**Project Report**

**Project Title: Strategic Score-Based Ludo Game with AI Decision-Making**

**Submitted By:** Abizar Qutb (22K-4288), Jamila Jamali (22K-4286), Zahra Safdar (22K-4510)  
**Course:** AI Lab  
**Instructor:** Sir Abdullah Yaqoob  
**Submission Date:** 13 May, 2025

1. **Project Overview:**

This project involves developing a strategic, score-based Ludo game in Python using the Pygame library. The key innovation lies in the integration of an intelligent AI player who competes alongside up to four human players. The AI leverages a Decision Tree-based logic system combined with a heuristic evaluation function to make strategic moves, such as selecting the most beneficial token to move, capturing opponents, and exploiting warp zones. Players earn points by looping their tokens around the board and capturing others. The highest scorer wins.

**2. Introduction**

**Background:**

Ludo is a classic multiplayer board game where players race tokens around a square board. This project introduces a modern, AI-integrated version that focuses not just on reaching home, but on **strategically capturing opponents** to score points. The addition of scoring, warps, and AI planning introduces tactical depth into an otherwise luck-based game.

**Objectives of the Project:**

1. Develop a fully playable Ludo game for up to five players (including AI).
2. Implement strategic AI using Decision Tree logic and heuristic scoring.
3. Introduce warp zones and loop tracking.
4. Provide an interactive Pygame-based GUI.
5. Enable human vs AI gameplay and display real-time logs and scores.

**3. Game Description**

**Original Game Rules:**

1. Each player controls 4 tokens.
2. Tokens are moved based on the roll of two dice.
3. Players capture opponent tokens by landing on them (excluding safe zones).
4. Completing a full loop is tracked for each token.
5. First to reach the winning score wins.

**Innovations and Modifications:**

* Introduced **Score-based mechanics** instead of traditional “reach home” rules.
* Warp zones randomly transport tokens for unpredictable strategies.
* Integrated **Decision Tree AI** with **heuristic evaluations** for move planning.
* Extended support to **5 players**, including 1 AI.
* Real-time game logs and player stats displayed in-game.

**4. AI Approach and Methodology**

**AI Techniques Used:**

* **Decision Tree Logic:** AI evaluates all possible tokens and chooses the one with the highest heuristic score.
* **Heuristic Evaluation Function:**
  + +10 points for potential captures.
  + +5 points for landing on a warp zone.
  + +1 point per step moved (as progress incentive).

**AI Strategy Design:**

1. **Capture Priority:** AI first checks if any opponent token can be captured.
2. **Token Placement:** If a six is rolled, the AI prefers placing new tokens.
3. **Best Move Selection:** If no immediate capture is possible, AI evaluates all moves using heuristics and selects the one with the highest score.

**Unpredictability:**

Because decisions depend on game state (token positions, scores, warp zones, dice), AI behavior varies per game, adding replayability and human-like strategy.

**5. Game Mechanics and Rules**

**Modified Game Rules:**

* Turn-based gameplay using two dice.
* Looping around the board does not end the game; instead, it adds score and progress.
* Warp zones dynamically shift token positions.
* Safe zones (start tiles) protect tokens from being captured.
* Score is increased through:
  + Captures (+1 point).
  + Full board loops (tracked).

**Winning Conditions:**

* First player to reach **5 points** wins the game.

**6. Implementation and Development**

**Development Process:**

* Designed modular classes: Board, Player, Dice, Token.
* Used Pygame for GUI and rendering dynamic gameplay.
* Integrated token-trail visualization, dice rolling, and move validation.
* Implemented AI logic for automated turns.
* Developed event-based system for user and AI interactions.

**Programming Languages and Tools:**

* **Language:** Python
* **Library:** Pygame
* **Other Files:** board.py, player.py, dice.py, config.py
* **AI Logic Location:** Integrated in main.py through evaluate\_token\_moves() and AI loop handler.

**7. Challenges Encountered**

* **Large Board Design:**  
  Managing a large board layout in Pygame was difficult due to space constraints and rendering limitations. Adjusting token placement, safe zones, and warp visualization required iterative tuning.
* **AI Design Complexity:**  
  Creating a realistic, unpredictable AI that doesn't rely purely on randomness was a major challenge.  
  Using a **Decision Tree strategy** and designing a **custom heuristic scoring system** required significant testing and tweaking.
* **Handling Multiple Players and Tokens:**  
  Designing scalable logic for up to 5 players, each with 4 tokens, led to complex condition handling and event sequencing.

**8. Team Contributions**

| **Member Name** | **Role & Contributions** |
| --- | --- |
| **Abizer Qutb** | Developed AI logic and real player interaction system. |
| **Zahra Safdar** | Designed and implemented the Pygame-based game frontend. |
| **Jamila Jamali** | Developed external helper modules for board, dice, and player classes. |

**9. Results and Discussion**

* The game is fully functional with a smooth Pygame interface.
* AI performs competitively and uses calculated decision-making.
* Playtesting showed that AI wins approximately 25–40% of the games, indicating balanced but challenging behavior.
* Warp zones added unexpected but fair variability to movement.
* Logs and real-time feedback improved player engagement.

**10. References**

* Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*.
* Ludo game rules: [Wikipedia](https://en.wikipedia.org/wiki/Ludo_(board_game))
* Pygame documentation: https://www.pygame.org/docs/
* Heuristics in AI: GeeksforGeeks & Towards Data Science tutorials