
**Master CORO M1**
**Master Commande et Robotique**
**Master in Control and Robotics**

**Project Report**
July 23, 2023

# Bulls & Cows

**Sakthi Vikneswar**
**Anna Hauschild**

**Supervisor**

Olivier Kermorgant, CORO

**Abstract**

This report provides an overview of the Bulls and Cows project, which aims to develop an AI-powered game. In this game the players have to strategically guess the opponents secret number and receive feedback based on which they provide their guesses. The project involves the utilization of C++ and Python programming languages, with communication between the two implemented through the use of messages. The report outlines the project's objectives, challenges faced, strategies employed, and implementations.

# Contents

**1 Introduction** **3**

**2 Project Overview** **3**
2.1 System Architecture . . . . . 4
2.2 Messages . . . . . 4

**3 Implementation** **4**
3.1 Player Inputs and Game Feedback . . . . . 5
3.2 Game State . . . . . 5
3.3 GUI Development . . . . . 5
3.3.1 First attemp . . . . . 5
3.3.2 Components . . . . . 6
3.3.3 Coloring . . . . . 7
3.3.4 Design . . . . . 7
3.4 AI Integration . . . . . 9
3.4.1 AI level 2 . . . . . 9
3.4.2 Example . . . . . 10
3.4.3 Brut-Force algorithm . . . . . 12

**4 Conclusion** **12**

# 1 Introduction

The Bulls and Cows game is a popular logical deduction game where players attempt to guess a secret number. In this game, a 6-digit secret number is generated, consisting of unique digits from 0 to 9. The objective of the player is to guess this secret number first.

With each guess, the player receives feedback in the form of "bulls" and "cows." A "bull" represents a correct digit in the correct position, while a "cow" represents a correct digit in the wrong position. The player can use this feedback to narrow down the possibilities and make more accurate guesses in subsequent attempts.

This project aims to develop an AI-powered Bulls and Cows game. The objective is to create essential components for the game, including the guess, feedback, game mechanics, a GUI, and a basic AI algorithm.

# 2 Project Overview

At the beginning of the project, a framework of the three following programs was provided:

1. The server: it functions as the central game and should include the AI. It offers the option of multiple difficulty levels.
2. The client: it represents the player's code, which receives updates about the game state and interacts by providing input.
3. The display: it is written in Python and runs on the user computer. It receives display data from the game but does not send anything.

To create the game template a YAML file with different types of messages is defined.

1. init_display: This message contains the initial information that the game sends upon initialization.
2. input: It represents the expected input from the player.
3. feedback: This message conveys the feedback from the game to a player based on their input.
4. display: This message includes the necessary data for displaying the game state, but it is not accessible to the player directly.

During the project this framework was used to implement the bulls and cows game, include an algorithm to solve it and display the the game state on a GUI.

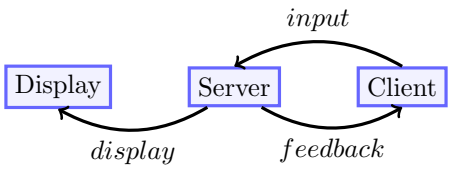



Figure 1: programs and messages

## 2.1 System Architecture

The game process is managed by the server, which communicates with the client and the display using messages, see Figure 1. Inside the server, the rules of the game are defined and applied. The client, on the other hand, provides its secret number during the first turn and submits guesses in subsequent turns. Each turn involves the server calculating the number of bulls and cows and determining the game state. The game state is then sent to the display for visualization purposes, while the number of bulls and cows is relayed to the player as feedback. Based on this information, the client generates a new guess, which is sent back to the server as input for the next turn.

## 2.2 Messages

The messages serve as an exchange format between the two languages and contain information to be transmitted from one language to the other. By using a unified message structure like the YAML file, C++ and Python can communicate with each other and exchange data. The message structure is typically defined in advance and made known in both languages. This allows C++ and Python to interpret and understand the information in the messages in the same way. The data transmitted from the game to the display are the two guesses from the players. After a guess was made it is stored as a message and sent to the display. On the Python side, the received display messages are accessed using the message object. The functions serialize and deserialize are used to convert the messages between C++ and Python. The message acts as a communication bridge between the the C++ logic and the Python display.

# 3 Implementation

The implementation of the Bulls and Cows game involves the development of several components, including player inputs, game feedback mechanics, game state changes, GUI development, and AI integration. This section provides an overview of each component and the strategies employed for their implementation.

## 3.1 Player Inputs and Game Feedback

The player inputs and game feedback form the core interaction between the AI and the game server. The players receive their feedbacks, the bulls and cows, from the server, make guesses, and transmit the guesses back to the server. The server, in turn, calculates the number of bulls and cows based on the guesses and provides the feedback to the players.

## 3.2 Game State

The outcome of the game, which can result in a win, loss, or draw, is determined by the game rules implemented within the server. The game concludes if either one player (resulting in a win) or both players (resulting in a draw) find the six bulls. Additionally, if a player provides an invalid guess, the game ends. Otherwise, if none of these conditions are met, the game continues and the state remains as "none."

## 3.3 GUI Development

To provide a user-friendly display for the game, a graphical user interface (GUI) is developed using the Pygame library in Python. The GUI receives display data from the game server and presents it to the user. It visually represents the game state, including the current guess, the number of bulls and cows, and any additional information necessary for gameplay. The GUI enhances the overall game and allows the AI's progress to be easily tracked.

### 3.3.1 First attemp

The first part involves creating an initial window and establishing a connection between Python and C++. This step sets an important part for the project by creating a basic user interface and enabling communication between the two programming languages. After studying the functionality of the messages, see chapter 2.2, and getting familliar with the pygame library following first User Interface, seen in Figure 2, was created.

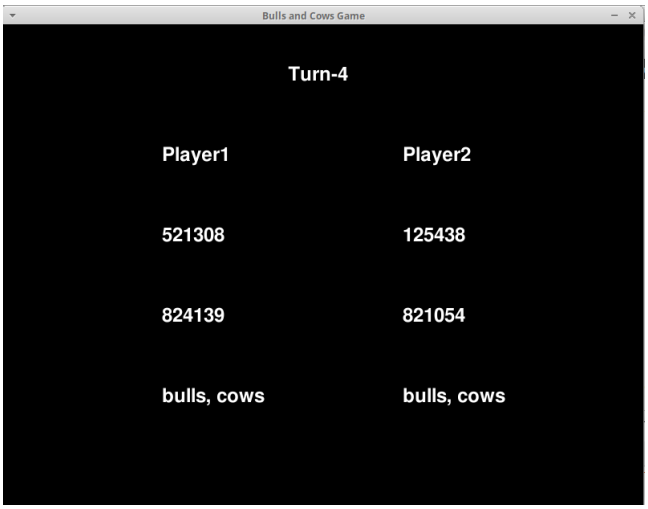



Figure 2: First attemp

### 3.3.2 Components

In the subsequent phase, all essential components for the display were created. Within the game, the only information transmitted to the display was the player's guess. As a result, the display program took on the responsibility of recalculating the feedback, i.e. the number of bulls and cows, internally.

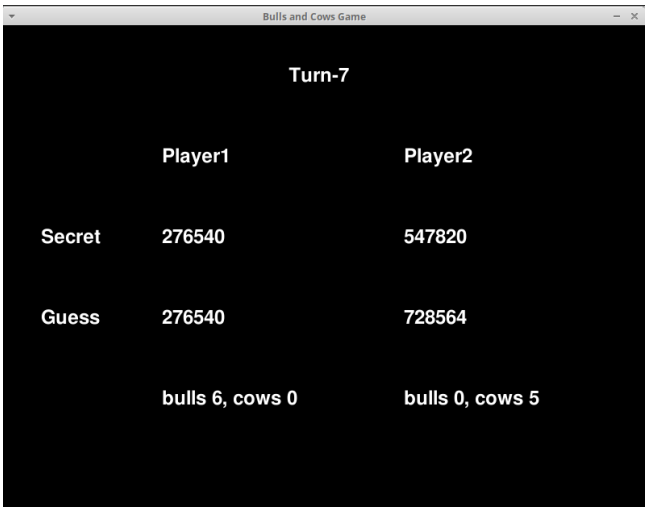



Figure 3: Added the bulls and cows feedback

### 3.3.3 Coloring

To enhance the game display and facilitate tracking, a color scheme was implemented. The scheme provides visual cues for the correctness of each digit in the guess. A digit that matches both in value and position is highlighted in red, indicating a bull. On the other hand, a digit that matches in value but is in the wrong position is highlighted in blue, representing a cow. Additionally, the number of bulls and cows is prominently displayed in corresponding colors, providing a clear and intuitive representation of the game's feedback.

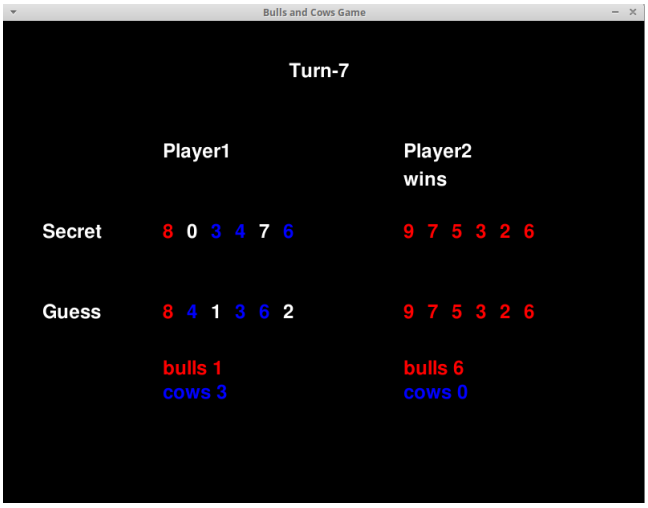



Figure 4: Coloring the bulls and cows

### 3.3.4 Design

In the final step, the game interface was crafted to be aesthetically pleasing and user-friendly.

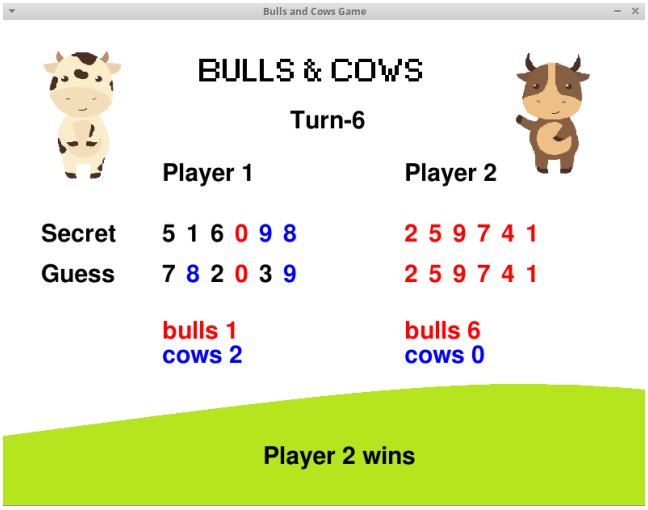



Figure 5: Aesthetic design

## 3.4 AI Integration

The integration of AI into the Bulls and Cows game involves developing different levels of intelligent algorithms that can make informed guesses based on the available information and feedback. Typically, AI level 0 should have no chance to win in order to test other AI's. It consistently repeats the same guess in each turn. AI level 1 may win by chance. It relies on random guesses. In both cases, the AI disregards the feedback provided by the game regarding the correctness of their guesses. Higher difficulty AIs should actually try to win. It continuously adapts its approach based on the received feedback and aims to minimize the number of attempts needed to guess the secret number. In this section the implemented AI level 2 algorithm will be derived and explained.

### 3.4.1 AI level 2

The approach involves creating a list containing all possible candidate numbers for a guess, and then iteratively filtering out inappropriate entries in each round. The idea is based on the available information, which includes the current guess and its corresponding bulls and cows and the fact that the number of bulls and cows remains consistent regardless of the calculation direction, whether from the secret number to the guess or vice versa. Only candidates from the list that share the same quantity of bulls and cows as the current guess are considered, while others are filtered out. From the remaining candidates, one is randomly chosen as the next guess. This process continues in subsequent rounds, gradually narrowing down the list of candidates until the correct guess is found.

The pseudo code of this algorithm is described as follows.

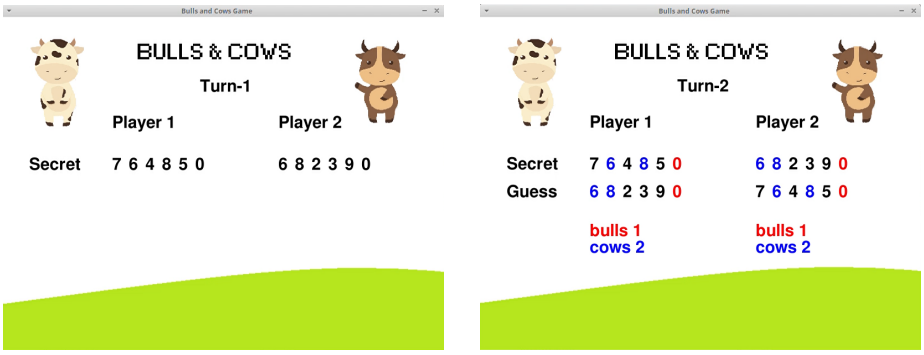
```
function updateInput()
    ...
    if difficulty is 2 then
        if turn is 1 then
            secretNumber = random ∈ [123456, 987654]
            list candidates = all possible numbers
        end if
        if turn is 2 then
            guess1 = secretNumber
        end if
        if turn >= 3 then
            list candidates = remove_if(feedback.guess ! = feedback.candidate)
            new guess = random entry of candidates
        end if
    end if
end function
```

On the first turn, the AI generates it's secret number for the opponent to guess and a list of all possible numbers that become the candidates for the guesses.

On the second turn, the AI uses the generated secret number as its first guess. From the third turn onwards, it removes any numbers that have inconsistent feedback, i.e. different bulls and cows regarding the current guess. The new guess becomes one of the remaining entries.

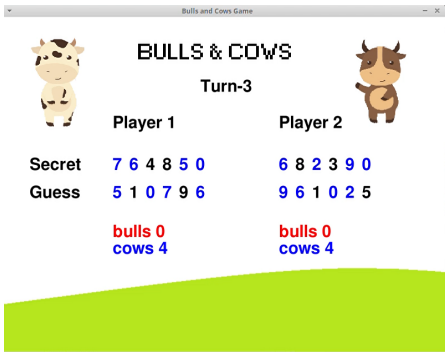
### 3.4.2 Example

In the folowing figure an example of the game is provided. Picture (a) shows turn 1 of the game. Both player provide their secret numbers. Picture (b) shows turn 2 where the players make their first guess which is the their own secret number. Based on the feedback in turn 3 the players make their first guess based on the feedback. Considering turn 3 and turn 4 one can see that the amount of bulls and cows in turn 3 (bulls:0, cows:4) corresponds to the same amount of bulls and cows between the guess in turn 3 and the guess in turn 4. There are 0 bulls and the 4 cows: 6,0,7,1. So the new guess 624071 is one of the remaining candidates of the list of possible guesses. As the game progresses, the number of suitable digits increases. Typically, no more than 8 guesses are needed to find the correct number. Since both players use the same algorithm, it is not surprising that the game ends in a draw.

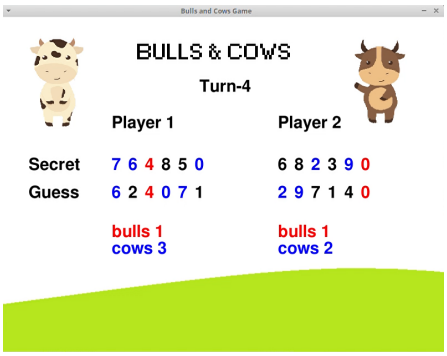



(a) Turn 1 (b) Turn 2

Figure 6: Example AI level 2 Turn 1 and 2

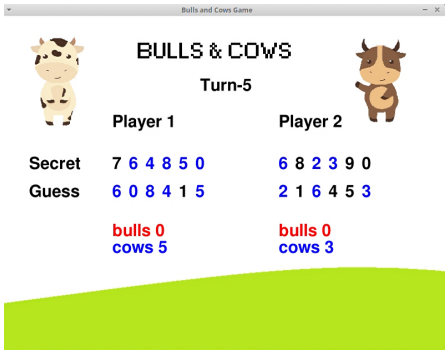



(a) Turn 3

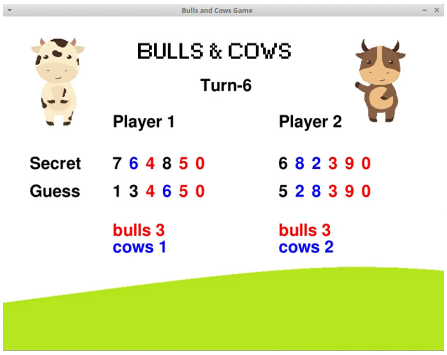



(b) Turn 4

Figure 7: Example Turn 3 and 4

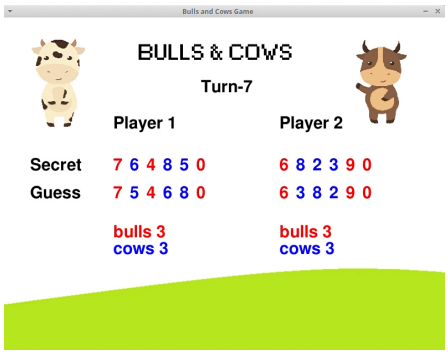



(a) Turn 5

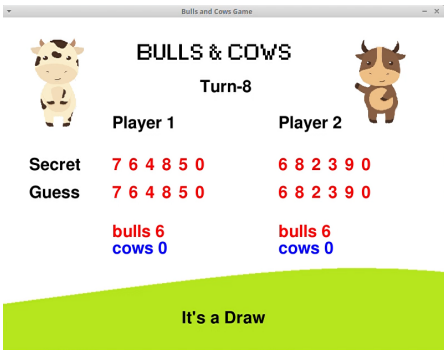



(b) Turn 6

Figure 8: Example Turn 5 and 6




(a) Turn 7




(b) Turn 8

Figure 9: Example Turn 7 and 8

### 3.4.3 Brut-Force algorithm

The algorithm described is referred to as a brute force algorithm. In a brute force approach, all possible combinations are tried one at a time until the right solution is found. In terms of algorithm efficiency, it is impressive that it tries all possibilities in less than a second. The running time of a brute force algorithm often depends on the size of the problem space. Since the Bulls and Cows game has a limited number of digits and a fixed length for the secret number, the problem space is relatively small and therefore quick to search.
It is important to note that brute force algorithm is not always the most efficient solution method, especially for larger problems. In such cases, optimized algorithms or heuristics can be used to speed up the search for the solution. Still, the brute force approach is a solid choice for the Bulls and Cows game as there are a manageable number of possibilities.
To calculate the number of possibilities, the combinations of 6 digits without repetition are considered, where the first digit is not 0 and no digit is repeated.

The number of possibilities for the first digit is 9, as it cannot be 0. For the second digit, there are 9 remaining possibilities (since it cannot be the same as the first digit). For the third digit, there are 8 possibilities available. Similarly, for the remaining digits, there is a decreasing number of possibilities.

Therefore, the total number of possibilities is obtained by multiplying the individual possibilities:

9 * 9 * 8 * 7 * 6 * 5 = 136,080

There are a total of 136,080 different combinations of 6 digits that meet the given restrictions.

# 4 Conclusion

The project provided a great opportunity to improve programming skills. It particularly helped in delving into existing code, understanding it, and using it as a framework, just as the person who wrote it envisioned. It taught the importance of deriving the structure of a function based on its defined scope, with the inner workings being individual and less significant from an external perspective. However, the parameters and return values play a crucial role. When they are clearly defined and remain unchanged, it enables collaboration with other team members in the project. Initially, this posed a significant challenge, but in hindsight, it was clearly a major advantage in understanding the structure in programming.