Project Report: Ludo 2.0

Ali Ansari, Anand Kumar and Suhail Ahmed

15 May 2025

Overview

Ludo 2.0 enhances the classic board game by introducing a strategy-based point system that allows players to influence dice outcomes, adding tactical decision-making to the traditional dice-roll mechanic. The game is currently implemented as a command-line interface (CLI) with core mechanics, including piece movement, captures, and a strategy point system. Future iterations will include an AI opponent that predicts optimal moves and adapts to player behavior to increase challenge and engagement.

Objectives

- Enhance player engagement through a point-based system for dice control.
- Implement core game mechanics for multi-player turn-based gameplay.
- Balance luck and skill to appeal to both casual and competitive players.
- Plan for an adaptive AI opponent to increase replayability and challenge.

Game Description

Innovations Introduced

- **Strategy Points System**: Players earn points through specific actions (e.g., capturing opponent pieces or reaching home) and can spend them to reroll dice, adding strategic depth.
- **Dynamic Game Mechanics**: Introduces skill-based elements, such as strategic dice rerolls and a unique dice mechanic where a roll of 3 moves pieces backward, while preserving Ludoś core gameplay.
- **Balanced Gameplay**: Players must decide between spending strategy points for immediate dice rerolls or saving them for critical moments.
- **Planned AI Opponent**: An intelligent AI is planned to adapt to player decisions and predict moves, offering a challenging experience (not yet implemented).

AI Approach and Methodology

AI Techniques (Planned)

- **Minimax Algorithm**: To be adapted for multi-player settings to simulate strategic moves and evaluate game states.
- **Alpha-Beta Pruning**: Will optimize AI decision-making by reducing the number of evaluated nodes in the search tree.
- **Heuristic-Based Decision Making**: The AI will evaluate factors like risk vs. reward for strategy point usage and piece movements.
- **Reinforcement Learning**: Planned to enable the AI to learn from past games and improve its strategies over time.

Current AI Status

The current implementation does not include an AI opponent. Players control all moves manually via the CLI. AI integration is a future enhancement.

Complexity Analysis

The strategy point system and potential AI analysis increase computational complexity compared to classic Ludo, which is primarily chance-based. Future AI heuristics will need to efficiently evaluate multiple player choices and game outcomes while maintaining responsive gameplay.

Game Rules and Mechanics

Modified Rules

- Players earn strategy points by capturing opponent pieces (2 points) or moving a piece to the home area (1 point).
- Players start with 5 strategy points and can spend 3 points to reroll the dice for a new outcome.
- A dice roll of 6 allows a piece to move from the yard to the board or grants an extra turn. A roll of 3 moves a piece backward by 3 spaces, adding a risk element.
- Pieces can capture opponentsṕieces on non-safe squares, sending them back to the yard and earning strategy points.
- Players can enter the home path when passing their colorś home entrance square and must move exactly to the final home position to finish a piece.

Winning Conditions

- A player wins by moving all their pieces into the home area (finished state).
- Strategic management of strategy points is key to optimizing piece movement and defending against opponents.

Implementation

Programming Language

Python

Libraries and Tools

- **Standard Libraries**: The current CLI implementation uses `random` for dice rolls and `time` for turn delays.
- **Planned Libraries**:
  - `Pygame`: For developing a graphical user interface (GUI).
  - `NumPy`: For efficient numerical operations and array manipulation in AI computations.
  - `Scikit-learn/TensorFlow`: For implementing reinforcement learning algorithms.

Current Code Structure and Functionality

The provided Python code implements a CLI version of Ludo 2.0, focusing on core game mechanics and the strategy point system.

Key Components

- **GAME_CONSTANTS and PLAYER_CONFIG**: Define parameters such as player count, piece count, board size, and strategy point settings.
- **BOARD_CONFIG**: Defines safe zones and home state parameters.
- **GameState Class**:
  - Tracks piece positions.
  - Manages strategy points.
  - Controls turn order and dice rolls.
  - Functions include `get_valid_moves()`, `move_piece()`, `apply_reroll()`, and `check_win_condit`
- **LudoCLI Class**:

  - Displays the board and game state.
  - Handles user input and runs the game loop.

Current Functionality

- Turn-based gameplay for 3 players with 3 pieces each.
- Dice rolling with special mechanics (e.g., roll of 3 moves backward).
- Full support for movement, captures, home path entry, and win detection.
- Strategy point system allows rerolling dice.
- Extra turns awarded for roll of 6, capture, or reaching home.
- Text-based CLI interaction.

Limitations and Future Enhancements

- **AI Integration**: Not yet implemented; planned using Minimax, Alpha-Beta, and Reinforcement Learning.
- **Graphical Interface**: Future version will include a GUI via Pygame.
- **Advanced AI Features**: Heuristic learning and adaptive behavior will enhance AI sophistication.
- **Expanded Strategy Options**: May include direct dice value manipulation in future versions.