Rock-Paper-Scissors C++ Game

Date: August-2025

Table of Contents

Introduction....2
- Document Purpose....2
- Tools used to build the project....2

Project Requirements and Explanation....3
- project requirements....3
- game logic....3

Code Explanation....4
- Enums....4
- Structs....4
- Functions....4

Project Behavior....6

Thing I learned....6

References and Links....10

Contact Me....10

Introduction

Document Purpose

The main purpose of this document is to end and finish the Rock-Paper-Scissors C++ Game. I put here the details, explanations, ideas and everything related to this project. I built this project during my study of programming basics at ProgrammingAdvices.com. I want to deepen my understanding of the project by explaining it.

I have benefited a lot from seeing other programmers' work, so I want to share my way of building this game and let others test and see it. With this document I want to market to myself, prove that I built the project, and create a new product to add as a history item in my software development journey.

I also want to take this as an opportunity to improve my writing skills.

Tools used to build the project

This game is simple and doesn't need a lot of tools. I have used Visual Studio and the C++ language to build it.

Project Requirements and Explanation

You can see this video (https://youtu.be/Fj_XoAX2zxU) for a better understanding of the game.

- You will play against the computer.
- You both have 3 choices : Rock, Paper, or Scissors.
- The computer's choice will be generated randomly.

project requirements

- Ask how many rounds the game will be.
- Start each round Player vs Computer.
- Show the results for each round.
- If the computer wins the round, ring a bell, and make the screen red.
- If the player wins the round, show a green screen.
- After all rounds, show "Game Over", and print game results, then ask the user if they want to play again?

game logic

If both players choose the same option, like Rock-Rock, then we consider that nobody wins in this round.

Player_01 choice	Player_02 choice	Winner
Rock	Paper	player_02
Rock	Scissors	player_01
Paper	Scissors	player_02

Code Explanation

Enums

`enGameTools`: defines the choices (Rock-Paper-Scissors).

`enWinner`: defines the winner, whether it is the user, the computer(the bot), or if there is no winner.

Structs

`stRoundInfo`: this structure collects round info (user/bot choices and the round winner).

Functions

1. **`GetGameChoice`**: this function takes as input a number and returns the right Enum item. I use this to convert the user input number to Enum, as dealing with Enum is much easier than dealing with a lot of numbers.

   you can do the following instead of using this function

```
// example of using the funciton
GetGetGameChoice(1);

// equivalent code, using casting
enGameTools choice = (enGameTools)1;
```

2. **`GetChoiceAsString`** and **`GetWinnerAsString`**: both functions take an Enum variable and return the name as String. For example if the input is `enGameTools::Paper`, the output will be "Paper". These two functions do the same work, but I wrote them in two different ways: the first one uses an array, and the second uses a switch statement. You can check the source code to see them (https://github.com/YasinHamad/Rock_Paper_Scissors_Cpp_Game).

3. **`ReadUserChoice`**: keeps asking the user until they enter a valid number (1,2 or 3), as each number refers to a choice, for example, 2→Paper.

4. **`GetWinner`**: takes as input the user and bot winning times, and returns the winner as Enum.

5. **`RandomNumber`**: takes as input two numbers, uses rand() to generate a random number based on these two numbers.

6. **`GenerateBotChoice`**: this function generates a random number using the previous function, converts it to `enGameTools` using function number 1, and returns it.

7. **`GetRoundResult`**: this function implements the game logic. It takes the player choices and returns the winner as `enWinner`.
8. **`CreateARound`**: takes as input the user choice and an `stRoundInfo` variable(as reference) to fill it using the previous two functions.
9. **`AddElementToArray`**: this function takes an element and an array with its size. It adds the element to the array and increments the size by 1.
10. **`ChangeScreenColor`**: takes as input an `enWinner` variable and changes the screen color based on it.
11. **`CountPlayerTries`**: takes as input the player(user/bot) and an array of `stRoundInfo`. It counts the number of winning rounds for this player.
12. **`PrintRoundResults`**: changes the screen color using function number 10, and prints on the screen the round info(results) on the screen in a formatted way. It uses functions from number 2.
13. **`StartRounds`**: takes as input the numbers of rounds the user chose, and array of **`stRoundInfo`** with its size(by reference) to fill it. It makes x rounds based on the number of rounds the user chose. In each round it calls function number 3 to read the user choice, function number 8 to create a new round and save it in a local variable, function number 12 to print the results of the current round, and function number 9 to add the round to the array.
14. **`ReadNumInRange`**: takes as input a message and two numbers, it keeps asking the user until they enter a valid number between these two numbers.
15. **`PrintGameResults`**: prints the "Game Over" sentence on the screen with the game final results. It uses function number 11 to count players winning times, function number 4 to get the winner, and function number 2 to convert it to a string. At the end, it uses function number 10 to change the screen color based on the results.
16. **`StartGame`**: this function starts the game. It uses a do-while to keep asking the user if they want to play again. It uses function number 14 to read the number of rounds from the user, function number 13 to let the user play the rounds, and function number 15 to print game results.

Project Behavior

As you can see the game implementation follows the functional programming approach.
The program starts by calling `StartGame`. This function keeps restarting the game until the user enters something other than the letter 'y' or 'Y'.

When the game starts, it asks the user for the number of rounds they want to play. Then it lets the user play the rounds with the computer.

In each round, the program reads the user's choice and generates a random choice for the computer. Then it decides which one is the winner and stores this data in a struct.

After finishing each round, it prints the data and makes the proper changes on the screen, like changing the color and ringing the bell if needed.
Finally it adds this struct to the array. At the end, the program prints the results based on the array we have.

Thing I learned

This image from the project course page on ProgrammingAdvices.com shows how many programming ideas this game covers.

Concept	Programming Problem Covered
1 Generating Random Numbers	Generate a random number within a range (`RandomNumber()`).
2 Handling User Input	Read and validate user input (Rock-Paper-Scissors choices).
3 Working with Enums	Define and use `enum` types for game choices and winners.
4 Conditional Logic	Implement decision-making using `if` and `switch-case` .
5 Using Structures (Structs)	Store structured game data (`stRoundInfo` , `stGameResults`).
6 String Manipulation	Convert `enum` values into readable strings (`ChoiceName()` , `WinnerName()`).
7 Looping (For & While Loops)	Iterate through multiple rounds using loops.
8 Creating Functions	Write reusable functions to handle various tasks (`ReadPlayer1Choice()` , `PlayGame()`).
9 Generating Computer Choices	Simulate AI behavior by randomly selecting a move.
10 Comparing Values	Determine winners based on user vs. computer choices.
11 Game Loop & Replay Feature	Allow the user to play multiple rounds with replay functionality.
12 Clearing the Screen (System Calls)	Use `system("cls")` to refresh the screen between games.
13 Tracking Scores	Store and update player wins, computer wins, and draws.
14 Working with Boolean Logic	Compare values to determine if the game should continue.
15 Reading and Returning Values from Functions	Use `return` values for `WhoWonTheRound()` , `WhoWonTheGame()` .
16 Switch-Case for Decision Making	Implement a structured approach to evaluating game logic.
17 Using `srand(time(NULL))` for Unique Random Numbers	Ensure different results in each game.
18 Handling Edge Cases	Validate user input to prevent invalid choices.
19 Implementing a Simple AI (Computer Player)	Generate non-human opponent decisions.
20 Debugging & Output Formatting	Print structured results with clear formatting (`Pri...undResults()`).

For me, I have learned some new things that I want to highlight:

1. To get a char based on a number:

```
char(ascii_number); // this function converts the number to char

// or you can do this

char c = ascii_number;
```

2. You can do this in cpp:

```
string word = "";
word += 'a';
```

3. To get a random number in cpp

```
// add this library, to use rand() and srand()

#include <cstdlib>

// add this library to use time()

#include <ctime>


// put this in main, you need to call it only once

// this tells cpp to generate the random number based on the time, so
we can get a new number each time

srand((unsigned)time(NULL));


// rand() will give you a number btw 0 and MAX_INT

cout << rand() << endl;

// you can get the rightmost digit like this. Now you have a number
btw 0 and 9

cout << rand()%10 << endl;


// you can use this. If you put RandomNumber(1, 10) then 1 and 10 will
be included
int RandomNumber(int From, int To)

{

int randNum = rand() % (To - From + 1) + From;

return randNum;
```

```
}
```

4. Writing many functions is not a crime, it is the right approach in functional programming.
5. I included Enum in my work, which I did not study in C at University.

References and Links

Source code: https://github.com/YasinHamad/Rock_Paper_Scissors_Cpp_Game

Academy Website: https://programmingadvices.com

My video on YouTube about this game: https://youtu.be/Fj_XoAX2zxU

Contact Me

Email: yasinhamadbusiness@gmail.com

My GitHub: https://github.com/YasinHamad