Software Requirements Specification

for

Trivia Maze

Version 1.0 approved

Prepared by Kyle Phillips

Pavlo Bilious

Zac Bowman

Kevin Murray

ProCrastinators

5/15/14

Table of Contents

Revision History

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

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 completing the maze to have their score added to a high scores database table. 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 enter their name, keep track of a player's score, allow 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 it's exit.

As a door will lock itself if a question is answered wrong, the player will be allowed to find keys around the maze to continue their quest. 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 end, at which point the name and score will be saved into a database for future access in a "high scores" option in the main menu.

2.3 User Classes and Characteristics

2.3.1 Trivia Maze Class

This is the main class, which will have control over the three major classes: player, maze, and database handler; these will be described below.

2.3.2 Player Class

This class controls all attributes which a player may need, the major ones being the player's name, player's score, and the player's position in the maze, along with all associated methods to set variables, get variables, etc.

2.3.3 Maze Class

The Maze class is in charge of creating a 2D array maze, which contains rooms. The shape and paths for the maze will be determined by a CSV file created by hand or in Microsoft excel, and will be stored in a database table.

2.3.4 Database Handler

This class will gain access to the main database, which will store score information, trivia question information, as well as maze structure information needed to make the program run.

2.3.5 Room Class

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.

2.3.6 Door Class

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).

2.3.7 Trivia Question Class

This class is created by information stored within the database containing trivia information. 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).

2.4 Operating Environment

This product will be made to run on Windows 7 and Windows 8 machines, using the Action Script language and GUI.

2.5 Design and Implementation Constraints

Possible issues with development in this project are the developer's lack of knowledge with the Action Script language; though it is very similar to Java, there will be some learning that will need to be done to translate Java code over to Action Script.

Similarly, there is little knowledge with SQLite. Some learning will need to be done to create and access databases using Action Script and SQLite together.

There should be little in the way of hardware limitations, as this will be a small, low-stress program.

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.

3. System Features

3.1 Main Menu

<Don't really say "System Feature 1." State the feature name in just a few words.>

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 select "New Game", "High Scores", or "Quit", each doing their respective jobs.

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 and it's 3 options.

3.2 Trivia Game

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1 Description and Priority

Clicking "New Game" allows the player to select what type of trivia they wish to fill the maze with, which will also change the GUI theme of the maze and sprites (theme change is not a high priority). The player must walk around the maze into different rooms, answering questions correctly, picking up keys to unlock locked doors, and find the exit.

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. 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

<Don't really say "System Feature 1." State the feature name in just a few words.>

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 2 tables at the very least: High Score [Player name (String), Score (int) and 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, as well as top scores, etc.

4. External Interface Requirements

4.1 User Interfaces

The main user interfaces (Main Menu, High Scores, and the Trivia Game) 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).

4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

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

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>