SE 3XA3: Module Guide
Mini-Arcade

Andrew Hum	William Lei	Arshan Khan	Jame Tran
huma3	leim5	khana172	tranj52
400138826	400125240	400145605	400144141

April 5, 2020

Contents

List of Tables

List of Figures

1 Introduction

1.1 Overview

Mini-Arcade is the re-implementation of three simple open-source Python games: Maze, Pong, and Flappy. These games can be accessed through a simple easy-to-use launcher. These games are to be completely re-designed with improved functionality, graphics and performance.

1.2 Context

This document is the Module Guide (MG) and is developed based upon the Software Requirements Specification (SRS) developed earlier. The SRS highlighted the functional and non-functional requirements of our system to ensure a thorough and correct solution. The MG focuses on the concept of modular decomposition to develop and display the modular structure of our project. After this document, the Module Interface Specification (MIS) will be created. The purpose of the MIS is to provide in-depth explanations of each modules individual syntax (exported access programs) and semantics (state variables, environment variables, assumptions and access program semantics).

1.3 Design Principles

The design principle guiding the module decomposition is Information Hiding. The use of this principle is important in module decomposition as it supports design for change, which is the concept of allowing each module to be modified often. This principle also supports the low-coupling, independence of modules, and high-cohesion, strong relation of elements, between the modules.

1.4 Document Structure

The document's structure is as follows.

- Section 2 lists the anticipated and unlikely changes of the software requirements
- Section 3 summarizes the decomposition of modules by Hardware-Hiding, Behaviour-Hiding, and Software Decision
- Section 4 identifies the connections between the requirements of our project and the modules
- Section 5 gives a detailed description of the module decomposition
- Section 6 includes three traceability matrices. The first matrix checks the completeness of the design in regards to the Functional Requirements provided in the SRS. The second matrix checks the completeness of the design in regards to the Non-Functional

Requirements provided in the SRS. The last matrix highlights the relations between the anticipated changes and the modules.

- Section 7 describes the use relation between modules

2 Anticipated and Unlikely Changes

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 of the program

AC2: The format of the input data.

AC3: The format and calculation of the score.

AC4: The details/style of the in-game graphics and the graphical user interface.

AC5: Player customization.

AC6: Theme customization.

AC7: The efficiency and variability of the maze generation algorithm

AC8: Create more options to configure the game (adjust sound, adjust brightness)

AC9: The number of mini-games that can be launched from the launcher

AC10: The number of high scores being kept for each game

2.2 Unlikely Changes

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

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

UC3: The purpose of the program to provide an easy and accessible way to launch mini-games.

3 Module Hierarchy

M1: Hardware-Hiding Module

M2: Launcher Modules

M3: Scoreboard Modules

M4: Maze Generator (Maze)

M5: Score Tracking (Maze)

M6: Draw Game (Maze)

M7: Player Movement (Maze)

M8: Menu and Settings (Maze)

M9: Ball Trajectory (Pong)

M10: Score Tracking (Pong)

M11: Draw Game (Pong)

M12: Player Movement (Pong)

M13: Menu and Settings (Pong)

M14: Bird Trajectory (Flappy)

M15: Score Tracking (Flappy)

M16: Map Generation (Flappy)

M17: Draw Game (Flappy)

M18: Player Movement (Flappy)

M19: Menu and Settings (Flappy)

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 and 4.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by the lecture content. 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: 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, Python

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: N/A

Table 1: **Revision History**

Date	Version	Notes
3/9/2020	1.0	Arshan and Andrew created document
3/11/2020	1.1	Arshan added modules to section 3
3/11/2020	1.2	Andrew updated section 1, and sections 2, 3, 5, 6 where Maze modules and changes were relevant.
3/12/2020	1.3	Arshan added Gantt chart and updated sections 2, 3, 5, 6 wherever Pong was relevant.
3/12/2020	1.4	William updated section 2, 3, 5, 6, 7.
3/13/2020	1.5	Revision 0
4/1/2020	1.6	William update section 6
4/3/2020	1.7	Andrew Revision 1
4/5/2020	1.8	Arshan - Revision 1

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Launcher Modules Scoreboard Modules Draw Game (Maze) Player Movement (Maze) Menu and Settings (Maze) Draw Game (Pong) Player Movement (Pong) Menu and Settings (Pong) Draw Game (Flappy) Player Movement (Flappy)
Software Decision Module	Maze Generator (Maze) Score Tracking (Maze) Ball Trajectory (Pong) Score Tracking (Pong) Bird Trajectory (Flappy) Score Tracking (Flappy) Map Generation (Flappy)

Table 2: Module Hierarchy

5.2.1 Launcher Module (M2)

Secrets: Graphics

Services: Draws the launcher and links to each mini-game

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.2 Scoreboard Module (M3)

Secrets: Graphics and score data

Services: Read/write score data from file, update score data and display them to the screen

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.3 Draw Game (Maze) Module (M6)

Secrets: Graphics

Services: Draws the Maze (generated by the Maze Generation Module), Player, and Elapsed Time (defined by the Score Tracking Module)

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.4 Player Movement (Maze) Module (M7)

Secrets: Inputs

Services: Moves the player based upon the user's keyboard inputs

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.5 Menu and Settings (Maze) Module (M8)

Secrets: Graphics

Services: Controls the input to allow the user to navigate the game features (How-to-Play, Settings, etc.) or return to the launcher.

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.6 Draw Game (Pong) Module (M11)

Secrets: Graphics

Services: Draws the current state of the game including the player's position, the ball's position, the opponent's position, and the score (received from Score Tracking (Pong) Module).

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.7 Player Movement (Pong) Module (M12)

Secrets: Input

Services: Allows the movement of the player's paddle on-screen from keyboard inputs.

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.8 Menu and Settings (Pong) Module (M13)

Secrets: Graphics

Services: Allows the user to navigate the game's features (How-to-Play, Settings, etc.) or return to the launcher.

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.9 Draw Game (Flappy) Module (M17)

Secrets: Graphics

Services: Draws the current state of the game, including the player's position, the pipe's position, and the score.

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.10 Player Movement (Flappy) Module (M18)

Secrets: Input

Services: Allows the movement of the player's bird on-screen from keyboard inputs.

Implemented By: Mini-Arcade, Python Libraries, pygame

5.2.11 Menu and Settings (Flappy) Module (M19)

Secrets: Graphics

Services: Allows the user to navigate the game's features (How-to-Play, Settings, etc.) or return to the launcher.

Implemented By: Mini-Arcade, Python Libraries, pygame

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: N/A

5.3.1 Maze Generator (Maze) Module (M4)

Secrets: Algorithm

Services: Generates a random maze based upon the desired difficulty

Implemented By: Mini-Arcade, Python Libraries

5.3.2 Score Tracking (Maze) Module (M5)

Secrets: Points

Services: Records the elapsed time to complete the maze and records the data for the scoreboard.

Implemented By: Mini-Arcade, Python Libraries

5.3.3 Ball Trajectory (Pong) Module (M9)

Secrets: Algorithm

Services: Handles the trajectory of the ball as well as the speed based on the desired difficulty.

Implemented By: Mini-Arcade, Python Libraries

5.3.4 Score Tracking (Pong) Module (M10)

Secrets: Points

Services: Keeps track of how many points are scored by the opponent and the player

Implemented By: Mini-Arcade, Python Libraries

5.3.5 Score Tracking (Flappy) Module (M15)

Secrets: Points

Services: Keeps track of how many points are scored by the player

Implemented By: Mini-Arcade, Python Libraries

5.3.6 Map Generation (Flappy) Module (M16)

Secrets: Algorithm

Services: Handles the distribution of the obtacles in the game.

Implemented By: Mini-Arcade, Python Libraries

6 Traceability Matrices

6.1 Functional Requirements Traceability Matrix

Req.	Modules
FR1	M2
FR2	M2
FR3	M2
FR4	M2, M3
FR5	M3, M5, M10, M15
~~FR6~~	~~M3, M5, M10~~
FR7	M8
FR8	M5
FR9	M4, M6
FR10	M8
FR11	M3, M5, M6
FR12	M7, M6
FR13	M8, M4, M6
~~FR14~~	~~M8~~
FR15	M19
FR16	M18
FR17	M16
FR18	M16
FR19	M15
FR20	M19
FR22	M19
FR23	M19
~~FR24~~	~~M13~~
FR25	M12, M11
FR26	M9, M10, M11
FR27	M13
FR28	M9, M10, M11
FR29	M11, M13
FR30	M13
FP1	M13, M11
FP2	M13, M11
FP3	M13, M11
FR31	M8 M13 M19

Table 3: Trace Between Functional Requirements and Modules

6.2 Non-Functional Requirements Traceability Matrix

Req.	Modules
NFR1	M6, M7, M11, M12, M19
NFR2	M2, M3, M6, M11, M13, M19
NFR3	M2, M3, M8, M13, M19
NFR4	M7, M8, M12, M13, M18
NFR5	M2, M3, M4, M6, M11, M16, M17
NFR6	M2, M3, M7, M8, M6, M12, M13, M11, M17, M18, M17
NFR7	M1, M2
NFR8	M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13
NFR9	M3, M7, M8, M12, M13, M15, M16, M17, M18, M19
NFR10	M2, M3, M6, M8, M11, M13, M15, M16, M17, M18, M19
NFR11	M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M15, M16, M17, M18, M19
NFR12	M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M15, M16, M17, M18, M19

Table 4: Trace Between Non Functional Requirements and Modules

6.3 Anticipated Changes Traceability Matrix

AC	Modules
AC1	M1
AC2	M7, M12
AC3	M5, M10
AC4	M2, M3, M6, M8, M11, M13
AC5	M6, M11
AC6	M6, M8, M11, M13
AC7	M4
AC8	M8, M11
AC9	M2
AC10	M3

Table 5: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

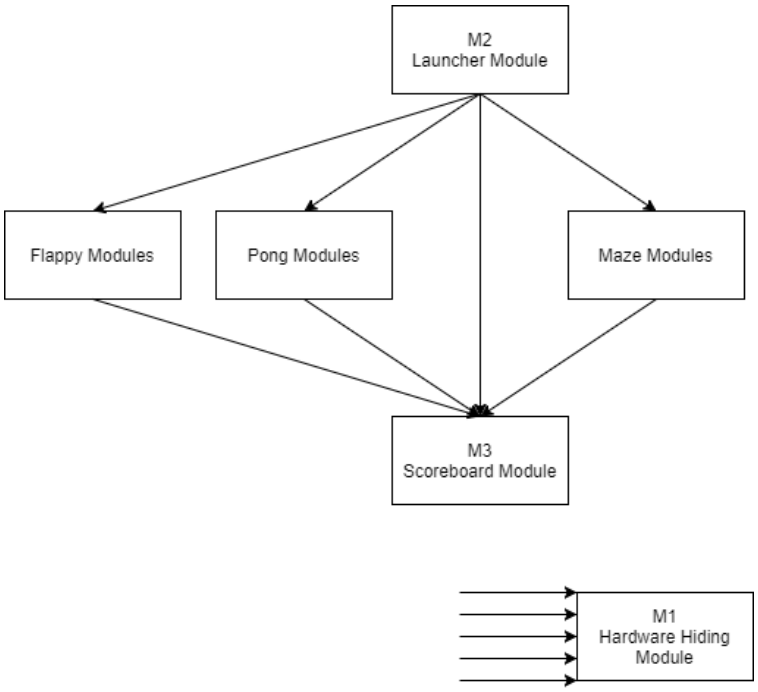


Figure 1: Use hierarchy among modules

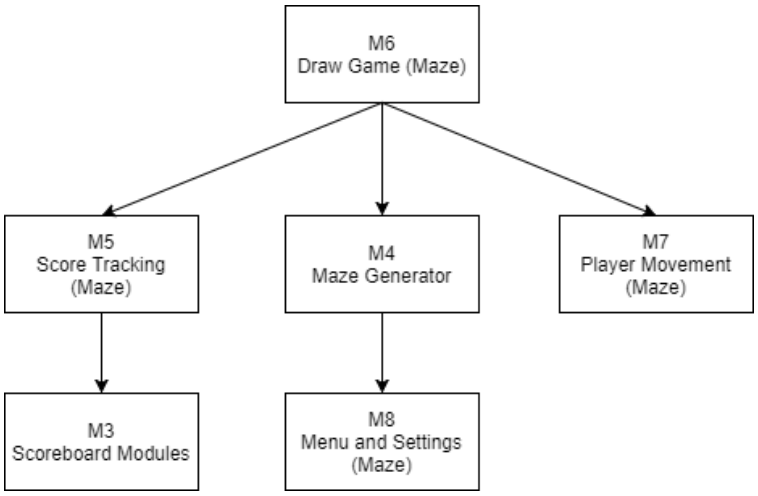


Figure 2: Use hierarchy among Maze modules

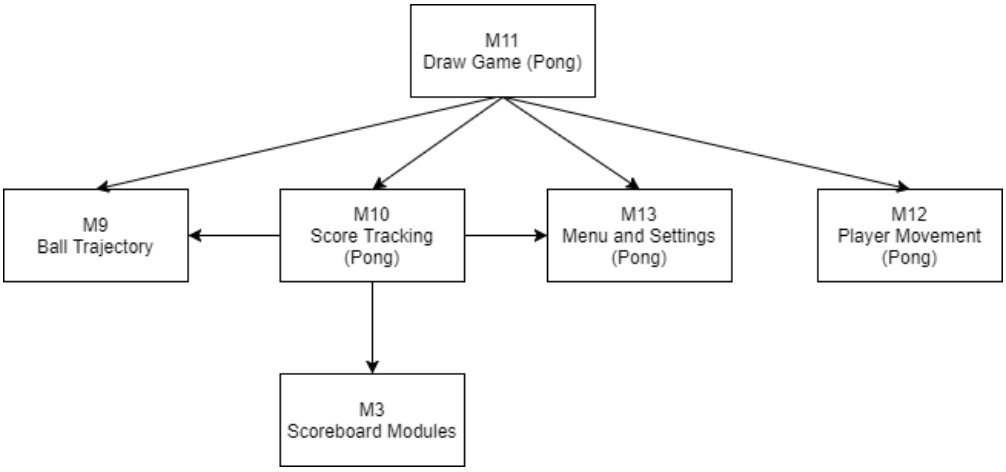


Figure 3: Use hierarchy among Pong modules

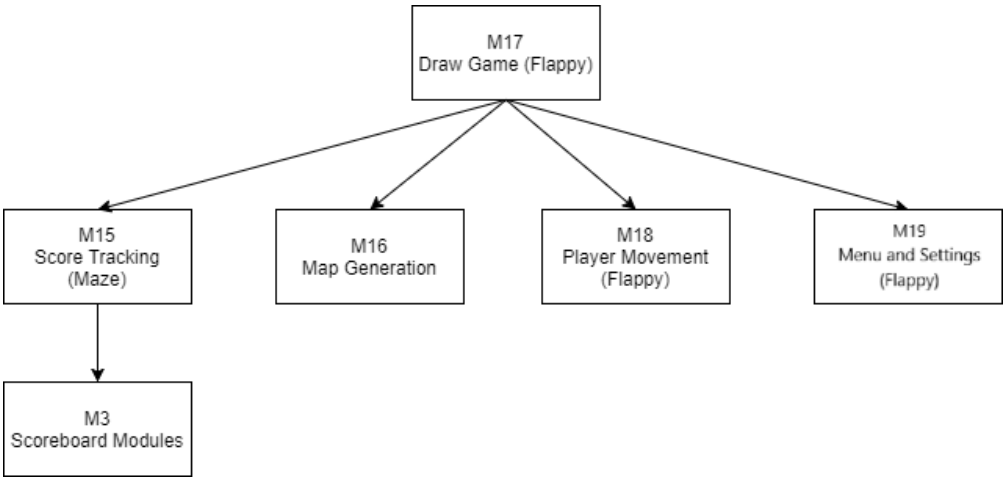
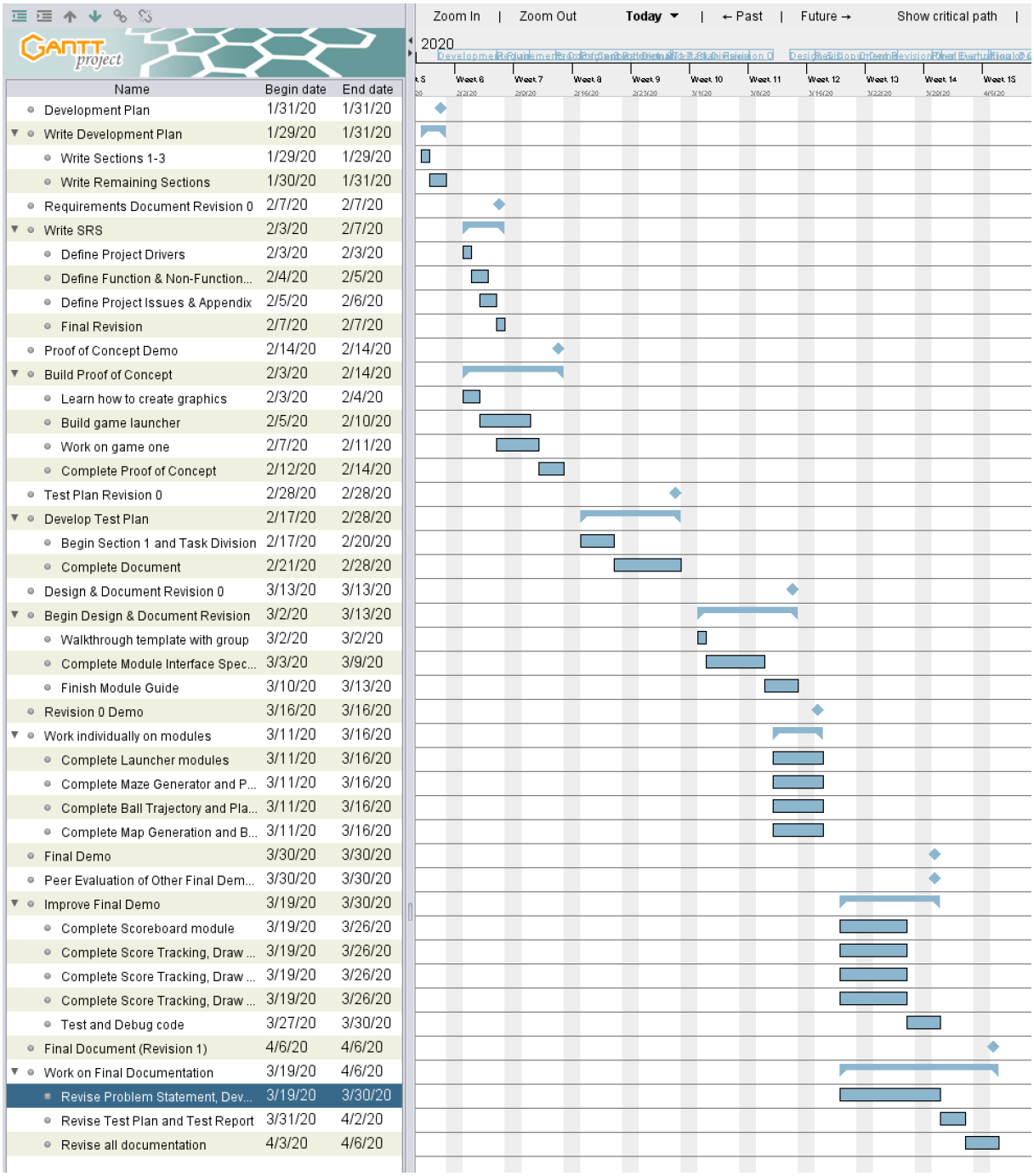
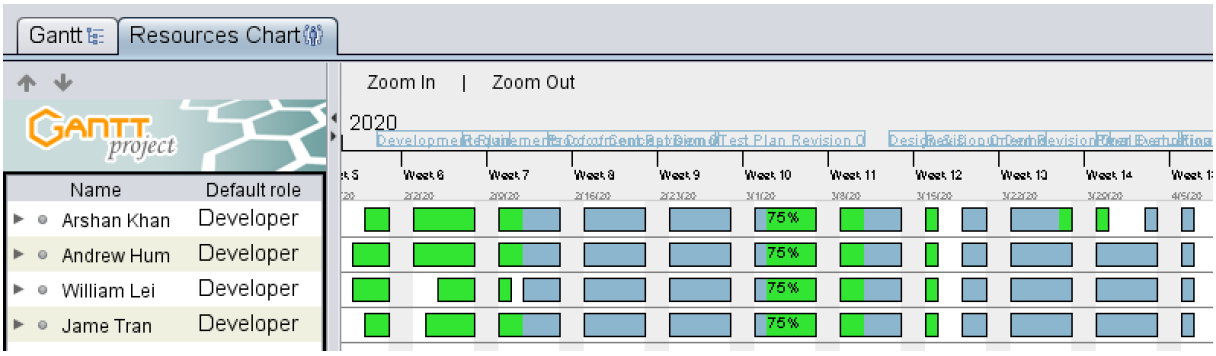


Figure 4: Use hierarchy among Flappy modules

8 Gantt Schedule



Name	Begin date	End date
Development Plan	1/31/20	1/31/20
Write Development Plan	1/29/20	1/31/20
Write Sections 1-3	1/29/20	1/29/20
Write Remaining Sections	1/30/20	1/31/20
Requirements Document Revision 0	2/7/20	2/7/20
Write SRS	2/3/20	2/7/20
Define Project Drivers	2/3/20	2/3/20
Define Function & Non-Function...	2/4/20	2/5/20
Define Project Issues & Appendix	2/5/20	2/6/20
Final Revision	2/7/20	2/7/20
Proof of Concept Demo	2/14/20	2/14/20
Build Proof of Concept	2/3/20	2/14/20
Learn how to create graphics	2/3/20	2/4/20
Build game launcher	2/5/20	2/10/20
Work on game one	2/7/20	2/11/20
Complete Proof of Concept	2/12/20	2/14/20
Test Plan Revision 0	2/28/20	2/28/20
Develop Test Plan	2/17/20	2/28/20
Begin Section 1 and Task Division	2/17/20	2/20/20
Complete Document	2/21/20	2/28/20
Design & Document Revision 0	3/13/20	3/13/20
Begin Design & Document Revision	3/2/20	3/13/20
Walkthrough template with group	3/2/20	3/2/20
Complete Module Interface Spec...	3/3/20	3/9/20
Finish Module Guide	3/10/20	3/13/20
Revision 0 Demo	3/16/20	3/16/20
Work individually on modules	3/11/20	3/16/20
Complete Launcher modules	3/11/20	3/16/20
Complete Maze Generator and P...	3/11/20	3/16/20
Complete Ball Trajectory and Pla...	3/11/20	3/16/20
Complete Map Generation and B...	3/11/20	3/16/20
Final Demo	3/30/20	3/30/20
Peer Evaluation of Other Final Dem...	3/30/20	3/30/20
Improve Final Demo	3/19/20	3/30/20
Complete Scoreboard module	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Complete Score Tracking, Draw ...	3/19/20	3/26/20
Test and Debug code	3/27/20	3/30/20
Final Document (Revision 1)	4/6/20	4/6/20
Work on Final Documentation	3/19/20	4/6/20
Revise Problem Statement, Dev...	3/19/20	3/30/20
Revise Test Plan and Test Report	3/31/20	4/2/20
Revise all documentation	4/3/20	4/6/20


Gantt
Resources Chart
Zoom In | Zoom Out
2020
Name
Default role
Arshan Khan
Developer
Andrew Hum
Developer
William Lei
Developer
Jame Tran
Developer
75%
75%
75%
75%