CS 202 – Final Project

By: Adrian Antonio, Benjamin Stream, Jay-Mark Pascua, & Solomon Himelbloom

Texas Hold’em Poker

Initial Ideas

- Four player game of poker
- Networking → Multiplayer → Pass and play
- Runs locally though possible hands
- Show a SFML window with rendered graphics (cards)
- Controls are mirrored in the console output

Goals (front end vs. back end)

- Basic user inputs
  - Check, bet, call, raise, fold
- Dynamically update cards on screen
- Hand analysis & ranking after every round
- Card storage

Demo

Adrian

Worked with the beginning of the program.

- displayCard.h
  - Handles the drawing of the cards in a SFML window
- Deck.h
  - Decided with how to handle cards. Worked with sorting the cards into different files.

```
static sf::Texture cardMap;
class displayCard {
    friend void cardDisplayValue(std::vector<displayCard>& putCards, std::vector<std::pair<int,
std::string>>& cards);//Do this first
    friend void offsetPosition(std::vector<displayCard>& cardPool, float y);//Moves cards to the right
by 50*wherever it is in the vector
    friend void initialPosition(std::vector<displayCard>& cardPool, float x, float y);
    friend void screenCards(std::vector<displayCard>& cardPool, sf::RenderWindow& display);//Draws on a
RenderedWindow

public:
    displayCard(int LorR, int UorB);
    sf::Sprite cardSprite;
private:
};
```

```
displayCard::displayCard(int LorR, int TorB) {
    if (!cardMap.loadFromFile("assets/momoko_Deck_of_52_Stylized_Playing_Cards.png")) {
        std::cout << "ERROR LOADING FILE" << std::endl;
    }
    sf::IntRect data(LorR, TorB, 71, 104);
    cardSprite.setTexture(cardMap);
    cardSprite.setTextureRect(data);
}
```

Benjamin

Mainly worked on ranking poker hands.

These three functions made the base of all of the rank conditions we have.

1. searchHandSuit();
2. getHandCard();
3. getHandSuit();

With these three functions I was able to recognize the 10 winning hands in poker. Solomon and Adrian both popped in occasionally to offer input.

```
std::vector<std::pair<int, std::string>>searchHandSuit(std::vector<std::pair<int, std::string>> hand)
  // WILL RETURN A VECTOR OF PAIRS THAT IS NOT A HAND, IT CONTAINS WHICH SUIT HAS MOST OCCOURENCES
{
    std::vector<std::pair<int, std::string>> counter = { { 0,"heart" }, { 0,"club" }, { 0,"diamond" }, { 0, "spade" } };

    for (auto i: hand) {

        if (i.second == "heart")
        {
            counter[0].first++;
        }
        else if (i.second == "club")
        {
            counter[1].first++;
        }
        else if (i.second == "diamond")
        {
            counter[2].first++;
        }
        else if (i.second == "spade")
        {
            counter[3].first++;
        }
    }
    sort(counter.begin(), counter.end(), [](std::pair<int, std::string>&i, std::pair<int, std::string> &j) {return i.first >
j.first;  });
    return counter;
}
std::vector<std::pair<int,int>>searchHandCard(std::vector<std::pair<int, std::string>> hand)
  // WILL RETURN A VECTOR OF PAIRS THAT IS NOT A HAND, IT CONTAINS WHICH CARDS HAVE MOST OCCOURENCES
{
    std::vector<std::pair<int, int>> counter = { { 0,1 }, { 0,2 }, { 0,3 }, { 0, 4 },{ 0,5 }, { 0,6 }, { 0,7 }, { 0, 8 },{ 0,9 }, {
0,10 },{ 0,11 }, { 0,12 }, { 0,13 }};

    for (auto i : hand) {
    counter[i.first-1].first++;
    }
    return counter;
}
std::vector<std::pair<int, std::string>>getSuitCards(std::vector<std::pair<int, std::string>>& hand,string suit)
  // WILL RETURN A VECTOR OF PAIRS THAT IS NOT A HAND, IT CONTAINS WHICH SUIT HAS MOST OCCOURENCES
{
    sort(hand.begin(), hand.end(), [](std::pair<int, std::string>& i, std::pair<int, std::string>& j) {return i.first < j.first;
});
    std::vector<std::pair<int, std::string>> sorted;
    std::vector < std::pair<int, std::string>> heart_cards;
    std::vector < std::pair<int, std::string>> club_cards;
    std::vector < std::pair<int, std::string>> diamond_cards;
    std::vector < std::pair<int, std::string>> spade_cards;
    for (auto i : hand) {

        if (i.second == "heart")
        {
            heart_cards.push_back(i);

        }
        else if (i.second == "club")
        {
            club_cards.push_back(i);

        }
        else if (i.second == "diamond")
        {
            diamond_cards.push_back(i);

        }
        else if (i.second == "spade")
        {
            spade_cards.push_back(i);

        }

    }
    if (suit == "heart")
    {
        return heart_cards;
    }
    else if (suit == "club")
    {
        return club_cards;
    }
    else if (suit == "diamond")
    {
        return diamond_cards;
    }
    else
    {
        return spade_cards;
    }

}
```

Jay-Mark

Created game.h and player.h

- These files really helped as a nice foundation for the rest of the program

Helped Solomon with handling user input

Code could be cleaner, but we did it

```
/*
 * CS 202 Final Project (Spring 2021) Texas Hold'em Poker
 * By: Adrian Antonio, Benjamin Stream, Jay-Mark Pascua, & Solomon Himelbloom
*/

#ifndef GAME_H
#define GAME_H

#include "player.h"
#include "deck.h"
#include "ranking.h"
#include "displayCard.h"

class Game
{
public:
    Player p1;
    Player p2;
    Player p3;
    Player p4;

    Game();
    ~Game();

    void setup(int& players); // Constructs deck and player objects before starting the game
    void displayPlayerCards(sf::RenderWindow& userWindow, Player& p);
    void gameLoop(); // Starts a round of poker
    void resetRound(); // Resets the _cards, _pot, _currentBet, and _highestScore, calls setup()
    void resetBets(); // Reset players' bets and current round's highest bet
    void getPlayerInput(Player& p);
    void getPlayerSecondInput(Player& p); // Use when a player needs to respond to a raise
    int getNumericInput() const;

    void setPot(const int& bet);
    int getPot(const int& bet) const;

    void setCurrentBet(const int& bet);
    int getCurrentBet() const;

    bool everyoneCalled(); // Advances round phase if everyone has called
    bool everyoneChecked(); // Advances round phase if everyone has checked
    void letPlayerCallARaise(); // Goes through all the players that need to call a raise
    void determineWinner(); // Returns the player that won the round

    friend void Player::raise(Game& game, const int& raise);

private:
    HandAnalysis _analysis;
    Deck _cards;
    std::vector<std::pair<int, std::string>> _river;
    int _numPlayers;
    int _pot = 0; // Money earned after winning round
    int _currentBet = 0; // The round's current highest bet
    int _roundPhase = 0; // What phase the round is currently on (Phase 0: Betting, Phase 1: Deal three
cards to river
                         // Phase 2: Deal one card to river, Phase 3: Determine winner) PHASES
ALTERNATE BETWEEN BETTING AND DEALING CARDS TO RIVER
    double _highestScore = 0; // The highest score a player has; this player wins
};


#endif // !GAME_H
```

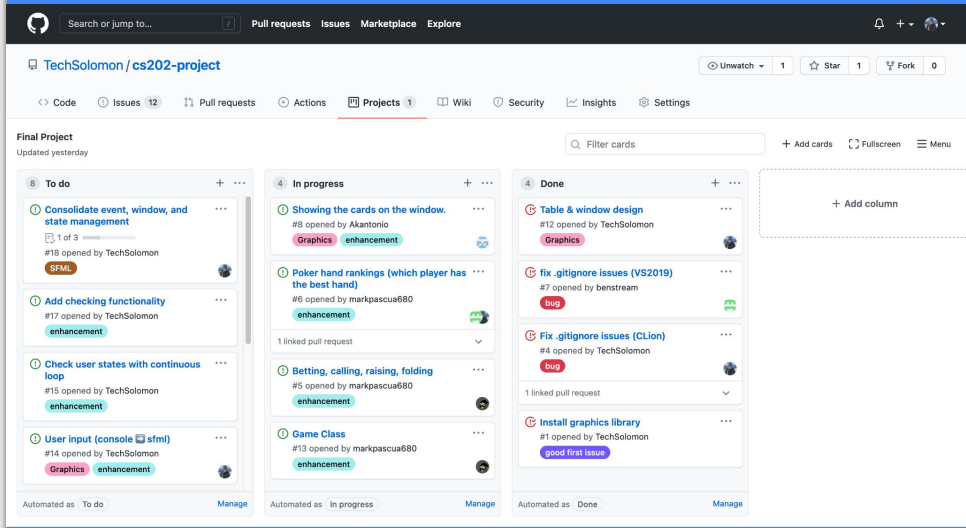
Solomon

```
sf::Texture texture;

if (!texture.loadFromFile("../assets/poker-table-design.png")) {
    throw EXIT_FAILURE;
}
sf::Sprite sprite(texture);

sf::Font font;
if (!font.loadFromFile("../assets/sansation.ttf")) {
    throw EXIT_FAILURE;
}

sf::Text playerCommands("Check (space) | Bet (b) | Call (c) | Raise (r) | Fold (f)", font, 50);
playerCommands.setFillColor(sf::Color::White);
playerCommands.move(100.f, 0.f);

sf::Text chipValues("Chip Presets (1-9)", font, 35);
chipValues.setFillColor(sf::Color::White);
chipValues.move(50.f, 500.f);

sf::Text totalPot("", font, 35);
totalPot.setString("Total Pot: $" + std::to_string(_pot));
totalPot.setFillColor(sf::Color::White);
totalPot.move(50.f, 550.f);
```

- Organizing file paths
  - macOS vs. Windows
- Resolving merge conflicts
- Keyboard input & SFML graphics/events

Key Takeaways

→ What we accomplished.



Questions?