Design Document for Snake Game

Keyur Patel Alex Guerrero Shafeeq Rabbani

November 6, 2015

Contents

1 Introduction

The following documentation is intended elaborate how the design of the snake game is implemented. The document will also explain how the functional and non-functional requirements mentioned in the Software Requirements Specification will be explained. This document is intended for the following readers:

- Designers: The document can serve to ensure that all functional and non-functional requirements are met. Further, designers can also use this document to verify any discrepancies among different modules.
- New Project Members: This document can bring new team members up-to-date with the overview and structure of the game.
- Maintainers: This document can also used to understand the structure of the game and all of the modules within. It would then be the responsibility of the maintainers to update the Design document by mentioning any change they have made to it.

The Snake game has been divided up into modules which hide information from other modules in the document in order to implement the Information Hiding principle of Software Engineering. Further, the modules only share the information amongst each other that is necessary. This is done to ensure the Low Coupling principle of Software Engineering. In order to read or modify values within different modules, there are getter and setter methods unique to the respective modules. This is done to implement the Encapsulation principle of Software Engineering.

This document consists of the Module Interface Specification (or MIS) which is intended for programmers who work to further develop the Snake Game.

The rest of this document consists of the GANTT and PERT chart meant to highlight the time frame for future deliverables for development of the Snake Game.

2 Anticipated and Unlikely Changes

There are two types of possible changes: Anticipated and Unlikely Changes. This section covers both of these changes.

2.1 Anticipated Changes

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input data.

AC3: The format of the input parameters.

AC4: The format of the final output data.

AC5: How the governing ODEs are defined using the input parameters.

AC6: How the energy equations are defined using the input parameters.

AC7: How the overall control of the calculations is orchestrated.

AC8: The implementation for the sequence (array) data structure.

AC9: The algorithm used for the ODE solver.

AC10: The implementation of plotting data.

2.2 Unlikely Changes

These are design decisions that have to be changed after they were fixed in the software architecture state (in order to simplify the design). It wasn't intended that these decisions would have to be changed.

UC1: Input/Output devices (Input: Keyboard, Mouse Output: Screen).

UC2: There will always be a source of input data external to the software.

UC3: Output data are displayed to the output device.

UC4: The goal of the system is to calculate temperatures and energies.

UC5: The ODEs for temperature can be defined using parameters defined in the input parameters module.

UC6: The energy equations can be defined using the parameters defined in the input parameters module.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Input Format Module

M3: Input Parameters Module

M4: Output Format Module

M5: Temperature ODEs Module

M6: Energy Equations Module

M7: Control Module

M8: Sequence Data Structure Module

M9: ODE Solver Module

M10: Plotting Module

Note that M1 is a commonly used modules and are already implemented by the operating system. It will not be reimplemented. Similarly, M8, M9 and M10 are already available in Matlab and will not be reimplemented.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Format Module Input Parameters Module Output Format Module Temperature ODEs Module Energy Equations Module Control Module
Software Decision Module	Sequence Data Structure Module ODE Solver Module Plotting Module

Table 1: Module Hierarchy

4 Connection Between Requirements and Design

The table below highlights the connection between the system requirements(which are listed in the Software Requirements Specification) and the modules.

AC	Modules
AC1	M1
AC2	M2
AC3	M3
AC4	M4
AC5	M5
AC6	M6
AC7	M7
AC8	M8
AC9	M9
AC10	M10

Table 2: Trace Between Anticipated Changes and Modules

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. The *Secrets* field in a module decomposition is 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. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. If the entry is *Matlab*, this means that the module is provided by Matlab. *SWHS* means the module will be implemented by the SWHS software. Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

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. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviors.

Services: Includes programs that provide externally visible behavior 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 Input Format Module (M2)

Secrets: The format and structure of the input data

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: SWHS

5.2.2 Input Parameters Module (M3)

Secrets: The format and structure of the input parameters.

Services: Stores the parameters needed for the program, including material properties, processing conditions and numerical parameters. The values can be read as needed. This module knows how many parameters it stores.

Implemented By: SWHS

5.2.3 Output Format Module (M4)

Secrets: The format and structure of the output data.

Services: Outputs the results of the calculations, including the input parameters, temperatures, energies, and times when melting starts and stops.

Implemented By: SWHS

5.2.4 Temperature ODEs Module (M5)

Secrets: The ODEs for solving the temperature, using the input parameters.

Services: Defines the ODEs using the parameters in the input parameters module.

Implemented By: SWHS

5.2.5 Energy Equations Module (M6)

Secrets: The equations for solving for the energies using the input parameters.

Services: Defines the energy equations using the parameters in the input parameters module.

Implemented By: SWHS

5.2.6 Control Module (M7)

Secrets: The algorithm for coordinating the running of the program.

Services: Provides the main program.

Implemented By: SWHS

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.

Implemented By: –

5.3.1 Sequence Data Structure Module (M8)

Secrets: The data structure for a sequence data type.

Services: Provides array manipulation, including building an array, accessing a specific entry, slicing an array etc.

Implemented By: Matlab

5.3.2 ODE Solver Module (M9)

Secrets: The algorithm to solve a system of first order ODEs.

Services: Provides solvers that take the governing equation, initial conditions and numerical parameters and solve them.

Implemented By: Matlab

5.3.3 Plotting Module (M10)

Secrets: The data structures and algorithms for plotting data graphically.

Services: Provides a plot function.

Implemented By: Matlab

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes. We should also consider documenting the mapping between these "abstract" modules and the Matlab files.

Req.	**Modules**
R1	M1, M2, M3, M7
R2	M2, M3
R3	M2
R4	M4, M7
R5	M4, M5, M7, M8, M9, M10
R6	M4, M5, M7, M8, M9, M10
R7	M4, M6, M7, M8, M10
R8	M4, M6, M7, M8, M10
R9	M4, M5, M7
R10	M4, M5, M6, M7

Table 3: Trace Between Requirements 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 **??** 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.