Initial Requirements

1. Introduction

1.1 Purpose of Product

The purpose of our product is to produce an interactive and enjoyable game for users that implements a consistent storyline, simple controls, a replayability aspect, and reasonable instructions/complexity, all while promoting a personalized experience. In essence, our game will take advantage of users' leisure time and transform it into a captivating, entertaining, and stress-relieving experience. In addition, our product's purpose will stress the importance of a task-driven user experience, meaning that the game will have a well-defined beginning and end.

1.2 Scope of Product

Our product's main scope emphasizes a task-driven user experience, which implies that our game will not be a 'sandbox game' and thus the user will not be allowed to do whatever he or she pleases. That is, the user will have to complete a *timed* task assigned to him or her in order to advance into the next level. As of now, we plan to only include three levels, with each level increasing in difficulty. In terms of user input, it will *only* consist of a finite/small number of keyboard keys. For example, numeric and alphabetic keys will be used by users to enter their name/username, the four arrows and enter keys will be used to control the game character and the user interface, and the numbers "1" and "2" will be used to represent which game character was chosen by the user. Furthermore, the 'ESC' keyboard key will exit the game at any given point in time. It important to note that other keyboard keys (Ctrl, etc.) will be ignored and a mouse listener will not be included in our product's scope, meaning that any user actions involving mouse clicks will be disregarded. Our product will also include two small cutscenes: one at the beginning of the game while another one at the end of the game (assuming that the user actually completes the final level). In essence, these cutscenes will provide context as to why the user is being assigned specific tasks. For example, the opening cutscene will illustrate that the game character was involved in a plane crash, with him or her surviving it and waking up in an isolated island. If the user completes the final level, a closing cutscene will display the game character being rescued by a passing plane that was close enough to see the flare gun signal. There will also be a title screen before the opening cutscene that will display the name of the game. Our game's storyline will be constrained by this time period, meaning that anything that happened before the plane crash or after being rescued will not be in our product's scope. It is important to note that our game will have sounds/music in order to create a sense of suspense while the user is playing it. In terms of product's graphics, we will display simple, 2-dimensional images with an angled backdrop. Therefore, complex 3-dimensional images and graphics are outside our product's scope. In addition, in order to provide a more personalized gaming experience, users will only be allowed to enter their name (with it being displayed at the bottom of the user interface while playing the levels) and will only be able to choose one of the two permitted characters (i.e. the

game will only have two characters: a woman and man). Other more elaborate game personalization options like character customization will be outside the general scope of our product. Lastly, in order to promote our product's replayability aspect, every game restart will put static objects (i.e. rocks, etc.) at a different/random location.

1.3 Acronyms, Abbreviations, Definitions

Some acronyms, abbreviations, and definitions used on this page are defined in the following manner, with a link to their original web source found in the Appendices section:

- Replay Value (Replayability) - *The quality in a video game ... of being suitable for or worth playing more than once. **(5.1)** **(https://www.oxfordlearnersdictionaries.com/us/definition/english/replay-value)***
- SOS - *An international code signal of extreme distress, used especially by ships at sea.* **(5.2)** **(https://en.oxforddictionaries.com/definition/sos)**
- Task-Oriented/Task-Driven - *Focusing on the completion of particular tasks as a measure of success.* ***(5.3)*** ***(https://www.dictionary.com/browse/task-oriented)***
- Sandbox Game - A *... style of game in which minimal character limitations are placed on the gamer, allowing the gamer to roam and change a virtual world at will.* ***(5.4)*** ***(https://www.techopedia.com/definition/3952/sandbox-gaming)***
- Static Entity - *An entity that does not move in the board but has its own associated location and image (e.g. rock, twig, flare gun, etc.).*
- User Controlled Entity - *The actual "game character" that will be controlled by the user and move along the screen.*
- Multiplayer - *When two or more users play the same game together and at the same time. (Note: Our game will not support multiplayer gameplay.)*

1.4 References

There are no further external references needed to understand our project documents, but references to our Problem Statement and Minimum Viable Product pages can be helpful.

- **Problem Statement (https://nmsu.instructure.com/groups/224287/pages/problem-statement)**
- **Minimum Viable Product (https://nmsu.instructure.com/groups/224287/pages/minimum-viable-product)**

2. General Description of Product

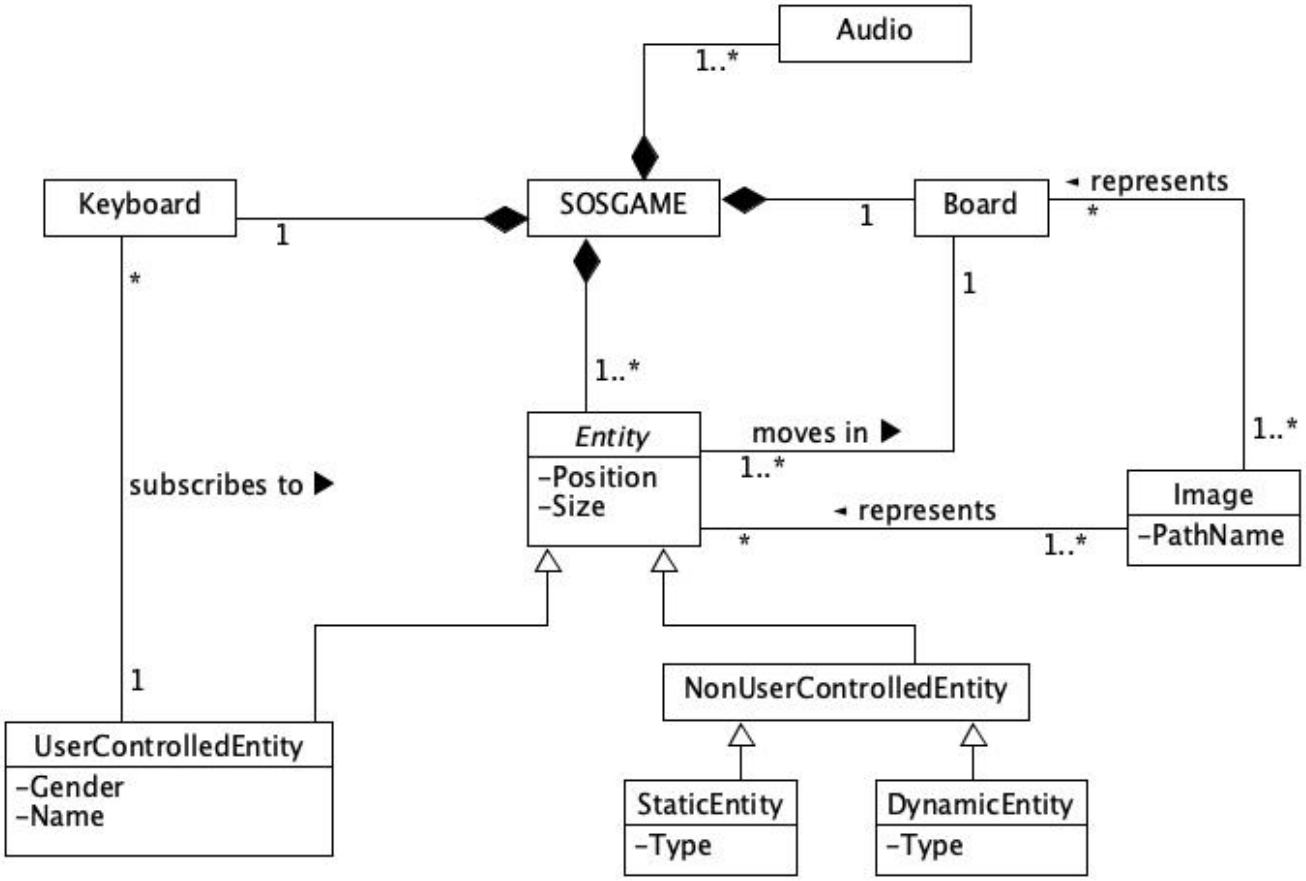
Note: This section will contain a longer but not exhaustive description of our product.

2.1 Context of Product

The context of a product describes the circumstances or environments in which that particular product will be used by users. For our particular product, users will play our game only as a pastime or to relive stress, so our user interface must be simple, but also entertaining.

We assume that the game will also be played by one player at a time, meaning that no multiplayer option will be allowed. We also assume that users will play this game anywhere where they might have both a compatible computer and the required source code.

2.2 Domain Model with Description



Description:

It is important to note that the main "object" that represents our entire project is *SOSGame.* The *SOSGame* object is composed of one *Keyboard* object, several *Entity* objects, one *Board* object, and several *Audio* objects. A *StaticEntity* object and a *DynamicEntity* object are a *NonUserControlledEntity*, and a *NonUserControlledEntity* object is a *Entity.* In addition, a *UserControlledEntity* is a *Entity* object. A *StaticEntity* object can have a type (e.g. rock, tree, apple, etc.), but cannot move along the screen. A *DynamicEntity* object can have a type (e.g. monkey, bee, etc.) and can move along the screen. An *Entity* can have a position (in the map) and a size (pixels wise). One or more *Entity* objects move in the *Board* object while one or more *Image* objects can represent an *Entity* or *Board* object. Lastly, one *UserControlledEntity* (i.e. there is only one player per game since a multiplayer option is outside the scope of our project) can have a gender (male or female, which will correspond to one of the two game characters selected by the user), a name (user inputted name), and is subscribed to the *Keyboard* object.

2.3 Product Functions (general)

In essence, our product will use basic user input through a set of keyboard keys and use this input to manipulate either the game character or the game's interface. An example using user input to manipulate the game character would be to move the character north if the up arrow key is pressed. An example of using user input to manipulate the game's interface would be to start the game once the enter key is pressed by the user during the title screen display. An example of manipulating both the game character and the game's interface would be to move the character on top of an object (e.g. rock, twig, etc.) to "pick it up" and thus preventing it from being displayed again (i.e. once a static object is picked up then it should not be displayed again). It is important to note that the game will use 2-dimensional graphics and will have an "angled-view" perspective. Users will use the keyboard as a user interface in order to accomplish specific tasks assigned to him or her, with each task increasing in difficulty as each level progresses. Because these tasks revolve around finding necessary materials to escape the isolated island, we are also (in essence) enhancing our game's storyline consistency.

2.4 User Characteristics and Expectations

We assume that our users will mostly be composed of game players with a moderate level of computer literacy, which implies that simple instructions on how to use a computer will be outside our product's scope. Because we assume that our users will mostly be composed of game players, we can also infer that they will know how to use a QWERTY computer keyboard effortlessly (i.e. they can find specific keyboard keys both easily and fast). We also assume that our users will have a basic understanding on how to compile and run Java source code (i.e. through a terminal window or a software application like JGrasp or Eclipse). If the user needs more guidance on how to, in essence, run the game, we expect them to refer to the README text file included with our source code since it includes detailed instructions. Furthermore, we expect our users to not change the source code and we also assume that they will not do something that deviates from the game's instructions (e.g. trying to use the mouse to control the character, pressing invalid keys, etc.).

2.5 Constraints

There are various design and process constraints on our system. The following categorized lists describe some of them.

Design Constraints

- Physical Environment:

1. The product will be located and used on any computer (desktop, laptop, etc.) that supports compiling and running Java source code.
2. The game will run only in one location (i.e. the same computer that it was started in).
3. Because pre-existing external software packages like JGrasp or Eclipse only compile and run Java source code, we will only use Java as the standard programming language.

(Note: There are no environmental constraints on our system. In addition, we assume that there are no constraints on power, heating, air conditioning, or size.)

- Interfaces:

1. Input will only come from a discrete number of pressed keyboard keys and NOT mouse clicks. That is, we will only use the alphabet, arrow, numeric (specifically "1" and "2"), ESC, and enter keyboard keys.
2. Output will consist only of graphics/text through the Java applet and sounds/music through the computer's speakers.
3. We need to implement a "publish-subscribe" interface that will emphasize a "cause and effect" gaming experience. For example, users pressing the "UP" arrow key (cause) should move the game character up (effect).

(Note: There are no other systems that our product will interact with, so no other constraints involving input or output will be addressed in our project.)

- Users:

1. Game players with a moderate level of computer literacy will be the main users of our system.
2. Users will know how to compile and run Java source code by using a terminal window or another external software package like JGrasp or Eclipse.

Process Constraints

- Resources:

1. The project will be completed by a total of three people, with each individual having a high-level of computer literacy.
2. Each developer should have a high skill level of Java object-oriented programming.
3. Only one team member is experienced enough to Photoshop all the required images for the game.

- Documentation:

1. A high-level of descriptive documentation will be required and will be stored/located within our Canvas pages and team's repository.
2. Each document should be written in a professional manner and its target audience should be the instructor and other classmates outside our group.

2.6 Assumptions and Dependencies

In terms of a required external software package, our system will only run on a computer that has some sort of Java software that compiles and runs source code (e.g. JGrasp, Eclipse, etc.). It is important to note that we assume that our system will be compiled and run on a computer that supports displaying moderate computer graphics and sounds through a Java GUI applet. In addition, we also assume that our system will only use a discrete number of keyboard keys as user input and will use graphics and sounds as output (i.e. through the Java GUI applet and computer speakers).

3. Functional Requirements

Here is a link to our user story page. It is important to note that a standard requirements document would have a LONG list of functional requirements here, but for our particular project, we will only have a link to our User Stories page. **User Stories Link (https://nmsu.instructure.com/groups/224287/pages/user-stories)**

4. System and Non-functional Requirements

4.1 External Interface Requirements (User, Hardware, Software, Communications)

Our product will have several interfaces that will interact with both the user and the system in order to promote a joyful gaming experience. The following is a list of main interfaces used by our system:

Computer Keyboard (KeyListener): A discrete number of keys from the computer's keyboard (i.e. arrow, alphabet, and enter keys) will be used as user input to control the entire game.

Java GUI Applet: The entire game will be displayed on a Java GUI applet interface.

Computer Speakers (Audio Output and Input Stream): The game will play music and audio through the computer's speakers in order to enhance users' gaming experience.

Observable and Observer: Will allow us to implement our system using the "publish-subscribe" architecture driver. (FUTURE WORK. Time constraints did not allow for the implementation of these classes.)

Specific Requirements:

NF.4.1.1: Any keyboard key pressed by the user that is not a arrow, alphabet, numeric, ESC, or enter key will be completely ignored by the system. That is, user input will *only* consist of this set of keyboard keys.

NF.4.1.2: There will be no MouseListener in the system, so mouse clicks will be completely ignored.

NF.4.1.3: The Java GUI applet must have a total size of 1000 by 680 pixels.

NF.4.1.4: The entire game will be displayed on this Java GUI applet and nowhere else (e.g. no information will ever be displayed on the terminal window).

NF.4.1.5: The game will play background music and sound effects.

4.2 Performance Requirements

Because our product will be a task-driven game, it is important to describe a few performance needs that must be meet.The following is a list of main performance needs that our game must meet:

Smooth Graphics Display: The game should display graphics in a smooth manner (e.g. the character's movement should not be choppy, the opening cut scene should run smoothly, etc.).

Non-Delayed Character Movement: The character should move without any time delay between user input and character movement (e.g. if the user presses the up key, the character will automatically move

up without any time delay).

Character Will Move with Accuracy: The game character will move with a certain level of accuracy (e.g. the character will not be allowed to move outside the Java GUI applet). In addition, the character will be allowed to "pick up" static objects simply by standing on top of them.

Non-Delayed Name Display: The game should display the user's alphabetic key presses as he or she types a string of characters to represent his or her name. Both upper-case and lower-case should be displayed, as well as any numeric number.

Specific Requirements:

NF.4.2.1: The game's graphics should run smoothly and not be "choppy" since this will enhance users' gaming experience.

NF.4.2.2: The time delay between user input and the applet's output will be as minimal as possible (e.g. pressing the enter key in the title screen should immediately start the game).

NF.4.2.3: The game character will move accurately throughout the map, which implies that the Java application will always have a correct approximation of the character's location.

NF.4.2.4: As the user enters his or her name, the applet will display it without any major delay.

4.3 Design Constraints

Our particular game will have a few design constraints since our system will have both pre-defined assumptions and a simple/straightforward user interface design. Most external requirements were discussed in sections 2.5 and 4.1 of this document. Nevertheless, the following is a list of external requirements assumed by our system:

Moderate Computer Literacy Skills: Users will be required to have a moderate level of computer literacy in order to run and play our game

Compatible Computer: We require users to have some sort of computer (desktop, laptop, etc.) capable of compiling and running Java source code, as well as displaying graphics through an applet.

Primary Programming Language: We are required to only use Java as the primary programming language for our system.

Required Keyboard: It is important to note that the computer where the game is run must have a working keyboard in order for our system to successfully "read in" user input.

Responsible Users: One external requirement that is crucial to our design constraints is that we assume that every user that will play our game will be responsible and not "change" the source code on purpose.

Specific Requirements: (Note: Most of these requirements overlap with those constrains mentioned in section 2.5 of this document.)

NF.4.3.1: The product will be located and used on any computer (desktop, laptop, etc.) that supports compiling and running Java source code.

NF.4.3.2: The game will run only in one location (i.e. the same computer that it was started in).

NF.4.3.3: Because pre-existing external software packages like JGrasp or Eclipse only compile and run Java source code, we will only use Java as the standard programming language.

NF.4.3.4: Input will only come from a discrete number of pressed keyboard keys and NOT mouse clicks. That is, we will only use the alphabet, arrow, ESC, and enter keyboard keys.

NF.4.3.5: Output will consist only of graphics/text through the Java GUI applet and sounds/music through the computer's speakers.

NF.4.3.6: Users will know how to compile and run Java source code by using a terminal window or another external software package like JGrasp or Eclipse.

NF.4.3.7: Game players with a moderate level of computer literacy will be the main users of our system.

NF.4.3.8: Users will not abuse the system to do anything illegal or unethical, so game security is outside the general scope of our project.

4.4 Quality Requirements

It important to note that our system is not life-critical since its failure does not poses a threat to life or health. Nevertheless, the following is a list of quality expectations that we assume that our users will have:

Reasonable Expectations: We expect users to have reasonable quality expectations that allows them to have an enjoyable gaming experience. For example, we assume users will expect a game to rarely crash while they are playing it, or they might expect that there is no time delay between user input and character movement.

Product-View Perspective: Users will most likely use the *user view* perspective while evaluating the quality of our product since they will look more at the product from the "outside" rather than from the "inside."

Specific Requirements:

NF.4.4.1: Users will only play the game in their leisure time or as a pastime, so our system must be as simple as possible.

NF.4.4.2: Users will expect the graphics to be as smooth as possible and without any "choppiness."

NF.4.4.3: The game will provide users with a joyful and lighthearted gaming experience with little frustration.

NF.4.4.4: User input must provide a correct output with little to no time delay.

NF.4.4.5: The controls to control both the character and game interface must be as simple as possible.

4.5 Other Requirements

There were other requirements specified in our user stories page that did not seemed to fit perfectly anywhere in the above sections. The following is a list of such requirements:

Consistent Storyline: The opening and closing cutscenes should be sufficient for the user to understand what the game's storyline is.

Background Music and Sound Effects: The game must have some sort of background music and sound effects in order to enhance the gaming experience of users.

Fluctuations in the Game's Environment: The game must manipulate the user's emotions in order to enhance their gaming experience.

Specific Requirements:

NF.4.5.1: The tasks assigned to the user must revolve around both the opening and closing cutscenes (i.e. they must make "sense").

NF.4.5.2: The game must have background music and sound effects that adapt to the current state of the game.

NF.4.5.3: Different tasks assigned to users must manipulate the user's emotions differently.

5. Appendices

We did not really used outside sources to create this document. However, here are some websites that were useful to define some of the words in our project.

5.1 Whats is Replay Value'?:

https://www.oxfordlearnersdictionaries.com/us/definition/english/replay-value (https://www.oxfordlearnersdictionaries.com/us/definition/english/replay-value)

5.2 What is a 'Sandbox Game'?:

https://www.techopedia.com/definition/3952/sandbox-gaming (https://www.techopedia.com/definition/3952/sandbox-gaming)

5.3 SOS Definition:

https://en.oxforddictionaries.com/definition/sos (https://en.oxforddictionaries.com/definition/sos)

5.4 What is Task-Oriented?:

https://www.dictionary.com/browse/task-oriented (https://www.dictionary.com/browse/task-oriented)

5.5 Create a Domain Model Free Using UMLet:

https://umlet.com/ (https://umlet.com/)