Software Requirements Specification

for

Trivia Maze

Version 1.0 approved

Prepared by Kyle Phillips

Pavlo Bilous

Zac Bowman

Kevin Murray

ProCrastinators

6/11/14

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Kyle Phillips	5/15/14	Initial SRS Draft	1.0 draft 1
Kyle Phillips	6/11/14	Final SRS Draft	1.0 Final

1. Introduction

1.1 Purpose

This SRS describes the software functional and nonfunctional requirements for release 1.0 of Trivia Maze. This document is intended to be used by the members of the ProCrastinators project team that will implement and verify the correct functioning of the system. Unless otherwise noted, all requirements specified here are of high priority and committed for release 1.0.

1.2 Document Conventions

Fonts used in this SRS will be times new roman, font size 12 single spaced for easy reading. All higher-level requirements are to be specified, along with what detail requirements will be needed for the higher-level requirement to work properly.

1.3 Intended Audience and Reading Suggestions

This document is meant to be read by the developers of the Trivia Maze game, along with any who are required to know how code and progress is documented. This document should be read in order to get the best understanding of how features and requirements will work together.

1.4 Project Scope

The Trivia Maze game is a simple GUI based game written in ActionScript, a language very similar to the Java language for use with Flash animation. The goal is to create a game where a player can select options from a menu, explore a maze by solving trivia questions pulled from a database table, and either locking themselves in the maze or completing it. The goal is to create a challenging game for trivia lovers to test their skills while problem solving their way through a maze.

1.5 References

1. Capaul, Tom. "CSCD 350 Software Engineering." CSCD 350: Software Engineering. N.p., n.d. Web. 15 May 2014. <http://penguin.ewu.edu/cscd350/Spring_14/index.html>.

2. Overall Description

2.1 Product Perspective

The Trivia Maze project originated as a class project for Tom Capaul's CSCD 350 *Software Engineering* course at Eastern Washington University, Spring Quarter 2014. This is a new, self contained project made for learning purposes.

2.2 Product Features

The major features this product will include allowing a player to choose the type of trivia they want their maze to contain (which will be selected from a SQLite Database), and will then give a GUI representation of the maze for the player to traverse, allowing them to answer questions to delve deeper into the maze to search for its exit.

As a door will lock itself if a question is answered wrong, the player will be give one, two, or three keys depending on the maze's size (set by difficulty). This means there will be a simple inventory system which keeps track of the current keys the player has. The game ends when the player reaches the final room and retrieves their trophy, at which point they will be directed back to the main menu screen.

2.3 User Classes and Characteristics

2.3.1 Trivia Maze Class

This is the main class that drives the game play and refreshes the screen according to the specified FPS. It is in charge of called all other major classes, including the menu, player, etc. Also contains the array which holds the array.

2.3.2 Player Class

This class contains all the functions the player needs to operate, including sprites, checking if the player is stuck in the maze, inventory, maze position and maze movement, as well as listeners for pressing buttons.

2.3.3 DatabaseHandler Class

This class makes a connection to the SQLite database based on what trivia is chosen by the player. It contains methods to create a connection to the database, as well as the ability to query for specific types of data by category and return them to the main class to be created into a maze.

2.3.4 MazeGen Class

This class pulls data from a created DatabaseHandler object and creates a maze from a passed in text file from main that contains a prebuilt maze. The main method here is the convertMap method, which does all the work of converting the text based map into a 2D array and assigning questions to it.

2.3.5 <u>QuestionFactory Class</u>

This class is simply used to create trivia items. Upon creation of the factory object, the factory stores an array of questions pulled from the database, and when instructed will return one of its questions in the form of a new Trivia Item object, erasing this question from its array so that it cannot be used again later while creating a maze, to avoid question duplication.

2.3.6 <u>Room Class</u>

This class contains 4 door objects, one for each major direction (north, east, west, south). It is made to be placed inside of a 2D Maze array, and has methods for creation, sets and gets, as well as all GUI functions needed to display properly.

2.3.7 <u>Door Class</u>

This class is meant to be placed into zero to four 'Door' object spots in the Room class (see 2.3.5). A door contains a trivia question, and has the ability to be open (traversable), closed (must answer a question correctly to open), or locked (must use a key item found in the maze). Also contains all GUI elements needed to function properly.

2.3.8 <u>TriviaItem Class</u>

This class will have 8 major variables: Question, Answer, A, B, C, D, Type, and Category. Type is used to categorize between True/False and Multiple choice, while Category is used to group trivia by what it is related to (IE, Video Game Trivia). For issues we had linking two doors that should be the "same", we have also included a lock feature to the questions in place of a lock feature in the door class.

2.3.9 <u>MenuScreen Class</u>

This class contains all GUI needed to properly display the main menu screens in the game, along with button listening, along with button listening needed to work properly.

2.3.10 <u>QuestionScreen Class</u>

This class contains all GUI needed to properly display the question screens in the game to answer a door's trivia question, along with button listening needed to work properly.

2.4 Operating Environment

This product will be made to run on any OS which can has Adobe Flash installed. All other things needed for the program to run (AIR, for example) are included in the installer.

2.5 Design and Implementation Constraints

The major constraints to this project is the amount of knowledge of the ActionScript language. While many have now gotten used to it, it is still a language which can become complicated. However, most issues have been resolved.

2.6 User Documentation

There will be a text-document based README included with the game, including how to play the game, the authors, etc.

2.7 Assumptions and Dependencies

We are assuming that almost all the code we are producing in Java (in which we are coding and debugging in first) can be easily translated to Action Script. This is the main assumption and dependency we are relying on for the game to work correctly.

We also assume that the user has Adobe Flash installed and updated on their computer to work effectively.

3. System Features

3.1 Main Menu

3.1.1 Description and Priority

The main menu screen is what shows up when the player starts up the Trivia maze Game. It will give the player the ability to play the game, choose a character (and thus the type of trivia), and the difficulty (maze size) that will determine how the maze will be constructed

3.1.2 Stimulus/Response Sequences

Once the player starts the game, they will be prompted with this screen. A simple click and execute is what this menu will look for, and should be obvious for the player when confronted with the screen.

3.1.3 Functional Requirements

REQ-1: GUI interface
REQ-2: Methods to implement the menu any of its options to run properly.

3.2 Trivia Game

3.1.1 Description and Priority

After choosing to play the game and choosing the type of trivia and difficulty, the player will be transported to the maze game, in which they must solve their way through by answering trivia questions related to their theme until they reach the final trophy room. Answering questions wrong may trap them in the maze, and when no keys are available, the player will be prompted a game over screen.

3.1.2 Stimulus/Response Sequences

The player will move with the arrow keys in order to move, and must left click when prompted questions after running into trivia doors and pressing "X" to enter the question screen. These are the main two ways the player will "communicate" with the game.

3.1.3 Functional Requirements

REQ-1: GUI interface
REQ-2: 2D Maze object housing rooms and information regarding what GUI to use.
REQ-3: Room objects to fill the maze; contain up to 4 trivia door items which the player must answer correctly to walk through.
REQ-4: Trivia door object, has 3 states (open, closed, locked), and contains a question object with information regarding trivia questions.
REQ 5: Trivia item (Question, Answer, A, B, C, D, Type, Category) which contains information used to ask the player questions and to determine if the answer is correct or not.
REQ 6: Database handler, see section 3.3.

3.3 Database Handler

3.1.1 Description and Priority

As the program will pull information from a database for trivia questions, scores, etc., there must be a database handler to deal with writing and retrieving data from the correct tables.

3.1.2 Stimulus/Response Sequences

This will not be a player feature directly, but will be used extensively in the creation of the maze, as well as when the player wishes to view high scores or write their score into the database.

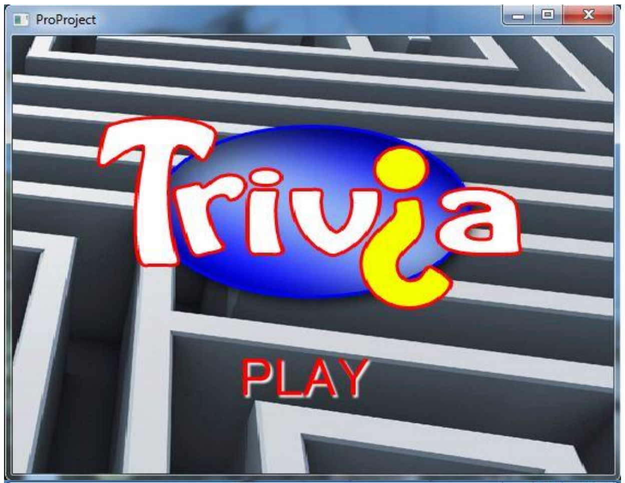
3.1.3 Functional Requirements

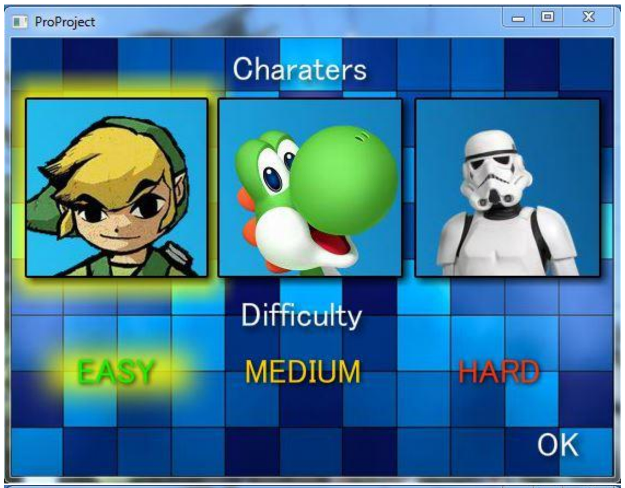
REQ-1: Database must be created using SQLite, as per instructions.
REQ-2: Must be 1 table at the very least: Trivia Question [Question (String), Answer(Character), A (String), B (String), C (String), D (String), Type (Int[1=T/F, 2=MC], Category (String)].
REQ-3: Should be options in the class to get data of specific categories from the table and any other queries that may be needed during maze creation.

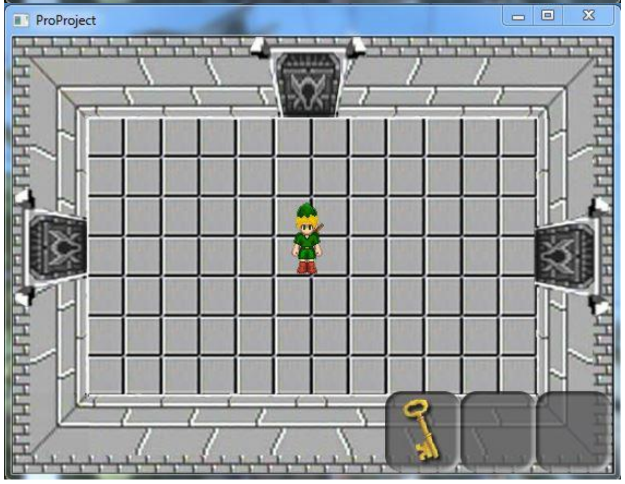
4. External Interface Requirements

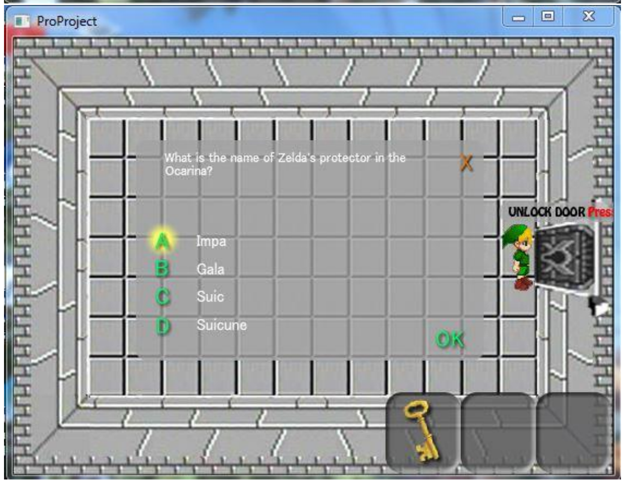
4.1 User Interfaces

The main user interfaces (Main menu screens, Trivia Game, etc.) will all be done using Action Script and Flash animation as a GUI (No WIP previews available). Standard buttons include arrow keys for directing player movement through the maze, as well as the left and right mouse buttons to select answers from a list of possible question answers (while "solving" a door).



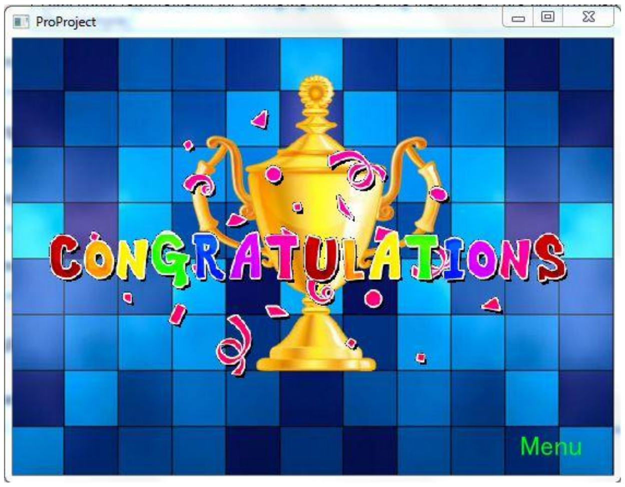


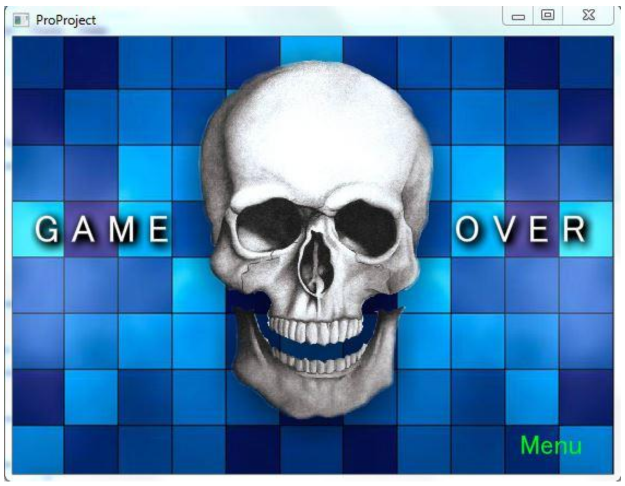




What is the name of Zelda's protector in the Ocarina?

A Impa
B Gala
C Suic
D Suicune





4.2 Hardware Interfaces

No hardware interfaces have been identified.

4.3 Software Interfaces

No interfaces were implemented during the construction of this software, and it works with all included classes alone and with no other specific software.

4.4 Communications Interfaces

The Trivia Maze will not communicate with anything/ anyone outside of the program itself.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

As this will be a low-stress game for a system, there are no known performance requirements for a system at the moment.

5.2 Safety Requirements

The program will create all the data it needs to complete its task without delving into a user's system, thus there should be no safety requirements needed.

5.3 Security Requirements

As this will be a non-networkable game or no saving/loading feature to save past data, there will be no login information needed or security for player data. The only data which is saved will be high scores, which will be stored in a database. All variables and/or methods not needed directly by any other class will be declared private and/or protected to protect any data from being changed on accident.

5.4 Software Quality Attributes

The software will be designed to be used with Windows systems and windows systems alone, for the time being. All code will be checked off by the team for correctness as per coding standards, as well as spelling and grammar when used in the GUI.

If the code meets these criteria, it should be fairly easy to be maintained, and used in the future to be adapted to other systems should the product expand from its original system.

6. Other Requirements

For the database to work correctly, it must be done in SQLite, and must have the following scheme:

Table name: trivia_item

Attributes:

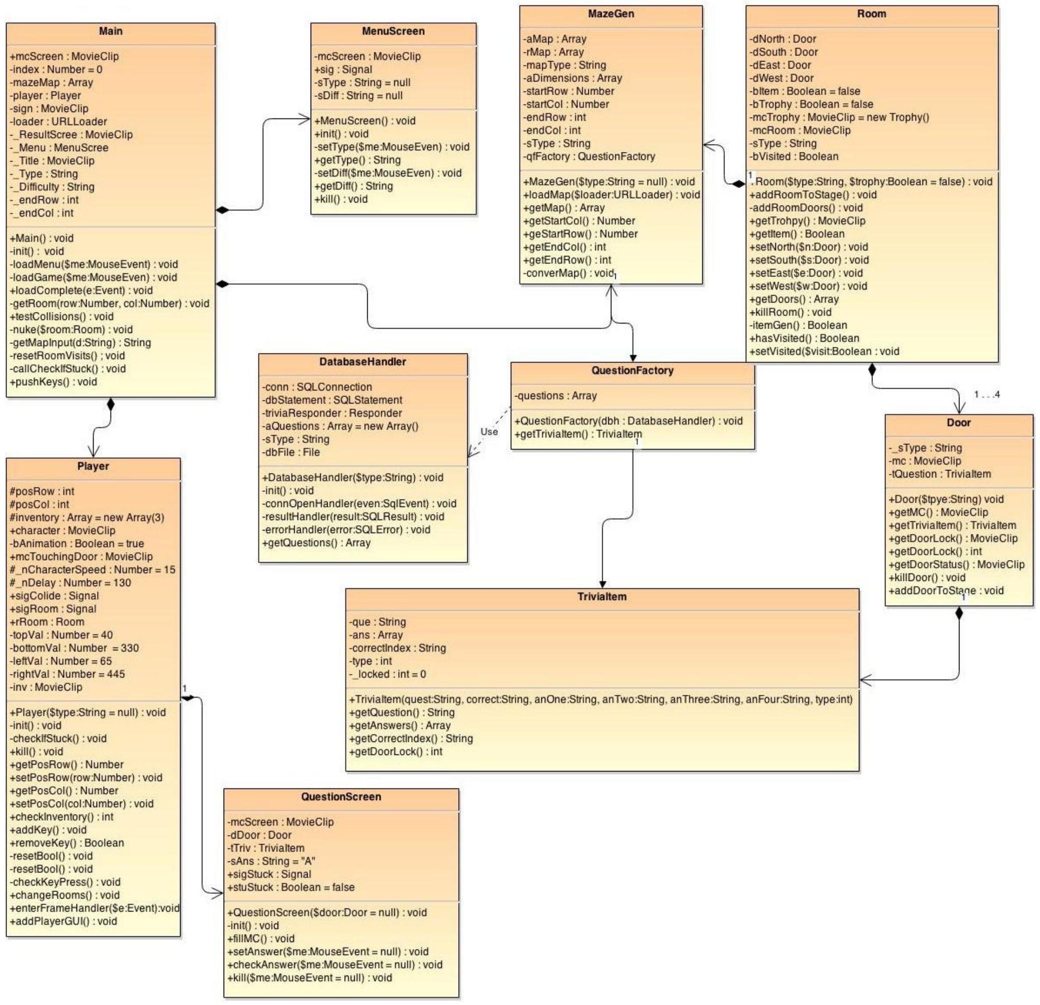
question	(varchar) (non-null)
answer	(varchar) (non-null)
a	(varchar) (non-null)
b	(varchar) (non-null)
c	(varchar)
d	(varchar)
type	(integer) (non-null)
category	(varchar) (non-null)

Appendix A: Glossary

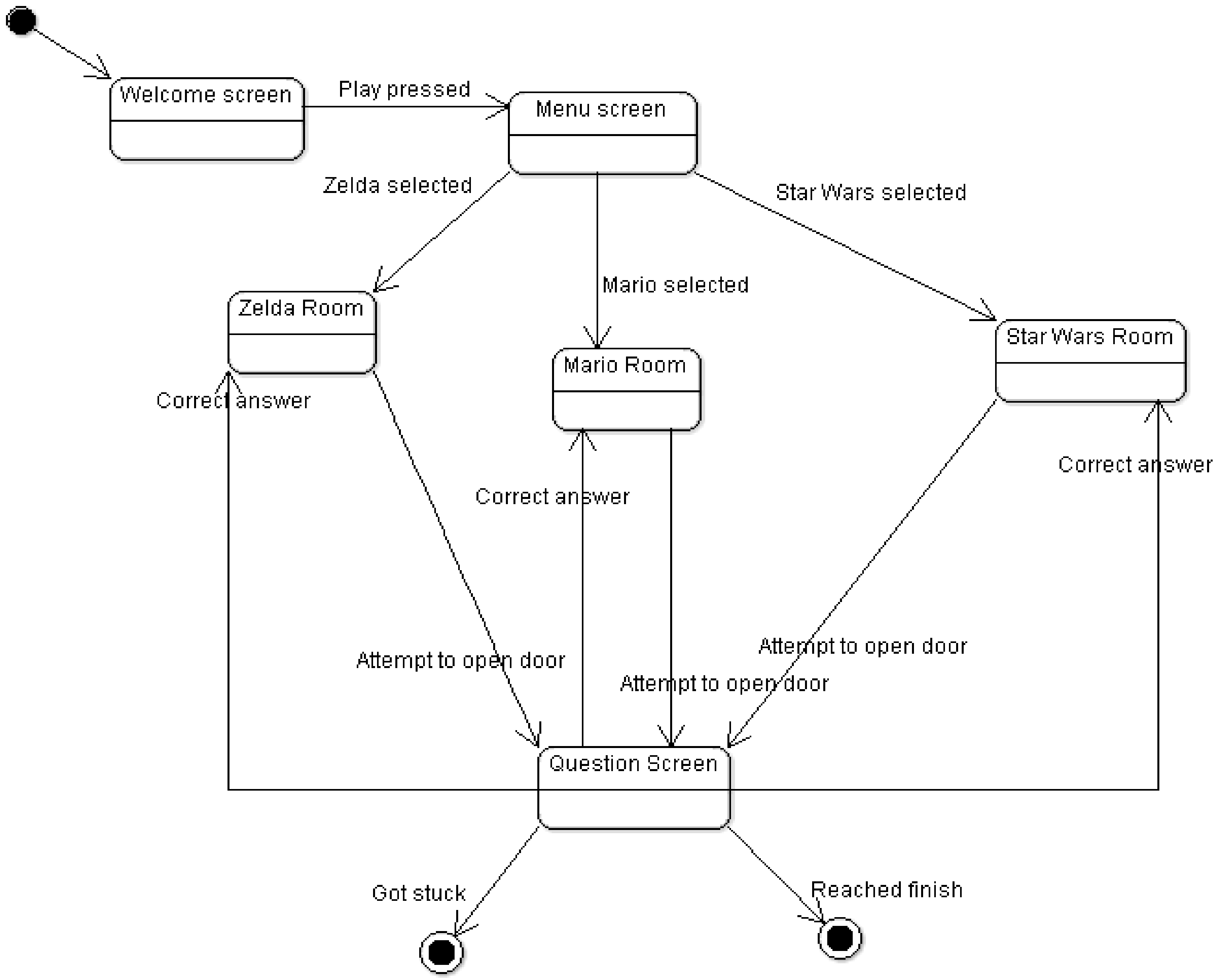
No extra details are needed to interpret this SRS document.

Appendix B: Analysis Models

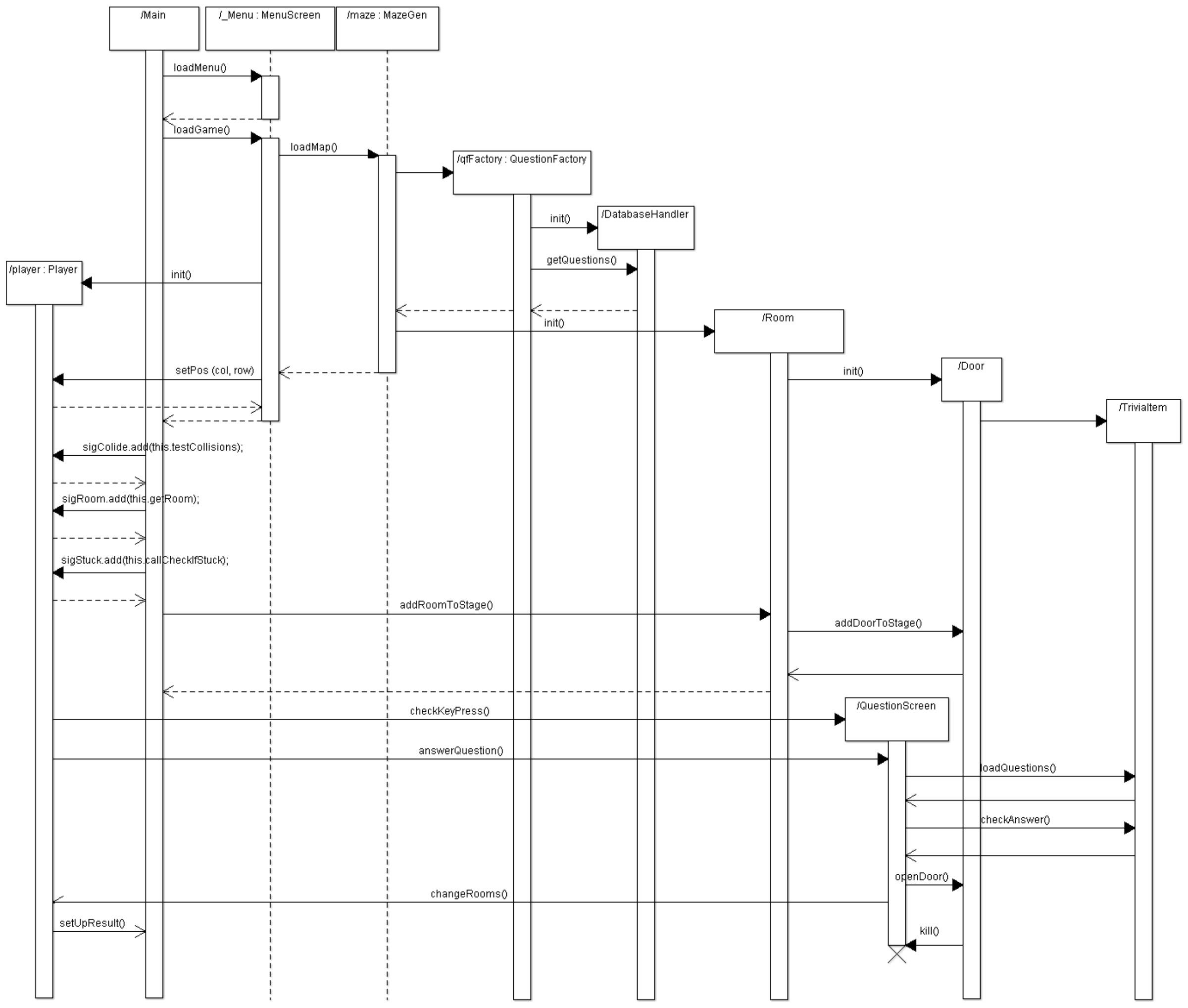
Class Diagram:



State Diagram:



Sequence Diagram:



Appendix C: Issues List

Main current bug is that the sign displaying the status of a door does not update properly until a player moves away from a door and approaches it again. Can be solved by moving the player for a single frame to the center of the room and back again, however this poses a harder bug to fix than we realized. Not game breaking, but annoying to say the least.