that if two people made changes to the game scene at the same time, we would corrupt our main branch when trying to merge. We ensure to update and communicate to make sure we know exactly who is making what so that we don't run into the same misfortune we did in Increment one, where we corrupted our entire main branch. Our weekly meetups keep us organized and on track so that everyone knows their tasks.