Test Plan Document

1. Introduction

In this document, we give the test plan and test results for unit testing of some of the key modules in Labyrinth.

Labyrinth is a 3D, networked, multiplayer game with three different environments one can play in. The environments include “Labyrinth”, which is the main game mode and has the most functionality. In Labyrinth, players race against one another to reach the endpoint of a maze, while at the same time avoid the patrolling cyclopes. The player’s movements should be limited to only the x and z directions so they cannot simply move above the maze walls to reach the endpoint. This movement in the x and z direction is accomplished with collision detectors and gravity using Unity’s built in physics library. The player’s movements should also be impaired in the event that they collide with anything, such as a wall, cyclopes, or another player. One of the other environments includes “Tanks”, where the players are deployed as a small tank in a military base in the desert. The objective of this game is to purely patrol around the military base and interact with other players. The player movement tests that will be performed on tanks is similar to the ones done in Labyrinth. The tanks movements shall be limited to only the x and z direction, and the tank’s movement shall be impaired by any obstacle it collides with. The final environment is “Mountains”, which is a very large open landscape with large mountains on all sides. The players can walk around the environment, but cannot fall off the edge of the terrain due to transparent walls built around the edge of the terrain. There is a bug in this environment, however, where once a player climbs a mountain, it takes a very long time for the player to fall back to the ground in the Y-direction. As a result of this bug, the player hovers above the terrain for an extending period of time until gravity pulls them back to the terrain. For this reason, the tests performed in mountains include if gravity is present, and if the player successfully collides with the transparent walls.

2. Quality Control

2.1 Test Plan Quality

Unity offers a built-in test framework that allows for automated testing of the different functionality of the game and player movements. The way this test framework works is by running each test individually by simulating player movement and comparing this movement to a test case. Our test cases will focus on the base functionality of the game, which is the realistic movement of a character on a 3D environment, while at the same time preforming bounds checking to make sure the player does not move off the terrain. The tests performed for the player’s movements will include:

1. If the player is grounded at all times.
2. In the event a player collides with an obstacle - wall, object, cyclopes, or another player - their movement is impaired from moving forward.
3. In Labyrinth, the cyclopes are tested to ensure they actually move in a random fashion, while at the same time staying within the confines of the maze.

2.2 Adequacy Criterion

Testing will terminate once Unity detects that the character controller of the player or the automated bots has moved outside of its allowed range of motion, defined in each test case. The termination cases are as follows:

Termination Cases		
Test	**Test Description**	**Termination Case**
Gravity Check	Check that the player is grounded on the terrain, and does not float away	The player collides with the ground after a certain passage of time when spawned above the ground
Collision Detection	Checks whether a player detects collisions with other players and objects. This test's objective is to ensure players cannot pass through objects	The player's collision trigger's value is zero in the event of a collision.
Cyclopes Motion	Ensures that the cyclopes move in a random pattern.	The cyclopes x and z direction does not change.

3. Test Strategy.

Our testing consists solely of integration testing. We are testing each module independently: player movement, collision detection, and bot movement, and how they interact with one another. Each test will inform us whether it passes or fails by whether the trigger detects anything. If the trigger detects a change in the coordinates of the players or bots after a couple of seconds, then this would mean that the test passed since the player is in a different location that the initial position. Otherwise, it will inform us that it failed the test. Regarding the collision detection, the trigger will pass the test if the players/bots hit the ground and remain grounded throughout gameplay. It will pass the wall collide test if a player or bot hits a wall and they are prevented from moving through it.

4. Test Cases

Test Case	Purpose	Steps	Expected Result	Actual Result
Gravity (maze)	Test if players fall to ground	Trigger if player collides with ground - result of this trigger determines if test passes or fails	Players fall to ground	Players remain on the ground - test passed

Gravity (tanks)	Test if tanks fall to ground	Trigger if player collides with ground - result of this trigger determines if test passes or fails	Tanks fall to ground	Tanks remain on ground - test passed
Gravity (mountains)	Test if players fall to ground	Trigger if player collides with ground - result of this trigger determines if test passes or fails	Players fall to ground	Players remain on ground until climb mountain, then they take a long time to fall back to earth - test somewhat passed
Wall collide	Test if players hit the boundary wall	Trigger if player collides with boundary wall - result of this trigger determines if test passes or fails	Players hit wall and are prevented from moving forward	Players hit wall and are prevented from moving forward - test passed
Cyclopes movement	Test if cyclopes move	Trigger checks if current cyclopes position is different from initial position - result determines if test passes or fails	Cyclopes move in a random fashion and don’t remain stationary	Cyclopes move in a random fashion and don’t remain stationary - test passed