Ernesto Valencia

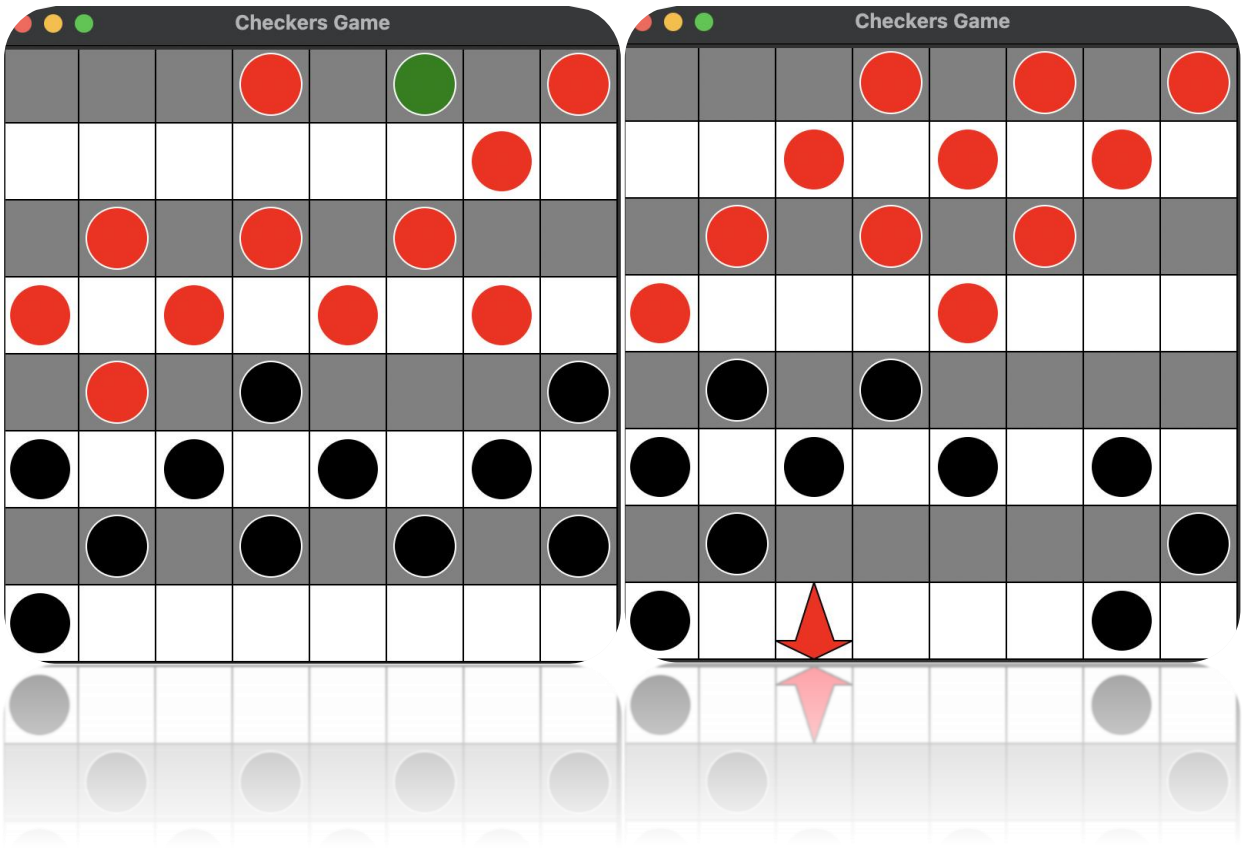
CS 470

Dr. Chris Archibald

Project Report: Checkers Game with AI Opponent

Overview

For this project, I embarked on creating an interactive Checkers game featuring both human and AI players. The core objectives were to develop a user-friendly graphical interface, implement the standard rules of Checkers, and integrate an AI opponent using Monte Carlo Tree Search (MCTS). This endeavor aimed not only to produce a playable game but also to delve into the complexities of AI programming within a board game context.



Time Investment Breakdown (22 Hours Total)

1) Conceptualization and Planning (4 hours): The initial phase involved brainstorming the project's scope, defining the game mechanics, and outlining the AI's strategic approach. Essential resources were identified, including Tkinter for the GUI and algorithmic guides for MCTS.
2) Graphical User Interface (GUI) Development (5 hours): Leveraging Tkinter, I designed and implemented a simple yet functional interface. This step was crucial for ensuring an engaging user experience. Key features included a checkerboard display, piece movement handling, and visual feedback for player interactions.
3) Game Logic Coding (5 hours): This phase focused on translating Checkers rules into code. It included setting up the board, defining legal moves, handling piece captures, and implementing kinging. Special attention was paid to ensuring that the game accurately followed standard Checkers rules.
4) AI Development and Integration (6 hours): The AI opponent was created using the MCTS algorithm. This involved coding the AI to simulate various game scenarios and make decisions based on those simulations. The AI's integration into the game required extensive testing to balance challenge and fairness.
5) Testing, Debugging, and Refinement (2 hours): The final hours were dedicated to rigorously testing the game, identifying bugs, and refining both the GUI and game logic. This process involved playing multiple game scenarios and adjusting the AI algorithm for optimal performance.

Project Highlights

- Interactive GUI: The development of a responsive GUI using Tkinter was a significant achievement, providing an intuitive platform for players to interact with the game.
- Robust Game Mechanics: Implementing the rules of Checkers in a digital format was challenging but rewarding, ensuring the game remained true to its classic form. Although it does not have good error handling and can get stuck at multi-jump captures, nevertheless it works for simple games.
- AI Strategy Implementation: Crafting an AI opponent that could challenge human players was a complex yet intriguing aspect of the project. MCTS was instrumental in developing an AI that could adapt and make strategic decisions.

Evaluation and Testing

The game's effectiveness was evaluated through multiple gameplay sessions, both with human participants and AI versus AI matches. The AI demonstrated competent strategic play but occasionally fell into predictable patterns, highlighting areas for future enhancement.

Resource Utilization

- Tkinter Documentation: Essential for mastering GUI development aspects.
- MCTS Guides and Articles: Provided valuable insights into AI implementation strategies.

Lessons and Insights

- AI Difficulty Balance: Creating an AI that is neither too easy nor impossibly hard to beat is crucial for engaging gameplay.
- Efficiency in Game Logic: Optimizing code for game rules and AI calculations is vital to prevent performance issues, particularly in real-time decision-making scenarios.

Opportunities for Improvement

- Enhanced AI Algorithm: Exploring advanced AI techniques or refining the existing MCTS implementation could yield a more formidable opponent.
- GUI Aesthetics: Further development could focus on graphical enhancements, making the game visually appealing and engaging.
- Enhancing Error Handling: Here and there, some bugs appear (multi-jumping bug, or King checker stops being King checker after multi-jump). Those need to be handled, and overall error handling needs to be implemented.
- Additional Features: Implementing game-saving options, multiplayer functionality, and a game history log could significantly enhance the playing experience.

Future Applications

The methodologies and code developed for this project could serve as a valuable learning resource for students exploring AI in gaming contexts, however, it needs to be developed further and be made more robust before it could be completely helpful in a college class.

The AI implementation provides a practical case study in applying theoretical AI concepts to real-world applications.

Conclusion

This project succeeded in creating a fully functional, AI-enhanced Checkers game, offering an engaging blend of classic gameplay and modern AI challenges. It served as an enriching learning experience in game development and artificial intelligence, laying a foundation for future explorations in these fields.