
SOFTWARE REQUIREMENTS SPECIFICATION

for

Game Client for Hanabi Card
Game

Version 1.0

Prepared by Alexander Lavis, Caim Chen,
Justin Pointer, Noah Kovacs, Sajith Rajapaksa

Group E3, CMPT 370, Department of Computer
Science - University of Saskatchewan

February 4, 2019

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Project Scope	3
1.3	The Game	3
1.4	The Team	4
1.5	Document Conventions	4
1.6	Glossary	4
1.7	Overview of Remaining Chapters	5
2	Overall Description	6
2.1	Product Perspective	6
2.2	Product Functions	7
2.3	Functional Requirements	7
2.3.1	Create Game	8
2.3.2	Join Game	9
2.3.3	Play Card	10
2.3.4	Discard a Card	12
2.3.5	Hint	13
2.3.6	Swap	15
2.3.7	Quit	16
2.4	Operating Environment	17
2.5	User Documentation	17
3	External Interface Requirements	18
3.1	User Interfaces	18
3.1.1	Start Menu	18
3.1.2	Create Game Menu	19
3.1.3	Join Game Menu	20
3.1.4	Game Board	21
4	System Features	23
4.1	Simple Graphical User Interface	23
4.2	Simple Artificial Intelligence	23
4.3	Advance Artificial Intelligence	24
4.4	Audio elements	25
4.5	Advanced Graphical User Interface	25
5	References	26

1 Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of a program that plays the card game, "Hanabi". This document illustrates the program's features, what it is capable of, how it reacts to user interaction, and its operating constraints. Furthermore, this document outlines more ambitious concepts that may be developed later in the software's life-cycle. It is meant to be presented to both the client (for their approval) and to the development team, as a reference in developing the program.

1.2 Project Scope

This document also gives an overview of what the project will be about. The product/-software will be a game developed for the University of Saskatchewan. It is designed to replicate the experience of the physical card game, "Hanabi," but on a digital platform. By letting users interact with a digital version of this game, the system will allow human players to play alongside computer players.

The program will present the user with an intuitive interface, that they will use to play the game. It will automatically perform essential functions of the game, such as drawing or shuffling cards. It will also work by communicating with a remote server.

1.3 The Game

It is worth providing a brief overview of what the game called "Hanabi" is. Hanabi is a cooperative card game in which players are aware of other players' cards but not of their own. They must attempt to play a series of cards in sequential order to create the largest fireworks show possible. The players must accomplish this by:

- giving other players information on the cards in their hand,
- discarding cards they deem unnecessary to the fireworks display, and
- playing cards to grow the display's size.

This must be done while also managing limited pools of information tokens and fuses.

1.4 The Team

With this project introduction, there is also a need to introduce the skills of those who will be working on the project going forward. The development team is made up of five member currently enrolled in Computer Science program at the University of Saskatchewan. The group possesses relevant course experience in artificial intelligence, computer networking, game design and development, Java programming, group and pair programming, English writing, and engineering.

1.5 Document Conventions

Furthermore, this team has followed a convention for writing this document. It is written using the ISO/IEC/IEEE International Standards for Requirements Engineering.

1.6 Glossary

As part of the IEEE convention, this document employs a set of terms whose meanings need standardization. The following table contains commonly used words and phrases in the document that is specific to this project:

User: The human interacting with the program (is a Player).

Player: The name of a specific actor (human or computer controlled, can be the user).

System: The program on the local machine.

Server: What the client is bouncing information off of.

Card Stack: This refers to the available cards that can be distributed to a player.

Discard Pile: All of the used cards.

Hint Tokens: A hint token gives a player the ability to use the hint action. there are total of 8 tokens and each hint takes one out.

Fuse Tokens: This token decreases each time a invalid card is played. There are 3 token in total. Once all tokens are spent the game ends.

Fire works stack: Stacks that are created at the center of the game. These stacks of cards are sorted by color and numbers. Players needs to place them from their card hand.

NSID : Network Services ID of the University Of Saskatchewan. An unique ID given to each student.

1.7 Overview of Remaining Chapters

This concludes the introduction. Following that, the document will first give a general overview of the products functionality, as well as what users can do to engage in the game. Then, it will introduce the operating environment (or the hardware limitations), followed by details of our proposed user interfaces. Finally, we conclude by indicating the likelihood of adding these system features. All this being said, we begin with what the user is able to do, and the what our product is.

2 Overall Description

2.1 Product Perspective

The product is designed to be a standalone replacement of the original card game. It is a part of no existing system and simply meant for the use of a client. Note that product development is in the context of Software Engineering course (CMPT 370), offered by the University of Saskatchewan. The product is also developed to have many uses and functionality.

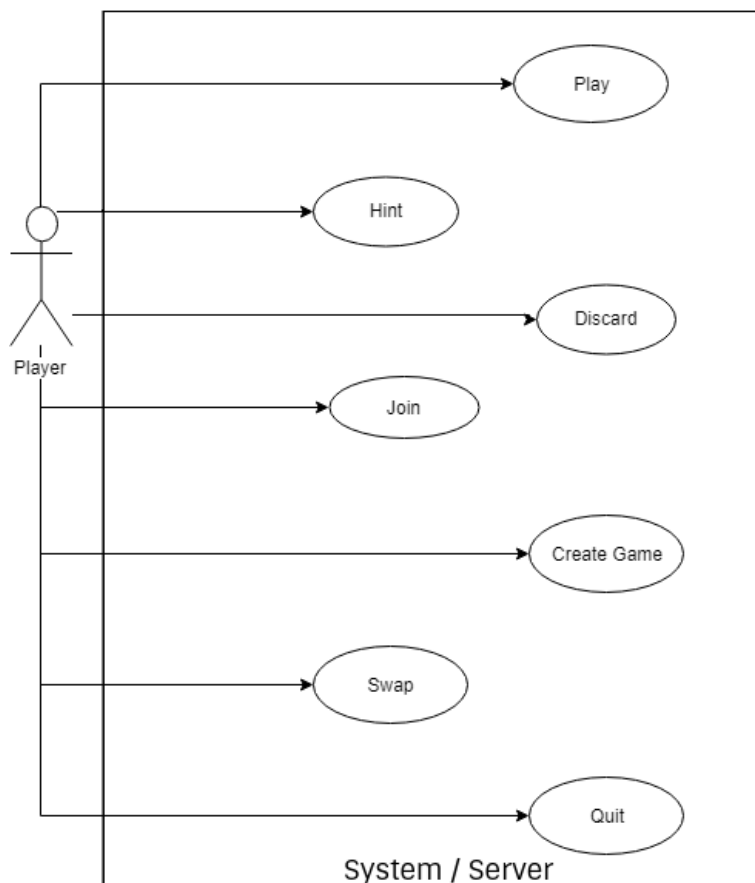


Figure 2.1: The System Environment

2.2 Product Functions

With this product, users will be able to engage in a multiplayer game of Hanabi, as previously discussed. Once the application is open, the users are shown a menu with the options to create or join a session of the game. If the player clicks join, they will be asked a games id, and a secret key to join the game that is waiting for additional players. If a session is created, a pair of game id and a secret key will be generated, and the host can wait for other players to join. Note that the communication between players and lobbies are dealt with via server, making it so that players must use separate devices to play with other human players. Once in the game, the players are shown a menu where several interface components appear when cards are clicked on. These components enable the player to perform 1 of 3 actions (per turn):

- Play a card
- Discard a card
- Give another player a hint as to which cards they hold in their hand

By placing cards, players earn points on a valid action and consume a fuse token when invalid. When a card discarded, players draw cards from their finite draw pile but regain an information token. When a hint is given, an information token is consumed, and the information is sent to all players. If players decide to leave the game temporarily, they are free to choose replacing themselves with a computer player. At any time, Players can choose to quit the game. In doing so, the system terminates the game for every player. These are all the ways players can engage with the game.

2.3 Functional Requirements

The game is played with only a single actor – the player. A player can engage in a session of Hanabi with up to four other players. Each player can make use of several actions to strategize or communicate with one another across the server. Two types of players exist within our game: a human or a computer AI.

When the player is a human, they can freely make choices presented within the confines of the game’s rules and restrictions. They will collaborate with other human or non-human actors to get the highest score possible. Similarly, when the player is computer controlled and will perform actions to achieve the highest score possible. Any combination of humans and computer can be made to play Hanabi, and actions are not identical between the two types of players. The computer controlled players can’t perform the following actions available to human players:

- Create game
- Quit game

- Swap with a computer

These actions that the actor can use are outlined in the following use cases.

2.3.1 Create Game



Figure 2.2: The Create Action

Brief Description: Create game action is taken by the host user to create a game. This is initiated by clicking the create button located in the start menu. Below is a detailed description of the paths a user may take by this action.

Trigger : The User clicks the Create button.

Precondition: None

Basic Path :

1. The User launches the game from their computer.
2. The User will click the create button.
3. The System will prompt the User to enter the Network Services ID (NSID) of the University Of Saskatchewan, Number of human players, Number of computer players and the turn time limit.
4. The User will then enter the NSID, number of human players up to 4, number of bots between 0 to 4 (such that the total number of players is less than five) and the turn time limit.
5. The System will send the information to the Server.
6. The Server will create a game id and a secret key for the game and notify the System.
7. The System will display the game id and secret key to the host User.

8. The System will then change the interface to the waiting area.
9. Once all required Players have joined, the System will then display the game's interface.

Alternative Paths :

- At any point before the step 7 host User can quit, this will result in a game not being created.
- At step 4 if the NSID does not match:
 1. Server will notify the System of a mismatch.
 2. System will prompt the user again for the NSID.

Post condition : A game ID and a secret key will exist in the Server.

2.3.2 Join Game

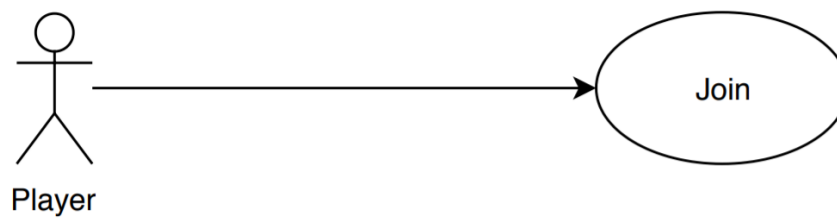


Figure 2.3: The Join Action

Brief Description: Join game action is taken by the all human Players except for the host User to join an existing game in the Server.

Trigger : The Player clicks the Join button.

Precondition: None

Basic Path :

1. The Player launches the game from their computer.
2. The System will prompt the player to enter the Network Services ID (NSID) of the University Of Saskatchewan.
3. The System then prompts the Player if the Player would like to join or create a game.

4. The Player will click the join button.
5. The System will prompt the player to enter game id and the secret key.
6. The Player will be moved to a waiting screen.
7. Once all required Players are joined The System will then display the game interface.

Alternative Paths :

- At step 5 if the game id or secret key does not match:
- At step 4 if the NSID does not match:
 1. Server will notify the system of mismatch.
 2. System will prompt the user again for the information again.

Post condition : The Player will be connected with the Server for the given game id.

2.3.3 Play Card

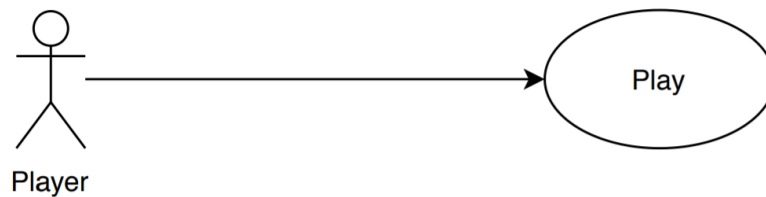


Figure 2.4: The Play Action

Brief Description: The play card action can be taken by any Player on their turn for placing a card from their hand to the a firework card stack.

Trigger : The Player attempts to play a card from their hand.

Precondition: Must be the Player's turn.

Basic Path :

1. The Player selects the card they wish to play within time limit.
2. The System displays message confirming the Player's selection with pop-up buttons yes/no assume yes.

3. The System notifies the server.
4. The Server notifies all players.
5. The System checks if the card played is valid for being placed on any of the stacks and returns with a positive result.
6. The card is moved from the Player's hand to the appropriate stack.
7. The Server checks if the deck has cards to draw and return with a positive result.
8. The Server draws a new card for Player from the deck and adds it to their hand.
9. The System displays new game state.
10. The Server then moves onto the next player's turn.

Alternative Paths :

- In step 5, if the System responds with an invalid card to be placed on stack result.
- The Server receives the signal that the Player has unsuccessfully played a card. The Server then burns one of the Fuse Tokens reducing their number by one and moves the card played from the Player's hand to the discard pile. Return to step 7.
- In step 7, if the deck no longer contains any cards to draw the Server returns a negative result.
- 6. The Server sets final turn flag to on. Return to step 9
- In step 5, if the card is a valid match and completes a suit of a color a information token will be granted to the game board given it does not exceed 8 information tokens.

Post condition : The Player's hand has a new card drawn from the deck or final turn flag set.

Exception Paths : The Player may cancel a Play action on Step 2 of the basic path. If the Player fails to act in Step 1 a pop-up message from the System will appear asking them to select an action. If they fail to act within the time limit more than 2 times, the game will terminate.

Other : The cards in the Player's hand will act as a clickable button/object that triggers the Play action or Discard action. This scenario assumes the Play action was selected at that time.



Figure 2.5: The Discard Action

2.3.4 Discard a Card

Brief Description: The discard a card action can be taken by any Player on their turn for placing a card from their hand to place it on the a discard card stack to obtain a hint token. At least one hint token should be spent.

Trigger : The Player attempts to discard a card from their hand.

Precondition: Must be the Player's turn and less than eight information tokens available.

Basic Path :

1. The Player selects the card they wish to discard within time limit.
2. The System displays message asking the Player if they wish to discard the selected card. With two buttons yes and no. Assume the Player selects yes.
3. The System checks if the information tokens have less than 8 available and receives a positive signal.
4. The System flips an information token to available position.
5. The Server checks if the deck has cards to draws and return with a positive result.
6. The Server draws new card for Player from the deck and adds it to their hand.
7. The System displays new game state.
8. The Server then moves onto next players turn.

Alternative Paths :

- In step 3, if the Server responds with a negative valid to be placed on stack result.
- 4. The Server receives the signal that the Player has unsuccessfully played a card. The Server then burns one of the Fuse Tokens reducing their number by one and moves the card played from the Player's hand to the discard pile. Return to step 5.
- In step 5, if the deck no longer contains any cards to draw the Server returns a negative result.
- 6. The Server sets final turn flag to on. Return to step 7.

Post condition : The Player's hand has a new card drawn from the deck or final turn flag set.

Exception Paths : The Player may cancel play action on Step 2 of the basic path. If the Player fails to act in Step 1 a pop-up message from system will appear asking for them to select an action. If they fail to act within the time limit more than 2 times, the game will terminate.

Other : The cards in Player's hand will act as a clickable button/object that triggers the Play action or Discard action this scenario assumes discard was selected at that time.

2.3.5 Hint



Figure 2.6: The Hint Action

Brief Description: The hint action can be taken by any Player on their turn to give another Player a hint about their hand. This can be either the number of a given colour or given number. One hint token should be available to use this action.

Trigger : The Player attempts to give hint about the cards in another Player's hand.

Precondition: Must be Players turn and at least one of eight information tokens available.

Basic Path :

1. The Player clicks a card in another player's hand within the turn time limit.
2. The System displays a pop-up message asking if the Player wants to give a hint to the Player holding the selected card. With two buttons Yes and No. Assume Yes is selected.
3. The System checks if the information token pool has at least 1 token available and receives a positive signal.
4. The System displays a message to the Player asking them if they wish to give a hint about the specific suit color or values of the cards in the other Player's hand.
5. The System then displays the appropriate pop-up for Player selection. For colours the pop-up will have buttons for all colour cards in other Player's hand, For Values the pop-up will have buttons for all the different value of cards in other Player's hand.
6. The System flips an information token to the unavailable position.
7. The System displays a new game state.
8. The System then moves onto the next Player's turn.

Alternative Paths :

- In step 3, if the System responds with a negative signal for the availability of information tokens.
- 5. The System signals to the Player that that is an invalid action then exits the hint action and lets Player select different action.

Post condition : Available information tokens decrease by 1.

Exception Paths : The Player may cancel hint action on Step 2 of the basic path. If Player fails to act in Step 1 a pop-up message will appear asking for them to select an action. If they fail to act for 2 more minutes the game will end.



Figure 2.7: The Swap Action

2.3.6 Swap

Brief Description: The swap action can be taken by any Player during anytime of the game. This will cause an computer Player to play on the human Player's behalf. The Player can also turn this off at any point of the game.

Trigger : The Player clicks on the swap button.

Precondition: None

Basic Path :

1. The Player clicks on the Swap button.
2. The System displays a message to confirm that the Player wants to swap IN if a computer Player is playing or OUT if human Player is currently playing.
3. The Player clicks YES.
4. The System starts to use the computer Player to make decisions on the Player's turns.

Alternative Paths :

- On step 2, If the Player clicks NO, the game will resume normally.
- On step 4, If the Player is swapping back in, the System will wait for the human Player's input for the Player's turns.

Post condition : None

Exception Paths : None

Other : None



Figure 2.8: The Quit Action

2.3.7 Quit

Brief Description: The quit action can be taken by any human Player at any given time. Once the game has started, quitting the game will cause the whole game to terminate.

Trigger : The Player clicks on the quit button.

Precondition: None

Basic Path :

1. Player clicks the Quit button.
2. The System displays a message asking if they wish quit the game with a YES or NO button pop up.
3. The Player clicks YES.
4. System then notifies the Server to end the game.
5. All Players are notified that the game has ended by the Server.
6. The System displays the end game screen.

Alternative Paths :

- In step 2 if the Player clicks NO the game resumes normally.
- If the game has not started and the Player quits, the System will notify the server and wait for another Player.

Post condition : The game will be at end game state

Exception Paths : None

Other : None

2.4 Operating Environment

With the use cases and functions outlined, attention is moved to where the game is playable. The Program will be operating on the University of Saskatchewan's 3rd floor Sphinx computers. These use a Red Hat 8.1.1-5 Linux 4.17.19 based system. As such, our program must be capable of running on Linux machines and communicate using Linux methods. The nature of our game client means that it should not have any particular difficulties functioning on any form of computer so long as it has the required local server containing the information of the Hanabi game.

Our Game Client will be communicating with a server local to the University of Saskatchewan. It is intended to be accessible only by students in CMPT 370. It will require an NSID in order to create a game (one game per NSID) and it will then give the creator the game-id and identifying token needed to join the game. The creator must then inform the other players of these things. If the game is not filled to capacity before ten times the turn time the game will be automatically end and the server will remove the hosted game. As a security measure each player joining will enter the game id and secret key.

2.5 User Documentation

To conclude the game overview, it should be noted that the development team will produce a formal user manual at the end of the project. In addition to the manual, each player will be offered a video tutorial at the start of each game. This tutorial will highlight the rules of the game and our easy to use graphical user interface.

3 External Interface Requirements

3.1 User Interfaces

Following the description of step by step user interactions, this document will also aim to show what the user will see as they interact with the product. Here we present the main elements of our graphical user interface. As a feature, we have decided to create an interface that prioritizes clicking to allow a future version to be easily transferred to touch screen devices. In the following subsections, we present purpose and navigation through each of these interfaces that will be presented to the user.

3.1.1 Start Menu

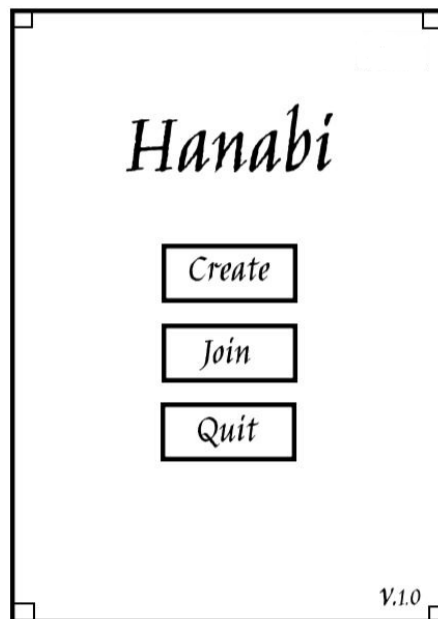


Figure 3.1: The Start Menu

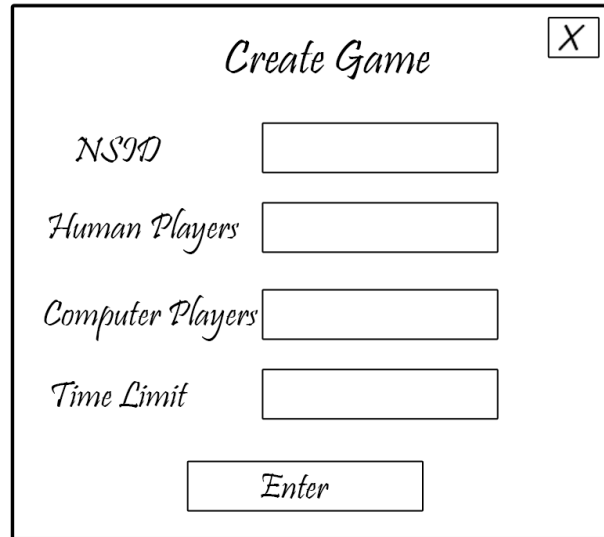
At the beginning of each game of Hanabi, all players are greeted by the start menu. This menu contains the following components:

1. Create button.
2. Join button.

3. Quit button.

Players can navigate the menu by clicking on one of the five buttons. Once a player click one of the buttons the associated use case from section 2.3 will be triggered.

3.1.2 Create Game Menu



The image shows a graphical user interface for creating a game. It is a rectangular window with a title bar at the top containing the text "Create Game" and a close button (X). The main area of the window contains four labels, each followed by a text input box: "NSID", "Human Players", "Computer Players", and "Time Limit". Below these input boxes is a button labeled "Enter".

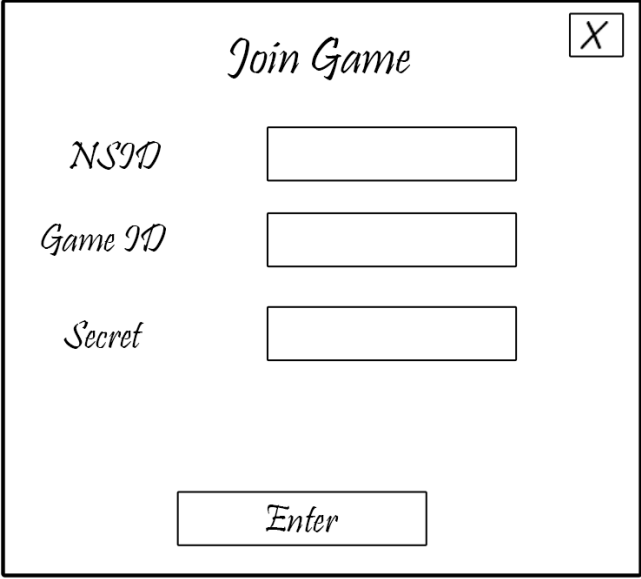
Figure 3.2: The Create Menu

Following the start menu, the user may be faced with a new menu. If a user clicks on the Create button from the start menu, the user will be shown the Create Game interface. This Interface contains the information required to create a game and it contains the following:

1. NSID for identification text box.
2. Human players number text box.
3. Ai players number text box.
4. Time limit text box.

To successfully create a game the host player is required to enter the number of human players and computer players (Total maximum player : 5). A invalid setting of player number will result a error message. Time limit for the game is each turn's duration. The successful creation of room will advance the user to waiting screen.

3.1.3 Join Game Menu



The image shows a graphical user interface for joining a game. It is a rectangular window with a black border. At the top center, the text "Join Game" is written in a cursive font. In the top right corner, there is a small square button with an "X" inside. Below the title, there are three text input fields, each preceded by a label in a cursive font: "NSID", "Game ID", and "Secret". At the bottom center of the window, there is a rectangular button labeled "Enter" in a cursive font.

Figure 3.3: The Join Game Menu

Alternatively to the menu mentioned above, the user may navigate from the start menu to the Join menu. When a player clicks on the Join button on the start menu they will be sent to the Join Game interface. In this interface will include the following items:

1. NSID text box.
2. Game ID text box.
3. Secret key text box.
4. Enter button.
5. Quit button.

To successfully join an existing game player is required to enter NSID, game id and secret key that matches with the server. Once the information is filled the player will click enter to proceed to the waiting room. Player can also click the quit button at any time.

3.1.4 Game Board

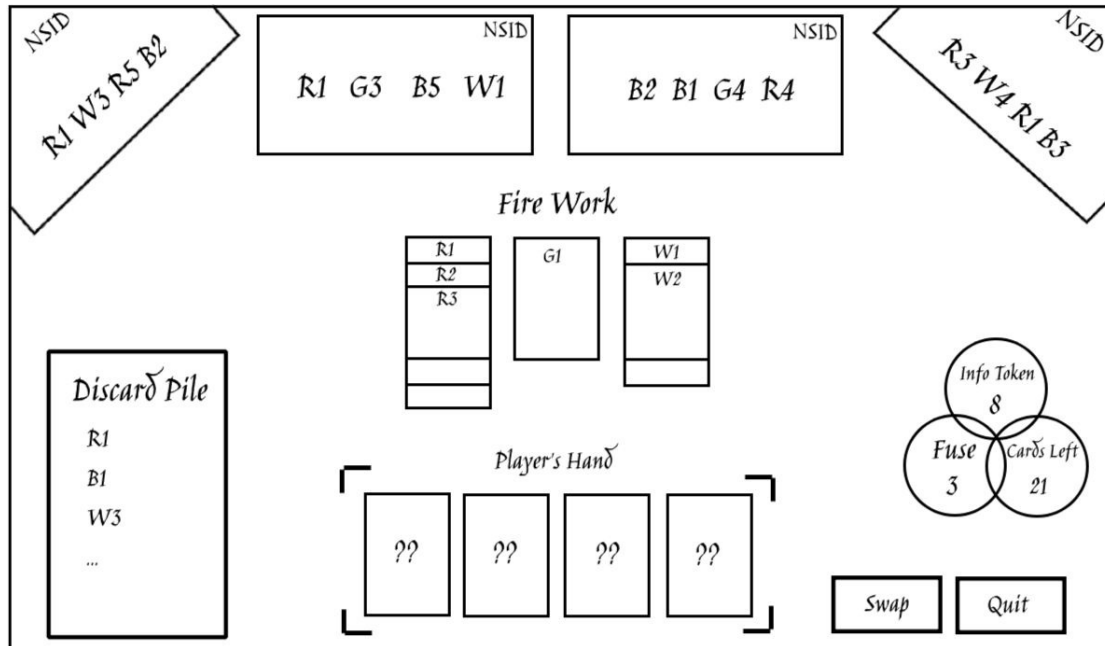


Figure 3.4: The Game Interface

Finally, after navigating the menus described above, the user finds themselves in the game. Once the game has started all users will be moved to the game board interface. This interface contains the following items:

1. Users card hand
2. number of information token
3. number of fuses left in the game
4. cards left in the deck
5. discard pile
6. fire work card stacks
7. other users card hands
8. swap button
9. quit button

Users can navigate through this interface in the following manner:

The upper part of this interface displays other player's hands. The middle part of this interface displays the fire work stack which shows what fireworks have been construct so far. The bottom part of this interface contain several components:

1. Discard pile is a scroll-able list which represents all the discarded card from the start of the game.
2. Player's own hand
3. Current game information contains:
 - a) Information Token
 - b) Fuses
 - c) Cards left in the deck
4. Swap enable player to replace himself/herself wit a bot.
5. Quit allow player to quit the game.

To interact with the interface players can follow the steps below:

- Press the card in player's own hand will give the player choice to play or discard a card.
- Press the card in other teammates' hand will give player choice to give a hint (Number or color).
- Press the Swap button will cause the game to be played by the AI bot on the players behalf.

4 System Features

In this previous chapter, we discussed the significant features that are included in the project. The following section will describe the priority and complexity of their implementation (given time constraints). Features with a high priority will be implemented first, and are considered essential. Should the time permit the more complex (low-priority) approaches for our system features, then those will be achieved following the high priority implementation. In essence, the essentials come before the upgrade, and high priority comes before low. The following section will outline these features and priority of implementation

4.1 Simple Graphical User Interface

Priority: High

A simple graphical user interface that includes pop-up windows, basic geometric shapes, and text. It will have a very basic colour pallet. Developing and implementing this feature is of high priority. The simple GUI will allow the user to click on certain graphical elements and make the experience smoother and more visually interesting.

The simple graphical elements will allow the user to see graphical representations of elements of the game Hanabi such as the cards, the tokens, and so on. This feature will be manageable for our development team and will be implemented with the use of external libraries.

4.2 Simple Artificial Intelligence

Priority: High

A simple artificial intelligence assigned to the computer controlled players of a game of Hanabi. Developing and implementing this feature is of high priority. The benefit of this system-feature is to improve the users experience playing the game; by including computer controlled players, human users can play when 4 other humans are not available. This feature will be of manageable cost to our development team.

The feature will likely be implemented with the use of external libraries. We also anticipate an issue with how the user interacts with this system feature. This issue

is the exploitation of the computer player's artificial intelligence that will negate the intended improvements that an artificial intelligence is meant to create.

4.3 Advance Artificial Intelligence

Priority: Medium

An advanced artificial intelligence that will be assigned to the computer controlled players of a game of Hanabi. Developing and implementing this feature is of Medium priority. Like the Simple AI, the benefit of including this system-feature is to greatly improve the user's experience by adding more players. This feature will be of high cost to our development team.

The feature will certainly be implemented with the use of external libraries. We do not anticipate major issues with this system feature such as the player finding methods of exploiting the artificial intelligence.

Below we describe few of the methods we can take to develop an advanced AI:

1. Naive Bayes Machine Learning: Builds a statistical model base on the known information: Other players' hand, discard pile, firework stack, and color and number hints. It uses the statistical model to predict the probability of which cards are in its hand. Also it is possible to use a random walk simulation to increase the accuracy.
 - Pros: quick and effective base on pure math formula.
 - Cons: may not be really accurate.
2. Monte Carlo simulations: A Simulation based on a statistical model. It simulates a given number of games from the game state and evaluates which percentage of games you will win given these cards. Since Hanabi is the game with incomplete and imperfect information, the simulation can adjust accordingly, while the player gets more information to result a better simulation result.
 - Pros: Yields a more-accurate winning result when more information is obtained.
 - Cons: Depending on the simulation complexity, the progress can be slow. In order to speed up the simulation progress, it may result reducing on simulation rounds, which cause less accuracy.
3. Reinforcement Learning with Neural Networks: Chooses the best sequence of actions using simulations and past experiences. This method is best used for a card game that has an incomplete and imperfect information environment such as Hanabi.
 - Pros: Highest winning rate with a good strategy (collection of actions).

- Cons: Need to alter the progress of calculating the strategy to make it also suitable for Hanabi this can be difficult given time constraints.

4.4 Audio elements

Priority: Low

A collection of audio elements such as background music and audio feedback for the interactions made by the players. Developing this feature is of low priority. The addition of sound will add interest to the users in the game. The feature will be implemented with the use of external libraries.

4.5 Advanced Graphical User Interface

Priority: Very Low

A detailed graphical representation of the game of Hanabi. It includes unique sprites, a wide colour pallet, and animations. Developing and implementing this feature is of low priority. The advance GUI will include visual feedback to the user's input.

5 References

- BoardGameGeek. (2019). Hanabi. [online] Available at: [https : //boardgamegeek.com/boardgame/98778/hanabi](https://boardgamegeek.com/boardgame/98778/hanabi)[Accessed4Feb.2019].
- IEEE Recommended Practice for Software Requirements Specifications,” in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998
- ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering,” in ISO/IEC/IEEE 29148:2011(E) , vol., no., pp.1-94, 1 Dec. 2011
- Spiel-des-jahres.com. (2019). Hanabi — Spiel des Jahres e.V.. [online] Available at: <https://www.spiel-des-jahres.com/en/hanabi> [Accessed 4 Feb. 2019].