Space Game

Software Requirements Specifications

Version 1.0

Johnathan Snyder, Craig Robinson, Brian Tillery
2/12/2014

Change History

Version	Summary	Author	Date
0.1	Initial template created	Brian Tillery	2/12/2014
1.0	Initial writeup	Johnathan Snyder Craig Robinson Brian Tillery	2/22/2014
1.1	Split non-functional requirements into two categories.	Craig Robinson Brian Tillery	2/24/2014
1.2	Added diagrams to section 5	Johnathan Snyder	2/24/2014

Note: This table summarizes all changes made to this document.

Table of Contents

1. Introduction

This section discusses the purpose, scope, and goals of the game. Also included are definitions that clarify any jargon used in this document.

1.1 <u>Motivation</u>

The successes of the many great free-to-play games on the current mobile app market are an inspiration to put out a fresh new take on the classic arcade-style spaceship games of yesterday mixed with the technological capabilities of today.

1.2 <u>Scope</u>

Space Game has been designed for Google's Android Platform, and is intended purely as a form of entertainment for any participating user on android based smart phones. All progress, high scores, and other data will be stored locally in a database.

1.3 <u>Goal(s)</u>

The goal of Space Game is simple: to create a fun, free, and engaging interactive mobile application that appeals to a general audience.

1.4 <u>Definitions</u>

<u>Perks</u> – Bonuses attainable in-game that can be purchased with Space Coins to modify long-term gameplay to the user's advantage.

<u>Space Coins</u> – Currency attainable in-game during the course of a flight that can be used to purchase perks as well as increase the player's score.

<u>Powerups</u> – Bonuses attainable in-game during the course of a flight that can temporarily modify gameplay to the user's advantage.

<u>2D Scroller</u> – A type of game whose artistic style typically consists of a two-dimensional level where the character(s) may progress as the screen scrolls to the next portion of the level.

<u>Achievements</u> – A set of sub-goals attainable in-game that may either yield in-game rewards or may not affect gameplay whatsoever.

4

2. Overall Description

2.1 <u>User Interfaces</u>

User interfaces will be touch screen menus found in typical games. The main menu will give the user the functionality to start the game, view the player's high score, view the player's achievements, buy perks for the player's ship, and change the game's settings. The in-game interface will allow the player to dodge obstacles and pickup coins or any other form of reward.

2.2 <u>Communication Interfaces</u>

At the moment, the game will not need to communicate with any outside source. This will obviously change if a social media sharing option is implemented. The game will communicate with a database used for storing information about settings, scores, and other data.

2.3 <u>Constraints</u>

Currently there are no perceived constraints that will affect the game.

2.4 <u>Application Features</u>

The user will move the spaceship to avoid obstacles and pick up space coins. Collision detection will be implemented to detect whether the player's spaceship touches an obstacle or space coin. Gameplay will continue infinitely until an obstacle hits the spaceship. Space coins will be used to buy perks. The player's high score and coin balance will be saved locally.

2.5 <u>Optional Functionality</u>

In addition to the functionality described above, the following items could also be implemented if time permits:

1. The ability to share your high score on a social media site. Sharing high scores on social media would promote competition against friends.
2. The ability to buy more space ships and change them before each round. Customization adds replay value to games. The user will play the game longer is there is an incentive to keep collecting space coins.
3. Implementation of a health system that would allow more perks and customization. The health system would allow the user's spaceship to sustain more damage. Also, the user could buy shields, repair kits, etc. to further his/her survival.

5

2.6 <u>Assumptions and Dependencies</u>

The minimum SDK version supported is Android 2.3 (Gingerbread), the maximum SDK version supported is SDK 4.4(KitKat), and the target SDK is 4.3 (Jelly Bean).

3. Functional Requirements

This section contains the functional requirements for the game. The requirements are arranged by level of priority.

Name	Description	Priority
Capturing touch events	Touch events will need to be captured so that the user can navigate menus and also control the space ship.	High Priority
Collision Detection	The game will need to handle the collisions that take place between the spaceship and the obstacles.	High Priority
Storage	The game will need to store information such as the user's high score, achievements unlocked, coins available, and perks bought.	High Priority
In-game Pausing	The ability to pause live games is needed in the case of the user receiving a phone call, or in the event the user needs to accomplish some other task.	Med. Priority
Game Settings	The user will be able to modify game settings such as sound, vibration, or reset game progress.	Low Priority

4. Non-Functional Requirements

This section contains the non-functional requirements for the game.

4.1 <u>Developer Non-Functional Requirements</u>

| Name	Description	Priority
Git Repository	Application files will be stored in a Git repository to track versioning.	High Priority
AndEngine	This game engine will aid in the production of the game.	High Priority
Box2D	This extension of AndEngine handles object collisions and physics.	Med. Priority

4.2 <u>User Non-Functional Requirements</u>

| Name	Description	Priority
Precise collision detection	Collision calculations should be accurate so errors in collision detection do not occur. Errors in collision detection hurt gameplay experience.	High Priority
Responsive swiping	The capturing and handling of touch events needs have little to no latency. A high degree of latency will cause users to stop playing the game.	High Priority
Ease of Use	Everything in application is accessible from the main menu by no more than two button touches.	Med. Priority

5. Requirements Diagrams

This section contains the use case diagrams, user activity diagrams, and high level class diagrams associated with the game.
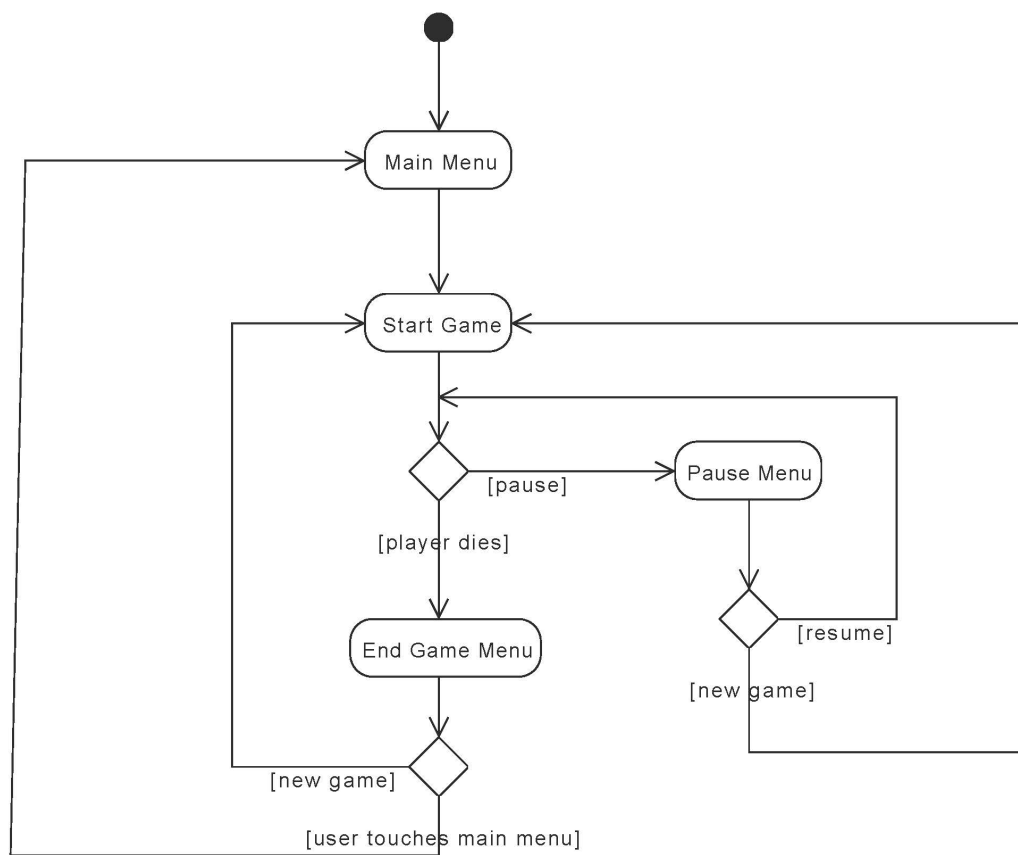
5.1 <u>Use Case Diagrams</u>
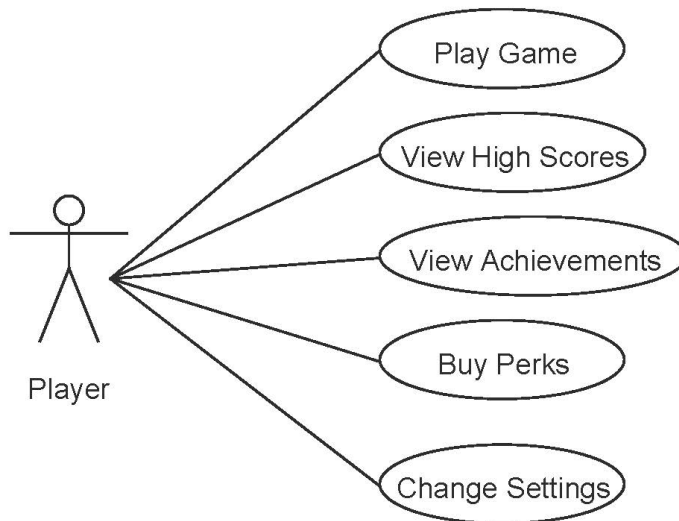
The use case diagram when the player is at the main menu.
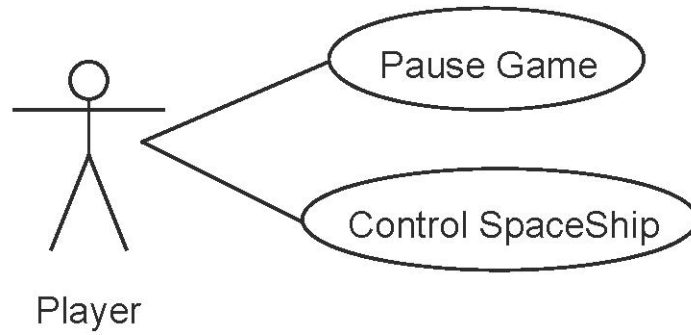
The use case diagram when the player is playing the game.

5.2 **User Activity Diagrams**

The activity diagram for when the player is playing the game. When the player is in the game, the player can either die or pause the game. If the player pauses the game, the player can then resume the game or start a new game. If the player dies, an end game menu pops up with the options to start a new game or go back to the main menu.
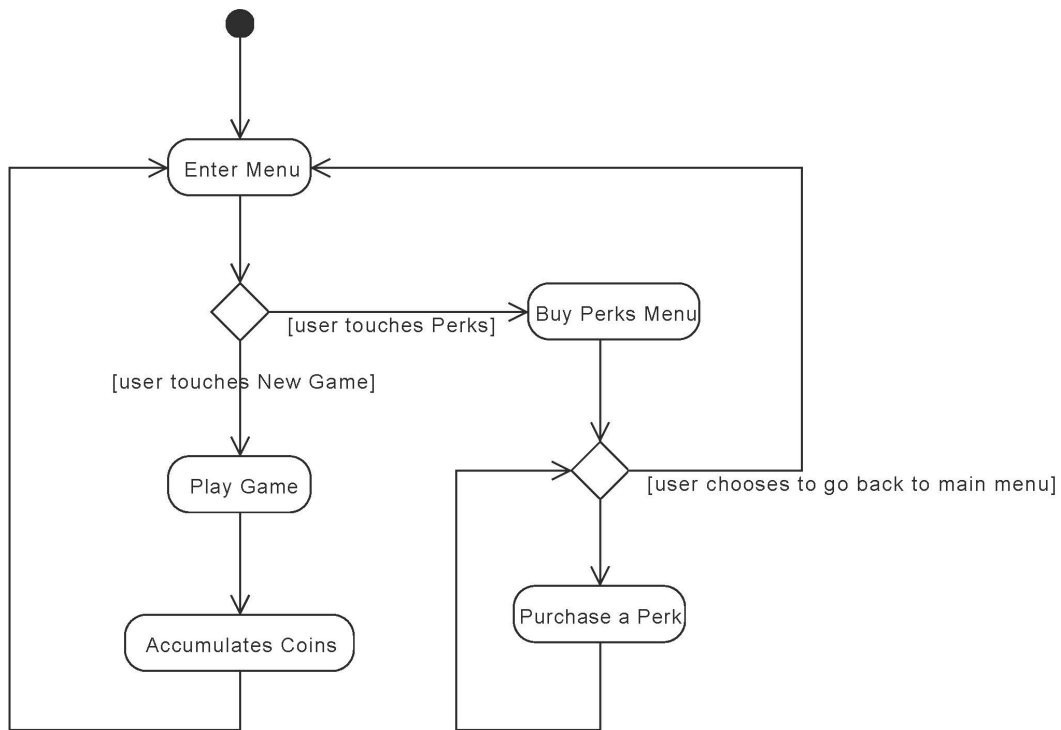
The activity diagram for the perk system. When the player enters the main menu, the player has the option to buy perks. In order to buy perks, the user has to accumulate coins during game play.

5.3 **High Level Class Diagram**
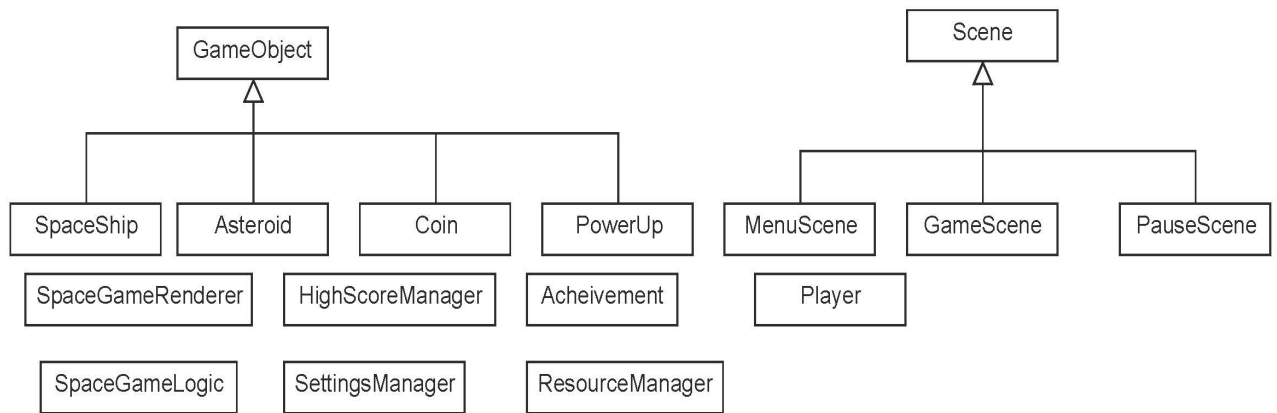
The high level diagram for the whole system. This diagram only includes classes that we think we might need. We might have to add or remove some of the classes when we get to the design phase.