Module Guide
Dragon Age

Group 8: Team Eight
Stanley Liu (MacID: liuz23)
Toni Miharja (MacID: miharjat)
Zhi Zhang (MacID: zhangz1)

December 6 2017

Contents

1 Revision History

Date	Author	Notes	Version
November 7, 2017	Toni	Introduction	1.0
November 8, 2017	Stanley	Anticipated and Unlikely Changes, Module Decomposition	1.1
November 8, 2017	Toni	Use Hierarchy between Modules	1.2
November 9, 2017	Toni, Stanley	Module Decomposition	1.3
November 10, 2017	Toni, Stanley	Traceability Matrix, Module Decomposition	1.4
December 6, 2017	Stanley Liu	Update use hierarchy and project schedule	2.1
December 6, 2017	Zhi Zhang	Improve	2.2

Table 1: Revision History: Module Guide

2 General Information and Introduction

This document serves as the Module Guide (MG) and Design Documentation for our Dragon Age (DA) Project. This document will be useful for the following target audience:

- New Project Contributors: New members can quickly get up to speed by understanding the big picture of the project as they read this document.
- Developers/ Designers: This document will be the central collection of documentation for designers and developers with detailed explanation of module specifications. It also helps the designers in determining if the product design will satisfy the requirements of the program, allowing for a traceback to the requirements.
- Maintainers: Under the anticipated changes section, maintainers will be able to design future alterations without severely affecting the rest of the modules by understanding the program architecture. They will also be able to plan ahead as they roll out improvements and changes to the next versions.

2.1 Modular Decomposition

The team acknowledge the importance of decomposing a system into modules and further into sub-modules as it is a crucial task to ensure that the large system is easily maintained, coded and understood. Modular decomposition prevents the system from becoming too complicated as it grows. By doing so, it allows multiple developers to concurrently develop components while minimising the risk of "breaking" the project. A highly modular system will allow new features to be added seamlessly without much alterations to the current working environment.

2.2 Document Organization

The rest of the document is organised in the following manner:

- Section 3 lists the anticipated and unlikely changes of the software requirements.

- Section 4 summarizes the module decomposition that was constructed according to the likely changes.
- Section 5 specifies connections between the software requirements and the modules.
- Section 6 gives a detailed description of the modules.
- Section 7 includes one traceability matrices. It shows the relation between anticipated changes and the modules.
- Section 8 specifies the Uses Relationship, describing the use relation between modules.

3 Anticipated and Unlikely Changes

This section lists some possible changes that may occur to our project. Anticipated changes are in section 3.1 and unlikely changes are in section 3.2.

3.1 Anticipated Changes

Anticipated changes are the changes that will be made to elements that hide in modules, and are easy to change without affecting the rest of the project.

AC1: More game controls (ex: keyboard shortcuts for user's convenience)
AC2: Sound features of the game
AC3: Additional functionalities (ex: additional dragon type or enemy type)
AC4: Images of the components of the game (map, dragons, buttons)

3.2 Unlikely Changes

Unlikely changes are the design decisions that affect the main components and functions of the game. In risk of having to modify multiple modules of the game, these decisions are unlikely to change.

UC1: The main goal of the game: the concept of defending base from enemies using attacking units
UC2: The programming language: *Python* and *Pygame*
UC3: The game controller: mouse
UC4: The input data (database of dragons and enemies), not including images

4 Module Hierarchy

5 Connection Between Requirements and Design

The system is designed to satisfy the requirements developed in the System Requirement Specification document. In this document, the system is decomposed into modules that ultimately meet

Level 1	Level 2
Hardware Hiding Module	
Behaviour Hiding Module	M1. Dragon Tower Module M2. Timer Bullet Module M3. Timer Enemy Module M4. Timer Hover Module M5. Timer Fired Module M6. Draw Module M7. Game Manager Module M8. Dragon Age Module
Software Decision Hiding Module	M9. Dragon Module M10. Enemy Module M11. Bullet Module M12. Path Module M13. Game Date Module

Table 2: Module Hierarchy

the requirements specified.

6 Module Decomposition

6.1 Hardware Hiding Modules

Secret: The implementation of the interpreter
Services: This module serves as the interface between the game and the hardware. It allows the system to communicate with the the code
Implemented By:*Python* interpreter and Operating System

6.2 Behaviour-Hiding Module

Secret: Behaviours
Services: This module describes the visible behavior of the game. It functions as the interpreter between hardware hiding modules and software decision modules
Implemented By:N/A

6.2.1 Dragon Tower Module

Secret: Dragon Tower
Services: The dragon tower checks if enemies are in range and shoots at enemies
Implemented By:*Python* Library

6.2.2 Timer Bullet Module

Secret: Move and remove bullets
Services: Get the Dragon Towers to fire the bullets to enemies
Implemented By:*Python* Library

6.2.3 Timer Enemy Module

Secret: Move enemy
Services: Waves of enemies move into the map following the path
Implemented By:*Python* Library

6.2.4 Timer Hover Module

Secret: Hover display
Services: Display a rectangle of size of dragon when player selects the dragon and hovering the dragon on board
Implemented By:*Python* Library

6.2.5 Timer Fired Module

Secret: Timer Fired
Services: Initiate all timer fired functions
Implemented By:*Python* Library

6.3 Draw Module

Secret: Draw objects
Services: Draw every game object onto the *Pygame* window
Implemented By:*Python* Library

6.3.1 Game Manager Module

Secret: Manage game data
Services: Initiates all game data
Implemented By:*Python* Library

6.3.2 Dragon Age Module

Secret: Run the game
Services: Initiates *Pygame*
Implemented By:*Python* Library

6.4 Software Decision Module

Secret: Data Structure
Services: Provides the data structure to store information from the game
Implemented By:N/A

6.4.1 Dragon Module

Secret: Dragon Party
Services: Dragon objects are created and are appended to the dragon party list for players to use
Implemented By:*Python* Library

6.4.2 Enemy Module

Secret: Enemy Wave
Services: Create enemy object and append enemies into enemy wave list
Implemented By:*Python* Library

6.4.3 Bullet Module

Secret: Bullet
Services: Create bullet object and set bullet attributes
Implemented By:*Python* Library

6.4.4 Path Module

Secret: Path
Services: Create enemy path
Implemented By:*Python* Library

6.4.5 Game Data Module

Secret: Game Data
Services: Set initial game data
Implemented By:*Python* Library

7 Traceability Matrix

The following are traceability matrices. The first trace is between the modules and the requirements specified in the requirement document. The second trace is between the modules and the anticipated changes.

7.1 Trace Between Anticipated Changes and Modules

Anticipated Changes	Modules
AC1: More game controls	M7. Game Manager Module
AC2: Sound features of the game	M7. Game Manager Module
AC3: Additional functionalities	M9. Dragon Module M10. Enemy Module M11. Bullet Module M12. Path Module
AC4: Images of the components of the game	M9. Dragon Module

Table 3: Trace Between Anticipated Changes and Modules

8 Use Hierarchy Between Modules

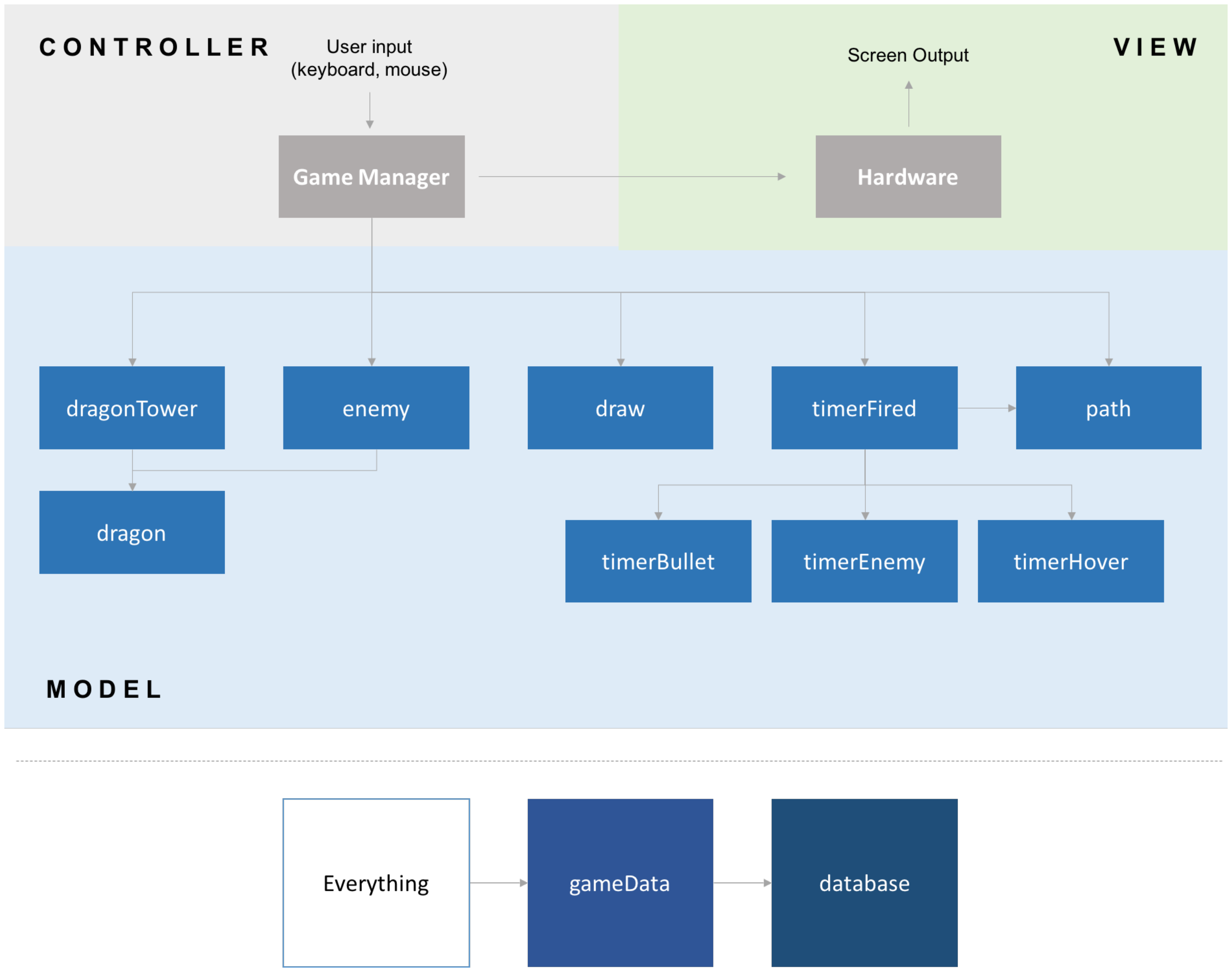
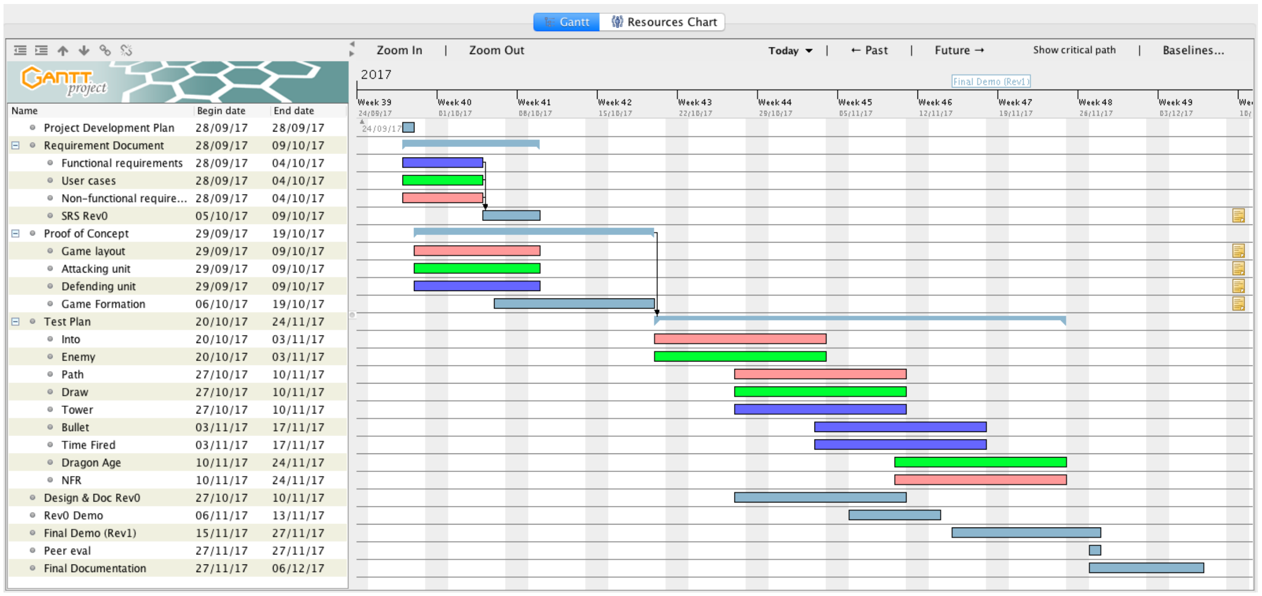


Figure 1: Uses Hierarchy Between Modules

9 Project Schedule



Name	Begin date	End date
Project Development Plan	28/09/17	28/09/17
Requirement Document	28/09/17	09/10/17
Functional requirements	28/09/17	04/10/17
User cases	28/09/17	04/10/17
Non-functional require...	28/09/17	04/10/17
SRS Rev0	05/10/17	09/10/17
Proof of Concept	29/09/17	19/10/17
Game layout	29/09/17	09/10/17
Attacking unit	29/09/17	09/10/17
Defending unit	29/09/17	09/10/17
Game Formation	06/10/17	19/10/17
Test Plan	20/10/17	24/11/17
Into	20/10/17	03/11/17
Enemy	20/10/17	03/11/17
Path	27/10/17	10/11/17
Draw	27/10/17	10/11/17
Tower	27/10/17	10/11/17
Bullet	03/11/17	17/11/17
Time Fired	03/11/17	17/11/17
Dragon Age	10/11/17	24/11/17
NFR	10/11/17	24/11/17
Design & Doc Rev0	27/10/17	10/11/17
Rev0 Demo	06/11/17	13/11/17
Final Demo (Rev1)	15/11/17	27/11/17
Peer eval	27/11/17	27/11/17
Final Documentation	27/11/17	06/12/17

Figure 2: Gantt Chart of Project Schedule