SE 3XA3: Software Requirements Specification
Staroids

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

December 5, 2018

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
Sept 26	0.1	Added team and project info
Sept 28	0.11	Divided document work
Sept 28	0.12	Added basis of functional requirements
Oct 2	0.13	Added stakeholders
Oct 2	0.135	Added Tasks with link to Gantt Chart
Oct 3	0.14	Added some non functional requirements, off the shelf solutions and documentation
Oct 3	0.145	Added Purpose of Project, Naming Conventions and Terminology and Risks
Oct 3	0.15	Added non functional requirements and basis of context of the work
Oct 4	0.16	Added Mandated Constraints, Relevant Facts and Assumptions, Work Partitioning, Open Issues, Tasks, Costs
Oct 5	0.17	Added Context of the Work
Oct 6	0.18	Added Individual Product Use Case
Nov 7	0.19	Added sound functional requirements
Nov 7	0.2	Modified max bullet requirement
Nov 28	0.21	Spelling corrections
Dec 5	0.3	Numbered requirements

1 Project Drivers

1.1 The Purpose of the Project

The purpose of the project is to create a fun and dynamic interactive web game, based on the classic 80s arcade game Asteroids, complete with keyboard controls and sound. This game will be designed to, hold the attention of the user and provide hours of entertainment as they work towards beating previous high scores. The game will use HTML5-Asteroids by Doug McInnes as a guide.

1.2 The Stakeholders

For the development of Staroids, there are some key shareholders that have impact on what decisions are made and in effect, have sway in the outcome of the project. The stakeholders primary role is to ensure that Staroids is developed properly and that all teams involved in development are satisfied with the project. The main stakeholder in Staroids are the developers as well as the client and the customer. In the case of the Staroids project, the Staroids team are the developers. The clients of the project are both the original HTML5 Asteroids developer and the Staroids team, and lastly the customer of the project is once again the developers and any online web game players.

1.2.1 The Client

Staroids is developed for the original creator of HTML5 Asteroids with the purpose of using the proper implementation and documentation techniques. The primary concern of the original creator is Staroids' faithfulness to the original as well as the adaptations of any edits that the creator wanted to make but did not get the chance to do. The Staroids team also takes the part of the client because the team very much wants to complete this project for themselves. It offers a chance to program in a language that is new and tackle a problem that the team has not attempted yet. The developer's main concern is that the project has a straightforward implementation method.

1.2.2 The Customers

The customers of the project are the developers again and web game users. The developers are the customers as this project is also being created for their sake as a challenge in JavaScript. As such, the developers concern as a customer is that the project provides an insightful and valuable learning opportunity. Web game players are also clients as they will also consume the project once it has been created. Their concerns are where and how the game will be played and how easy it is to get running on someone's machine.

1.2.3 Other Stakeholders

Some other stakeholders that could impact the project are advertisers and web game companies. Advertisers may look to spread the availability of the Staroids project to new users, so they require the project to be unique, special or different in some way so that they can advertise to others and bring the Staroids project into the attention of new users. Web game companies may also be stakeholders because they may look to advertise or host Staroids on their web sites. In terms of advertising, they would be similar to the advertisers, but for hosting, the companies will need the project to meet a set of technical requirements. The web game companies may need the project to be in a certain format, only use generic libraries or be written in a certain manner.

1.3 Mandated Constraints

There are three main mandated constraints for this project; Firstly, the project must be completed by the project due date of December 6, 2018 and all of the project deliverables as stated and outlined by the course instructor are completed on time. Secondly the final project must be a reimplementation of the original asteroids, an open source game found on Github, so therefore it will need to have a similar design and functionality to the original. Lastly, the implementation will be constrained by the users computer, because although the game can be made to run at as high of a frame rate as possible, the user will only be able to experience what their computer and monitor can handle.

1.4 Naming Conventions and Terminology

As described in the Development Plan, our main naming convention is to make use of camel case. This has a distinctive look, as well as being easy to type. To name files, we will be descriptive in the name of the file, making sure that it is obvious to all developers what file contains what information, to name variables, we will also use camel case, and variable names will be a combination of attribute and object, so for example, if we need to store the velocity of the ship, the variable will be named velocityShip.

1.5 Relevant Facts and Assumptions

There are many relevant facts and assumptions regarding this product. Most of these assumption are made on what it is thought that the user will do when playing the Staroids game. It is assumed the user will follow the on screen prompts and instructions. As well as know that the pause sign on the game state screen is to pause the game, and the play sign on the pause screen is to return to playing the game. Another assumption is that the user will recognize the symbols at the top of the screen shaped like spaceships are their number of remaining lives, and that the number increasing as they keep playing in the top right corner is their score. The main relevant fact to this project is the original implementation had 1218 lines of code, and is able to run on any web browser as well as on an iPad.

2 Functional Requirements

2.1 The Scope of the Work and the Product

The scope of the project is to duplicate the original asteroids project that Staroids is basing their work on, but to document the project properly and follow the principles we were taught in 3RA3. This includes the creation of a JavaScript game, an index.html file that are can be opened by any browser with a significant market share, as well as another JavaScript file for the

typeface.

2.1.1 The Context of the Work

In the original project the code of HTML5-Asteroids is a convoluted mess of functions and statements making it very difficult for a programmer to understand and work on if they desired to make edits. If the game were designed more in accordance to software design principles, the game would be better to both the user and other developers. While functional, HTML5-Asteroids lacks the process structure and documentation of a large software project. As a result, edits and revisions are much harder to implement. The game also does not follow the software engineering principles of modularity or information hiding. The goal of Staroids is to recreate the original HTML5-Asteroids while following the proper documentation guidelines and modularization techniques.

2.1.2 Work Partitioning

Table 2: Work Partitioning

Event Name	Input	Output	Summary
Staroids Creation	Developer code	Internet Browser	Recreating a web based game implementation that works on various browsers.
Staroids Audio	Wav File	Speaker	Adding more sounds into the game.
Staroids Sprite	Developer graphics and code	Internet Browser	Adding more visuals to the objects in the game.
Staroids Hits	Developer code	Internet Browser	Creating hit detection for when the spaceship shoots an asteroid.
Staroids Collisions	Developer code	Internet Browser	Creating collision detection for when the spaceship hits an asteroid.
Staroids Final Collision	Developer code	Internet Browser	Creating an end screen once a max number of collisions has been met and final collision occurs.

2.1.3 Individual Product Use Cases

This product is primarily made to be used by people looking to enjoy retro video games. 80s arcade games on modern platforms have been making a large resurgence in recent years, so our product can be used by anyone looking for that kind of experience. Another use case more general entertainment, as a JavaScript file, the code itself can be used as an educational experience, but the game itself can have many more goals to obtain, like highest score or fastest time.

2.2 Functional Requirements

1. Run on Google Chrome, Mozilla Firefox, Microsoft Edge and Apple Safari browsers.

2. The game shall contain pre-game, playing, post-game, and paused states.

3. When initially ran, the pre-game screen shall show first.

4. On the pre-game screen, if the play button is pressed, the playing screen shall show.

5. On the press of the pause button during playing, the pause state shall show.

6. On the press of the pause button while paused, the playing state shall appear again.

7. The playing screen shall always display the player character, score and lives.

8. Every time the player character is hit by an enemy, the lives count shall decrease by one.

9. When the fire button is pressed, the player character will fire a projectile.

10. If a projectile hits an enemy, the enemy will be removed, that enemy's death action will occur and the score will be incremented.

11. If zero lives remain and the player character is hit, the game shall enter the post game screen.

12. After a set amount of time, an enemy character shall appear. He shall shoot at the player character.

13. If a large asteroid is hit, it will separate into 3 medium asteroids and the score shall be incremented.

14. If a medium asteroid is hit, it will separate into 3 small asteroids and the score shall be incremented.

15. If a small asteroid is hit, it will be removed and the score shall be incremented.

16. No player character can spawn if there is an asteroid near by.

17. Bullets must be removed after a set amount of time on screen
18. Projectiles move relative to player speeds.
19. If the game is not muted the game will play a sound when the user fires.
20. If the game is not muted the game will play a sound when the alien fires.
21. If the game is not muted the game will play a sound when the user loses a life.
22. If the game is not muted the game will play a sound when an asteroid is destroyed.
23. If the game is muted no sound will play.

3 Non-functional Requirements

3.1 Look and Feel Requirements

1. Staroids should have visually appealing graphics.
2. Staroids should have intuitive controls.
3. The player character should move fluently.

3.2 Usability and Humanity Requirements

1. On the pre-game screen, game controls should be shown.
2. In pre-game, post-game and pause states, relevant prompts should be shown.
3. All text should be written in English in a readably sized font.

3.3 Performance Requirements

1. The playing state should not stutter or freeze.
2. The project should always run above 60 frames per second.
3. Staroids should load to the pre-game state in a few seconds or less.
4. Game states should change immediately after the relevant button is pressed.

3.4 Operational and Environmental Requirements

1. Users should not need to install any external software to run Staroids as long as there is a web browser present.
2. Staroids should not use so many computer resources as to impact other running programs.

3.5 Maintainability and Support Requirements

1. Staroids should be sufficiently modularized so that edits to a specific aspect of the game are quick.
2. Documentation should be kept up to date.
3. Each internal function, object and complicated component should be documented and commented to allow an outsider to understand that component.

3.6 Security Requirements

1. Staroids should not have an accessor methods to any of the internal variables or game state objects and variables.
2. Staroids should not transmit any malicious code or programs to a user's machine
3. Staroids should not take any personal information from the user.

3.7 Cultural Requirements

1. Staroids should not use any symbols that may be considered offensive or rude in its target demographics.

3.8 Legal Requirements

1. Staroids should not use logos, characters or symbols that are owned by or licensed to people other than the creators of Staroids.

3.9 Health and Safety Requirements

1. The controls of Staroids should not put strain on the hand.
2. No screens containing flashing colours should be shown.

4 Project Issues

4.1 Open Issues

As updates to the hardware and software of computers occur and evolve them, it is not known how the project will react to these changes whether it will work fine, with bugs, or become unplayable. Another issue is that the project has not been tested on all of the four main web browsers that is intended to run on yet and as well if any new web browsers are created and reach main stream popularity in the future it will not be known whether it will work or not. Also since the project is a web based game any issues with the web server hosting it will also affect the game, such as the website receiving too much traffic or being taken down.

4.2 Off-the-Shelf Solutions

There are many implementations of asteroids in a variety of languages that could offer inspiration to Staroids. The core game that underlines Staroids and the others is the same, but implementation methods and add-ons are what separates them. Some games offer a more realistic physics simulation while others focus on the visual experience. The other implementations also

are written in different languages for different platforms, so if a user cannot run Staroids with its JavaScript and HTML implementation, those users have alternatives. Input methods also vary amongst the implementations.

4.3 New Problems

Making a game with similar look as a copyrighted game may result in legal trouble, so the Staroids team has to be careful not to infringe on any rights.

4.4 Tasks

The steps to deliver the project are all covered in the dynamic Gantt chart Gantt Chart that is being used and updated throughout the project. It covers all tasks and deliverables throughout the project with a breakdown of what goes into each as well as who is working on it, plus the amount of their efforts that are going into it. The Development phases of the project are also shown on their and are color coded to easily distinguish the different ones.

4.5 Migration to the New Product

None

4.6 Risks

As this is an online game, there is very little physical risk or ability to cause injury. The software itself is not running any processes that could be vital to the safety of a person. However, they may be other risks, not involving the safety of people. For example, if our code is poorly optimized, it could result in slowdown or overheating of the computer, which can cause problems.

4.7 Costs

The three main types of costs associated with the project are monetary, storage and CPU. The goal is for the project have a monetary cost of $0 CAD. In

terms of storage take up a minimal amount of space ideally either less than the original implementation, or if greater in the maximum range of double the originals size. Finally the CPU usage is not very concerning for this project due to how small and simple the project is as well as it being run through a web browser. Any modern computer with a web browser installed will be able to easily run Staroids.

4.8 User Documentation and Training

Staroids is to be documented using the JSDoc 3 documentation solution. JSDoc 3 allows for inline commenting in JavaScript documents to be converted into a documentation website. Since JSDoc documentation is written in the JavaScript source code, the Staroids team can modify it simultaneously with any edits to any function. This allows for completely up to date documentation while making it easier on the developers. All functions, classes and states are to be documented using JSDoc as soon as they are created, and any edits made to the them must be reflected in the JSDoc comments. A special documentation file will be created for training people on how to use Staroids. The document will have a table of what each keyboard key will do and how they can be used. There will also be a short explanation of the rules and goals of the game.

4.9 Waiting Room

The original project is implemented in a black and white palette which is not very visually appealing. A possible improvement would be to colourize all game objects. This would make the game more visually appealing and more people would play Staroids. Another possible improvement over the original would be the implementation of a more expansive sound library for in-game actions. The old implementations only has two sounds.

4.10 Ideas for Solutions

Through JavaScript and HTML canvas, there are several colourizing options. The game objects could be coloured dynamically or sprites could be used for

characters. The sprites would enable quick and effective graphical changes, but they would also require more work to set up and collisions may not be accurate to the sprites. In regards to sounds, more sound clips would need to be collected for use and then implemented into the game. The implementation shouldn't be difficult. New sounds could be added for the thrusters, shooting, explosions, post-game screen as well as background music.

References

5 Appendix

This section has been added to the Volere template. This is where you can place additional information.

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.