

report2

by Silvia Settle

Submission date: 03-May-2023 01:18AM (UTC+0100)

Submission ID: 204428226

File name: 5780_Silvia_Settle_report2_499749_1052515156.docx (314.22K)

Word count: 1092

Character count: 6225

2023

Socket Programming and Concurrency

ACCA6028

SILVIA SETTLE 1708386

Contents

Introduction	2
Design and scope	2
Design diagrams	2
Main states	3
Implementation	3
Networked application Implementation	4
Conclusion.....	4
Appendix 1	5
Sources.....	10

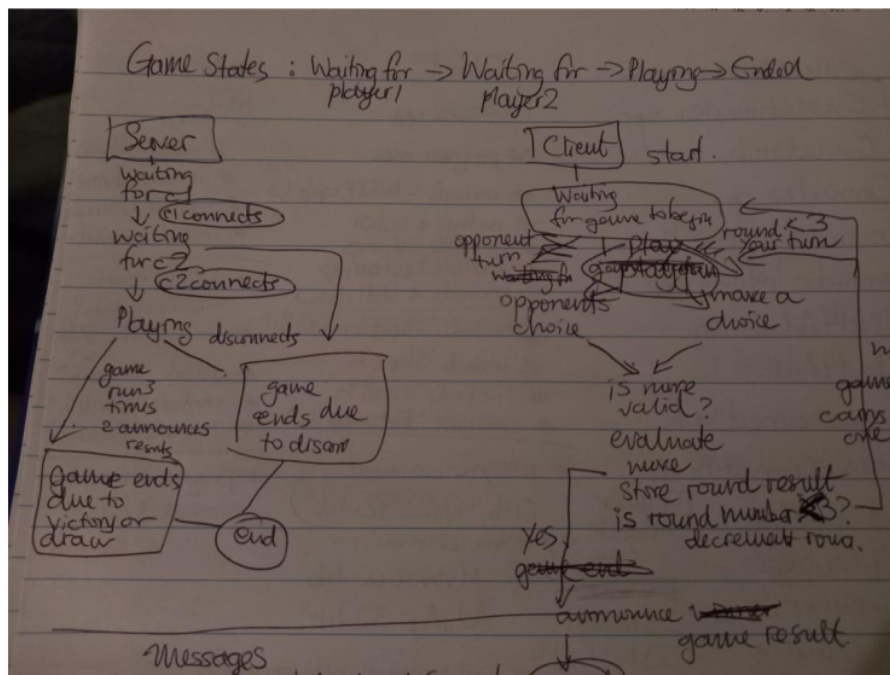
Introduction

The application I developed in C++ and partially implemented as a networked application is a Rock-Paper Scissors game.

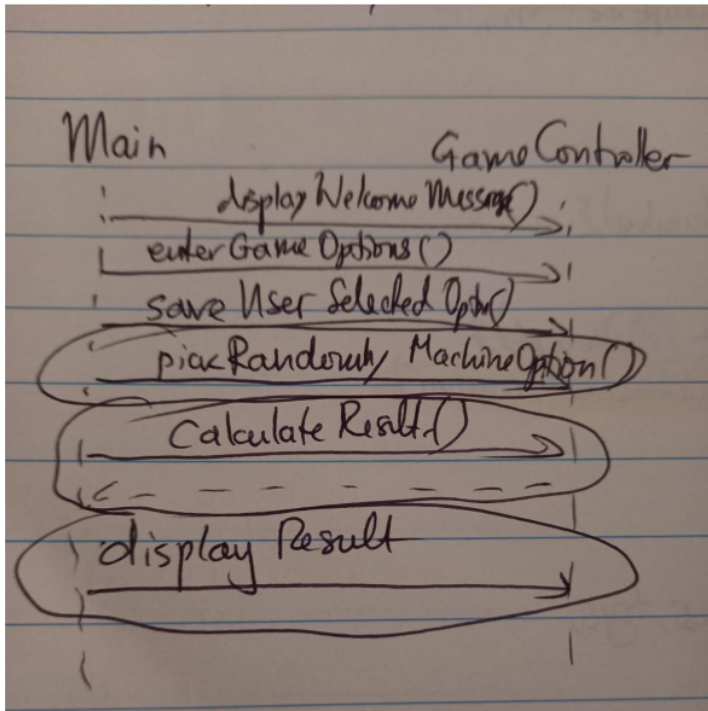
Design and scope

The game is designed as a two-player game using object-oriented programming (OOP) principles, that allow the creation of classes and object instances, which gives more control of the programmer over the networking aspect of its implementation. Some of the code used is based on the sample code provided during our lectures on module ACCA6028 Socket Programming and Concurrency with Dr Tim Bashford.

Design diagrams



Main states



Implementation

The initial solution had two classes: a Player and a Game. The Player class has a name, score and a choice as private attributes, and public setter and getter methods for the name, the choice and to get score and increment the score. The choice for having private attributes and public methods was led by the OOP principles. I have also created a constructor and a destructor methods as part of the class in view of using these for the networked part of the implementation.

The Game class has two objects of the type Player class, four private methods evaluateChoices(), getChoiceName(), takeTurns(), announceWinner(), and one public method that makes use of the other four: play(). The evaluateChoices() method takes two players as parameters and returns an integer 0, 1 or 2 to indicate who has won the round: 0 is the equivalent of a draw, 1 is indicating that the winner is player 1, and 2 that the winner is player 2. The getChoices() method returns a string to indicate the player's choice between 'Rock', 'Paper' and 'Scissors', and it takes in a parameter of the type Choice (an enum from the Player's class). The takeTurns() method takes two parameters – player1 and player2 and loops through taking turn sequence 3 times, while calling the getChoices() and evaluateChoices() methods. Finally, the announceWinner() method takes two players as parameters and compares the player's overall score after the third round. The returned integer again indicates 0 – draw, 1 – winner is player 1, 2 – winner is player 2.

Finally, the play() method controls the flow of the game: welcomes the players, takes their names and calls the takeTurns() and the announceWinner() methods, before ending the game with a "thanks for playing" message.

The source.cpp file contains the instance of the Game class and is calling the play() method.

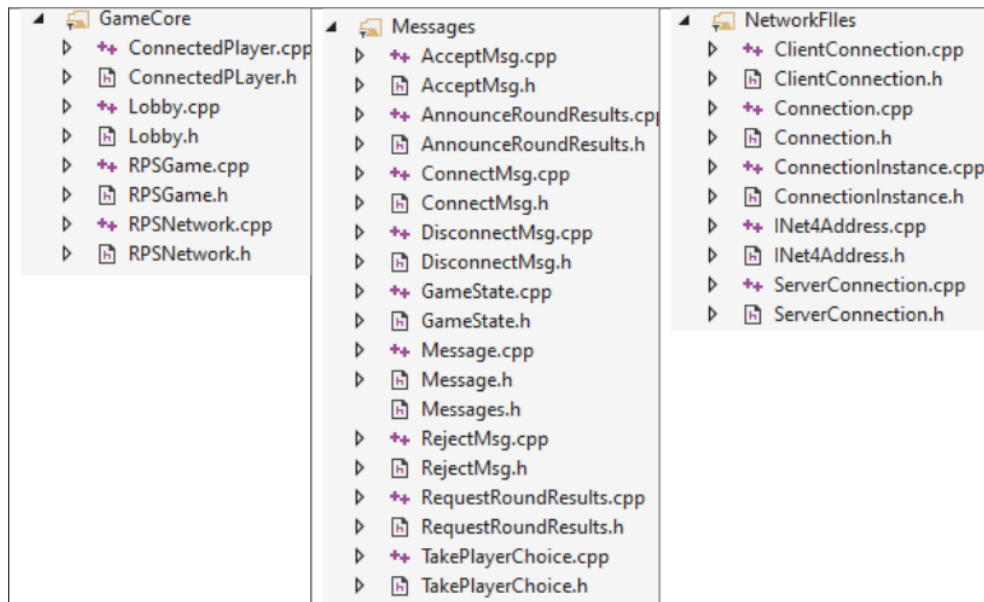
Please find this code attached in Appendix 1.

Networked application Implementation

The networked application was only partially developed to demonstrate how the game can be played by two separate players over a network. The application has a Network Files section, messages, and Game core section. The NetworkFiles and Messages sections were closely based on Dr. Tim Bashford's solution for networked Tic Tac Toe game.

Conclusion

The game makes use of threading to allow separate player connection instances. The messages provide the facility for the server to communicate and exchange information with the clients, while controlling the flow of processes.



Appendix 1

Initial Solution

```
//Player.h
#pragma
#include <string>

using namespace std;

enum Choice { ROCK = 1, PAPER, SCISSORS };

class Player {
public:
    Player(const string& pName);
    ~Player();
    string getName() const;
    void setName(string pName);
    int getScore();
    void incrementScore();
    Choice getPlayerChoice() const;
    void setPlayerChoice(int pChoice);
private:
    string name;
    int score;
    Choice choice;
};

//Player.cpp
#include "Player.h"
#include <iostream>

using namespace std;

Player::Player(const string& pName) {
    this->name = pName;
    this->score = 0;
}

Player::~Player() {
}

string Player::getName() const {
    return name;
}

void Player::setName(string pName) {
    this->name = pName;
}

Choice Player::getPlayerChoice() const {
```

```

        return choice;
    }

    void Player::setPlayerChoice(int pChoice) {
        this->choice = static_cast<Choice> (pChoice);
    }

    int Player::getScore() {
        return score;
    }

    void Player::incrementScore() {
        score++;
    }

//Game.h
#pragma once
#include "Player.h"
#include <iostream>
#include <string>

class Game {
public:
    Game();
    ~Game();
    void play();

private:
    Player player1Name;
    Player player2Name;
    void takeTurns();
    int evaluateChoices(Player& p1, Player& p2);
    string getChoiceName(Choice choice) const;
    int announceWinner(Player& p1, Player& p2);
};

//Game.cpp
#include "Game.h"

Game::Game() : player1Name("Player 1"), player2Name("Player 2") {}

Game::~~Game()
{
}

void Game::takeTurns() {
    for (int i = 0; i < 3; ++i) {
        int choice = 0;
        cout << player1Name.getName() << ", choose (1)Rock (2)Paper (3)Scissors: ";
        cin >> choice;
    }
}

```

```

player1Name.setPlayerChoice(choice);
cout << player2Name.getName() << ", choose (1)Rock (2)Paper (3)Scissors: ";
cin >> choice;
player2Name.setPlayerChoice(choice);

int result = evaluateChoices(player1Name, player2Name);

if (result == 0) {
    cout << "It's a tie!" << endl;
}
else if (result == 1) {
    cout << player1Name.getName() << " wins the round!" << endl;
    player1Name.incrementScore();
}
else {
    cout << player2Name.getName() << " wins the round!" << endl;
    player2Name.incrementScore();
}

cout << player1Name.getName() << " chose " <<
getChoiceName(player1Name.getPlayerChoice()) << " and scored: " << player1Name.getScore() <<
endl;
cout << player2Name.getName() << " chose " <<
getChoiceName(player2Name.getPlayerChoice()) << " and scored: " << player2Name.getScore() <<
endl;
}
}

int Game::evaluateChoices(Player& p1, Player& p2) {
    Choice c1 = static_cast<Choice>(p1.getPlayerChoice());
    Choice c2 = static_cast<Choice>(p2.getPlayerChoice());

    if (c1 == c2) {
        return 0;
    }

    if (c1 == ROCK && c2 == SCISSORS) {
        return 1;
    }
    else if (c1 == PAPER && c2 == ROCK) {
        return 1;
    }
    else if (c1 == SCISSORS && c2 == PAPER) {
        return 1;
    }
    else {
        return 2;
    }
}

int Game::announceWinner(Player& p1, Player& p2) {

```

```

Choice score1 = static_cast<Choice>(p1.getScore());
Choice score2 = static_cast<Choice>(p2.getScore());

if (score1 == score2) {
    cout << player1Name.getName() << " scored: " << player1Name.getScore() << " and " <<
player2Name.getName() << " scored: " << player2Name.getScore() << " so this game is a draw" <<
endl;
    return 0;
}
else if (score1 > score2) {
    cout << player1Name.getName() << " won the game and scored: " << player1Name.getScore()
<< endl;
    return 1;
}
else {
    cout << player2Name.getName() << " won the game and scored: " << player2Name.getScore()
<< endl;
    return 2;
}
}

std::string Game::getChoiceName(Choice choice) const {
    switch (choice) {
        case ROCK:
            return "Rock";
        case PAPER:
            return "Paper";
        case SCISSORS:
            return "Scissors";
        default:
            return "Invalid Choice";
    }
}

void Game::play() {
    cout << "Welcome to Rock Paper Scissors game!" << std::endl;
    cout << "Enter Player 1 name: ";
    string name;
    cin >> name;
    player1Name = Player(name);
    cout << "Enter Player 2 name: ";
    cin >> name;
    player2Name = Player(name);
    takeTurns();
    announceWinner(player1Name, player2Name);
    cout << "Thanks for playing!" << std::endl;
}

//Source.cpp
#include <iostream>

```

```
#include <string>
#include <cstdlib>
#include <ctime>
#include "Game.h"
```

```
using namespace std;
```

```
int main() {
    Game game;
    game.play();
    return 0;
}
```

Sources

Dr. Tim Bashford's lecture notes and code.

report2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

62/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11