Software Requirements Specification

for

Hookshot

Version 1.0

Prepared by: Tan Wei Wen / Tan Egi / Lau Yong Hui / Liu Ke

Team: Hook

28th January 2021

Table of Contents

Table of Contents

1. Introduction 1
1.1 Product Scope 1
1.2 Document Conventions 2
1.3 Intended Audience and Reading Suggestions 2
1.4 References 3
2. Overall Description 3
2.1 Product Perspective 3
2.2 Product Functions 4
2.3 User Classes and Characteristics 5
2.4 Assumptions and Dependencies. 6
3. External Interface Requirements 6
3.1 User Interfaces 6
4. System Features 7
4.1 Hookshot Mechanic 7
4.1.1 Description and Priority 7
4.1.2 Stimulus/Response Sequences 7
4.1.3 Functional Requirements 7
4.2 Games State Manager 7
4.2.1 Description and Priority 7
4.2.2 Stimulus/Response Sequences 7
4.2.3 Functional Requirements 8
4.3 Load System & Basic Map Editor 8
4.3.1 Description and Priority 8
4.3.2 Stimulus/Response Sequences 8
4.3.3 Functional Requirements 8
4.4 Objects List 8
4.4.1 Description and Priority 8
4.4.2 Stimulus/Response Sequences 8
4.4.3 Functional Requirements 9
4.5 Input System 9
4.5.1 Description and Priority 9

4.5.2 Stimulus/Response Sequences 9
4.5.3 Functional Requirements 9
4.6 Physics System 9
4.6.1 Description and Priority 9
4.6.2 Stimulus/Response Sequences 9
4.6.3 Functional Requirements 10
4.7 Collision System 10
4.7.1 Description and Priority 10
4.7.2 Stimulus/Response Sequences 10
4.7.3 Functional Requirements 10
4.8 Rendering System 11
4.8.1 Description and Priority 11
4.8.2 Stimulus/Response Sequences 11
4.8.3 Functional Requirements 11
4.9 Game level design 11
4.9.1 Description and Priority 11
4.9.2 Stimulus/Response Sequences 11
4.9.3 Functional Requirements 11
4.10 Enemy Ai and Interaction 12
4.10.1 Description and Priority 12
4.10.2 Stimulus/Response Sequences 12
4.10.3 Functional Requirements 12

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Product Scope

This will be a pixel game that includes adventure, cave exploration and 2D side scroller. The game will revolve around the player, Mr Hook and his dinosaur buddy, Rex. The main objective is to clear all the game levels to get the sacred artifact lying deep in the cave. Hence, to get through the different game levels, the player will be using our *unique selling point*, a 'hookshot' mechanic. This main feature allows our player to either hook onto walls or cause damage to enemies. Additionally, the game will be a non-linear level based system with a minimum of 3 levels implemented.

Our game will mostly target people in *Millennial* and *Generation Z* which range around the age of 6 to 41. The reason for this is because the majority of them are knowledgeable in games. To support this, an article named "Two-Thirds of Gen Z Males Say Gaming is a Core Component of Who They Are" written by Research Services has shown that *91% (Generation Z)* and *84% (Millennials)* males regularly play video games. Additionally, more than *3 in 4 Generation Z* males watch video games regularly which is a *25% increase over Millennials*. Therefore, we can conclude that they are the learning curve for them will not be as steep as the other age groups.

One of our business strategies is to make sure our players benefit from our game. In order to achieve this, we will be making use of our game's puzzle aspect to carry out problem solving training. Research has shown that this could improve critical thinking, problem-solving and decision-making skills (Ahmady & Shahbazi, 2020). For example, breaking down and analyzing a problem that they have faced in the game. By taking the game features into consideration, they will need to come up with solutions to clear the levels.

Our corporate goal is to increase our overall brand awareness. This can be achieved by marketing our game's unique gameplay that includes an uncommon hookshot mechanic. This allows us to stand out from the game market.

In terms of the user experience, we want our user to be able to fully understand the game to enjoy it. Hence, we will be building a tutorial level to guide our players. At that level, they will learn how the game and hookshot mechanic will work. Afterwhich, the other game level will also be builded with subtle directional hints to guide the player on where to go.

1.2 Document Conventions

Default Font	Arial	
Font sizes	Heading 1	18
	Heading 2	14
	Heading 3	12
	Body Paragraph	11
Standard	Table header	Background is highlighter in light blue and the title will be bolded.
	Italic	To emphasize or highlight on something
	Bolded	Main points
System Features Color code	**Magenta**	Features related to engine building - First priority
	Orange	Features needed to build base levels - Second priority

1.3 Intended Audience and Reading Suggestions

For easy reading, stated in the table below are the various stakeholders and the recommended sequence for each stakeholder:

Type of Readers	**Suggested sequence for reading**
Developers	Document Conventions → Product Scope → Assumptions and Dependencies → Product Perspective → Product Functions → System Features
Documentation writers (Technical writer)	
Designers / Artists	Document Conventions → Product Scope → Assumptions and

Marketing staff	Dependencies → User Classes and Characteristics → User Interfaces → Product Perspective
Project managers	Document Conventions → Product Scope → Assumptions and Dependencies → User Interfaces → Product Perspective → Product Functions → System Features
Testers	Document Conventions → Product Scope

1.4 References

Research Services. (2019, Feb 27). *Two-Thirds of Gen Z Males Say Gaming is a Core Component of Who They Are.* 4A's Agency. https://www.aaaa.org/gen-z-males-say-gaming-core-component-who-they-are/

Ahmady, S. & Shahbazi, S.(2020, Oct 07).*Impact of social problem-solving training on critical thinking and decision making of nursing students.* BMC Nursing. https://bmcnurs.biomedcentral.com/articles/10.1186/s12912-020-00487-x

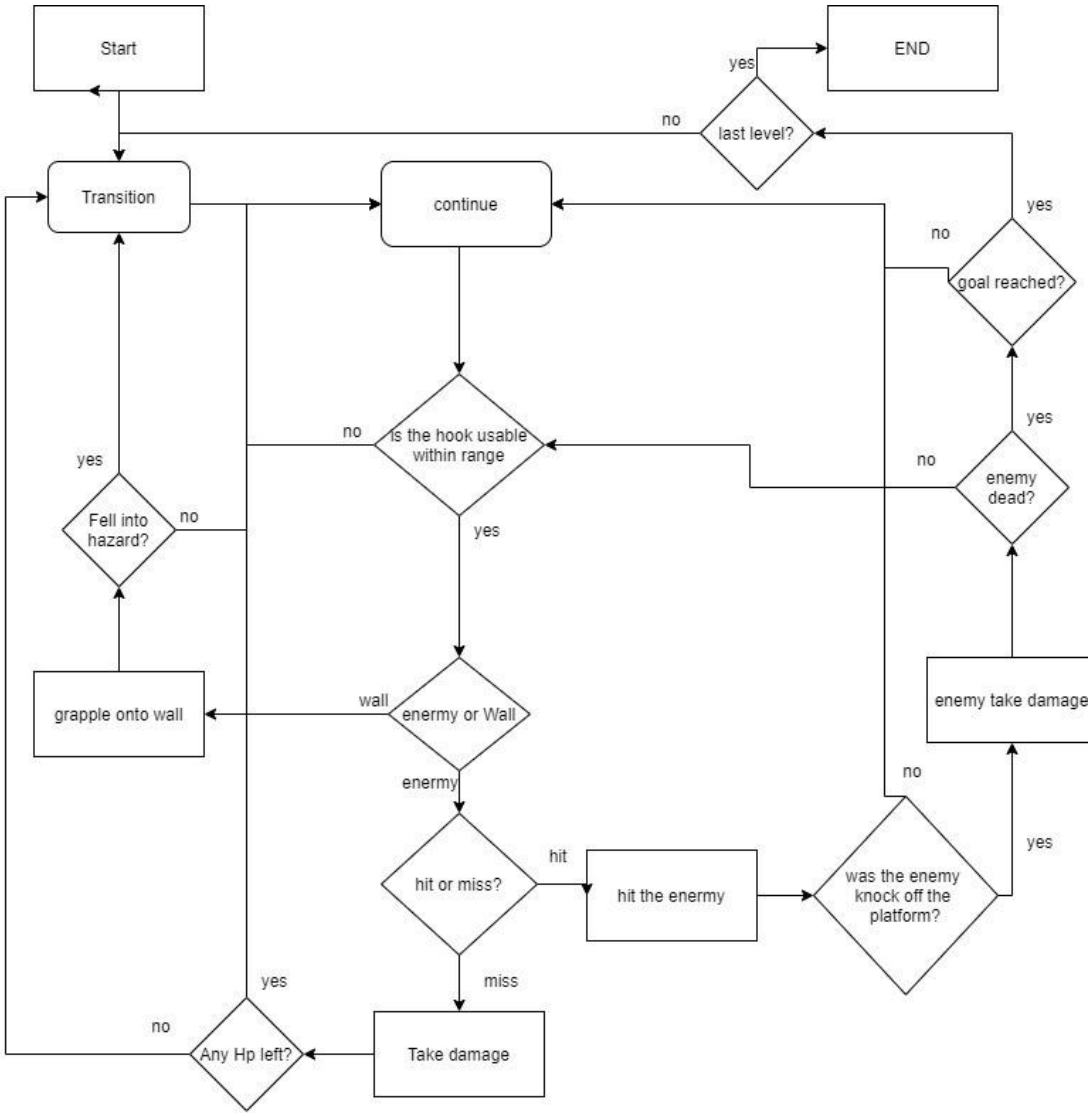
VectorStock. (n.d). *Cartoon game user interface with control elements vector image.* VectorStock. https://www.vectorstock.com/royalty-free-vector/cartoon-game-user-interface-with-control-elements-vector-6317814

2. Overall Description

2.1 Product Perspective

While hookshot is not a continuation or a follow-on member to any existing product, it is based around the 2D platformer genre that already existed. As, most of the well known platformer games usually have a unique gameplay feature tied to them. (Mario jumps, Sonic runs and spin, kirby suck, etc). Hookshot was created based around this unique feature concept which results in the hookshot system of tether and swing.

A simple story idea was proposed, in hope of making the game feel more alive and engaging, while giving a clear goal to the players. The story element can also be used to guide the player along while teaching them any potential new mechanic in the game.



2.2 Product Functions

Listed below are the major functions the game must perform or must let the user perform:

i) Hookshot
The core feature of the game; the hookshot system is a tether and swing system that is done via point and click.

ii) Physics
Physics is another important function of the game as it provides a somewhat realistic or weighted feel to the game.

iii) Pitfall Hazards
A classic death system the player will have to try to navigate across. Falling into it will result in an instant death in both player and enemies, in the case of the player the level will be reseted upon death.

iv) Collision Detection
A system that detects if there is any collision between the player, enemies, hook, wall and ground. This function will return true if there is any collision. Else, it will return false.

iv) Rendering
A generalized system that handles meshes and the drawing of game level by rendering texture and loading source file.

v) Menu System
- The main menu that starts during the beginning of the programme which includes simple UI such as setting and game modes.
- The pause menu is important if the player needs to stop the game halfway. This function may contain the abilities to reset or return to the main menu.

vi) Input System
This system handles user input such as controlling the character to move left, right or jump.

vii) Enemies
The Main obstacle the player will face throughout the game where the player will need to overcome.

2.3 User Classes and Characteristics

- **Casual Gamer**

  Users who have played video games before and have adequate gaming experience and usually play during their free time and only wish to enjoy the game they play. This will be our primary focus as the game is not designed to be played competitively.

- **New gamers**

  Users who have no experience or played any video games in the past. This is one of the least important classes as they are rare, however this should not be overlooked. The first level design will ease them into the game and allow them to be more comfortable with the gameplay without requiring prior knowledge.

- **Speedrunners**

  This user class is the second least important, as the game is still mostly meant for casual gaming. However plans for a time attack mode are being considered for people like speedrunners who wish to have a challenging experience.

2.4 Assumptions and Dependencies.

Our current assumption is that the game's code will be built solely on the alpha engine and the standard library. It should be more than sufficient for us to create the game that we are envisioning as the alpha engine provides almost all the utility that we need for the creation of the game. It is able to provide multiple features such as inputs, math, graphics, time, vectors and more. Although sufficient, we should remain open to using other libraries to aid us in development as we currently do not know the limits of using the alpha engine until we have more experience in it.

As for the game's art, Pixel Studio the android / computer app will be used to draw the sprites.

Extension of the Sprite will be in png with its size being 512 x 512 pix despite its 32 x 32 pix appearance. while animating the sprite everything will be done frame by frame which can be somewhat difficult to do.

3. External Interface Requirements

3.1 User Interfaces

For the Main menu on the title screen 3 button can be found, the first two buttons select the mode the player wishes to play while the third one is the setting where audio and skipping of tutorial can be selected. choices made by the player will be selected primary by mouses. Below is an example of the simplicity that we want to achieve for the main menu.

(VectorStock, n.d)

In the Game portion, The mouse will be the reticle used to fire the hookshot however, there is a limit to how far the hook can go, upon exiting said range the cursor will display an X to indicate that hook is out of range.

Control will be using the Standard 'A' for moving left and 'D' for movement and space for jumps. The player health will be indicated by a 3 heart system, each damage will remove a quarter or more of said heart. A Life will be deducted and level will be reset if the life point reaches zero. Upon reaching zero life, the game will end.

The will be a score system that will increase in point for every enemy killed and every treasure found.

In the event of a time attack mode there will be an additional Clock interface which will timed the player.

Enemy health will not be displayed as this will clutter up the screen with too much UI.

4. System Features

4.1 Hookshot Mechanic

4.1.1 Description and Priority

The hookshot is the heart of the game, which the player will be using to navigate and progress through the game, whether they are crossing Hazards or fighting an enemy, the location of the anchor can be selected but only within the maximum range. Anything exceeding the maximum will disable the hookshot and a X will be used to indicate that the hook is out of range.This will be a *high priority feature* as it is the unique selling point of the game.

This feature is subjected to feasibility <TBD>: While In the air the player can fire the hook again after a few frames of recovery animation, this allows for swing chaining to be performed, helping the player to build up speed.

4.1.2 Stimulus/Response Sequences

Upon left clicking the hook will collide with an object the cursor has selected. If said object is a wall or ceiling, the player will automatically swing forward, if the left mouse button is held, the player will continue to swing back and forth. At this stage the player can choose to lengthen the rop or shorten it with the W and S key respectively. If the key is let go at the correct timing, It is possible to retain the momentum of the swing.

4.1.3 Functional Requirements

<TBD>

4.2 Games State Manager

4.2.1 Description and Priority

The Games State Manager is a highly organized main game loop. It can be used to control the current state of the game. i.e Main Menu, Level_1, Level_2, Restart and Quit. It can also be used to control the game loop, exiting when necessary. This is to ensure that all the necessary functions are called in the correct order. This feature is a high priority as it will be the backbone of the engine.

4.2.2 Stimulus/Response Sequences

When the user first launches the software the game state will first be initialized to the main menu, subsequently when the user completes a level, the next state will then be assigned to the next

level, thereby changing the function calls to match with the level. If the user chooses to restart or quit the game as well, the corresponding state will also be assigned.

4.2.3 Functional Requirements

The requirements for the feature to work well, is a clear and organised list of all the states that can be present in the software, thereby preventing a call to an undefined state. The user should also be unable to call the states themselves but rather an UI should be provided to assist them, preventing any unwanted calls.

4.3 Load System & Basic Map Editor

4.3.1 Description and Priority

The load system will be responsible for reading any external files. i.e Textures, Images, Level_Data so that it can be processed by the engine for use. Integrated into the system as well is a basic map editor that will load in a text file that represents the mapping of the objects in-game. When usage of the asset is no longer required, it will also need to be freed in order to prevent memory leaks for optimal performance. This feature is high priority as it is related to the engine building.

4.3.2 Stimulus/Response Sequences

As soon as the player starts the software, the load system will be immediately executed to retrieve the assets needed for displaying the UI and the Main Menu. When the player then chooses to change states/levels, except restart, the load system will first free the assets being used by the UI and the Main Menu, followed by loading the corresponding assets for the new state.

4.3.3 Functional Requirements

When unable to load an external file, the software should output to the debug console “Unable to open file <File Name> and the program should return to either to the main menu or to an error state. <TBD>

4.4 Objects List

4.4.1 Description and Priority

The objects list will store all the structures of all the objects that will be used for the game. i.e Character, Walls, Enemies. The structure will contain all the necessary variables / characteristics for that specific object to allow for better organisation and manipulation. The feature is a high priority as it is related to the engine building.

4.4.2 Stimulus/Response Sequences

There are no predefined fixed stimulus sequences for this feature as it depends on the user input. The input system will change the variables within the object structure accordingly. For example,

when the user presses left, it will then change the variable named acceleration in the object structure of the character.

4.4.3 Functional Requirements

There are no real functional requirements for this feature as it is just a list of all the objects, perhaps something to take note of is that each object will need to be uniquely named and the name should be clearly representative of the object for clarity.

4.5 Input System

4.5.1 Description and Priority

The input system will be responsible for handling all the inputs made by the user and updating the necessary variables to create the intended action. The input for this game will be done through keyboard and mouse. It includes interacting with both the UI and the gameplay elements. For example using the mouse cursor to select a menu button or moving the character with the WASD keys. This feature is high priority as the product we are trying to create is a game that requires the user's interaction and not a simulation.

4.5.2 Stimulus/Response Sequences

Upon startup, the user will use the mouse cursor to navigate through the main menu, once the game has been started WASD will be used to move the character, Mr Hook. The mouse cursor will then be used again for targeting the hookshot. For more information regarding the hookshot input, refer to the earlier <4.1.2>. The player can use the esc key to either return back to the Main Menu or pause the game.

4.5.3 Functional Requirements

One of the requirements for it to be used properly is that each input should only result in a single action to prevent confusing the player. For example pressing the "W" key should make the character jump and nothing more.

4.6 Physics System

4.6.1 Description and Priority

The physics system will be responsible for all the calculations that are related to physics of the game. i.e acceleration, friction, gravity. As a simple 2D platformer, the physics used will not be following actual real life physics but using a more of a close approximation. In this case, we will not be using mass for calculation and assume that all objects have the same mass for simplicity. The main selling point, hookshot relies on this physics system to calculate the force of the swing and to preserve the momentum as well. This feature is high priority as it is related to the engine building.

4.6.2 Stimulus/Response Sequences

The physic system will only be called during the gameplay-loop, after the input system and the enemy ai and interaction is called. The physics system will take in the updated values and perform the necessary calculations needed for simulating the physics of the game. The values are then passed to the collision system if required, else it will be passed to the rendering system for player feedback.

4.6.3 Functional Requirements

The game's main mechanic, hookshot relies on the physics system for it to feel “right” while playing it. Using the wrong formulas can result in a rigid movement, hindering the flow of the game or possibly cause other problems. Careful attention will be needed for simulating the physics of the hookshot.

4.7 Collision System

4.7.1 Description and Priority

Collision system or collision detection is a form of algorithm or computational problem for detecting 2 or more objects intersecting with one another. There are a few ways of doing it such as Axis-Aligned bounding boxes (AABB) or spheres. This will be a high priority feature as it is involved in engine building.

4.7.2 Stimulus/Response Sequences

For game collision sequences it begins with 2 or more rigid objects moving towards the same points. Once the objects or bodies are intersecting, the distance between these objects will fall below a small tolerance. For hookshot, the collisions will occur in 2 different scenarios, one is when the player is running, hooking and swinging, the other is when the player attacks the enemies and the enemies falls off the map or disappear from the map.

4.7.3 Functional Requirements

3 types of colliders:

1. **Static collider** - use in level geometry, it is a collider that doesn’t move at all through the game regardless of situation. There will be incoming colliders that will collide with static colliders, but the static colliders will never move. Examples of such colliders are brick walls, buildings or platforms. For the static collider in Hookshot, it will be a cave wall, platform ground or a static point where the hookshot can hook.

2. **Passive Collider** - A collider that can move throughout the game. it can be controlled through implementing momentum and physical impact. They do not move by themselves, they are usually targets of forces which means the character can move, push or carry them. They can collide with other colliders such as static and active colliders. Examples of such colliders are boxes, crates, accessories. For the passive collider in Hookshot, it will be hooking devices (character’s tool), pickups(crates / boxes) and artifacts.

3. **Active Collider**- A collider made by using Artificial Intelligence or a programmable way. Examples of this include Input controllers like main character and spawning enemy. The

active colliders in Hookshot will be the main character, Mr Hook and enemies such as spiders, snakes, animals that live in the cave.

4.8 Rendering System

4.8.1 Description and Priority

Rendering system generates animated 2D graphics by program. The resulting image is known as render. It contains geometry, viewpoint and texture. For Hookshot, we will be using the “graphics” functionality of the alpha engine to achieve this. To simplify the process, we will be using a single quad mesh for all the objects if possible. This will be a high priority feature as it is involved in engine building.

4.8.2 Stimulus/Response Sequences

As the player starts opening the software, there should be a rendered display of the game. The rendering system will be called throughout the entire gameplay-loop. For the geometry, it will be made through coordinates (x , y). The colours will be in RGB hexadecimal values. For transparency and texture, the values will be between 1 to 10.

4.8.3 Functional Requirements

The game Hookshot main rendering image starts off with a simple background image of a cave. This is to give the player a feeling of proper adventurous gameplay. There will be use of matrix formulas such as Scaling, rotation and homogeneous coordinates. All this will be done by using AEGfxtriAdd, to create the shape of the characters and assets.

4.9 Game level design

4.9.1 Description and Priority

For game level design, level designers will first sketch the ideas by imagining the playing experience and putting themselves in the position of the player. Afterwards, this will be translated into the engine where the whole game level will be built based on these designs. Function such as rendering will be used to achieve this. This will be a medium priority feature as it is a feature that should be established after the base engine is completed.

4.9.2 Stimulus/Response Sequences

The logic and flow of events and actions have to be thought throughout such as which object to draw first or later. This system will build the game level based on inputs such as the game object location and the type of object to draw.

4.9.3 Functional Requirements

All functions and requirements are only applied after the game idea is being set. So functional requirements are not necessarily needed here.

4.10 Enemy Ai and Interaction

4.10.1 Description and Priority

AI is used to generate responsive and adaptive behaviours primarily in non-player characters such as enemies, similar to human-like intelligence. AI enemies were largely popularized in the form of graduating difficulty levels. Most modern games have pathfinding techniques and decision trees to guide the action of the NPCs. This will be a medium priority feature as it is a feature that should be established after the base engine is completed.

4.10.2 Stimulus/Response Sequences

During a gameplay, once an enemy comes into view. It will first start by moving towards the player and attack. If the player hurts the enemy, the enemy will either disappear from the map or keeps advancing till it's value drops to zero. For Hookshot, enemies will be called throughout the gameplay-loop. There is a total of *3 type of enemy (*Grunt, Elite and Ace type*)* planned and how they will react to the player:

- **Skitter** - Grunt type enemy is the most common type of enemy and has the simplest of AI. They only move forward towards the player and change direction upon colliding with a wall or reaching a cliff. They only have 2 HP, and carry 500 points per kill. They can be easily killed by the player with the hook shot and deal 2 quarter heart damage to the player's health The enemy is designed to look like a 4 legged spider.

- **Hopper** - Elite type enemy Is a lot harder to kill with a total of 4 HP and has the ability to jump over gaps and onto platforms. The design resembles a Cave Frog. they will inflict ¾ heart as damage, and may have a high knockback if they are hit while in the middle of the jump.

- **Titan** - Ace type enemies can be considered a form of a Sub boss, they are the hardest to kill and have the lowest knockback, However there is a weak point that allows the player to deal additional damage. They have a total of 8 Hp and hitting the weak point will deal 2 damage while anywhere else will be the usual 1 damage point. They resemble an Ancient Golem with its weak point being located at the back of its neck. They will fire slow moving projectiles at the player.

Once the player hurts the enemy, the enemy will be knocked back and will either die or keep advancing till the end. Additionally, the player will receive damage if they were to come in contact with any of the enemies.

4.10.3 Functional Requirements

In the games, where NPCs move dynamically around the area, as opposed to predetermined paths, there needs to be some way of moving around the world. Pathfinding or by utilizing some algorithm in order to do so to find a viable path. For hookshot, enemy pathfinder will start from the right side of the screen and end up at the left side of the screen. In order to find a way without walking into solid objects. One solution is to convert the mapo into a 2D grid. Where one of the cells in the 2D grid is either occupied or empty. Empty means the NPC can walk through that cell whereas occupied means that the place is unreachable.