SE 3XA3: Test Plan
Staroids

Team 20, Staroids
Eoin Lynagh, lynaghe
Jason Nagy, nagyj2
Moziah San Vicente, sanvicem

December 5, 2018

Contents

List of Tables

List of Figures

This document is meant to show

1 Functional Requirements Evaluation

1.1 Loading into pre-game

To test this functional requirement, we must verify that the game always starts in pre-game state. This can be verified by manual code inspection, as upon start-up, the code has no option but to begin in the pre-game phase. To see this, look at the "start" function in Staroids/src/gamestate.js. It sets the state to "pregame" on start, which cannot be avoided.

1.2 Starting game

To test this functional requirement, we must verify that for all Staroids game in the "pregame" state, they will be moved from the pregame state to the playing state within one second after pressing the spacebar. This can be verified by code inspection as well as manual testing. The game is monitoring for the space input 30 times per second, so it cannot miss a human input. The pregame state is also restrictive to what states can be reached from the pregame state, so it has to move into the playing state within a second. This has also been the subject of extensive testing by many people, and all empirical evidence points to this behavior occurring consistently.

1.3 Prohibit game start on pause

To test this functional requirement, we must verify that when the game is paused, all game functions must stop, all inputs must not be accepted (except un-pause) and all game variables must be preserved. To test this, we can use a combination of code inspection, to verify that the paused state does not allow certain inputs and that it pauses the game state. This can also be verified by the teams testing, which shows that this behavior is consistent as well.

1.4 Permitting pause

To test this functional requirement, we must verify that the game will accept the pause input while the game is in a playing state. This can be verified by code inspection and manual bug testing. We can inspect the code to make sure that when the game is in the running, nothing will obstruct the input or the game from changing state to paused. To test this, we tested how the game would respond to a series of pause commands during different times during the game execution, and from this testing, it seems like the pause commands works regardless of other situations.

1.5 Displaying player information above all

To test this functional requirement we must verify that the game always displays accurate user information, as well as verfying that the player info is drawn above all over graphical objects. To do this, we can use code inspection and manual testing. The code itself shows that the game info is displayed above all, as it drawn on top of all asteroids, players and bullets. Manual testing also confirms that this is held, as observation has shown that all objects are drawn under the player information.

1.6 Player firing and alien destruction

To test this functional requirement, we must verify that the game will remove the alien object from the screen once a player bullet collides with it. This can be verified by code inspection and manual testing. The code itself will show that there is a case in the alien object for when it occupies the same area on the screen as the player bullet, and in this case it set's its own state to die and removes itself from the sprites array. Manual testing also confirms this by seeing the alien removed from the screen and game.

1.7 Asteroid collision with player

To test this functional requirement we must verify that when an asteroid collides with a player, both the player and the asteroid is destroyed. Asteroids are perfect circles, so checking collision is easy, as you can simply check the distance between center points, and if it is less than the combined radius of both objects, then you know that they have collided. This has been experimental verified quite intensely as all meaningful player actions use this code quite extensively.

1.8 Asteroid separation

To test this functional requirement we must verify that any asteroid (that is not of the smallest size) that is destroyed breaks into 3 smaller asteroids, with random trajectories. This can be verified with code inspection, as collision checks the size of the asteroid before breaking, as well as manual testing, as in all the player testing we have had on the latest version, we have experienced no errors with asteroid separation.

1.9 Withholding player spawn

To test this functional requirement, we must verify that the player will not respawn if their is an adversary within 100 pixels of the spawn point. This can be verified by code inspection as well as manual testing. The code itself uses a modified version of the collision to test for this, so as that has been extensively tested, we know that the functionality is okay, so to test

the logic, the team inspected the code, and made sure that all flags are correct and placed in the right spots.

1.10 Post-game

To test this functional requirement, we must verify that once the player is dead, the game stops accepting input. Asteroids and aliens will still spawn, making the game more like a screensaver. We can verify that this occurs by code inspection by checking the post game state of the FSM. It shows that player inputs will no longer have

2 Nonfunctional Requirements Evaluation

2.1 Usability

2.1.1 Player test

To test usability, playability and feel of the game, the Staroids team recruited ten outside people to play Staroids and give feedback. Several questions were asked of the test players, particularly about the feel of the game and how it played. The most common feedback we received related to the alien and the appearance of the ship. The alien was annoying and not a challenge to the play testers, so his spawn chance and times were reduced and he was made easier to hit. When the testers faced the new alien, there were much fewer complaints about him. The second large complaint from testers was the appearance of the ship. The ship, alien and asteroids used to be all white filled shapes with a black border. Testers often could not tell the direction the ship was facing, or even which object was their ship. To fix this, the ship was made narrower which helped show the front of the ship and the player ship and alien were made into solid black shapes so they are visible at a glance. The alien bullets and player bullets were more differentiated due to feedback as well.

2.1.2 Browser test

To test Staroids, the Staroids team used all major browsers for development and testing in an effort to notice Staroids bugs and issues that were specific to one browser. None were found.

2.2 Informational

2.2.1 State Change

The Staroids state transitions are programmed into the game, so through code inspection and testing, all state transitions were tested. An error involving the game starting while paused was found through this test.

2.3 Internal

2.3.1 JSDoc Documentation Inspection

Through code inspection it was determined that all functions and classes have been adequately for JSDoc and the comments follow the syntax of JSDoc. All code was also verified that it belonged in the most appropriate module. The only changes brought by this test were the correction of improper JSDoc syntax.

2.3.2 JSDoc Documentation Compilation

Code inspection and referring to the JSDoc Documentation Guides allowed for the JSDoc program to correctly run on the Staroids source code and produce proper documentation. The only changes brought by this test were the correction of improper JSDoc syntax.

2.3.3 Complicated Function Documentation Inspection

An outside programmer was brought in to look at the source code of Staroids and ensured that the code was either self explanatory or sufficient comments were used

2.4 Sound

2.4.1 Brake Sound

Through playing Staroids numerous times and code inspection, it has been verified that the brake sound will only play when the user is applying the brakes. No Staroids edits were made because of the results of the test.

2.4.2 Fire Sound

Through playing Staroids numerous times and code inspection, it has been verified that the fire sound will only play when the user presses the fire button and a bullet is produced by the player's ship. No Staroids edits were made because of the results of the test.

2.4.3 Collision Sound

Through playing Staroids numerous times and code inspection, it has been verified that the collision sound will only play when asteroids collide with the player's and alien's ship. It is also played when an asteroid is destroyed, like expected. No Staroids edits were made because of the results of the test.

2.4.4 Mute Sound

Through playing Staroids and code inspection, the mute functionality was verified to be usable in any state at any time and will turn off game sounds globally. No Staroids edits were made because of the results of the test.

3 Comparison to Existing Implementation

In comparing Staroids to the original open source game it is based on, the first thing that jumps out is the change in styling on the home screen "Press Space to Play" as well as all of the other text in the game has been given a revamped look. It has been changed to more of a bubble font rather than the traditional square-ish letter stylings of it's predecessor. All the concepts, controls and rules from the original however have been maintained, all that has been changed in that sense is just the shapes of things such as the asteroids, and speeds and movements of objects such as player, bullets, asteroids etc. Plus the addition of the player to be able to break and slow there speed down using the down arrow key. This being said both games could be created from the same set of requirements, and should be able to pass the same test plan.

4 Unit Testing

Unit testing nor automated testing was used in the development of Staroids.

5 Changes Due to Testing

So far there have been a couple of major changes due to testing: the alien movement, player movement, player color and alien bullet. The alien changes came about during testing when it was found that the alien was moving in the pattern desired (sine wave) but the amplitude was too small, and did not give the player enough time to get out of the way. This lead to the changes in the alien's speed (slower) as well as its amplitude (higher) so it covered a larger path. The next change was player movement and this change simply involved limiting the players speed, due to the fact that during testing it was found to be volatile to the gameplay by just allowing the player to go unlimitedly fast. The max speed that was settled on was based on a trial and error method to get it to a max speed that seemed natural to all team members. The next change was a change to the actual player ships appearance. This change was made based on third party testing that involved people with no prior experience playing the game, and it was relayed back that the player's ship color being changed to a different color than the background would be helpful by making it easer to see. Another thing that was brought up during this was changing the ships shape slightly from an equilateral triangle to a isosceles so that the head of the ship would be easier to identify. The final change made due to testing was that of the alien bullet, this was another third party test and the feedback given lead the team to changing the alien bullet to look different from the player bullet, based on the confusion relayed by this tester.

6 Automated Testing

In this project it was found that automated testing was not feasible for a number of reasons. One being that the language chosen (JavaScript) does not have a testing framework that supported canvas integration which is how the program draws all of it's spites to the screen. Due to this, in order to do automated testing, it would require the creation of a version of Staroids that does not reference the canvas at all. The second reason for the omission of automated testing was that because Staroids is a game, the most complete, practical and appropriate way to test it is play the game. Creating a Staroids scene that was conducive to automated testing would require an exorbitant amount of specialty code that could be replaced by a user simply setting up those situations themselves. The manual testing also ensured that the game 'felt' nice and that the user interactions were smooth.

7 Trace to Requirements

Test Case	**Requirement**
TC1	F1, F3
TC2	F2, F4
TC3	F2
TC4	F5, F6
TC5	F7
TC6	F9, F10, F12
TC7	F8, F11, F13, F14, F15, F17, F18
TC8	F13, F14, F15
TC9	F11, F16
TC10	NF2, NF5, NF6
TC11	F20
TC12	F19
TC13	F21
TC14	F22
TC15	NF1, NF2, NF4, NF5, NF6, NF7, NF8, NF9, NF10, NF11, NF12, NF19, NF20, NF21, NF22
TC16	F1, NF3, NF6, NF7, NF8, NF9, NF10
TC17	NF5
TC18	NF13, NF14, NF16, NF17, NF18, NF19
TC19	NF13, NF14, NF16
TC20	NF13, NF14, NF16, NF17, NF18, NF19

Table 2: Trace Between Test Cases and Requirement

8 Trace to Modules

Please Refer to Module Guide section 6

9 Code Coverage Metrics

No formal code coverage software was used for this project based on the decision made by the Staroids team to move away from automated testing as explained above and rely strictly on manual testing. However throughout the development of Staroids, and specifically during the MIS portion, the Staroids team went through all individual functions, variables, objects and methods and either documented them if they were used or deleted them if they were not used. This helped increase the percentage of code covered in during testing. Also in conjunction with the MIS, each module method was determined to be a part of any particular test and the correct result of that test would verify that method worked.

Table 1: **Revision History**

Date	**Version**	**Notes**
Nov 23	0.1	Added personal info to document
Nov 23	0.15	Code coverage, Automated testing, Changes due to testing
Nov 23	0.2	Added usability requirement results
Nov 25	0.25	Comparison to existing Implementation
Nov 28	0.3	Finished functional requirements
Nov 28	0.35	Added last non functionals
Nov 28	0.35	Generic edits
Nov 28	0.4	Added sound tests
Dec 5	0.45	Traceability matrix for test cases to requirements