Olde Worlde Phunne – Maze Game

Design Specification 29/07/2019

Version 1.0

Contents

[1. Introduction 3](#_Toc15573605)

[1. Requirements 4](#_Toc15573606)

[1.1. Maze Structure: 4](#_Toc15573607)

[1.2. Game Items 5](#_Toc15573608)

[1.3. Overall Program Flow: 6](#_Toc15573609)

[1.4. In-Game Program Flow: 7](#_Toc15573610)

[2. Mock-Ups 9](#_Toc15573611)

[2.1. Enter name 9](#_Toc15573612)

[2.2. Read configuration files 10](#_Toc15573613)

[2.3. In-Game Loop 11](#_Toc15573614)

[2.4. Game Finished 12](#_Toc15573615)

[3. Configuration File Data Formats 13](#_Toc15573616)

# Introduction

Olde Worle Phunne games requires a new video game for it’s website to attract more visitors. As per the provided design documentation, this game shall be a simple maze-based adventure game in which a a Maze is generated for the user from a set of values in a configuration file, the single player character is able to progress through a series of rooms in this maze, collecting treasure and defeating threats on the way in an attempt to reach an exit point with the most treasure possible.

As the purpose of this game is to increase traffic to Olde Worlde Phunne’s website, and considering the week long development timeframe, the game shall be simple, accessible and easy to play for users. As no format or interaction style has been specified for the game, it is assumed in this document that the initial version of the game shall be a command-line base text adventure, in which the user inputs commands and the game responds with text explaining the updated state of the game. It is also assumed that the game shall be built using the C# programming language and the .Net Core 2.2 framework from Microsoft.

With these points in mind, the following requirements have been derived from the Project Definition document.

# Requirements

Ranking Column scale: **Essential (E), Non-Essential Desired (NE-D), Non-Essential Nice to Have (NE-NTH)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Maze Structure: | | | | | |
|  | A Maze shall consist of a set of any number of rooms, based on a configuration file value. |  |  | **E** |
|  | A Room shall have a number of connecting passages (between 1 and 4 on each room), with all exit passages connecting to another room, except for the final exit passage which shall exit the Maze. | Passage directions are:   * North * East * South * West |  | **E** |
|  | Each passage that is not the exit passage shall connect two Rooms, which might be the same Room. |  |  | **E** |
|  | A Passage shall be bi-directional, allowing the player to move back and forth between the same rooms. |  |  | **NE - D** |
|  | There can be only one exit passage in the Maze. |  |  | **E** |
|  | A Room shall contain a number of interactable Items. | Items consist of collectables, such as money, and enemies. It is a design decision, but for the purposes of this development phase, the maximum number of Treasures and Threats in any one room shall be 4. |  | **E** |
|  | One room within the Maze shall contain the final exit point of the Maze, which is the end-goal of the game. |  |  | **E** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Game Items | | | | |
|  | An Item within a room shall be one of two types – Threat or Treasure. |  |  | **E** |
|  | Each Treasure item shall have an integer property representing its value |  |  | **E** |
|  | Each Treasure item shall be collectable by the player. | Collection of the Treasure shall remove it from the game. |  | **E** |
|  | The player’s total amount of collection Treasure shall tracked by the Player object. |  |  | **E** |
|  | The player shall be able to perform actions upon a Threat item in an attempt to remove the threat item from the game. Only one action shall be successful in removing a Threat. | Successful actions performed against a Threat shall remove the Threat from the game. |  | **E** |
|  | Attempting unsuccessful actions on a Threat or attempting to leave a room before dealing with a Threat shall result in the Threat removing a value of Treasure from the Player’s total treasure count, if they have any Treasure. |  | MS – Added as I think this will make the risk/reward design of attempting to defuse Threats more interesting. | **NE - NTH** |

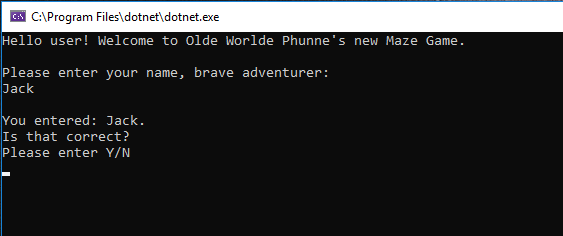
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Overall Program Flow: | | | | |
|  | The User shall be able to start the game executable file. |  |  | **E** |
|  | The User shall be able to enter their chosen player name. |  | MS – Added as a game-design element to make the dialog more interesting. Not essential. | **NE – NTH** |
|  | The program shall be capable or reading and parsing the contents of all configuration files. |  |  | **E** |
|  | The program shall alert the user of any errors encountered whilst reading and parsing the configuration file. | Error messages should specify the type of error encountered. For example, when attempting to parse data from a file, but the file isn’t present, the error message should specify this:  “Attempting to read file at {filepath}. File not found. Please make sure the filepath is correct, or the file can be found in the correct location.” |  | **E** |
|  | The program shall generate a Maze based on the values in the configuration files, including the Maze Seed value which will be used to seed the random number provider. | Providing the same seed value in the configuration file should generate the same maze layout and content. | MS – I intend to use some elements of randomisation for the Maze generation algorithms, to make the game more interesting. This should be done using a seed, so that the user can recreate their previous mazes should they wish to attempt them again. | **E** |
|  | Once a new Maze has been generated, the User shall be able to start a new instance of a game. |  |  | **E** |
|  | The user shall be able to restart their game-instance at any point during the course of the In-Game loop. |  | MS – Ideally, the game would take the User back to the Maze creation screen to be able to reseed their Maze, but initially it may be easier to set up a simple restart that clears all values. | **E** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In-Game Program Flow: | | | | | |
|  | On beginning the in-game loop, the game shall present the user with introductory text. | The introductory text shall be a short description of the scenario the player is in, and their goal. | MS – Added for usability. | **NE - NTH** |
|  | The user shall begin the game from any one of the generated Maze Rooms. | The potential Maze Rooms that the user can initially start the game from shall not include the final room that contains the exit passage. | MS – I don’t think it is a good design possibility to allow the user to begin the game from the room with the exit passage. Added comment in addition details to explain that this shall not be the case in the design. | **E** |
|  | The game shall present the user with a list of available user-input commands at all points the user is able to perform a command. |  | MS – Providing the commands every time the user is able to perform an action will prevent confusion about what the user is able to do. | **NE - D** |
|  | If the user enters a command that is not recognised, the game shall inform the user that their command is not recognised, and present the user with the command list. | The message following an incorrect command should clarify what was incorrect about the command. For example: “{entered command} Command not recognised.” |  | **E** |
|  | On user entry of a successful command, the game shall perform the appropriate action and provide the user with feedback on how their scenario has changed. |  |  | **E** |
|  | On entering a Maze Room, the game shall present the user with a description of the room they are currently in. | The description of the current room may include the following hints:   * Whether the room they have entered is closer to the exit point of the Maze or not. * Whether they have dropped a marker coin in this room. | MS – This is something I’ve added as I think it might help the user to understand if they’re moving towards the exit point or not. | **NE - D** |
|  | The User shall be able to drop Treasure in their current room, as long as they have the available Treasure to do so, and view this treasure upon returning to the room | This is intended to allow Users to mark rooms they’ve already been in. The game should tell the user of a coin’s presence when they re-enter the room. | MS – I’ve made this non-essential as it isn’t critical to the game working, or being able to complete the game. | **NE - D** |
|  | The user shall be able to progress to different rooms through the use of passages. | Upon attempting a passage that doesn’t exit, the game shall tell the user “You may not go that way.” |  | **E** |
|  | If the user does not clear all Threats from a room before attempting to leave, they shall be prevented from leaving the room. | The User shall also lose Treasure from this instance. | MS – For future developments, we could look at adding a health systems, separate from the Treasure collection system. | **E** |
|  | The game shall track the total number of moves that the User has made during the course of the In-Game loop. |  |  | **NE - NTH** |
|  | If the user reaches the exit point of the maze, the game shall present a summary of their progress | This summary shall show the number of rooms traversed, as well as the amount of money the player has accumulated. |  | **E** |
|  | The Maze Game shall have a fail state, in which the user has died before reaching the end of the maze. |  | MS – this is not outlined in the requirements documentation, but is instead my idea for a potential future project improvement to make the game more interesting. | **NE – NTH** |

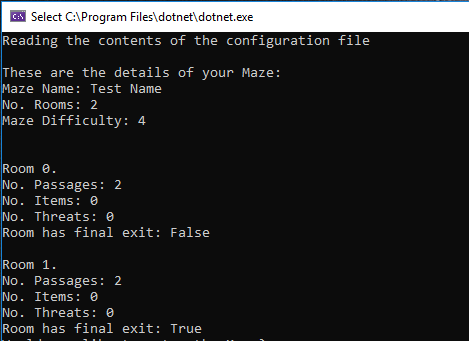
# Mock-Ups

The following mock-ups outline a vision for how the Maze Game’s user interface shall look. As it has been assumed that the Maze Game is to be developed as a command-line based text adventure, these mock-ups have been made by printing text out to the command line, simulating what the final game’s output is intended to look like.

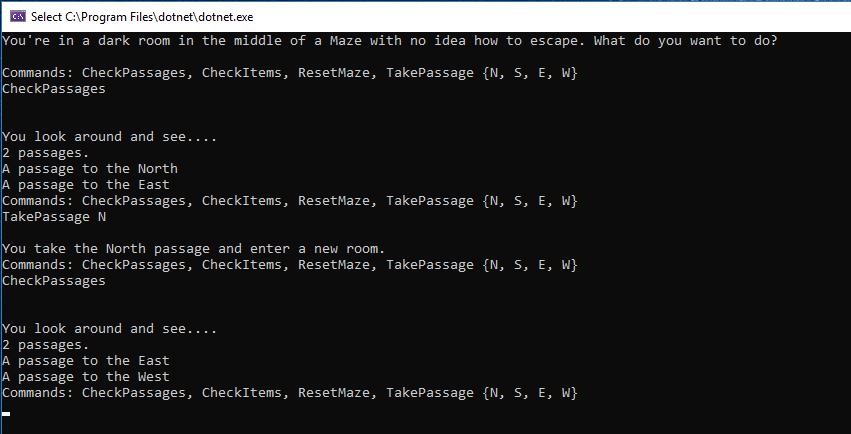
## Enter name



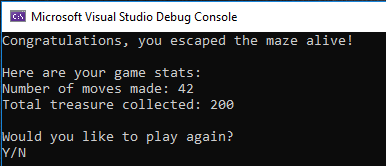
## Read configuration files



## In-Game Loop



## Game Finished



# Configuration File Data Formats

The Data formats used for the maze configuration file and any other configuration/resource files shall be JSON (JavaScript Object Notation) files. JSON is a data format suitable for the task of storing configuration data as it is structured and has strong support with .Net projects in the form of libraries. It is also used throughout the Witherslack Group’s development projects as a data storage and transfer medium, and therefore the developer of this project is familiar with it.