LIFDOFF

Project Requirements Document

Conquer the Seas

Benson Perry, Matt Dannenberg, Brian Shaginaw

November 4, 2011

Contents

1. Introduction

***Conquer the Seas* is a turn-based strategy game in which both players select moves simultaneously. The game puts two players against each other as they maneuver underwater ships to destroy oncoming enemies and collect resources, winning when the other player's ships have been destroyed.**

1.1 Purpose

This Software Requirements Specification (SRS) is a description of all functions and specifications of *Conquer the Seas*. It gives a comprehensive look at all aspects of the game. This document is based on the IEEE 830 Standard.

1.2 Scope

This section outlines the function and goals of this project. It gives a sense of what problem this software sets out to solve and the market it seeks to occupy.

This product is a single- and multi-player game with a nautical theme. It can be played against AI controlled opponents or other humans over a network (either internet or local network). The goal of this software is to provide a fun and complex game that can provide entertainment to a variety of different users. It will contain a resource system, multiple paths for upgrading of units, and complex strategies that will develop over the lifespan of the game.

1.3 Definitions, acronyms, abbreviations

This section outlines the terminology and acronyms used throughout this document.

1.3.1 Definitions

- **Client:** The local instance of the game GUI on a player's machine.
- **Server:** The unique hosted game to which other clients connect. The server is hosted on the machine of one of the players.
- **Player:** The user playing the game using the client GUI.
- **Host:** The player who is running the server in addition to running their client. This is the player "hosting" the multiplayer game.
- **Action:** A command that players can choose to issue during their turn.
- **Kick:** To remove a player from the multiplayer lobby and close the connection to their computer.

1.3.2 Acronyms

Acronym	Definition
AI	Artificial Intelligence
(G)UI	(Graphical) User Interface
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area Network
SRS	Software Requirements Specification
TCP	Transfer Control Protocol
UML	Unified Modeling Language

1.4 References

- IEEE Recommended Practice for Software Requirements Specifications. New York: Institute of Electrical and Electronics Engineers, 1998.

1.5 Overview

Chapter 2 provides a description of the game itself, with textual explanations of the possible interactions a player can have with the game, such as actions, victory conditions, and basic gameplay. Chapter 3 contains annotated screenshots and use cases to detail all processes explained in Chapter 2.

2. Overall Description

This section describes the game scope and functionality. The subsections below outline an overview of the product, the available functions, associated risks and solutions, user characteristics, and apportioning of requirements.

2.1 Product Overview

This section is a summary of the software and its functions.

This product will be a complete single- and multi-player game. It will have functionality for the saving and loading of games, for networked multiplayer from different machines, and for single player versus a computer controlled player (or players) of different difficulties. The game features customizable units with multiple upgrade paths, allowing for many unique combinations which gives a large amount of replayability to even the single player game.

It will use a package of external libraries for game design called pyGame. Other resources like graphics will be created by the LIFDOFF team. Multiple GUIs will be available throughout the game for players to interact with:

- The **Main Menu** screen will be displayed as soon as the game application is launched. Several options will be available: New Game, Load Game, Options, Credits, and Exit.
- The **Multiplayer Lobby** screen will be displayed after the Join Multiplayer or Host Multiplayer options are chosen from the Main Menu. In the Multiplayer Lobby, current players will be displayed, as will a chat box for players to talk to one another before the game begins.
- The **Game Screen** is the main view for each player once they have entered a game. This screen shows both player's units, along with the game board and current statistics. It also has buttons to open the Shop and Upgrade screens, and a button for the Pause Menu.
- **In-Game Views** are the multiple screens accessed from within the Game Screen:
    - The **Shop Screen** is accessed from the Game Screen or from the Upgrades Screen. This allows players to purchase new offensive units to send at the opposing players using the money they have collected. Players can also upgrade the units they have already purchased.
    - The **Upgrades Screen** is accessed from the Game Screen or the Shop Screen. This is the screen where players spend the experience points they have accrued on upgrades for their three defensive ships.
    - The **Pause Menu** is accessed from the Game Screen. It has options for adjusting sound settings, opening help, and saving or quitting the current game.
    - The **Help Menu** is accessed from the pause menu. It explains game features and keyboard shortcuts.
- The **Credits** screen displays information about the developers of this software.

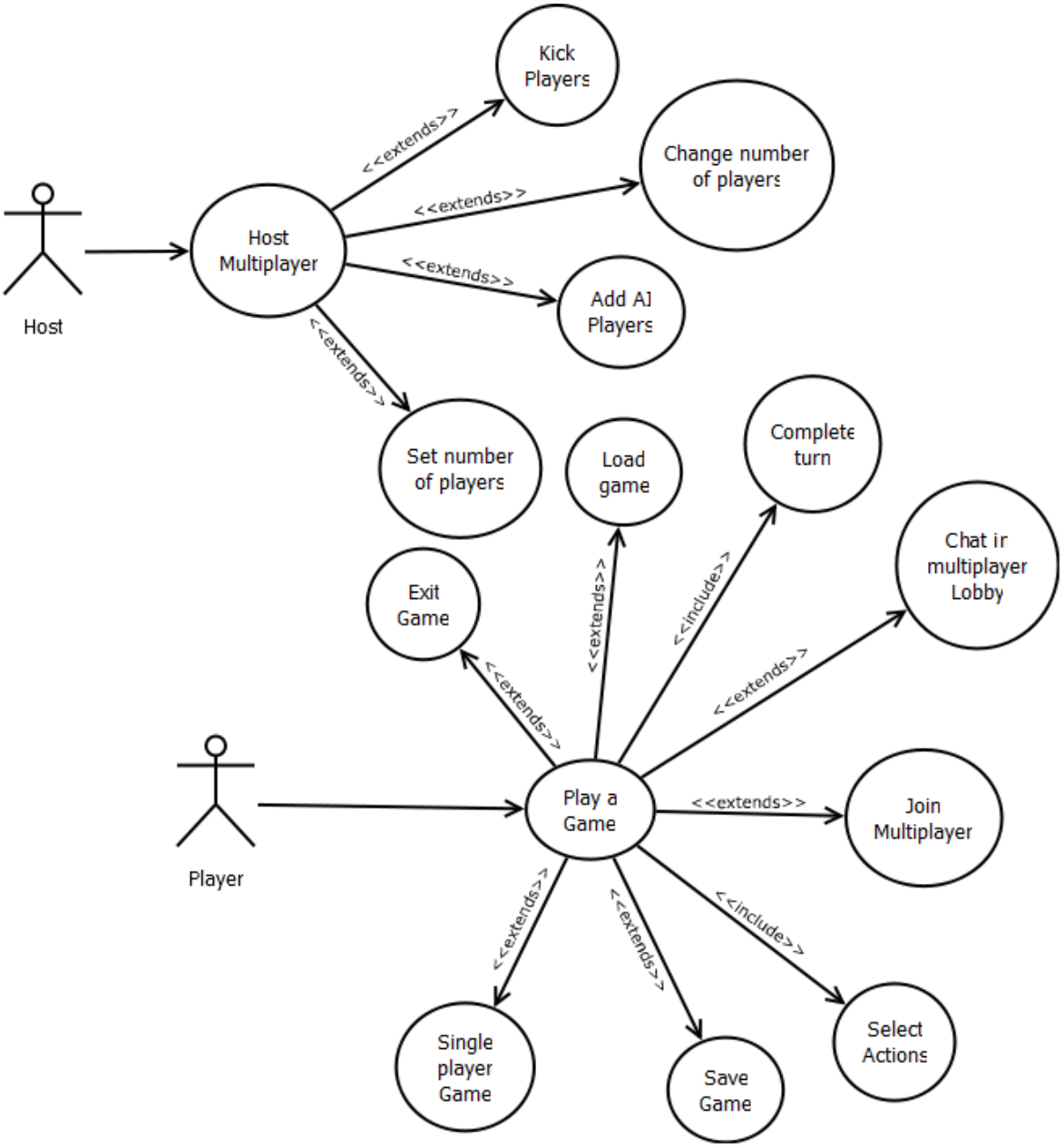
Because *Conquer the Seas* will be written in Python 2.7, this version or a compatible version of Python must be installed on all machines running the game. *Conquer the Seas* will be an entirely stand-alone game that will not require any resources besides Python on the machine it runs on.

2.2 Product Functions

This section outlines the different functions of the *Conquer the Seas* software. Main functions are the overall objectives of the software, and user functions are the actions users may take while inside one of the main functions.

- Main Functions:
  - Host a game
  - Play a game
- User Functions:
  - Host multiplayer
    - Set number of players
    - Add AI players
    - Kick players
    - Change number of players
  - Join multiplayer
  - Start a single player game
  - Load game
  - Chat in multiplayer lobby
  - Select actions
  - Complete turn
  - Save game
  - Exit game

2.2.1 Use Cases



2.3 User Characteristics

This section outlines the different goals, features, and interface expectations of different types of users.

User	Specific Goals	Features	Interface Expectations
Casual Gamer	Have fun during first time playing game	Easy to learn	Intuitive even for inexperienced gamer
Moderate Gamer	Have fun quickly, enough depth to entertain on multiple plays	Not too difficult to learn, enough of a learning curve to make experience valuable	Intuitive, could be complicated if it is similar to other games of the same style
Intense Gamer	Lots of strategy, many levels of depth to learn	Lots of features, experienced player can beat inexperienced player almost every time	Intuitive, offers hotkeys for easier and quicker pace, can be as complicated as necessary
Server Host	Can easily set up server, little to no options to configure	Easy to set up, lack of unnecessary or overly complicated features	Simple interface

2.4 Constraints

This section outlines the constraints placed upon this software. It details individual constraints and the solutions that must be taken to prevent them from negatively impacting the software.

Obviously, hardware limits certain features of the game. The game must run in a small enough amount of RAM that most computers can play it with no decrease in functionality. This means that the number of units on the screen may need to be limited to stay at an optimal level of performance.

Because players will be playing from their own unique machines, steps must be taken to ensure stable (or at least predictable) performance when outside factors interfere with gameplay. Turns will be time-limited, so that if a player is taking too long, other players will not have to suffer indefinitely. Also, if a player disconnects due to loss of power or network issues, the game can be restored from a temporary save that will be written periodically without player interaction.

Player skill and game familiarity must be taken into account. As developers, the game will be incredibly familiar, but a player who has never seen the game before needs to be able to quickly pick up and understand the game without too much help.

2.5 Assumptions and Dependencies

This section details the assumptions the LIFDOFF team will make in regards to the users and machines that are running the software in addition to the necessary dependencies for successful execution.

It is assumed that all machines have Python 2.7, as this is necessary for the game to run. It is also assumed that any users that wish to engage in a multi-player game are connected to the internet or a local network. It is also necessary that each machine has the ability to connect to or host a server, and thus the network cannot be blocked by a firewall that restricts the ability of the *Conquer the Seas* software to communicate with a remote server or client.

2.6 Apportioning of Requirements

This section details the features that may not make it into the *Conquer the Seas* software. These features may be delayed because of a lack of time or because of a lack of experience from the development team.

ID	TITLE	DESCRIPTION	DELAY REASON	WHEN
1	Multiple Upgrade Trees (per ship)	Each ship could have 3 possible upgrade trees resulting in 9 unique ships, this can be reduced to one per ship	Run out of time	Starting April 10, takes 2 additional weeks
2	Screen Transitions	Screen transitions between different menus (main menu background scaling down to be the actual background of the game, etc)	Inexperienced development team	Starting April 24, 1 additional week
3	>15 Items	Having a large number of items might be unrealistic	Could be unrealistic to expect so many items, inexperienced development team	Starting May 1st, 2 additional weeks

2.7 Risk Management

This section outlines the risks this project will face and details their impact, probability, and exposure. It also describes the risk control methods that will be used to account for these risks.

2.7.1 Risk Assessment

ID	Title	Description	Probability	Impact
1	PyGame library lacks functionality	PyGame library could not have methods or modules that are necessary for our game	High	High - the cost of impact would be having to change or rewrite features incompatible with PyGame libraries
2	AI Issues	The AI player for single player could be too strong/too weak	Medium	Medium - imbalanced AI would not be game-breaking but would make it much less fun for single player.
3	Randomly generated terrain is unfair	Each player has randomly generated terrain - if it isn't fair, one player could be at a significant disadvantage	Medium	Medium - this would make the game much less fun
4	Difficult for users to understand interface	The game is too difficult for new players to pick up and play without having lots of problems	Low	Medium - this could severely deter one portion of the userbase from enjoying this product

The risks associated with this project are not too major, although some have a high chance of occurring. The chance of PyGame lacking some functionality that *Conquer the Seas* requires is very likely. It is moderately likely that unfair AI or unfair randomly generated terrain will occur. The interface being difficult for new users to understand is not something that is likely to happen, so making large changes to accommodate for this may not be worth it.

2.7.2 Risk Control

ID	Type	Leverage	Solution Summary
1	Assume	5	Plan for the fact that adjustments will have to be made to compensate for incompatibilities or missing features from PyGame
2	Transfer	2	Make AI too weak and make it stronger as time permits
3	Assume	2	Develop a metric to judge impact of terrain, if two random terrains have very different impacts, re-generate them
4	Avoid	.3	It is not worth the time to redesign the UI to accommodate specific user concerns

The risks involved are not extremely dangerous. The PyGame library lacking functionality seems like a large problem, but it contributes enough functionality that writing the missing functionality is less work than rewriting the functionality it does contribute. We will be able to easily supplement any missing features with those of our own. AI issues only requires maintaining a weak AI and making it stronger until it reaches a fair level of intelligence. Randomly generated terrain can be unfair because of the nature of randomness, but it can be assumed and compensated for by regenerating terrains that have different levels of impact.

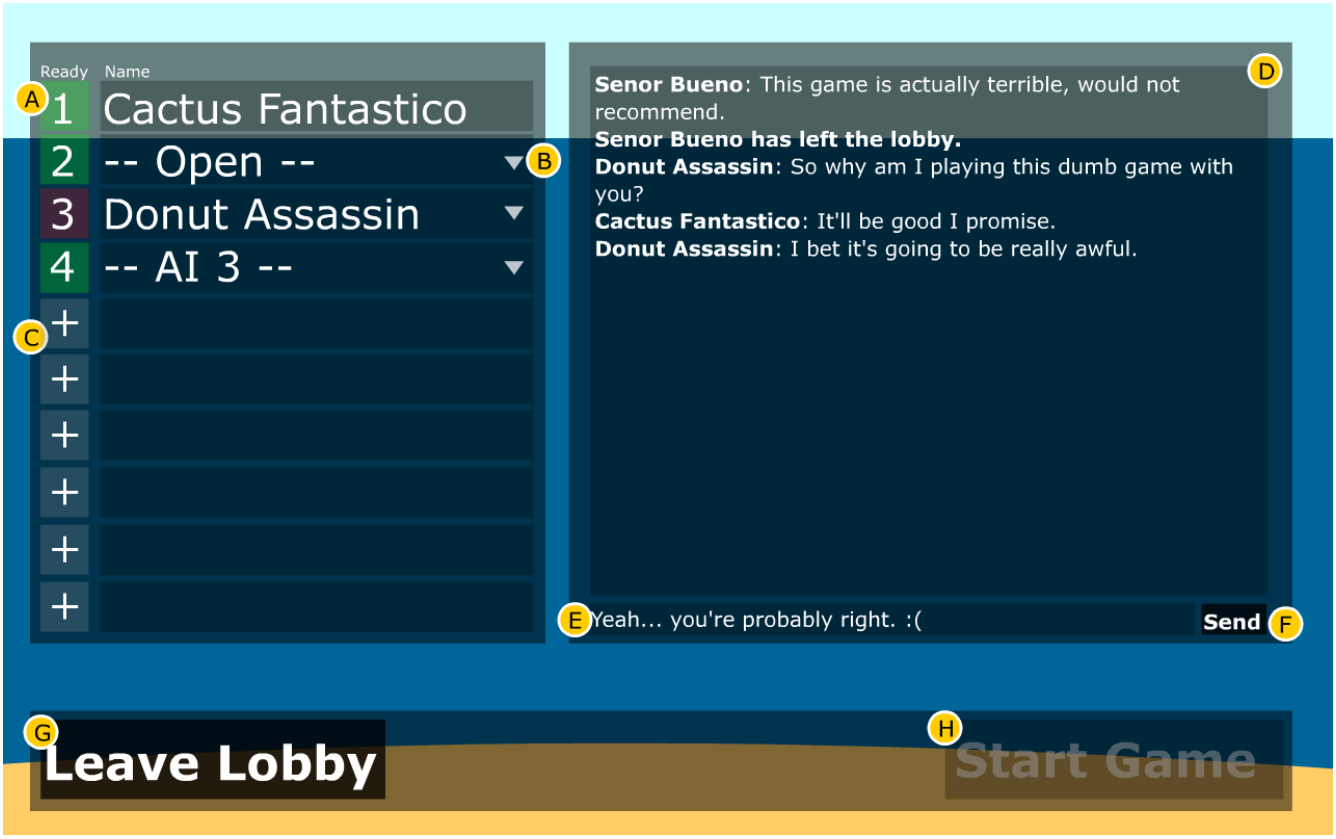
3. Specific Requirements

3.1 Screenshots

3.1.1 Main Menu

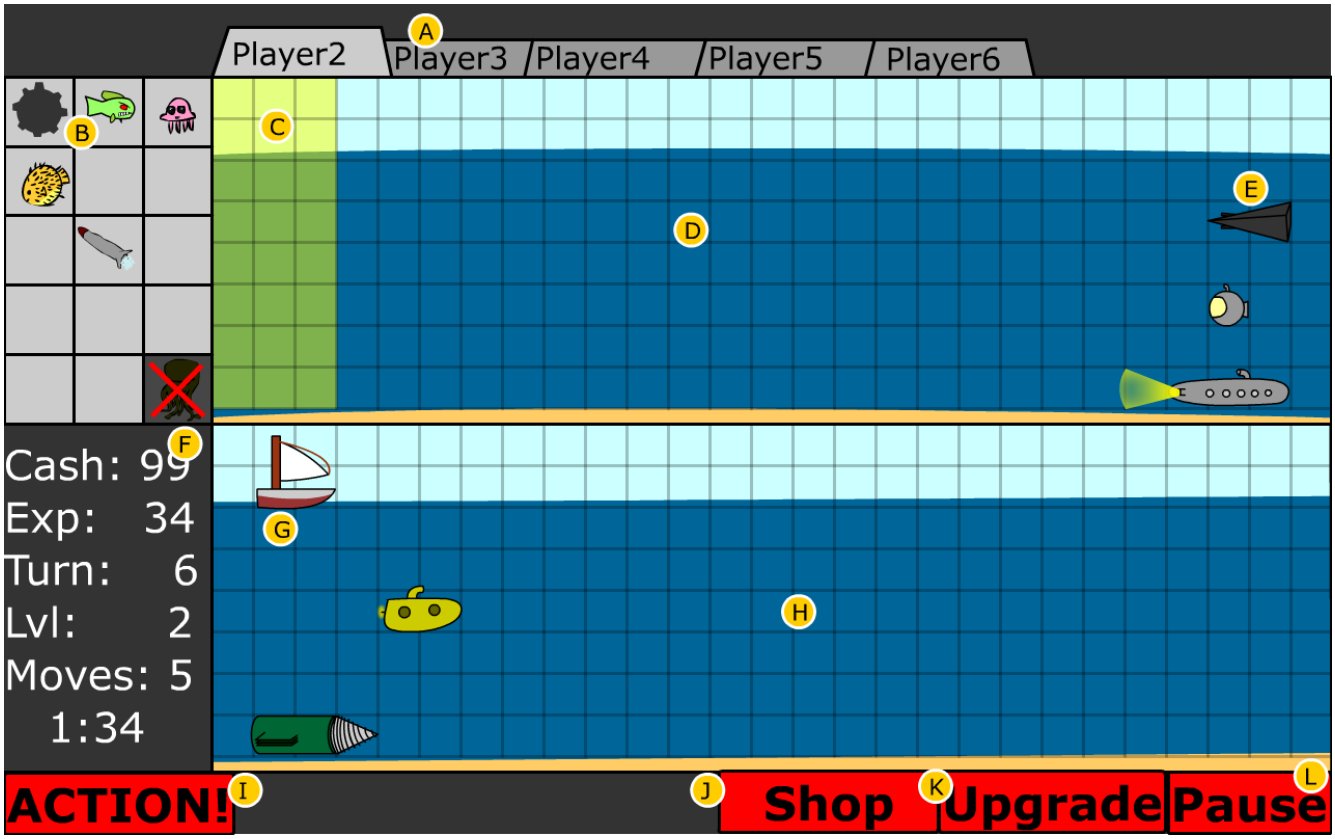
A. **Logo:** The games logo
B. **New Game:** Pops up submenu for game type (Bi, Bii, Biii)
    Bi. **Single Player:** Starts single player game
    Bii. **Host Multiplayer:** Starts a new server and puts you in the multiplayer lobby
    Biii. **Join Multiplayer:** Opens a box to put in the IP address to connect to, once entered takes user to multiplayer lobby
C. **Load Game:** Pops up a system-native dialogue box for selecting a save file
D. **Options:** Pops up submenu for setting options (Di, Dii, Diii, Div)
    Di. **Multiplayer Name:** Text box to enter handle
    Dii. **AI Level:** Difficult level of the AI (higher number is more difficult)
    Diii. **SFX:** Volume of game noises that are not music (1-10 scale)
    Div. **Music:** Volume of in game music (1-10 scale)
E. **Credits:** Displays credits
F. **Exit Game:** Quits game and cleans up

3.1.2 Multiplayer Lobby



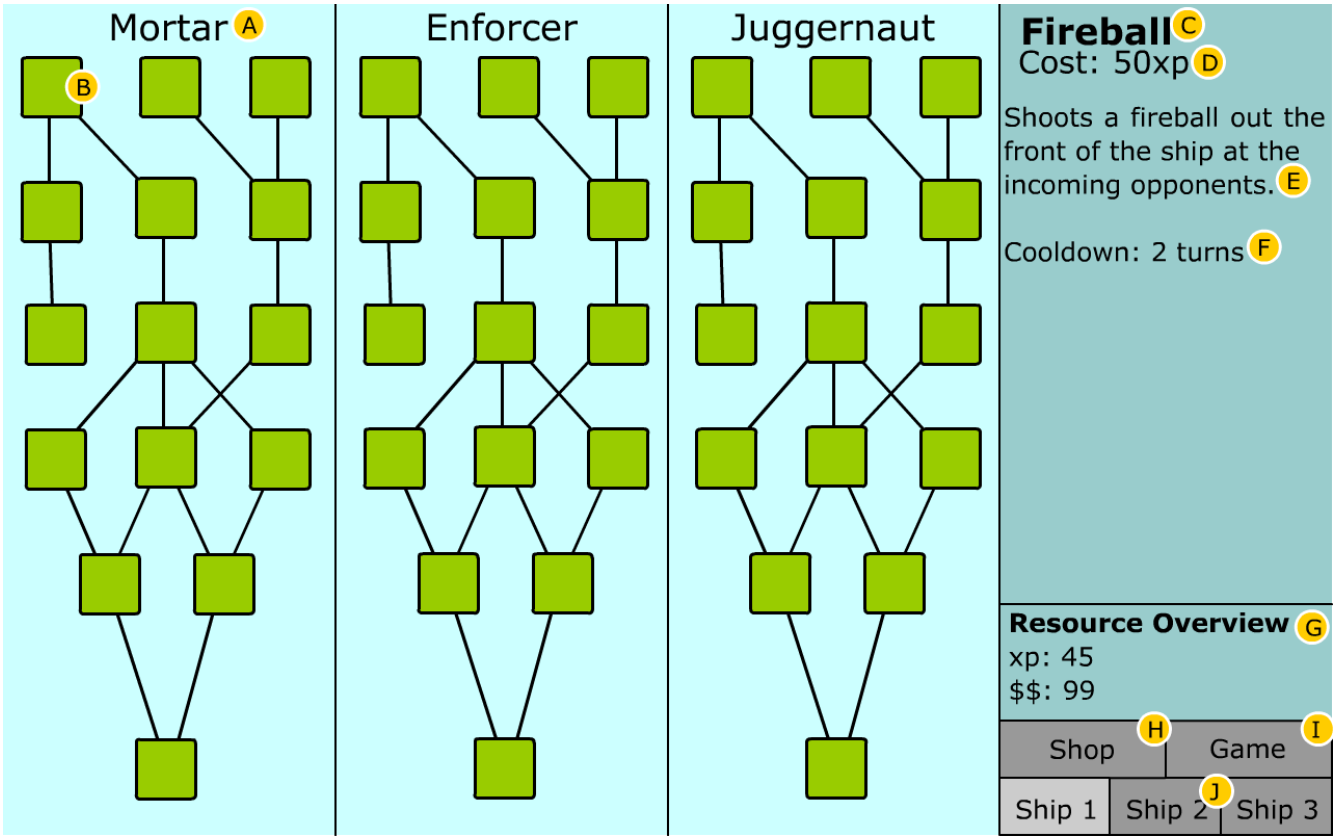
A. **Player Box:** Player number and readiness indicator (green is ready, red is not)
B. **Player Menu:** Drop down menu to select player type (choice between AI with difficulty level, open to allow another player to join, and close). If a player has joined, his handle is displayed in this field. This is only changeable by the server host.
C. **Add a player:** Creates another slot with the drop down menu in B
D. **Chat area:** Displays messages sent by the players and notifies users when players join/leave
E. **Text entry field:** Displays text to be sent to the other players
F. **Send button:** When clicked sends text from the text entry field (enter will work if one does not want to click)
G. **Leave Lobby:** Puts player back to the main menu
H. **Start Game:** Starts game immediately, only the player who hosted the game can click this

3.1.3 In Game UI



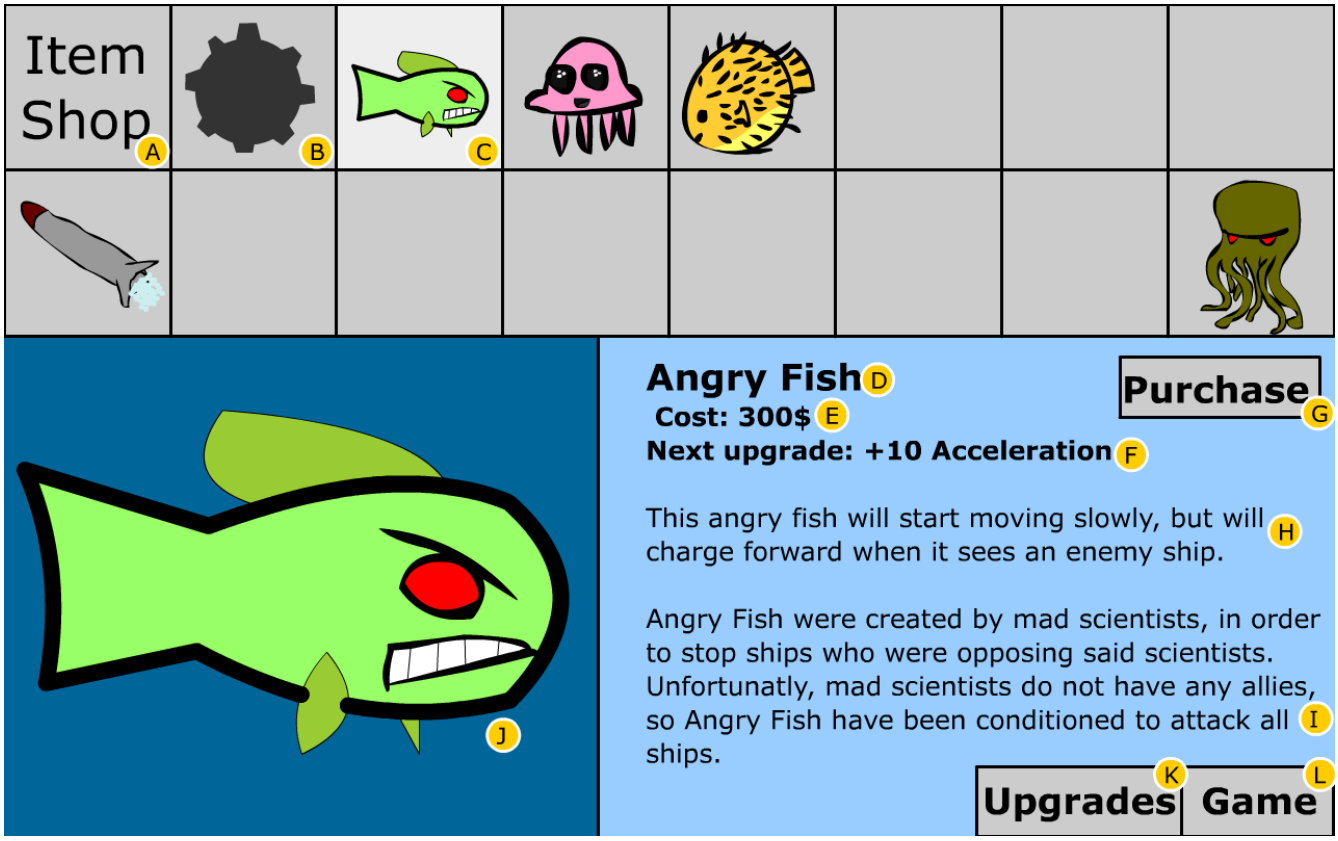
A. **Enemy Tabs:** Clicking one of these displays that opponent's ocean in D. The light colored larger tab indicates the currently selected opponent. Instead of Player2, it will display that opponent's handle.
B. **Offensive Unit Area:** A graphical display of the units the player can send at their opponent
C. **Deployable Space:** The area in which the player can deploy their offensive units
D. **Opponent's Ocean:** A display of the currently selected opponent's ocean
E. **Opponent's Defensive Units:** Shows the current state of the opponent's defensive units. This will allow players to know if the opponent's vessels are injured and how they have been upgraded.
F. **Stats:** Displays all the values relevant to the current turn: how much money the player has, how much experience the player has, what turn it is, what level the player has reached, and how many moves that player has left in their turn.
G. **Player's Defensive Units:** The player can select these to order them to move or shoot as desired
H. **Player's Ocean:** Displays the current status of the player's ocean, so that they can plan their moves accordingly.
I. **Action! Button:** Submits player's move to the server. Once all players have submitted their moves, the server returns the results to each player. Then, each player can make their next move.
J. **Shop Button:** Brings player to the shop screen
K. **Upgrade Button:** Brings player to the upgrade screen
L. **Pause Button:** Brings player to the pause menu
M. **Timer:** Shows the amount of time remaining before the player's turn is automatically sent to the server.

3.1.4 Upgrades Menu



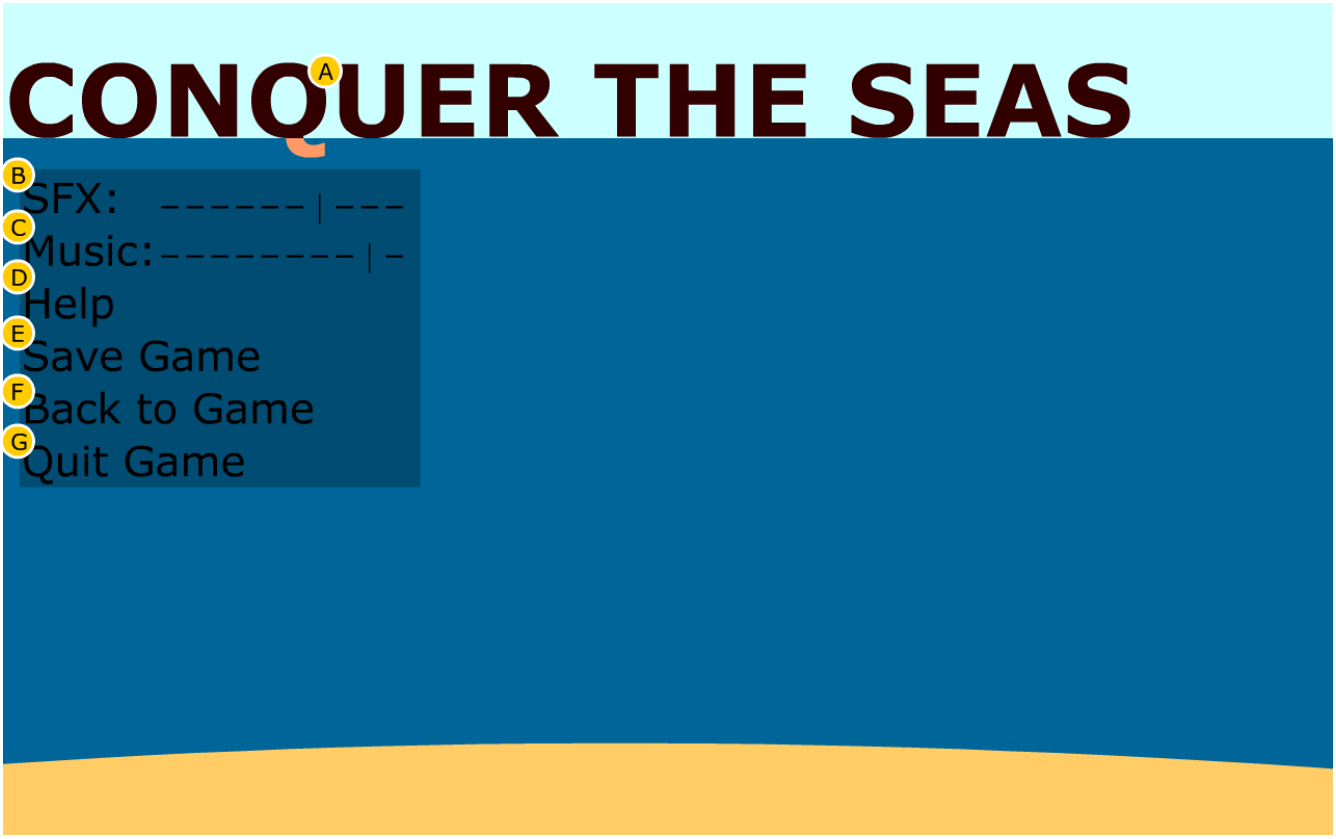
A. **Tech Tree Title:** Each of the three tech trees operate independently of one another and are unique to each of the three ships
B. **Tech:** Each of these squares represents a technology that can be upgraded. These squares will contain an unique icon for each tech upgrade. Any technology with a line extending to one above it depends on the one above it. That is, you must have the one above it to be able to purchase it. Some may have level requirements and some may not be able to combine with others. Click on a tech purchases it.
C. **Tech Title:** The title of the currently displayed technology (the displayed technology will be whichever the mouse hovers over)
D. **Cost:** The cost of the upgrade (in experience)
E. **Description:** A short explanation of what this upgrade allows the ship to do
F. **Cooldown:** The number of turns (or moves) a player must wait before using this technology again. Some technologies will grant a passive bonus which will not have a cooldown.
G. **Resource Overview:** Displays the users experience and money
H. **Shop Button:** Moves player to the shop menu
I. **Game Button:** Exits the upgrade menu and returns the game
J. **Ship Tab:** Switches between the ships to select which one to upgrade. Current ship indicated by lighter box.

3.1.5 Shop Menu



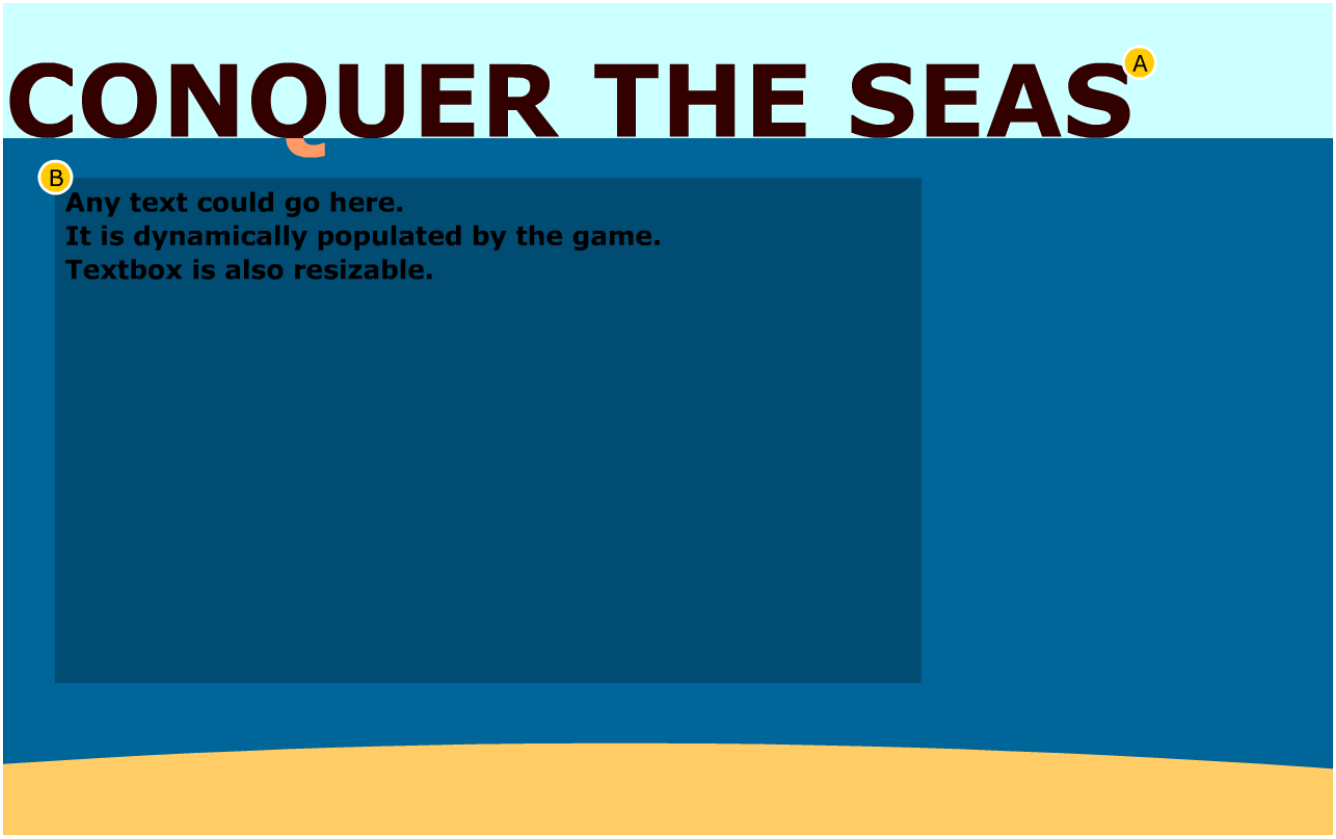
A. **Place Holder Square:** Tells user they are in shop since there will only be 15 items
B. **Unselected Item:** It should be noted that the empty squares will contain unique pictures of items the user can purchase and upgrade
C. **Selected Item:** The item currently displayed in the lower half of the screen. To change which is the selected item simply click on an unselected item.
D. **Item Name:** Name of the item
E. **Item Cost:** Cost of the item (in dollars)
F. **Next Upgrade:** Description of what purchasing the next level will of this item will alter. At the start of the game, most items will simply be deployable on their first purchase (players will not start with all items).
G. **Purchase Button:** When clicked, the item/upgrade is purchased and the player's arsenal is upgraded accordingly
H. **Item Description:** Description of how the offensive unit works
I. **Flavor text:** Comedic description of the item
J. **Selection Display:** Large icon of currently selected item or upgrade
K. **Upgrades Button:** Moves to upgrades menu
L. **Game Button:** Close the shop menu and returns the player to the game

3.1.6 In Game Paused Menu



A. **Logo:** The game's logo
B. **SFX:** Volume of game noises that are not music (1-10 scale)
C. **Music:** Volume of in-game music (1-10 scale)
D. **Help:** Opens help screen
E. **Save Game:** Pops up a system-native dialogue box allowing the player to create a save file for their game
F. **Back to Game:** Returns the player to their current game
G. **Quit Game:** Quits the current game and returns the user to the main menu

3.1.7 Text Only Screen

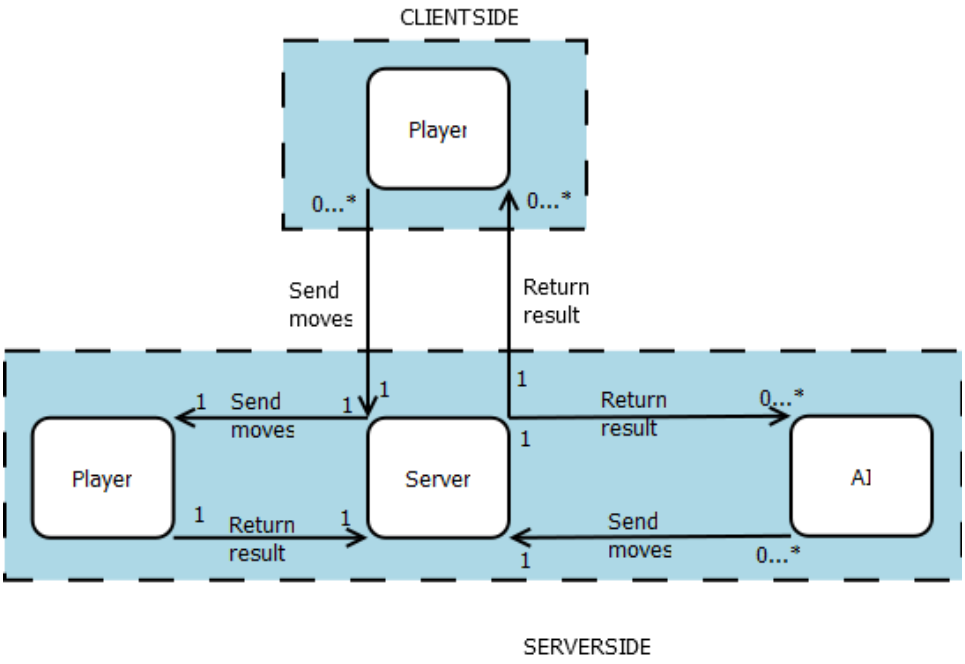


A. **Logo:** The game's logo

B. **Text Area:** This area will be populated by the relevant text. A screen like this will be used for Help and Credits.

3.2 Client-Server Concept Model

3.2.1 Client-Server Concept Model Diagram



3.2.2 Client-Server Concept Model Explanation

The Client-Server concept model describes how the clients (individual players) connect to the host player.

This project has two main components: Clients and the Server. The Server exists on one machine only. Its task is to do all game computations. Clients are the individual instances of the game - one for each player. The Client relays information to the Server, which performs calculations and sends information back to the Client. The Player makes decisions and actions, inputs these into the Client, which then sends this information to the Server. This means there are two types of interactions: Player-Client where the user is inputting commands to the game, and Client-Server, where the local game is sending or receiving information from the host Server.

Once a player chooses to host a multiplayer game his machine hosts the Server for that game. The Server will receive one turn from each of the client players (who joined the game by selecting join multiplayer game) and the local player, and calculate turns for each of the AI players. Once it receives turns from each of these sources, it determines the outcome of each move in the turn and returns the results to all the players, including the local player and AI players. Each player than sees the results of the previous turn and plans their next set of turns accordingly.

3.3 Use Cases

These use cases were authored by Matt Dannenberg on November 3, 2011.

3.3.1 Launching and Interacting with the Game

Use Case Name: Playing the Game
Iteration: Finished
Summary: How a player launches, starts, and quits the game.
Basic Course of Events:

1. The player launches the game by double clicking on the *Conquer the Seas* executable.
2. (optional) The player opens the options menu and tweaks the settings to their liking.
3. The player joins a multiplayer game.
4. The player plays the game (see 3.3.3 Taking a Turn).
5. The player wins the game.
6. The player returns to the main menu.
7. The player quits the game.

Alternative Paths:

1a. The player could launch the game from a command prompt by entering "python conquertheseas.py."

3a. The player joins a single player game.

3b. The player hosts a multiplayer game (see 3.3.2 Managing the Multiplayer Lobby).

5a. The player loses the game.

7a. The player chooses to play another game which returns him to step 3.

Exception Paths: If Python is not installed on the computer, the executable will do nothing, ideally the player will have enough sense to check the README file, realize the issue, and follow the steps to remedy it.
Extension Paths: None
Triggers: The player has decided he would like to play the game.
Assumptions: The game is installed on the computer on which the player intends to play.
Preconditions: Python 2.7 is also installed on the computer on which the player intends to play.
Postconditions: A clean exit leaving nothing new on the computer with the possible exception of a save file.

3.3.2 Managing the Multiplayer Lobby

Use Case Name: Managing the Multiplayer Lobby
Iteration: Finished
Summary: How a player controls the multiplayer game lobby; including adding/removing AI/human players and launching the game.
Basic Course of Events:

1. Multiplayer lobby screen pops up.
2. Player can increase number of players by clicking on the plus icon.
3. Player selects player type for each of the slots from the following options: AI (with difficulty level), Open (to allow another human player to connect), or Close to remove the added slot.
4. Other human players join.
5. (optional) Players communicate through the chat and discuss whether to alter the AI difficulty or number of players.
6. Other players set themselves to ready.
7. Player selects start game and the game begins.

Alternative Paths:

6a. The host player does not want one of the recently joined players to be in the game and selects Kick from the drop down menu to remove this player from the game.

7a. Player starts the game before the other players are all set to ready and is met with a warning message from which the player can decide to start anyway or cancel and wait for readiness.

7b. Player cancels the game and is returned to the main menu.

Exception Paths: The player is not connected to any sort of network and upon selecting host multiplayer game is presented with an error informing him of this.
Extension Paths: None
Triggers: Player selected host multiplayer game.
Assumptions: None
Preconditions: The player must have an internet connection.
Postconditions: Game has begun, or the player is returned to the main menu.

3.3.3 Taking a Turn

Use Case Name: Taking a Turn
Iteration: Finished
Summary: How a player plans and submits a single turn in the game.
Basic Course of Events:

1. The player performs any combination of the following actions in any order:
    - Player selects his units and distributes his limited moves among them.
    - Player selects offensive units and places them in the enemy's deployable space.
    - Player clicks “Upgrade” button bringing up the upgrades page where upgrades are viewed and/or purchased, then the player returns to the Game screen.
    - Player clicks “Shop” button bring up the shop page where items (or their upgrades) are view and/or purchased, then the player returns to the Game screen.
2. Player clicks “ACTION!” button to submit his turn to the server.
3. Player waits to receive the results of his turn and then starts at step 1 to begin his next turn.

Alternative Paths:

3a. If the game reached an end state once the results are returned, it would lead to a game termination screen. If all of the player's defensive units were destroyed or if all of the enemy players' defensive units were destroyed, there will be a Game Over or Victory screen respectively.
Exception Paths: None
Extension Paths: None
Triggers: Player is in an unfinished game.
Assumptions: The game has not ended yet and the server returned a valid result.
Preconditions: Player receives the previous turn's result from the server.
Postconditions: The server calculates the result of the turn and returns this result to all the players.