Software Test Plan

Interrobang

Interactive Puzzle Game

# **‽**

# 

# 10.0 Testing

Many of the features in Interrobang are complex and require a wide swath of testing to ensure their functionality. Because of this, we have implemented a series of individualized and generalized testing schemes to test all areas of our interactive puzzle game. These including everything from manual benchmarking of specific tasks to full-blown beta testing with inexperienced users.

# 10.1 Unit Test Plan

When designing individual tests for each unit of our software, it is imperative that we test all aspects of each module. Otherwise, exploits and bugs will quickly become apparent in our game and will diminish from its immersiveness.

## 10.1.1 Unit Test Descriptions

Our unit tests mostly consist of manual self-performed tests on the module in question. Most often, this is as simple as running the game and testing that there is no glitchiness or unexpected behavior with that given component. Such testing has been conducted on the following modules to verify the following results:

10.1.1.1 The W, A, S, and D keys control movement in the forward, left, back, and right directions respectively.

10.1.1.2 The mouse can be moved upwards, downwards, left, right, and in any combination of any two directions to created camera movement in that respective direction.

10.1.1.3 Holding the shift key while the player is moving increases the player’s movement speed.

10.1.1.4 Pressing the spacebar causes the player to “jump” by lifting them off the ground temporarily.

10.1.1.5 Placing the player over a button causes it to be pressed.

10.1.1.6 Holding the player over a pressed button keeps the button in the pressed position.

10.1.1.7 After a short period of time while not pressed by the player, the button will return to the “up” state. Button functionality continues as normal.

10.1.1.8 Doors can be opened, and will not open any further from their desired open position.

10.1.1.9 Doors can be closed, and will not close any further from their desired closed position.

10.1.1.10 When the player touches spikes or falls out of the map, they “die” and are moved to a spawn location.

10.1.1.11 The players spawn location changes as they move through the levels.

10.1.1.12 When the player lands on a moving platform, they stick to it and are moved along with the platform.

# 10.2 Integration Test Plan

When designing a fully functional game with minimized bugs, it is imperative that all components not only function as expected on an individual basis, but that they work as one cohesive unit during gameplay.

## 10.2.1 Integration Test Descriptions

Our integration tests mostly consist of what would be considered ad hoc testing. The idea was to run the game environment in as close to the true environment it would be run in once it is finished. This consisted of having many people play through various sections of the game to ensure no unexpected behavior or bugs within the game. This was was deliberately chosen to most closely reflect game state, as well as *adhere* to certain game design practices that involve game testing and debugging. The integration of various unit tests yields the following results:

10.2.1.1 When the button is pressed, the door opens, swinging out 90 degrees exactly, allowing player passage.

10.2.1.2 When the player dies, they are spawned at the beginning of the level they were currently playing through, but only after they have pressed the button from the previous level. Otherwise, the player respawns as the beginning of the game.

10.2.1.3 The player navigates through progressively harder levels by pressing a button to open a door leading to the next level.

10.2.1.4 When the player dies, the spawn location that corresponds to the level that the player died in is used to respawn the player.