Deliverable #2

iSpaceship
Group 4, T02

Pedram Yazdinia, yazdinip
Will Conry, conrywm
Torja Istiaque, istiaqum
Akila Kavisinghe, kavisina
Xiangxin Kong, kongx9

April 8, 2020

1 Introduction

1.1 Purpose

The purpose of this document is to define the software architecture of the computer game iSpaceship from a high-level perspective. The document provides a class analysis, description of the high-level architecture of the system, description of the subsystems, and responsibility cards for each identified class.

The document is intended for developers as well as business stakeholders, including the Prof, and anyone else interested in a high-level description of the games architecture.

1.2 System Description

"iSpaceship" is a Rogue-like, turn-based, spaceship battle simulator, video game. The software product will provide the user with an engaging gaming experience where they can build their own spaceship and battle other spaceships. The software product will be used for the enjoyment of the user and provide them with a sense of self-accomplishment. The objective is for users to have fun and feel rewarded when they put in the time and dedication to progress in the game. The application should hold interest of users over a long period of time.

1.3 Overview

This document is divided into five sections, including the above Introduction section. In section 2, the analysis class diagram is given. This diagram emphasizes the structure of the objects within the system. In section 3, the architectural design is described. This section includes descriptions of the overall system architecture and the divided subsystem architecture. Finally in section 4, the class responsibility collaboration (CRC) cards are provided. This section outlines each class required in the system, what it does and its collaborators.

2 Analysis Class Diagram

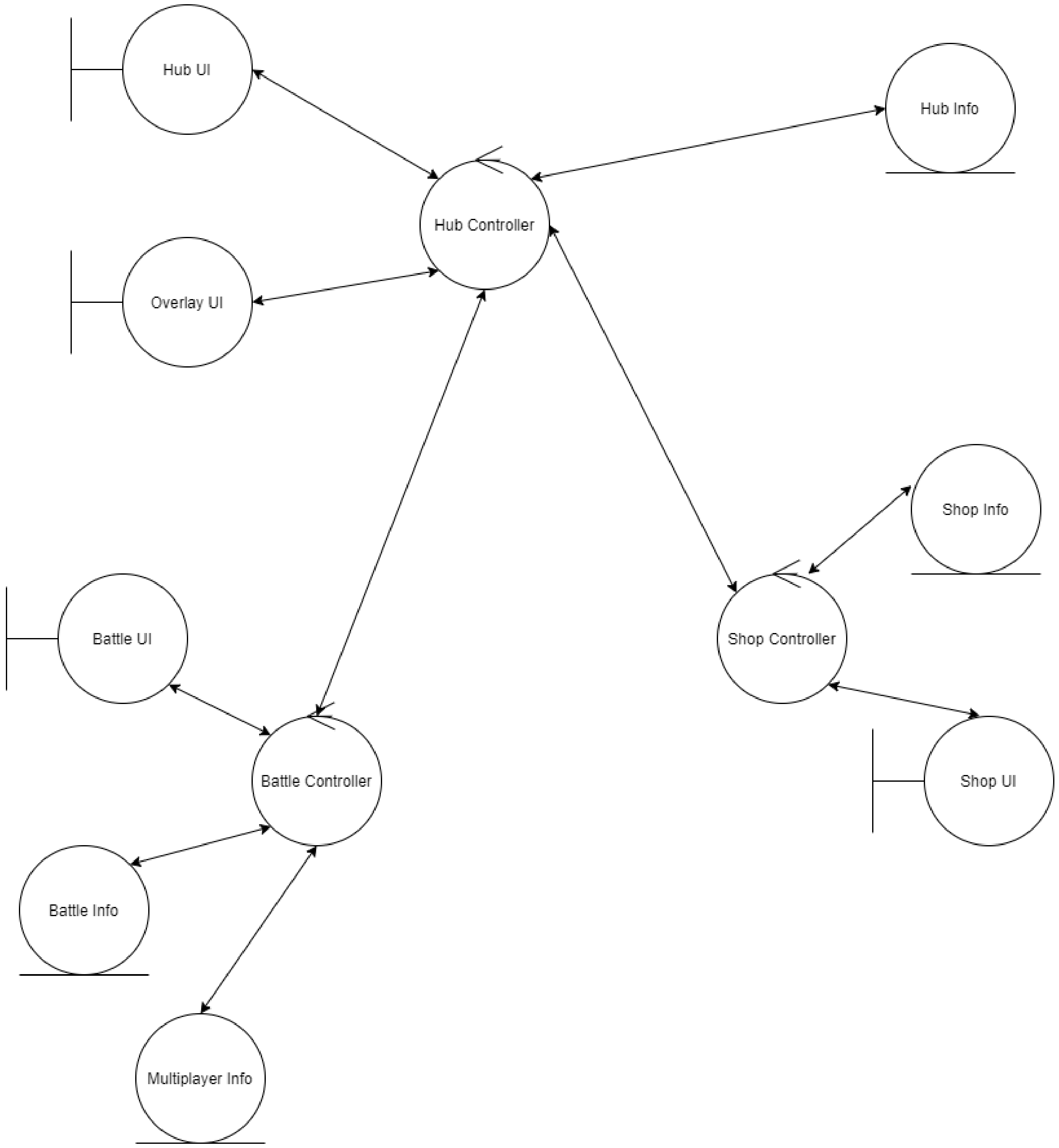

Figure 1: Class analysis diagram for the iSpaceship game.

3 Architectural Design

3.1 System Architecture

The system will implement the Presentation-Abstraction-Control (PAC) architecture. PAC is an interaction oriented software architecture that breaks a system into a hierarchy of communicating agents. Each agent has three components known as Presentation, Abstraction, and Control. The Presentation component handles interactions with the system interface. The Abstraction component takes care of data manipulation and storage. The Control component is the communication point for the Abstraction and Presentation components. The Control component also communicates between agents.

PAC architecture was chosen because of the interactive nature of the system being built; iSpacehip requires user input to drive its functionality. PAC allows interactions to be easily designed for due its distribution of system responsibilities. By having a separate presentation, abstraction, and control component user-interactions can be built in a more precise and less-coupled manner. iSpaceship has entity modules that hold information such as players stats and ability information; these can be easily translated to abstraction components. The game also has several boundary classes such as battle and hub which will display screens and received user input; these can be transformed to presentation components. Finally the controller classes such as the Shop or Battle help control and update data from each other; these can be translated to Control components. Finally the hierarchical aspects of PAC can be seen in the system as the top subsystem can be considered to be the Hub. The hierarchical relationship through which the subsystems are connected can be better seen in the package diagram below.

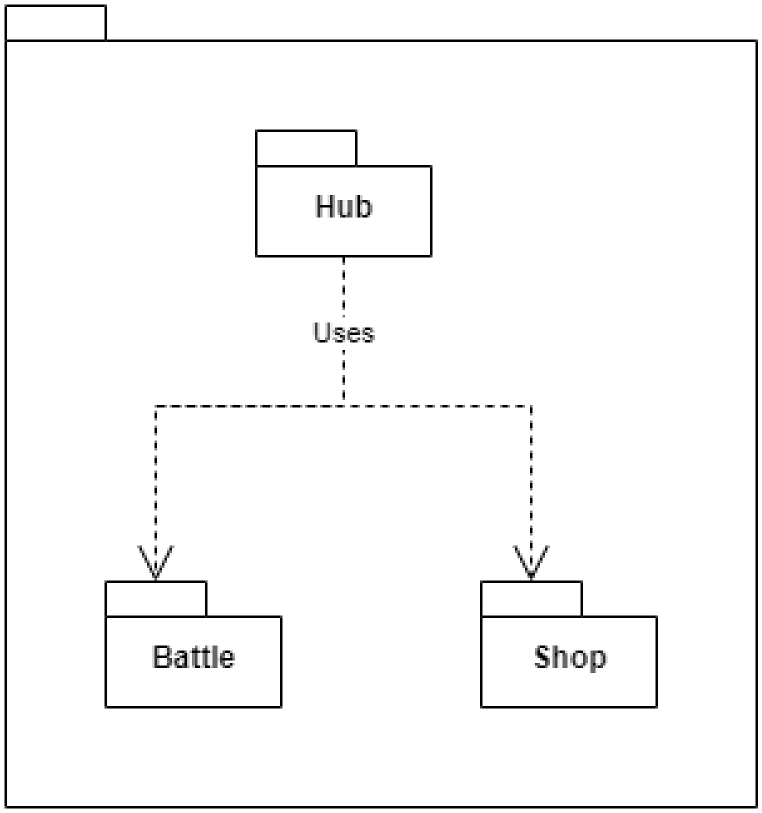

Figure 2: Structural architecture diagram for the iSpaceship game.

3.2 Subsystems

i **Hub**: The top level agent of our PAC architecture. It also displays gateways to the main views of the game: the Shop, and the Battle. It also controls the game state and session data and allows the user to save or start a new game.

Also controls the state of the players stats. Including: number of lives, amount of currency, and purchased items. It is stores information for the Shop and Battle subsystems.

ii **Shop**: Displays the in game item shop where the user can purchase items with in game currency. It controls which items the user can buy and which item the user has by modifying the purchased items in the Hub.

iii **Battle**: Displays the battle mode of the game where the user battles an opponent. An option is presented for either single player or multiplayer. It records the winners of battles and is responsible for unlocking levels. The player selects abilities to use from another subsystem and they are displayed here.

Also displays ability options for the user to choose from. It calculates how much damage the player and opponent take each turn using a function of their stats, chosen ability, and a random number. Has a logical component for single and multiplayer.

4 Class Responsibility Collaboration (CRC) Cards

Class Name: Hub Controller	
Responsibility:	**Collaborators:**
Game Initialization	
Main controller of the game	
Accesses game data to transfer to UIs	Hub UI, Overlay UI
Regulates and computes information being relayed by other controllers	Shop Controller, Battle Controller
Updates mission results to hub info.	Hub Info, Battle Controller
Generates in-game currency	Hub UI, Hub Info
Switches game states and gives main control to other controllers	Shop Controller, Battle Controller

Class Name: Battle Controller	
Responsibility:	**Collaborators:**
Accesses battle data to transfer to Battle UI.	Battle UI
Computes damage calculations and changes information accordingly.	Battle Info
Updates battle results to hub controller.	Hub Controller
User is either in multiplayer or in single player story mode.	Multiplayer Info

Class Name: Shop Controller	
Responsibility:	**Collaborators:**
Accesses shop data to transfer to shop UI and vice versa.	Shop UI, Shop Info, Hub Controller
Updates shop info based on input of shop UI.	Shop UI, Shop Info, Hub Controller

Class Name: Hub UI	
Responsibility:	**Collaborators:**
Presents interactive hub interface for user.	Hub controller
Takes input from user to change game states.	Hub controller
Displays spaceship status (level, currency).	
Users can collect generated currencies.	

Class Name: Overlay UI	
Responsibility:	**Collaborators:**
General UI to display basic information during all states of the game.	Hub Controller
Acts as a main "desktop" to overlay other UIs.	Hub Controller

Class Name: Battle UI	
Responsibility:	**Collaborators:**
Provides interactive battle interface for user.	Battle Controller
Takes in ability input from user.	Battle Controller
Displays statistics and battle events.	Battle Controller

Class Name: Battle Info	
Responsibility:	**Collaborators:**
Contains mission information for all missions.	Battle Controller
Contains damage and other ship stats of current battle.	Battle Controller

Class Name: Multiplayer Info	
Responsibility:	**Collaborators:**
Contains battle information being relayed from multiplayer opponent.	Battle Controller
Contains multiplayer stats of the player.	Battle controller

Class Name: Shop Info	
Responsibility:	**Collaborators:**
Contains shop entities and respective prices to be displayed in shop.	Shop Controller
Keeps tabs on whether item is owned or not.	Shop Controller

Class Name: Hub Info	
Responsibility:	**Collaborators:**
Contains all general game information (player name, currency amount, global time, story progression, etc.).	Hub Controller

A Division of Labour

All members are responsible for 20% of the work for each milestone. The work for this document was divided equally amongst all group members.

Approved: ______________________________
Group Member 1

Approved: ______________________________
Group Member 2

Approved: ______________________________
Group Member 3

Approved: ______________________________
Group Member 4

Approved: ______________________________
Group Member 5