UNO!

Presented by Alex Ringelheim &
Soo Yuan Kong

Introduction

This is a Jump-In Uno Card Game for 2-4 players.

Instructions:

- JUMP-IN Uno
- First playing card will always be **Blue-7**
- Input Typing:
  - Regular Cards: Blue-9 / Red-4 / Yellow-7 / Green-2
  - Special Cards: Red-+2 / Blue-Skip / Yellow-Skip / Green-Reverse
  - Special Cards-Wild Cards: Wild-Card / Wild-+4
    - To change color: Red/Blue/Green/Yellow (<- is how you should type it)
  - Draw Cards: Draw-Card
- Your hand is in a **text (.txt) file** in the project folder
  - Each player must re-open the file after each turn for an updated version
  - Player# .txt

Class Uno

```
// Overloads the prefix -- operator to subtract from deck count and check if the deck is empty.

Uno operator --()
{
    // Checks if deck is empty whenever it is subtracted from

    --deckCount;
    if (deckCount == 0)
    {
        std::cout << "The deck is empty, reshuffling a new dack!" << std::endl;
    }

    return Uno(deckCount, pileCount);
}

// Each function is explained above their definition. Each is used in tandem with each other to make the game work

void makeDeck();
void makeHands(int playerCount);
void playCard(int playerIndex, int playerCount);
void rotate(int playerCount, int& playerIndex, bool isSkip);
void winnerDisplay(int playerCount, int playerCardCounts[]);
void playGame(int playerCount, int playerIndex, int playerCardCounts[]);
void chooseWildColor();
void handleSpecialCards(const std::string& card, int playerIndex, int playerCount, int direction);
void updateDirection(int& playerIndex);
int skipPlayer(int& playerIndex, int playerCount, int& direction, bool& isSkip);
void drawCards(int numCards, int playerIndex, int playerCount);
```

makeDeck(): A Random set of Cards

```cpp
void Uno::makeDeck()
{

    // Holds the combination potential for the cards

    std::string numberedSet[15] = { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "+2", "Reverse", "Skip", "Wild-Card", "Wild-+4" };
    std::string colorSet[4] = { "Red-", "Yellow-", "Blue-", "Green-" };

    srand(time(0)); // Creates a seed for the randomization (initialization)

    int numberAmount = 15;
    int colorAmount = 4;
    int numberChoice = 0;
    int colorChoice = 0;
    std::string newCard;

    // 108 cards are needed for an Uno Deck, so a random number (or special card) is chosen as well as a color)

    for (int i = 0; i < 108; i++)
    {
        numberChoice = rand() % numberAmount;
        colorChoice = rand() % colorAmount;

        // Wild cards do not have a color so this prevents for example, Blue Wild, which does not exist

        if (numberedSet[numberChoice] == "Wild-Card" || numberedSet[numberChoice] == "Wild-+4")
        {
            newCard = numberedSet[numberChoice];
        }

        // Otherwise, the color and number are concated into one string

        else
        {
            newCard = colorSet[colorChoice] + numberedSet[numberChoice];
        }
```

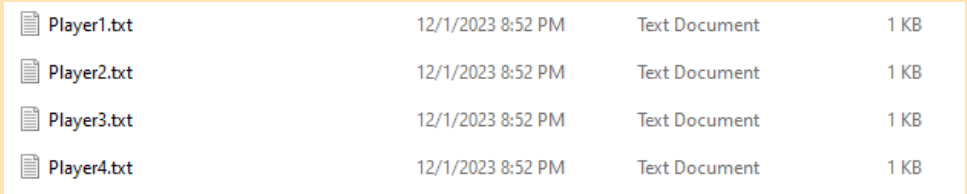
makeHands() --- files(.txt)

```
void Uno::makeHands(int numOfPlayers)
{

    // Each case is the same, just changes the file name and which vectors are added to

    switch (numOfPlayers)
    {
    case 2:
    {
        std::ofstream hands;    // Opens file
        hands.open("Player1.txt");  // Names file
        std::vector<std::string> hand1; // Two vectors for both hands
        std::vector<std::string> hand2;

        // Adds seven cards to each hand

        for (int i = 0; i < 7; i++)
        {
            hand1.push_back(deck[i]);
            hands << hand1[i] << "\n";
        }
        hands.close();  // Closes file
    }
    {

        std::ofstream hands;
        hands.open("Player2.txt");
        std::vector<std::string> hand2;
        for (int i = 0; i < 7; i++)
        {
            hand2.push_back(deck[7 + i]);
            hands << hand2[i] << "\n";

        }
        hands.close();
```

- Each case creates the files needed to start the game with.

makeHands() --- files(.txt)

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?:
```



Player1.txt

```
Red-Reverse
Green-Skip
Wild-Card
Blue-Reverse
Blue-Skip
Yellow-4
Wild-+4
```

Player2.txt

```
Red-8
Red-3
Yellow-2
Green-1
Green-+2
Green-+2
Wild-Card
```

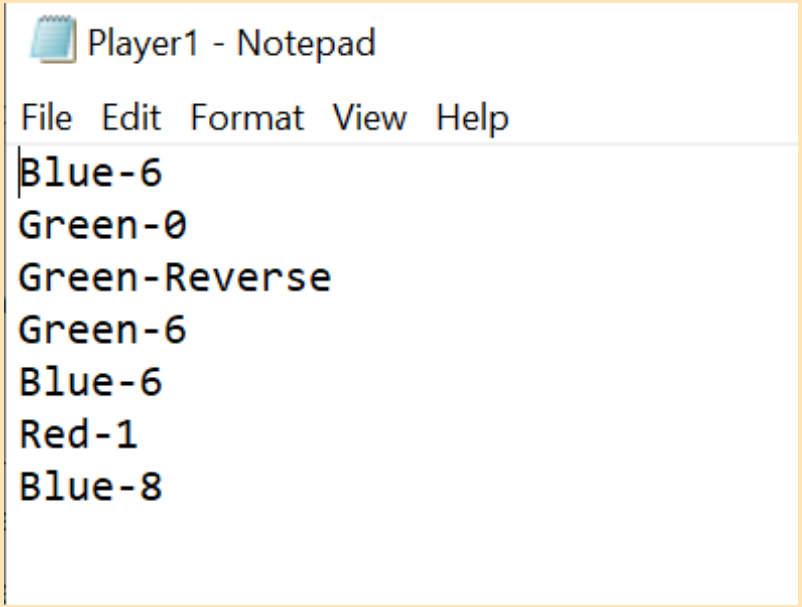
Player3.txt

```
Yellow-Reverse
Green-4
Blue-9
Yellow-Reverse
Green-2
Yellow-1
Wild-Card
```

Player4.txt

```
Yellow-5
Yellow-4
Wild-+4
Wild-+4
Wild-Card
Red-1
Blue-6
```

Example of a Hand



- Type the card as you see it to play, keeping in mind the first card in the center pile is always Blue-7

Cards!!

7 cards per player
Colors: Red/Blue/Yellow/Green

Numbers: 0/1/2/3/4/5/6/7/8/9

Special Cards: Reverse/Skip/+2/Wild/Wild+4

Special Card Handling

```
// Func. for all special cards (Reverse/Skip/+2/Wild/Wild+4)

void Uno::handleSpecialCards(const std::string& card, int playerIndex, int playerCount, int direction)
{
    if (card == "Blue-Reverse" || card == "Yellow-Reverse" || card == "Red-Reverse" || card == "Green-Reverse")
    {
        updateDirection(playerIndex);
    }
    else if (card == "Blue-Skip" || card == "Yellow-Skip" || card == "Red-Skip" || card == "Green-Skip")
    {
        skipPlayer(playerIndex, playerCount, direction, isSkip);
    }
    else if (card == "Blue-+2" || card == "Yellow-+2" || card == "Red-+2" || card == "Green-+2")
    {
        drawCards(2, playerIndex, playerCount);
    }
    else if (card == "Wild-Card")
    {
        chooseWildColor();
    }

    else if (card == "Wild-+4")
    {
        drawCards(4, playerIndex, playerCount);
        chooseWildColor();
    }
}
```

Rotate Direction-Clockwise

```
// Func. that moves to next player based on direction (clockwise/counterclockwise). This func. also works with Skip card to properly change playerIndex

void Uno::rotate(int playerCount, int& playerIndex, bool isSkip)
{
    if (direction == 0)
    {
        if (isSkip == 1)
        {
            playerIndex = (playerIndex + 2) % playerCount;  // Gets correct player index based on how many players there are
            isSkip = 0;
        }

        else if (playerIndex < playerCount - 1)
        {
            playerIndex++;
        }
        else
        {
            playerIndex = 0;
        }
    }
```

Rotate Direction-Counterclockwise

```
    if (direction == 1)
    {
        if (isSkip == 1)
        {
            playerIndex = (playerIndex - 2) % playerCount;
            isSkip = 0;
        }

        else if (playerIndex == 0)
        {
            playerIndex = playerCount - 1;
        }

        else if (playerIndex < playerCount && playerIndex != 0)
        {
            playerIndex--;
        }
    }
}
```

Draw Cards & Special Cards: +2/ Wild+4

```
// Draws cards of any amount

void Uno::drawCards(int numCards, int playerIndex, int playerCount)
{
    // Amount of cards to draw

    for (int i = 0; i < numCards; i++)
    {
        int nextPlayerIndex = (playerIndex + 1) % playerCount;  // The next player recieves the cards
        std::ofstream hands("Player" + std::to_string(nextPlayerIndex + 1) + ".txt", std::ios::app);

        // Subtracts from deck count

        if (deckCount > 0)
        {
            hands << deck[--deckCount] << "\n";
            hands.close();
        }
        else
        {
            std::cout << "Cannot draw a card. The deck is empty." << std::endl;
            hands.close();
            break;
        }
    }
}
```

Special Cards: Reverse

```
// Func. for Reverse card to switch direction of the game

void Uno::updateDirection(int& playerIndex)
{
    if (direction == 0) // clockwise direction
    {
        direction = 1;
    }

    else if (direction == 1) // counterclockwise direction
    {
        direction = 0;
    }
}
```

Special Cards: Reverse

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?: Wild-Card
Here is the current pile card: Blue-7
Card Found
Card in play: Wild-Card
Choose a color (Red/Blue/Green/Yellow): Red
Here is the new pile color: Red
Which card would you like to play from your hand, Player 2?: Red-Reverse
Here is the current pile card: Red-7
Card Found
Card in play: Red-Reverse
Card Played!
Here is the current pile card: Red-Reverse
Which card would you like to play from your hand, Player 1?:
```

Special Cards: Skip

```
// Func. for Skip card to skip players in clockwise/counterclockwise direction

int Uno::skipPlayer(int& playerIndex, int playerCount, int& direction, bool& isSkip)
{
    isSkip = 1;
    if (direction == 0)
    {
        playerIndex = (playerIndex + 2) % playerCount;
    }
    else if (direction == 1)
    {
        playerIndex = (playerIndex - 2 + playerCount) % playerCount;
    }

    return playerIndex;
}
```

Special Cards: Skip

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?: Blue-Skip
Here is the current pile card: Blue-7
Card Found
Card in play: Blue-Skip
Card Played!
Here is the current pile card: Blue-Skip
Which card would you like to play from your hand, Player 3?:
```

Special Cards: Wild/ Wild+4

```
// Func. for Wild and Wild+4 special cards to change the pile color

void Uno::chooseWildColor()
{
    std::cout << "Choose a color (Red/Blue/Green/Yellow): ";
    std::cin >> pileColor;
    std::cout << "Here is the new pile color: " << pileColor << std::endl;
}
```

- Allows user to change the color when a wild/wild+4 is played

Special Cards: Wild/ Wild+4

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?: Wild-Card
Here is the current pile card: Blue-7
Card Found
Card Found
Card in play: Wild-Card
Choose a color (Red/Blue/Green/Yellow): Yellow
Here is the new pile color: Yellow
Which card would you like to play from your hand, Player 2?: Yellow-9
Here is the current pile card: Yellow-7
Card Found
Card in play: Yellow-9
Card Played!
Here is the current pile card: Yellow-9
Which card would you like to play from your hand, Player 3?:
```

Special Cards: Wild/ Wild+4

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?: Wild-+4
Here is the current pile card: Blue-7
Card Found
Card in play: Wild-+4
Choose a color (Red/Blue/Green/Yellow): Red
Here is the new pile color: Red
Which card would you like to play from your hand, Player 2?:
```

Player2.txt

Before

```
Green-2
Blue-9
Red-3
Green-6
Blue-3
Yellow-1
```

After

```
Green-Skip
Green-4
Blue-5
Red-2
Red-7
Yellow-6
Red-1
Blue-Skip
Yellow-0
Green-2
Yellow-1
```

Winner-Bubble Sort

```
// Displays the winner using bubble sort. It sorts the winner from lowest to highest card counts when one p

void Uno::winnerDisplay(int playerCount, int playerCardCounts[])
{
    for (int i = 0; i < playerCount; i++)
    {
        for (int j = i + 1; j < playerCount; j++)
        {
            //comparing votes row by row
            if (playerCardCounts[i] > playerCardCounts[j])
            {
                int temp = playerCardCounts[i]; //Create temp as a token for # of counts
                playerCardCounts[i] = playerCardCounts[j];
                playerCardCounts[j] = temp;
                std::string tempp = playerNames[i]; //Create tempp as a token for names
                playerNames[i] = playerNames[j];
                playerNames[j] = tempp;
            }
        }
        //output results
        std::cout << std::setw(10) << playerNames[i] << std::setw(12) << playerCardCounts[i] << std::endl;
    }
    // output winner
    std::cout << playerNames[0] << " wins!" << std::endl;
}
```

Winner Sort Example

```
        bbb                 0
        ccc                 3
        ddd                 4
        aaa                 6
bbb wins!
```

playGame()

- Play game is a vital part of the code. It strings together all the functions to make the game work.
- On first time run, it creates the deck and hands Detects when a winner is found

```
        if (deckCount == 0)
        {
            makeDeck();
        }

        // Otherwise the game loops until there is a winner

        else
        {
            playCard(playerIndex, playerCount);
            rotate(playerCount, playerIndex, isSkip);
        }

        // Once a player has no more cards, they win

        if (playerCardCounts[playerIndex] == 0)
        {
            winner = true;
            winnerDisplay(playerCount, playerCardCounts);
            break;

        }
    }
}
```

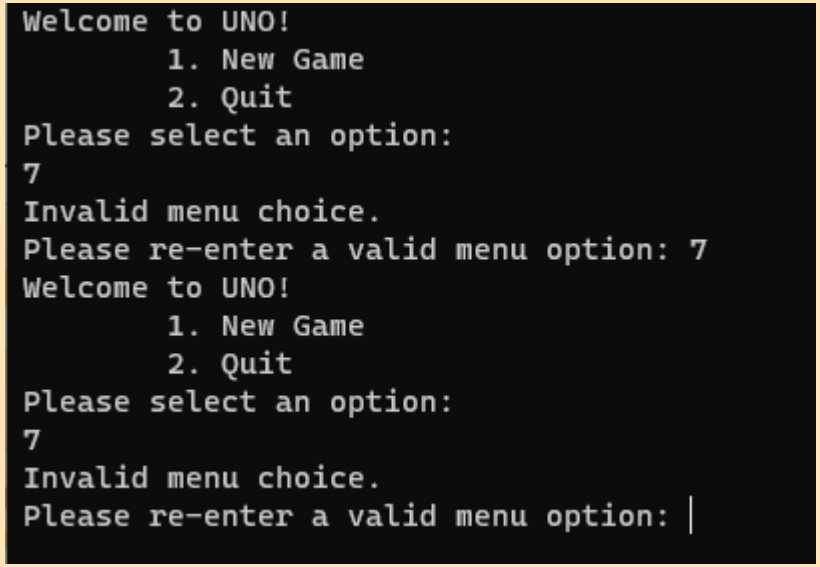
Main func-Game Menu

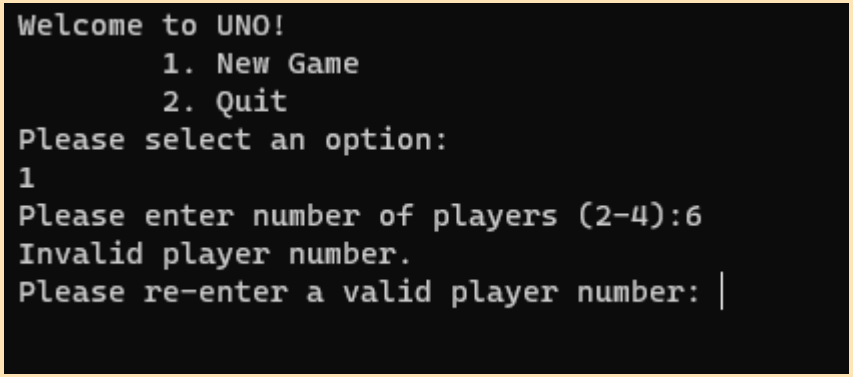
```
// Loop keeps looping until user enters 2 to exit

while (true)
{
    int option;
    int numOfPlayers;
    // Outputing game menu
    std::cout << "Welcome to UNO!" << std::endl;
    std::cout << "\t1. New Game" << std::endl;
    std::cout << "\t2. Quit" << std::endl;
    std::cout << "Please select an option:" << std::endl;
    std::cin >> option;

    // Error handling for user inputs during menu

    try
    {
        if (option != 1 && option != 2)
        {
            throw std::exception();
        }
    }

    catch (std::exception& e)
    {
        std::cout << "Invalid menu choice." << "\nPlease re-enter a valid menu option: ";
        std::cin >> option;
    }
```

```
    if (option == 1)
    {
        std::cout << "Please enter number of players (2-4):";
        std::cin >> numOfPlayers;
        unoGame.playGame(numOfPlayers, 0, unoGame.playerCardCounts);    // Plays the game
        |
        try
        {
            if (numOfPlayers < 2 || numOfPlayers > 5) // as it is a 2-4 players game
            {
                throw std::exception();
            }
        }

        catch (std::exception& e)
        {
            std::cout << "Invalid player number." << "\nPlease re-enter a valid player number: ";
            std::cin >> numOfPlayers;
        }
    }
    else if (option == 2)
    {
        exit(0);    // exits game if user does not want to play again
    }
}
}
```

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
```

Exception handling for menu option and number of players





Exception handling for menu options and number of players

option==1...Example of Game Start

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
1
Please enter number of players (2-4):4
Please enter name for Player 1: aa
Please enter name for Player 2: bb
Please enter name for Player 3: cc
Please enter name for Player 4: dd
All 4 players may begin their match!
Which card would you like to play from your hand, Player 1?: Blue-4
Here is the current pile card: Blue-7
Card Found
Card in play: Blue-4
Card Played!
Here is the current pile card: Blue-4
Which card would you like to play from your hand, Player 2?:
```

Player1.txt

```
Blue-4
Blue-8
Blue-Skip
Yellow-Reverse
Red-0
Wild-+4
Green-5
```

Before

```
Blue-8
Blue-Skip
Yellow-Reverse
Red-0
Wild-+4
Green-5
```

After

option==1...Example of Game End

```
        bbb                 0
        ccc                 3
        ddd                 4
        aaa                 6
bbb wins!
```

option==2

```
Welcome to UNO!
        1. New Game
        2. Quit
Please select an option:
2

C:\Users\kongs\source\repos\CSC1061_FinalProject\x64\Debug\CSC1061_FinalProject.exe (process 13184) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Exit Game