SE 3XA3: Software Requirements Specification
Snake 2.0

Team #34, Send Help
George Mo (moz)
Harrison Lau (lauh3)
Vanessa Truong(truonv1)

December 2, 2017

Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

Our project purpose is to reimplement the classic arcade game, Snake, by adding newly designed features that will add a modern flair to the gameplay. Snake 2.0 will encompass Snake's basic functionality along with some twists to the game which will include map changes, power-ups and an enhanced GUI.

1.2 Overview

This document will specify the modular structure and decomposition of our reimplementation, and will serve as the Module Guide (MG) for our project. The decomposition will follow the design principles of information hiding and encapsulation, and the uses hierarchy, as they allow great flexibility for potential changes to our modules. With the basis of these principles, our modular design will abide by the following rules:

- Details in our modules with prospects of change will be kept private to that respective module.
- Any changes to the module "secrets" (anticpated changes) will not change the module design interface.
- There will be a lenience towards low-coupling and high cohesion within the modules.
- The uses relation between modules will encompass a "fan-in" characteristic.

The MG is intended to be easily understood by all project members, for efficient maintenance and design, and should be easy to follow for the course instructors and assistants. Having the module guide follow a hierarchical structure will provide project members with a better understanding of the modules when changes need to be implemented.

The rest of the document is organized as follows:

- Section 2 lists the anticipated and unlikely changes of the software requirements.

Table 1: **Revision History**

Date	**Version**	**Notes**
Nov 10, 2017	1.0	Finished Sections 1, 2, 3
Nov 10	1.1	Finished Sections 4, 5, 6, 7, 8
Dec 2	2.0	Final revision - Added Scoreboard module and updated uses hierarchy

- Section 3 summarizes the module decomposition that was constructed according to the likely changes.
- Section 4 specifies the connections between the software requirements and the modules.
- Section 5 gives a detailed description of the modules.
- Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules.
- Section 7 describes the use relation between modules.

2 Anticipated and Unlikely Changes

This section highlights the possible changes that may be made to Snake 2.0. According to the likeliness of the change, the possible changes are classified into two categories; anticipated changes (Section 2.1), and unlikely changes (Section 2.2).

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change. The following lists the anticpated changes to the module design:

AC1: The options for in-game map changes (i.e. what mape sizes to incorporate)

AC2: The numeric scale to which a pellet increases the Snake's speed.

AC3: The numeric scale to which a pellet decreases the Snake's speed.

AC4: The length of time for which invincibility to boundary collision will last.

AC5: The length of time for which invinicibility to self-collision will last.

AC6: Additional in-game features (e.g. move keys produce unexpected Snake direction)

AC7: The graphical user interface design for the menu.

AC8: The prospect of other platform compabilities (e.g. mobile, web)

2.2 Unlikely Changes

Although module design is more favourable when it is as general as possible, complexity is proportional to the generality of the module design. Thus, once design decisions are being simplified, other parts of the design will also require changes. The following decisions that are listed will not be changed:

UC1: Input/Output devices to the game (Game will assume keyboard and monitor is available).

UC2: The arrow keys for dictating Snake's movement.

UC3: The basic functionality of the original Snake game will be retained (i.e. eating pellets, growing in size, avoiding collisions)

UC4: There will always be an in-game scoreboard to track player's score.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Snake Module

M3: Menu Module

M4: Board Module

M5: Scoreboard Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Scoreboard Module Menu Module Scoreboard Module
Software Decision Module	Snake Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

The Board module contains all of the in-game functional requirements, and the in-game non-functional requirements, exhibiting the roles of the controller and model in the MVC design pattern. It deals with the input and output functionalities, as well as the look and feel, usability, and performance requirements. Such requirements include the input/output of the arrow keys for moving the snake, the speed in which the Snake moves at, the colour choices for the in-game board, and the response time to user inputs.

The Menu module will be the model and controller for the menu GUI. It will control all the design elements incorporated into the game's menu, such as the buttons and the different pages. The menu will essentially encompass a simplistic and colourful interface to satisfy the look and feels requirements, as well as the usability requirements.

The Scoreboard module will be part of the model component. It will simply display the player's score at the bottom of the application window.

The Snake module plays the role of the view component, which connects all the modules together and generates the output for the user based on their inputs processed by the Board module and Menu module.

5 Module Decomposition

The principle of "information hiding" proposed by **?** will be used as the basis for our module decomposition. The *Secrets* field provides a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title.

The following is the list of modules that will be implemented, noting that only the leaf modules in the hierarchy require implementation. Modules that don't require implementation will be marked with a dash (-).

5.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. It processes and communicates to the software all I/O actions from the hardware.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

5.2.1 Board Module (M4)

Secrets: The set-up variables for the game (window size, snake directions, pellet position, etc.)

Services: Sets up all the game features and stores variables for controlling the game's objects.

Implemented By: Java Abstract Windows Toolkit and Swing Library

5.2.2 Menu Module (M3)

Secrets: Menu set-up

Services: Sets up the start-up game menu window, buttons, and pages

Implemented By: Java Abstract Windows Toolkit and Swing Library

5.2.3 ScoreBoard Module (M5)

Secrets: The in-game scoreboard set-up

Services: Sets up the player's score during in-game sate and game over state

Implemented By: Java Abstract Windows Toolkit and Swing Library

5.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user. Changes in these modules are more likely to be motivated by a desire to improve performance than by externally imposed changes.

Implemented By: –

5.3.1 Snake Module (M2)

Secrets: Format window and display

Services: Sets up game window and displays Board and Menu modules

Implemented By: Java Abstract Windows Toolkit and Swing Library

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
Functional Req.	
FR1	M1, M4
FR2	M4
FR3	M4
FR4	M1, M4, M2, M3
FR5	M4
FR6	M4, M5
FR7	M2, M4, M3
Non-functional Req.	
NFR1	M3, M4
NFR2	M3, M4
NFR3	M1, M4, M5
NFR4	M1, M4
NFR5	M3, M4
NFR6	M1, M3, M4, M2
NFR7	M1, M4
NFR8	M1
NFR9	M1, M3, M4, M2
NRF10	M3, M4
NFR11	M3
NRF12	M3, M4

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M4, M2, M3
AC2	M4
AC3	M4
AC4	M4
AC5	M4
AC6	M4
AC7	M2, M3
AC8	M1

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

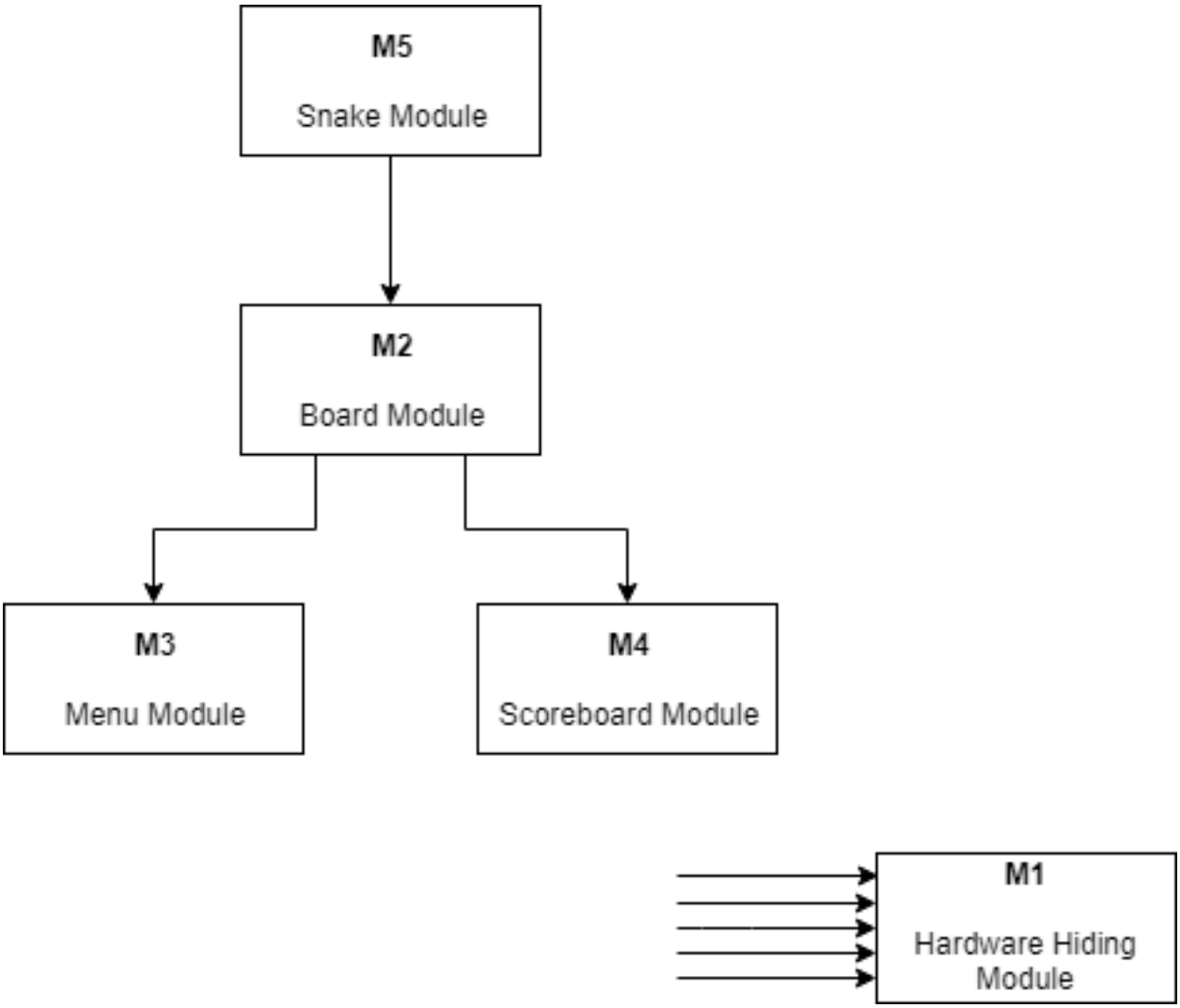


Figure 1: Use hierarchy among modules

8 Gantt Chart

The following pages include an updated Gantt Chart (updated in appropriate folder as well).

Team Send Help: Gantt Chart

10-Nov-2017

McMaster University

https://gitlab.cas.mcmaster.ca/truonv1/SendHelp_Snake_2.0

Project manager	
Project dates	29-Sep-2017 - 7-Dec-2017
Completion	0%
Tasks	30
Resources	3

This Gantt Chart outlines the recreation of the classic game Snake as our SE 3XA3 project.

Tasks

Name	Begin date	End date
(DevPlan) Sections 1 - 3	29/09/17	29/09/17
(DevPlan) Sections 4 - 6	29/09/17	29/09/17
(DevPlan) Sections 7 - 8	29/09/17	29/09/17
(SRS) Section 1 & 2	30/09/17	06/10/17
(SRS) Section 3	30/09/17	06/10/17
(SRS) Section 4	30/09/17	06/10/17
Proof of Concept Demonstration	07/10/17	19/10/17
Test Plan Revision 0	20/10/17	27/10/17
Sections 1, 2.1-2.4, 3.1, 4	20/10/17	27/10/17
Sections 2.5, 3.3, 5, 7	20/10/17	27/10/17
Sections 3.1.3, 3.2, 6	20/10/17	27/10/17
Testing	28/10/17	09/11/17
Test Menus	28/10/17	05/11/17
Test Map Change	28/10/17	06/11/17
Test Speed Modifications	28/10/17	31/10/17
Test Boundary Invincibility power-up	03/11/17	06/11/17
Test Performance	03/11/17	09/11/17
Test Memory Storage (high scores)	07/11/17	09/11/17
Design & Documentation Revision 0	28/10/17	10/11/17
(MG) Sections 1 - 6	28/10/17	10/11/17
(MG) Section 7	28/10/17	10/11/17
MIS	28/10/17	10/11/17
Revision 0 Demonstration	11/11/17	15/11/17
Implementation Final Dates	16/11/17	20/11/17

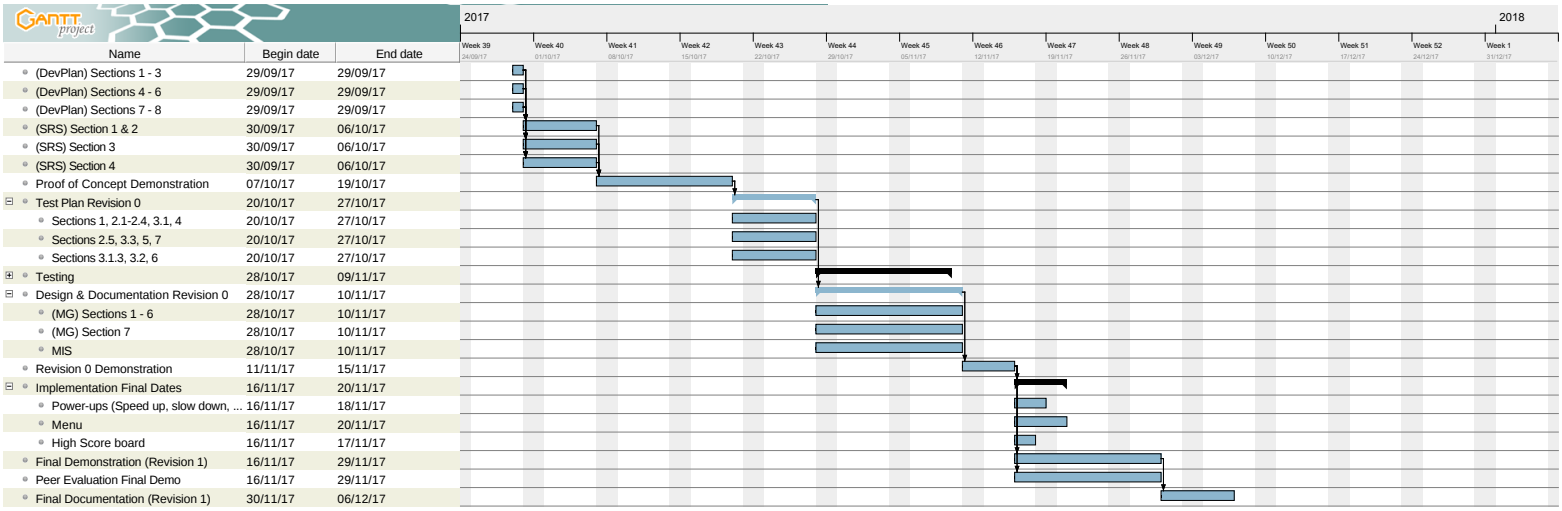
Tasks

Name	Begin date	End date
Power-ups (Speed up, slow down, collision immunity)	16/11/17	18/11/17
Menu	16/11/17	20/11/17
High Score board	16/11/17	17/11/17
Final Demonstration (Revision 1)	16/11/17	29/11/17
Peer Evaluation Final Demo	16/11/17	29/11/17
Final Documentation (Revision 1)	30/11/17	06/12/17

Resources

Name	Default role
Harrison Lau	Team Leader
George Mo	Developer 1
Vanessa Truong	Developer 2

Gantt Chart



Name	Begin date	End date
(DevPlan) Sections 1 - 3	29/09/17	29/09/17
(DevPlan) Sections 4 - 6	29/09/17	29/09/17
(DevPlan) Sections 7 - 8	29/09/17	29/09/17
(SRS) Section 1 & 2	30/09/17	06/10/17
(SRS) Section 3	30/09/17	06/10/17
(SRS) Section 4	30/09/17	06/10/17
Proof of Concept Demonstration	07/10/17	19/10/17
Test Plan Revision 0	20/10/17	27/10/17
Sections 1, 2.1-2.4, 3.1, 4	20/10/17	27/10/17
Sections 2.5, 3.3, 5, 7	20/10/17	27/10/17
Sections 3.1.3, 3.2, 6	20/10/17	27/10/17
Testing	28/10/17	09/11/17
Design & Documentation Revision 0	28/10/17	10/11/17
(MG) Sections 1 - 6	28/10/17	10/11/17
(MG) Section 7	28/10/17	10/11/17
MIS	28/10/17	10/11/17
Revision 0 Demonstration	11/11/17	15/11/17
Implementation Final Dates	16/11/17	20/11/17
Power-ups (Speed up, slow down, ...	16/11/17	18/11/17
Menu	16/11/17	20/11/17
High Score board	16/11/17	17/11/17
Final Demonstration (Revision 1)	16/11/17	29/11/17
Peer Evaluation Final Demo	16/11/17	29/11/17
Final Documentation (Revision 1)	30/11/17	06/12/17

Resources Chart