Salvage Mobile Game

Software Requirements Specification

Version 1.1
January 28, 2014

Prepared by:
Joseph ALLISON
Jon ANTOLIN
DJ BALANE (Graphic Designer)
Aaron FLORES
Ryan HAGLUND (Assistant Manager)
Jared JOHNSON
Gerardo PARKER (Project Manager)
Marques STONE

Advisor:
Dr. Arturo I CONCEPCION

Table of Contents

1 Introduction

1.1 Purpose

This software requirements specification is intended to provide a complete and working description of the Salvage mobile game-application. This document provides a detailed overview and system requirements of the software. This document's expected audience is Dr. Concepcion, and Gerardo Parker the Game Designer; as well as the software engineering team, graphics designers, and anyone wishing to review the game as a whole.

1.2 Scope of the Project

The software will be designed for mobile devices, and is intended for anyone with a smartphone that has either IOS or Android operating system.

The goal of this software is to provide a space game is both enjoyable and repeatable.

In the first iteration the player will be able to:

- Roam in a procedurally generated space
- Shoot down enemy ships
- Be hit by enemy chaser
- Destroy asteroids

In this class section the player will not be able to:

- Land on planets
- Customize their own ships
- Load different worlds
- Collect minerals
- Use the Inventory system
- Go into Store menu

1.3 Definitions, Acronyms, and Abbreviations

Android
A mobile phone operating system developed by Google and several other partners.

Apple devices
Devices distributed by Apple Inc., which are the iPhone, iPod, and IPad.

C#
The programming language the video game will be using.

Draw Calls
The amount of materials being drawn to the screen, materials can be batched, or rather displaced multiple times, to reduce the number of draw calls while still looking the same.

IOS
A mobile phone operating system, which is distributed by Apple Inc. solely for Apple mobile devices.

MB Megabyte
A unit of computer memory typically rounded to be equivalent to 1000 bytes.

Mobile Device
A device used by the consumer that does not require the consumer to be plugged into a power supply, and can move from location to location while the device is still on, such as a phone.

MOGA Controller
A third party video game controller, used as an interface to add functionality to the button-less iPhone and Android devices.

MonoDevelop
The development environment used to write and edit code for the game.

Procedural Generation

The process of generating an image, or environment through a series of algorithms that allow for minimal level design and can be formulated to be a random generation each time.

RAM Random Access Memory

A type of computer storage that is typically used for faster access, it is also "volatile" memory, meaning that anything stored in RAM will be lost when the application closes, or the device turns off.

Unity3D

3-Dimensional game engine used for developing video games. Has the advantage of being one of the most multiplatform engines allowing for development across multiple platforms with the utmost ease.

XML

A file format that is typically used for saving information to memory, and can be modified to have any type of structure, otherwise known as a schema.

1.4 References

IEEE STD 830-1998 Software Requirements Specifications
Salvage Game Design Document (version 1.1)
CSUSB Student Advising Mobile Application SRS (version 2.2)

1.5 Overview

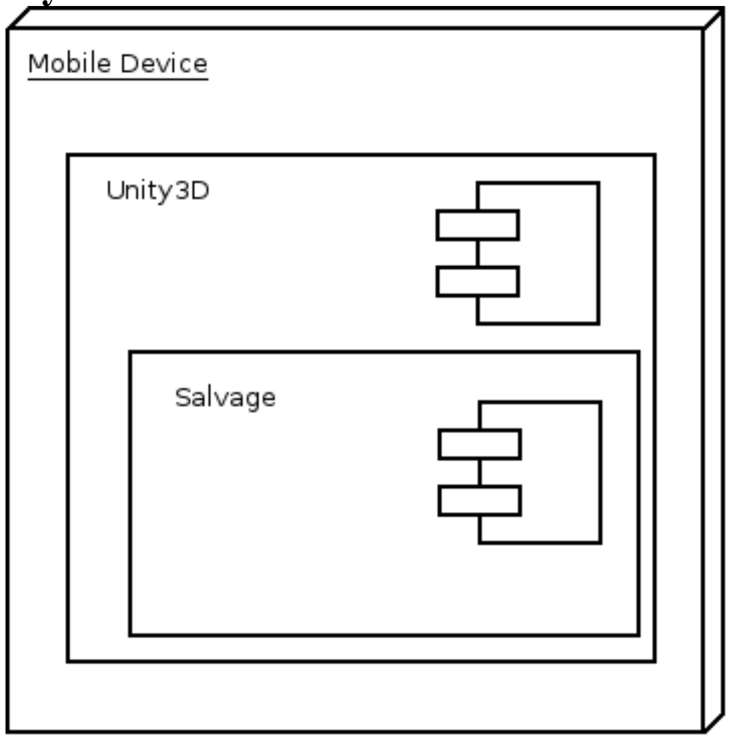
This document is split into two different sections; the first section is the SRS, which is a detailed overview of the User Interfaces, dependencies, functionalities, and requirements that the application will be using; the second section is a Game Design Document, which will be the original design of the game and is used as a basis for this SRS.

2 Overall Description

2.1 Product Perspectives

The following features will be implemented in the first iteration of the game:
Only main menus and in-game menu
Ability to create and save a generated world
Available on both Apple Devices and Android Devices with at least 2.3.3 Android OS.
Procedurally generating worlds

2.1.1 System Interfaces



2.1.2 User Interfaces

We will be making a whole user interface that integrates seamlessly with the theme of the game.

Main Menu
Options Menu
Credits Page
In-Game Menu
Load Game Menu
Store Menu

Please see section 3 .2 User Interfaces for a detailed look and explanation of each menu.

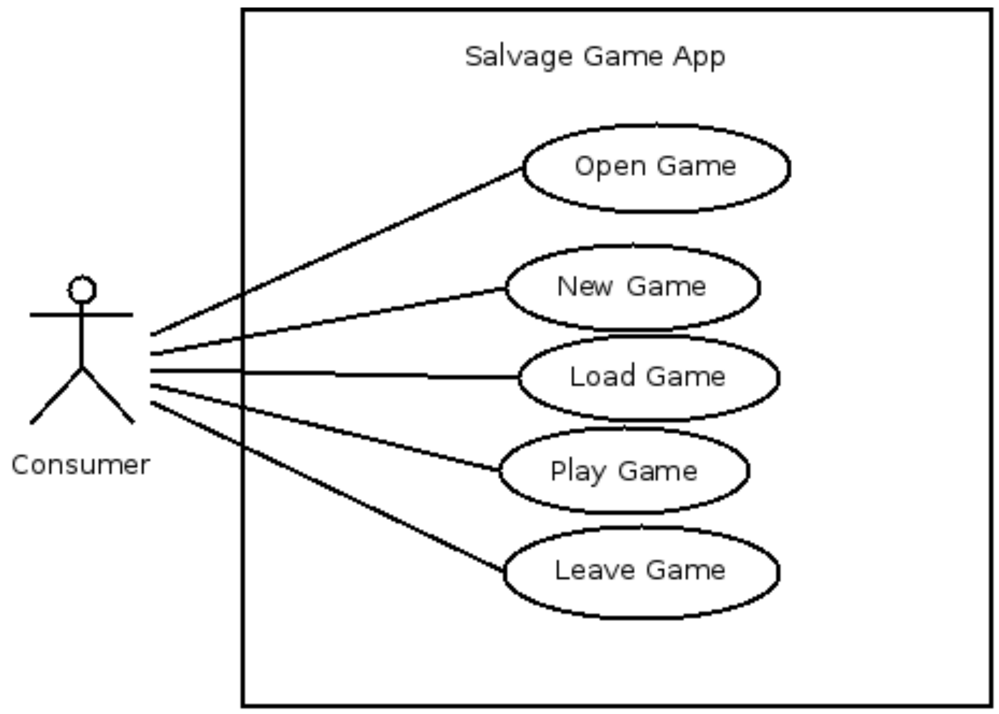
The game will not be using the CSUSB color scheme, except for the splash screen, or possible customizations of a ship.

2.1.3 Communication Interfaces

2.1.4 Memory

The application will not exceed 256 MB of RAM
The application will not exceed 100 MB of disk space
Either Android 2.3.3 or IOS 7 operating systems
Available on Google Play and Apple App Store
Can be downloaded over 3G, or 4G, or any WI-FI connection

2.2 Use Cases



2.3 User Characteristics

A typical user is expected to be able to:

Play the game online/offline
Is intended to be above the age of 10
Casual Gamer

2.4 Constraints

2.4.1 Operating System

The usage of this application will require a mobile device, such as a tablet or phone, with an operating system of Android 2.3.3+ or IOS 6.0+

2.4.2 Connections

The user is not required to have a connection to play the game but an initial connection via WIFI, 3G or 4G is necessary to download the game.

2.4.3 Platforms

The application must be written using Unity in order to support as many platforms as possible with the simplicity of buying a new license.

2.4.4 Content Creation

The player will be able to generate a new world with the input of a seed. As well as new content being streamed to them via updates through apple.

2.4.5 Accessibility

Will not have colorblind support initially but can be added in iteration 2.

2.5 Assumptions and Dependencies

We will require:
Unity Pro 4.3.3(latest version at the time of writing)
MonoDevelop (installed with Unity)
C#

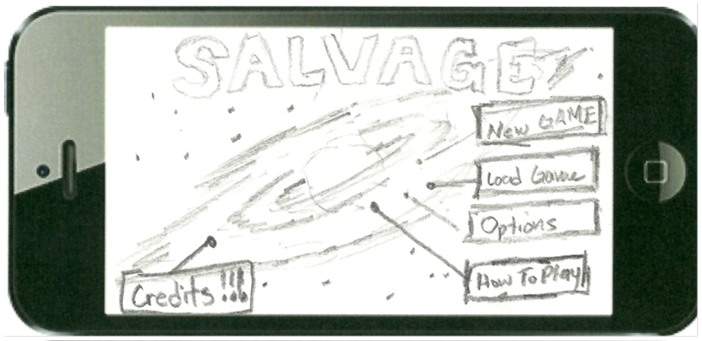
Android 2.3.3+
IPad IOS 6.0+

3 Specific Requirements

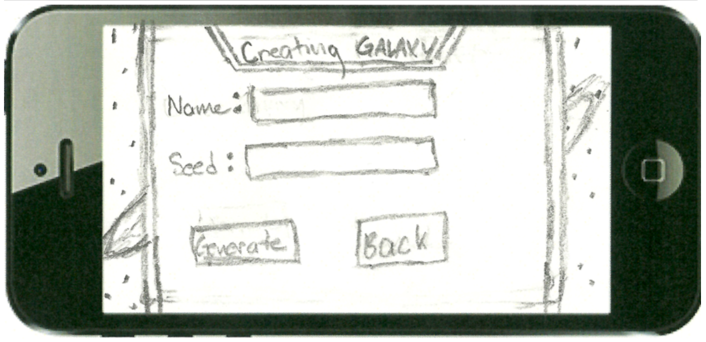
3.1 External Interface Requirements

No external interfaces are required, but support for MOGA controllers or other third party controllers are an option, but are not planned for in this iteration.

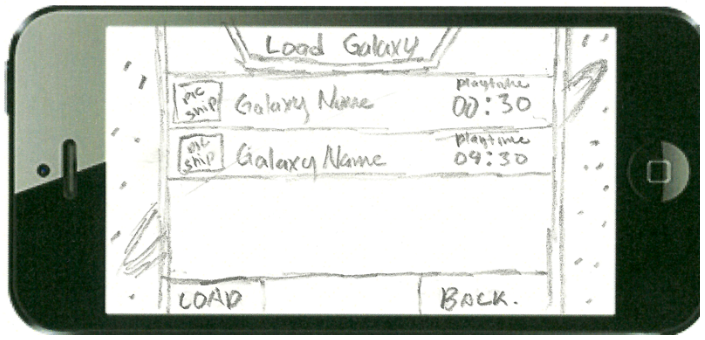
3.2 User Interfaces



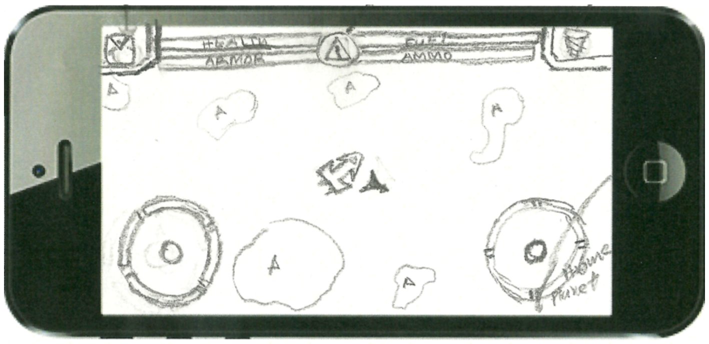
This is the main menu of the game. Here the player will be able to choose which other menu to go to



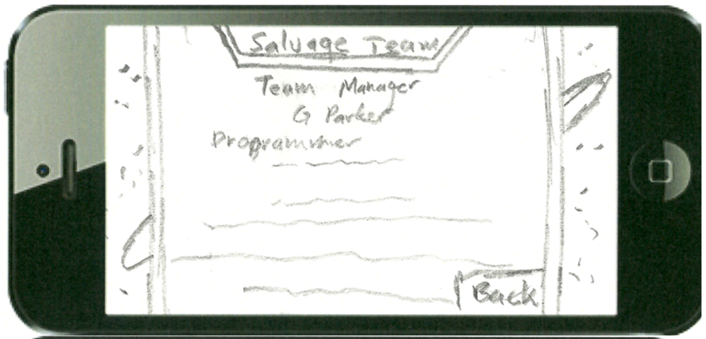
This is the new game menu. Here the player will be able to name their world and create a random world with that name. This will lead to In-Game Menu if they proceed or go back to the main menu.



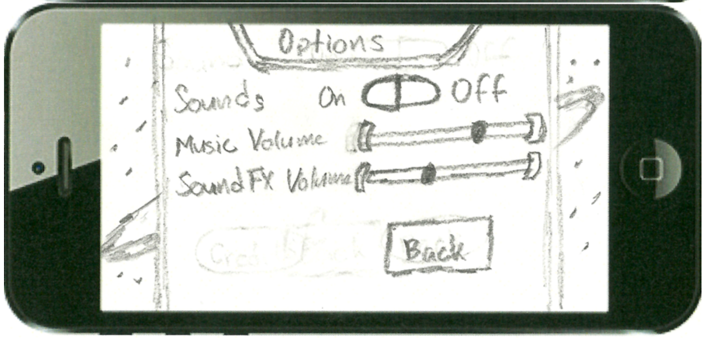
This is the load game menu. Here the player loads previously generated worlds, and player information to use. This will lead to the In-Game Menu or the Main menu



This is the In-Game menu. Here the player will control their ship, fly around, fight enemies, collect minerals and items to trade in for new ships at the store menu



This is the credits page. Here the player will be able to see the development team of Salvage. This menu will have a scrolling wall of text and will only lead to the main-menu



This is the options menu. Here the player can adjust their settings to make the game better suited for them. Will be available in game as well from the main menu.

The store ui has been ommitted from this document as it is not intended to be in this iteration and so planning has not been done for this at the momement.

3.3 Hardware Interfaces

A device running IOS 6.0+ or android 2.3.3+ will be required to run the application. Both OS's will be developed in parallel with one another.

3.4 Functional Requirements

The following functional requirements shall be in the game:
The player will be able to

Create a world : New Game Menu
Travel within the world : In-Game Menu
Create a ship : Store Menu
Customize a ship : Store Menu
Mine for minerals and materials : In-Game Menu
Save and load information : In-Game Menu, Load Game Menu

3.5 Performance Requirements

The software must remain under 256 MB of RAM.

The total draw calls must not exceed 100.

Input detection from the user will be aimed to be less than 20 ms (milliseconds), but this number can vary due to the rendering of the scene and how many object are on the scene, object pooling can decrease the cost in performance but it will vary on what the player will be doing.

3.6 Design Constraints

We will maintain an app size under 100 MB.

Graphics will be made to scale properly between low-resolution mobile devices, and high-resolution devices from both tablets and phones.

3.7 Security Requirements

This application will only be using xml encryption to secure player information

Encrypting xml

All data that is being saved in the open will be encrypted so that copying saves is harder, each person will have their name saved to the save file so that files cannot be shared across users.

3.8 Document Approval

This document must be approved by:

Gerardo Parker
Project Manager
Game Designer
Client

Dr. Arturo Concepcion
CEO

4 Appendix

SALVAGE

Table of Contents

Genre

2D Space-Adventure

System Requirements

Primary Platforms:

Apple Mobile platform IOS 7.0+

Android Mobile platform 4.0+

Secondary Platforms:

Apple/Mac/Linux platform with a GPU that supports OpenGL 2.0+ and .Net 2.0

Game Mechanics

The player will be "roaming" around a procedurally generated "Space" envirnment, which they must collect pieces of "garbage" or "recyclables" that will give the player "money" in order to increase the power/speed/fuel/weapons of the current ship that they have. Along the way the player will encounter multiple enemies from which they will have to fight off with their guns or run away from if the enemies are too powerful, as well as find valuable loot . The win condition of this game would be to fully upgrade their ship and leave their crumbling solar system.

Mechanics:

-- there will be an inventory system to allow you to arrange or drop items
-- ships will be custom, in that you can upgrade "modules" of the ship
-- if a player finds planets they can set up bases on the planet by dropping off supplies.
-- players cannot modify their ship if they are off the "home planet" or "bases"
-- Rare unlocks can be obtained through "blueprints" small chance to drop from enemies
-- a shop to sell/ trade-in inventory and upgrade ship
-- Gain supplies by asteroid mining, or planet-side resources gained from exploring a planet.
-- The player must fight in-world against other ships or just run away which can cause player to lose a lot of fuel.
--(Optional) shooting asteroids can break them into smaller pieces or completely so you can just pick up the valuables, but reduces the amount of scavenged valuables gained.

The Minecraft approach (not art-style) can be used to create our world:

--procedurally generated environment upon world creation.
--enemies randomly spawn in areas of "conflict"
--resources are limited upon the specific seed so the player must be careful what to spend/collect
--If the player dies the collected supplies will be either be:

-- scavenged by pirates

-- float in space for a set amount of time
--no map system will exist (up for debate)

Game Story

You are a space privateer in a crumbling galaxy. Your goal is to survive, venture out into space and collect resources to save your colony. There's one problem, you're not alone in space, and that becomes quickly evident when you venture further into the space. In order to escape, your crew must collect all available resources and build the biggest, strongest, or fastest ship around. Learn the tricks of the venturing trade and salvation will be almost guaranteed for your colony.

Game Controls & User Interface

Keyboard Controls for in-game (MAC / LINUX / PC ONLY):
W – strafe world-space up
A – strafe world-space left
S – strafe world-space down
D – strafe world-space right
Mouse Drag – Aims the direction of the ship
Mouse L – Fires the guns
Mouse R - Locks the ship for docking If aimed at an asteroid or planet
WASD buttons can be combined to change strafe vector.
Touch based controls for in-game (IOS / ANDROID ONLY):
-----*Virtual Controller*
----------1 joystick for directed movement player
----------1 joystick for firing direction
----------1 button to pause the game
----------1 button to mine asteroids
---------- planets will essentially be buttons in order to dock onto them

User Interface:
-- Health Bar
-- Fuel Bar
-- Armor/Shield Bar
-- Pointer To Home Planet / Nearest Base (not necessary if we choose touch-based controls)
-- Visibility Indicator (Threat) : can be just a warning symbol

Monetization

The game will be available for free, due to this being for educational purchases, if we add monetization we will need a contract with CSUSB.

Replayability

The ability to have upgrades and customize the ship will allow for different configurations of in different run-throughs.

Sequel

Completely dependent on the success of the game and the school..

Inspirations & Examples

All these games are available on IOS, android has some of these but not all of them.

Tilt to Live – for use of accelerometer

Galaxy on Fire 2 – for use of mining and fighting very similar to this idea except with 3d and
not procedurally generated

Isotope – for use of twin stick controls with relative ease

Asteroids – scavenging for gold by "mining" which is just shooting asteroids, but has zero
story and very, very repetitive gameplay with no sense of direction.

GigaForce Lite- implementation of 3 different control schemes, Very basic clean UI.

Asset List

"~" = Approximately
~ 4 ship parts per "module"
--thrusters
--guns
--main hull
--Wings

~ 4 enemy ships to begin with, we can expand this later, purely for the sake of getting the project out
~ 6 differently shaped asteroids these can all be resized to add variation
a black background with some stars
a clear background with stars, distant solar systems, to essentially add a parallax affect to the background
~ 9 planets that can be used as bases, can be based on our solar system or just pretty looking
Music:
Background ambient music
Background ambient sound effects such as (thrusters, from ships).
Different Weapon Blasts
Background encounter music when high threat enemies

Job List

Job Title:
1) **Project Manager** – Parker
2) **Art Designer** – DJ
3)**Q.A. Tester –** Everyone
4) **Assistant Manager –** Ryan Haglund
5) **Game Programmer (Artificial Intelligence) –** Joseph Allison
6) **Game Programmer (Game Logic) –** Jared Johnson
8) **Game Programmer (Game Logic) –** Jon Antolin
9) **Game Programmer/Musician (Music/Sound)** - Aaron Flores
10) **UI Artist/ Programmer –** Marques Stone

Closing Statement

REMEMBER GUYS COMMUNICATION IS KEY TO FINISHING THIS PROJECT.
A little side note from me, I realize that not all of you are interested in the game design world, or want to do anything with game programming, but when it comes down to it, a completed project can be put on your resume, and with the scope of this project I have kept in mind the complexity so that when we finish this project, we can all put it on our resumes and show that we can finish a job we are told to do. Trust me I've heard it straight from recruiters mouths and even from several sources from within Sony those who do hires don't care about where you come from or what project you had in mind of doing, you may be a savant at coding, art, or music but if you don't have anything that is completed on your resume chances are you won't be hired unless you are interning or doing a really low position job.