Simon Memory Game: Technical Design Document

**1.0 Revision History**

|  |  |
| --- | --- |
| **Version** | **Notes** |
| 1.0 | Initial version, nothing added yet |
| 1.1 | Pre-existing Bootstrap project mostly cleaned up, and sequence class under works |
| 1.2 | Sequence class complete, direction enum added, and arrow key checking w/ basic 'game continues' functionality added |
| 1.3 | Arrow sprites added, very basic graphical response to input added, and HUD cleaned up |
| 1.4 | Improved graphical response to input, initial sequence flashing added, and game over implemented |
| 1.5 | Score system added, and minor tweaks to arrow sprite design to improve visability |

**2.0 Development Environment**

**2.1 Programming Language and Graphical Framework**

The programming language used for this project will be C++, with OpenGL as the graphical framework (used by the AIE Bootstrap engine).

**2.2 IDE**

The IDE that will be used for the project is Visual Studio 2017.

**2.3 Source Control Procedures**

Source code will be using the version control software Tortoise SVN, and project files will be regularly uploaded to the GitHub website and updated.

**2.4 Third-Party Libraries**

This project will use the Standard Template Library (STL) and the AIE Bootstrap.

**2.5 Project Management Tools**

This project will be managed through the HacknPlan website:

<https://hacknplan.com/>

**2.6 Other Software**

This project will feature custom-made assets using Microsoft Paint.

**3.0 Timeline**

**3.1 Alpha Milestone**

The project should have core implementation by 28/4/19.

**3.2 Beta Milestone**

The project should have no additional features to add by 5/5/19.

**3.3 Gold Milestone**

The project should be complete by 10/5/19.

**4.0 Game Overview**

**4.1 Gameplay**

Players watch a random sequence of flashes in cardinal directions, and must memorise them, before then attempting to repeat them. Successfully repeating the directions causes the same sequence to flash but with one additional flash at the end. Unsuccessfully repeating the directions causes a game over.

**4.2 Game Objects and Logic**

Coloured directional arrows to indicate which directions need to be pressed, and to also display which direction has just been input. When activated, they will turn into a darker colour momentarily to show they’ve been activated. A two-digit score will keep track of how many sequences have been repeated successfully.

**4.3 Game Flow**

Players will begin on a main menu with options to play or quit. Selecting play will take the player to the game screen. The player will be prompted to press any key to begin. The first sequence will flash, and it will only be one random direction. If the player hits the correct corresponding direction, the game continues, and the same sequence will flash with another random direction at the end added on. Otherwise, they will be taken to a game over screen and their score will be displayed, and a prompt will be given to press any key, which will take them back to the main menu.

**4.4 Mechanics**

Players will be able to use the four arrow keys to repeat the sequence they see on the screen. The game will double check each direction inputted to determine if they are identical, and therefore whether the game should continue or game over.

**5.0 Game Architecture**

**5.1 Core Data Structures**

The directions of the sequence will be added to a dynamic array. The sequence will be stored in chronological order.

**5.2 Classes**

The project will need a game class (Game2D.h), a sequence class (Sequence.h), and a dynamic array class (DynamicArray.h). The game class will include the menu and rendering functionality of the project. The sequence class will contain the direction generation and storage, as well as the checking function.

**5.3 Algorithms**

No algorithms will be used for this project.

**6.0 UI and HUD**

**6.1 Heads-Up Display**



 **6.2 User-Interface**