**Protoform:**

**An Action Based Platformer**

**Software Requirements Specification**

**Author:** William Chen

Chenergy IndustriesTM

November, 2017

Version 1.0

**Abstract:** This document describes the program requirements for Protoform, a 2-D action based platformer where the player traverses terrain while avoiding and/or defeating enemies. The program interface, functionality, and overall goals will be described.

**Based on IEEE Std 830TM-1998 (R2009) document format**

Copyright © 2017 Chenergy Industries, which is a made up company and doesn’t own any of the art work used. This project and document is developed for CSE 260 to practice writing a design document, and does not serve any other purpose.

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

**1 Introduction**

Throughout the years, iconic games such as Mario and Sonic have helped to establish the genre now known as the platformer. In the most basic version of this genre, the player moves along an established route while jumping over or ducking under obstacles, winning upon reaching the end destination. Additions to the game include power ups, hazardous terrain, and, most importantly, enemies. Along with these additions come considerations such as health bars, attack options, and NPC (non-player character) movement patterns.

In this project, a 2-D platformer incorporating the aforementioned elements will be developed. This project is heavily inspired by the popular Mega Man games, and as such will reflect many of the patterns present in the series.

* 1. **Purpose**

The purpose of this document is to specify exactly how ***Protoform*** will look and play. Members of the development team (game designers, artists, engineers), marketing team, and potential investors are the intended audience of this document. The contents of this document reflect the intended ultimate design of the project, agreed upon by all parties. Readers should learn how the game will look, function, and how to play it by the end of this document.

* 1. **Scope**

***Protoform*** will begin as a basic, bare bones platformer implementing player movement, NPC movement, collision, and simple physics. However, the game may be updated in the future with more content or player experience adjustments, so components will be made as modular and easily modifiable as possible. As such, ***Protoform*** and its functions (including map generation) will be designed with reusable components allowing for any game designer to easily implement changes.

* 1. **Definitions, acronyms, and abbreviations**

**IEEE –** Institute of Electrical and Electronics Engineers, the “world’s largest professional association for the advancement of technology”.

**Framework** – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

**GUI** – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

**Platformer** – A genre of video games where the player moves along a map while avoiding obstacles, winning upon reaching a set destination.

**Protoform** – The name of the platforming game to be developed.

**2-D** – 2 dimensions, as in with only an X and Y axis.

**NPC** – Non-player characters with scripted actions.

**Game Setup Screen** – A game screen that allows the player to progress through the game, selecting the level to play and showing all the levels as they are ordered according to dependencies.

**UML** – Unified Modeling Language, a standard set of document formats for designing software graphically.

**Use Case Diagram** – A UML document format that specifies how a user will interact with a system. Note that these diagrams do not include technical details. Instead, they are fed as input into the design stage (stage after this one) where the appropriate software designs are constructed based in part on the Use Cases specified in the SRS.

* 1. **References**

**IEEE Std 830TM-1998 (R2009) –** IEEE Recommended Practice for Software Requirements Specification

* 1. **Overview**

This SRS will define how ***Protoform*** looks, plays, and operates. This document is not for software design, and therefore will not go into implementation. Section 2 of this document will provide context and specify conceptual design, including game rules and parameters. Section 3 will present how the game interface should be laid out and all program functionality. Section 4 provides a Table of Contents, Index, and References.

1. **Overall description**

***Protoform*** is a platforming, action game for all ages. It is a single player game composed of:

* A player character that can stand, jump, walk, and attack
* Platforms that the player may collide with (stand/walk/jump on)
* NPC’s with set movement patterns that damage the player upon collision
  + Player attacks damage NPC’s if within range

**2.1 Game Description**

The player attempts to escape a dungeon while traversing over obstacles and avoiding or defeating enemies. Touching enemies or falling off the map can damage and potentially send the player back to the start. Ideally, players will be pushed in their timing, pattern recognition, and route planning skills in their journey to the exit.

**Movement**

The player can move left, right, and jump up. Gravity will affect the player, increasing downwards speed in the air. Platforms will provide jumpable points for the player.

**Platforms**

Platforms will be set up for each scene by the game designer beforehand. Players collide and are stopped by terrain, with no other effects.

**Monsters**

The enemies the player will encounter. Colliding with them will damage the player. Several types of monsters will be implemented (names and concepts subject to change).

* **Slime**: A slow moving, relatively small, easily destroyed monster.
* **Skeleton**: A slow moving, medium sized, difficult to destroy monster.
* **Wolves**: A fast moving, relatively small, easily destroyed monster.

**Death**

Monsters and players are killed upon reaching zero or lower health. In this case, monsters are removed from the map, and players are given the option to quit or restart.

**End of the Game**

The player wins and sees a victory screen upon reaching the end of the map.

* + 1. **System Interfaces**

***Protoform*** will need to provide a reusable map generating function to easily make new platform and monster configurations. The game will not feature saves, as the player is expected to complete each scenario in one run. Note that ***Protoform*** is a standalone application.

**Game Over Screen**

**Gameplay Screen**

**Game Setup Screen**

**Protoform Application**

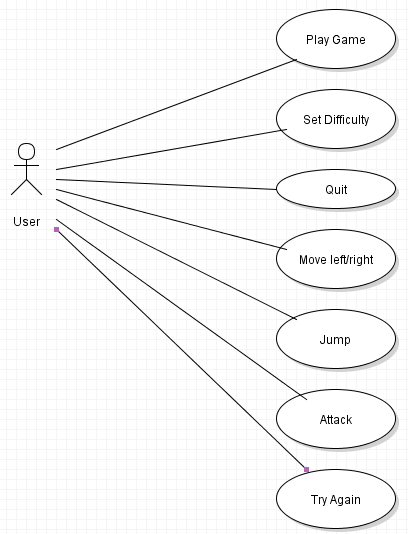
**Figure 2.1: The various screen contexts for *Protoform*.**

* + 1. **User Interfaces**

As this application is not expected to be moved to mobile platforms, key controls will be used to control the character and mouse controls will be used to navigate menus. Figure 2.2 summarizes the ways the user will interact with ***Protoform***, which will be further detailed using a UML Use Case diagram for each.

**Figure 2.2: Overview of Use-Case Diagrams**

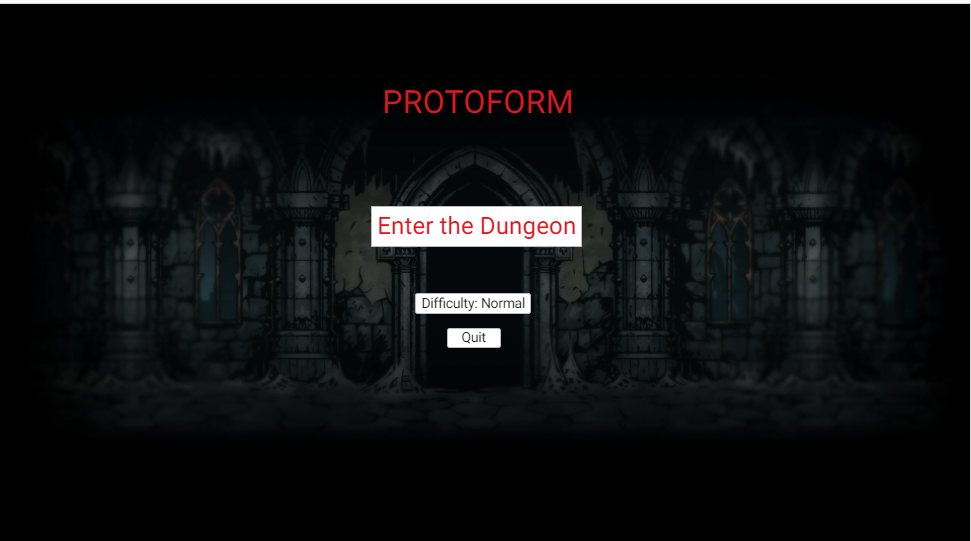
|  |  |  |
| --- | --- | --- |
| **Figure** | **Screen** | **Use Case** |
| 2.3 | Game Setup Screen | Play Game |
| 2.5 | Game Setup Screen | Set Difficulty |
| 2.6 | Game Setup Screen | Quit Game |
| 2.8 | Gameplay Screen | Move left or right |
| 2.9 | Gameplay Screen | Jump |
| 2.10 | Gameplay Screen | Attack |
| 2.3 | Game Over Screen | Try Again |
| 2.5 | Game Over Screen | Set Difficulty |
| 2.6 | Game Over Screen | Quit Game |



**Figure 2.3: Play Game Use Case**

|  |  |
| --- | --- |
| Use-Case: | Play Game |
| Primary Actor: | Player |
| Goal in Context: | Start a new game. |
| Preconditions: | The game application has been started. |
| Trigger: | The player clicks on the “Start” button |
| Scenario: | 1. Player starts the application and is in the Game Setup or Game Over Screen. 2. Player wants to play so presses “Enter the Dungeon” or “Try Again” button respectively. 3. Application renders the player, map, and monsters and loads the Gameplay Screen. |
| Exceptions: | This button should always be enabled from the Game Setup Screen |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Used every time the player starts the application |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Figure 2.4: Game Setup Screen**



**Figure 2.5: Set Difficulty Use Case**

|  |  |
| --- | --- |
| Use-Case: | Set difficulty |
| Primary Actor: | Player |
| Goal in Context: | Change the difficulty of the game (game speed, number of monsters) |
| Preconditions: | The game application has been started. |
| Trigger: | The player clicks on the “Difficulty: [x]” button (where [x] is difficulty) |
| Scenario: | 1. Application is started in the Game Setup Screen or the player has died and is in the Game Over Screen. 2. Player presses the “Difficulty: [x]” button if it is enabled. 3. Button changes to next difficulty in the cycle (easy, normal, hard) and updates next game start accordingly. |
| Exceptions: | This button should always be enabled from the Game Setup and Game Over Screens |
| Priority: | Non-essential, implemented only if difficulty settings are implemented |
| When available: | Third Benchmark |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Figure 2.6: Quit Game Use Case**

|  |  |
| --- | --- |
| Use-Case: | Quit Game |
| Primary Actor: | Player |
| Goal in Context: | Quit and close the game application when the player requests it. |
| Preconditions: | The application has started and is in either Game Setup Screen or Game Over Screen. |
| Trigger: | The player clicks on a “Quit” button. |
| Scenario: | 1. Player is viewing either the Splash or Gameplay Screens. 2. Player clicks on the “Quit” button. 3. Application closes. |
| Exceptions: | None. |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Frequency of use: | Once per session. |
| Open Issues: | Size, location, and style of button should be finalized by UI designer |

**Figure 2.7: Gameplay Screen**

****

**Figure 2.8: Move Left or Right Use Case**

|  |  |
| --- | --- |
| Use-Case: | Move the player left or right |
| Primary Actor: | Player |
| Goal in Context: | Move the player left or right and update the game accordingly. |
| Preconditions: | A new game is in progress (started through the Game Setup Screen or Game Over Screen) |
| Trigger: | The player presses the left or right arrow key. |
| Scenario: | 1. Game is running with Gameplay Screen 2. Player presses the left or right arrow key 3. Player sprite moves in the appropriate direction |
| Exceptions: | This button should always be enabled from the Gameplay Screen |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Open Issues: | Consider how far each tap should move the player; too fast or too slow may impact user experience. |

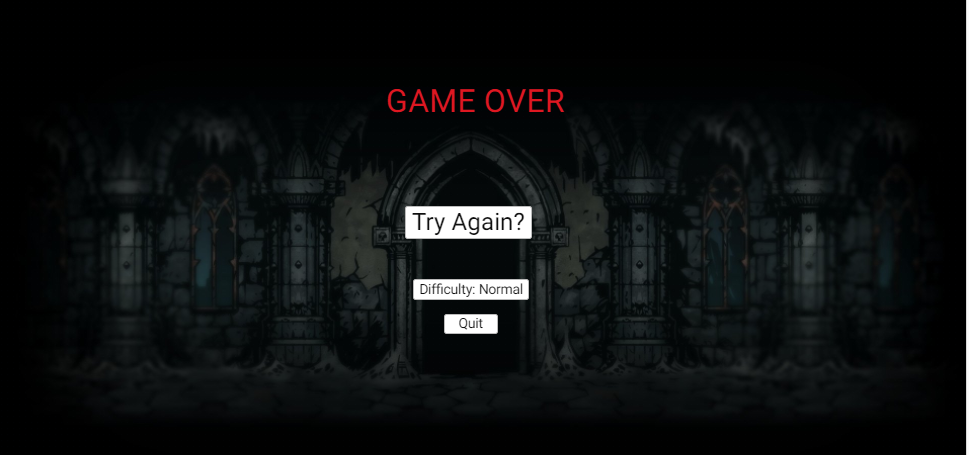
**Figure 2.9: Jump Use Case**

|  |  |
| --- | --- |
| Use-Case: | The player wants to jump. |
| Primary Actor: | Player |
| Goal in Context: | Give the player an upwards speed and update the game accordingly. |
| Preconditions: | A new game is in progress (started through the Game Setup Screen or Game Over Screen) |
| Trigger: | The player presses the up arrow key. |
| Scenario: | 1. Game is running with Gameplay Screen 2. Player presses the up arrow key 3. Player sprite jumps up. |
| Exceptions: | This button should always be enabled from the Gameplay Screen |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Open Issues: | Consider how far each tap should move the player; too fast or too slow may impact user experience. |

**Figure 2.10: Attack Use Case**

|  |  |
| --- | --- |
| Use-Case: | The player wants to attack. |
| Primary Actor: | Player |
| Goal in Context: | Make the sprite perform an attack animation and update appropriate monsters, if any. |
| Preconditions: | A new game is in progress (started through the Game Setup Screen or Game Over Screen) |
| Trigger: | The player presses the “A” key. |
| Scenario: | 1. Game is running with Gameplay Screen 2. Player presses the “A” key. 3. Player sprite attacks and monsters within range lose health. |
| Exceptions: | This button should always be enabled from the Gameplay Screen |
| Priority: | Essential, must be implemented |
| When available: | First Benchmark |
| Open Issues: | Consider how to make attack ranges fair (can be too hard or too easy). |

**Figure 2.11: Game Over Screen**

****

*Appears after the player’s health has dropped to 0 or lower in the Gameplay Screen; buttons have the same functionality as in Game Setup.*

**Hardware Interfaces**

The game will only be built for the PC, with the implementation done in java.

* + 1. **Software Interfaces**

***Protoform*** will be developed using the Java language and the JavaFX library, including the framework for Graphical User Interfaces and 2D graphics. A map creation engine will be built to easily generate new configurations of obstacles. Java version 8.0 will work on all desired platforms with the notable exception of the iPhone.

* + 1. **Communications Interfaces**

This game will operate solely as a local game. There will be no networking requirements.

* + 1. **Memory Constraints**

Since this game will only be built for the PC, there will be no notable memory constraints. The only files required will be the sprites and the application itself.

* + 1. **Operations**

It is the goal of the player to escape the dungeon.

* + 1. **Site Adaptation Requirements**

N/A

* 1. **Product functions**

The game does not need to save multiple accounts data or player settings, only the progress of the current game needs to be saved and loaded. The assumption is that the game is owned by one person and only one person will play it.

* 1. **User characteristics**

These map-based games appeal to people of all ages, and *Journey Through Europe* should be no different. Even young children should find the point and click interface easy to pickup and use. Also note that the game is presented in a humorous context such that young players will find the maps to be harmlessly humorous and entertaining.

* 1. **Constraints**

Creating too many monsters in high difficulty settings may cause performance drops. Note possible design patterns to avoid this.

* 1. **Assumptions and dependencies**

N/A.

* 1. **Apportioning of the Requirements**

This will be a solo project, so I will construct a robust and modifiable program for future updates.

**3 Specific requirements**

***Protoform*** will use a simple interface with few controls. Note that sprites should be clearly drawn to convey their exact location.

* 1. **External interfaces**

The player will only use mouse input to start the game and keyboard input to play it, so the GUI will be straightforward. See Figures 2.4 (Game Setup Screen), 2.7 (Gameplay Screen), and 2.11 (Game Over Screen).

* 1. **Functions**

One of the important things to consider in our game application is providing the appropriate feedback to the user while the game is happening. Players need feedback to enjoy their game experience. This is typically done with visual cues or sound. We will not be employing sound in this project, but we will certainly need to use visual cues.

**Animation Cues**

The entire game will be represented through sprite animations, so animation cues must be crisp and clear to the user.

* **Actions** – Acting players and monsters will cycle through their sprites appropriately.
* **Damage** – All actors should visibly appear hurt when taking damage.
* **Boundaries** – Platform and monster sizes must be clear and obvious to the player.
  1. **Performance requirements**

The primary performance concerns will be with rendering, since it is a real-time graphical application. Note that rendering performance testing should be an important component of the development process.

* 1. **Logical database requirements**

N/A

* 1. **Design constraints**

As mentioned, a primary concern should be efficient rendering. When available, the program should be able to take advantage of hardware (GPU) acceleration for rendering. However, in order to appeal to the maximum number of users, it should also work well when that is not available. This is a reasonable constraint, since this is a 2D game with a small world and little artwork to render.

* 1. **Software system attributes**

As professionals, all members of this project must take this project seriously. We are dedicated to producing robust software that exceeds the expectations of our customers. In order to achieve this level of quality, we should build a product with the following properties in mind:

**3.6.1 Reliability** – The program should be carefully planned, constructed and tested such that it behaves flawlessly for the end user. Bugs, including rendering problems, are unacceptable. In order to minimize these problems, all software will be carefully designed using UML diagrams and a Design to Test approach should be used for the Implementation Stage.

**3.6.2 Availability** –Customers may download and install the game application for free, but will be required to pay a fee to access the full game content, meaning anything past level 10.

**3.6.3 Security** – All security mechanisms will be addressed by future revisions

**3.6.4 Extensibility** – It is important that more levels can be added to the game, so file formats for levels should be carefully considered such that the game can be easily extended.

**3.6.5 Portability** – To start with, the game will target all platforms that support Java. An Objective C port should follow thereafter to make the game playable on the iPhone.

**3.6.6 Maintainability** – Update mechanisms will be addressed by future revisions.

* 1. **Organizing the specific requirements**

Note that the game is simple enough that we need not worry about using an alternative arrangement of the content of this document. The specific requirements for this application already fit neatly into the sections listed in the IEEE’s recommended SRS format.

* 1. **Additional comments**

It is important to keep in mind that the sprites used in the mockup are not final. The UI should be revisited and revised to appear more graphically appealing to the user.

**4 Supporting Information**

Note that this document should serve as a reference for the designers and coders in the future stages of the development process, so we’ll provide a table of contents to help quickly find important sections.

* 1. **Table of contents**

1. Introduction 2
   1. Purpose 2
   2. Scope 2
   3. Definitions, acronyms, and abbreviations 2
   4. References 3
   5. Overview 3
2. Overall description 4
   1. Game description 4
   2. Product functions 11
   3. User characteristics 11
   4. Constraints 12
   5. Assumptions and dependencies 12
3. Specific requirements 13
   1. External interfaces 13
   2. Functions 13
   3. Performance requirements 13
   4. Logical database requirements 13
   5. Design constraints 13
   6. Software system attributes 14
   7. Organizing the specific requirements 14
   8. Additional comments 14
4. Supporting Information 15
   1. Table of contents 15