Project Report

Connect 4 Game with Twist
(with GUI)

Author: Muhammad Ashir
Course: Artificial Intelligence
Language: Python 3

---

1. Executive Summary

This project extends the classic Connect 4 game by introducing a twist mechanic that allows shifting entire rows horizontally during gameplay. The twist introduces additional complexity and strategy beyond standard token placement. The game includes a graphical user interface (GUI) built with Pygame and features an AI opponent powered by the Minimax algorithm with Alpha-Beta pruning. The project successfully combines traditional game rules with innovative mechanics, intelligent decision-making, and interactive visuals.

---

2. Introduction

Background:
Connect 4 is traditionally a two-player game in which players alternate dropping tokens into a vertical grid with the objective of aligning four tokens in a row—horizontally, vertically, or diagonally.

Project Motivation:
To increase the game's strategic depth, a unique twist mechanic was added. The twist feature enables players or the AI to rotate rows either left or right after a set number of moves.

Objectives:

- Implement a twist-based version of Connect 4.
- Develop an AI capable of handling twist-based decisions.
- Build a fully interactive GUI for gameplay.

---

3. Tools and Technologies

- **Programming Language:** Python 3
- **Libraries:**
    - Pygame – for GUI, input handling, and rendering
    - NumPy – for board representation and state manipulation
    - Math, Random, Sys – for logic, exit handling, and AI behavior
- **Environment:** Desktop execution via Python

---

4. Game Description

Original Rules:

- Two players alternate dropping tokens into a 6×7 grid.
- The first to align four tokens wins (horizontal, vertical, or diagonal).

Modifications:

- A row can be twisted left or right after a configurable number of moves.
- After a twist, gravity is reapplied to stack the tokens correctly.
- Move thresholds depend on which side moves first (6/3 rule).

---

5. AI Design and Methodology

Algorithm Used:

- The AI uses the Minimax algorithm with Alpha-Beta pruning.
- A depth of 5 is used for decision-making.

Heuristic Evaluation:
The `score_position()` and `evaluate_window()` functions assess the board based on:

- Center column control
- Token alignment potential
- Blocking opponent moves

Twist Logic:
The AI uses `ai_choose_twist()` to evaluate all row-direction combinations by simulating the resulting board and applying gravity. It selects the twist that improves its score the most.

---

6. Game Mechanics and Flow

Phases:

1. **Token Placement** – Players drop pieces into valid columns.
2. **Twist Phase** – Triggered when the move count reaches a threshold (6 or 3 based on turn order).

Twist Rule Implementation:

- Player and AI twists are handled separately.
- Row numbers are clicked (player) or auto-selected (AI).
- Twists shift rows circularly and then apply gravity.

Turn Management:

- The game supports starting as either player.
- Player and AI alternate based on move completion and twist triggers.

Winning Conditions:

- The winning_move() function checks four-in-a-row in all directions after each move or twist.

---

7. Implementation Details

Board Model:

- A 6×7 NumPy array with values 0 (empty), 1 (player), and 2 (AI).

GUI Elements:

- Rendered using Pygame with circles representing pieces.
- Rows, grid slots, and twist buttons are displayed dynamically.

Event Handling:

- Mouse clicks determine token placement and player twist selection.
- AI executes moves and twists automatically.

Key Functions:

- drop_piece(), is_valid_location(), twist_row(), apply_gravity()
- minimax(), score_position(), get_valid_locations()
- draw_board() and draw_twist_ui()

8. Challenges Encountered

- Synchronizing twist phases with the turn system
- Ensuring gravity worked correctly after each twist
- Displaying correct GUI elements during twist vs move phases
- Handling simultaneous win conditions post-twist
- Avoiding index errors in row/column operations

---

9. Results and Observations

- The twist mechanic successfully introduces unpredictability and additional strategy.
- The AI effectively selects beneficial twists using heuristic evaluation.
- The game runs smoothly with responsive GUI interactions.
- The Minimax algorithm makes competent decisions, though timing depends on move complexity.

---

10. Conclusion

The project achieves its primary goals: implementing a twist-based Connect 4 variation with an AI opponent and a functional GUI. The twist mechanic adds an innovative strategic dimension to the traditional rules, while the Minimax algorithm with Alpha-Beta pruning ensures competitive AI gameplay. The interface is intuitive, and the logic is consistent with expected behavior.

---

11. Future Enhancements

- Add AI difficulty levels
- Include animations and sound effects
- Introduce multiplayer mode (local or online)
- Allow custom twist intervals via settings
- Improve heuristic evaluation for late-game situations
- Add restart and menu screens

---

12. References

- Pygame Documentation – https://www.pygame.org
- GeeksforGeeks – Minimax Algorithm in Game Theory
- Connect 4 strategy guides and AI references
- StackOverflow discussions on heuristics and board evaluation