Line Wars

Test Reports

Testers:

Ryan Frahm

Ezra Cheron

Product Generation and Operation:

Two distinct family members were tested, with a different map and building type in each. As of now, control of both players is given to the user, but otherwise both products fully functioned to completion. They were fun to play, and relatively bug-free. The side-scrolling, zooming and building selection works as expected. The networking works, and as a whole the games are in working order.

Test Cases:

ConfigFileHandler

Test 1: Read invalid file

Step	Description	Result
1	Input Invalid File	Threw expected exception

Test 2: Read valid file

Step	Description	Result
1	Input Valid File	Read as expected

Test 3: Sequence of gets

Step	Description	Result
1	Input Valid File	
2	Perform sequence of gets	All expected values were returned

GameLogic

Test 1: Construct initial state

Step	Description	Result
1	Create initial gamestate to use in a game	Game initialized correctly, with the correct networking references and starting configurations

Test 2: Accepts and transmits messages

Step	Description	Result
1	Obtain a messageReceiver instance	Got a valid instance
2	Add a series of messages to the Receiver	
3	Check on server end that all messages were received	On server end, all messages were received properly

Test 3: Stress-testing for threads

This test was performed by running a number of games on a number of computers repeatedly, watching for any signs of threading issues or tearing. None were reported.

Step	Description	Result
1	Run full game	No threading issues were reported

GameState

Test 1: Construct initial state

Step	Description	Result
1	Create initial gamestate to use in a game	Game initialized correctly

Test 2: Update to next GameState, from first

Step	Description	Result
1	Construct GameState	
2	Update GameState	All first-game-state updates happened correctly

Test 3: Update to next GameState, from subsequent

Step	Description	Result
1	Construct GameState	
2	Update GameState a series of times, with input from the user interspersed	All updates handled correctly

Test 4: Test all information-retrieval

This test was performed by creating an instance of the game, then pausing it and testing all the getters for player info, node info and lane info.

This was performed multiple times, to ensure accurate results.

Step	Description	Result
1	Construct GameState	
2	Run for a specified period	
3	Test all getters/setters	Tests were as a whole correct, with a few minor inconsistencies due to rounding errors

Initializer

To test the initializer, all that was done was to simply create a new game, since that is the only function of the initialize.

Network

The network was tested both in conjunction with, and separately from, the game.

Test 1: Initialize

Step	Description	Result
1	Initialize the network with valid IP addresses	Network initialized correctly

Test 2: Initialize with invalid IP

Step	Description	Result
1	Initialize the network with invalid IP addresses	Obtained expected exception, IP address not found

Test 3: Manually send messages to the network

Step	Description	Result
1	Construct network	
2	Send non-game messages across the network	All messages were eventually received, and packet-handling worked as expected

Test 4: Test in-game message sending

This test was performed by creating an instance of the game, then pausing it and ensuring that all messages that should have been sent, were.

This was performed multiple times, to ensure accurate results.

Step	Description	Result
1	Construct a full game	
2	Run for a specified period	
3	Test received messages.	All messages were received (there were some dropped packets, but the network handled resend requests correctly).

Server

The server was tested both in conjunction with, and separately from, the game.

Test 1: Initialize

Step	Description	Result
1	Initialize the server with valid IP addresses	server initialized correctly

Test 2: Initialize with invalid IP

Step	Description	Result
1	Initialize the server with invalid IP addresses	Obtained expected exception, IP address not found

Test 3: Manually send messages to the server

Step	Description	Result
1	Construct network	
2	Send non-game messages across the network	
3	Obtain full copies of all messages sent on other computers	All computers, including the server computer itself, those that had sent messages and those that hadn't, received the full collated message info correctly.

Test 4: Test in-game message collation

This test was performed by creating an instance of the game, then pausing it and ensuring that all messages that should have been sent and subsequently received, were.

This was performed multiple times, to ensure accurate results.

Step	Description	Result
1	Construct a full game	
2	Run for a specified period	
3	Test sent then received messages.	All messages were received (there were some dropped packets in both directions, but again the network resolved all the issues correctly).

UI

Because the UI is so closely coupled to the game, it was solely tested in use. All the different options were selected in many possible orders. There were a few bugs reported, such as text showing up inadvertently or comboboxes not containing the correct info. There bugs were all reported via a GoogleDoc and then fixed. The fixes were then verified by the testers that identified the issues.

JUnit Tests

In addition to these test cases that tested the publicly exposed pieces of the modules, JUnit tests were constructed and run on internal pieces of the program that were suited for this. Examples of this were the primitive shapes used to represent items in the game, and the collision detection that this system entailed. These tests encountered a few minor mathematical errors in boundary checking, that were promptly addressed.