Group 102-5 MILESTONE 5

Names: Christopher Gonzalez-Millan, Connor Eamon, Matthew White, Peter LeCavalier, Michael Sciarabba, Remy Vancil

Feature 1: Physics Engine/Movement

In order to test the physics engine, we will be performing trials similar to those typically performed by QA testers within the games industry. The majority of testing will probably consist of simple gameplay loops; we will run through the main game and see that the average player can reasonably complete it in the intended way, tweaking difficulty or scoring values as needed and readjusting boundaries if need be. This might involve completing the level normally, interacting with environments and items and enemies, ensuring that the "goal state" of the game correctly terminates the current level and sets the player up for the next one, and that a failure state (i.e. player character death) works as intended. This will ensure that the general user experience is fun as well as functional. We also want to test edge cases, looking for places where players might accidentally break the game irreparably. Such instances might include falling through the map on unusual terrain (such as stairs), using an item in an aerial position, or hitting multiple enemies at once. These events may not occur very often, but if they cause the game to malfunction or become unplayable they could be quite serious. Additionally, we want to test instances of players deliberately breaking the game; since we are including a leaderboard, we don't want players to feel that they have been cheated out of a high score by someone using illegitimate strategies. This can be tested by looking for exploits in the scoring system with prior knowledge of the game's code.

Feature 2: Leaderboard

In order to test the leaderboard system, we need to ensure that we can add a name and score to the leaderboard and access a name and score from the leaderboard. Additionally, we need to ensure that our code correctly places the names and scores on the leaderboard in their correct order (from highest score to lowest score). We need to ensure that the leaderboard is accessible on the local machine that holds the game, and that the leaderboard is updated instantly. As mentioned with the physics engine test cases, we can test exploits in the scoring system to initially square off any malicious intent. To test the basic functions of the leaderboard, we can run through the game normally to get a score, and use our UI to add the score to the leaderboard. Upon viewing the leaderboard, we need to make sure that the score was added correctly (both individually and sequentially). This will be tested on several machines to ensure continuity.

Feature 3: Animation

Testing for animation may vary based off of how high quality we want to make the animations (for example, very high quality animations could involve many different frames of animation which could be specific to particular scenarios, requiring significant testing for each of these scenarios; in contrast, cruder animation might only occur when walking or jumping and would involve fewer frames, meaning that less testing would be required). Unlike the physics and leaderboard, mere functionality is not sufficient; animations have to look as smooth or as natural as we want them to. In this regard, we would probably have to develop the animations concurrently with the physics engine. Because of this, animation could help test the physics engine and vice versa. For example, if the player character was stuck in their jumping animation after leaving the ground once and landing again, then we might conclude that the game was not properly recognizing the ground state for that object, or that the jump animation was looping infinitely and did not have a properly established point of termination.