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, and a vertical dash for the player's movement written in our player movement script. 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 began working on a third level utilizing the player dash mechanic and platforms that spawn at the start of the level, move in the direction of a deletion point, and once hit, the platforms respawn at the spawning point and do it again.

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 finished working on our second level, the maze, and began developing the third, the future city level. Unfortunately, we could not finish its development, but we created a vertical dash mechanic for the player in our player movement script and “car” platforms as obstacles; we created a spawning and despawning platform script that moves between a set spawn and despawn point. We finished developing our assets and animations for our first and second levels and our player model, for which we have complete sprite animations, hurt animations, one for when a player uses the jetpack mechanic, and a player dash animation. We have also developed animations for spinning blades as an obstacle for our levels. We have also play-tested through our first and second levels 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 finished 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. We have added fuel pick-up objects for each level so a player can have enough fuel to get through the entire level without getting caught.

*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 began testing our menu system for the game and how to move the player between levels. We decided to create a script that takes the player to the next level scene from the one they are currently in. We have tested the modularity and maintainability of our game by making each level its scene so whatever mechanics we have and use on one level will not affect their uses on 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. We have now implemented checkpoints so the player can respawn in certain parts of the level as they progress if they lose a life. But a player must touch a checkpoint before losing a life. Otherwise, our respawn program has no point of respawn, and the player gets stuck at the part of the level they died.

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