**Test Report**

**<** Space Crucible**>**

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Author | Revision Date | Comments |
| 1.0 | Parth Patel, Isaac Colon | September 8, 2021 | Added unit tests results, integration test results, acceptance results, and additional test results.  Added List of known problems |

**Table of Contents**

[Document Overview 4](#_Toc89629816)

[System Overview 4](#_Toc89629817)

[Unit Test Results 6](#_Toc89629818)

[Integration Test Results 6](#_Toc89629819)

[Acceptance Test Results 8](#_Toc89629820)

[Additional Test Results 15](#_Toc89629821)

[Known Problems 16](#_Toc89629822)

## Document Overview

The document will provide a high-level overview of the game as well as show the results of all the testing procedures that we will implement during our development phase. The document includes the following sections:

* System Overview
* Unit Tests Results
* Integration Test Results
* Acceptance Test Results
* Additional Test Results

## System Overview

Space Crucible is a two-dimensional, top-down perspective action-puzzle game with support for both single and multiple players. Levels will be defined as a series of square tiles on a fixed grid- however, player and monster movement will not be locked to discrete points on this grid. The combat will be in real-time and take place on tile-based levels. Space Crucible will have a science fiction theme and will emphasize run-and-gun combat style. Players take on the role of “Exterminators” and must use a variety of tactics to combat foes, from weaponry to coercing them to fight each other.

The objective is to explore the map while eliminating monsters, dodging traps, and finding the exit to proceed to the next level. Multiplayer is a key feature; levels can be designed that require more than one player to complete (in fact, the multiplayer-oriented level design will be prioritized). A level editor will be included so users can create their scenarios and save them in a simple text-based format. Levels can be compiled into level packs using a simple archive format called a “.WAD”, which contains and organizes any necessary graphics, sounds, music, and monster code (defined in scripts, rather than hard-coded Java).

Players will receive a launcher upon starting the game. The launcher will allow the user to pick a level pack, a specific level from the pack, choose a difficulty, and either launch a single-player session or join a multiplayer session by inputting the lobby code in the join lobby menu. Real-time action will use WASD or arrow keys to move the player, while the player can simultaneously aim using the mouse. Certain map tiles or objects can be interacted with to proceed in the level or trigger traps. An in-game chat will allow players to communicate with each other. A straightforward light system will allow tiles far from light “sources” to darken, obscuring important puzzle components or hiding sneak attacks. MIDI files will be used as background music.

There will be a Master server that will handle all clients connecting to the multiplayer game mode. Once a player has selected create a lobby or join a lobby option, they will be directed to a specific game server. Each lobby will have a game server that will handle all incoming connections and data from the clients. In multiplayer mode, the game server controls the whole game to ensure that all players are in sync and the host does not have an advantage over other players. The clients are only responsible for sending their input data to the game server; the game server handles moving the players and sending the updated rendering data back to the clients.

Players will be able to host their own lobbies by selecting Create Lobby option in the co-op mode menu. Each lobby will have its unique 4-digit code that the host can share with other players to invite them to the lobby. The 4-digit code is linked with the lobby’s IP address and port number. When the client makes a request to create a lobby, the master server sends the lobby info to the client, and then the client joins the lobby. The players who want to join an existing lobby will choose the Join Lobby option and input the unique 4-digit code. The join lobby request is sent to the master server, and the master server finds the lobby associated with the code and sends the lobby details back to the client. The lobby host will have access to the difficulty selection menu and start game option that other players won’t be able to see. Once the host selects the start game option, the level beings for all players in the lobby. If the host leaves the lobby before starting a game, the host privileges are passed on to the player who joined second.

Once the lobby host starts the level, new players can still join the ongoing level as long as they have the lobby code. This feature also allows players to leave and join the ongoing game as they wish. Another feature we implemented is if a player leaves in an ongoing game, the player will be replaced by a bot/AI player who will help the players finish the level. The bot player can follow the closet player and shoot and chase the visible enemies.

Server administrators can use the remote control (RCON) application to monitor the servers and send instructions to perform certain tasks. The RCON client can log into both the master server and the game servers using a lobby code (or MASTER for the master server) and a password (the master server’s host chooses the master password, the game server password is generated randomly upon lobby creation). When RCON is connected to the master server, it can monitor the number of game servers the master is hosting and check their lobby codes, RCON passwords, and add-on files. When connected to a game server, player positions and health and packet information can be monitored in real-time, chat can be sent to the players from the server, and settings such as game skill and level can be changed.

The program will be written in Java and will support all desktop operating systems (i.e., Windows, macOS, and Linux). There will be separate executable JAR files for the master and game servers, client programs.

A Java IDE will be necessary to develop this project, preferably a common one shared by all developers. The Java LibGDX library is a game development library that includes graphic, sound, music, and networking functionality and will be the framework we use to develop the game. An open-source library, DoomStruct, exists to manipulate .WAD files, which will be used to access game data neatly in an archive format. The KryoNet library is used for network operations, exchanging packets between the game client and the master and game servers over the TCP protocol.

## Unit Test Results

While writing the tests, automating UI testing proved too complex and difficult to be implemented via JUnit. Therefore, any UI testing was moved to the acceptance testing, where it could be done manually.

1. TestIsPortAvailable

Tests to see if the port specified is free to use

1. TestCreateRandomLobbyCode

Tests to see if the lobby code created is random every time

1. TestConnectionToMasterServer

A client tries to connect to the master server

1. TestConnectionToGameServer

A client connects to the game server

1. TestJoinLobby

A client joins the lobby and the lobby since increases to 2

1. TestStartGame

Checks to see if the clicking the start button starts the game

1. TestLevelChange

Tests server-side level transition by sending a level change instruction via a packet (using an RCON message).

1. TestDifficultyChange

Tests server-side skill change by sending a skill change instruction via a packet (using an RCON message).

Results: All tests pass without any problems.

Text

Description automatically generated

## Integration Test Results

Four integration tests were written for this project:

1. The Wad Functions Test
   1. This test makes sure that basic .WAD reading methods work, by opening the default .WAD file, retrieving levels and Entities, and making sure that the resulting data structures are correct by checking specific information (I.e. The default game should have 4 levels and 7 Entities, the Worm Entity should have 150 HP, etc).
   2. Result: All assertions in this test returned true. The program was able to read a .WAD file and convert raw string data into proper Java objects representing game information.

A computer screen capture

Description automatically generated with medium confidence

1. The Editor Functions Test
   1. This test checks .WAD editing capabilities by reading both default and add-on files (as well as a standalone level). One level is opened from a .WAD and the other from a standalone file. Then, one of these levels is modified. The levels were then written into an entirely new .WAD file, and the data was verified by comparing the original LevelData objects to the ones stored in the resulting .WAD file. The first level should be different from the original, while the second level should be identical.
   2. Result: All assertions in this test returned true. The program was able to read the default .WAD, an add-on, and a level stored in a standalone .WAD. The program was then able to modify one of the loaded levels, then write both to a new .WAD.

Text

Description automatically generated with medium confidence

1. The Game Logic Test
   1. This test checks the GameLogic itself by running a non-rendered version of the game. The default .WAD is loaded, the first level is prepared, and the GameLogic starts. Assertions are made that the tic counter is running and that the player moves after having its input set to walk.
   2. Result: All assertions in this test returned true. The test method was able to prepare a non-rendered version of the GameLogic and ensure that the game loop was running properly, and that Entities were being updated.

A picture containing graphical user interface

Description automatically generated

1. The Multiplayer Test
   1. This test checks server functionality. The default .WAD is loaded as usual, a master and game server are created, and two clients are created. One client creates a new lobby and the other joins that lobby. Afterwards, the game is started.
   2. Result: All assertions in this test returned true. The master and game servers start, the first client is able to create a lobby and the second is able to join with the generated code. Then, when the host sends the signal, GameLogic commences similar to Integration Test 3, except in multiplayer.

Text

Description automatically generated with low confidence

## Acceptance Test Results

|  |  |
| --- | --- |
| **Run the Space Crucible application.** For Windows, run SpaceCrucible.jar For Linux, run SpaceCrucible.sh For Mac, run LegacyCrucible.sh in terminal | The application should load up and the main menu should show |

Text

Description automatically generated with low confidence

|  |  |
| --- | --- |
| **Change username.** On the main menu, under the settings option, there will be another option to change the player’s name. | The player’s name from default, which is player(number), will be changed to what you type in and would show up in the game. |

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Change Volume.** On the main menu, go to settings. There will be an option to change the volume. | The volume should toggle from 0 to 100, initially set to 50. |

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Play Single Player Mode.** On the main menu, click on single-player option, a menu will pop-up for level difficulty, choose a difficulty from the dropdown menu and click ok to load the game**.** While in the game use WASD to move and use the mouse to aim and shoot. | After selecting the level of difficulty, the game should start in a single-player mode. |

A picture containing graphical user interface

Description automatically generatedA screenshot of a video game

Description automatically generated

|  |  |
| --- | --- |
| **Select Co-op option** On the main menu, click on Co-op option | If the master server is running you will see the Coop mode menu. If the master server is not running, then you will see an appropriate error. |

Text

Description automatically generatedText

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Create a Lobby.** Under the co-op option, there are two options, one of which is Create a Lobby. Player selects this option if he wishes to host a lobby. | After a short pause, a new lobby will be created with a lobby code and an RCON password for the host. The host will be able to start the game whenever they want. If the host leaves, the next lowest player number becomes the host. If the lobby goes empty, it is removed. |

A picture containing graphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Join a Lobby.** Under the co-op mode in the main-menu, there are two options, one of which is joining a lobby. Select that option and enter the lobby code one wishes to join. | If that lobby code exists, the player will be able to get into the lobby and wait for the host to start the game. If the lobby does not exist, then an error message will appear saying that the lobby code does not exist. If each client's add-ons match the host's, they should be able to join, else they'll get an error that the file names/hashes don't match. |

A screenshot of a video game

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Change Lobby Host.** When in a lobby with at least 2 players the player who created the lobby should select Exit Lobby option or close the game. | Once the host leaves, the player who joined second should be the new lobby host and he should have the start game button. |

A screenshot of a video game

Description automatically generated with medium confidenceA picture containing diagram

Description automatically generated

|  |  |
| --- | --- |
| **Leave and rejoin the same lobby** Open two games if testing by yourself. Use one to create lobby and use another one to join the lobby that was created. Record the lobby code because you will need it later. Once both players have joined let host select start game. After loading into the level have any one player leave. After leaving, the player who left should try to join the same lobby again with the same lobby code. | When the player leaves a bot player should replace the player who left. Once the player who left joins the lobby again his player entity should spawn in and be able to play. |

Graphical user interface

Description automatically generatedA screenshot of a video game

Description automatically generated

|  |  |
| --- | --- |
| **Load Add-On.** From the settings menu, clicking the "Add-ons" button will open the users file directory. The user may then select an appropriate .WAD file to load additional assets stored within it into the game | If the loaded .WAD file contains assets recognizable to the game, the player should be able to access new levels, enemies, and features |

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a video game

Description automatically generated

|  |  |
| --- | --- |
| **Edit Level** Go to the main menu and choose the "level editor". Open a .WAD (archive of many levels) or .LMP (single level in a standalone file). Right-click to add a tile or shift and right-click to add an Entity. Hold Ctrl and drag mouse to select many tiles, or Ctrl-Shift and drag to select many Entities. Middle click deletes tiles and Entities. Ctrl-C and Ctrl-V copy-paste tiles, and Shift Ctrl-C/V copy-pastes Entities. Ctrl-O opens a new file, and Ctrl-Shift-O opens a new level in the current file (if it is a .WAD). Ctrl-S saves. The map name and music track can be modified in the bottom left text fields. | Levels should be able to have their tiles and entities modified, loading new resources from add-on .WADs as necessary. Saving a .WAD should add or edit the current Level inside the .WAD while saving a .LMP should modify the single, standalone level file. |

A screenshot of a computer

Description automatically generated with medium confidenceDiagram

Description automatically generated

|  |  |
| --- | --- |
| **Edit .WAD** Using the SLADE 3 editor (third-party), open a .WAD and edit its assets. Define a new entity using the ENTITIES lump, and provide sprites by adding graphics between S\_START and S\_END marker lumps and sounds between FX\_START and FX\_END. New tile graphics for mapping can be added by placing 64x64 graphics between G\_START and G\_END lumps. | If after editing the .WAD and saving the changes, all the additions appear in the game as the player added/edited in .WAD, then this test is passed. |

Graphical user interface, application, Word

Description automatically generated

|  |  |
| --- | --- |
| **Level Progression**. At the end of each level, there are blue tiles that will send the player to the next level | This test passes if the player is sent to the succeeding level upon reaching the blue progression tiles at the end of the current level |

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Settings Menu**. Select settings from main menu | Takes the user to the next screen where level add-ons, volume, and resolution options are. |

Graphical user interface

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Volume** | Click on the checkbox | If the box is checked the background music will be muted and if the box is not checked the background music will be unmuted. |

Graphical user interface

Description automatically generated with medium confidence

## Additional Test Results

A Stress test for the master server was created to figure out if the server has a limit to how many people can consecutively join and stay connected and also see if the server crashes when its overloaded. In the test a master server is created and in an infinite loop new clients are created, and the clients try to connect to the server. Once connected they send a ping to the server and when the server receives the ping, the server sends a ping back.

Stress Test Result:

2021-11-12 16:55:06.733 Client 1 has connected to the master server

2021-11-12 16:55:06.772 Client 2 has connected to the master server

2021-11-12 16:55:06.792 Client 1 Sent a ping to server and server sent a ping back

2021-11-12 16:55:06.792 Client 2 Sent a ping to server and server sent a ping back

2021-11-12 16:55:06.798 Client 3 has connected to the master server

2021-11-12 16:55:06.799 Client 3 Sent a ping to server and server sent a ping back

2021-11-12 16:55:06.802 Client 4 has connected to the master server

2021-11-12 16:55:06.807 Client 4 Sent a ping to server and server sent a ping back

2021-11-12 16:55:06.816 Client 5 has connected to the master server

2021-11-12 16:55:06.818 Client 5 Sent a ping to server and server sent a ping back

…

2021-11-12 16:57:18.115 Client 160 has connected to the master server

2021-11-12 16:57:18.116 Client 160 Sent a ping to server and server sent a ping back

2021-11-12 16:57:18.119 Client 161 has connected to the master server

2021-11-12 16:57:18.119 Client 161 Sent a ping to server and server sent a ping back

2021-11-12 16:57:18.122 Client 162 has connected to the master server

2021-11-12 16:57:18.124 Client 162 Sent a ping to server and server sent a ping back

2021-11-12 16:57:18.125 Client 163 has connected to the master server

2021-11-12 16:57:18.126 Client 163 Sent a ping to server and server sent a ping back

2021-11-12 16:57:49.441 Client 164 has connected to the master server

Each time the test was performed the test results varied but there was a pattern. Once enough clients had connected the server would slow down and take additional time to connect a client. As you can see the in the result above Client 163 connected at 18.126 seconds and Client 164 connected at 49.441 seconds even though previously they were connecting to the server every few milliseconds. This delay continues and increases every time a new client connects. So, although the server slows down once a lot of users have joined, it does not crash.

## Known Problems

1. Rarely when a client is trying to connect to the servers, there is a little delay. A deadlock fix by the library causes the delay and is out of our control. The only potential fix is to roll back the Kryonet library to the commit before the deadlock was fixed, but it might add more problems to our game, so we decided not to do it.
2. When too many disconnections occur in a game, an error randomly freezes the game server. Since we found this error a little late, we did not implement a permanent fix. But we added error handling so the server and client do not completely crash. When a freeze occurs, the clients can use the ESC menu to exit to the main menu, and once everyone has left the server, the server is reset and ready to be reused.