HANABI CLIENT

Requirements Document

Team H2

Amanda Zimolag (aaz713)
Brendan Nykoluk (ban803)
George Shi (hos540)
Spencer O'Lain (sto119)
Xinnan Xie (xix227)
Zeshan Ahmad (zea405)

1. Introduction

1.1 Purpose

The purpose of this document is to detail the requirements for a Hanabi client as requested by the customer.

1.2 Scope

The final product shall operate on the University of Saskatchewan Spinks Lab computers and connect to a given server on start-up where it will be able to create or join a game someone else created. The client is specified to be used by one person per machine. The client is not required to function on any other computers, although it may due to system similarities.

1.3 Overview

Hanabi is a card game designed to be played by two to five players all in the same room. In the modern world, it can be difficult to keep people together for an extended period that is required for a game of Hanabi to be played. Our team aims to solve this problem. Many people have access to a computer as a part of their daily lives. By having Hanabi be playable on a computer over a network, the need to have all the players be in the same room is eliminated.

This project is made possible by the abilities of the team working on it. Art direction will be led by Amanda because she has experience as an artist. Brendan will lead writing and document compilation due to his experience in technical writing from his engineering background. The automated intelligence team will be led by Spencer and Zeshan since they have taken the AI class offered at the University of Saskatchewan. Spencer has the most experience with Git so he will curate our repository and handle any conflicts that arise within. The group have all taken classes on Java with Brendan, Jim, and Amanda currently taking CMPT 280. These qualifications give us a strong basis to be able to create the project required.

1.4 Definitions

AI – A computer-controlled player (Automated Intelligence)

User – The human player interacting with the client

Client – The program the user uses to connect with the server and play the gam

Shall – Denotes a hard requirement. If the feature denoted by shall is not in the program, it shall not be considered complete.

Should – Denotes a soft requirement. If a feature denoted by should is not in the program, the program can still be complete but may be lacking something important

May – Denotes something that would be beneficial to the project but will not hinder the completeness with its absence.

2. Hanabi

2.1 Rules of Hanabi

2.1.1 Summary

Hanabi is a cooperative game; players work together to achieve a common goal. In this case, players need to work together by playing the cards of the same suit in a series order to set up 5 fireworks (blue, green, red, white, yellow). Every misstep shortens the fuse. When the fuse runs out, the game is over.

2.1.2 Starting a game

At the beginning of the game, here will be 50 cards, 8 information tokens and 3 fuse tokens. Cards are in different suits (colors). Each suit has three 1's, two 2's, two 3's, two 4's and one 5. At the start of the game, cards are dealt to each player. Each player is dealt five cards if there are two or three players or four cards if there are four or five players. The players cannot look at their own cards. They can only see the cards of the other players. In general, players cannot talk to each other, but the players will have a chance to give each other information.

2.1.3 Building a firework

Each suit represents a firework of its corresponding color. The players must play cards of the same suit in counting order from one to five.

2.1.4 Actions on your turn

There are 3 different actions a player can perform on their turn.

The first action is to give a piece of information to another player. Whenever a player gives a piece of information, 1 information token is used. Only one piece of information can be shared per turn and the information must take one of the following forms:

- Specific suit in that players hand
    - "These cards in your hand are red."
- Specific rank in that players hand
    - "These cards in your hand are 2's"

The information given must be complete. A player cannot only point to one of the suit or rank if the other have 2 or more cards that have the same color or number.

The second action a player can perform is to discard a card from their hand. Whenever the player discards a card, it returns one information token to the players, and the discarded card is removed from play. After the player discard a card, they will draw a new card from the deck unless the deck is empty. A player cannot discard a card if all eight information tokens are available.

The third action is to play a card from your hand. When a player plays a card, they will draw one card from the deck unless there are no cards remaining. If the card is a "1" in a suit that does not yet have a firework started, or the next rank in a firework that has been started, the card is placed into the play area starting a new pile or adding to the corresponding one respectively. If the card completes a pile (i.e. is a 5 being played on a 4), one information token is returned to the

players. If the card does not start a new pile or continue an existing one, the card is discarded, and one fuse token is removed.

2.1.5 Ending the game

The game ends in one of three ways. The first way to end this game occurs when the third fuse token is removed from play. The second way is by completing all five stacks of fireworks by building them to level five. The final way to end this game occurs when a player draws the last card from the deck. Each player then gets one more turn before the game ends.

2.1.6 Scoring

The score of the game is the total number of cards successfully played. The following descriptions describe the players performance based on their score.

- 0-5: horrible,
- 6-10: mediocre,
- 11-15: honourable but forgotten,
- 16-20: excellent,
- 21-24: amazing,
- 25: legendary.

3. Requirements

3.1 User Actions

3.1.1 Pre-game Actions

There are seven actions a user can take when the program first opens. The actions are

- create a game,
- join a game,
- quit,
- view a tutorial,
- view developer information,
- view help information, and
- change the default turn length.

These actions are described in figure 3.1 and elaborated on in figure 3.2 below.

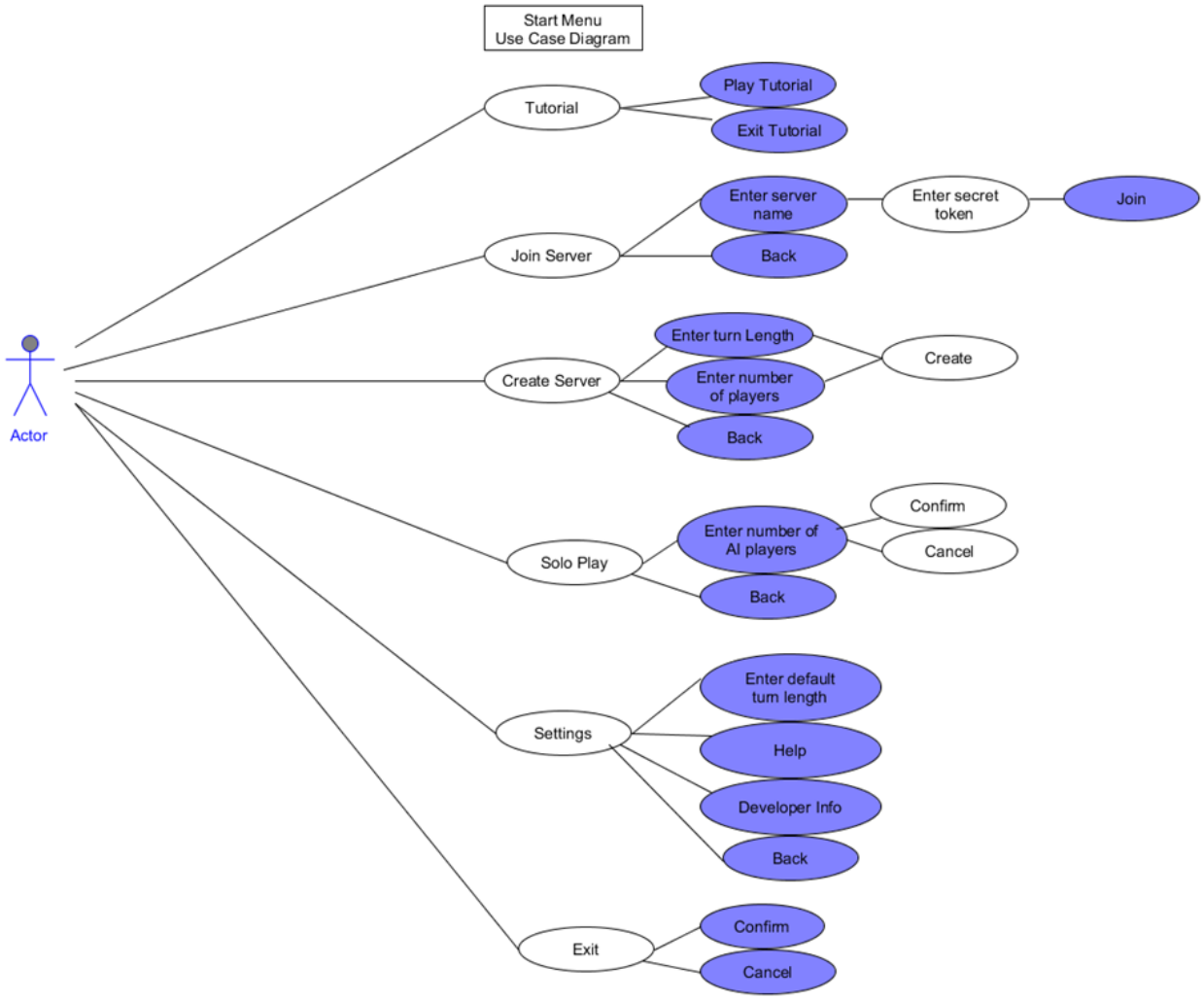

Figure 3.1 – Use Case Diagram for Pre-game actions

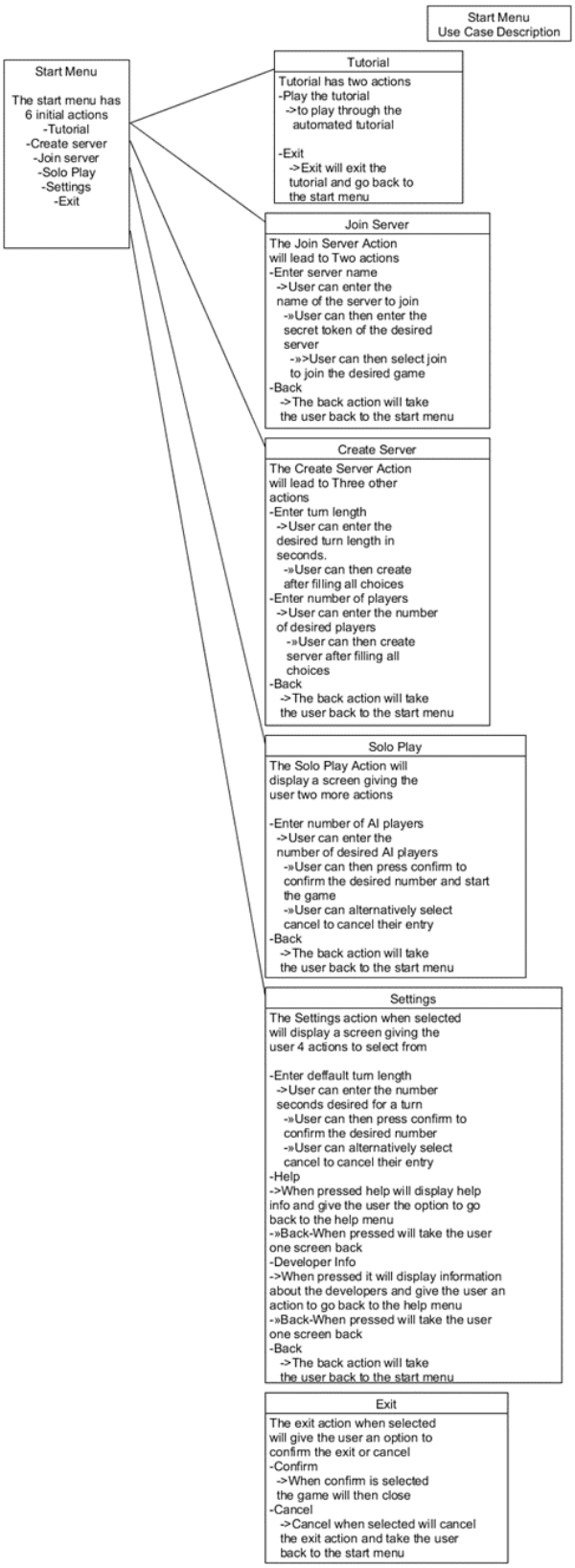

Figure 3.2 – Use case text description for pre-game actions

The first two options are how the user gets into a game using our client. The create a game action shall allow the user to start a game for other players to join. The player shall be able to set the length of a turn and the number of players before the game is created. At that point, other players shall be able to connect to the game. Once the set number of players have joined, the player shall be taken to the game interface.

The other way a player can start playing a game is by joining a session that was created by another client. The user shall enter another game by providing the system with the game id and secret token of the desired game.

The user shall be able to quit the client at any time. This will be a way to close the program when the user is finished with it.

The other actions that a player can take before the game are not integral to the function of the client but are informational for the user. The first such action is to view a tutorial of the game. The user shall be able to view a tutorial that explains the basics of Hanabi and how to play it in the client. This benefits a user that is new to the game or one that is unfamiliar to the client by showing them around before sending them straight into a game.

The user shall be able to view information about the developers and the game before they enter a game. This will allow the developers to gain recognition for their work and allow the user to read about the game.

Finally, the user should be able to change the default turn length before they start creating a game. This has potential to save a user time if they create a lot of games and wish for a nonstandard turn length.

These actions shall be available from an on-screen menu like the one displayed below (Fig 3.1). The latter three options (viewing developer information, viewing the help document, and changing the default turn length) will be contained within a settings submenu. Once the user has finished with the actions available before a game, they can enter a game using either the create a game or join game actions. Once the party fills up, they will be taken to the gameplay interface.

Figure 3.3 – A mock up of the menu that the user will see when they start the program.

3.1.2 Pre-game Interface

The following section will go through our games process in creating a game in the Hanabi client. It is our goal to ensure a smooth experience for the customer when they create a game. Because a large part of the games experience relies on online play between players, the user will spend a large amount of time in this menu. It should be easy to use.

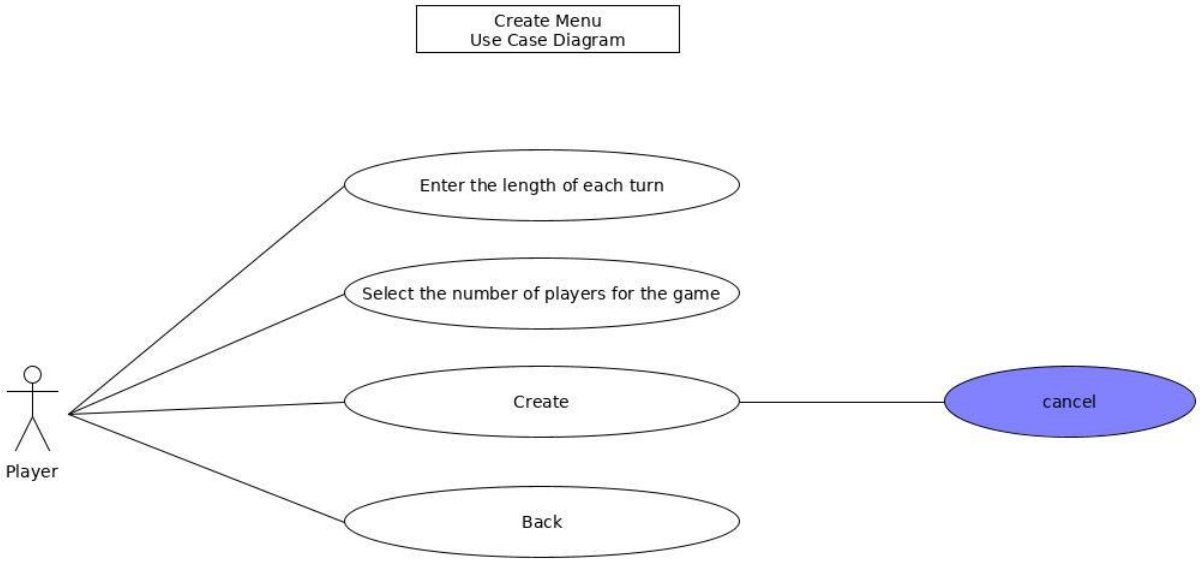

Figure 3.4 - The use case diagram above describes the actions that can be taken by the player in the create game menu.

As we can see in Figures 3.4 and 3.5 there are a variety of actions that can be taken by the host. The host may:

- Decide on the duration of the turn timer.
- Enter the number of players in the game.
- Create the game.
- Go back to the previous menu.

We will discuss these in detail below.

For the first point above. We would like to give the customer a choice in how long they would like their turns to take. This could take the form of a Field in which the host could enter the time in seconds or set of pre-set times to be selected from.

Another key aspect of a game is the number of players that will playing. This will be important as the rules change depending on how many players there are. With this importance in mind we have decided to have an action that will allow the host to specify the number of players in

the game. Just like selecting the time, this can be done through the user entering values into a field or selecting from a pre-set list of choices.

After adding all the information regarding the game to be created, it is time to create the game. This will be an action that the host can carry out in the menu. As we can see in Figure 3.3, the client will make sure that there are valid inputs in the fields required to create the server. We will also have a secondary action which allows the host to cancel the server that had been made and return them back to the create menu.

If the user changes his mind about creating a server, it would be a good choice to allow the user to go back out of the current menu. This will allow a smooth way of transitioning between menus and provide a user-friendly interface. It will be a simple action which will take the user back to the start menu.

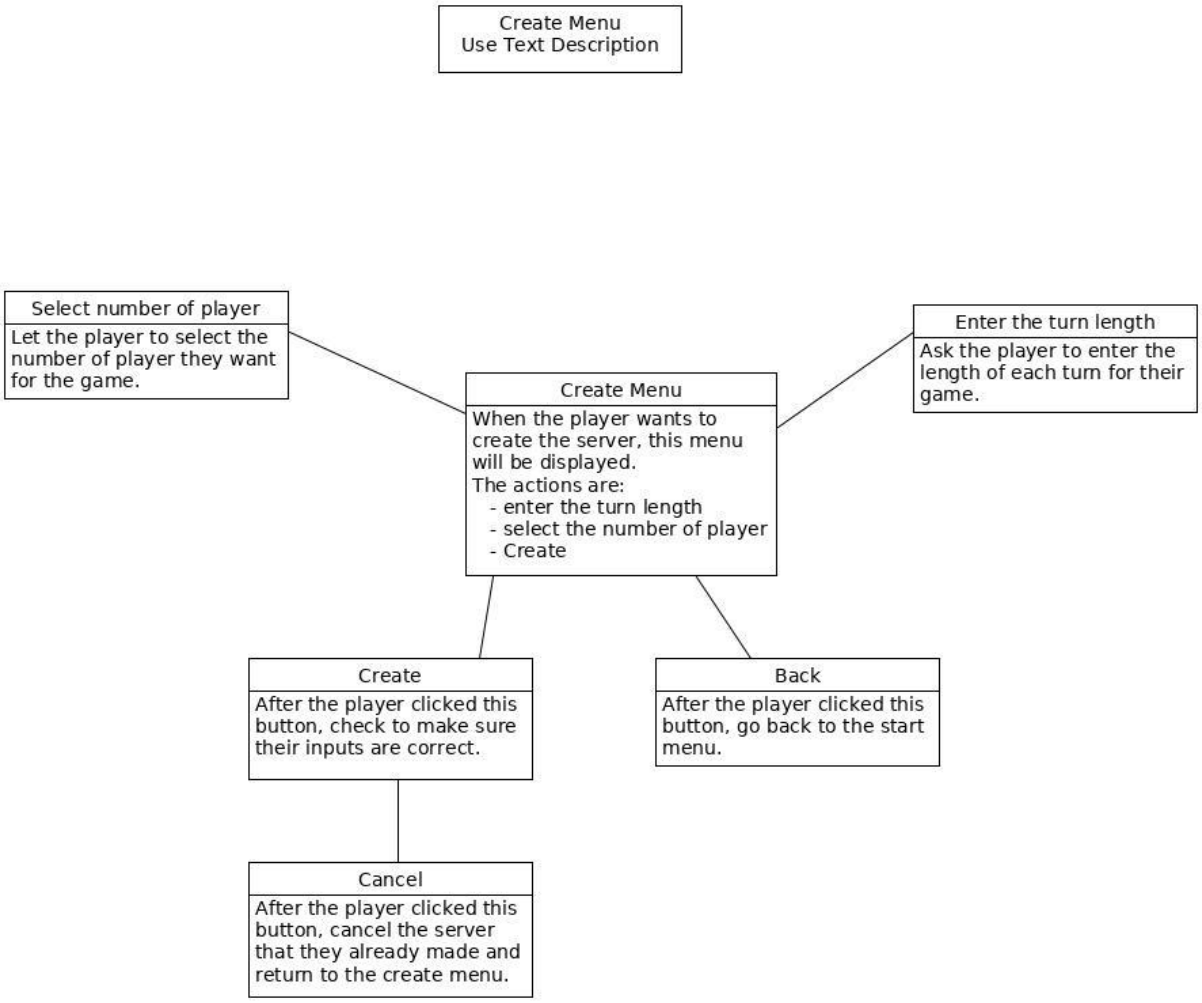

Figure 3.5 - This is the text description of the use case diagram (Figure 3.4).

A diagram has been included below which will help you visualize the menu which the player will navigate within the client.

After creating a server, it will need to be populated by players to play the game. This will be done through our join menu. Our goal here is the consistent Implementation of our UI throughout the menus in the game client. This will ensure smooth for the player and allow for a more present experience.

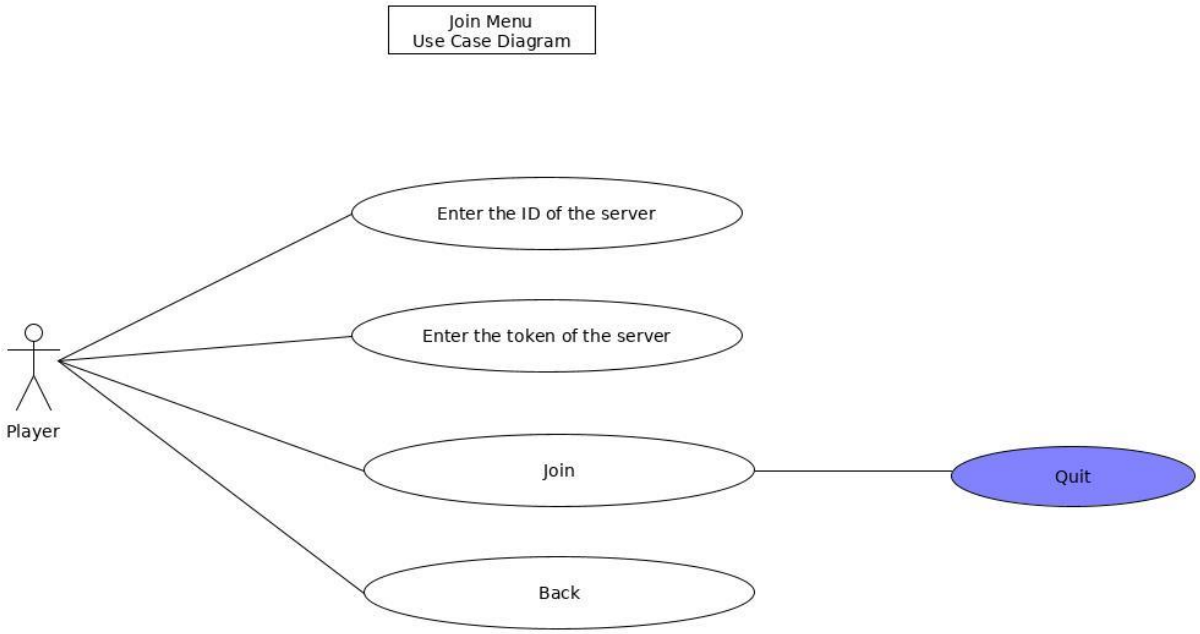

Figure 3.6 - A use case diagram for the join game menu.

Figure 3.6 shows us that the actions that can be taken by the user in this menu. These actions include:

- Enter the ID of the server.
- Enter the token of the server.
- Join the server with the secondary action of quitting the server.
- Backing out of the current menu which takes the player to the main menu.

These actions are also described in Figure 3.7 which is shown below.

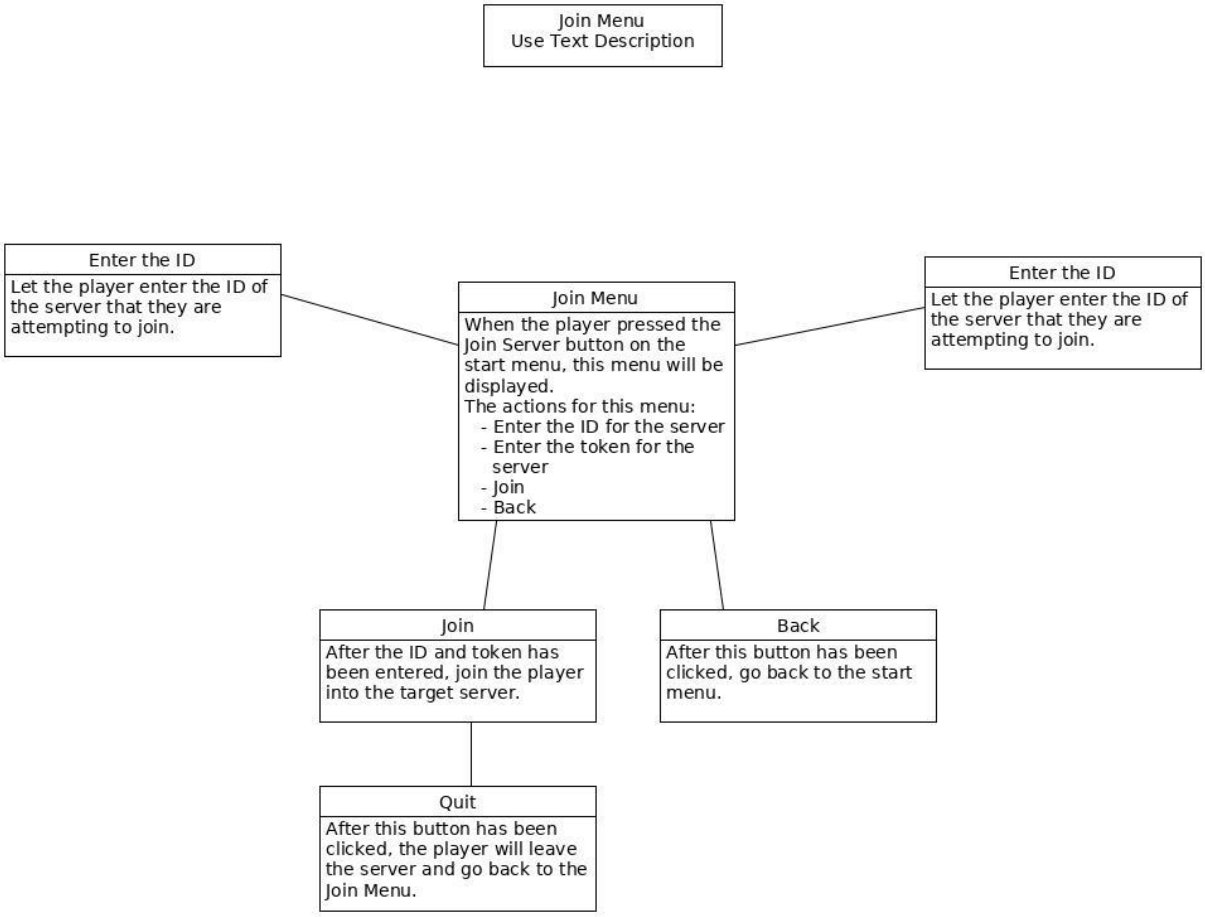

Figure 3.7 - This is the text description of our use case diagram (Figure 3.6).

As shown in the Figure 3.7 above, this menu is found by choosing the Join option in the main menu.

3.1.3 In Game Actions

There are five actions that the user shall be able to perform while in a game. The first three of the actions are integral to the game of Hanabi itself. The other two actions are possible to make the user experience better. These actions are shown in figure 3.8 and detailed in figure 3.9. These actions are

- play a card,
- discard a card,
- give a hint to another player,
- quit the game, and
- take a break by substituting in an AI player.

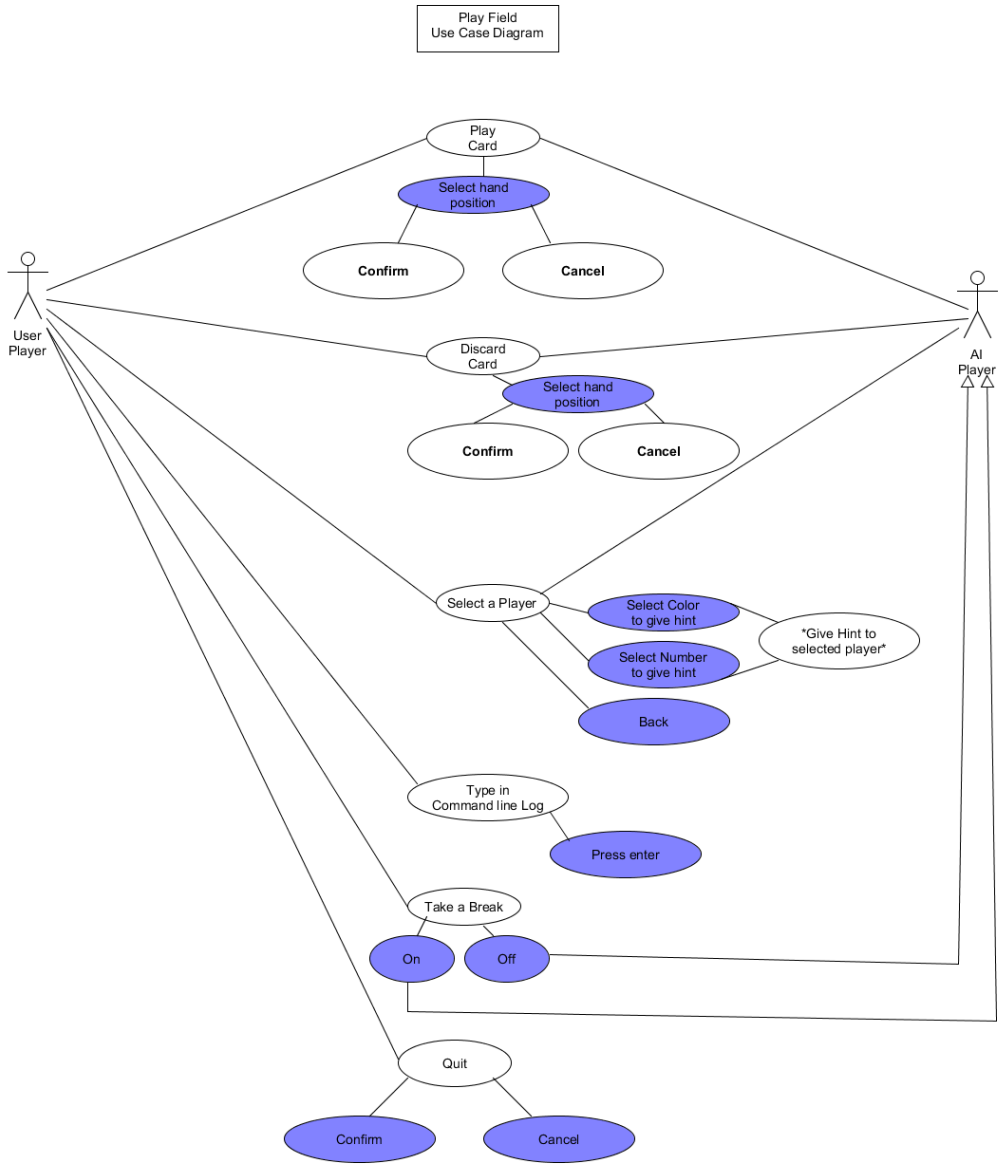

Figure 3.8 – Use case diagram for gameplay

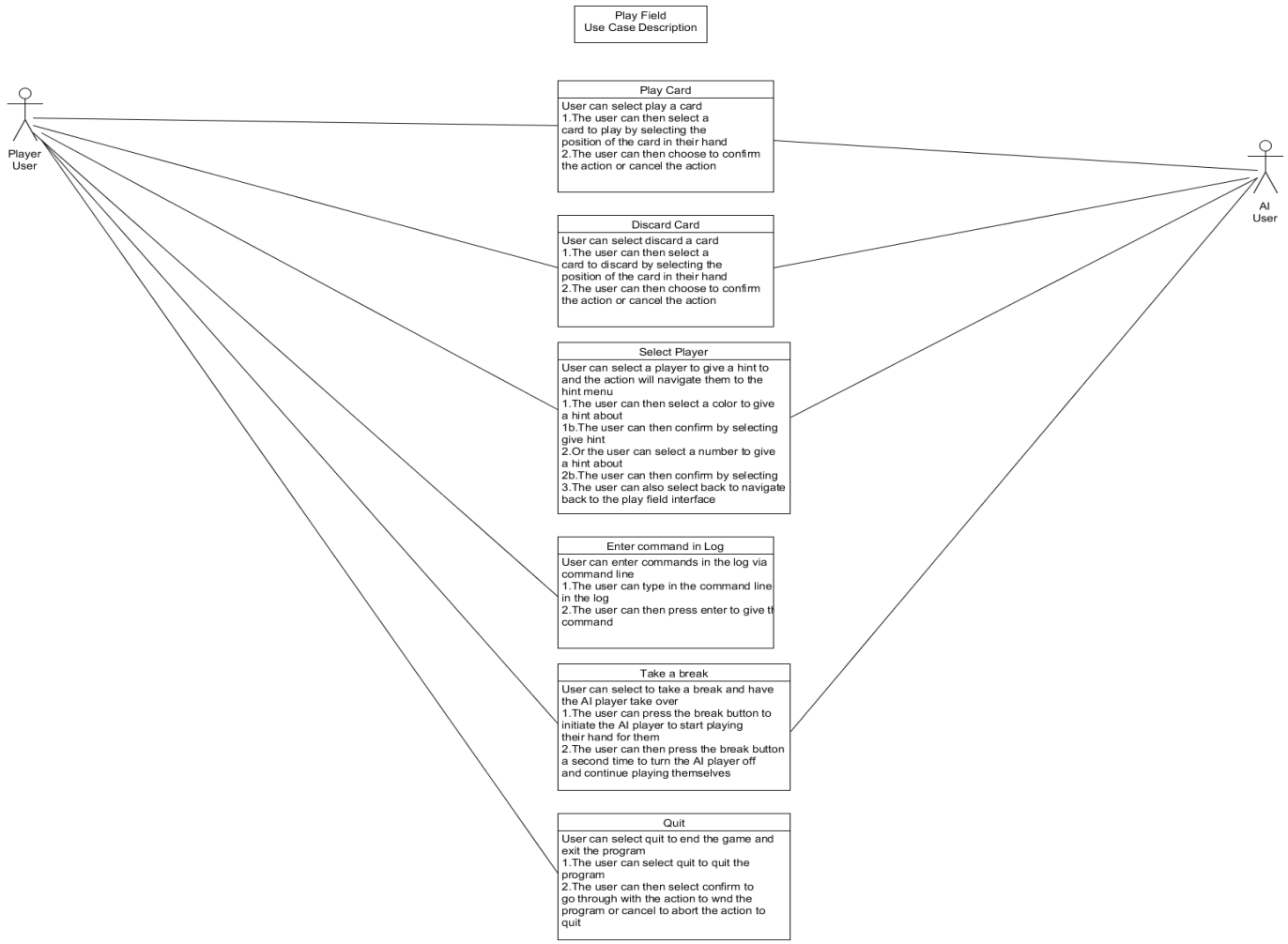

Figure 3.9 – Use case text description for in game actions

Play a card is an action that shall only occur on the user's turn. The action consists of taking one of the cards from the user's hand and either adding it to the corresponding firework stack in the play area or discarding it depending on whether it meets the requirements from the rules of Hanabi (See section 2.1).

The next action the user can take is to discard a card. This action shall only occur on the user's turn. This action is when a player removes a card from their hand and places it directly into the discard pile where it is no longer available to play. This action shall return one information token to the players, and this action shall be unavailable when all eight information tokens are still available.

The third action available to the user is to give a hint to another player. This action is more complicated than the others and is described by Figures 3.10 and 3.11. This shall also only be available on the user's turn. This action lets the user select one of the other players, and then select either a suit or rank. This gives the selected player knowledge of all the cards in their hand that are of that suit or rank. We have created a preliminary mock up of the interface that will be used to send hints (Figure 3.12). The hint function shall only allow valid hints to be selected by the user by making invalid option not clickable.

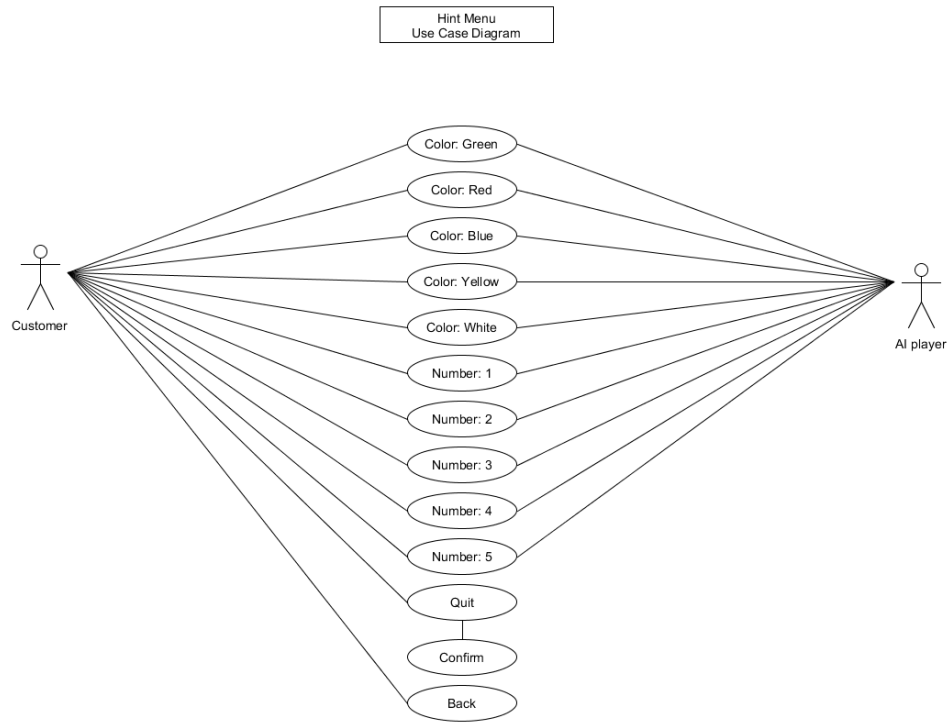

Figure 3.10 – Use case diagram for the Hint Menu

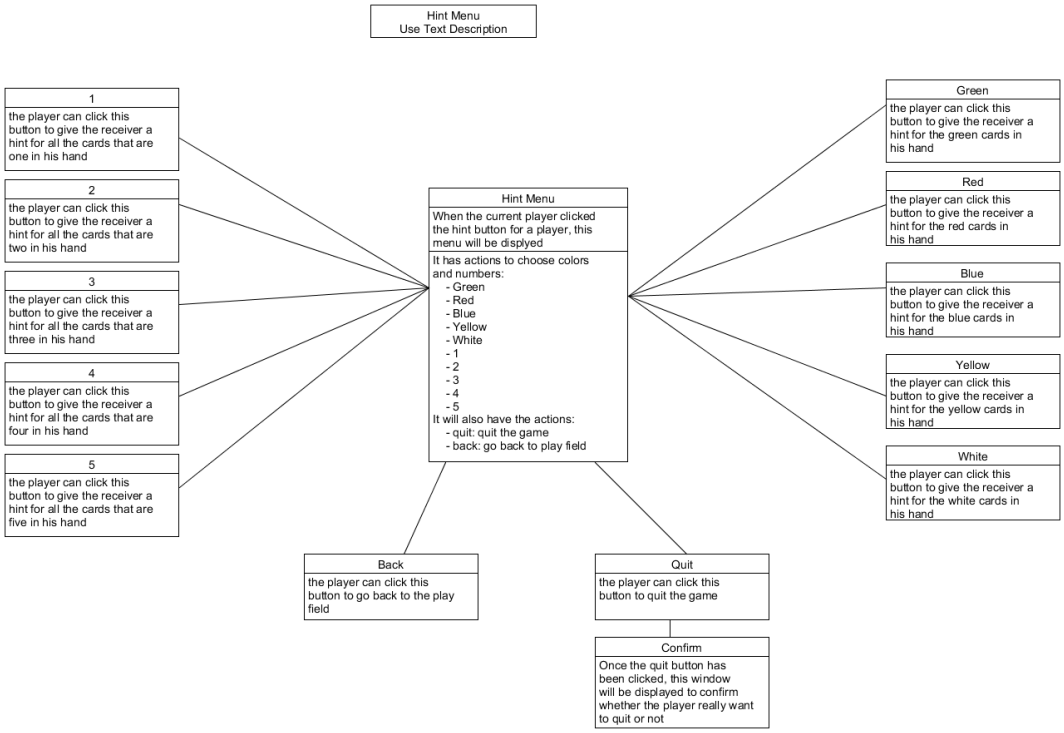

Figure 3.11 – Use case text for the Hint Menu

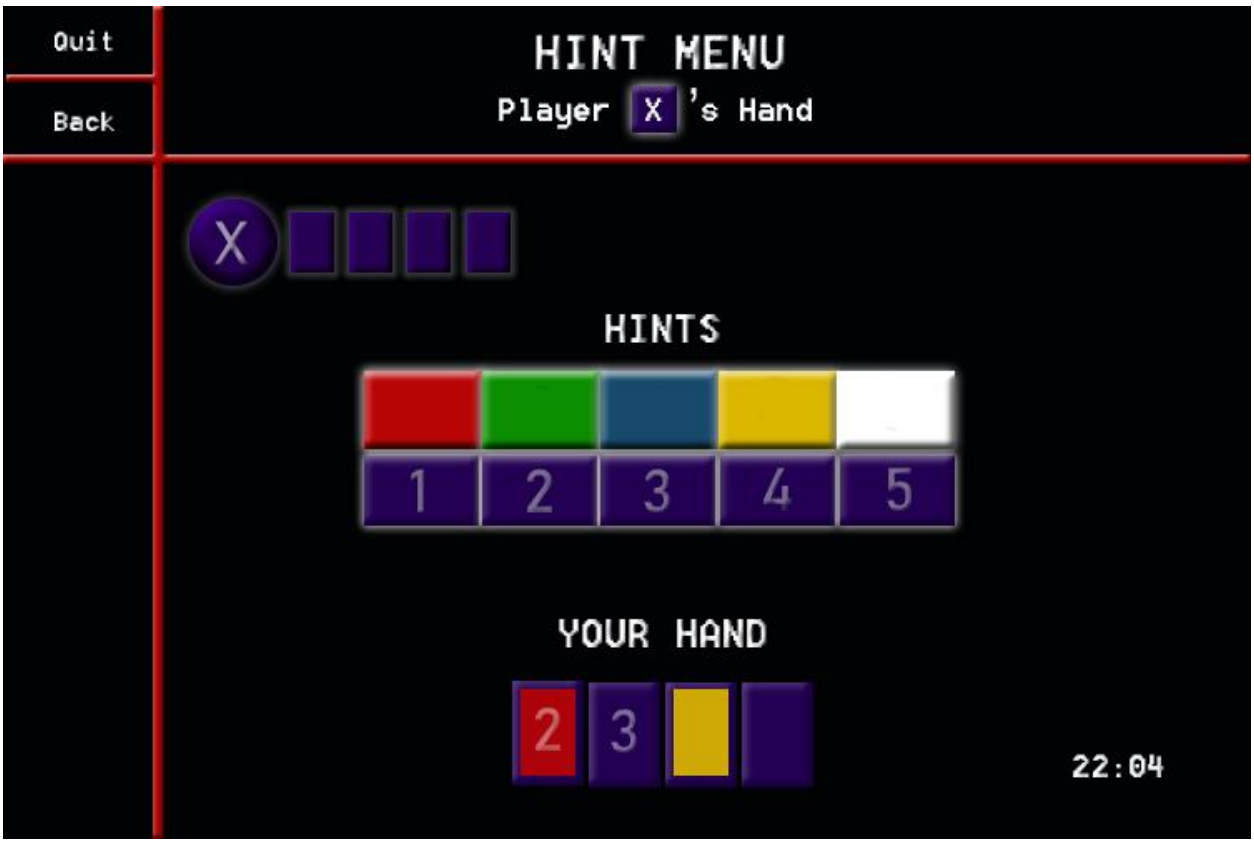

Figure 3.12 – Sample interface for giving hints to other players

The previous three actions are integral because they mimic the actions that can be performed by a player of Hanabi on their turn. The user must be able to make any of these actions on their turn for a game of Hanabi to be played properly. There are two other actions that the user shall be able to make at any time.

The first of these is to be able to quit the game. The user shall be able to end the game at any time by quitting from it. This will allow the user to join or create a new game without having to wait for the current game to end.

The other action is to take a break from the game. The user shall be able to take a break at any time by substituting in an AI player to make moves for them. The AI shall take over at the user's command and the user can return to making his/her own moves by issuing another command. This feature will allow the user to leave the game for an extended period without ruining it for the other players. The next thing the user requires is information from the game to make intelligent choices.

3.1.4 In Game Interface

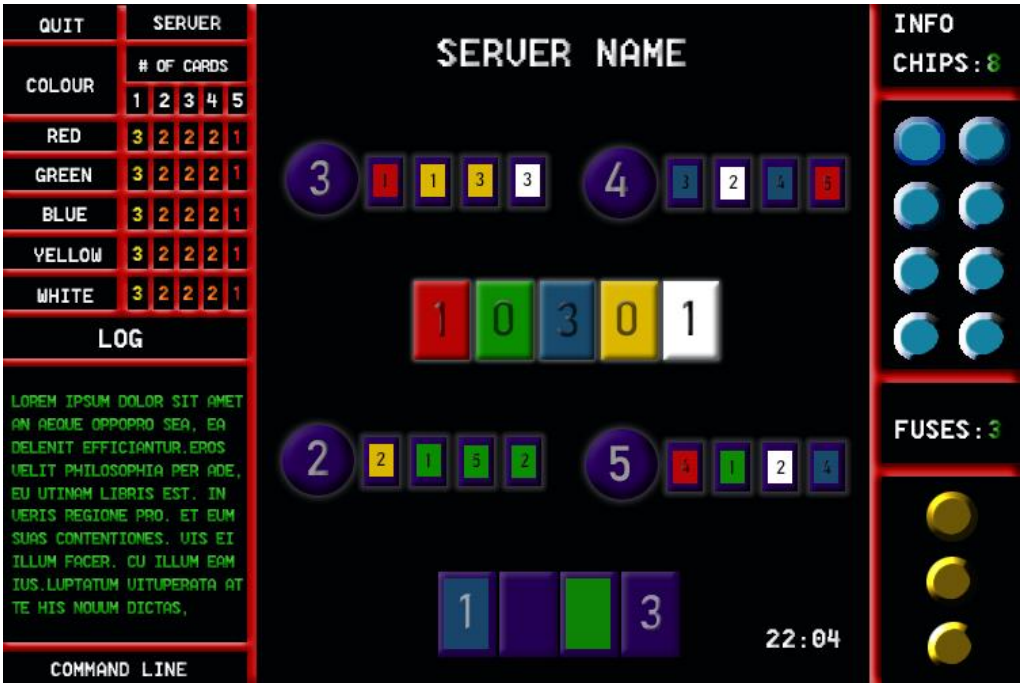

Figure 3.13 – Sample interface for in the game

The in-game interface is how the user will receive information about the state of the game in order to make an informed choice about what to do next. A sample of the user interface is pictured below (Fig 3.13). The interface shall always display all key pieces of information for a game of Hanabi including

- the number of information tokens available (top right of the screen),
- the number of fuse tokens remaining (bottom right),
- known information about the user's hand (bottom center of the screen),
- the hands of the other players (around the center),
- the stacks of cards that have been successfully played (center of screen), and
- a list of all cards that remain in play (top left corner).

The interface shall also display a few key pieces of information in the appropriate scenarios. These scenarios are

- when another player leaves the game,
- when another player stops responding,
- when the game ends,
- when the user makes an invalid move, and
- when the user is given a hint by another player.

Each of these scenarios shall display a message to the user in the log area explaining what happened by describing one of the above scenarios. In the first three scenarios, the end game fireworks display will also be set off (See section 3.3). These pieces of information inform the user about the state of the game, allowing them to act accordingly.

3.2 Automated Intelligence

The goal with this project is for the client to have a smooth and fun experience with the game of Hanabi. A major part of that experience is immersion. To immerse a customer in our product, we will implement an AI (Artificial Intelligence) player into the client. Our intention is for a human player to be able to enjoy (him/her) self even if actual human players are not present in the game. This will increase the scenarios in which the product will useful to the customer.

We have described our AI as an actor rather than a portion of the system. We concluded that it would make more sense for this to be the case as the AI will be making its own decisions and acting on what it decides. These actions are also a mirror image of a human players actions. Due to these reasons we have decided to put our AI in the category of actor.

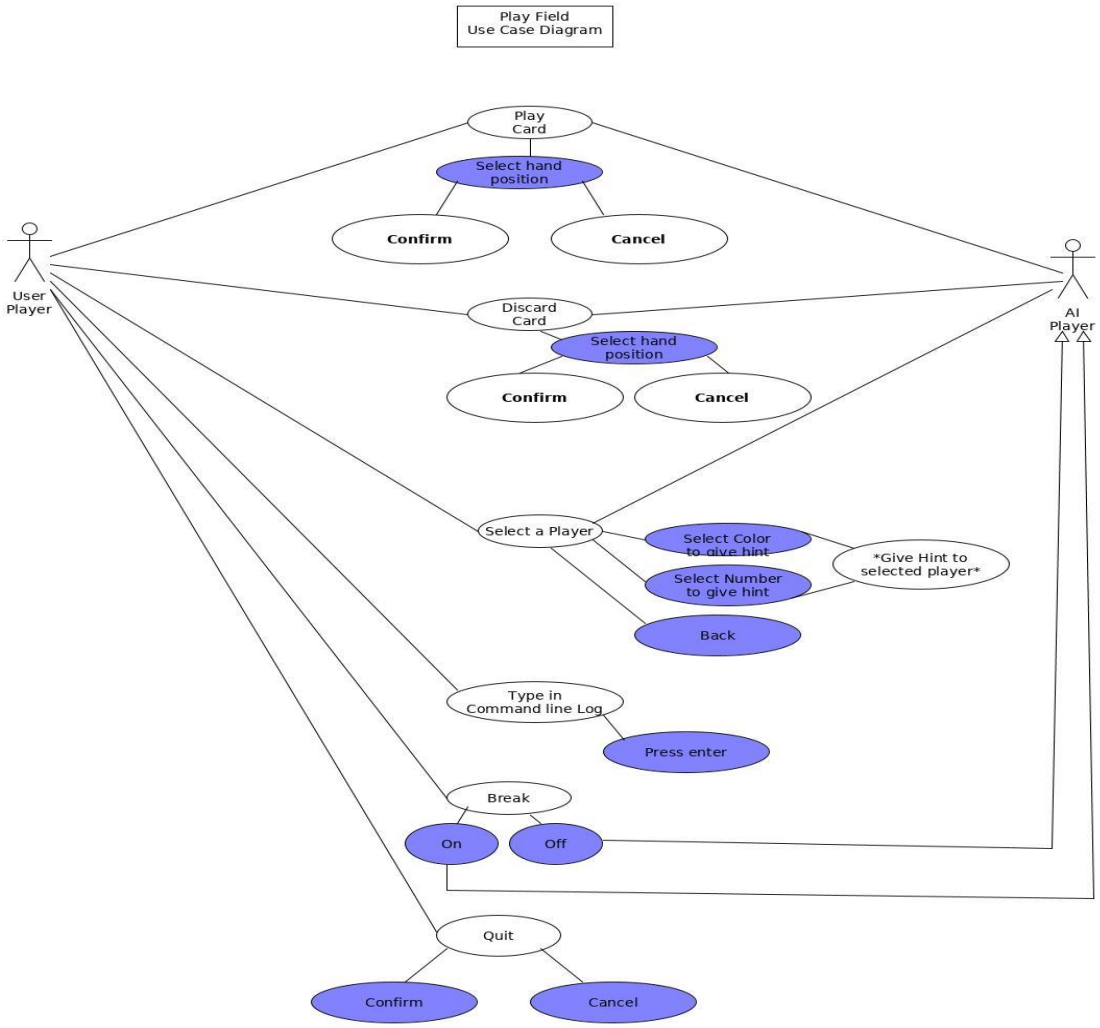

Figure 3.14 - The figure above describes the use cases of a Human player and the AI player.

The AI player must behave as if it were a human player. As we can see in the diagram above, there are some basic behaviours which shall be implemented for the customer to feel that this is the case:

- Hints must be given by the AI player to other players of the game.
- The AI player shall play by the rules and only take legal actions within the game which includes discarding and playing cards.
- AI players have no ability to leave a game or end a game by invalid actions.

These actions will provide a basic level in which an AI player will play the game. We can go through these in detail.

Point 1, hints must be given by the AI player to other players of the game. This is quite important for a successful as the game revolves around giving hints. At the most basic level the AI shall give hints to all the other players. These hints should be unique so that there is not an endless loop of the AI player giving the same hints repeatedly. After we implement this as our baseline and will start to look at methods which will increase intelligence of our AI player with regards to the hints it gives.

Second point, the AI player shall play by the rules and only take legal actions within the game which includes discarding and playing cards. For an AI player to feel like a human player it must play by the rules. The AI player cannot play any invalid actions. If time permits, we would like to enhance the AI's ability to play cards intelligently and discard cards which have a high probability of being safe to discard.

Finally, AI players have no ability to leave a game or end a game by invalid actions. As a minimum requirement, we should not have an AI which can leave half way through a game. It would ruin the players experience. The most basic way an AI would leave the game would be to quit it by itself and this can be easy to restrict. Another case we should consider is AI going past the inactivity timer. The AI should take time to think but it should always have a result by the time the turn ends.

3.3 End Game

Before the end game graphics come, the game needs to end. There are six way to end this game.

1. When all fuse tokens are removed,
2. When the players build all 5 fireworks to level 5,
3. When a player draws the last card from the deck,
4. When a player does not perform an action for an extended period,
5. When a player is disconnected, or
6. When players make 3 invalid moves.

The first three ways are the normal ways to end a game of Hanabi. When an invalid card is played, one of the fuse tokens is removed. The game ends when all three are removed. The game also ends if all five fireworks are built up to rank five. The third way to end the game is to run the

deck out of cards, which means players are running out of time to building the firework. Each player will have one more turn and then the game will end.

The last three are the unusual way to end this game. If any player does not make a move for an extended period, which is given by the timeout set by the creator of a game, the game will end automatically. If a player is disconnected, the game also ends because there is not enough player for the game. Invalid moves are the last way for the game to end. The game will end when a player makes three invalid moves. There are two different invalid move types. The first one is one player discards a card when all of the information tokens are still in play. The second is give an information (suit or rank) to another player that does not have any cards of that suit or rank/

When the game ends, it will play 5 different color fireworks all together, one for each suit. Depending on the firework level that the players build on each color, the firework will have a different appearance. These are the levels:

- Level 1: Roman Candle,
- Level 2: Sparkler,
- Level 3: Kamuro,
- Level 4: Ring, and
- Level 5: Palm Shell.

The first level is Roman Candle which only shoots one or more balls of color into the playfield (Figure 3.15).

Figure 3.15 – Roman Candle Firework


Figure 3.16 – An example of a sparkler type fireworks

The second level is a sparkler. Sparklers shoot out a large quantity of small sparkling stars all at once into the playfield. The picture above (Figure 3.16) is an example for this firework.

When a firework is at the third level, it will shoot a firework in to the playfield and then emit a dense burst of glittering silver or gold or other colored stars which leave a heavy glitter trail and shine bright in the screen. The picture below (Figure 3.17) is an example of a kamuro.


Figure 3.17 – An example of a Kamuro firework

The fourth level is a ring. At this level, the ring can emulate a dragon type firework flying in a circle like a colored orb emitting a tail of bright stars that travels in a perfect circle or a display of several stars that explode outward in a perfect ring. The picture below (Figure 3.18) is an example for this firework.

Figure 3.18 – The top firework is an example of a ring firework.

The fifth level is a palm shell. This firework features a thick rising tail that displays as the shell ascends. The picture below (Figure 3.19) is an example for this firework.

Figure 3.19 – An example of a palm shell firework

The fireworks display may contain sound if time allows. After the firework display, the interface will show players their score for the game. The score levels are described in the rules of Hanabi (Section 2.1) After the fireworks are done and the score is shown, players will automatically be taken back to the start interface.

3.4 Future Considerations

There are a couple of features that would benefit the user, but due to time constraints will not make it into the initial release. These features are

- Solo Play,
- Log improvement,
- AI improvement.

Solo Play is the first such feature we would like to implement. There will be a “SOLO PLAY” button at the start menu which let the user can build a room for himself. In this mode, all the player except the user is AI. The system will ask the user how many bots join this game, how long for the turn length. Then the game will be start as a normal game except you are playing with AIs. The main purpose for this function is provide an easy way for playing alone.

The current log only shows the commands the user did. In the future, we plan to improve the log such that it will prints the other player’s moves as well. Whenever a player made a move, it will record it and print it into the log. This function would help for user to organized what they had done.

For the AI, its quality depends on how well it plays the game, i.e. its “intelligence”. After we implement the basic requirements, we will look toward improving our AI even further. One of the ways we would do this would be to implement methods such as Monte Carlo simulations or a neural network. These would increase the ability of our AI and allow it to “think” faster when it is its turn. Another goal of ours would be to have an auto play mode where if one player needed to take a break, they could toggle the AI to play for them until he/she came back. These features are not required for the basic functionality of our game but would be more of a “quality of life” improvement which would enhance the experience.