Software Implementation and Testing Document

For

Group <3>

Version 1.0

Authors:
Grace B
Elias D
Jason G
Miguel M
Santiago R

1. Programming Languages (5 points)

The programming language we are using in the project is C#. The reason we are using C# is because the game engine utilizes C# scripts in order to produce different actions and events that the user may want. C# is used in all occurrences for scripts.

We also use the HLSL and Shaderlab code generated by TextMeshPro.

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

TextMeshPro on menus to have the resolution of text scale to different screens and look cleaner than basic Unity text.
Odin (https://odininspector.com/). Not implemented yet but will eventually be used to aggregate variable sliders into singular menus rather than having them spread out throughout the project.
Octave3D (https://assetstore.unity.com/packages/tools/level-design/octave3d-level-design-45021). Unity plugin that we will use for level design as it allows for easier 3D movement of objects through grids and such. Not used yet.
Other Unity Store packs (primarily for visual assets and FX). These will be used throughout the entirety of the project. t).

3. Execution-based Functional Testing (10 points)

Movement System, First person arm animation, and general VFX were all tested as they were in the first iteration, so this is copy pasted:

- 1. Movement System
    - All movement system implementations were tested by trying all combinations of inputs simultaneously / chained after the other to test for conflicts / glitches that could arise from two separate mechanics conflicting with each other.
    - They were also tested for input lag by swapping to different keyboards or mice, and on other gaming PCs. 2.

- 2. First person arm animation
    - These were tested by doing extremely quick mouse movements combined with fast player movement to test that the overlap between movement due to player rotation and movement due to spatial movement overlapped correctly.

- 3. General VFX
    - These are tested by looking at them from all angles that a player could feasibly look at them from, and making sure that they do not break from those angles.

- 4. Enemy Animation
    - These were tested by making sure that the enemy would switch from its moving animation to its attack animation visually through the Unity game engine.

- 5. Damage Animation
    - This was tested in the unity game engine by making sure that when the enemy would take damage the screen overlay would flash red and quickly. Getting the fade to work had

a lot of modifications of the panel's alpha color. Making sure the timing between the attacks and the red damage overlay required extensive number manipulation and timing.

- 6. Damage Decay Formula
    - This was tested by playing with all kinds of numbers until it was a logistical number that could properly affect the player's health. It was tested by testing different differences between the enemy and the player and making sure that the amount of damage that the player was taking between said distance made sense.

- 7. Enemy Patrolling
    - This was tested by getting away from the max distance needed for the enemy to chase the player. I had that distance to 30m, so I tested it by going farther than 30m from the player and making sure that the enemy instead of floating in mid-air went back to patrolling its patrol points.

- 8. Surfaces Affecting Player Movement
    - This was tested by experimenting with the different parameters (ex: cooldown, speed, etc) for each surface that affects player movement.
    - For each surface, I would run the player through and test WASD and jump motion.
    - I also had other people who were not working on this project experiment with the surfaces that affect player movement.

- 9. Dialogue System
    - This was tested by creating sample dialogue objects then playing the game in the dialogue scene to test the dialogue.
    - I made sure to implement various forms of punctuation and dialogue of varying lengths to ensure that the different features that were implemented during this increment were all properly and sufficiently tested.

- 10. Transitions
    - This was tested by running the game many times with the transition set at different time intervals to get the best results.

- 11. Save/Load Systems
    - This was tested by saving and loading with different information and seeing if it will load correctly during the game and after quitting the game.

- 12. Audio
    - This was tested by loading the game many times and testing the menus for audio ques and listening for music and effects.

- 13. Minimap
    - Tested by moving around the entire map and adjusting layers based on object height. Player icon is prioritized and is set to the top layer.
    - Camera zoom was also adjusted to give the player an acceptable view of the map layout.
    - Minimap icon positions were adjusted to accurately represent the map layout.

- 14. Waypoint Marker

- Tested by moving the player camera through all possible angles, ensuring that the marker can be seen at the position of its attached object and that the marker does not move off of the screen.
- Distance tracker was tested by moving towards and away from the goal.

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

Ran a few playtests with friends and acquaintances that were unfamiliar with fast paced platformers to see how well the mechanics could be learned. Mainly observed for time it took to become familiar with the mechanics given no guidance other than a basic tutorial. All players were familiar with traditional first person controls. Most players did struggle a lot and as such a slowdown mechanic is being added.

To test the “feel” of the surfaces that affect player movement, I reached out to some friends and acquaintances to test out the features for me. I had 5 of them use my laptop and see how they felt about the different surfaces and player movement. Overall, they were pleased with the unidirectional speed boost and slow down features. At first, they did not get that for the jump boost, you already had to be jumping and landing on the box in order to receive the jump boost. They said that the boost felt fine, but that it wasn’t necessarily intuitive. As a result, I will implement a jump boost regardless of prior player motion to address that concern. They also said that the multidirectional speed boost felt clunky and could use further refining, which will be addressed in increment 3.

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

We reviewed each other’s work visually through the Unity game engine and suggested ways to improve the project. Just anything to help each other was our form of a demo/inspection. Any impediments or blockages were handled through the discord channel to give everyone on the team an understanding of what the other person was working on and how they were able to fix it / what they needed to fix it.