Final Report

Hanabi Client G3

Authored by:

Joel Siemens <jms676>

Olivier Lacoste-Bouchet <oll096>

Troy Belsher <twb431>

Destin Astrope <dja392>

User Documentation

Supplied to user

The user will be provided with a JAR file that will be runnable on all platforms that support Java.

How to install

The installing process involves ensuring that there is a valid Java Runtime Environment on the system.

How to use

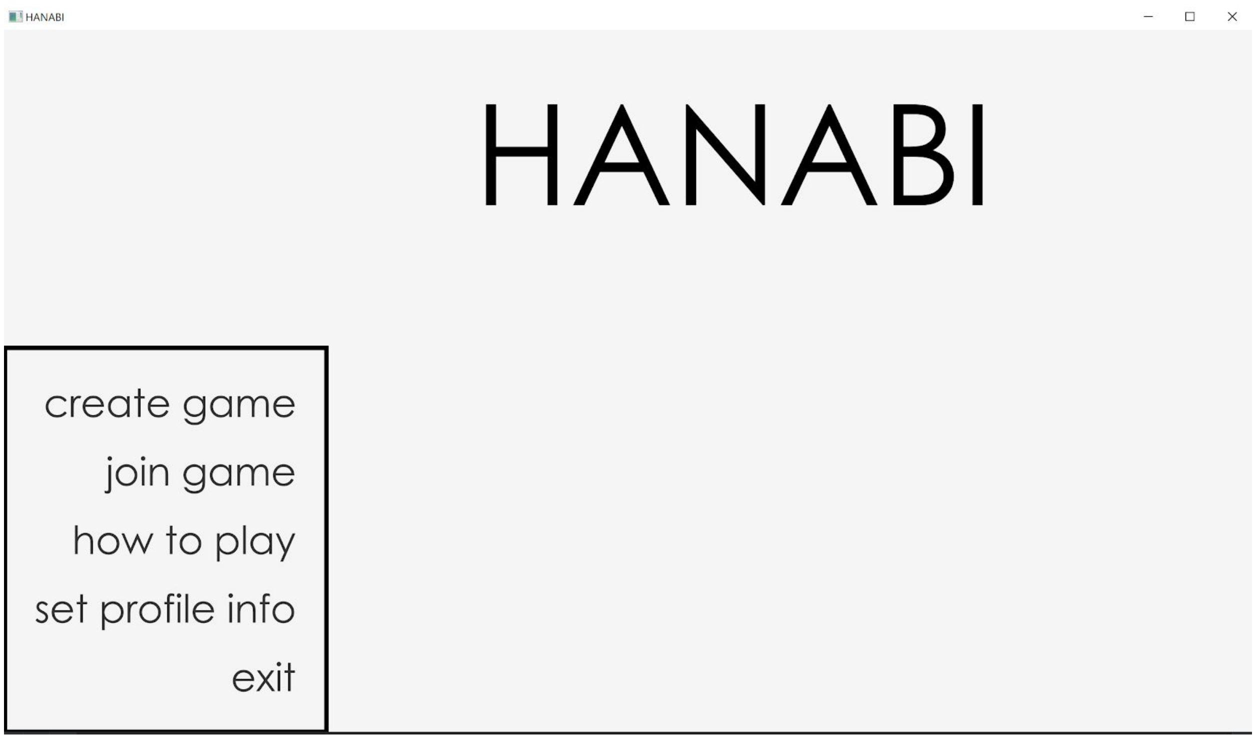
The user must navigate to the JAR directory using a terminal and enter the command:

```
    java -jar g3.jar
```

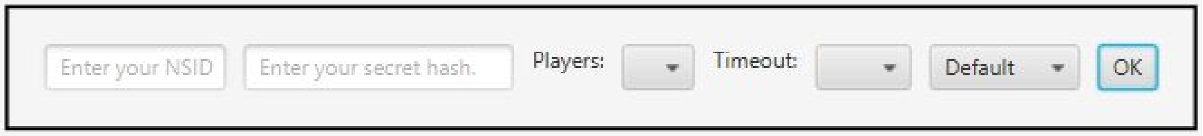
The game should launch, provided that the user has a proper Java Runtime Environment (JRE). The user will see updates on the command line. When creating a game, the game-id and secret token will be listed on the command line and must be sent to other users.

Menu

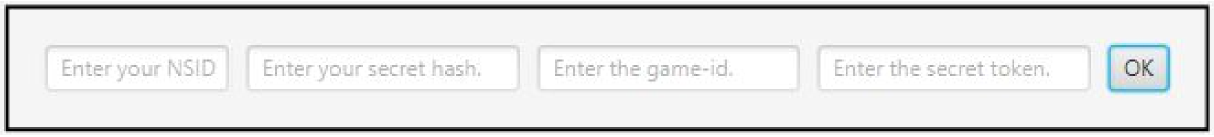
The in game menu contains five path options: create game, join game, how to play, set profile info, and exit.



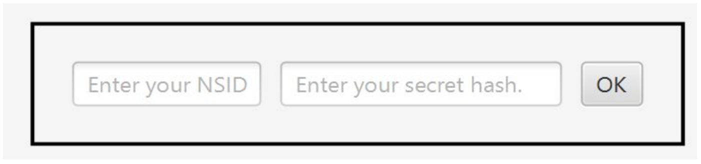
- Create a game will prompt the user for an NSID, a secret hash, number of desired players, timeout period, and the desired gamemode. Having the designated total number of players join will lead to game screen.



- Join a game will prompt the user for an NSID, a secret hash, the game ID and secret token provided by the server to the host player. Having the designated total number of players join will lead to game screen.



- How to play will bring up a window containing all the rules of the Hanabi game. This window can be kept open even in game for use as a reference to the player.
- Set profile Info will prompt the user for their NSID and secret hash. Upon pressing "OK", the game will save these values and will insert them into their respective fields when creating or joining a game by default to save the user time.

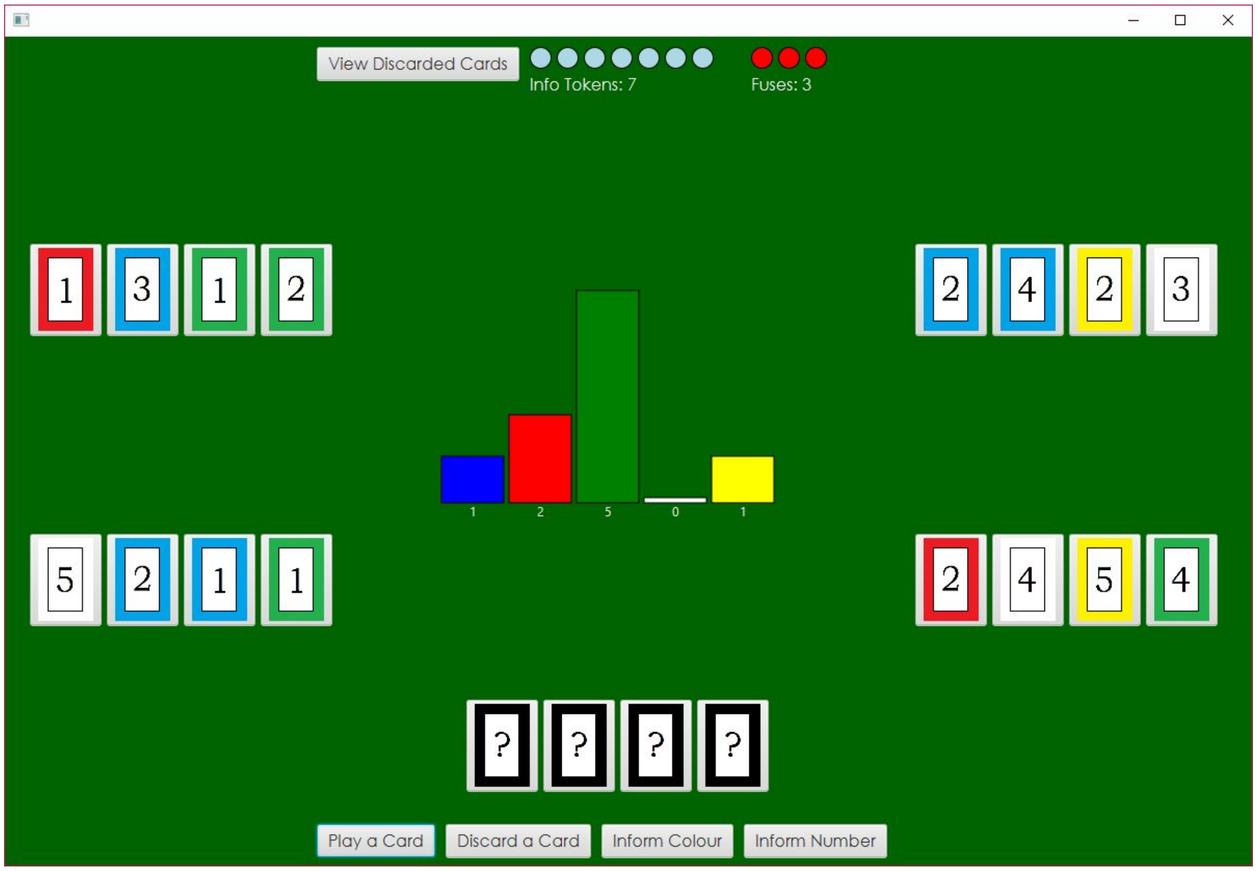


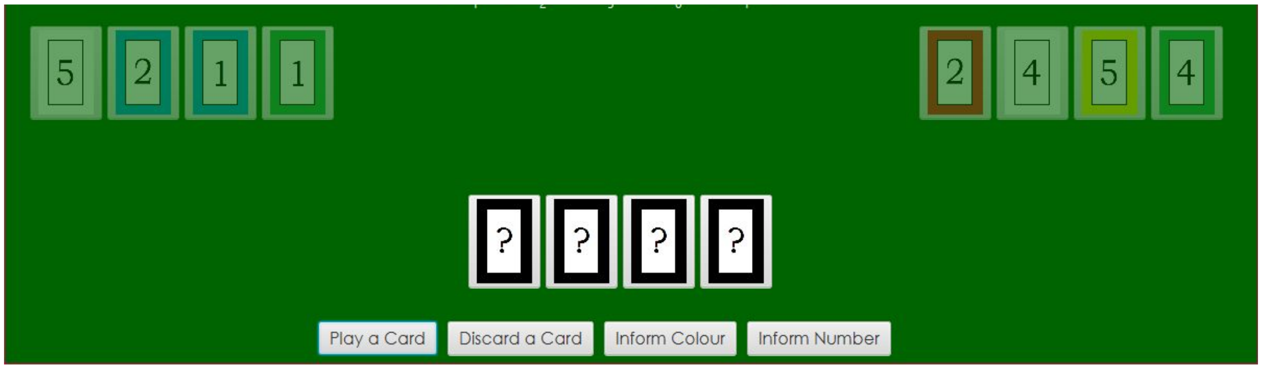
- Exit will terminate the program.

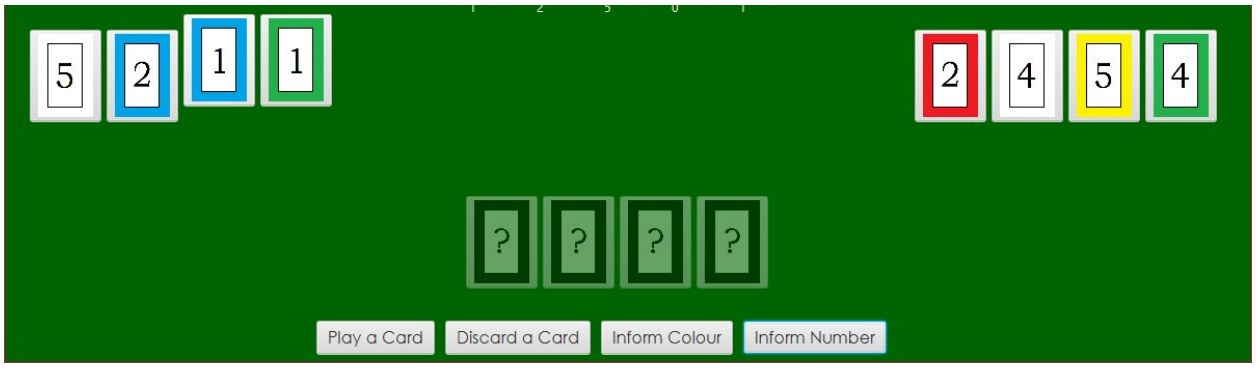
In-Game

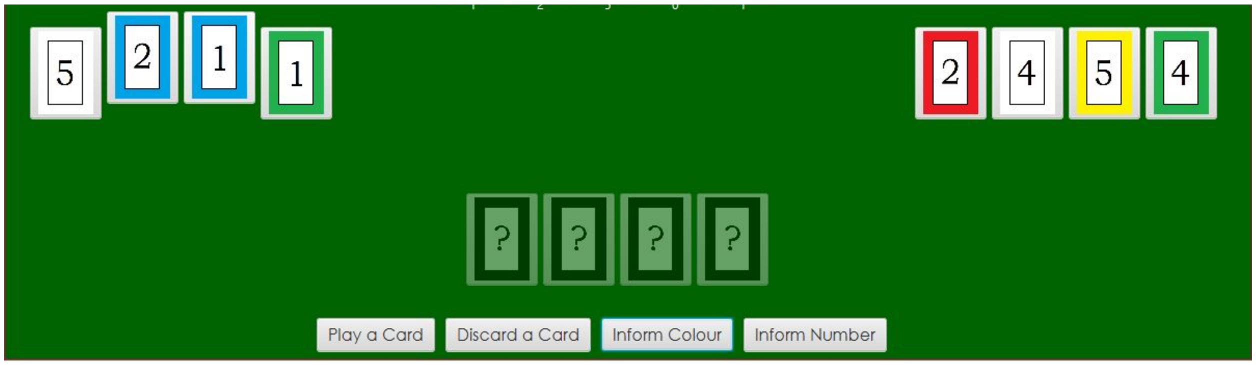
There are two possible states for the in-game window: active and inactive. The active state signifies that it is the user’s turn and inactive which signifies another player’s turn.

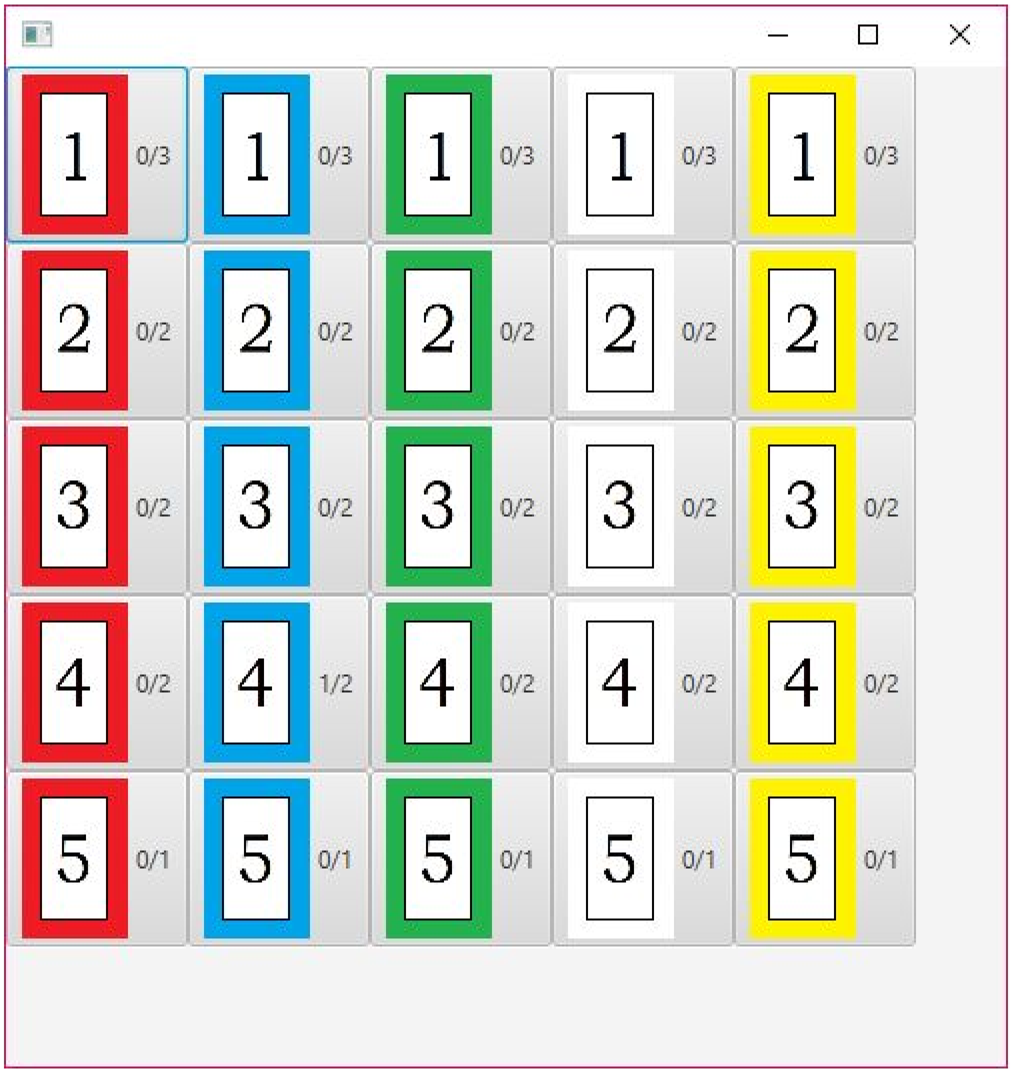
- The active state will allow players to choose one of four actions: play a card, discard a card, inform colour and inform number. All of these actions except for play a card have specific requirements, which can be read in the rules. Informing a player will raise the matching cards that will also be informed in a hand; and grey out the player’s hand. Similarly, choosing to play or discard a card will gray out teammate cards. The user will be able to see all discarded and played cards.


View Discarded Cards
Info Tokens: 7
Fuses: 3
1
2
5
0
1
Play a Card
Discard a Card
Inform Colour
Inform Number


Play a Card
Discard a Card
Inform Colour
Inform Number


Play a Card
Discard a Card
Inform Colour
Inform Number





1 0/3	1 0/3	1 0/3	1 0/3	1 0/3
2 0/2	2 0/2	2 0/2	2 0/2	2 0/2
3 0/2	3 0/2	3 0/2	3 0/2	3 0/2
4 0/2	4 1/2	4 0/2	4 0/2	4 0/2
5 0/1	5 0/1	5 0/1	5 0/1	5 0/1

- Inactive state will not allow the player to choose any action. Instead, the user must wait for other players to finish their turn.

As-Built Requirements

Several changes were made over the course of development to the program as compared to what was listed in the requirements document. Many of these differences were cosmetic. There are some slight differences in the User Interface when compared with the requirements, mostly with respect to the locations of some buttons. When the user chooses the “how to play” option on the starting screen, a new window will now appear with the information, which can then be left open for reference during the game. The layout of the discard pile window has also been adjusted to be more easily readable by the player, by showing how many cards are in the pile as well as how many of each type are in the game. The originally planned method of displaying which cards in other players’ hands had been previously informed about didn’t work, and another way of displaying them has not been decided at this time.

A few quality-of-life improvements have also been added to the program. The user is now able to save their NSID and secret for future use, so that they don’t have to input them every time they start up the game. Once in the game, the game will highlight valid choices for cards when the user chooses which action they are going to take, preventing them from trying to play another player’s cards or inform themselves about something.

Programmer Documentation

Compilation

The compilation of the program requires Java 1.8 in order to achieve a successful compilation. This is due to the use of JavaFX. The main class can be found in the HanabiClient class.

Third party libraries

There are a few third party libraries used in the compilation of this project.
JUnit version 5.2 found at https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api was used for unit testing each class that required it.
Gson version 2.8.5 found at http://central.maven.org/maven2/com/google/code/gson/gson/2.8.5/ was selected to send and receive JSON messages to and from the server.

As-Built Design

Several changes have been made to the design. Some of the in-game logic has been transferred from the controller to the view as this makes the user interface more accurate and interactive. Logic such as the the raising of the card is an example of this.

Many classes have been added since the design document, such as Server, HandBox, CardButton, FireworkRectangle, and HanabiClient. These were implemented at a later date due to unexpected essential operations and objects, mostly with regard to the game's view.

The following link leads to an updated class diagram.

https://git.cs.usask.ca/370-19/g3/blob/master/FinalProject/Top-Level_Package.png

JavaDoc

The following link leads to the index file of the JavaDoc documentation.

https://git.cs.usask.ca/370-19/g3/blob/master/FinalProject/JavaDoc/index.html

Known Bugs, Incomplete Features and Workarounds

In the software’s current state, it is unable to be used to complete a full game of Hanabi.
After the completion of the Design Document we lost our member who knew the most about networking. This made it difficult for us as now we only had 4 members of our team and none of us knew how to successfully read from a socket continuously. As a result we managed to get sending and receiving of JSON messages working on our last day of coding. This left us very little time to debug our numerous issues with the handling of JSON messages. Therefore the game does very little after the initial dealing of the cards.
Currently it is unknown if our AI works, it has been coded to perform the basic moves but we have not attempted to run it as we were unable to properly run a game as a user.
The final incomplete feature of our software is the end game screen, we never ended up coding a simple end game screen like the one presented in the requirements document.

Highlights

The highlights of this project lie in its interactive view. When it is the user’s turn, the different play options will be displayed clearly to the user. Once a play has been selected, the cards that are unable to be selected during that action will be faded, to clearly show they are not a viable option. On the other hand, when hovering over a card during the inform action, all other cards within that specific hand will be raised, in order to show the user which cards will also be informed if selected.