Software Requirements Specification

for

TwinVisitor

Version 1.1 approved

Prepared by Jianna Angeles, Daniel Arias, Vanja Venezia, Lucas Wilkerson

Mushroom People

9/28/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Missing Pieces	10/19/22	Initial Document	1.0
TwinVisitor	11/01/22	Revision for project consistency	1.1

1. Introduction

1.1 Purpose

The product we are developing is a 3D Mystery game called *TwinVisitor*. It will be an offline, single-player PC game. This SRS describes the functional areas, deliverables, and development goals that will guide the process, along with the technical information contained in the SDS.

1.2 Document Conventions

Priorities for higher-level requirements are assumed to be inherited by detailed requirements, other than that there were no other special conventions used in this document.

1.3 Intended Audience and Reading Suggestions

This document is intended for people who wish to gain insight into the development process of *TwinVisitor*. It is also for the developers to track what it is that still needs to be done and plan out the development of the game in a more detailed way.

1.4 Product Scope

TwinVisitor is a 3D Mystery game set in a cyberpunk setting with a spooky and retro atmosphere where you play as a detective and his psychic partner to solve a crime. Our main goal is to make the game entertaining to the player. At this point we plan on publishing the game to Itch.io when we are finished but that is subject to change.

1.5 References

No outside references were directly used in game or asset creation

2. Overall Description

2.1 Product Perspective

Missing Pieces is a new, self-contained product. We are developing it using the Godot engine and using Blender for most of the 3D assets. FLStudio will be used for audio assets.

2.2 Product Functions

The major of the functions of the product are as follows:

- The user can move around the game world (see 4.1)
- The user can switch between the two playable characters (see 4.1)
- The user can interact with NPCs and objects in the game world (see 4.1)
- The user can interact with main menu and pause menu to control game state (see 4.2)
- The user can save and load data to prevent loss of progress in game (see 4.3)
- The user can trigger different scenes to be loaded facilitating game progression (see 4.4)
- The user can examine and use items in their inventory (see 4.5)

2.3 User Classes and Characteristics

The only user class for this product is the user because there is no backend management or administration to be maintained once the game is installed. The users that would be interested in the product are gamers who like puzzle games as well as anyone that enjoys a spooky atmosphere or cyberpunk/supernatural stories.

2.4 Operating Environment

As of right now the software is being targeted at Windows PCs with a moderate possibility of porting it to Mac and Linux.

2.5 Design and Implementation Constraints

We will be using Godot for development so we will be constrained by the limitations of that engine. We will be using Blender for asset creation and as such will be similarly constrained by its limitations. We will also be constrained by the amount of time that we have to work on the project, so we will be focusing on an MVP with stretch goals that will be addressed, time permitting.

2.6 User Documentation

The game will feature an interactive tutorial as a part of the starting scene; this tutorial will communicate basic controls and mechanics to the user. A user manual and a technical manual will be included with the game files. We will also include recommended hardware specifications and relevant health and safety warnings on whichever publishing platform we decide to use, if the project is made publicly available.

2.7 Assumptions and Dependencies

We assume that we will be able to schedule development to finish the bare bones of the project by the end of October and we will have the completed project ready by the end of November.

3. External Interface Requirements

3.1 User Interfaces

The two main user interfaces will be in actual gameplay and menus. During actual gameplay the user will interact with the game using a keyboard and mouse or a gamepad to control the character. There will also be menus such as the pause menu and main menu where the player will still use the same input device to select options on the screen.

3.2 Hardware Interfaces

The game will be developed primarily for keyboard and mouse, but we will be implementing gamepad functionality as well if time permits.

3.3 Software Interfaces

We will be primarily developing the game to run on current Windows machines and will be aiming to distribute it using itch.io.

3.4 Communications Interfaces

This product will not be using any communications interfaces.

4. System Features

Functional requirements across three areas: GAME, MENU, and PAUSE.

4.1 Player Controller [Area: GAME]

4.1.1 Description and Priority

This describes the implementation of the interface that the player will use to control their character. This feature is of the highest priority as the game will be completely unplayable if it fails.

4.1.2 Stimulus/Response Sequences

User Action: The player presses w,a,s, d, or spacebar.
System Response: The player character moves forward, left, backwards, right, or jumps respectively.

User Action: The player presses the interaction button.
System Response: The game checks if there is anything in front of the user that has been tagged as “interactable.” If there is not, it does nothing. If there is, it checks if that thing is an item, NPC, or environment detail and it will prompt the user accordingly.

User Action: The player presses the switch character button.
System Response: Player control is switched to whichever playable character the player is not currently controlling.

4.1.3 Functional Requirements

The software should be able to read player input and handle it by doing the proper functions. If this fails there is nothing that the player would be able to do outside of killing the program from the OS. Valid inputs will do something in the game and invalid inputs should do nothing.

4.2 Game Menus [Area: MENU, PAUSE]

4.2.1 Description and Priority

This describes the implementation of the interface that the player will use to access the menus of the game. This would be the pause menu and the main menu. This is a high priority item as it is the main way that the player can start playing.

4.2.2 Stimulus/Response Sequences

User Action: The player presses esc during regular gameplay.
System Response: The pause menu comes up letting the player exit the game, save, load, or change options.

User Action: The player loads up the game.
System Response: The main menu should appear allowing the player to start the game, exit, load a game, or access options.

4.2.3 Functional Requirements

The software should be able to read player input to allow them to press different buttons on the menus to access different portions of the game.

4.3 Loading / Saving [Area: MENU, PAUSE]

4.3.1 Description and Priority

This feature allows the player to save their progress in the game and load from a previous save point. This is a medium priority item as it will be important for the player to be able to come back to the point that they were at in their previous game.

4.3.2 Stimulus/Response Sequences

User Action: The player presses esc during regular gameplay.
System Response: The pause menu comes up letting the player exit the game, save, load, or change options.

User Action: The player loads up the game.
System Response: The main menu should appear allowing the player to start the game, exit, load a game, or access options.

4.3.3 Functional Requirements

The software should be able to read player input to allow them to press different buttons on the menus to access different portions of the game.

4.4 Switching Active Areas [Area: GAME]

4.4.1 Description and Priority

This feature allows the player to travel between different areas of the game world. This feature of the game is of medium priority as the size of our game is too large for it to just take place in one small area.

4.4.2 Stimulus/Response Sequences

User Action: The player leads their character outside of the starting room.
System Response: There is a loading screen while the current area becomes inactive and the next area becomes active.

4.4.3 Functional Requirements

The software should be able to make different areas of the game active or inactive. This should be achieved using 3D scene nodes in Godot.

4.5 Inventory System [Area: GAME]

4.5.1 Description and Priority

This feature allows the player to pick up and use items and clues that they find throughout the game. This is of medium to low importance as this will be the main way that we will use to create puzzles in the game for the player to have fun figuring out.

4.5.2 Stimulus/Response Sequences

User Action: The player presses the interaction button on an item in the game.
System Response: The player is told what they just picked up and it is put into their inventory for them to use later.

User Action: The player presses the button to access their inventory.
System Response: A screen shows up showing the player what items and clues they have available to them.

User Action: The player uses an item on something in the game world.
System Response: If the item is used correctly then the player is allowed to progress, otherwise they are told that it didn't do anything.

4.3.3 Functional Requirements

The software should keep track of what items the player is currently carrying and it should also be able to check attributes of items so that it knows if they are being used in the correct way.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The game should be able to run without significant frame drops at any point on most modern Windows machines. It should also not be prone to crashes or bugs of any kind that would halt progression.

5.2 Safety Requirements

- Staring at a screen for prolonged periods can lead to eyestrain.
- Flashing lights can lead to seizures for those with epilepsy

5.3 Security Requirements

This game has no components that would require the use of user data at any point and so will never be allowed access to such information.

5.4 Software Quality Attributes

Since the product is a game its quality will be more subjective than traditional software. That being said; player input should be responsive, the puzzles should be neither obvious nor frustrating for the average player, and the entire game should be able to be beaten in under two hours.

5.5 Business Rules

As this is an entertainment product and there are no online components there will be no business rules to follow during gameplay. Outside of gameplay we may decide to implement anti-piracy measures, but this is unlikely to be an issue for a game of this scope.

6. Other Requirements

There are no other requirements that have not been covered at the moment.

Appendix A: Glossary

- **Blender -** Software for 3D modeling. Blender can also be used to create animations, physics simulations, lighting, and textures.
- **Game Engine -** Framework designed specifically for the development of video games. Most Game engines have predefined code to handle physics simulation, player input, graphics rendering, and more.
- **Godot -** Free game engine that is beginner friendly. Development centers around nodes that can be of different types to handle different aspects of the game.
- **Itch.io -** Publishing platform that allows users to upload files that other users can download. The platform is mainly geared for Indie game developers to share their work, but you can upload files of any type to the site.

Appendix B: Stretch Goals

- Publish to Itch.io
- Gamepad support
- Additional content