Kingdom of Saudi Arabia
Ministry of Education
King Faisal University
College of Computer Sciences & Information Technology

Puzzle Game

Using Generative AI Project

Selected Topics in CS

by

Khaled Fahad Alhamad (217028190)

Hosam Sultan Alotibi (**218011513**)

Course Instructor

Dr. Abdulelah Algosaibi

December, 2023

Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 Significance 3

1.3 Target Audience 3

2 Rules of the Game 4

2.1 Game Board 4

2.2 Objective 4

2.3 Winning Condition 4

3 Puzzle Generation Algorithm 5

3.1 Solvability Check 5

3.2 Implementation 5

4 Coding Language and Libraries 5

4.1 Programming Language 5

4.2 GUI Library - Tkinter 6

5 Puzzle Generation 6

5.1 `PuzzleGame` Class 6

5.2 Dynamic Puzzle Resizing 7

6 GUI Implementation 7

6.1 `PuzzleGameUI` Class 7

6.2 Player Interaction 7

6.3 User Feedback 7

7 Conclusion 8

8 References 9

1 Introduction

The "Puzzle Game Using Generative AI" represents a sophisticated blend of algorithmic ingenuity and user interface design, providing a captivating and intellectually stimulating gaming experience. Developed in Python, with the Tkinter library handling graphical user interface intricacies, this puzzle game aims to immerse users in a world of solvable puzzles, each dynamically generated for optimal challenge.

1.1 Purpose

The primary purpose of this game is to offer users an enjoyable and interactive challenge, requiring both strategic thinking and precision in tile manipulation. By implementing a generative algorithm, the game ensures a diverse array of puzzles. This adaptive and dynamic approach aims to captivate players, fostering engagement and repeated gameplay.

1.2 Significance

In an era where digital gaming experiences often prioritize high-end graphics and complex narratives, the "Puzzle Game Using Generative AI" provides a refreshing departure. Focused on the fundamental appeal of puzzles, it aims to stimulate cognitive abilities and problem-solving skills while maintaining a user-friendly interface accessible to all audiences.

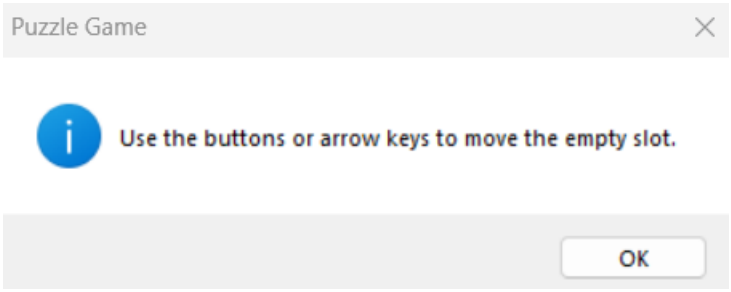
1.3 Target Audience

Designed with versatility in mind, this puzzle game is suitable for a wide range of audiences. From casual gamers seeking a brief mental challenge to enthusiasts of puzzle-solving looking for a deeper and more dynamic experience, the game caters to various preferences. Its simplicity in design belies the complexity of the underlying generative algorithm, offering depth to those who seek it.

2 Rules of the Game

2.1 Game Board

- The game unfolds on a configurable square grid, adapting to different puzzle sizes.
- Each grid cell holds a numeric value, except for one designated empty cell.

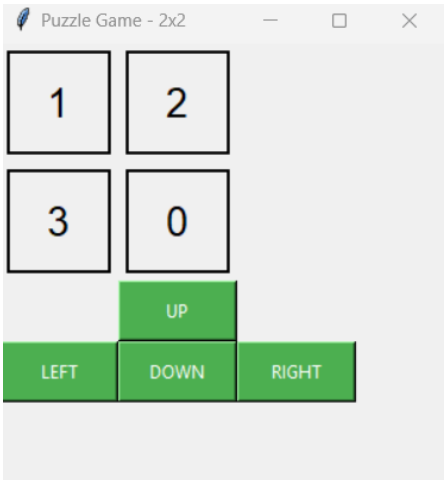


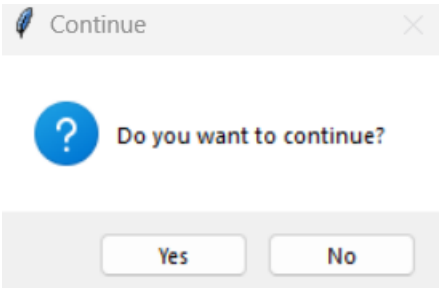
2.2 Objective

- Players engage in the strategic rearrangement of numeric values, striving to organize them in ascending order.
- Tiles move by sliding horizontally or vertically into the vacant cell.

2.3 Winning Condition

- Victory is achieved when all numeric values align in ascending order, culminating in the placement of the empty cell in the lower-right corner.





3 Puzzle Generation Algorithm

3.1 Solvability Check

- The generative algorithm ensures each puzzle is solvable.
- The inversion count method evaluates the order of numbered tiles to guarantee solvability.

```
def generate_solvable_puzzle(self):
    while True:
        puzzle = [[0 for _ in range(self.size)] for _ in range(self.size)]
        numbers = [i for i in range(1, self.size**2)]
        random.shuffle(numbers)

        for i in range(self.size):
            for j in range(self.size):
                if numbers:
                    puzzle[i][j] = numbers.pop(0)

        if self.is_puzzle_solvable(puzzle):
            self.board = puzzle  # Set the generated solvable puzzle to the board
            return
```

```
def is_puzzle_solvable(self, puzzle):
    inversion_count = 0
    flattened_puzzle = [num for row in puzzle for num in row]

    for i in range(len(flattened_puzzle) - 1):
        for j in range(i + 1, len(flattened_puzzle)):
            if flattened_puzzle[i] > flattened_puzzle[j] and flattened_puzzle[i] != 0 and flattened_puzzle[j] != 0:
                inversion_count += 1

    return inversion_count % 2 == 0
```

3.2 Implementation

- The algorithm initiates with a shuffled grid, providing a unique starting point for each game.
- A meticulous inversion count calculation guarantees the puzzle's solvability, contributing to a fair and challenging gaming experience.

4 Coding Language and Libraries

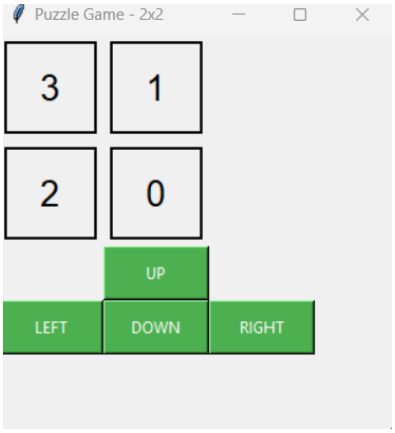
4.1 Programming Language

- Python is chosen for its readability, simplicity, and adaptability.
- The language seamlessly integrates algorithmic logic with graphical user interface development, supporting a cohesive and efficient development process.

4.2 GUI Library - Tkinter

- Tkinter is employed for creating the graphical user interface, offering a robust toolkit for building interactive and cross-platform applications.

- Its simplicity aligns with the game's design philosophy, fostering an intuitive user experience.

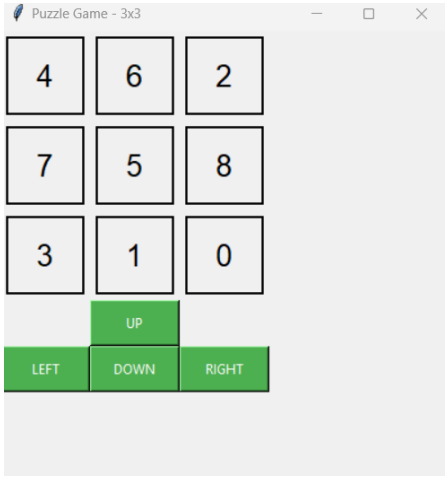


5 Puzzle Generation

5.1 `PuzzleGame` Class

- The `PuzzleGame` class encapsulates the puzzle generation logic, ensuring that each puzzle is solvable.

- The dynamic resizing feature enhances user experience, adapting the puzzle window size based on the chosen grid size.



5.2 Dynamic Puzzle Resizing

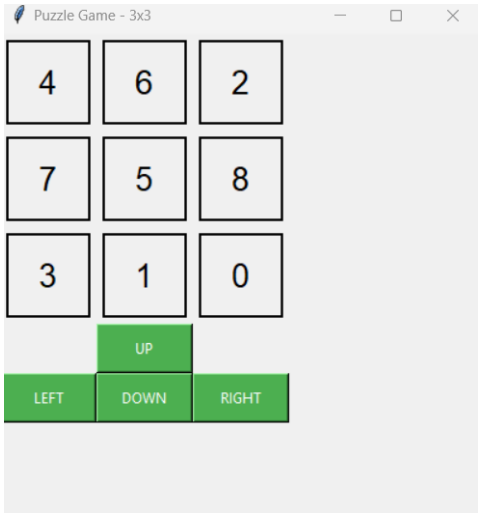
- The game's adaptive design dynamically adjusts the puzzle window's size, delivering a visually pleasing and responsive user interface.

6 GUI Implementation

6.1 `PuzzleGameUI` Class

- The `PuzzleGameUI` class orchestrates the graphical user interface, seamlessly integrating numbered tiles, directional buttons, and informative message boxes.

- User interactions, facilitated through directional buttons, trigger real-time updates to the display.



6.2 Player Interaction

- Intuitive directional buttons controls enable users to engage effortlessly with the puzzle.

- Real-time updates provide immediate feedback, enhancing the player's connection with the gaming experience.

6.3 User Feedback

- Message boxes deliver informative feedback, notifying players of successful puzzle resolutions and prompting them to continue their gaming journey.

7 Conclusion

The "Puzzle Game Using Generative AI" stands as a testament to the harmonious convergence of algorithmic sophistication and user-centric design. By crafting an experience centered on the fundamental allure of puzzles, the game not only entertains but also cultivates cognitive skills. Its adaptive generative algorithm ensures a dynamic and evolving challenge, while the Python and Tkinter combination guarantees accessibility and platform versatility. This puzzle game seeks to redefine the gaming experience, placing the focus on timeless challenges and engaging gameplay.

8 References

[1]	*Tkinter Documentation. (n.d.). Retrieved from https://docs.python.org/3/library/tkinter.html*
[2]	Python Software Foundation. (2022). Python 3.9.7 Documentation. Retrieved from https://docs.python.org/3/
[3]	GeeksforGeeks. (2022). Inversion Count for an array. Retrieved from https://www.geeksforgeeks.org/counting-inversions/
[4]	Random module documentation. (n.d.). Retrieved from https://docs.python.org/3/library/random.html
[5]	Tutorialspoint. (n.d.). Python GUI Programming - Tkinter. Retrieved from https://www.tutorialspoint.com/python/python_gui_programming.htm