The Treasure Island

SFWRENG 3A04 Deliverable 2
Software Architecture Design Document

Mohinder Kallay	Kallaym	400034724
Abdullah Abdul Maksoud	abdulmaa	400205373
Namik Karaata	karaatan	400198684
Gengyun Wang	wangg40	400137547
Junhong Chen	chenj297	400213422

Contents

List of Figures

1 Introduction

The primary purpose of this document is to provide a description of the system with regards to **The Treasure Island project**. In addition to providing a system description this document will include an overview of system components, the class structure of the system and responsibilities of classes, and the overall high level architecture to be employed in the development of the system. The intended audience of this document include software engineers/developers, software architects, future developers of the system, and the management team.

1.1 Purpose

a) The purpose of this document is to begin blocking out the components of the project system. It will describe the entire system and its architecture by text and diagrams. This document will also explain the relationships and cooperation among subsystems. The audiences will have better understanding for this application.

b) The intended audience of this document are the technical stakeholders of the game. This includes:

   a) The management team (Dr.Khedri, Thien Trandinh, and Andrew Le Clair)

   b) The software developers and architects of this project (Abdullah, Mohinder, Namik, Gengyun, Jun-hong).

   c) Future developers responsible for maintenance of the project.

1.2 System Description

The Treasure Island is a game in which users compete with each other to reach the treasure island. The user that reaches the treasure island first is the winner. Competition between users will be facilitated based on the user's performance in mini games.

1.3 Overview

The remainder of the this document is divided into 3 additional subsections and is organized as follows:

- Section 2: Analysis Class Diagram - The classes that make up the overall system and their relationships.
- Section 3: Architectural Design - An overview of the software architectural design and how the overall system is decomposed into subsystems.
- Section 4: Class Responsibility Collaboration (CRC) Cards - an overview of the classes from which the system is composed, their responsibilities and how they interact amongst each other in order to accomplish system use cases.

Each of these sections is described using the appropriate textual and visual descriptions in order to convey design decisions and provide insight into the overall software architecture of project.

2 Analysis Class Diagram

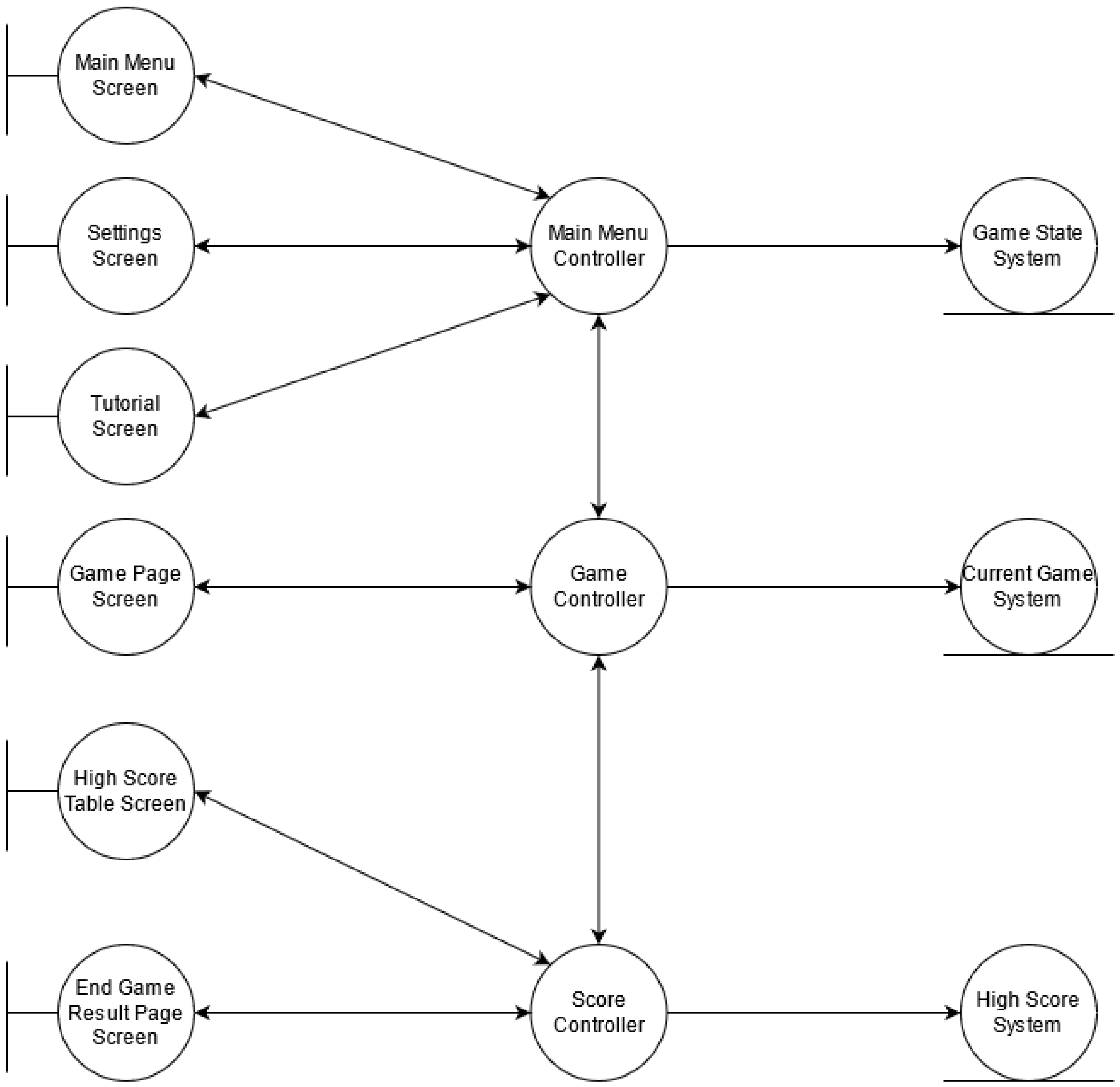

Figure 1: Analysis Class Diagram

3 Architectural Design

This section provides an overview of the overall architectural design of The Treasure Island. Overall architecture shows the division of the system into subsystems with high cohesion and low coupling.

3.1 System Architecture

The Treasure Island is a game which relies heavily on user input/output interactions and specific user interfaces for multiple subsystems. In software this type of systems are called interaction driven systems. There are two commonly used interaction oriented software architecture styles. Presentation-Abstraction-Control (PAC) is one of the two interaction oriented software architecture and is more suitable to our system than the other architecture - which is Model View Controller (MVC). PAC is a better fit then MVC in our case since MVC only has one model, view and controller component set for the entire systems which controls the business logic of the system and provide GUI. However, The Treasure Island has multiple distinct subsystems/agents (mini games) and only one component set is not enough (in a sustainable and maintainable manner) to meet each subsystems requirements. On the other hand, PAC provides us more flexibility by decomposing the overall system into cooperating agents and providing distinct component sets to each agent. Therefore, the system will use PAC architecture. The Treasure Island has multiple mini games (corresponds to agents in PAC) which each of them have their on unique program logic and GUI. PAC architecture provides us with multiple benefits to meet the unique requirements of each mini game and the overall logic of The Treasure Island:

1. PAC is decomposed into a hierarchy of cooperating agents - agents are mini games in our case. This helps to separate the business logic and GUI's of each subsystem into different parts and greatly increases maintainability and modifiability of the system.

2. PAC provides a top-level agent which connects and controls core data and business logic of The Treasure Island with each cooperating agents (mini games).

3. PAC provides three components to each agent: Presentation, Abstraction, and Control which helps us to have and control each mini game's own unique GUI in presentation component, business logic in abstraction component and user input and system's response in control component. This allow the system to make any necessary changes to presentation component or abstraction component independent from each other by reducing the coupling between the components.

4. PAC provides high cohesion between subsystems and low coupling between components.

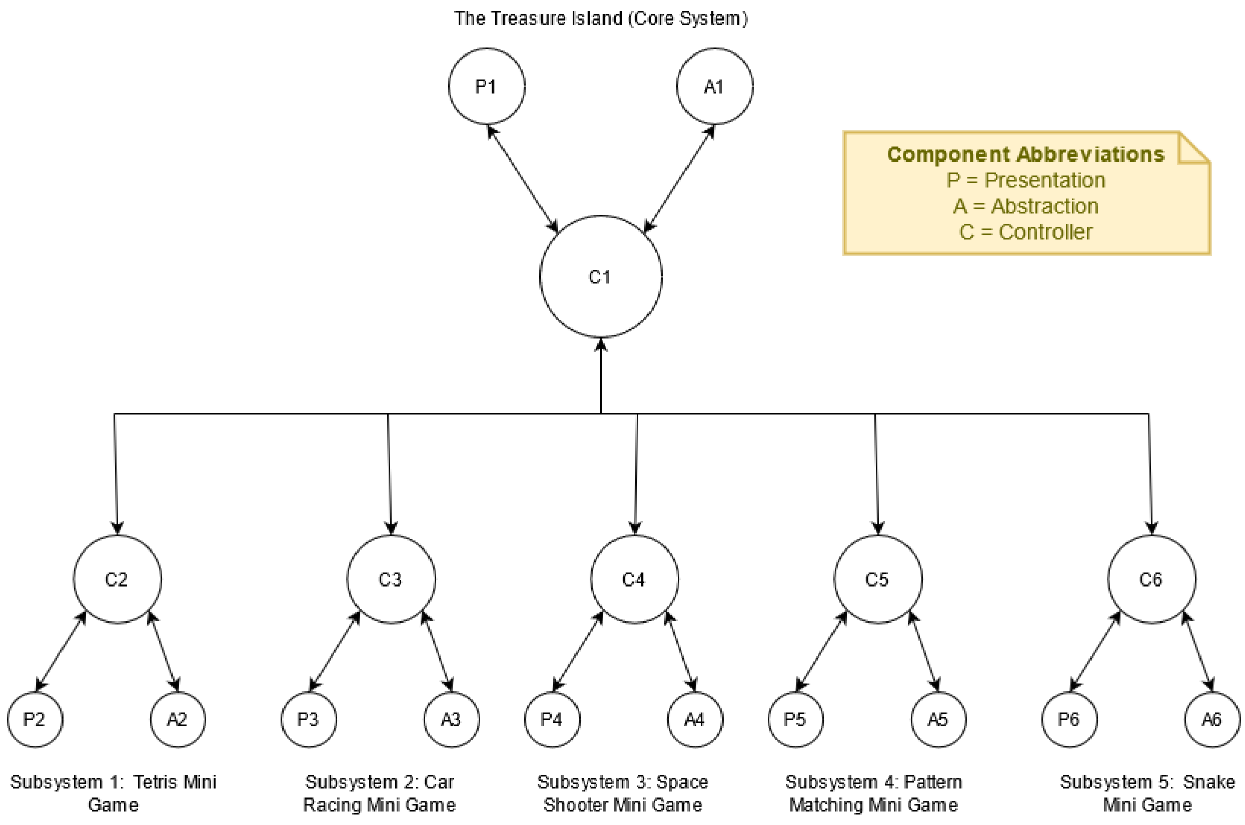
The Treasure Island (Core System)
P1
A1
C1
Component Abbreviations
P = Presentation
A = Abstraction
C = Controller
C2
C3
C4
C5
C6
P2
A2
P3
A3
P4
A4
P5
A5
P6
A6
Subsystem 1: Tetris Mini Game
Subsystem 2: Car Racing Mini Game
Subsystem 3: Space Shooter Mini Game
Subsystem 4: Pattern Matching Mini Game
Subsystem 5: Snake Mini Game

Figure 2: System Architecture Diagram

3.2 Subsystems

Subsystem 1) **Tetris Mini Game:** In this mini game, users will control the dropping blocks one by one and complete the rows like traditional Tetris game in a given time. Users gain scores for each completed row. The score will stop to increase if the accumulated blocks exceed the top of game area or time out.

Subsystem 2) **Car Race Mini Game:** In this mini game, each user controls a car, and they must avoid crashing into other objects. At the beginning of this stage, the score for each user is set to 0. Every time the user avoids crashing into an object, the score for the user is incremented. To overcome this stage, the user shall reach a specific score. If the user crashes into an object, the score is reset to 0.

Subsystem 3) **Space Shooter Mini Game:** In this mini game, users will control a spaceship to dodge enemies' bullets and shoot enemies. At the beginning of this mini-game, each player's score will be set to zero. Each player can attain scores by destroying enemies' spaceship and pick up props. Each player will have three lives, and If the player is hit by the enemies' bullets or crashed by the enemies' spaceship, the number of lives will decrease by one. Once the lives become zero, the scores will be reset to zero and that player has to replay this mini game. To end this mini game, players have to defeat the enemy boss.

Subsystem 4) **Pattern Matching Mini Game:** In this mini game, players will see multiple highlighted patterns on a grid-like map. After seeing the patterns, players will have a time period to recreate the patterns they saw. The goal of the game is correctly recreating the patterns in the shortest possible time. Players will earn points based on their patterns' correctness and their performance against time.

Subsystem 5) **Snake Mini Game:** In this mini game, each player will play a classic version of the snake game, where they will be required to control a snake and direct it to collect objects. At the beginning of the the game, the score for each player is set to 0 and the object representing the snake will be of a fixed size. As the player(s) direct the snake and successfully collect object on the screen their score(s) are incremented and the length of the snake increases as well. This cycle continues indefinitely until the user either crashes the snake into a wall or directs the snake such that the head of the snake meets the tail of the snake.

4 Class Responsibility Collaboration (CRC) Cards

This section contains CRC cards.

Class Name: Main Menu Screen	
Responsibility:	**Collaborators:**
* Provides a GUI for users to navigate to a different screen. * Allows user to make a request to exit the game.	Main Menu Controller

Class Name: Settings Screen	
Responsibility:	**Collaborators:**
* Provides a GUI for users to change the settings of the games. * Allows users to request to save the game settings they prefer such as sound settings, changing the player name etc.	Main Menu Controller

Class Name: Tutorial Screen	
Responsibility:	**Collaborators:**
* Provides a GUI to allow users to request a tutorial for The Treasure Island. * Allows users to view the rules of The Treasure Island, rules for each of the mini-games, and how to play each one.	Main Menu Controller

Class Name: Game Page Screen	
Responsibility:	**Collaborators:**
* Provides a map like GUI to show which stage the user is in. * Provide a GUI for the current mini game when the game starts.	Game Controller

Class Name: High Score Table Screen	
Responsibility:	**Collaborators:**
* Provides a GUI for users to see the top 10 high scores. * Allows user to make a request to see the high score table.	Score Controller

Class Name: End Game Result Page Screen	
Responsibility:	**Collaborators:**
* Provides GUI for users to see their performance and scores at the end of The Treasure Island Game. * Shows whether the user scored a high score and can be placed at the high score table.	Score Controller

Class Name: Main Menu Controller	
Responsibility:	**Collaborators:**
* Gets the user request to navigate to a specific screen (Tutorial Screen, Settings Screen, or Main Menu Screen)	Main Menu Screen
* Updates the Game State System based on the input provided by the user	Settings Screen
* Updates the user preferences in the Game State System if the user requests an update from the Settings Screen	Game State System
* Loads the user preferences from the Game State System every time the system is launched	Tutorial Screen
* Validates if the System shall keep running or should it stop executing based on the data stored in the Game State System	Game Controller
* Notifies Game Controller class to start a new game and update the GUI in Game Page Screen.	

Class Name: Score Controller	
Responsibility:	**Collaborators:**
* Gets the user request to see high score table.	High Score Table Screen
* Validates whether the user scored a high score by communicating with High Score System.	End Game Result Page Screen
* Provides high score data from High Score System to the appropriate Boundary Class.	High Score System
* Communicates with Game Controller class to get the user scores in the current game and calculates end of game statistics.	Game Controller

Class Name: Game Controller	
Responsibility:	**Collaborators:**
* Provides the current game statistics and data to the Score Controller class for processing end game results, validating and saving high scores.	Score Controller
* Communicates with the Main Menu Controller to get the user's preferences and game state data.	Main Menu Controller
* Gets user input and provides system response by collaborating with Game Page Screen.	
* Chooses the correct mini game based on the current game state.	Game Page Screen
* Provides current game data (such as player score, which stage - mini game - they are on etc.) to the Current Game System datastore.	Current Game System

Class Name: Game State System	
Responsibility:	**Collaborators:**
* Stores the state of the system (that is should the system keep running or should it stop executing and exit.)	
* Stores the user preferences such as player name, sound settings etc.	

Class Name: Current Game System	
Responsibility:	**Collaborators:**
* Temporary datastore for the current game (the game which is in progress). Once the user exits the game or starts a new game, Current Game System will reinitialize the data for the next game. * Stores the player locations (which island/mini game they are on), scores, and state in the current mini game.	

Class Name: High Score System	
Responsibility:	**Collaborators:**
* Stores the top 10 user scores.	

A Division of Labour

The following below includes a description of the division of labour for this document.

Mohinder Kallay
Made contributions to all subsections of section 1, section 2, and section 4.
Abdullah Abdul Maksoud
Made contributions to section 1, 2, 3.2, 4
Namik Karaata
Made contributions to section 2, 3 and 4.
Gengyun Wang
Made contributions to section 1.1, part of section 2 and part of 3.2.
Junhong Chen
Made contributions to section 2 and section 3.2.

M.K	March 5, 2021
Mohinder Kallay	Date
A.A.	March 5, 2021
Abdullah Abdul Maksoud	Date
N.K.	March 5, 2021
Namik Karaata	Date
G.W	March 5, 2021
Gengyun Wang	Date
J.C	March 5, 2021
Junhong Chen	Date