SE 3XA3: Software Requirements Specification
super-refactored-mario-python

Team 203, Abstract Connoiseurs
Alexander Samaha, samahaa
Daniel Noorduyn, noorduyd
David Jandric, jandricd

April 6, 2020

Contents

1 Project Drivers **1**
- 1.1 The Purpose of the Project . . . 1
- 1.2 The Scope of the Project . . . 1
- 1.3 The Stakeholders . . . 1
  - 1.3.1 The Client . . . 1
  - 1.3.2 The Customers . . . 2
  - 1.3.3 Other Stakeholders . . . 2
- 1.4 Mandated Constraints . . . 2
- 1.5 Naming Conventions and Terminology . . . 3
- 1.6 Relevant Facts and Assumptions . . . 3
  - 1.6.1 Facts . . . 3
  - 1.6.2 Assumptions . . . 3

2 Functional Requirements **4**
- 2.1 The Scope of the Work and the Product . . . 4
  - 2.1.1 The Context of the Work . . . 4
  - 2.1.2 Work Partitioning . . . 5
  - 2.1.3 Individual Product Use Cases . . . 5
- 2.2 Functional Requirements . . . 6

3 Non-functional Requirements **7**
- 3.1 Look and Feel Requirements . . . 7
- 3.2 Usability and Humanity Requirements . . . 7
  - 3.2.1 Ease of Use Requirements . . . 7
  - 3.2.2 Personalization and Internationalization Requirements 8
  - 3.2.3 Learning Requirements . . . 8
  - 3.2.4 Understandability and Politeness Requirements . . . 8
  - 3.2.5 Accessibility Requirements . . . 9
- 3.3 Performance Requirements . . . 9
  - 3.3.1 Speed and Latency Requirements . . . 9
  - 3.3.2 Safety-Critical Requirements . . . 9
  - 3.3.3 Precision or Accuracy Requirements . . . 9
  - 3.3.4 Reliability and Availability Requirements . . . 9
  - 3.3.5 Robustness or Fault-Tolerance Requirements . . . 10
  - 3.3.6 Capacity Requirements . . . 10
  - 3.3.7 Scalability or Extensibility Requirements . . . 10

3.3.8 Longevity Requirements . . . 10
3.4 Operational and Environmental Requirements . . . 10
3.4.1 Expected Physical Environment . . . 10
3.4.2 Requirements for Interfacing with Adjacent Systems . . . 11
3.4.3 Productization Requirements . . . 11
3.4.4 Release Requirements . . . 11
3.5 Maintainability and Support Requirements . . . 11
3.5.1 Maintenance Requirements . . . 11
3.5.2 Supportability Requirements . . . 11
3.5.3 Adaptability Requirements . . . 12
3.6 Security Requirements . . . 12
3.6.1 Access Requirements . . . 12
3.6.2 Integrity Requirements . . . 12
3.6.3 Privacy Requirements . . . 12
3.6.4 Audit Requirements . . . 12
3.6.5 Immunity Requirements . . . 12
3.7 Cultural Requirements . . . 12
3.7.1 Cultural Requirements . . . 12
3.7.2 Political Requirements . . . 13
3.8 Legal Requirements . . . 13
3.8.1 Compliance Requirements . . . 13
3.8.2 Standards Requirements . . . 13
3.9 Health and Safety Requirements . . . 13

4 Project Issues **13**
4.1 Open Issues . . . 13
4.2 Off-the-Shelf Solutions . . . 14
4.3 New Problems . . . 14
4.3.1 Effects on the Current Environment . . . 14
4.3.2 Limitations in the Implementation Environment . . . 14
4.3.3 Problems Resulting from the Solution . . . 15
4.3.4 Possible User Related Issues . . . 15
4.4 Tasks . . . 15
4.4.1 Project Schedule . . . 15
4.5 Migration to the New Product . . . 15
4.5.1 Requirements for Migration to the New Product . . . 15
4.5.2 Data That has to Be Modified or Translated for the New System . . . 16

4.6 Risks . . . 16
4.6.1 Excessive Schedule Pressure . . . 16
4.6.2 Low Quality . . . 16
4.7 Costs . . . 16
4.8 User Documentation and Training . . . 16
4.8.1 User Documentation Requirements . . . 16
4.8.2 Training Requirements . . . 17
4.9 Waiting Room . . . 17
4.10 Ideas for Solutions . . . 17

5 Appendix **18**
5.1 Symbolic Parameters . . . 18

List of Tables

1 **Revision History** . . . i
2 **Symbolic Parameter Table** . . . 18

List of Figures

1 Game Context Diagram . . . 4
2 Gantt Chart reflecting Proof of Concept updates . . . 15

Table 1: **Revision History**

Date	Version	Notes
Feb 4 2020	1.0	Initial changes to the document
April 6, 2020	1.1	Rev 1 Touch-ups

This document describes the requirements for super-refactored-mario-python, which is an implementation of the Super Mario Bros. game in Python. The template for the Software Requirements Specification (SRS) is a subset of the Volere template. This document serves to inform users and stakeholders of the overview of the project including constraints and objectives this project hopes to meet.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of super-refactored-mario-python is to re implement and modify the age old Super Mario Bros. game in an open source environment. The team will finish the work started and left uncompleted, then look to add features ~~that diverge from the original game~~ that round of the base functionality of the original game. The end result is a version of Super Mario Bros. that can be enjoyed by anyone with access to a computer with the proper libraries installed.

1.2 The Scope of the Project

The final version of Super Refactored Mario Bros will be a replica of the original Super Mario Bros game at its base functionality. It will be created using python3 and the corresponding libraries scipy and pygame. The special features such as multiple power-ups and moving pipes, flying enemies, etc. will not be implemented due to time constraints. Therefore the goal will be to organize and/or rewrite the existing code to create a base functionality replica.

1.3 The Stakeholders

1.3.1 The Client

The client for super-refactored-mario-python is the instructor and teaching assistants of the SFWRENG 3XA3 course.

1.3.2 The Customers

The customers of this project are all with an interest in the original Super Mario Bros. game and open-source game development in general. This project will be available for anyone to download if they have access to a computer that satisfies the hardware and software requirements.

1.3.3 Other Stakeholders

The original project owner has a vested interest in the completion of their unfinished project. This may include the integration of the changes and features put forward during the development cycle. Furthermore, the open-source Pygame community will have a vested interest in the propagation of its use to make video games. Finally, the development team also has a vested interest in the success of the project.

1.4 Mandated Constraints

Description: The project shall use the Pygame library for Python game development.
Rationale: The project is originally written with the Pygame library and will make it easier to continue using it.
Fit Criterion: Running the game files with Python includes and runs the Pygame library.

Description: The project shall follow the structure of deliverables outlined in the project schedule.
Rationale: The project needs to follow a logical schedule to ensure completion of the product within the allotted time.
Fit Criterion: The project is completed and all deliverables submitted by April 6th 2020.

Description: The project shall use the Pygame library for Python game development.
Rationale: The project is originally written with the Pygame library and will make it easier to continue using it.
Fit Criterion: Running the game files with Python includes and runs the Pygame library.

1.5 Naming Conventions and Terminology

SFWRENG 3XA3 - The Software Engineering Practice and Experience: Software Project Management course instructed by Dr. Asghar Bokhari.
Super Mario Bros. - The original arcade game released in 1985 for the Nintendo Entertainment System in North America.
Pygame - Popular library for small open-source games written in Python.
Koopa - Refers to a non-playable character in the Super Mario Bros. game that is encountered as an enemy. This character has a shell that can be used as a projectile.
Koopa Shell - The shell of the **Koopa** that can be used as a projectile.
Regular enemy - Refers to a non-playable character in the Super Mario Bros. game that is encountered as an enemy such as a Goomba.
Main Menu - The initial menu where the user can determine what they want to do in the game.

1.6 Relevant Facts and Assumptions

1.6.1 Facts

- Pygame is a set of modules used to create video games written in Python. It encompasses graphics and sound libraries.
- The original repository contains 1390 lines of Python code.

1.6.2 Assumptions

It is assumed that the user has access to a modern computer with the necessary Python interpreter installed along with the Pygame library and is able to run the files. Along with this, the user is expected to have working knowledge of running Python code from the command line. The user is assumed to have a keyboard and a mouse to be able to interact with the game. It is assumed that the user has installed all the required artefacts in a compatible operating system environment.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

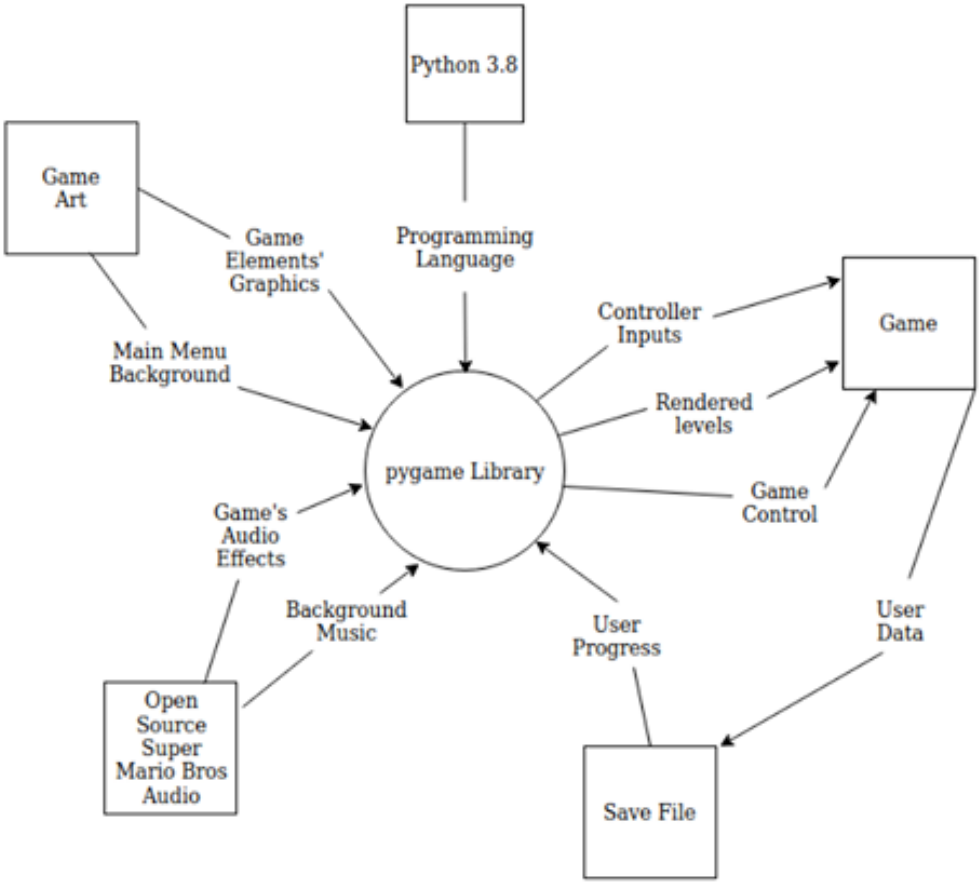

Figure 1: Game Context Diagram

2.1.2 Work Partitioning

Event	Input/Output	Summary
Player action	Player action(IN) Change in game state(OUT)	Respond to the users input by updating the game state accordingly.
Collision	Entity location(IN) Change in game state(OUT)	Update the game state when there is an entity collision.
Save Game	Finish Level(IN) Change in High Score(OUT)	Update the highscore database for every level completion.

2.1.3 Individual Product Use Cases

The primary use case of this product is to be played from start to end, and get the best high-score. There will be a number of levels, and each will have a high score associated with it, for players to challenge themselves and constantly improve their score. Some specific use cases:

1. 'Main Menu': The user controls which option to select using the ARROW_KEYS. The user is able to choose between entering levels, changing sound settings, and exiting the game.

2. Controlling Mario: Once in a level, the user uses KEYBOARD_LEFT, KEYBOARD_RIGHT and KEYBOARD_JUMP to control a player. Holding down KEYBOARD_LEFT or KEYBOARD_RIGHT moves Mario left or right until another entity blocks the way. Pressing KEYBOARD_JUMP causes Mario to jump up.

3. Pausing the Game: While midgame, the user can press KEYBOARD_ESC to freeze gameplay. The user can then use ARROW_KEYS to either return to the 'main menu' or continue playing.

4. Beating a Level: When the user reaches the end of a level, the gameplay freezes and the user's score is compiled. The user is then transported to the beginning of the next level.

5. Beating the Game: When the user reaches the end of the last level, having sequentially beat all previous levels, the user's score is compiled

and the highscore is updated if the user's score is greater than the current highscore. Lastly, the user is transported to the 'main menu'.

2.2 Functional Requirements

FR1. When the user presses ~~the 'W' key~~ KEYBOARD_JUMP, the character shall jump.

FR2. When the user ~~presses the 'A' key~~ holds down KEYBOARD_LEFT, the character shall move left.

FR3. ~~When the user presses the 'S' key, the character shall duck, making the characters size half of what it is.~~

FR4. When the user ~~presses the 'D' key~~ holds down KEYBOARD_RIGHT, the character shall move ~~left~~ right.

FR5. ~~When the character is ducking, they shall be prevented from moving, until they stop ducking.~~

FR6. When the user jumps on top of a regular enemy, they shall die.

FR7. When the user jumps on a 'Koopa', the 'Koopa' shall retract into its shell.

FR8. If the user walks into, or jumps onto, the 'Koopa shell', it shall start sliding across the ground, and will bounce off walls and move in the opposite direction.

FR9. If the player character who doesn't have a power up or enemy is hit by the shell, they shall die.

FR10. If the player dies, they shall be reset to the beginning of the level and a life will be deducted.

FR11. The player shall have 3 lives.

FR12. If the player loses all 3 lives, they shall be returned to the main menu, and their high-score will be saved.

FR13. When the player reaches ~~a 'flagpole'~~ the end of level, they shall win the level, and be transported to the next.

FR14. When the player finishes all levels in sequence, they shall ~~be shown a 'You Win' screen, and~~ be transported to the main menu. Their high-score shall be saved.

FR15. When the user collides with the bottom of a power up box, a Mushroom will pop up out of the top and begin moving left and right resulting in it falling towards ground level.

FR16. When the user collides with the mushroom power up, the mushroom will disappear and the user will changes from 'small Mario' to 'big Mario'.

FR17. When the user is 'big Mario' and collides with an enemy, they shall be transformed back to 'small Mario' and have a period of invincibility.

FR18. When the user collides with the underside of the random box, a coin will pop out of the top and revolve before dissappearing. The user's highscore will be updated accordingly.

FR19. When the user collides with a floating coin in any level, the coin will disappear and the user's highscore will be updated accordingly.

3 Non-functional Requirements

3.1 Look and Feel Requirements

LF1. The product shall look extremely similar to the original Super Mario Bros.
Fit Criteria: ~~Users who are familiar with the original game~~ TEST_USERS shall be able to navigate through the first level within 1 minute and 30 seconds.

LF2. The product shall have a modern looking menu.
Fit Criteria: Users shall be able to recognize all menu options within 30 seconds of booting the game.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

UH1. The product shall be usable for anybody over the age of 10.
Fit Criteria: Users shall be able to start a level within 30 seconds of booting the game

UH2. The product shall have simple, standard controls.
Fit Criteria: Users shall move their character within 10 seconds of starting a level.

3.2.2 Personalization and Internationalization Requirements

N/A

3.2.3 Learning Requirements

UH3. The product shall be playable by a user after reading the in game tutorial.
Fit Criteria: A test group of 20 users shall read the tutorial and play the first level, 90% of the test group shall rate their experience as favourable regarding understanding game controls and rules.

UH4. The product shall be playable by some users without reading the in game tutorial.
Fit Criteria: A test group of 20 users shall read the tutorial and play the first level, 50% of the test group shall rate their experience as favourable regarding understanding game controls and rules.

3.2.4 Understandability and Politeness Requirements

UH5. The product shall use simple language which can be understandable by 95% of users.
Fit Criteria: A questionnaire will be authored and 95% of users shall rate their experience positively in understanding the rules of the game.

UH6. The product shall use standard icons indicative of their function to avoid confusion of the user.
Fit Criteria: The game shall include all standard images of the original game.

UH7. The product shall use a simple font which is easily read by all users.
Fit Criteria: A questionnaire will be authored and 95% of users shall rate their experience as favourable reading through the game menus.

3.2.5 Accessibility Requirements

N/A

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

PR1. The product shall run at 60 frames per second.
Fit Criteria: A play test shall be conducted with a group of 10 testers whose framerates shall be captured through the first level of the game, the framerate shall maintain 60 frames per second 95% of the time under normal conditions.

PR2. The product shall take no more than 3 seconds to load and display all resources.
Fit Criteria: A play test shall be conducted with a group of 10 testers to confirm the game loads within 3 seconds 90% of the time.

PR3. The product shall not significantly slow down if there are 100 entities displaying on the screen.
Fit Criteria: A playtest shall be conducted with a group of 10 testers to confirm the game stays above 60 frames per second 95% of the time.

3.3.2 Safety-Critical Requirements

N/A

3.3.3 Precision or Accuracy Requirements

PR4. The product shall respond correctly to inputs 100% of the time.
Fit Criteria: N/A

PR5. The product shall correctly calculate the high-score 100% of the time.
Fit Criteria: N/A

3.3.4 Reliability and Availability Requirements

PR6. The product shall be able to be run by a user for at least 1 hour.
Fit Criteria: Testing will be done on multiple machines with different configurations, and be run for 1 hour each.

3.3.5 Robustness or Fault-Tolerance Requirements

PR7. The product shall not crash if given many inputs at a time.
Fit Criteria: N/A

PR8. The product shall not crash if there are under 1000 entities visible.
Fit Criteria: N/A

3.3.6 Capacity Requirements

PR9. The product shall be able to handle a single user.
Fit Criteria: N/A

3.3.7 Scalability or Extensibility Requirements

PR10. The product shall be easily modifiable for the addition of features.
Fit Criteria: N/A

PR11. The product shall contain an easy way to add levels.
Fit Criteria: N/A

PR12. The product shall be easily modifiable for the addition of enemies and blocks.
Fit Criteria: N/A

3.3.8 Longevity Requirements

PR13. The product shall run until a major Python or Pygame update.
Fit Criteria: N/A

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

OE1. The product shall run on a personal computer.
Fit Criteria: N/A

3.4.2 Requirements for Interfacing with Adjacent Systems

OE2. The product shall run on computer using Windows 10, MacOS Catalina, and Ubuntu 18.
Fit Criteria: N/A

OE3. The product shall run on a computer with Python 3.7 and Pygame.
Fit Criteria: N/A

3.4.3 Productization Requirements

OE4. The product shall be available on GitLab for download.
Fit Criteria: N/A

3.4.4 Release Requirements

OE5. The product will have a final release in early April, 2020.
Fit Criteria: N/A

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

MS1. The product shall be maintained by the developers until early April, 2020.
Fit Criteria: N/A

MS2. The product shall receive updates to fix bugs.
Fit Criteria: N/A

MS3. The product shall be open source, and may receive updates from other programmers.
Fit Criteria: N/A

3.5.2 Supportability Requirements

MS4. The product shall be supported on OS's mentioned previous.
Fit Criteria: N/A

3.5.3 Adaptability Requirements

N/A

3.6 Security Requirements

3.6.1 Access Requirements

SR1. The products source code shall be available for any user.
Fit Criteria: All downloads of the game shall include all the game files locally for anyone to see.

3.6.2 Integrity Requirements

SR2. All data shall be stored locally on the users machine.
Fit Criteria: N/A

3.6.3 Privacy Requirements

SR3. The product shall not save any personal user data.
Fit Criteria: N/A

3.6.4 Audit Requirements

N/A

3.6.5 Immunity Requirements

SR4. The developers of the project shall not provide malicious links.
Fit Criteria: N/A

SR5. Official versions of the product shall only come from the developers GitLab repository.
Fit Criteria: N/A

3.7 Cultural Requirements

3.7.1 Cultural Requirements

CR1. The product shall not contain any controversial content.
Fit Criteria: N/A

CR2. The product shall use Canadian English spelling.
Fit Criteria: All text in-game shall be run through a spell-checking engine ie. Grammarly.

3.7.2 Political Requirements

N/A

3.8 Legal Requirements

3.8.1 Compliance Requirements

LR1. The product shall not violate copyright.
Fit Criteria: N/A

LR2. The product shall give credits to creators of any original content used by the product, that was not created by the developers.
Fit Criteria: The main screen of the game shall include the names of all the developers and the original owner.

3.8.2 Standards Requirements

N/A

3.9 Health and Safety Requirements

HS1. The product shall not endanger its users in any way.
Fit Criteria: N/A

4 Project Issues

4.1 Open Issues

A specific version of the Apple Mac-Book operation system combined with a specific version of the python language doesn't allow the Pygame library to be executed correctly. Investigation into solutions is currently being performed.

4.2 Off-the-Shelf Solutions

Considering this project is attempting to remake Super Mario Bros (howbeit in a unique way) there are many open source, free versions that have created to attempt something similar. These versions can be referenced to create key aspects of the game that would be shared across any re-creation attempts.

There are also the official version of Super Mario Bros that will be referenced for the creation of characters and certain desirable aspects that this project aims to mimic. However, it is key to note that there will be no code so the official game is more like a creativity mine.

~~Lastly, there are multiple online resources such as YouTube tutorials or coding forums that can help familiarize the development team with new tools.~~

~~The game engine called Pygame will be used to do most of the heavy processing.~~

4.3 New Problems

4.3.1 Effects on the Current Environment

The team cannot think of this project as a replacement for the official Super Mario Bros game, or as an improvement; it is simply a mimic with some fun twists made out of appreciation for the original. When the game is running, the user's personal computer will not experience any lag with any other functionalities. The processor load should be light. The game should not crash on any user's device that fits the requirements for install. Lastly, the game will run the same regardless of which operating system the user is running as long as it is a support OS.

4.3.2 Limitations in the Implementation Environment

Due to the fact that the development team has no experience designing and coding a game utilizing the Pygame library, issues could arise when trying to implement the game. If too many core aspects prove impossible to create it may result in the team having to choose a new library or even coding language to implement the game. This means the team must first attempt to code the base functionality of the game to determine possibility, before adding the fun aspects of the game.

4.3.3 Problems Resulting from the Solution

The resultant project will obviously draw heavily from the official Nintendo game. This may cause any distribution of the project to be a violation of the Digital Millennium Copyright Act.

4.3.4 Possible User Related Issues

A lack of accessible controls for end users with movement restrictions could pose a large issue. This would mean the team must develop alternative control methods with better accessibility. The team must factor this into the decision to publicly distribute the game.

4.4 Tasks

4.4.1 Project Schedule

A link to our project schedule can be found in the project repository linked below: Link to Gantt chart and Resource chart

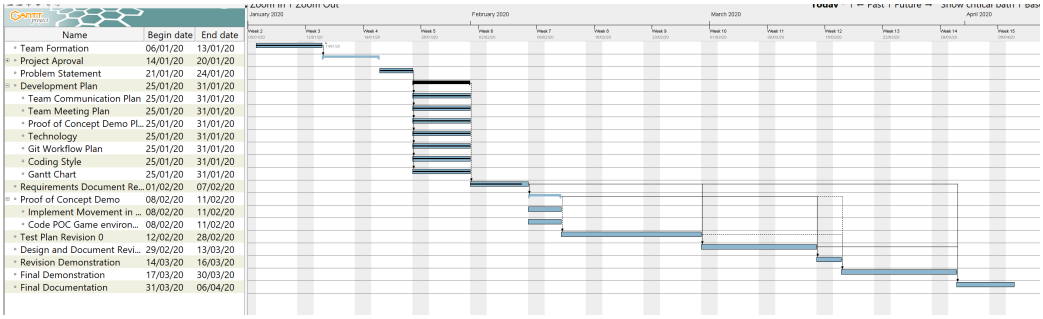

Figure 2: Gantt Chart reflecting Proof of Concept updates

This Gantt chart is updated throughout the project

4.5 Migration to the New Product

4.5.1 Requirements for Migration to the New Product

N/A

4.5.2 Data That has to Be Modified or Translated for the New System

N/A

4.6 Risks

4.6.1 Excessive Schedule Pressure

There are serious time constraints on the project's development period. Therefore, the team be keen to stay on schedule or risk falling behind which would result in the final deliverable not functioning.

Probability: ~~0~~ 15 percent

4.6.2 Low Quality

If the team looses sight of the end goal and becomes focused on trying to create something in-essential to the project, the main functionality of the end result will be full of issues, such as game lag. This could render the game not playable. Thus, the team must complete the core functions before creating fancy aspects.

Probability: 10 percent

4.7 Costs

The entire project will consist of the free to use coding language python and its corresponding free to use library pygame. Therefore, there is no monetary cost to the design and development of the game.

4.8 User Documentation and Training

4.8.1 User Documentation Requirements

The team will create a game controls outline and add it to the game settings. If the team has time, the first game level might come with a run through tutorial to help first time players understand how to play.

If the game is distributed publicly, a Read-Me file will accompany the game with instructions on how to download and install.

4.8.2 Training Requirements

There will be no learning curve; the controls must be intuitive enough to understand without training.

4.9 Waiting Room

A story line that plays out as the user beats levels as well as support for mobile devices will not be released as part of this version.

4.10 Ideas for Solutions

~~N/A~~ The game could also be rendered using a different game engine such as Unity or Cry Engine 3. The highscore could be saved to a server database which would create a ultimate highscore for users to aspire to.

5 Appendix

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Table 2: **Symbolic Parameter Table**

Symbolic Parameter	Description	Value
ARROW_KEYS	Keyboard keys that move the selector on the menus.	Keyboard Arrow Keys
KEYBOARD_LEFT	Keyboard key that moves the onscreen character sideways to the left.	Left Arrow
KEYBOARD_RIGHT	Keyboard key that moves the onscreen character sideways to the right.	Right Arrow
KEYBOARD_JUMP	Keyboard key that moves the onscreen character vertically upwards.	Up Arrow
KEYBOARD_ESC	Keyboard key that pauses the game mid play.	Escape Key
TEST_USERS	A group of users chosen by the development team for their unique background and help round of an age range from 12 to 25. These users will also have ot self identity as familiar with the orginal game.	N/A