Software Implementation and Testing Document

For

Group <6>

Version 1.0

Authors:
Reece B
Benjamin R
Aiden L
Kevin O
Nate L

1. Programming Languages (5 points)

We used C# for our scripts in our Unity project because C# is better for novice game developers like us since it handles garbage collection automatically and avoids memory leakage. We use C# for our Unity scripts to create our game components, trigger game events, modify our component properties over time, and determine how our game responds to user inputs. We currently use C# for our player movement script, user inputs to move left and right, jumping, camera functionality, following the player through the level, and a jetpack mechanic that allows the player to have higher vertical movement as well as a fuel gauge for the jetpack that limits the amount of uses a user gets in a level. We also developed a death box and hitbox mechanic so when a player lands on a spike, they lose one of their three hearts, and if the player falls off the level, a death box is activated to reset the player to the beginning of the level with full health and refilling their jetpack fuel. We also created several scripts for platform movement and puzzles, such as a moving platform script and a toggle platform script that allows for specific platforms to be usable while others are not, switching based on a time interval. We plan to use C# for more of our scripts as we develop more levels, such as a dash mechanic for a player to have an extended horizontal movement option and implementing the jetpack fuel mechanic to limit the number of dashes a player can use per level.

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).

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

We are using Unity as our main platform to develop our 2D puzzle solving platformer. Using Krita for some sprite animation.

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).

3. Execution-based Functional Testing (10 points)

Since the last increment, we have developed one complete level. We are working on our second and third, centered around navigating a maze and a future level revolving around platforming on flying cars. We have also begun developing our assets and animations, especially for our player model, for which we have complete sprite animations, hurt animations, and one for when a player uses the jetpack mechanic. We have also developed animations for spinning blades as an obstacle for our levels. We have also play-tested through our first level several times to ensure no buggy movements or player interactions with the environment and correct any bugs when they are found. We messed around with the functionality of our jetpack mechanic for a while, deciding whether it should be more of a hover mechanic or purely vertical movement, and we landed on the ladder for its functionality. We have begun testing our maze level, where the player must use their jetpack, which is currently bound to the Q key, to navigate through the maze without getting stuck or running out of fuel before making it to the exit. Our main issue with it so far is the player running out of fuel, which we plan on fixing by adding jetpack fuel tanks to areas in the maze and our first level.

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

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

We performed non-functional testing on the player's movement as, at times, depending on the angle of the player's wall jumping or platform landing, the sprite will get flipped and land horizontally and is stuck in that position, but the player is still able to move and jump which may cause the sprite to flip again. Also, we are still messing around with the colliders on our platforms, as walking on some of them causes the player's sprite animation to get stuck on one frame and will only allow the user to jump once they get off the platform. We have yet to begin testing the UI and menuing. Still, we plan on moving to it very soon as we finalize the designs for our current levels, and we have yet to make any final decisions on how we are moving a player from level to level. Still, as we finish the current levels we are working on, we shall finalize how to do this. We have tested the modularity and maintainability of our game by making each level its own scene so whatever mechanics we have and use in one level will not affect their uses in others; we have also been following the industry standard for programming when creating our scripts.

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

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

We performed non-execution-based testing on our player movement script to try and determine the cause of the sprite being flipped instead of always staying vertical, which we have corrected. We also spent a lot of time working on the platform movement script as it was causing issues with the player model when the player would move on the moving platform, which we were able to fix by just adding one line of code to our player movement script to prevent those errors from occurring. We also had to walk through the code for damaging and killing the player and figuring out how to teleport them back to the beginning of the level when they run out of health or fall off the platforms. Still, we have them functioning correctly and are working on creating checkpoints for the player so they do not have to restart the whole level every time.

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