SE 3XA3: Module Guide
SnakeGame Project

Team L03G09
Qiang Gao (gaoq20)
Zhiwei Li (liz342)
Longwei Ye (yel16)

March 18, 2022

Contents

1 Introduction **1**
- 1.1 Overview of the Project . . . 1
- 1.2 Context . . . 1
- 1.3 Design Principles . . . 1
- 1.4 Document Structure . . . 1
  - 1.4.1 Naming Conversion and Terminology . . . 2

2 Anticipated and Unlikely Changes **2**
- 2.1 Anticipated Changes . . . 2
- 2.2 Unlikely Changes . . . 3

3 Module Hierarchy **3**

4 Connection Between Requirements and Design **4**

5 Module Decomposition **4**
- 5.1 Hardware Hiding Modules (M1) . . . 4
- 5.2 Behaviour-Hiding Module . . . 5
  - 5.2.1 View Module (M2) . . . 5
  - 5.2.2 BGM Control Module (M3) . . . 5
  - 5.2.3 Inputs Checking Module (M4) . . . 5
  - 5.2.4 Game Control Module (M5) . . . 5
- 5.3 Software Decision Module . . . 6
  - 5.3.1 SnakeGame Module (M6) . . . 6
  - 5.3.2 History Module (M7) . . . 6
  - 5.3.3 Snake Color Module (M8) . . . 6

6 Traceability Matrix **7**

7 Use Hierarchy Between Modules **8**

8 Project Schedule **8**

List of Tables

1 **Revision History** . . . ii
2 **Definitions** . . . 2
3 **Module Hierarchy** . . . 4
4 **Trace Between Requirements and Modules** . . . 7
5 **Trace Between Anticipated Changes and Modules** . . . 7

List of Figures

1 Use hierarchy among modules 8
2 Time schedule 8
3 Resource divide 9

Date	Version	Notes
2022/3/15	1.0	Complete Section 1, 2
2022/3/16	1.1	Complete Section 3
2022/3/17	1.2	Complete Section 4, 5
2022/3/18	1.3	Complete Section 6, 7, 8
2022/3/18	1.4	Revise MG Document

Table 1: **Revision History**

1 Introduction

1.1 Overview of the Project

The SnakeGame is a re-implementation of the open-source game that allows the user to cutomize and control the "snake", and gain points by eating the blocks on the gameboard.

1.2 Context

This document is the Module Guide(MG), which is based on the previous Software Requirements Specification document(SRS). The purpose of the MG document is to establish a modular break-down of the project and to show the project's structure. Besides, as the SRS document defines the functional and non-functional requirements of the project, MG should document how the system meets the requirements defined in SRS.

In addition to MG, the Module Interface Specification(MIS) is also documented, which specifies the external details(state and environment variables, Assumptions, Access Program Semantics, Inputs, Outputs, Exceptions) of the module, which provides a further view of how the project is implemented.

1.3 Design Principles

As MG is showing the decomposition of the system into modular subsystems, this is accomplished following the design principle of Information Hiding and Encapsulation. Besides, the Use Hierarchy between modules should follow the principle of High Cohesion, Low Coupling.

To be more specific, the principle of information hiding states that each module hides a secret and the implementation of encapsulation is when the information is changing, this information is performed in the module and the information becomes the secret of it. Last but not least, High cohesion means that the functions within a module has a strong relation, whereas the relation between modules are tight, which is principle of low coupling.

1.4 Document Structure

The structure of the document are listed at the following:

- Section 2 lists the anticipated and unlikely changes of the program's implementation. The listed items are used in the Traceability Matrix section.
- Section 3 shows the details (modules and their secrets) of the module hierarchy.
- Section 4 demonstrates the close relationship between SRS and MG by showing their connections.
- Section 5 gives the details of the decomposed modules of the system, including:

  - Module Name
  - Secrets
  - Services
  - Responsibility

- Section 6 provides the Traceability Matrix, showing the connection between requirements in SRS and the modules, as well the relationship between Anticipated Changes and the modules.
- Section 7 is the Use Hierarchy between modules.
- Section 8 lists the project schedule.

1.4.1 Naming Conversion and Terminology

Terms	Definitions
SRS	Software Requirements Specification
MG	Module Guide
MIS	Module Internal Specification
JavaScript	A programming language used to implement core script for World Wide Web
BGM	Background Music
N/I	Not Implemented

Table 2: **Definitions**

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in 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.

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

AC2: The graphical user interface elements used to retrieve user input and their format.

AC3: Additional functionality or mechanism for the game.

AC4: The format of the output data.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and Keyboard, Output: File, Memory, and Screen).

UC2: The original mechanism of the game.

UC3: The Graph of nodes that represents the playable grid.

UC4: There will always be a source of input data external to the software.

UC5: The format of the input data.

UC6: Default settings for input.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 3. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: View Module

M3: BGM Control Module

M4: Inputs Checking Module

M5: Game Control Module

M6: SnakeGame Module

M7: History Module

M8: Snake Color Module

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	View Module BGM Control Module Inputs Checking Module Game Control Module
Software Decision Module	SnakeGame Module History Module Snake Color Module

Table 3: **Module Hierarchy**

4 Connection Between Requirements and Design

The system is intended to satisfy all of the functional and non-functional requirements that were initially specified in the SRS. At this stage, in order to meet information hiding principle as well as low coupling and high cohesion, we decomposed the system into modules, assign responsibilities to those modules and ensure that they fit together to achieve our global goals. The connection between requirements and modules is listed in Table 4. Section 5 specifies how we decomposed the 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. Also indicate if the module will be implemented specifically for the 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: This module serves as a virtual hardware used by the rest part of the system, which allows the system to implements actions such as display outputs and accept inputs from the user.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: This module serves as the connector between the Hardware Hiding Modules and the Software Decision Module, which describes the externally visible behaviour of the system listed in the SRS document.

Implemented By: N/A

5.2.1 View Module (M2)

Secrets: GUI.

Services: Allows the user to see the game components and to interact with the game.

Implemented By: HTML

5.2.2 BGM Control Module (M3)

Secrets: BGM and volume control

Services: Allows the user to play/pause the BGM and also to change the BGM's playing volume.

Implemented By: JavaScript

5.2.3 Inputs Checking Module (M4)

Secrets: Inputs

Services: Allows the user to specify valid snake's color and username.

Implemented By: JavaScript

5.2.4 Game Control Module (M5)

Secrets: Game control algorithm

Services: Accepts user inputs and updates the game status.

Implemented By: JavaScript

5.3 Software Decision Module

Secrets: Data Structures and the secrets of the module that are *not* discussed in the SRS document.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: N/A

5.3.1 SnakeGame Module (M6)

Secrets: SnakeGame objects

Services: Stores the in-game elements status(i.e. position of the snake and supplements, speed of the snake, points that the user has gained.)

Implemented By: JavaScript

5.3.2 History Module (M7)

Secrets: Array

Services: Stores the previous game history(player name and score.)

Implemented By: JavaScript

5.3.3 Snake Color Module (M8)

Secrets: Color of the snake

Services: Stores the color specification of the snake.

Implemented By: JavaScript

6 Traceability Matrix

Req.	Modules
R1	M2, M4, M5, M6, M8
R2	M2, M4, M5
R3	N/I
R4	M2, M5, M7
R5	M2
R6	M2
R7	N/I
R8	M5, M6
R9	M6, M1
R10	M6
R11	M6
R12	M6
R13	M2, M6
R14	M5
R15	M5, M7
R16	M6
R17	N/I
R18	M6

Table 4: **Trace Between Requirements and Modules**

AC	Modules
AC1	M1
AC2	M2, M5, M4, M6
AC3	M6, M5
AC4	M6

Table 5: **Trace Between Anticipated Changes and Modules**

7 Use Hierarchy Between Modules

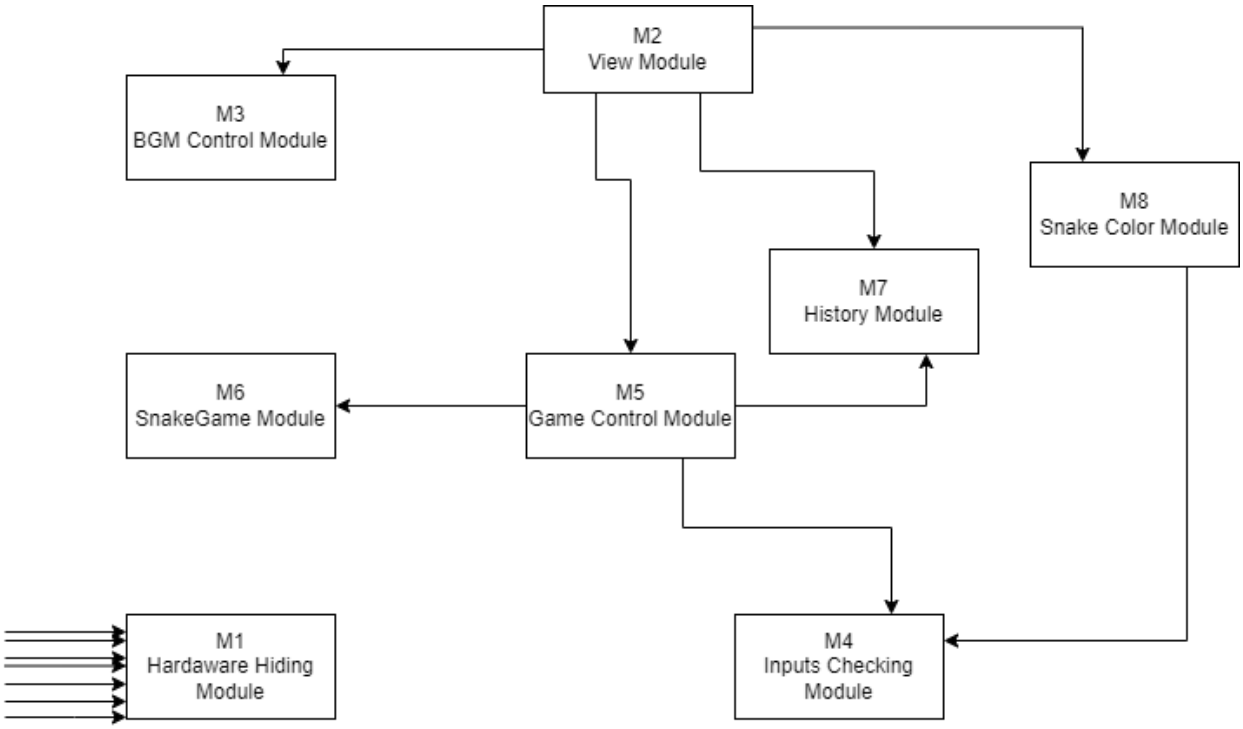

Figure 1: **Use hierarchy among modules**

8 Project Schedule

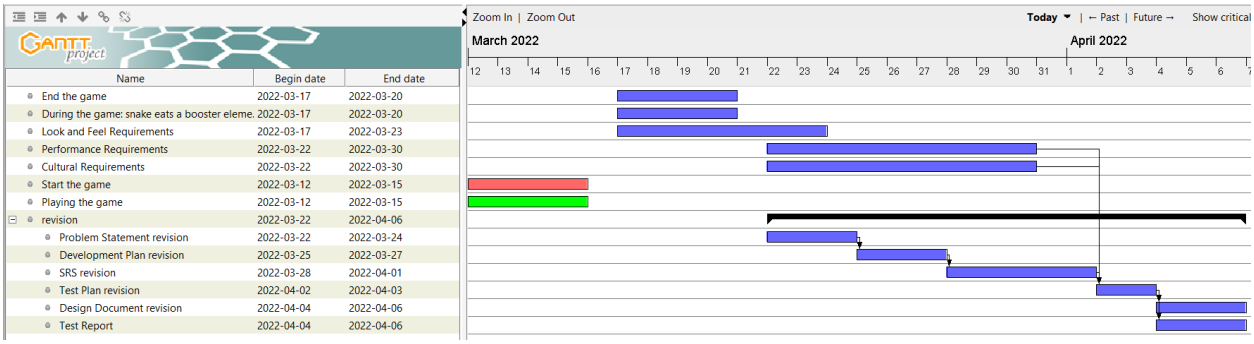

Figure 2: **Time schedule**

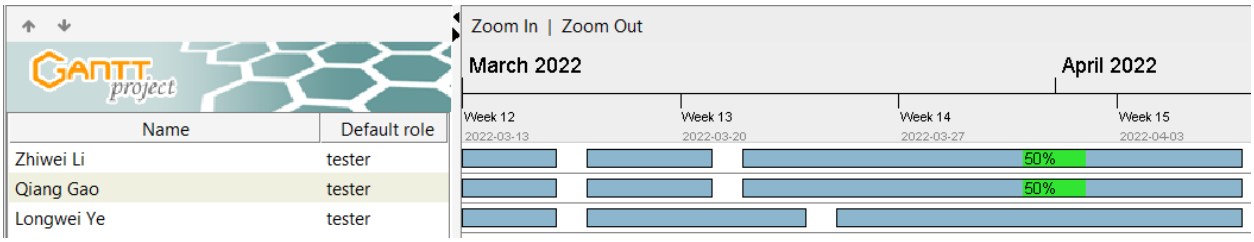

Figure 3: **Resource divide**