# SE 3XA3: Test Plan
# Staroids

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

October 26, 2018

# Contents

# List of Tables



# List of Figures

Table 1: **Revision History**

| Date | Version | Notes |
|---|---|---|
| Oct 22 | 1.0 | Added Purpose, Test Team, Scope, a couple Acronyms, abbreviations, and symbols |
| Oct 24 | 1.1 | Added Software Description, Overview of Document, Automated Testing Approach |
| Oct 24 | 1.15 | Added functional requirement tests |
| Oct 25 | 1.2 | Added non functional tests |
| Oct 25 | 1.3 | Added Testing tools |
| Oct 25 | 1.35 | Added Testing Schedule |
| Oct 25 | 1.5 | Added Unit Testing Plan and all subsections |
| Oct 25 | 1.55 | Added PoC tests |
| Oct 26 | 1.6 | Reviewed all sections for spelling and finished up unfinished ones |

# 1 General Information

## 1.1 Purpose

This document is designed to show the detailed test plan for the Staroids game. This will include a description of the testing, validation, and verification procedures that will be implemented. All the tests in this document have been created before the final implementation has been completed or tests have actually occured, so it will be the guide followed during the testing phase of the project.

## 1.2 Scope

The scope of the test plan is to provides a basis for testing the functionality of this re-implementation of asteroids. The objective of the Test Plan is to prove all the functional and non functional requirements listed in the SRS document.

## 1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

| **Abbreviation** | **Definition** |
|---|---|
| POC | Proof of Concept |
| SRS | Software Requirements Specification |

## 1.4 Overview of Document

This Test Plan's main goal is to inform on how Staroids will be tested for correctness in its various objectives and requirements. All tools are stated and their use case is explained. All planned test cases that are used to verify the correctness of Staroids with regards to its functional and non-functional requirements are also listed.

Table 3: **Table of Definitions**

| **Term** | **Definition** |
|---|---|
| Functional Testing | Input-Output type of testing apporach known Input, expected Output |
| Static Testing | Just looking at code, no actual execution |
| Dynamic Testing | Testing that requries code execution |
| Structural Testing | A whitebox type of testing approach, cases are derived from internal structure of the software |
| Automated Testing | Testing is handeled by the testing framework (Mocha and Chai) (testing done by software) |
| Manual Testing | Manual individually written test cases (testing done by people) |
| Stress Test | Testing the limits of a system, usually refers to amounts of data the system can handle |

# 2 Plan

## 2.1 Software Description

Staroids is a recreation of the HTML-5 Asteroids project created by Doug McInnes, it itself is a recreation of the Asteroids arcade game. It allows the user to pilot a space ship through a rectangular piece of space with wrapping edges. Cohabiting with the space ship, there are several asteroids and an alien. These entities are considered hostile to the space ship and will damage it if they come into contact. The space ship and alien can defend themselves from any hostiles by firing at them with a laser bullet. This will damage anything (except the shooter) that the bullet comes into contact with. Staroids allows the user to have 3 lives and keeps a running score for the current game that is based off of how many asteroids and aliens have been destroyed.

## 2.2 Test Team

The test team for this project consists of the following members who are each responsible for writing and executing tests for modules later to be specified:

- Moziah San Vicente
- Eoin Lynagh
- Jason Nagy

## 2.3 Automated Testing Approach

Automated tests are to be used for all game situations where the situation result is expected to be the same for each one. Since the result is always true, these tests can be quickly run after every Staroids edit to ensure that no game functionalities have been broken by the edits. These tests are not done automatically at game load for several reasons. Firstly, if there is an assertation error, it would be meaningless to the user and there would be no action done on the user's part. Additionally there would be time added to the startup with no benefit to the user. Therefore, all major versions of Staroids will run through the test cases before they are pushed as a final Staroids build.

## 2.4 Testing Tools

The tools that will be used for testing are Mocha & Chai and JSCover. Mocha & Chai is the unit testing framework and assertion library that will be used to facilitate our test cases outlined in this document. This will be created once consisting of all unit tests, and then can be re-run automatically after each major revision of the software in turn automateing Staroids unit testing. JSCover is the codecoverage tool that will be used to check how much of the Staroids game code is being executed when we the test suite is being run. The only other things that may be considered as "testing tools" could be the internet browsers that the team are going to test the game on: Safari, Edge, Firefox and Chrome.

## 2.5 Testing Schedule

See Gantt Chart

# 3 System Test Description

## 3.1 Tests for Functional Requirements

### 3.1.1 States

**Pre-game State Tests**

1. Loading into pre-game

   Type: Functional dynamic manual test

   Initial State: Closed

   Input: Opening file

   Output: In pre-game state

   How test will be performed: On game start, the game should be in the pre-game state. This can be acomplished by an assertation as part of the game set-up that states that the current game state is the pre-game.

2. Starting game

   Type: Functional dynamic manual

   Initial State: Pre-game state

   Input: Spacebar pressed

   Output: In playing state

   How test will be performed: Once the pre-game screen is loaded, the spacebar registers as pressed and Staroids should enter the playing state. Staroids is allowed to go through a few intermediate states but must end on the playing state within one second.

3. Prohibit game start on pause

   Type: Functional dynamic automatic test

   Initial State: Pre-game state

   Input: Pause button and spacebar pressed

   Output: Paused state

How test will be performed: In the pre-game state, the pause button is registered as pressed and then the game enters the paused state. The paused state will prohibit progression from the pre-game state to the playing state when the spacebar is pressed. Upon registering the spacebar again, the game shall return to the pre-game state.

**Playing State Tests**

1. Permitting pause

   Type: Functional dynamic automatic test

   Initial State: Playing State

   Input: Pause button pressed

   Output: Paused state

   How test will be performed: When the spacebar is registered as depressed, the game shall enter the paused state. Upon registering the spacebar again, the game shall return to the playing state.

2. Displaying player information above all

   Type: Structural static manual test

   Initial State: None

   Input: None

   Output: None

   How test will be performed: For the player information to be displayed above all other sprites, those sprites need to be drawn last. Inspecting the code will ensure that the player, score and lives are drawn above all other sprites.

3. Player firing and alien destruction

   Type: Functional dynamic automatic test

   Initial State: Playing State

   Input: Player object, alien object, and fire button pressed

Output: Bullet generation, bullet movement, alien destruction

How test will be performed: By placing the player and alien opposite of each other, the spacebar should generate a bullet that originates from the tip of the player ship. This bullet should then proceed forward and impact the alien. The bullet speed is derrived from the player ship speed plus a constant. The alien death should trigger, which can set a detectable flag. The bullet must only move in one direction.

4. Asteroid collision with player

   Type: Functional dynamic automatic test

   Initial State: Playing State

   Input: Player object and asteroid object

   Output: Player loss of life, player location reset, asteroid destruction, asteroid separation, Post-game state (if number of lives == 0)

   How test will be performed: Set up the game to have a player facing a single asteroid moving towards the player. When the asteroid collides with the player, both the player and asteroid trigger their death functions, setting flags to register their deaths. On player death, the lives count is decresed which can be compared to the initial value. The asteroid child array will also increase which can once again be compared to the initial value.

5. Asteroid separation

   Type: Functional dynamic automatic test

   Initial State: Playing State

   Input: Asteroid object

   Output: Production of child asteroids

   How test will be performed: Starting with a large asteroid, a signal is sent to the asteroid to trigger its death response. The asteroid itself should become disabled and should spawn 3 children asteroids. By transmitting another signal, those children asteroids should disable themselves and then create 3 of their own children. One last signal is used to destroy those children. They should not produce any children.

The original large asteroid object should detect that all of its children are dead and those children's children are dead. The asteroid should broadcast that it has been completely destroyed.

6. Witholding player spawn

   Type: Functional dynamic automatic test

   Initial State: Pre-game state

   Input: Spacebar pressed

   Output: Delayed player spawning

   How test will be performed: When the player changes from the pre-game state to the playing state, the player shall not spawn if an asteroid is within 100 pixels. If there is, the game shall wait for the asteroid to pass and then spawn the player

**Post-game State Tests**

1. Post-game

   Type: Functional dynamic manual test

   Initial State: Post-game State

   Input: The restart game button "R" is pressed

   Output: Pre-game State

   How test will be performed: Set up the game in the post-game state and then press the "R" key and check if the new state after the key has been pressed is the pre-game state. It will also be tested that if any other keys are pressed no changes to the game state or game will be performed.

## 3.2 Tests for Nonfunctional Requirements

### 3.2.1 Usability and Style

**Player Test**

1. Player test

   Type: Functional dynamic manual test

   Initial State: pre-game state

   Input/Condition: An outside person

   Output/Result: The outsider playing through Staroids a few times

   How test will be performed: A friend or colleague will open Staroids with only the knowledge of what they can do in the game. After a few plays, the person is informed about all the functions of Staroids and how to perform them. The person then proceeds to play a few more rounds. After they are finished playing, they will be asked some questions with regards to the game's usability and playability.

2. Browsers

   Type: Functional dynamic manual test

   Initial State: Closed

   Input/Condition: A developer and all major web browsers

   Output/Result: An operational Staroids instance

   How test will be performed: Staroids is to run on any major browser, so a developer will open staroids using all major browsers and play through a few games to ensure that the game is functioning as expected. There should be no functional differences between the browsers, but performance is allowed to vary. The developer may judge whether a possible performance decrease is unacceptable.

**Informational Tests**

1. Player information

   Type: Functional dynamic manual test

   Initial State: pre-game state

   Input/Condition: A person to advance the game states

Output/Result: Ensurance that any relevant game prompts are shown to the user

How test will be performed: A member of the Staroids team will start Staroids and advance through the game states manually to ensure that all states have on screen prompts to inform the user about their input options. In the pre-game state, the ship controls, pause button and start button must all be shown to the user on screen. The playing state does not need to show anything. The post-game state must show the restart button and lastly the pause state must show the unpause button.

### 3.2.2 Internal Requirements

**Documentation**

1. JSDoc Documentation Inspection

   Type: Structural static manual test

   Initial State: Closed game

   Input/Condition: Staroids source code and an inspector

   Output/Result: Verification of proper JSDoc documentation

   How test will be performed: To ensure all functions and classes are properly commented in accordance to JSDoc, a developer of Staroids will need to inspect the Staroids source code to ensure that all functions are documented and documented correctly. The developer will also ensure that all the modules contain only elements that are related to the game's operation.

2. JSDoc Documentation Compilation

   Type: Structural static automatic test

   Initial State: Closed game

   Input/Condition: JSDoc executable and the Staroids modules

   Output/Result: Properly created JSDoc documentation

How test will be performed: In addition to an inspector looking over the JSDoc comments, the executable should also compile proper documentation based on the JSDoc commenting within the source code.

3. Complicated Function Documentation Inspection

   Type: Structural static manual test

   Initial State: Closed game

   Input/Condition: Staroids source code and an inspector

   Output/Result: Verification of commented code

   How test will be performed: Similar to the JSDoc commenting inspection, all complicated pieces of code must be commented to help a reader understand what a particular piece of code is doing. An external programmer can also be brought in to ensure that the code sufficiently commented to allow an outside user to understand what is occuring.

## 3.3 Traceability Between Test Cases and Requirements

# 4 Tests for Proof of Concept

## 4.1 Usability

**Browser test**

1. Playability on all major browsers

   Type: Functional dynamic manual test

   Initial State: None (Game closed)

   Input: Open game with browser

   Output: Game runs in pregame state on browser

   How test will be performed: The test will be performed by opening the game with the specified browsers (Edge, Firefox, Safari, Chrome) and looking to see if the pregame screen shows up.

2. Player test

   Type: Functional dynamic manual test

   Initial State: pre-game state

   Input/Condition: An outside person

   Output/Result: The outsider playing through Staroids a few times

   How test will be performed: A friend or colleague will open Staroids with only the knowledge of what they can do in the game. After a few plays, the person is informed about all the functions of Staroids and how to perform them. The person then proceeds to play a few more rounds. After they are finished playing, they will be asked some questions with regards to the game's usability and playability.

## 4.2 Sound Tests

**Sounds**

1. Brake sound

   Type: Functional dynamic manual test

   Initial State: Playing

   Input: Brake button pressed

   Output: Brake sound plays through speakers

   How test will be performed: While the game is in the player state the down key will be pushed and it will be monitored to see if the break sound is heard.

2. Fire sound

   Type: Functional dynamic manual test

   Initial State: Playing

   Input: Fire button pressed

   Output: Fire sound through speakers

How test will be performed: While the game is in the player state the spacebar key will be pushed and it will be monitored to see if the fire sound is heard.

3. Collision sound

   Type: Functional dynamic manual test

   Initial State: Playing

   Input: Collision between player and asteroid

   Output: Collision sound will be played through speakers

   How test will be performed: While the game is in the player statea collision will be simulated and monitored to see if the fire sound is heard.

4. Mute sound

   Type: Functional dynamic manual test

   Initial State: Playing with mute on

   Input: All sound actions will be simulated

   Output: Animations still show on screen but no sounds acompanie them

   How test will be performed: The three sound actions above will be simulated and the volume of the computer will be monitured to make sure that no sounds are being produced.

# 5 Comparison to Existing Implementation

# 6 Unit Testing Plan

Mocha and Chai are to be used in Staroids unit testing. The testing suite will be a seperate file that runs over top of all the other files and controls what occurs on the screen. All of the games modules are complete, so there is no need for stubs or drivers for testing. The coverage method will be a combination of Mocha's metric as well as the Staroids team ensuring that each method has been tested at least once.

## 6.1 Unit testing of internal functions

To test Staroids functions the team will use test cases on all functions, having at least one test per function, perhaps more. These tests will be created until there is full coverage over all the internal functions. Testing will also run in developer mode, rather than having tests automatically run every time the game is exectuted. The developer mode will have to be enabled from the developer side, and does not require any user inputs (besides activing the mode).

## 6.2 Unit testing of output files

Staroids will not create any output files, and as such, no testing on output files will be nessescary.

# 7 Appendix

## 7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. **Constants**
FPS = 30; The curent frames per second
SHIP_SIZE = 30; The ship size in pixels
TURN_SPEED = 180; Player turn speed in degrees per second
SHIP_THRUST = .2; Player thrust power in pixels per second squared
SHIP_BRAKE = 0.98; player airbrake power (¡0.9 = full stop 1 = no brake)
MIN_SPEED = 0.1; minimum speed
MAX_ACC = 2; maximum ship acceleration
MAX_SPEED = 20; Maximum ship speed (velocity)
CVS_WIDTH = 500; canvas width
CVS_HEIGHT = 400; canvas height
BULLET_EXTRA = 5; Extra velocity on bullet on top of ship's velocity
KILLABLE = true; Testing invulnerability
MAX_ASTEROIDS = 2; Maximum amount of asteroids
TEST = false; experimental features

## 7.2 Usability Survey Questions

1. Do the controls feel comfortable? Do they place the hand in an awkward position?
2. Do the controls make sense?
3. Do the buttons used make sense given their function? Are they relatively consistant with other programs of similar nature you have used?
4. Is the game too difficult or easy? If so, how could this be corrected?
5. Is the Staroids art style appealing? Would colour improve the experience?
6. Is the text in the game readable?
7. Did the game run smoothly? Was there any screen tearing or stuttering?

8. Were there any symbols or graphics that you recognize?