JSTanks - Test Report

Jiahao Li
LI577
001416646

Pavithran Pathmarajah
PATHMAP
001410729

Viren Patel
PATELVH3
001419057

December 8, 2016

Contents

List of Tables

List of Figures

Overview of Document

This document describes the purposes of the testing done, the scope to which it was done the tools used, followed by a summarized table to test results linking to the tests themselves, and closing with a comparison to the original game.

1 Revision History

Table 1: Revision 1

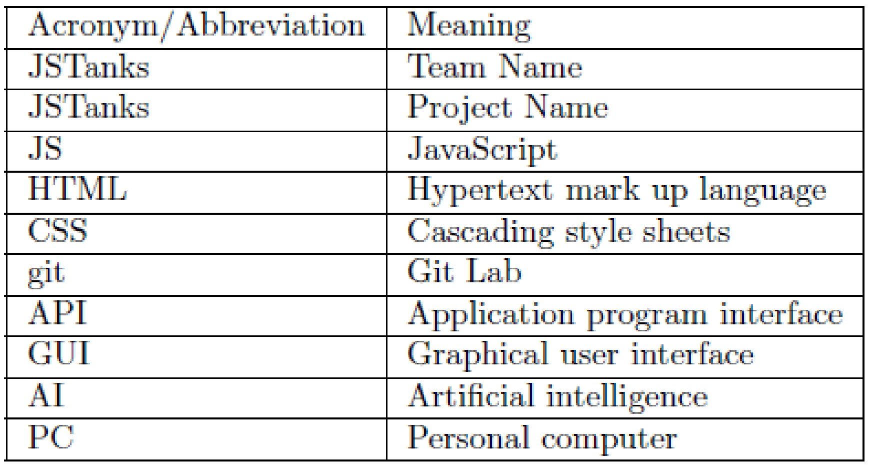
Date	Developer	Change
December 6	Jiahao Li	Initial Draft
December 6	Pavithran Pathmarajah	Initial Draft
December 6	Viren Patel	Initial Draft
December 8	Pavithran Pathmarajah	Include Links

2 General Information

2.1 Purpose

This document is a report outlining the results of the testing, validation and verification process that was to be followed by JSTanks after the build process.

2.2 Acronyms, Abbreviations, and Symbols



Acronym/Abbreviation	Meaning
JSTanks	Team Name
JSTanks	Project Name
JS	JavaScript
HTML	Hypertext mark up language
CSS	Cascading style sheets
git	Git Lab
API	Application program interface
GUI	Graphical user interface
AI	Artificial intelligence
PC	Personal computer

Figure 1: Acronyms

2.3 Scope

This project was designed to port an existing Java based game to the web where it will no longer be required to be download and ran locally, instead it will be able to run in the browser. The tests cover all functionality of the game both functional and non-functional.

3 Tools-Used

The team used an automated test script as well as Google forms, as tools, to help in the testing process. The script was used to perform many tests as fast as possible, as many times as needed to ensure repeatable results to be certain that the game passed its requirements. Where as google forms was used as a means to build and distribute a questionnaire, in order to get qualitative feedback on the game to check if it passed the non-functional requirements set for it.

3.1 Automated Test Script

An automated testing script was made., to check for most functional requirements. The script was able to do many repetitive tests, much quicker then an individual is able to set up a scenario and run through a test. The script was also able to pick out specific calls as they occurred in order to see if the correct result occurred.

3.1.1 Load In Scripts

- Each test ran is located in a separate test file labelled testN, where N is the test number. This script dynamically tries to load in test scripts numbered form 0 to 100, to allow for tests to be added on the fly.
- No other tests will begin until this script loads in the tests, if the script is unable to find the scripts then their is an error and no tests are ran.

3.1.2 Page Load

- Ensures that all HTML pages and subsequent scripts on each page load up, this is checked by the script binding to the webpage and waiting for loading of files to complete. Once all files are loaded for a specified page the script moves onto the next page.
- If any one page takes more than two seconds to load then the test fails.

3.1.3 Menu Functionality

- Runs through all menus on every page and checks that Javascript functions are linked and function correctly.

- If any one javascript call on a menu does not work, or throws an error, the whole test fails

3.1.4 Board Boundaries

- Creates a blank board and places a user tank in one corner, then tries to move the tank 20 times in one direction, (Trying to throw and error on a 15 x 15 board).
- The game then places wall objects on the board in a cross pattern, then runs the same test failing if the tank tile moves to a spot where a wall object is located.
- The script then replaces the wall objects with steel walls and tries again.
- The script then replaces the steel wall object with home bases tries again,
- If the tank entity moves to a location that it should not the whole test fails. The check is done via a script cross checking the x and y position of the tank with the board.

3.1.5 Game Pause

- Checks the pause and unpause features works correctly
- When the game is paused a script runs code and takes snapshots of the board array checking if the board changes at all while paused.
- The script mimics key press of 'p' to pause
- The script then replaces the steel wall object with home bases tries again,
- The script then mimics the continue button being pressed as well
- If the board updates while paused or the continue feature or 'p' key do not work correctly the test fails.

3.1.6 Ai Randomness

- Checks the AI functions by starting a new game and removing all tanks but one AI tank.
- If over the course of two seconds the tank has a net displacement of 2 tiles it passes.

3.1.7 Projectiles

- Checks that the projectile fire function fires a projectile
- Checks that when a projectile hits a wall it takes damage
- Checks that when a projectile hits a steel wall it takes damage
- Checks that when a projectile hits a base it takes damage
- Checks that when a projectile hits a tank it takes damage
- Checks that a projectile is dequeued when it hits something or goes off screen
- Does this by binding to multiple functions in the game javascript

3.1.8 Player Tank

- Checks that user input works correctly, does so by triggering keyboard events.
- The script cross checks the old tank position with new positions on the board, to check if it moved accordingly.
- The script checks that the fire command had worked correctly by placing a wall where the tank should, be and seeing if it was damaged.

3.1.9 Game End

- Checks the game ends when the home base is destroyed.
- Checks with multiple home bases.
- Checks the game ends when the AI is killed.
- Checks with multiple AI.
- Checks if the game ends when the player tank is destroyed.
- Test is done by script binding to the end game call, if any scenario does not end the game then the test is failed.

3.1.10 Link to Test Script

JSTank Automated Tester

3.2 Questionnaire

A questionnaire was issued via google forms to ensure non-functional requirements were met through the process of performing a qualitative analysis to determine whether the requirements were met. The questionnaire issued was 7 pages long and was filled by 8 people. The links below will forward you to a PDF of the questionnaire and the google form itself.

3.2.1 Link to Questionnaire

Google Form - Questionnaire

3.2.2 PDF of Questionnaire

PDF of Questionnaire

4 Test Summary

The game has passed all the tests designed by the team in order to check that it meets all functional and non-functional requirements. These tests were done via automated testing using a custom script, manual testing by the team as-well as by issuing a questionnaire and using feedback to determine if the game succeeded or failed. Below are a series of tables, listing the requirement tests status and last test date (the requirements link to the full requirements located further down this document).

4.1 Functional Tests

Table 2: Functional Requirements

Test #	Team Member	Comments	Date
3.1.1	Automated Script	SUCCESSFUL	12\6\2016
3.1.2	Automated Script	Loads HomePage	12\6\2016
3.1.3	Automated Script	SUCCESSFUL	12\6\2016
3.1.4	Automated Script	SUCCESSFUL	12\6\2016
3.1.5	Automated Script	SUCCESSFUL	12\6\2016
3.1.6	Automated Script	Moves Accordingly	12\6\2016
3.1.7	Automated Script	SUCCESSFUL	12\6\2016
3.1.8	Automated Script	SUCCESSFUL	12\6\2016
3.1.9	Automated Script	SUCCESSFUL	12\6\2016
3.1.10	Automated Script	SUCCESSFUL	12\6\2016
3.1.11	Automated Script	SUCCESSFUL	12\6\2016
3.1.12	Automated Script	SUCCESSFUL	12\6\2016
3.1.13	Automated Script	SUCCESSFUL	12\6\2016
3.1.14	Automated Script	SUCCESSFUL	12\6\2016
3.1.15	Automated Script	SUCCESSFUL	12\6\2016
3.1.16	Automated Script	SUCCESSFUL	12\6\2016
3.1.17	Automated Script	SUCCESSFUL	12\6\2016
3.1.18	Automated Script	SUCCESSFUL	12\6\2016
3.1.19	Automated Script	SUCCESSFUL	12\6\2016
3.1.20	Automated Script	SUCCESSFUL	12\6\2016
3.1.21	Automated Script	SUCCESSFUL	12\6\2016
3.1.22	Automated Script	SUCCESSFUL	12\6\2016
3.1.23	Automated Script	SUCCESSFUL	12\6\2016
3.1.24	Automated Script	SUCCESSFUL	12\6\2016
3.1.25	Automated Script	SUCCESSFUL	12\6\2016
3.1.26	Automated Script	SUCCESSFUL	12\6\2016
3.1.27	Automated Script	SUCCESSFUL	12\6\2016
3.1.28	Automated Script	SUCCESSFUL	12\6\2016

4.2 Unit Tests

Table 3: Unit Tests

Test #	Team Member	Comments	Date
3.2.1	Automated Script	SUCCESSFUL	12\6\2016
3.2.2	Automated Script	SUCCESSFUL	12\6\2016
3.2.3	Automated Script	SUCCESSFUL	12\6\2016
3.2.4	Automated Script	Wall Visible	12\6\2016
3.2.5	Automated Script	Steel Wall Visible	12\6\2016
3.2.6	Automated Script	Home Base Visible	12\6\2016
3.2.7	Automated Script	SUCCESSFUL	12\6\2016
3.2.8	Automated Script	SUCCESSFUL	12\6\2016
3.2.9	Automated Script	SUCCESSFUL	12\6\2016
3.2.10	Automated Script	SUCCESSFUL	12\6\2016
3.2.11	Automated Script	SUCCESSFUL	12\6\2016
3.2.12	Automated Script	SUCCESSFUL	12\6\2016
3.2.13	Automated Script	SUCCESSFUL	12\6\2016
3.2.14	Automated Script	Tank Visible	12\6\2016
3.2.15	Automated Script	SUCCESSFUL	12\6\2016
3.2.16	Automated Script	SUCCESSFUL	12\6\2016

4.3 Non-Functional Tests

Table 4: non-Functional Tests

Test #	Team Member	Comments	Date
4.1	Automated Script	SUCCESSFUL	12\6\2016
4.2	Automated Script	SUCCESSFUL	12\6\2016
4.3	Automated Script	SUCCESSFUL	12\6\2016
4.4	Automated Script	SUCCESSFUL	12\6\2016
4.5	Automated Script	SUCCESSFUL	12\6\2016
4.6	Automated Script	SUCCESSFUL	12\6\2016
4.7	Automated Script	SUCCESSFUL	12\6\2016

5 Tests Performed

5.1 Tests for Functional Requirements

5.1.1 HTML file test

Name: Loading the game
Description: Test if the game dependencies load in browser
Type: Unit test (dynamic, automatic, black box)

Initial State: Testing Script Loaded in
Input: Click the Run Script button
Output: Script is successful, Loads Dependencies
Pass: All files are able to be loaded up.

Status: PASSED

5.1.2 Standby state test

Name: Standby state
Description: Test if the game automatically runs or waits for user initialization.
Type: Unit test (static, manual, black box)
Initial State: A new browser
Input: Go to home page of game
Output: The home page of the game
Pass: The browser remains on the home page.

Status: PASSED

5.1.3 The game section of the menu test

Name: Menu of game section
Description: Test if the sub menu of new game section which has choices of level selection and map selection show up.
Type: Unit test (dynamic, automatic, black box)
Initial State: Menu in the standby state
Input: Script selects new game
Output: Menu Tests successfully
Pass: The sub menu with choice of starting a new game and quit.

Status: PASSED

5.1.4 The pause section of the menu test

Name: Menu representation
Description: check if the menu contains Home Page/Continue/ Instructions/New Game/Quit
Type: Unit test (dynamic, automatic, black box)
Initial State: A new browser
Input: Click the Run Script button
Output: Menu Tests successfully
Pass: The menu with five sections is represented in the standby state.

Status: PASSED

5.1.5 The level section of the menu test

Name: Menu of level
Description: Test if the sub menu of level section which has choices of level 1, level 2, and level 3 shows up when the level section is clicked.
Type: Unit test (dynamic, automatic, black box)
Initial State: Menu in the new game state
Input: Script runs through levels
Output: New Game Success
Pass: The sub menu with choice of level 1, level 2, level 3, level 4 and level 5.

Status: PASSED

5.1.6 Game start test

Name: Start the game
Description: Test if the game shall be reset and start when [starting a new game] is clicked.
Type: Unit test (dynamic, automatic, black box)
Initial State: The sub menu of the game section
Input: Click the choice of starting a new game
Output: New Game Success
Pass: The standby state of the game with all objects on their initial position on the map.

Status: PASSED

5.1.7 Game quit test

Name: Quit the game
Description: Test if the game comes to ends and the user is redirected to the repo if the quit is clicked
Type: Unit test (dynamic, automatic, black box)
Initial State: The running state of the game
Input: Click the Quit option
Output: Menu Tests successfully
Pass: The game is redirected to the gitlab repository.

Status: PASSED

5.1.8 Game pause test

Name: Pause the game
Description: Test if the game comes to the pause state when the letter 'p' is pressed on the keybaord.
Type: Unit test (dynamic, automatic, black box)
Initial State: The running state of the game

Input: Script triggers a keyboard event 'p'
Output: The game pauses and the pause menu shows up
Pass: The game comes to a pause state. Game content freezes and stays in the temporal positions whilst the Pause menu is overlayed .

Status: PASSED

5.1.9 Game continue test 0

Name: Continue the game
Description: Test if the game comes back to the running state state when the letter 'p' is pressed on the keyboard.
Type: Unit test (dynamic, automatic, black box)
Initial State: The pause state of the game
Input: Script triggers a keyboard event 'p'
Output: The game resumes
Pass: All stuff frozen in the pause state are activated and back into the routine.

Status: PASSED

5.1.10 Game continue test 1

Name: Continue the game in the running state
Description: Test if there is any effect on the game when the choice of continue is clicked in the running state.
Type: Unit test (dynamic, automatic, black box)
Initial State: The running state of the game
Input: Script triggers a click on continue
Output: No effect
Pass: No effect on the running state.

Status: PASSED

5.1.11 AI test

Name: The routine of AI
Description: Test if the AI controls tanks to move and fire randomly.
Type: Unit test (dynamic, automated, white box)
Initial State: Single AI in centre of empty game board
Input: The script is run
Output: Ai Test succesful
Pass: The script tracks the AI movement over the course of 2 seconds, and if the AI has a net travel of more then 2 tiles, then it passes.

Status: PASSED

5.1.12 Level test

Name: Levels of the game
Description: Test if the moving speed of tanks controlled by the AI change when level 1, level 2, level 3, level 4 or level 5 is clicked.
Type: Unit test (dynamic, automated, black box)
Initial State: The running state of the game
Input:The script is run
Output: New game test is succesful
Pass: The script checks the AI movement delay value of each game, and if they correspond high to low with the level chosen, then it passes.

Status: PASSED

5.1.13 Default level test

Name: The default level of the game
Description: Test if level one is chosen if no level is selected. Since the game pareses the level form the URL.
Type: Unit test (static, automated, white box)
Initial State: The new browser windwo
Input: Script redirects to the /JSTanks.Html page directly
Output: The game starts with tanks controlled by the AI moving in the lowest speed
Pass: The script checks the AI movement delay value is at the lowest normal value it can be.

Status: PASSED

5.1.14 Instructions test

Name: The Instructions of the game
Description: Test if The window with the Instructions of the game in it pops up when the section of Instructions is clicked.
Type: Unit test (dynamic, automated, black box)
Initial State: New browser window
Input: Script opens instructions page
Output: Pages Section of script is succseful
Pass: The instructions modal opens with the instructions displayed.

Status: PASSED

5.1.15 Tank test

Name: The movement of the tank controlled by the user
Description: Test if the tank controlled by the user moves left, right, up or down when the left, right, up or down key on the keyboard is pressed and fires when

the f key is pressed.
Type: Unit test (dynamic, automated, black box)
Initial State: The running state of the game
Input: Script triggers left, right, up, down or 'f' keyboard events
Output: Script tank is succesful
Pass: The tank moves accordingly with the key board input and fires when "F" is pressed.

Status: PASSED

5.1.16 Continuous movement test

Name: The continuous movement of the tank controlled by the user
Description: Test if the tank controlled by the user keeps moving in the direction of left, right, up or down when the left, right, up or down key on the keyboard is held.
Type: Unit test (dynamic, manual, black box)
Initial State: The running state of the game
Input: Hold the left, right, up or down key on the keyboard
Output: The continuous movement of the tank controlled by the user
Pass: The tank keeps moving in the correct direction according to the key held by the user until the user release the key.

Status: PASSED

5.1.17 Bullet launch test

Name: Launch the bullet
Description: Test if a bullet is correctly launched by the fire commands
Type: Unit test (dynamic, automatic, black box)
Initial State: Blank game Screen
Input: Script calls the fire functionality of the game
Output: Script projectiles is successful
Pass: A bullet is fired.

Status: PASSED

5.1.18 Bullet movement test

Name: The movement the bullet
Description: Test if the bullet keep moving in the same direction after being launched.
Type: Unit test (dynamic, automated, black box)
Initial State: The bullet is launched
Input: –
Output: Script projectiles is successful

Pass: A bullet moves continuously in the same direction.

Status: PASSED

5.1.19 Bullet disappearance test

Name: The bullet disappearance
Description: Test if the bullet disappears when it hits the tank, wall, steel, home base or the boundary of the map.
Type: Unit test (dynamic, automated, black box)
Initial State: The bullet is moving in a specific direction
Input: –
Output: Script projectiles is successful
Pass: The bullet disappears when it hits the tank, wall, steel, home base or the boundary of the map.

Status: PASSED

5.1.20 Wall hit test

Name: The wall hit by the bullet
Description: Test if the wall disappears when it is hit by the bullet.
Type: Unit test (dynamic, automated, black box)
Initial State: Bullet fired toward a wall
Input: –
Output: Script projectiles is successful
Pass: The wall disappears immediately when it is hit by the bullet.

Status: PASSED

5.1.21 Steel hit test 0

Name: The steel hit by the bullet twice
Description: Test if the steel stays the same when it is hit by the bullet twice.
Type: Unit test (dynamic, automated, black box)
Initial State: Bullet fired toward a steel wall
Input: –
Output: Script projectiles is successful
Pass: The steel tile remains on screen but its strength is decreased.

Status: PASSED

5.1.22 Enemy tanks hit test

Name: The tank controlled by the AI hit by the bullet
Description: Test if the tank controlled by the AI disappears when it is hit by the bullet.

Type: Unit test (dynamic, automated, black box)
Initial State: The tank controlled by the AI is on screen
Input: Bullets fired at tank
Output: Script end game is succesful
Pass: The tank controlled by the AI disappears when it is hit by the bullets.

Status: PASSED

5.1.23 Home base hit test 0

Name: The home base hit by the bullet for first four times
Description: Test if the home base stays the same when it is hit by the bullet for first four times.
Type: Unit test (dynamic, automated, black box)
Initial State: The home base is on screen
Input: Bullets fired at base
Output: Script projectiles is successful
Pass: The home base stays the same when it is hit by the bullet for the first four times.

Status: PASSED

5.1.24 Home base hit test 1

Name: The home base hit by the bullet at the fifth time
Description: Test if the home base disappears when it is hit by the bullet at the fifth time.
Type: Unit test (dynamic, automated, black box)
Initial State: The damaged home base is on screen
Input: Bullets fired at tank
Output: Script end game is successful
Pass: The home base disappears when it is hit by the bullet a fifth time.

Status: PASSED

5.1.25 User tank hit test 0

Name: The tank controlled by the user hit by the bullet for the first time
Description: Test if the tank controlled by the user stays the same when it is hit by the first bullet.
Type: Unit test (dynamic, automated, black box)
Initial State: The tank controlled by the user is on screen
Input: bullets fired at tank
Output: Script projectiles is successful
Pass: The tank controlled by the user stays the same when it is hit by the bullet for the first four times.

Status: PASSED

5.1.26 User tank hit test 1

Name: The tank controlled by the user hit by the bullet at the fifth time
Description: Test if the tank controlled by the user disappears when it is hit by the bullet at the fifth time.
Type: Unit test (dynamic, automated, black box)
Initial State: The damaged tank controlled by the user is on screen
Input: Bullets fired at tank
Output: Script end game is successful
Pass: The tank controlled by the user disappears when it is hit by the bullet a fifth time.

Status: PASSED

5.1.27 Game over test 0

Name: The player tank is destroyed
Description: Test if the game comes to the end state when the tank controlled by the user disappears.
Type: Unit test (dynamic, automated, black box)
Initial State: The tank controlled by the user disappears
Input: bullets fired at tank
Output: Script end game is successful
Pass: The player tank is destroyed and the game enters an End state.

Status: PASSED

5.1.28 Game over test 1

Name: The home base is destroyed
Description: Test if the game comes to the end state when the home base disappears.
Type: Unit test (dynamic, automated, black box)
Initial State: The damaged home base is on screen
Input: Bullets fired at base
Output: Script end game is successful
Pass: The base is destroyed and the game enters an End state

Status: PASSED

5.2 Unit testing for internal functions

5.2.1 Wall type test

Name: Ask for the type of the wall
Description: Test if the program returns the type of the wall when you ask for it.
Type: Unit test (dynamic, automated, white box)
Initial State: Barriers on screen made of specified wall typel
Input: wall.type()
Output: "BARRIER"
Pass: The program returns the type "BARRIER" when wall.type() is called.

Status: PASSED

5.2.2 Steel type test

Name: Ask for the type of the steel
Description: Test if the program returns the type of the steel when you ask for it.
Type: Unit test (dynamic, automated, white box)
Initial State: Barriers on screen made of type steel
Input: steel.type()
Output: "BARRIER"
Pass: The program returns the type "BARRIER" when wall.type() is called.

Status: PASSED

5.2.3 Home base type test

Name: Ask for the type of the home base
Description: Test if the program returns the type of the home base when you ask for it.
Type: Unit test (dynamic, automated, white box)
Initial State: Barriers on screen made of type home base
Input: homebase.type()
Output:""BARRIER
Pass: The program returns the type "BARRIER" when wall.type() is called.

Status: PASSED

5.2.4 Wall draw test

Name: Draw the wall on the game board
Description: Test if the image of the wall shows up on the position we set on the game board in the right size when we call this function.
Type: Unit test (dynamic, manual, white box)

Initial State: The game board with no image on the position (startX,startY)
Input: wall.draw(canvas,startX,startY,tileSize,t)
Output: The image of the wall shows up on the position (startX,startY) of the game board
Pass: The image of the wall shows up on the position (startX,startY) of the game board in the tileSize.

Status: PASSED

5.2.5 Steel draw test

Name: Draw the steel on the game board
Description: Test if the image of the steel shows up on the position we set on the game board in the right size when we call this function.
Type: Unit test (dynamic, manual, white box)
Initial State: The game board with no image on the position (startX,startY)
Input: steel.draw(canvas,startX,startY,tileSize,t)
Output: The image of the steel shows up on the position (startX,startY) of the game board
Pass: The image of the steel shows up on the position (startX,startY) of the game board in the tileSize.

Status: PASSED

5.2.6 Home base draw test

Name: Draw the home base on the game board
Description: Test if the image of the home base shows up on the position we set on the game board in the right size when we call this function.
Type: Unit test (dynamic, manual, white box)
Initial State: The game board with no image on the position (startX,startY)
Input: homebase.draw(canvas,startX,startY,tileSize,t)
Output: The image of the home base shows up on the position (startX,startY) of the game board
Pass: The image of the home base shows up on the position (startX,startY) of the game board in the tileSize.

Status: PASSED

5.2.7 Wall hit test

Name: Hit the wall
Description: Test if the program decreases the points of strength of the wall after it has been hit.
Type: Unit test (dynamic, automated, white box)
Initial State: Wall on screen

Input: projectile hits wall
Output: Script projectiles is successful
Pass: The wall is destroyed and thus disappears from the game board.

Status: PASSED

5.2.8 Steel hit test

Name: Hit the steel
Description: Test if the program decreases the points of strength of the steel after it has been hit.
Type: Unit test (dynamic, automated, white box)
Initial State: Steel Walll on screen
Input: projectile hits steel wall
Output: Script projectiles is successful
Pass: One point of strength of the steel wall is decreased after a hit.

Status: PASSED

5.2.9 Home base hit test

Name: Hit the home base
Description: Test if the program decreases the points of strength of the home base after it has been hit.
Type: Unit test (dynamic, automated, white box)
Initial State: home base on screen
Input: projectile hits base
Output: Script projectiles is successful
Pass: One point of strength of the home base is decreased after a hit.

Status: PASSED

5.2.10 Wall health get test

Name: Get health of the wall
Description: Test if the program returns the remaining points of strength of the wall after calling this function.
Type: Unit test (dynamic, automated, white box)
Initial State: wall on screen
Input: projectile hits wall
Output: Script projectiles is successful
Pass: Return the remaining points of strength of the wall after calling this function.

Status: PASSED

5.2.11 Steel health get test

Name: Get health of the steel
Description: Test if the program return the remaining points of strength of the steel wall after calling this function.
Type: Unit test (dynamic, automated, white box)
Initial State: steel wall on screen
Input: projectile hits steel wall
Output: Script projectiles is successful
Pass: Return the remaining points of strength of the steel wall after calling this function.

Status: PASSED

5.2.12 Home base health get test

Name: Get health of the home base
Description: Test if the program return the remaining points of strength of the home base after calling this function.
Type: Unit test (dynamic, automated, white box)
Initial State: home base on screen
Input: projectile hits base
Output: Script projectiles is successful
Pass: Return the remaining points of strength of the home base after calling this function.

Status: PASSED

5.2.13 Tank Constructor Test

Name: Tank Constructor Testing
Description: Test if a tank object is created with the specified attributes when the constructor is called.
Type: Unit Test (dynamic, automated, white box)
Initial State: empty screen
Input: new tank object is created by script
Output: usertank succesful
Pass: The following methods create the specified tank object in position.

Status: PASSED

5.2.14 Tank Draw Test

Name: Tank Graphics Testing
Description: Test if the image of the tank shows up at the position set by the game board when the function is called.
Type: Unit Test (dynamic, manual, white box)

Initial State: The game board with no image on the position (startX, startY) newline Input: tankObject.draw(canvas, startX, startY, tileSize)
Output: The image of the tank appears on the position (startX, startY) of the game board.
Pass: The tank is successfully displayed on the correct position on the game board.

Status: PASSED

5.2.15 Tank Hit Test

Name: Projectile Impact Testing
Description: Test if the health points of the tank are reduced after the tank has been hit with a projectile.
Type: Unit Test (dynamic, automated, white box)
Initial State: Tank on screen projectile fried at tank
Input: projectile hits tank
Output: Script projectiles is successful
Pass: Specified points of the tank?s health are reduced.

Status: PASSED

5.2.16 Tank Health Test

Name: Tank Health Getter Testing
Description: Test if the method returns the correct number of health points for the tank when called upon.
Type: Unit Test (dynamic, automated, white box)
Initial State: A tank object has been created, and hit by projectile
Input: projectile hits tank
Output: Script projectiles is successful
Pass: The method returns the correct number of health points remaining for the tank object.

Status: PASSED

6 Nonfunctional Requirements

6.1 Appearance / Style

Description: A survey will be provided to classmates whom will test the game and fill out the survey.
Questions:

- Can user Tank be distinguished from AI.

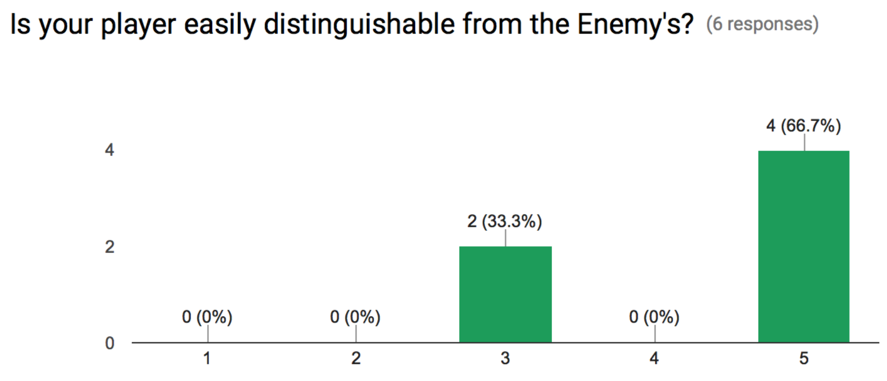


Figure 2: Tank distinguishability Feedback

- Can wooden walls be distinguished from Steel walls.

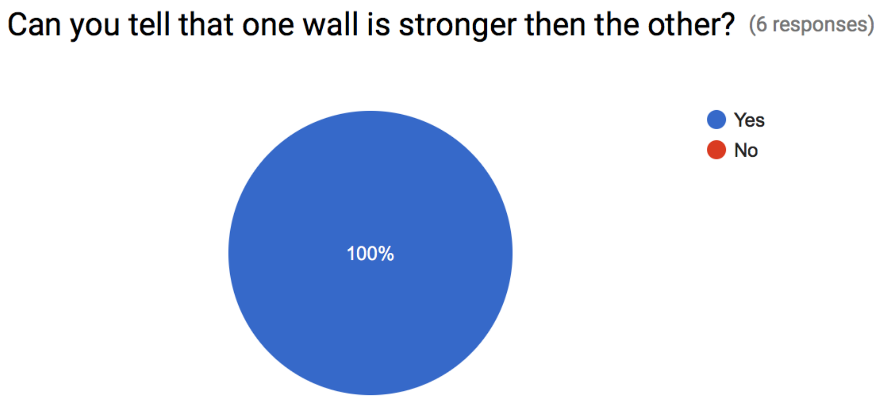


Figure 3: Rate wall strength by material

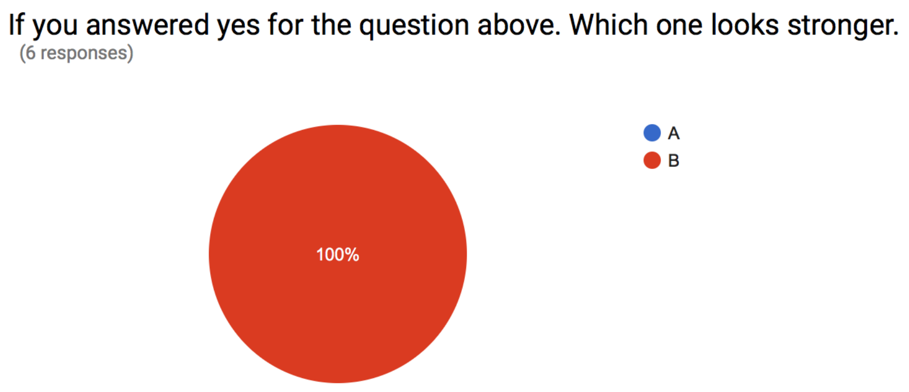


Figure 4: Distinguish wall materials

- Can the home base be distinguished

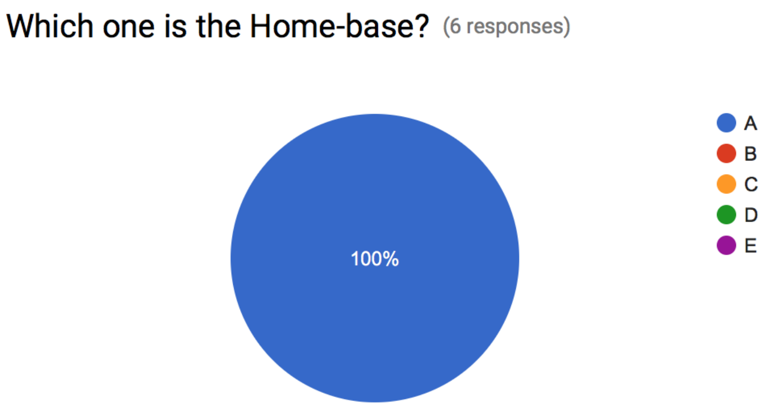


Figure 5: Distinguish Home Base

- Is the overall colour scheme is an eyesore.

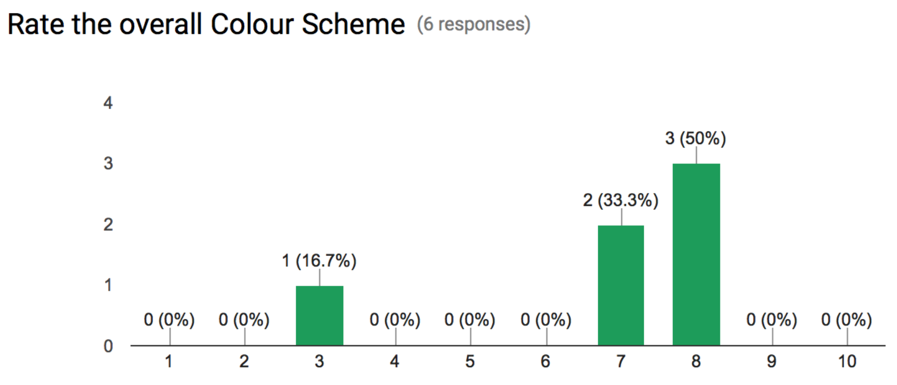


Figure 6: Colour Scheme Feedback

- Does the menu cover everything needed to play the game.

Pass: If the game passes all manual unit tests on all browsers, the requirement is met.

Status: PASSED

6.2 Ease of Use

Description: A survey will be provided to classmates whom will test the game and fill out the survey.
Questions:

- Is the response time satisfying

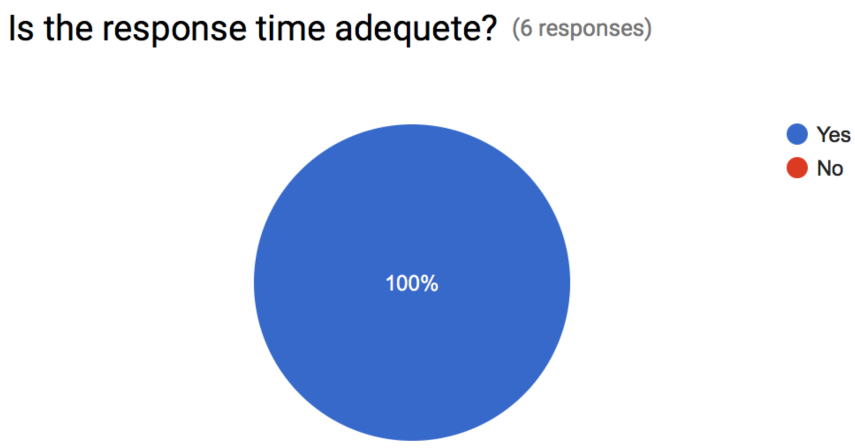


Figure 7: Response Time Feedback

- Is the game straight forward to play

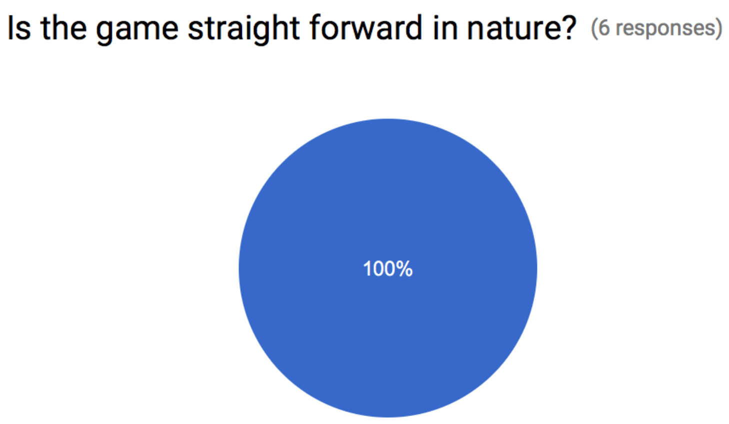


Figure 8: Game Play Feedback

- Are the instructions convoluted in nature.

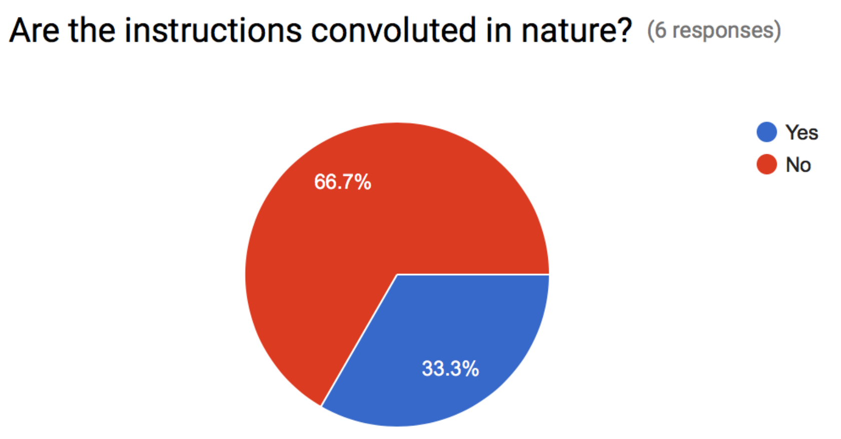


Figure 9: Instructions Feedback

- Are all menus understandable.

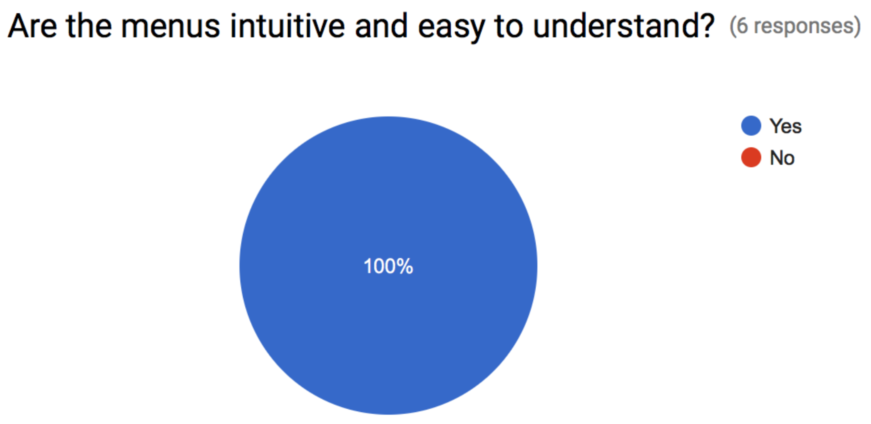


Figure 10: Menu Feedback

Pass: If the game passes all manual unit tests on all browsers, the requirement is met.

Status: PASSED

6.3 Accessibility Requirements

Description: Test if the game functions on stated browsers. How: running all manual unit tests from section 4.1 on the following browsers:

- Google Chrome
- Mozilla Firefox
- Apple Safari

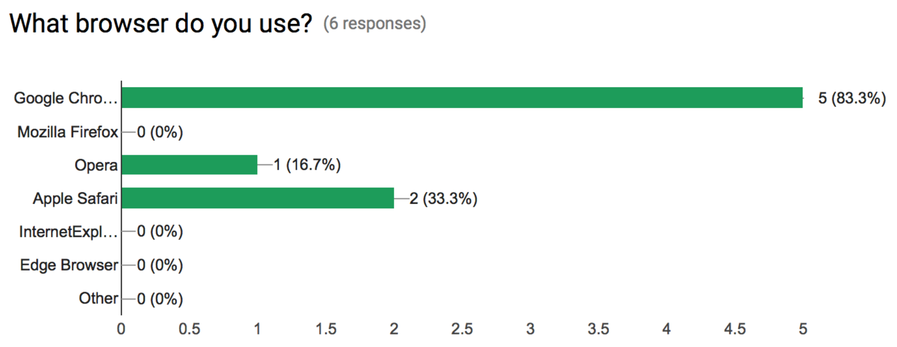


Figure 11: Browsers Used

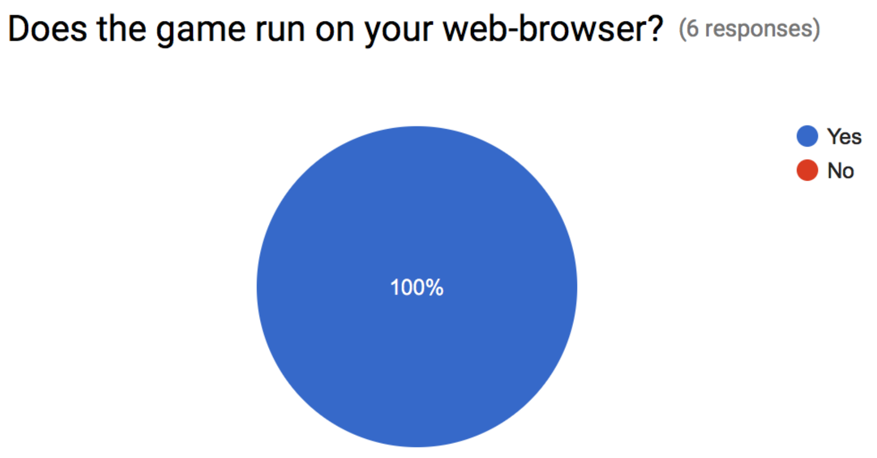


Figure 12: Browsers Feedback

Pass: If the game passes all manual unit tests on all browsers the requirement is met.

Status: PASSED

6.4 Performance

Description: The game should run at an equal speed across all platforms and hardware specifications.
How: Playing a standard game on the following systems:

- Late 2013 - MacBook Pro
- Lab - Virtual Computer
- Thode - Virtual Computer
- 2014 - Surface Pro 3

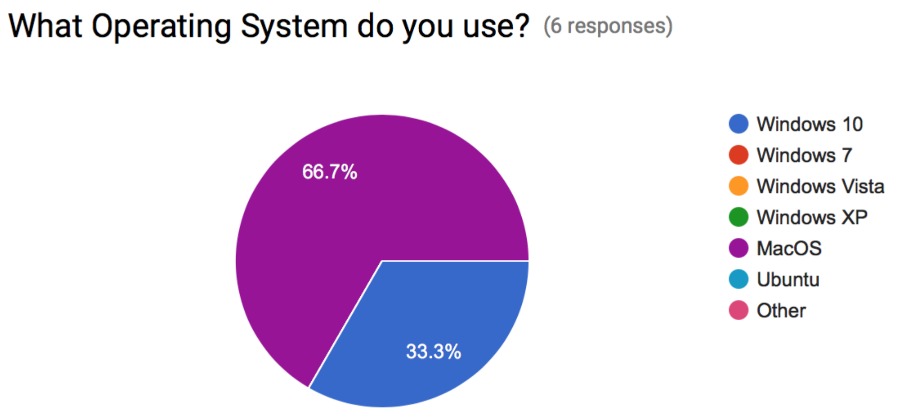


Figure 13: Operating System Feedback

Pass: If game runs at similar speeds across all platforms, where relative similarity will be decided by tester.

Status: PASSED

6.5 Maintainability

Description: The game's source code should be easy to read, maintain, and learn from.
How: Have a classmate whom does not work with JS to read over an object file, and have them tell us if they find the code easy to understand.
Pass: Classmate states that code is easy to read and learn from.

Status: PASSED

6.6 Security

Description: The game should not access and send user data to an external source
How: Download the source files, then close all network connections.
Pass: If the game is able to run off-line, it is not sending any data back. Reasoning: If one method in JS fails the entire script usually crashes

Status: PASSED

6.7 Legal Requirements

Description: The game should follow Canadian Anti-Spam legislation
How: Review legislation and look through code to ensure legislation is followed.
Pass: Legislation is not violated.

Status: PASSED

7 Comparison to Existing Implementation

The open source game software we are working on is simply a Java application in its existent form. We are uploading the same game on a web browser which requires a different programming language altogether. As a result, we have not been able to use any existing code and have had to program the game completely from scratch. We have also not looked into the Java code to learn its implementation style or use any ideas for specific functions. Therefore, any similarities between our code and the existing code are coincidental.

The major difference between the two implementations is that we have HTML, and CSS files in addition to the JavaScript source code which are required for any kind of web development. The existing code works with multiple classes representing different aspects of the game which can be seen in our code as well. However, it has more classes, three of them which are main method classes which can be explained by the programming language used. Our implementation has no main method or class, but instead the HTML file is used as the main class which drives the whole game. Another important difference is the style of programming; the existing implementation has made use of threads which we have not as we are still learning to work with JavaScript. The use of object oriented programming is evident in both implementations. For example, objects for tanks, bots, and barriers are included in both.