Eric Crandall Poker

Final Report

McEwan Bain
Jake VanZyverden
Gaberiel Hoing

Course: CPSC 224 - Software Development

I. Introduction and Project Description

Our team made a South Park themed GUI poker game, specifically the variant of Texas Hold'em. Texas Hold'Em is a variant of poker where players have 2 card hands and 5 community cards. Players win by making the best 5 card combination of cards in their hand and the community cards. The game starts with a round of betting after hands are dealt, then the first three community cards are revealed (the flop) followed by another round of betting, then the 4th card (the turn) is revealed and another round of betting follows, then the 5th card (the river) is revealed and one final round of betting ensues. During any round of betting the first player has the option to bet their desired amount, fold, or "check" the bet, subsequent players have the option to call, raise or fold. If the first player checks, the next player is also allowed to check. If all players check, the betting round ends. When a player makes a bet, that bet goes into the pot and whichever player wins the round keeps the whole pot. A round ends when all but one player folds or immediately after the last round of betting (after the river is overturned). A game is over after a set # of rounds or when one player obtains all the available chips (all other players go bankrupt).

Our version of the game allows for between two and seven players. On the start screen, players are able to enter their names, get assigned an icon, and select the number of rounds they wish to play. The number of rounds is set to five by default, and there is a setting to switch to "bust mode," where the game will progress until one player gets all the chips. A standard deck of 52 cards is used and reshuffled at the end of every round.

II. Team Members - Bios and Project Roles

Gabe Hoing is a computer science student interested in bioinformatics and artificial intelligence. His past projects include a multiplayer online trivia game and a digital rendition of Farkle. Gabe's skills include React.js, Java, C++, HTML, and Javascript. Gabe was responsible for the GUI aspects of this project, and for the game's responses to user interaction.

McEwan Bain is a computer science student with concentrations in data science and cybersecurity, and an interest in working with artificial intelligence. Past projects include simple terminal based games, dataset analysis, and GPT based chatbots. McEwan's skills include C/C++, assembly, Java, Python, Git, and GitHub. McEwan was mostly responsible for the scoring system as well as unit testing, and player icons.

Jake VanZyverden is a student studying Computer Science at Gonzaga University looking to acquire a concentration in software development. In the past he has worked on numerous java based projects. Along with some other projects in languages such as C++, HTML, and JS. Jake has strengths in the use of git/github, as well as backend java development. Jake's main responsibilities within the project were the creation of an event API for custom events/listeners. As well as major components within the flow of the game and backend functionality.

III. Project Requirements

The following section includes a table describing all of the major features we expected to incorporate into our finished product. The features included here are the elements we believe are the most central aspects to Texas Hold'em poker. The table includes descriptions of our expectations for player actions, community cards, hand scoring, the deck contents, and the ability for players to view their own cards. We were successfully able to incorporate all of these features into the game, albeit in some cases a modified form than what we originally had planned. For example the bet/raise/call feature was intended to use a custom “chips” class, but due to time constraints we ended up just using an int to store a number of chips instead.

Table 1: Major Features

Feature	*Description*
Bet/Raise/Call/Etc	Add dynamic buttons for all of your available options during your turn. These buttons will allow a player to bet, raise, call, check or fold.
Community Cards (Flop/Turn/River)	5 total community cards. The first three (flop) are shown after an initial round of betting. The next two are revealed one at a time. After each card is shown, a round of betting should ensue.
Automatic Scoring/Hand detection	After betting is finished and the river is shown, automatically determine who has the better hand
Standard Deck of Cards	A deck of 52 cards (by default) with 4 different suits, each suit has 13 cards each with different face values (2-10 + J + Q + K + A).
Mucking Cards *Peek at Cards* *(If time allows)*	Allow the winner to choose whether or not they show their hand to their opponent. A button that when held will temporarily show your hand on screen.

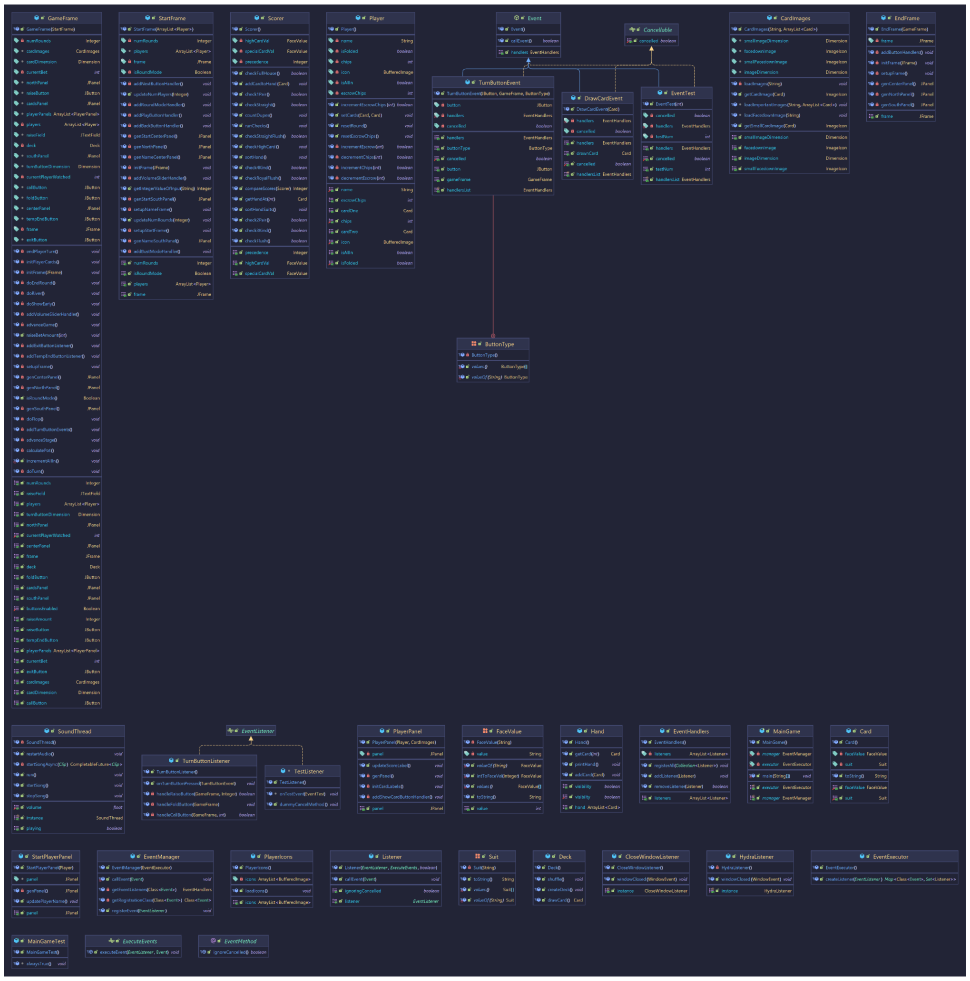
IV. Solution Approach

We designed our game from the backend first. We built up and tested all the classes we thought we would need in order to be able to put the game together. At the same time we built up a rough GUI for each of the screens we planned on having in our game. After both the GUI looked good and we had everything we thought we would need built, we started working on putting it all together.

We went ahead and skipped a terminal based form of the game and went straight into trying to work everything together with our GUI. It definitely wasn’t the prettiest or most efficient approach, but it worked out well enough in the end for us. We started by getting the community cards and player hands to flip and show at the right moments. Then we got the betting system working, this is also where we scrapped the idea for a chips class in favor of a chips integer. And finally we worked on implementing the scoring and round change system.

After we got what appeared to be a working version of the game running, we spent a lot of time play testing in order to find and fix bugs. We also spent a fair amount of time tidying up the GUI and making everything look nice.

This ended up being our final UML Class design:



V. Test Plan

The testing for a lot of the backend work like event frameworks and for a couple of the different classes was done mostly through unit testing. We ended up having 4 test classes for Scorer, Card, Deck, and Event with varying amounts of tests in each, totaling 25 different test cases (Appendix A).

For the GUI it was all just trial and error. There weren't any dedicated unit tests or anything like that, we just compiled and ran every time we implemented something new to see if it worked properly.

To test the game itself we just played it over and over, trying combinations of different options and inputs and seeing how it affected the game. We would keeped track of each player's hand and make sure the winner was chosen correctly every time we played to ensure the scoring was working and that the right amount of chips were being allotted to the right player(s). Then when we thought it was working well enough, we handed it to our friends and roommates and had them play it to see if anything they tried would break the game. After that we were pretty confident that we got the game up to snuff (Appendix B for examples).

VI. Project Implementation Description

Raise/Check/Fold -

We implemented three buttons and one text field to handle the players available options during their turn. If the player has less chips in escrow than the current bet, the check button will be replaced with a “call” button that displays how many chips they must call to be able to continue. This also works for raising in the sense that raising while you still need to call will subtract your raise amount + the amount you need to call. Folding simply removes you from rotation for the round and you lose any escrow chips.

Community Cards (Flop/Turn/River) - ***Appendix B***

Community cards are automatically revealed after betting for the round has commenced. The method calling for the advancement of community cards is run in a separate thread to allow for a delay (animation) between card flips without hanging the main thread.

Automatic Scoring/Hand detection - ***Appendix N/A (no UI)***

After the end of a round has occurred, all players who have not folded will have their hand sent to the “scorer” ; this scorer class then determines which player(s) has the best hand, at which point the pot will be added to their balance, and all players escrows will be reset to 0.

Mucking Cards - ***Appendix n***

Mucking of cards was an unintentional feature that made it into our final design. Realistically we would have liked to give the player the option to show their cards, but as it stands we only had time to just move on to the next round before showing player cards.

Peek at Cards - ***Appendix B***

Simply a button that will reveal your hidden cards while held, as soon as you let go they are reh-idden to protect other players from seeing your cards.

Volume Slider - ***Appendix n***

A slider bar that adjusts the volume of the background music.

Settings menu - ***Appendix n***

Allows the user to choose between a bust mode (play until one player owns all chips), and a round mode (play for X number of rounds, or until one player owns all chips).

Git Repository: https://github.com/GU-2023-Fall-CPSC224/final-game-project-eric-crandall-poker-face

VII. Future Work

If we were to continue this project, the first logical step would be to fully debug the game to ensure that it always works. While we extensively tested our project, Texas Hold’em has so many different aspects that some areas likely have bugs. The next step after this would be to ensure the gameplay progression perfectly follows the official rules for Texas Hold’em. There are some outlier rules which we were not able to incorporate. These rules include who gets the extra chip in a tie with odd numbers, the fact that the big blind is able to re-raise after they are called in the initial stage, and the incorporation of side-pots when players go all in. After this, I believe our next steps should be to include chips, as well as a panel in the middle of the screen where players can view the total amount of chips in the pot. It would also be good to restructure our code to make it somewhat cleaner and more readable for an outside viewer. When all this is done, I believe our initial Texas Hold'em game would be finished. The only step from here would be to include more games, Blackjack for example.

VIII. Glossary

Define technical terms used in the document.

IX. References

Player Icons:

Cartman1235. "South Park Icons." DeviantArt. Accessed December 3, 2023. https://www.deviantart.com/cartman1235/gallery/82728903/icons.

Poker Rules:

Poker.com. "Texas Hold'em Rules - An Idiot's Guide." Poker. Accessed December 3, 2023. https://poker.com/poker-games/texas-holdem-rules-an-idiots-guide/.

Other Resources:

"File:Speaker Icon.Svg." Wikipedia. Accessed December 3, 2023. https://en.m.wikipedia.org/wiki/File:Speaker_Icon.svg.

Curt. "Playing Cards (Vector & PNG)." OpenGameArt.org, April 17, 2013. https://opengameart.org/content/playing-cards-vector-png.

Card game playing - free vector graphic on Pixabay. Accessed December 3, 2023. https://pixabay.com/vectors/card-card-game-playing-card-game-7031432/.

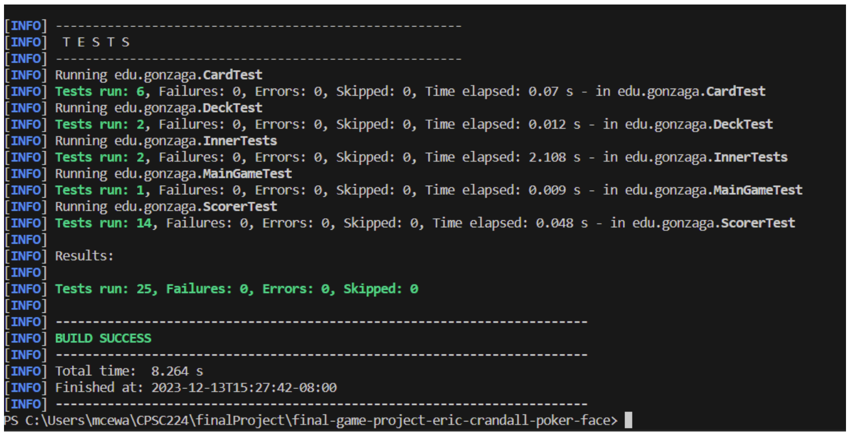
"Eric Cartman." Wikipedia, November 28, 2023. https://en.wikipedia.org/wiki/Eric_Cartman.

Lungu, Cristian. "Ace of Hearts, Casino, King of Spades, Playing Cards, Poker Icon - Download on Iconfinder." Iconfinder. Accessed December 3, 2023. https://www.iconfinder.com/icons/274869/ace_of_hearts_casino_kingof_spades_ playing_cards_poker_icon.

X. Appendices

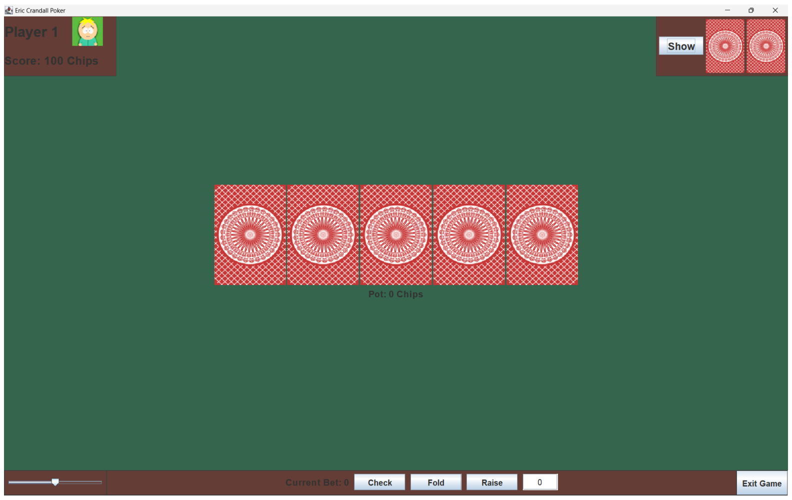
As needed, copy over your appendices for the various sections. You can have as many appendices as required. Normally, they're numbered with letters:

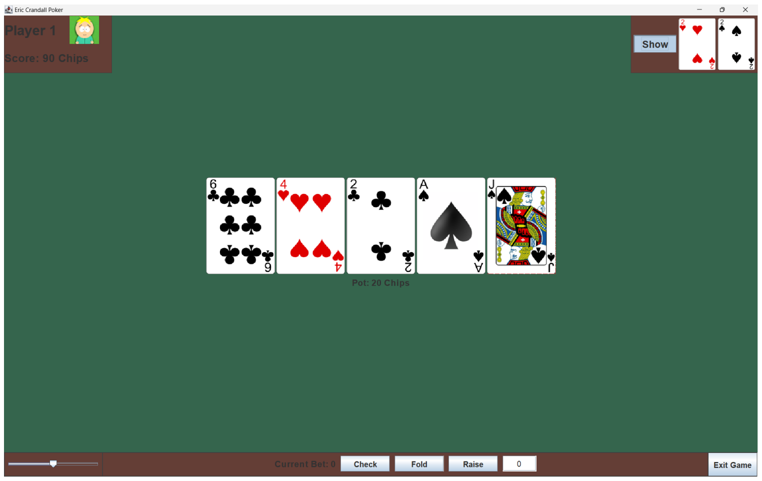
Appendix A:



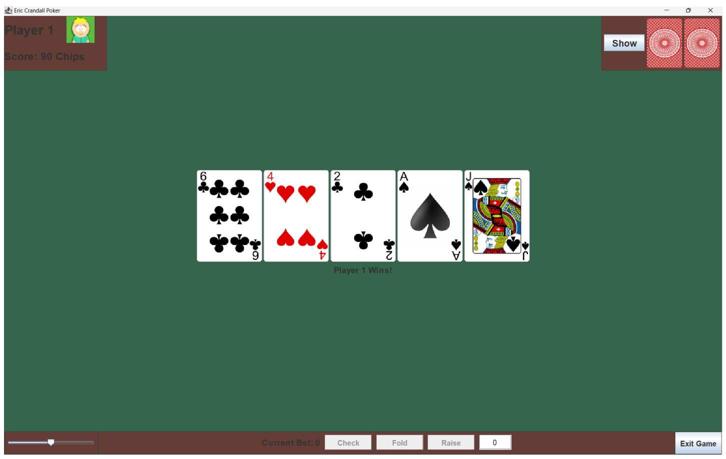
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.gonzaga.CardTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.07 s - in edu.gonzaga.CardTest
[INFO] Running edu.gonzaga.DeckTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 s - in edu.gonzaga.DeckTest
[INFO] Running edu.gonzaga.InnerTests
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.108 s - in edu.gonzaga.InnerTests
[INFO] Running edu.gonzaga.MainGameTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 s - in edu.gonzaga.MainGameTest
[INFO] Running edu.gonzaga.ScorerTest
[INFO] Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.048 s - in edu.gonzaga.ScorerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 25, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.264 s
[INFO] Finished at: 2023-12-13T15:27:42-08:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\mcewa\CPSC224\finalProject\final-game-project-eric-crandall-poker-face>
```

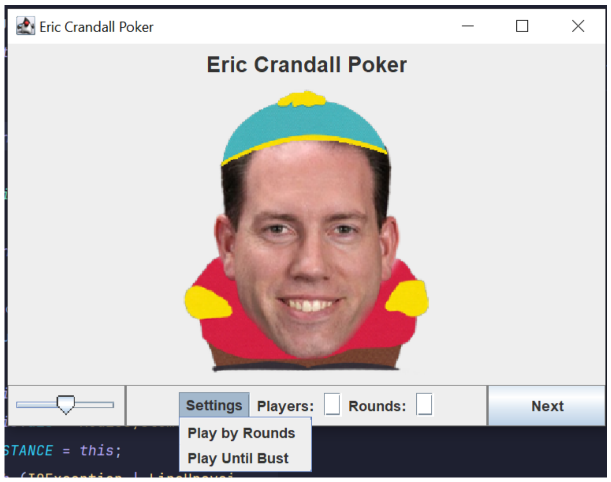
Appendix B:





Appendix *n*


Eric Crandall Poker
Player 1
Score: 90 Chips
Show
Player 1 Wins!
Current Bet: 0
Check
Fold
Raise
0
Exit Game


Eric Crandall Poker
Eric Crandall Poker
Settings
Players:
Rounds:
Next
Play by Rounds
Play Until Bust