Department of
Computer Science

Software Requirements Specification

for

Line Bounce

Version: 0.1
26 September 2013

This is the place to include the abstract. The aim of the abstract is to briefly informing the reader what the document is about.

Copyright notice

Department of Computer Science
The University of Massachusetts Amherst
Amherst, Massachusetts
United States
01003

Computer Science Building
140 Governors Drive
University of Massachusetts
Amherst, MA 01003-9264

Tel: +1 413 545 2744
Fax: +1 413 545 1249
`https://www.cs.umass.edu/`

Version: 0.1
26 September 2013.

Credits

This document was written by Trevor Droeske, Ben Setzer, John Holowczak, Max Otsuka, Lily Li, Adam Tabis, Dylan Sullivan, Chris Paika, David Itkin, Emily Henriksen, Artyom Bychkov and Aaron Sewall.

Acknowledgments

The following people provided assistance with or were involved in the development of the document: Lee Osterweil, Aaron St. John

CONTENTS

LIST OF FIGURES

ONE

Introduction

1.1 Background

The casual gaming industry is huge and is responsible for supplying over 200 million people worldwide with entertainment.The purpose of Line Bounce is to supply people with temporary entertainment in a safe environment. Since Line Bounce is so closely related to Doodle Jump, it might be beneficial to talk about the success of Doodle Jump in order to express our hope for Line Bounce.

Doodle Jump has sold more than 3.5 million copies of its game for users on cell phones, iPads, and other devices. This accomplishment has made the developers of Doodle Jump the world's most successful iPhone and iPod touch developers. It's a simple but addicting game that has easily provided the developers over a million dollars before taxes and excluding Apple's 30% cut. With such success from a similar game, there is a hope that Line Bounce will be similarly successful.

1.2 Concept

The concept of the game is to have an avatar jump as high as possible using platforms that will be drawn by the player. There will be enemies in the path of the avatar, so the player will need to draw angled platforms to dodge the enemies. Also, the player can angle their avatar's path to obtain power-ups and coins in the environment. The levels will also have a limitless height and the user's avatar can use the walls to bounce off of as well. In addition, the player will be rewarded with coins after every 1000 feet they clear. The environments can be changed from game to game to keep the player entertained.

1.3 Purpose

The purpose of this document is to clearly outline the client requirements and the system requirements for the game Line Bounce. This document should cover all key concepts and and fundamental requirements. The requirements should both satisfy the client's wishes in the game and provide accuracy and functionality for the game.

1.4 Intended Audience

- Team Members: To completely understand what will be needed for implementation when developing Line Bounce.
- Project Client: To ensure the designed game meets the client's requirements.
- Supervisor: To understand what the project entails and entrust the group with implementation of said project.
- Maintainers: For those who will maintain the game after launched, they may refer to this document to gain insight of the system and an understanding of the project goals.
- Future Developers: For the future developers, they may look at this document to gain insight into the original game design in order to develop and enhance the game appropriately.

1.5 Scope

- All requirements for the game will be clearly explained and expressed in this document.
- This document states all requirements, functional or non-functional.
- When requirements meet the standard of the client, the client will sign the document as an agreement to the given requirements for the game Line Bounce.

TWO

Functional Requirements

2.1 Gameplay

2.1.1 FUN-01: Player starts the game from the top of the screen

Description: The player should begin at the ceiling and accelerate downward, leaving a few seconds to draw the first platform and begin playing.

2.1.2 FUN-02: The user draws lines to bounce their character upward

Description: The player must draw line segments to bounce off of. This is done by swiping your finger on the touch screen or clicking and dragging with the mouse.

2.1.3 FUN-03: The user loses when the avatar falls off the screen or runs into a hazard

Description: The player loses by falling off the bottom of the screen or running into an enemy.

2.1.4 FUN-04: Screen moves upward in height during gameplay

Description: The screen will only scroll up when the avatar is moving vertically upward. The avatar will not pass the top of the screen before the screen is readjusted.

2.1.5 FUN-05: The avatar will bounce back down off of the bottom of platforms

Description: If the user collides with a platform through the bottom, it will pass through without any change in velocity.

2.1.6 FUN-06: Side walls are solid

Description: The avatar can bounce off of the wall to avoid enemies and other hazards.

2.1.7 FUN-07: The user has a limited length of line they can draw

Description: When the player has drawn platforms of combined length equal to the width of the screen and continues to draw, their previous drawings are steadily erased. To visualize this, imagine the movements of the snake in the classic game 'Snake.'

2.1.8 FUN-08: Power-ups will be activated immediately upon acquisition

Description: Power-ups will be activated as soon as soon as the avatar acquires them.

2.1.9 FUN-09: The user can pause during gameplay

Description: At any point during gameplay the user can hit the pause button allowing them bring up a separate screen menu to suspend play. This menu does not allow the user to see the gameplay in order to strategize. The pause menu allows the user to quit the game, go to the store (loses progress), restart the current game, and to be able to resume the game. There will be a one-second countdown when they click resume until the game begins again.

2.1.10 FUN-10: Screen never moves downwards

Description: The screen will only move upwards as the player jumps and will never move downward to follow the player's fall.

2.1.11 FUN-11: The activated power-ups will be displayed on the screen

Description: A list of the power-ups that are currently activated (in the form of a row of icons), will be displayed at the top of the screen during gameplay. These power-ups will be passive only and not be clickable.

2.1.12 FUN-12: Smaller platforms cause the player to bounce higher

Description: The length of the platform will determine how much upward acceleration the player gains from bouncing, with shorter length equaling more bounce.

2.1.13 FUN-13: A power-up will be collected when the avatar comes into contact with it

Description: The avatar can pass completely through or just graze the power-up to acquire it. Any form of contact between the two sprites will count as a hit.

2.1.14 FUN-14: Users receive experience points and coins after each playthrough

Description: At the end of the game, the user receives experience points proportional to the height reached and number of enemies slain.

2.1.15 FUN-15: The user has a level, determined by total experience points

Description: Levels are determined by total experience points. Gaining levels allows the user to purchase new items in the store.

2.1.16 FUN-16: Enemy generation occurs randomly

Description: An arbitrary number of enemies are generated during gameplay every few seconds. The number of enemies is random, but within a certain range that is actively changing during play.

2.1.17 FUN-17: Difficulty of game increases with height and level

Description: The range described in FUN-16 is based on the user's level as well as the current height at the time of generation. As the player gets to higher levels (overall) and heights (in a playthrough), both the maximum and the minimum of the range will increase.

2.1.18 FUN-18: The user's current score will be displayed on the screen

Description: During gameplay, the user's current score will be displayed on the screen and updated in real time to reflect the current playthrough. The user will initially have a score of zero at the beginning of the game. This score will never decrease. As the avatar rises and their height increases, the score in the corner will change.

2.1.19 FUN-19: A drawn line will "materialize" upon completion

Description: A line will only become solid, and allow the avatar to bounce, when the user lifts his finger, and the second endpoint of the line has been established.

2.1.20 FUN-20: Lines drawn through an avatar will do nothing

Description: Regardless of whether or not the user has completed the line, the avatar will always pass through the line, if the user has drawn the line through the avatar.

2.2 Menu

2.2.1 FUN-21 Game Over menu will feature several courses of action

Description: This menu will offer the player a chance to start a new game, post a score to the Social page, go to the Store, go to the leaderboards or return to the Main Menu.

2.2.2 FUN-22 The Main Menu should feature many options

Description: The menu should allow the user the choice to begin a new game, check the leaderboards, access the in-game store, invite friends through facebook, or adjust settings.

2.2.3 FUN-23 The Online Store will feature many purchasable items

Description: Users will be able to generate as well as buy in-game currency and other items such as power-ups, new skins/stages, new music, etc. This will represented by a list of categories that users click on in order to purchase the item in that category. Each item will have their price written next to them.

2.2.4 FUN-24 The Settings menu should allow users to change options

Description: The user should be able to log in and out of Facebook, sync his/her data, change whether or not they get in-game notifications, and reset their local history.

2.2.5 FUN-25 The start screen will give the option of connecting to Facebook

Description: There will be a button that links to the Facebook login screen[c] so that the user can sign into the game. The user can also press a "Play Offline" button to play the game without connecting to Facebook.

2.2.6 FUN-26 The preplay screen which will allow the user to select items

Description: On the preplay screen the user's avatar will be displayed along with his/her stats. A list of the items the user has already purchased (including power-ups, skins, etc.) will be visible, and each item will have a corresponding check box to allow the user to select it for their next playthrough.

2.3 Settings

2.3.1 FUN-27 Facebook authentication will be required for purchases

Description: The user can view the store's contents without logging, in, however he/she will need to authenticate via Facebook before being able to purchase any item. The user will be able to buy coins through the Facebook store interface.

2.3.2 FUN-28 Users can reset their in-game progress

Description: Users can reset their progress, statistics and achievements on both their facebook account and on their offline/local account. This distinction will be made clear in the settings menu so that users don't accidentally reset the wrong account.

2.3.3 FUN-29 Resetting progress should be simple with user confirmation

Description: The reset function should be simple and easy and have user confirmation to prevent accidental reset.

2.3.4 FUN-30 The user can logout of Facebook from the settings menu

Description: The user can logout of Facebook and will be able to continue to play in offline mode. A confirmation dialog will pop up asking the user to verify that they wish to log out. The confirmation dialog will also notify the user that their progress will be synced to Facebook the next time they log in.

2.3.5 FUN-31 Backgrounding the application mid-playthrough will pause the game

Description: Backgrounding the application (via a home button press, for example) will pause the game. The state of the game will be saved so that when the user returns he/she will be brought to the pause menu and can resume play.

2.3.6 FUN-32 Exiting the application mid-playthrough will end the playthrough

Description: Exiting the application (e.g. closing the browser window or quitting the application entirely) will completely lose all of the current playthrough's progress

2.4 Social

2.4.1 FUN-33 The user will be able to post a score to Facebook

Description: The user will be able to post a score to Facebook with a short message attached. This should be done without loading the social page.

2.4.2 FUN-34 The user must be able to access and post any previous scores

Description: This can be done from a tab on the Social Features page. The posts go to their Facebook wall.

2.4.3 FUN-35 The user will receive coins for inviting friends to play

Description: An sent invite will earn the user a few coins, an accepted invitation will earn a much larger sum.

2.4.4 FUN-36 Users will be able to send power-ups to friends

Description: Next to each name on the leaderboard, there will be an arrow icon. Tapping it will pop-up a confirmation page to send a power-up to that friend. After sending a power-up there will be a cooldown period during which the user will be unable to send power-ups to the same friend.

2.4.5 FUN-37 The user will be alerted when they score higher than their friends

Description: After a playthrough, a popup screen (before the game over screen) will notify the user if he/she scored higher than a friend's personal best during that playthrough.

2.4.6 FUN-38 The user will receive notifications about friends' activities

Description: The user will only receive notifications about activities involving the user directly. For example, the user will receive a notification if a friend gifts them a power-up, or if the friend surpasses their high score.

THREE

Non-functional Requirements

3.1 Environmental

3.1.1 ENV-01: Game will run on multiple platforms

Description: The game should run on iOS and Android devices as well as Facebook through a modern browser.

3.1.2 ENV-02: Game will support all screen sizes regardless of platform

Description: The game will run on all sizes of iOS and Android screens.

3.1.3 ENV-03: Game will be in a portrait view with a flexible aspect ratio

Description: Our game will be displayed in portrait mode only. The aspect ratio will be exactly the same on all platforms, regardless of the screen's true aspect ratio. When fitting the display to the device's screen, the image will be scaled, and there may or may not be blank space along the edge of the screen to accommodate the change in aspect ratio.

3.1.4 ENV-04: Game resolution will use vector graphics to scale viewport

Description: The game will use vector graphics for all of the graphical components (menus, avatars, sprites, etc.) so that scaling does not cause pixelation issue on platforms with larger screens.

3.1.5 ENV-05: The user should be able to mute the in-game sound

Description: The in-game sound should be split up into two different groups, background music and sound effects. The two should be able muted/unmuted independently of each other.

3.1.6 ENV-06: The user does not need to finish the game in one sitting

Description: The target user is someone who has a small amount of free time, for example sitting on a bus. The game should be "addicting" and make the user want to play repeatedly, but individual games should be fairly short. The user will also be able to pause the game, and continue at a later time.

3.1.7 ENV-07: The Pause button will be displayed on the screen[s]

Description: The top-right corner of the screen during gameplay will always feature the pause button (except when in pause mode), which can be pressed at any time during gameplay.

3.1.8 ENV-08: The menu screens will display the user's current level

Description: The top-right corner of the screen while the user is navigating the menus should always say what level the user is currently at and their progress within that level.

3.1.9 ENV-09: The screen should tell the user how much money they have.

Description: The top-left corner of the screen should always say how much in-game currency the user has remaining.

3.1.10 ENV-10: The store will have a list of categories for the user to choose.

Description: Clicking on each of the categories will bring the user to a scrollable menu featuring all of the relevant items to the category. The user will then be able to purchase the items.

3.1.11 ENV-11: The user will have access to global and friend-only leaderboards

Description: The user will be able to access a global leaderboard or a leaderboard of his/her Facebook friends. There will be a toggle button to switch between the two views.

3.2 Performance

3.2.1 PER-01: Small game file size (¡50 megabytes)

FOUR

Non-functional Requirements

4.1 Environmental

4.1.1 ENV-01: Game will run on multiple platforms

Description: The game should run on iOS and Android devices as well as Facebook through a modern browser.

4.1.2 ENV-01: Game will support all screen sizes regardless of platform

Description: The game will run on all sizes of iOS and Android screens.

4.1.3 ENV-02: Game will be in a portrait view with a flexible aspect ratio

Description: Our game will be displayed in portrait mode only. The aspect ratio will be exactly the same on all platforms, regardless of the screen's true aspect ratio. When fitting the display to the device's screen, the image will be scaled, and there may or may not be blank space along the edge of the screen to accommodate the change in aspect ratio.

4.1.4 ENV-03: Game resolution will use vector graphics to scale viewport

Description: The game will use vector graphics for all of the graphical components (menus, avatars, sprites, etc.) so that scaling does not cause pixelation issue on platforms with larger screens.

4.1.5 ENV-02: The user should be able to mute the in-game sound

Description: The in-game sound should be split up into two different groups, background music and sound effects. The two should be able muted/unmuted independently of each other.

4.1.6 ENV-03: The user does not need to finish the game in one sitting

Description: The target user is someone who has a small amount of free time, for example sitting on a bus. The game should be "addicting" and make the user want to play repeatedly, but individual games should be fairly short. The user will also be able to pause the game, and continue at a later time.

4.1.7 ENV-04: The Pause button will be displayed on the screen[s]

Description: The top-right corner of the screen during gameplay will always feature the pause button (except when in pause mode), which can be pressed at any time during gameplay.

4.1.8 ENV-05: The menu screens will display the user's current level

Description: The top-right corner of the screen while the user is navigating the menus should always say what level the user is currently at and their progress within that level.

4.1.9 ENV-06: The screen should tell the user how much money they have.

Description: The top-left corner of the screen should always say how much in-game currency the user has remaining.

4.1.10 ENV-07: The store will have a list of categories for the user to choose.

Description: Clicking on each of the categories will bring the user to a scrollable menu featuring all of the relevant items to the category. The user will then be able to purchase the items.

4.1.11 ENV-08: The user will have access to global and friend-only leaderboards

Description: The user will be able to access a global leaderboard or a leaderboard of his/her Facebook friends. There will be a toggle button to switch between the two views.

4.2 Performance

4.2.1 PER-01: Small game file size (¡50 megabytes)

Description: As there is a fairly limited amount of content in Line Bounce, and we have decided not to implement a level-based game structure, we should be able to keep

our game size down to under 50 MB. Over a 3G connection speed of 200 KB/s, the game file should not take any longer than 3 minutes to download. Over a 4G connection speed of 1 MB/s, the game file wouldn't take any more than a minute to download. All Wi-Fi connections are different, so the two above listed download rates should be used as benchmarks to gauge how quickly the game file can be downloaded

4.2.2 PER-02: Non-maximum CPU usage

Description: Our application should occupy no more than 80% of the mobile device's CPU during gameplay for rendering the game.

4.2.3 PER-03: Game frame rate of at least 30 frames per second

Description: Game will run at 30 frames per second to ensure smoothness of animation and game speed.

4.2.4 PER-04: Fast menu response time

Description: For the menu, response time should be reasonably fast so as not to infuriate users, so a 0.01-0.25 second window of lag time will be implemented.

4.2.5 PER-05: Fast in-game response time

Description: The actual gameplay, we must provide a responsiveness that appears to be near-instantaneous to the human eye. The amount of lag time should be less than 24-33 milliseconds for each action. This includes time will be implemented.

4.2.6 PER-05: Fast in-game response time

Description: The actual gameplay, we must provide a responsiveness that appears to be near-instantaneous to the human eye. The amount of lag time should be less than 33 milliseconds for each action. This includes the lag time between the moment a user presses the screen to draw a line and when it appears on screen. It also includes the time between the obtaining of a power-up and the assimilation of its effects.

4.2.7 PER-06: Fast database access

Description: When accessing the store, and when logging in, the delay time will remain under 5 seconds. The user's device must access the server, which accesses the stored data in the Database, and then the data is sent back to the user. So the speed of both the user accessing the server and the server communicating with the DB to under 2 seconds each.

4.3 Accuracy

4.3.1 ACC-01: Input response should be accurate

Description: The game should respond accurately, where touch and mouse input should respond in the correct location on the screen. This will be with an accuracy of +/- 2 pixels.[e]

4.3.2 ACC-02: Straight line approximation of path that player draws

Description: The player draws a platform for the avatar to bounce from by touching and dragging across the screen, and then lifting their finger to mark the end of the platform. The accuracy of this is reliant on the API used to gather the touch information, which will differ between iOS and Android as well as among the individual devices using these platforms. On Facebook this accuracy is inherent, as the mouse is a much more exact means of drawing than a finger. The first touch and lifting of the finger will mark the start and end of the line. For the Facebook game, the left click, drag and release of the left click will mark the start and end of the line respectively.

4.3.3 ACC-03: Accidental touches will be ignored by the game engine

Description: The game will ignore taps (without finger dragging) in the main, gameplay portion of the screen, as only swipes draw platforms in game and this prevents accidental touches. The user will have to drag their finger a distance at least equal to the width of the avatar in order for the line to register. The pause button, however, will be activated by single-point touches, and not by swipes.

4.3.4 ACC-04: People with larger fingers will still be able to play the game

Description: For people with larger fingers their line will start drawing where the middle of their finger touches the screen , allowing even large fingers to draw lines with a reasonable degree of accuracy. When playing the game on Facebook this is not an issue, as the mouse is used to draw lines.

4.3.5 ACC-05: In an optimal game the player climbs forever

Description: While an optimal game is not technically possible it allows us to set a standard. A good game would be one where the player reaches a new high score for their avatar's height, reaching a higher point on the map than previously achieved. The map will auto-generate as the avatar climbs. At least the height of two screens will be generated past the top of the screen in preparation for the avatar to climb.

4.3.6 ACC-06: Avatar will bounce upon touching the platform

Description: The avatar should bounce off of a platform as long as the avatar touches the platform and they have downward velocity.

4.3.7 ACC-07: Bounce acceleration calculation has at least 3 significant figures

Description: When the avatar collides with a platform the acceleration due to this bounce must be correctly calculated to within at least 6 significant figures. Using linear algebraic mathematical formulas, the game will calculate the angle that the avatar collided with a platform and find the reflection vector. That information is then used to apply an impulse to the avatar in the appropriate direction with an accurate acceleration based on the size of the line as stated in FUN-23.

4.3.8 ACC-08: Avatar will die upon touching any portion of an enemy

Description: As long as a portion of the user's sprite touches touches an enemy, the user will either die or lose their shield if they have a shield powerup equipped.

4.4 Robustness

4.4.1 ROB-01: Game should never crash the user's device

Description: The game will never cause a fatal crash of the user's device. The game may crash but should never crash the device itself.

4.4.2 ROB-02: Buttons should always perform desired functionality

Description: The buttons should be straightforward in what they do and the user should never have to question what a button does.

4.4.3 ROB-03: The game must never corrupt any database

Description: Game player status and payment records must never be lost, especially after crash or loss of connection. No queries sent to the database should corrupt any tables.

4.4.4 ROB-04: Purchases should be easily recovered

Description: A recovery function linked to the account of the player will be implemented. This will allow the user to sync any data to their platform if something is to go wrong with the auto-sync functionality. It is accessible in the Settings menu.

4.4.5 ROB-05: The game will be easily extensible

Description: It will be easy to add new features to the game. This may include game modes, power-ups, enemy types, store categories and avatar customization.

4.5 Safety

4.5.1 SAF-01: Game should be family friendly (G-rated)

Description: Game will be made suitable to play for all audiences.

4.5.2 SAF-02: Warn users of extended play sessions

Description: The game will warn users who have been playing for a long continuous period of time that they should take a break ("It appears you've been playing for a while, please take a break"). However, the game will never force users to exit. This will happen after two hours of continuous play.

4.5.3 SAF-03: Bugs should be easily patchable

Description: The application should be written in a clear, concise manner and thoroughly abstracted and DRYed (Don't Repeat Yourself) out so that writing patches for bugs is simple and easy. A single code-base ensures that one patch will fix the bug for every platform

4.5.4 SAF-04: Game should keep track of in-game purchases

Description: Local and remote purchase tracking required. This data will be automatically synced if needed (e.g. The user plays a game on a different platform than the one they made purchases on).

4.5.5 SAF-05: Confirm transaction details

Description: Before processing transaction, ask customer to confirm the transaction details. A small popup will be displayed on the current page with the item the user is purchasing as well as the price they are agreeing to pay.

4.5.6 SAF-06: Game will include a fail-soft database

Description: Keep database information minimal so that the game does not have to connect to the database very often. There will also be a second database that will be synced up regularly with the original in case of a failure.

4.5.7 SAF-07: Backup database regularly

Description: The database will be backed up every 15 minutes to minimize lost customer purchases.

4.6 Security

4.6.1 SEC-01: Game will use Facebook authentication

Description: Facebook OAuth will be used for authentication. This allows the user to log in to their Facebook account securely. The game will need to use OAuth correctly for the game to be secure. Facebook provides information on correct implementation.

4.6.2 SEC-02: Game will use Facebook payment system

Description: Facebook's API for secure payment transactions will be used, as requested by the client. This is the only method of payment accepted. The game will need to use the Facebook payment system correctly for the game to be secure. Facebook provides information on correct implementation. Transactions will not be lost, as specified in ROB-02.

4.6.3 SEC-03: Data will be stored on a secure database server

Description: User purchase history and scores will be stored on the server. The database will only be accessed by the game server, not the game clients. This allows for only the game server to hold the credentials to the database.

4.6.4 SEC-04: Data storage for both client and server

Description: Server-side data will be used as temporary storage until the secure database is updated with the user's progress. It will push the changes to the database when the user logs out of Facebook or a connection is lost. Client-side data will be for the user to play in offline mode if he/she does not want to utilize these features or if they does not currently have an active internet connection.

4.6.5 SEC-05: Data will be sanitized

Description: Data sanitization will protect against attacks by malicious users. For all input boxes, the use of HTML tags, JavaScript or any other possible code that could jeopardize the security of the game **Description**: As there is a fairly limited amount of content in Line Bounce, and we have decided not to implement a level-based game structure, we should be able to keep our game size down to under 50 MB. Over a 3G connection speed of 200 KB/s, the game file should not take any longer than 3 minutes

to download. Over a 4G connection speed of 1 MB/s, the game file wouldn't take any more than a minute to download. All Wi-Fi connections are different, so the two above listed download rates should be used as benchmarks to gauge how quickly the game file can be downloaded

4.6.6 PER-02: Non-maximum CPU usage

Description: Our application should occupy no more than 80% of the mobile device's CPU during gameplay for rendering the game.

4.6.7 PER-03: Game frame rate of at least 30 frames per second

Description: Game will run at 30 frames per second to ensure smoothness of animation and game speed.

4.6.8 PER-04: Fast menu response time

Description: For the menu, response time should be reasonably fast so as not to infuriate users, so a 0.01-0.25 second window of lag time will be implemented.

4.6.9 PER-05: Fast in-game response time

Description: The actual gameplay, we must provide a responsiveness that appears to be near-instantaneous to the human eye. The amount of lag time should be less than 24-33 milliseconds for each action. This includes time will be implemented.

4.6.10 PER-05: Fast in-game response time

Description: The actual gameplay, we must provide a responsiveness that appears to be near-instantaneous to the human eye. The amount of lag time should be less than 33 milliseconds for each action. This includes the lag time between the moment a user presses the screen to draw a line and when it appears on screen. It also includes the time between the obtaining of a power-up and the assimilation of its effects.

4.6.11 PER-06: Fast database access

Description: When accessing the store, and when logging in, the delay time will remain under 5 seconds. The user's device must access the server, which accesses the stored data in the Database, and then the data is sent back to the user. So the speed of both the user accessing the server and the server communicating with the DB to under 2 seconds each.

4.7 Accuracy

4.7.1 ACC-01: Input response should be accurate

Description: The game should respond accurately, where touch and mouse input should respond in the correct location on the screen. This will be with an accuracy of +/- 2 pixels.[e]

4.7.2 ACC-02: Straight line approximation of path that player draws

Description: The player draws a platform for the avatar to bounce from by touching and dragging across the screen, and then lifting their finger to mark the end of the platform. The accuracy of this is reliant on the API used to gather the touch information, which will differ between iOS and Android as well as among the individual devices using these platforms. On Facebook this accuracy is inherent, as the mouse is a much more exact means of drawing than a finger. The first touch and lifting of the finger will mark the start and end of the line. For the Facebook game, the left click, drag and release of the left click will mark the start and end of the line respectively.

4.7.3 ACC-03: Accidental touches will be ignored by the game engine

Description: The game will ignore taps (without finger dragging) in the main, gameplay portion of the screen, as only swipes draw platforms in game and this prevents accidental touches. The user will have to drag their finger a distance at least equal to the width of the avatar in order for the line to register. The pause button, however, will be activated by single-point touches, and not by swipes.

4.7.4 ACC-04: People with larger fingers will still be able to play the game

Description: For people with larger fingers their line will start drawing where the middle of their finger touches the screen , allowing even large fingers to draw lines with a reasonable degree of accuracy. When playing the game on Facebook this is not an issue, as the mouse is used to draw lines.

4.7.5 ACC-05: In an optimal game the player climbs forever

Description: While an optimal game is not technically possible it allows us to set a standard. A good game would be one where the player reaches a new high score for their avatar's height, reaching a higher point on the map than previously achieved. The map will auto-generate as the avatar climbs. At least the height of two screens will be generated past the top of the screen in preparation for the avatar to climb.

4.7.6 ACC-06: Avatar will bounce upon touching the platform

Description: The avatar should bounce off of a platform as long as the avatar touches the platform and they have downward velocity.

4.7.7 ACC-07: Bounce acceleration calculation has at least 3 significant figures

Description: When the avatar collides with a platform the acceleration due to this bounce must be correctly calculated to within at least 6 significant figures. Using linear algebraic mathematical formulas, the game will calculate the angle that the avatar collided with a platform and find the reflection vector. That information is then used to apply an impulse to the avatar in the appropriate direction with an accurate acceleration based on the size of the line as stated in FUN-23.

4.7.8 ACC-08: Avatar will die upon touching any portion of an enemy

Description: As long as a portion of the user's sprite touches touches an enemy, the user will either die or lose their shield if they have a shield powerup equipped.

4.8 Robustness

4.8.1 ROB-01: Game should never crash the user's device

Description: The game will never cause a fatal crash of the user's device. The game may crash but should never crash the device itself.

4.8.2 ROB-02: Buttons should always perform desired functionality

Description: The buttons should be straightforward in what they do and the user should never have to question what a button does.

4.8.3 ROB-03: The game must never corrupt any database

Description: Game player status and payment records must never be lost, especially after crash or loss of connection. No queries sent to the database should corrupt any tables.

4.8.4 ROB-04: Purchases should be easily recovered

Description: A recovery function linked to the account of the player will be implemented. This will allow the user to sync any data to their platform if something is to go wrong with the auto-sync functionality. It is accessible in the Settings menu.

4.8.5 ROB-05: The game will be easily extensible

Description: It will be easy to add new features to the game. This may include game modes, power-ups, enemy types, store categories and avatar customization.

4.9 Safety

4.9.1 SAF-01: Game should be family friendly (G-rated)

Description: Game will be made suitable to play for all audiences.

4.9.2 SAF-02: Warn users of extended play sessions

Description: The game will warn users who have been playing for a long continuous period of time that they should take a break ("It appears you've been playing for a while, please take a break"). However, the game will never force users to exit. This will happen after two hours of continuous play.

4.9.3 SAF-03: Bugs should be easily patchable

Description: The application should be written in a clear, concise manner and thoroughly abstracted and DRYed (Don't Repeat Yourself) out so that writing patches for bugs is simple and easy. A single code-base ensures that one patch will fix the bug for every platform

4.9.4 SAF-04: Game should keep track of in-game purchases

Description: Local and remote purchase tracking required. This data will be automatically synced if needed (e.g. The user plays a game on a different platform than the one they made purchases on).

4.9.5 SAF-05: Confirm transaction details

Description: Before processing transaction, ask customer to confirm the transaction details. A small popup will be displayed on the current page with the item the user is purchasing as well as the price they are agreeing to pay.

4.9.6 SAF-06: Game will include a fail-soft database

Description: Keep database information minimal so that the game does not have to connect to the database very often. There will also be a second database that will be synced up regularly with the original in case of a failure.

4.9.7 SAF-07: Backup database regularly

Description: The database will be backed up every 15 minutes to minimize lost customer purchases.

4.10 Security

4.10.1 SEC-01: Game will use Facebook authentication

Description: Facebook OAuth will be used for authentication. This allows the user to log in to their Facebook account securely. The game will need to use OAuth correctly for the game to be secure. Facebook provides information on correct implementation.

4.10.2 SEC-02: Game will use Facebook payment system

Description: Facebook's API for secure payment transactions will be used, as requested by the client. This is the only method of payment accepted. The game will need to use the Facebook payment system correctly for the game to be secure. Facebook provides information on correct implementation. Transactions will not be lost, as specified in ROB-02.

4.10.3 SEC-03: Data will be stored on a secure database server

Description: User purchase history and scores will be stored on the server. The database will only be accessed by the game server, not the game clients. This allows for only the game server to hold the credentials to the database.

4.10.4 SEC-04: Data storage for both client and server

Description: Server-side data will be used as temporary storage until the secure database is updated with the user's progress. It will push the changes to the database when the user logs out of Facebook or a connection is lost. Client-side data will be for the user to play in offline mode if he/she does not want to utilize these features or if they does not currently have an active internet connection.

4.10.5 SEC-05: Data will be sanitized

Description: Data sanitization will protect against attacks by malicious users. For all input boxes, the use of HTML tags, JavaScript or any other possible code that could jeopardize the security of the game

A

Glossary

The following definitions, acronyms and abbreviations are used throughout this document:

Avatar: The in-game character that the player manipulates through use of the drawn platforms

Character: see Avatar

Coins: The in-game currency with which the user may use to buy items in the store

CS: Computer Science

DB: Database

Enemies: An avatar that hinders the player's success by either hurting the avatar or causing the avatar to fall

Environment: The background the player will choose during gameplay

Experience: The point system that is used solely to track user level

GUI: Graphical User Interface

Hazard: Anything that can potentially harm the Avatar, such as enemies.

Level: A term used to record a user's overall progress across many playthroughs

Platform: A line drawn by the player with a finger swipe on iOS/Android and a mouse click and drag on Facebook. When the avatar collides with the platform the avatar vertical velocity reflects off of the platform

Player: The user playing the game through their iOS/Android/Facebook device

Playthrough: The entire game loop from the first jump until the score screen shows up after the avatar has died/the timer has run out/etc.

Power-Up: An item that floats down the screen and can be picked up by the player

SRS: Software Requirements Specification

SDD: Software Design Document

Stage: The in-game background

UMass: University of Massachusetts Amherst

User: see Player