Bulls & Cows

Sakthi Vikneswar
Anna Hauschild

Master Coro M1
Master in Control and Robotics
Supervisor: Prof. Olivier Kermorgant

Project Presentation

June 26, 2023

Overview

Introduction: Bulls and cows game

- 2-Player code-breaking game
- Goal: guess secret number of opponent
- Guess: **6 8 2 3 9 0**

Introduction: Bulls and cows game

- Feedback in form of bulls and cows, where
  - Bulls: correct digit in correct position
  - Cows: correct digit in wrong position
-

Secret number:	**7**	**6**	**4**	**8**	**5**	**0**
Guess:	**6**	**8**	**2**	**3**	**9**	**0**

  - Bulls $= 1$
  - Cows $= 2$

Introduction: Bulls and cows game

- The guess and secret number
  - Contains 6 digits, no 0 in first position
  - No duplicates
- Outcome
  - Player who guesses first the right number wins
  - Draw, if both player find the secret number in the same turn
  - Invalid guess results in a loss

Project Overview

The framework consists of three programs

- Server
  - Central game
  - Includes game mechanics
- Client
  - Represents player's code
  - Receives feedback
  - Interacts by providing input
- Display
  - Written in python
  - Receives display data
  - Interacts by providing input

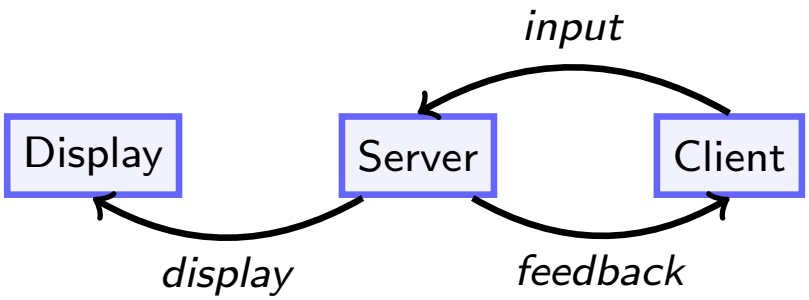

Figure: Programs and messages

Project Overview

These programs communicate with different types of messages

- Input
  - It represents the expected input of the player
- Feedback
  - Conveys the feedback from the game to the player based on the input
- Display
  - Includes the necessary data for displaying the game state
  - Not accessible to the player directly

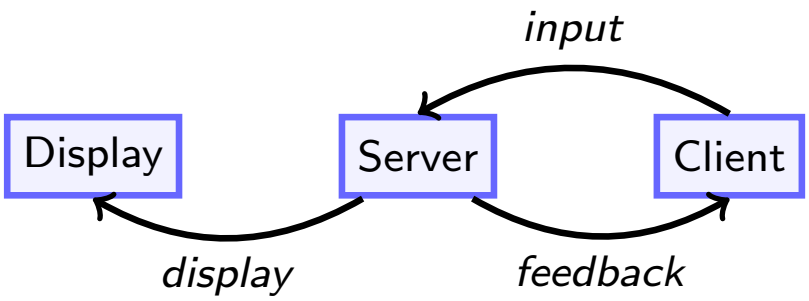

Figure: Programs and messages

Implementation

- Player Inputs
  - Algorithm to compute guess based on provided feedback
- Game Feedback
  - Computation of bulls and cows based on the current guess
- GUI Development
- AI Integration

Duels Framework

- Repository that contains code generation scripts
- Provides a skeletal to create a game based on the interaction between the mentioned nodes

The YAML file

bulls_cows_gui.py bulls_cows.yaml

```yaml
timeout: 1000 #Time given to player for guessing in ms
refresh: 1000 #Display refresh in ms
server_timeout: 15000

init_display: []
display: [int guess1, int guess2]
input: [int guess]
feedback: [int bulls, int cows]

turn_based: False
```

Figure: Yaml file

In Python

```
if (turn == 1):
    #receive message guess
    secret1 = str(msg.guess1)
    secret2 = str(msg.guess2)
```

Figure: Receive guess in python code

GUI Development

- First attempt

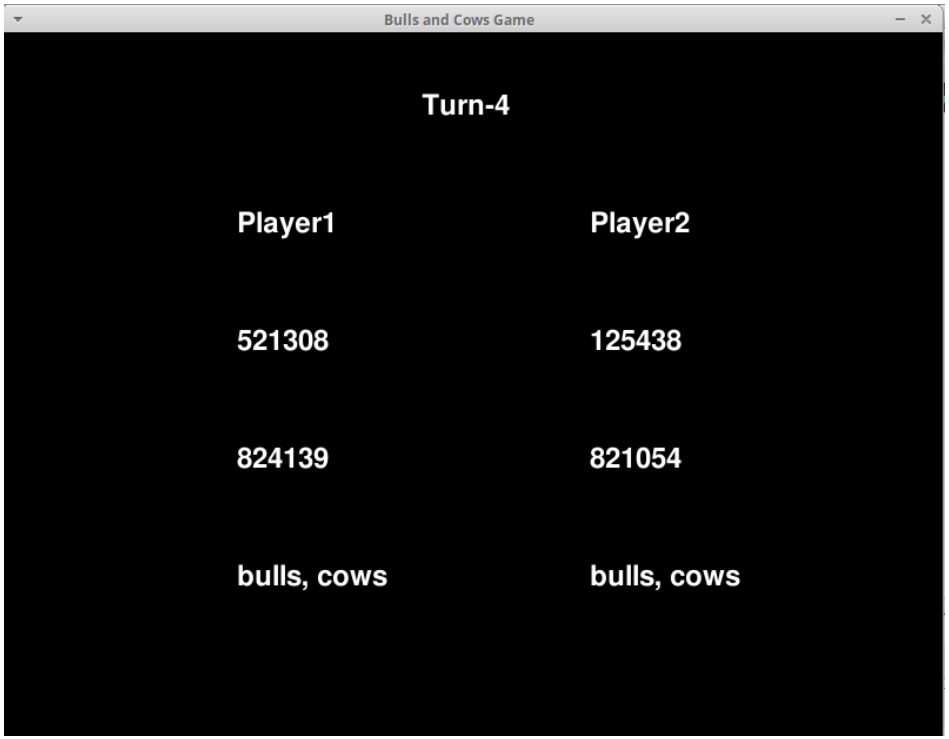

Figure: First attempt

GUI Development - Calculation of bulls and cows

```
#function: compute bulls and calls
def count_bulls_and_cows(secret_str, guess_str):
    bulls = 0
    cows = 0
    for i in range(len(secret_str)):
        if secret_str[i] == guess_str[i]:
            bulls += 1
        elif secret_str[i] in guess_str:
            cows += 1
    return bulls, cows
```

Figure: Calculation of bulls and cows

GUI Development

- Components

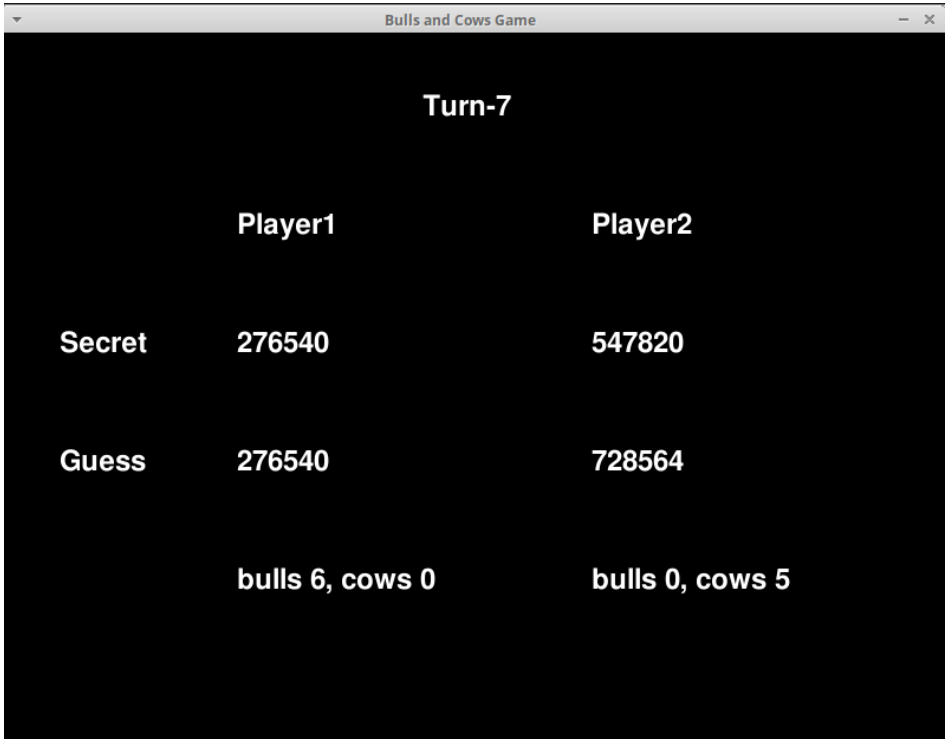

Figure: Added the bulls and cows feedback

GUI Development

- Coloring

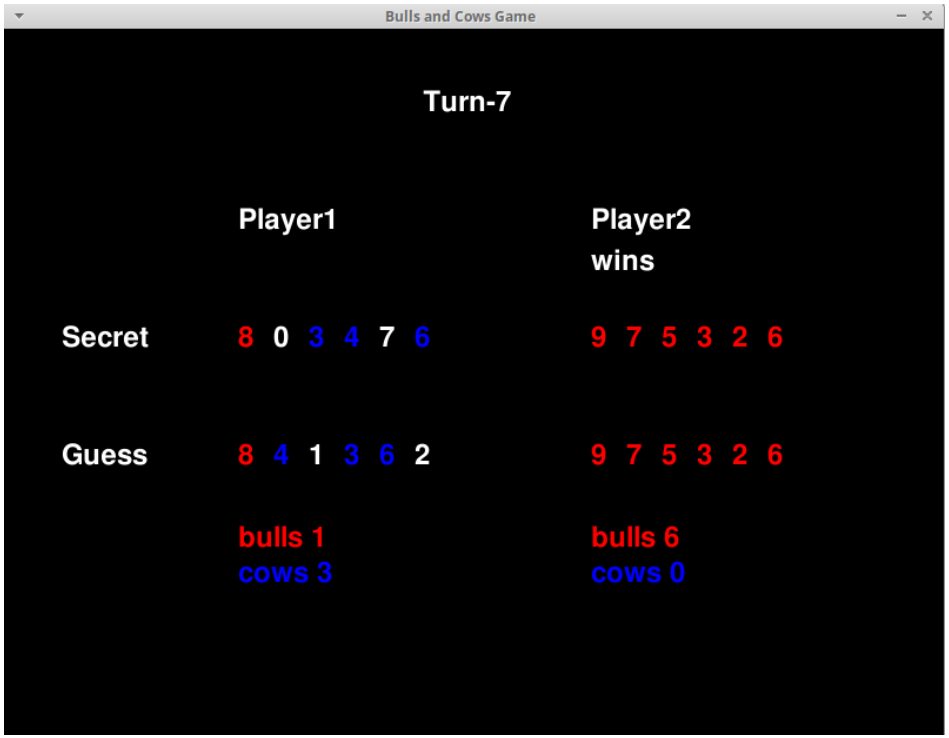

Figure: Coloring the bulls and cows

GUI Development

- Design

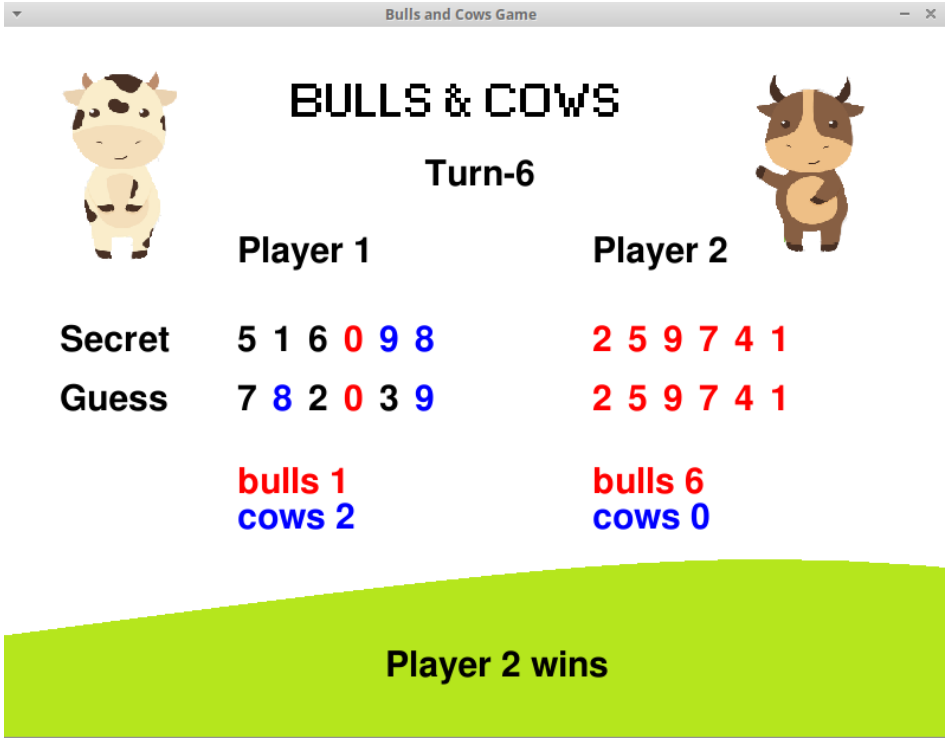

Figure: Aesthetic design

AI Integration

- AI level 0
  - Should have no chance to win
  - Consistently repeats the same guess in each turn
- AI level 1
  - May win by chance
  - Relies on random guesses
- AI level 2
  - Tries to win
  - Adapts its approach based on the received feedback
  - To minimize the number of attempts needed to guess the secret number

AI Integration

- Consider all possibilities
- Calculate bulls and cows for each entry regarding the current guess
- Compare to actual feedback
- Filter the list

1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 5 8
⋮
9 8 7 6 5 3
9 8 7 6 5 4

AI Integration

- AI level 2

```
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
```

AI Integration: Example

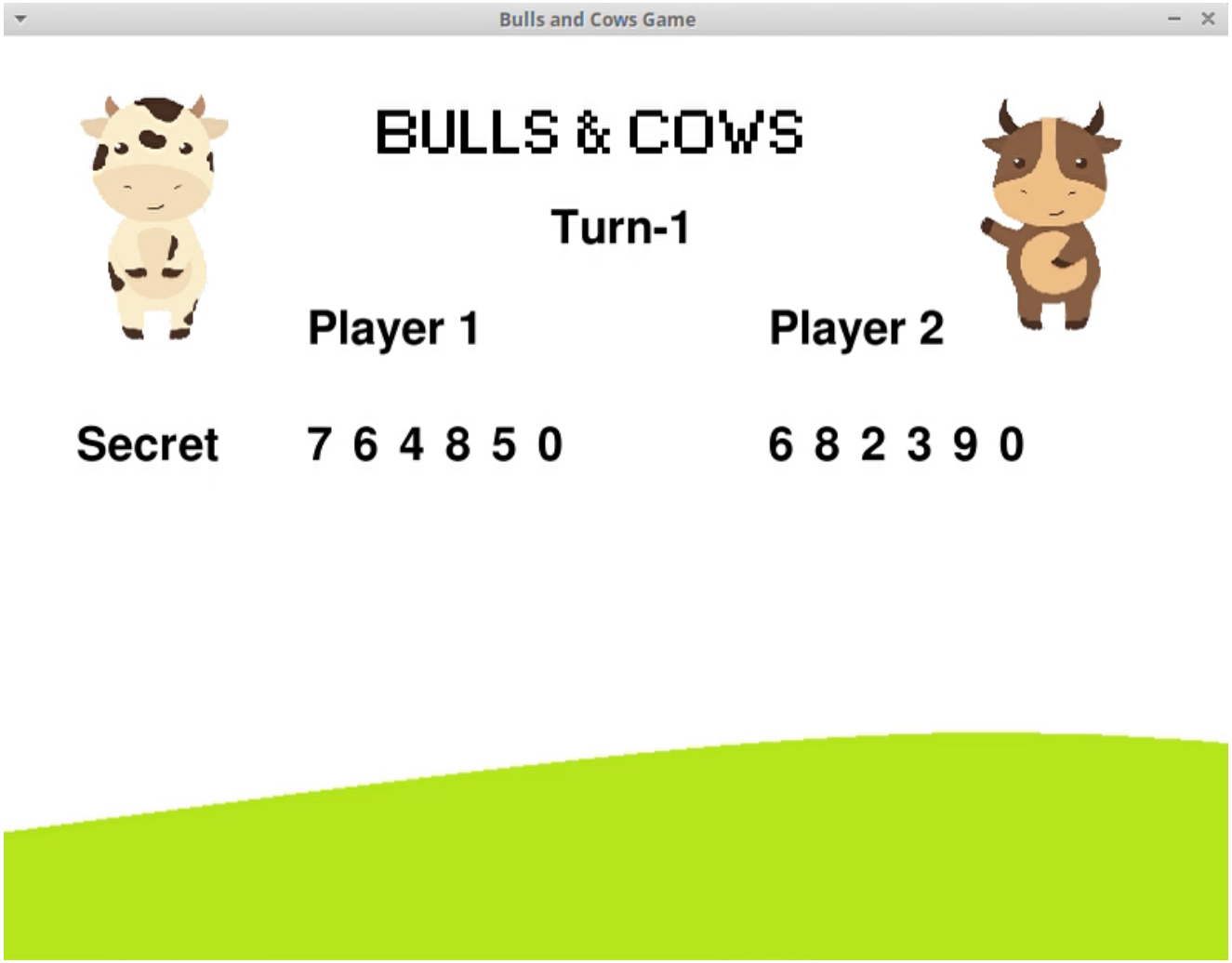

Figure: Turn 1

AI Integration: Example

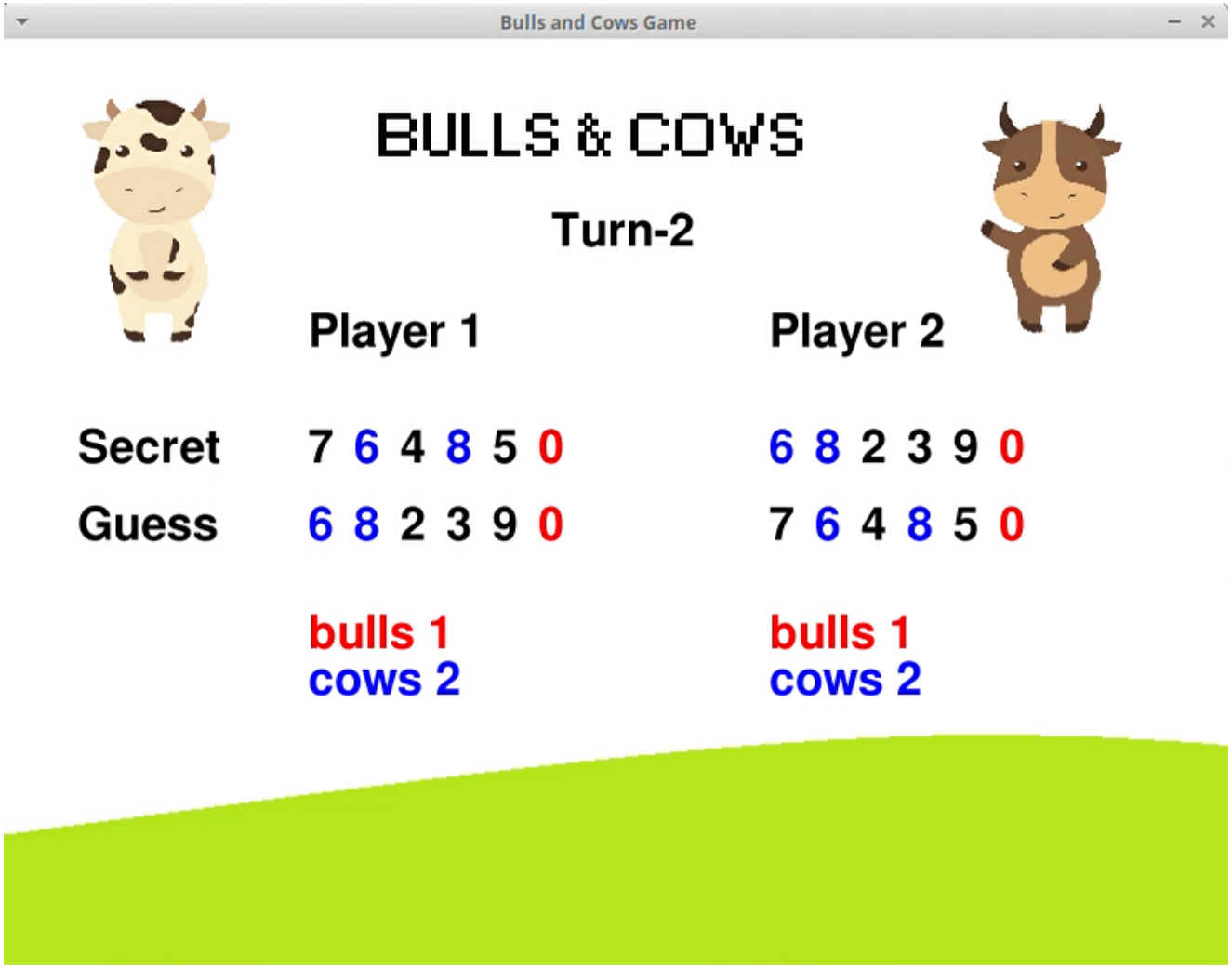

Figure: Turn 2

AI Integration: Example

- Example

1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 5 8
⋮
6 8 2 3 9 0
⋮
7 6 4 8 5 0
⋮
9 8 7 6 5 3
9 8 7 6 5 4

Secret number:	**7**	**6**	**4**	**8**	**5**	**0**
Guess:	**6**	**8**	**2**	**3**	**9**	**0**

Bulls 1

Cows 2

AI Integration: Example

1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 5 8
⋮
6 8 2 3 9 0
⋮
7 6 4 8 5 0
⋮
9 8 7 6 5 3
9 8 7 6 5 4

Secret number:	7	6	4	8	5	0
Guess:	6	8	2	3	9	0
	Bulls 1					
	Cows 2					

Current guess: **6 8 2 3 9 0**

AI Integration: Example

1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 5 8
⋮
6 8 2 3 9 0
⋮
7 6 4 8 5 0
⋮
9 8 7 6 5 3
9 8 7 6 5 4

Secret number:	7	6	4	8	5	0
Guess:	6	8	2	3	9	0

Bulls 1
Cows 2

Current guess:	**6**	**8**	**2**	**3**	**9**	**0**
Candidate:	**1**	**2**	**3**	**4**	**5**	**6**

Bulls 0 $\neq$ Bulls 1
Cows 3 $\neq$ Cows 2

AI Integration: Example

~~1 2 3 4 5 6~~
1 2 3 4 5 7
1 2 3 4 5 8
⋮
6 8 2 3 9 0
⋮
7 6 4 8 5 0
⋮
9 8 7 6 5 3
9 8 7 6 5 4

Secret number:	7	6	4	8	5	0
Guess:	6	8	2	3	9	0

Bulls 1
Cows 2

Current guess:	**6**	**8**	**2**	**3**	**9**	**0**
Candidate:	**1**	**2**	**3**	**4**	**5**	**6**

Bulls 0 $\neq$ Bulls 1
Cows 3 $\neq$ Cows 2

AI Integration: Example

~~1 2 3 4 5 6~~
~~1 2 3 4 5 7~~
~~1 2 3 4 5 8~~
⋮
~~6 8 2 3 9 0~~
⋮
7 6 4 8 5 0
⋮
~~9 8 7 6 5 3~~
9 8 7 6 5 4

Secret number:	7	6	4	8	5	0
Guess:	6	8	2	3	9	0

Bulls 1
Cows 2

Current guess:	**6**	**8**	**2**	**3**	**9**	**0**
Candidate:	**1**	**2**	**3**	**4**	**5**	**6**

Bulls 0 $\neq$ Bulls 1
Cows 3 $\neq$ Cows 2

AI Integration: Example

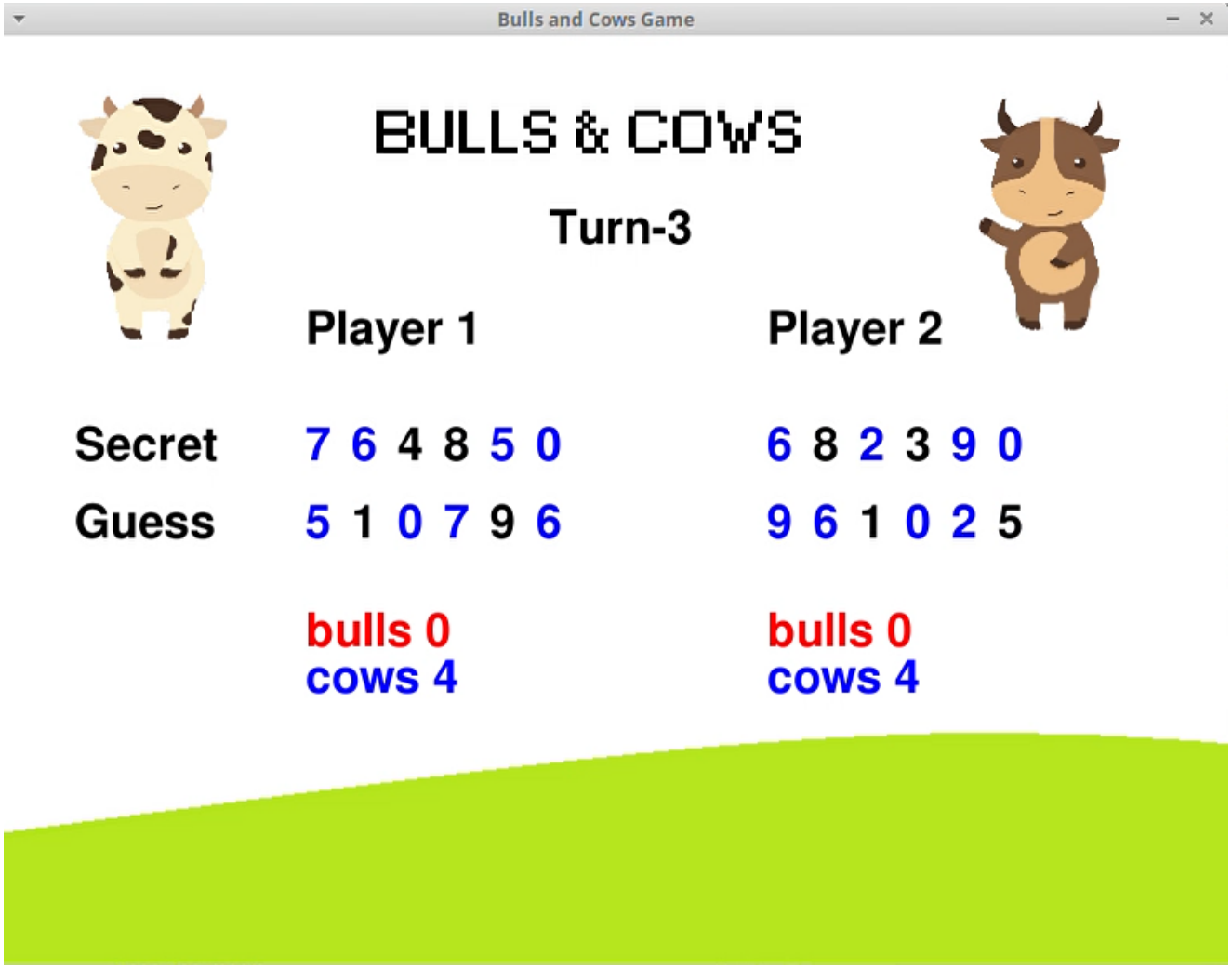

Figure: Turn 3

AI Integration: Example

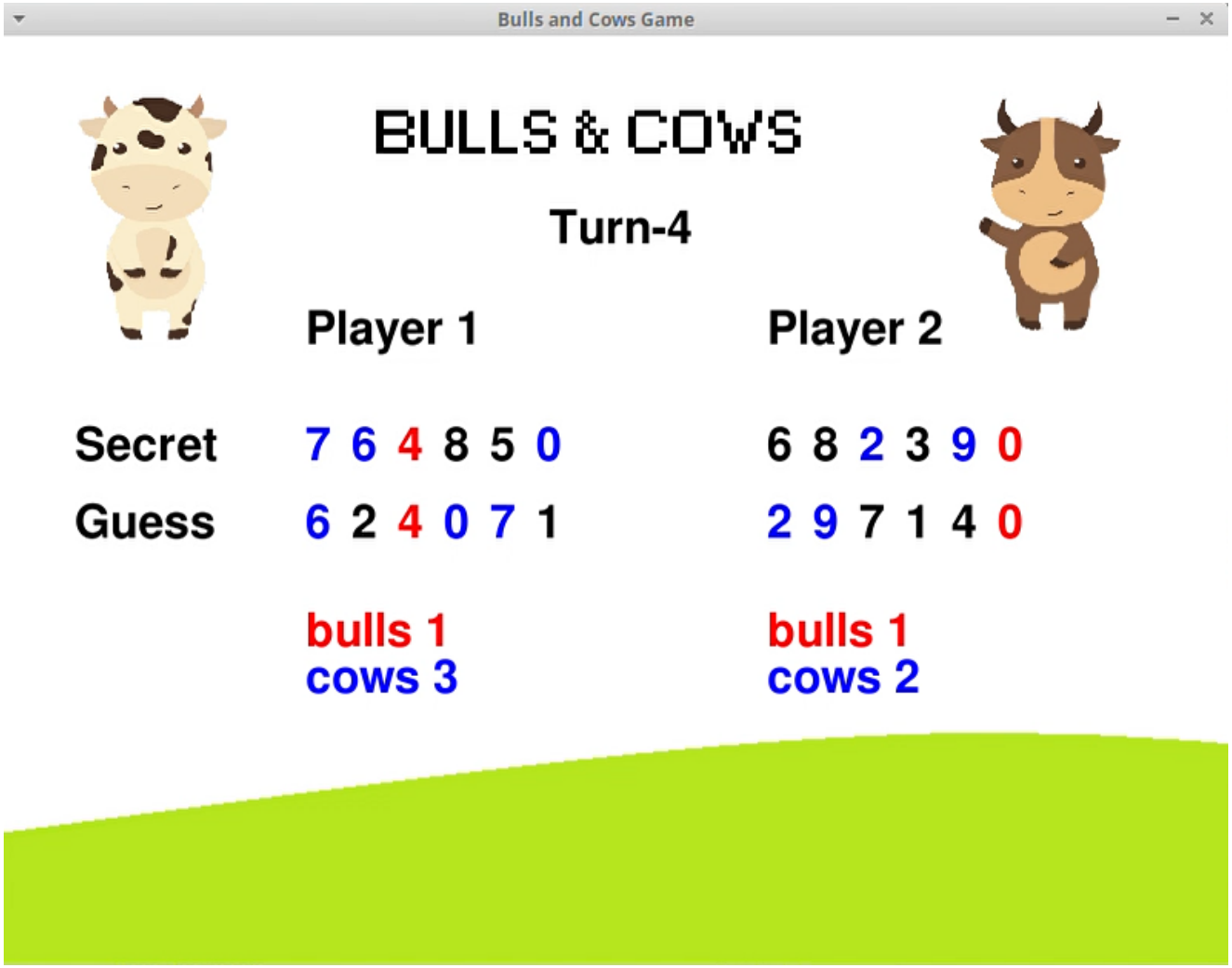

Figure: Turn 4

AI Integration: Example

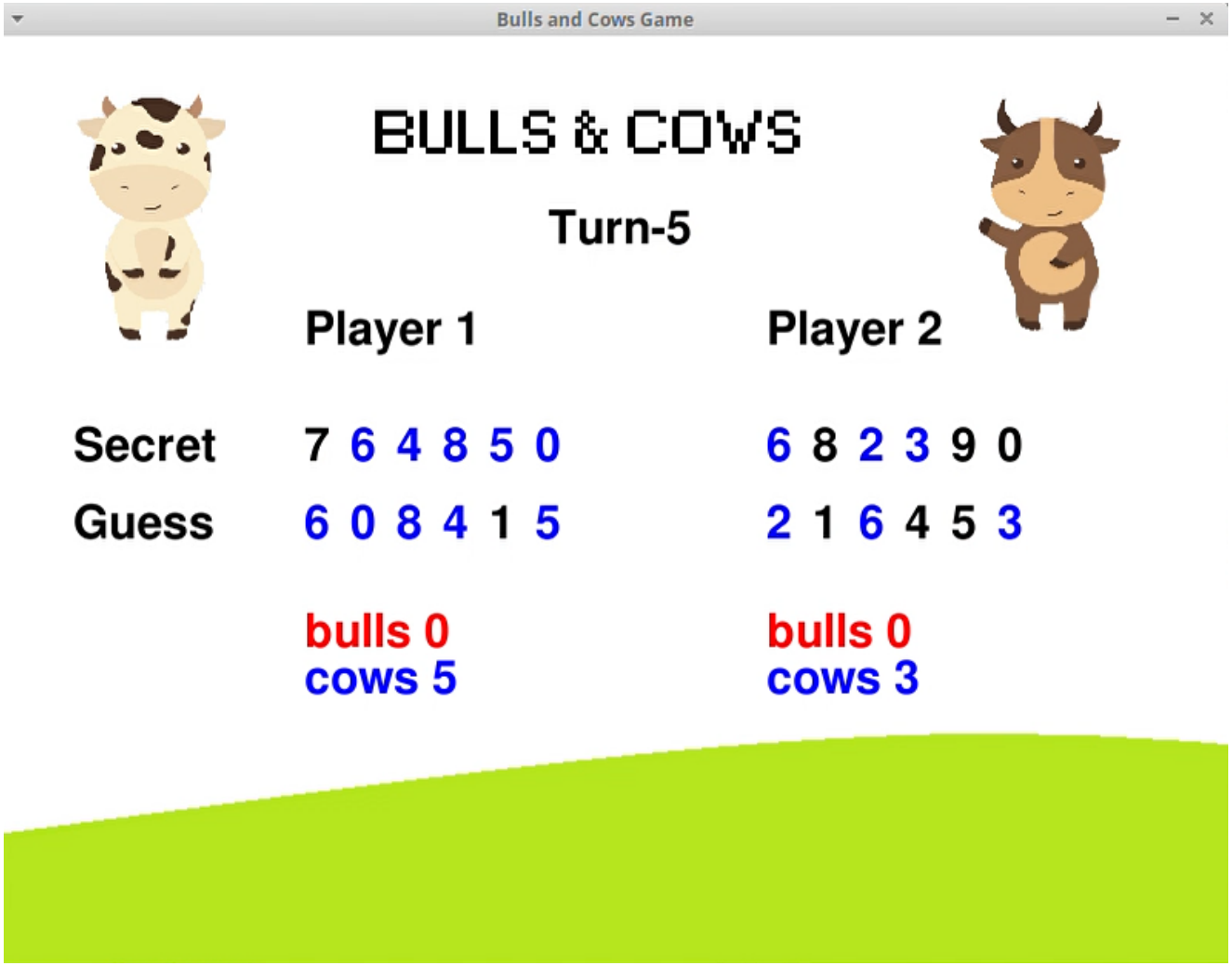

Figure: Turn 5

AI Integration: Example

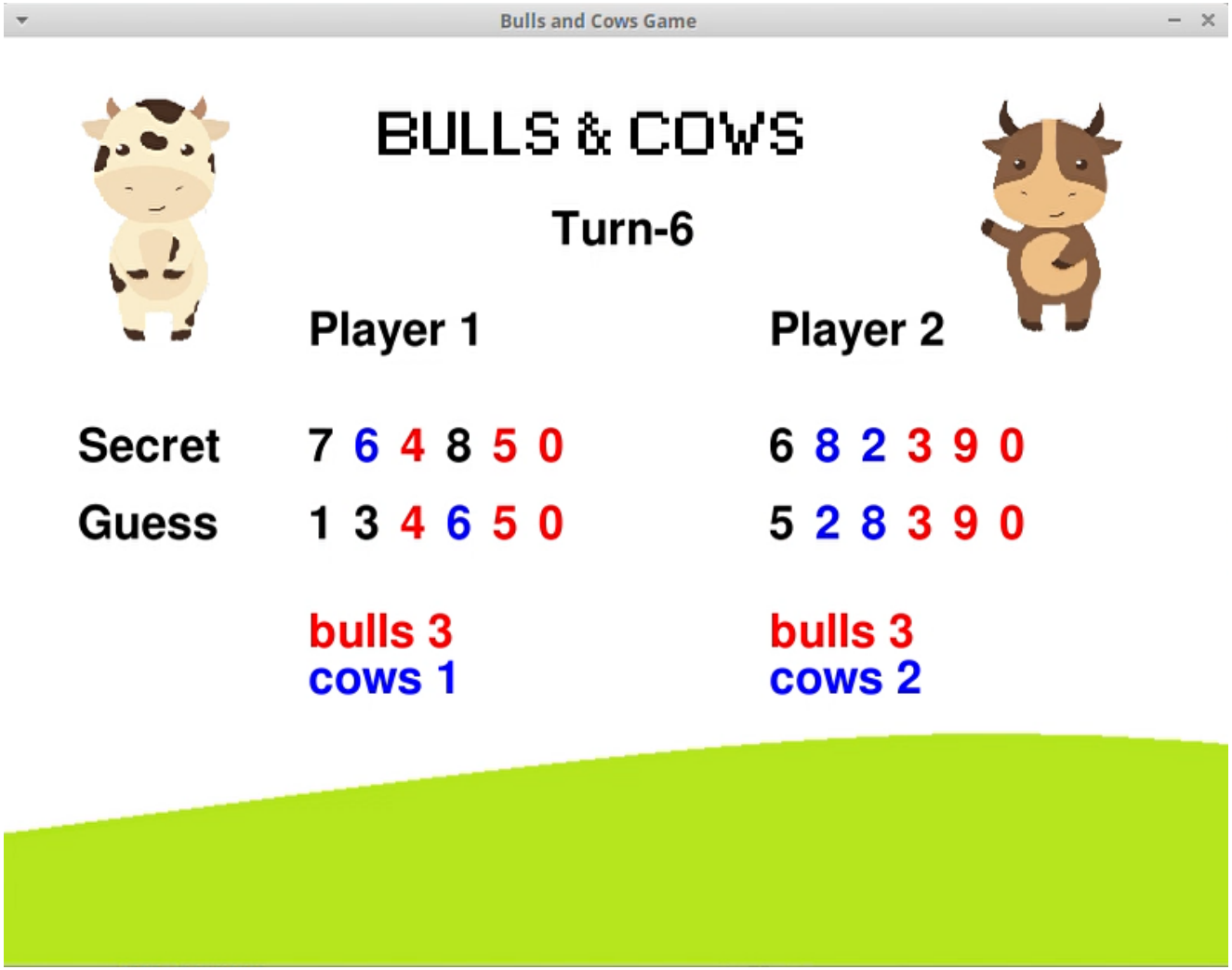

Figure: Turn 6

AI Integration: Example

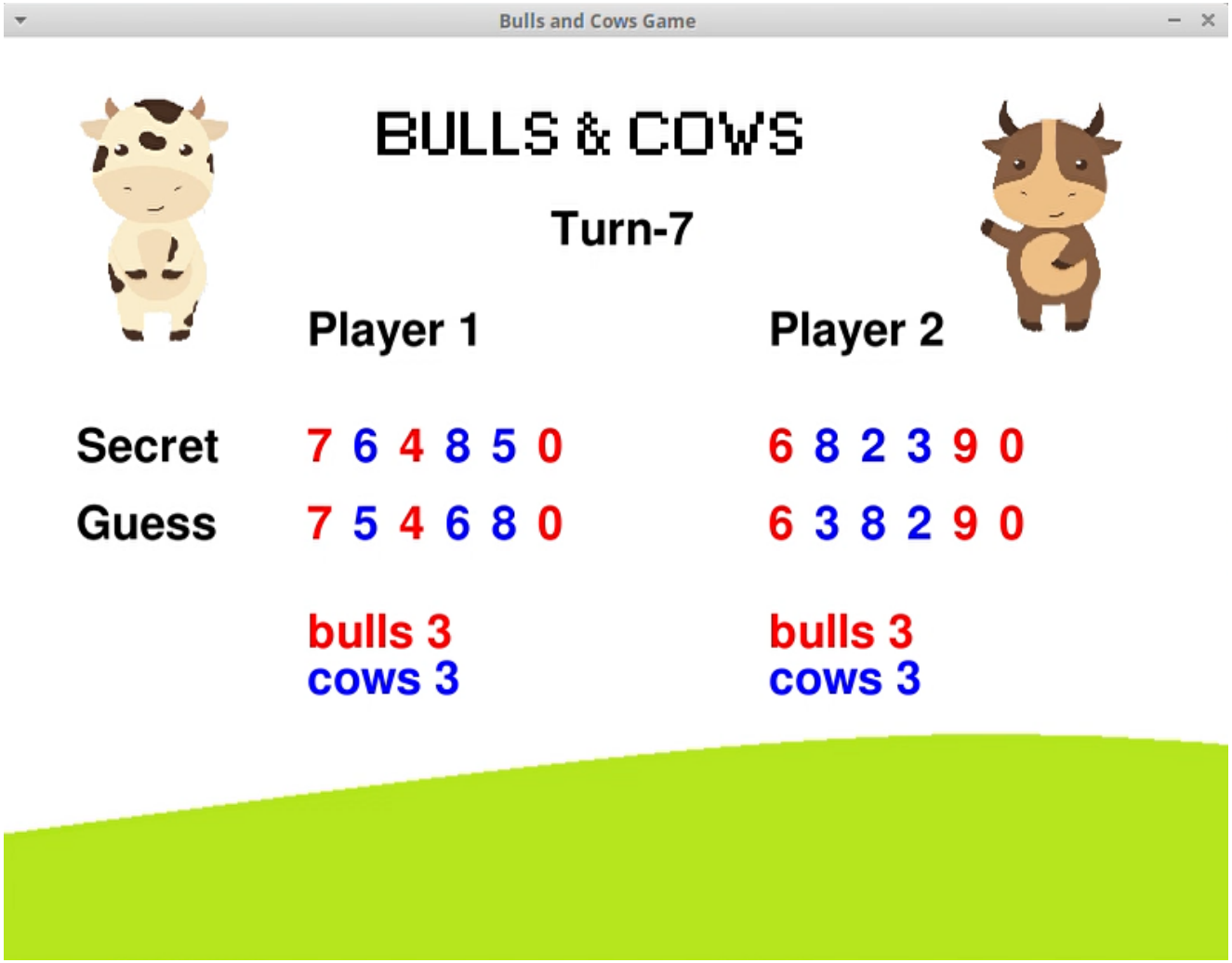

Figure: Turn 7

AI Integration: Example

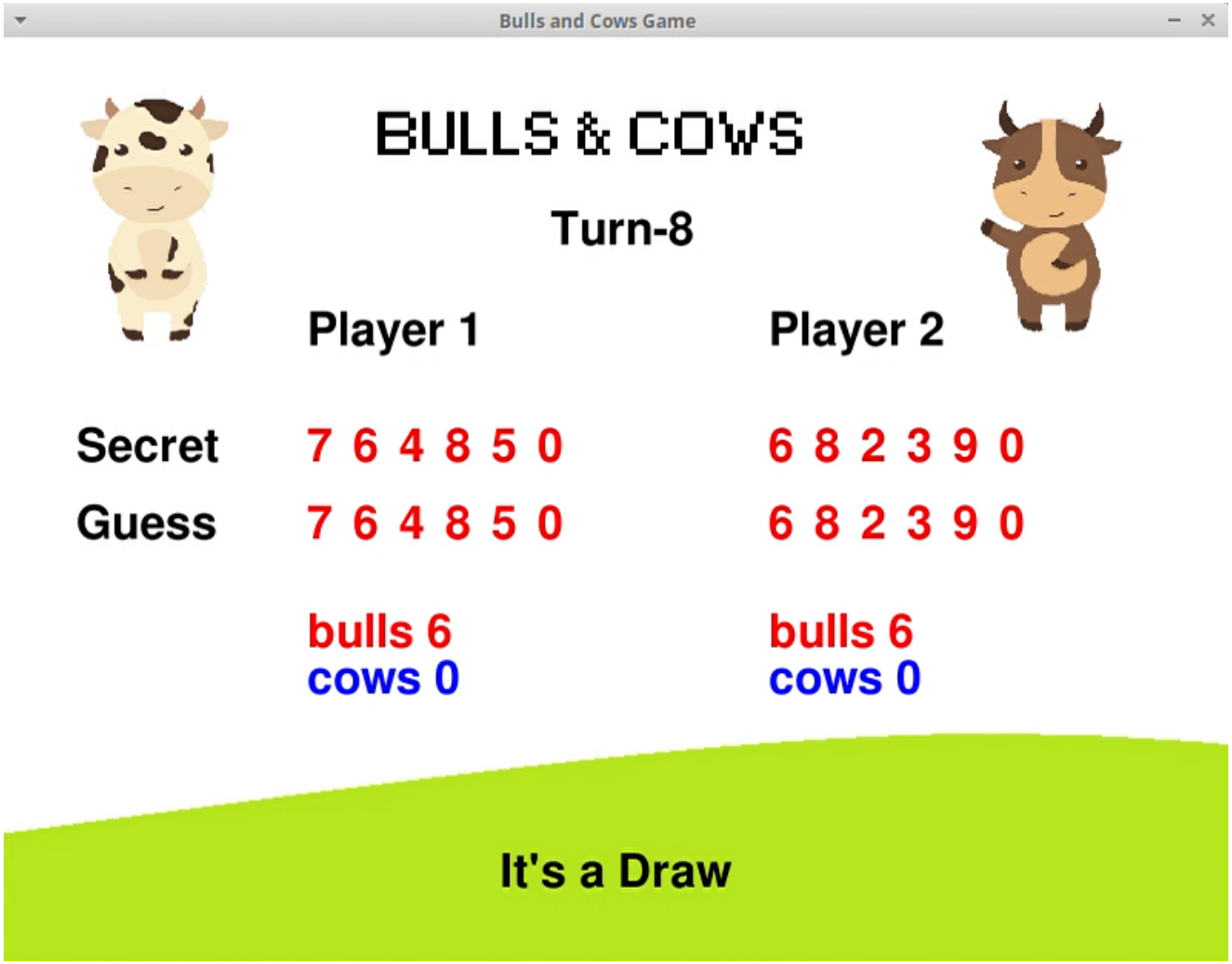

Figure: Turn 8

AI Integration: Brute-Force algorithm

- All possible combinations are tried one at a time until the right solution is found
- Tries all possibilities in less than one second
- Since fixed length of numbers the problem space is small and therefore quick to search
- $9 * 9 * 8 * 7 * 6 * 5 = 136,080$ Possible candidates
- No more than 8 guesses needed

Conclusion

- Great opportunity to improve programming skills
- Familiarize with existing code and adapt to it
- Sticking to the framework was challenging, but it taught a lot about structure
- It enables collaboration with other team members