

# Module Guide

SFWRENG 3XA3

Group 30

Alan Yin (yins1)

Huajie Zhu (zhuh5)

Junni Pan (panj10)

November 09, 2017

# Contents

<b>1</b>	<b>Major Revision History</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
2.1	Overview . . . . .	3
2.2	Context . . . . .	3
2.3	Design Principles . . . . .	3
2.4	Document Structure . . . . .	3
<b>3</b>	<b>Anticipated and Unlikely Changes</b>	<b>4</b>
3.1	Anticipated Changes . . . . .	4
3.2	Unlikely Changes . . . . .	4
<b>4</b>	<b>Module Hierarchy</b>	<b>4</b>
<b>5</b>	<b>Connection Between Requirements and Design</b>	<b>4</b>
<b>6</b>	<b>Module Decomposition</b>	<b>5</b>
6.1	Game Module . . . . .	5
6.2	Game Controller Module . . . . .	6
6.3	ArtistSwing Module . . . . .	6
6.4	Gameclock Module . . . . .	6
6.5	MouseAndKeyboardHandler Module . . . . .	6
6.6	Critter Module . . . . .	6
6.7	Tower Module . . . . .	6
6.8	MapTile Module . . . . .	6
6.9	Player Module . . . . .	7
6.10	Point Module . . . . .	7
6.11	Closest Module . . . . .	7
6.12	Farthest Module . . . . .	7
6.13	IStrategy Module . . . . .	7
6.14	Strongest Module . . . . .	7
6.15	Weakest Module . . . . .	7
6.16	GameApplicationFrame Module . . . . .	7
6.17	GameState Module . . . . .	8
6.18	MainMenu Module . . . . .	8
6.19	MenuApplicationFrame Module . . . . .	8
6.20	SetupClass Module . . . . .	8
<b>7</b>	<b>Traceability Matrices</b>	<b>8</b>
7.1	Modules and Requirements . . . . .	8
7.2	Modules and Anticipated Changes . . . . .	9
<b>8</b>	<b>Uses Hierarchy Between Modules</b>	<b>9</b>
<b>9</b>	<b>Schedule</b>	<b>10</b>
9.1	Gantt . . . . .	10

## List of Tables

1	Major Revision History . . . . .	3
2	Module Hierarchy . . . . .	5
3	Trace Between Requirements and Modules . . . . .	8

4	Trace Between Requirements and Modules . . . . .	9
---	--	---

## List of Figures

1	Uses hierarchy between modules . . . . .	9
2	Gantt chart of project schedule . . . . .	10
3	Gantt chart(resources) of project schedule . . . . .	10

# 1 Major Revision History

Date	Revision
November 6, 2017	Rough draft of sections
November 7, 2017	Revised sections
November 9, 2017	Revision 0 complete
December 6, 2017	Revision 1 complete

Table 1: Major Revision History

## 2 Introduction

### 2.1 Overview

The Tower Defense game project is a partial re-implementation and refinement based on an existing open source project. The completed project allows the user to build towers on the map to defend enemies from a certain route.

### 2.2 Context

The Module Guide (MG) document is an introduction to modules in this project, where the module functionality and specification are created based on Software Requirement Specification (SRS) document. The MG covers all the anticipated and unlikely changes to functional and non-functional requirements specified in SRS, and provides a modular structure for the entire system. The MG also includes the relationship between modules and the corresponding requirements they achieved. At the same time, a Module Interface Specification (MIS) document is also created in the design documents to provide details on every module which are mentioned in MG.

### 2.3 Design Principles

Design Principles being used in this project are Information Hiding and Uses Relation. Information hiding ensures that each module design interface will not change even when the module secret changes.

### 2.4 Document Structure

- Section 1  
(Current Section) A brief review of the project and the introduction of Module Guide Document.
- Section 2  
List anticipated and unlikely Changes from Software Requirement Specification document.
- Section 3  
Give details for the Module Hierarchy, provide the module hierarchy by secrets.
- Section 4  
Give explanation for the Connection Between Requirements and Design, which shows how the software requirements are accomplished by the listed modules.
- Section 5  
Provide the Module Decomposition details, including module names, secrets, service or responsibility, and implemented by information for each module listed above.

- Section 6  
Provide the Traceability Matrices which connect the requirements to the modules, and anticipated changes from Section 2 to the modules. This provides a straightforward representation of how the requirements in SRS are accomplished.
- Section 7  
Provide the Uses Hierarchy for the project, which shows the uses relations between modules.
- Section 8  
Provide the Project Schedule by including up-to-date Gantt chart.
- Section 9  
List the Major Revision History for the document.

## 3 Anticipated and Unlikely Changes

### 3.1 Anticipated Changes

AC1: The game start menu and looking of interface.  
 AC2: The addition of Java gaming library.  
 AC3: The visual representation of tower and critters.  
 AC4: The deletion of map editor feature.  
 AC5: The logic and property of critters and towers for balancing modification.  
 AC6: The addition of AreaAttack tower.  
 AC7: The addition of animation effect and background music.  
 AC8: How wave are grouped into stages with different maps and winning strategies.

### 3.2 Unlikely Changes

UC1: Integration of swing and Slick2D windows.  
 UC2: The running environment and devices.  
 UC3: The algorithm for target selection.  
 UC4: The integration of map editor and stage selection option.

## 4 Module Hierarchy

## 5 Connection Between Requirements and Design

The modules are designed into 5 categories, including controllers, helpers, models, strategies, and views. Game module is the main module where the running request is called, and Game Controller module calls every required frames and panels to form the game window. Modules under models category are abstract data types which store the properties of players, towers, critters, maps, etc. These data can be accessed by ArtistSwing module to convert text data into graphical representations, and then the game panels can use graphical representations to display the game to the users on GUI. All the user inputs are done by mouse and keyboard, and they are collected by MouseAndKeyboardHandler module.

In order to enhance the gameplay experience and to facilitate development process, an external library called slick2D is used in this project. It provides a better way of implementing a clean and unambiguous user interface. Also, buttons, dropdown menus and sliders are included in the game controller window for easier user operation. All the buttons and operation areas are properly sized, and the game window is set in high resolution so that it satisfies the usability requirement. The project can be exported into a jar file. This means it could run on any platform that support Java. It satisfies the requirement where the game is supposed to be cross-platform. The usage of licensed library and open sourced code ensures that the legal requirement is not violated.

Level 1	Level 2
Hardware Hiding Module	MouseAndKeyboardHandler
Behaviour Hiding Module	Game Module GameController Module ArtistSwing Module Gameclock Module Critter Module Tower Module MapTile Module Player Module GameApplicationFrame Module GameState Module MainMenu Module MenuApplicationFrame Module
Software Decision Module	Point Module IStrategy Module Farthest Module Closest Module Strongest Module Weakest Module SetupClass Module

Table 2: Module Hierarchy

## 6 Module Decomposition

M1: Game Module  
 M2: Game Controller Module  
 M3: ArtistSwing Module  
 M4: Gameclock Module  
 M5: MouseAndKeyboardHandler Module  
 M6: Critter Module  
 M7: Tower Module  
 M8: MapTile Module  
 M9: Player Module  
 M10: Point Module  
 M11: Closest Module  
 M12: Farthest Module  
 M13: IStrategy Module  
 M14: Strongest Module  
 M15: Weakest Module  
 M16: GameApplicationFrame Module  
 M17: GameState Module  
 M18: MainMenu Module  
 M19: MenuApplicationFrame Module  
 M20: SetupClass Module

### 6.1 Game Module

**Secret:** Contents in game panel

**Services:** This module serves as the initiator of the game project. It starts the application, and sets the

resolution to 1000\*700.

**Implemented By:** Java Library

## 6.2 Game Controller Module

**Secrets:** Data on map, towers and critters.

**Services:** Handles all the complex interfacing with map, the towers placed on the map, and critters traversing on shortest path.

**Implemented By:** Java Library

## 6.3 ArtistSwing Module

**Secret:** Data on DrawableEntities

**Service:** Transform game logic and data into drawable entity information, in order to create graphical representation of game.

**Implemented By:** Java Libraries

## 6.4 Gameclock Module

**Secret:** Drawable entity information.

**Service:** Determine the refresh rate for redrawing entities, and enable pause function for the game project.

**Implemented By:** Java Libraries

## 6.5 MouseAndKeyboardHandler Module

**Secret:** User input

**Service:** Provide corresponding functions for mouse operations.

**Implemented By:** Java Libraries

## 6.6 Critter Module

**Secret:** Default setting for every critter.

**Service:** Abstract data type of a critter object.

**Implemented By:** Java Libraries

## 6.7 Tower Module

**Secret:** Default setting for every tower.

**Service:** Abstract data type of a tower object.

**Implemented By:** Java Libraries

## 6.8 MapTile Module

**Secrets:** Usable data from map.

**Services:** Refers to tiles associated with map.

**Implemented By:** Java Libraries

## 6.9 Player Module

**Secret:** Player information.

**Service:** Abstract data type for player to store player information.

**Implemented By:** Java Libraries

## 6.10 Point Module

**Secret:** Usage for points

**Service:** Simply has an x point, and a y point. Used for positioning

**Implemented By:** Java Libraries

## 6.11 Closest Module

**Secret:** Tower information

**Service:** Finds the target based on who is closest.

**Implemented By:** Java Libraries

## 6.12 Farthest Module

**Secret:** Tower information

**Service:** Finds the target based on who is farthest.

**Implemented By:** Java Libraries

## 6.13 IStrategy Module

**Secret:** Algorithm details

**Service:** Find a target critter.

**Implemented By:** Java Libraries

## 6.14 Strongest Module

**Secret:** Tower information

**Service:** Finds the target based on who is strongest.

**Implemented By:** Java Libraries

## 6.15 Weakest Module

**Secret:** Tower information

**Service:** Finds the target based on who is weakest.

**Implemented By:** Java Libraries

## 6.16 GameApplicationFrame Module

**Secret:** Detailed data in the frame.

**Service:** Swing implementation of game window.

**Implemented By:** Swing



## 6.17 GameState Module

**Secret:** Data in game states.

**Service:** Transition from main menu to game application.

**Implemented By:** Slick2D

## 6.18 MainMenu Module

**Secret:** Contents in main menu.

**Service:** Slick2D main menu setup.

**Implemented By:** Slick2D

## 6.19 MenuApplicationFrame Module

**Secret:** contents in menu.

**Service:** Set default map and all button application.

**Implemented By:** Slick2D

## 6.20 SetupClass Module

**Secret:** Data from Slick2D window.

**Service:** Create link between main menu and game application.

**Implemented By:** Slick2D

# 7 Traceability Matrices

## 7.1 Modules and Requirements

Requirement	Modules
Functional Requirements	
FR1	M16, M17, M18, M19, M20
FR2	M3, M6, M7, M8
FR3	M3, M7, M10, M16
FR4	M2, M9, M17, M18, M19
Non-functional Requirements	
NF1	M17, M18, M19
NF2	M16
NF3	M1
NF4	M2, M4
NF5	M17
NF6	M1
NF7	M1, M3
NF8	M1-M20
NF9	M17, M18, M19, M20

Table 3: Trace Between Requirements and Modules

## 7.2 Modules and Anticipated Changes

Anticipated Changes	Modules
AC1	M1, M17, M18, M19, M20
AC2	M17, M18, M19, M20
AC3	M3, M6, M7
AC4	M2
AC5	M7, M8
AC6	M7, M8
AC7	M17, M18
AC8	M17

Table 4: Trace Between Requirements and Modules

## 8 Uses Hierarchy Between Modules

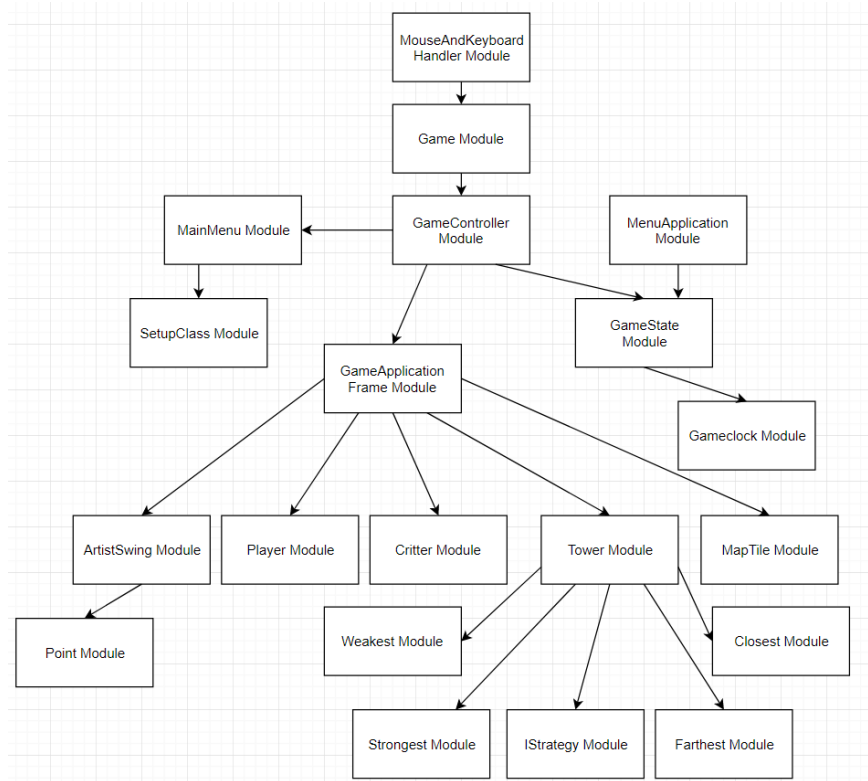


Figure 1: Uses hierarchy between modules

## 9 Schedule

### 9.1 Gantt

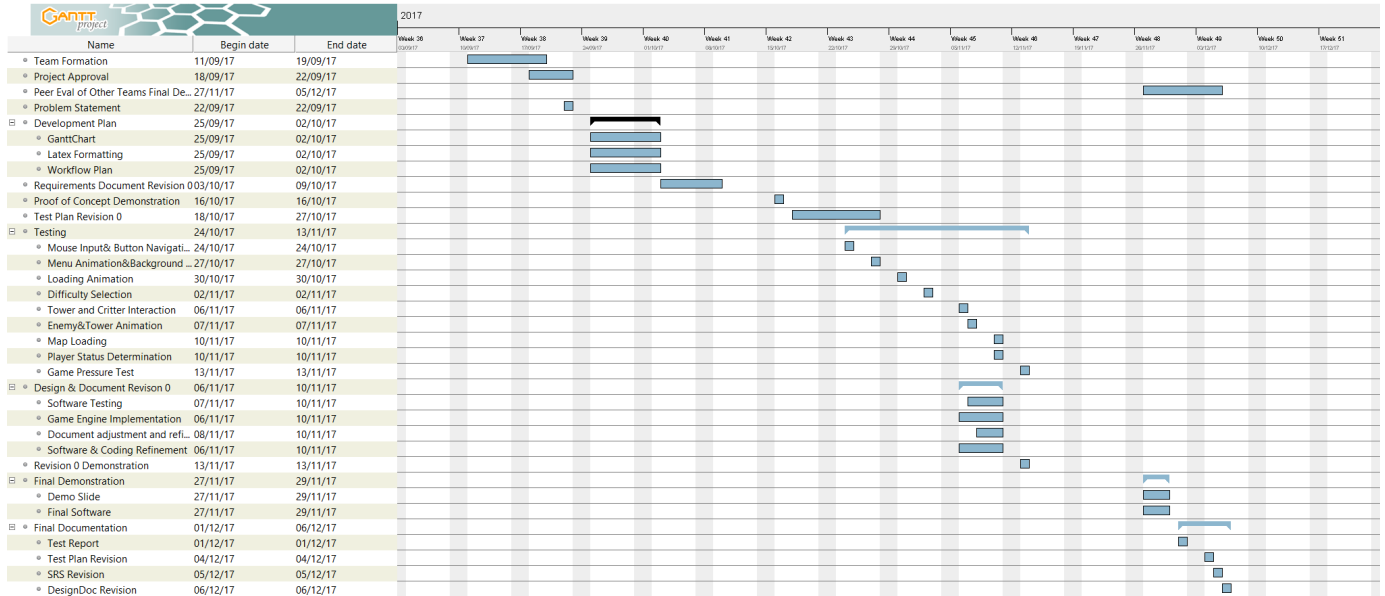


Figure 2: Gantt chart of project schedule

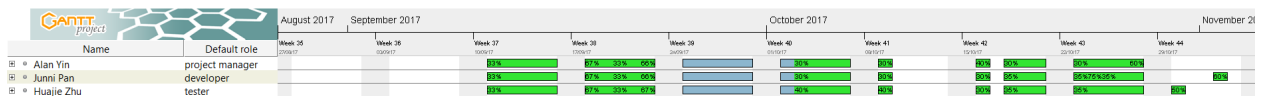


Figure 3: Gantt chart(resources) of project schedule