Home About Contact

Go Game with AI and GUI Using Python 🎮✨🕊️

Comprehensive Report on Design, Algorithms, and Implementation

Presented by:

Ahmed Gamal

Project URL :

GitHub Link

Introduction and Overview

Project Idea:

🧩 Develop a functional Go game with AI and a user-friendly GUI.

Objective:

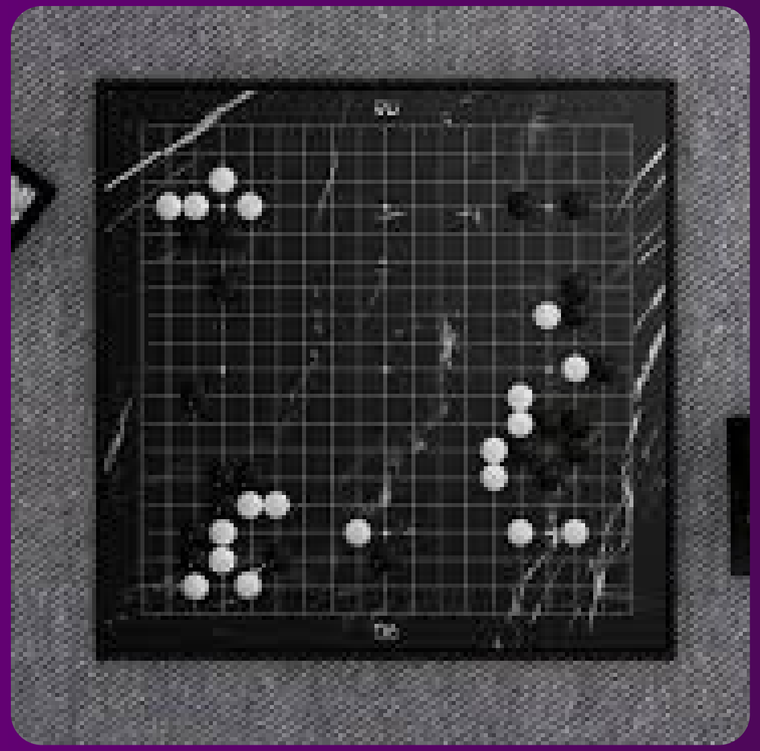
🎯 Provide an interactive gaming experience with customizable board sizes and AI difficulty levels.

Academic Context:

📚 Combines game development and AI optimization, utilizing Python's libraries.

Similar Applications:

- KGS Go Server: 🌐 Online platform for Go players.
- SmartGo: 📱 AI-based Go game for iOS and Android.
- Comparison: 🔍 Our project focuses on Python-based desktop implementation with a focus on simplicity and user interaction.



Literature Review

1. “The Elements of Go Strategy” by Bozulich and Davies 📖
    - Offers insights into strategic approaches in Go.
2. “AlphaGo Zero: Mastering the Game of Go” by Silver et al. 🧠
    - Highlights AI advancements in playing Go with deep reinforcement learning.
3. “Minimax Algorithm with Alpha-Beta Pruning” (AI-focused papers) 📂
    - Discusses optimization techniques for decision-making algorithms.
4. “Tkinter GUI Development” from official Python documentation 🛠️
    - Explains best practices for building user-friendly interfaces in Python.
5. “Heuristic Evaluation in AI” (Practical guide to heuristics) 🔬
    - Guides the design of evaluation functions for AI decision-making.

- Takeaways: 💡 These resources guided the design of game rules, AI implementation, and GUI development.

Timmerman Industries

Proposed Solution

Main Features (from user perspective):

- 🧩 Interactive board (sizes: 9x9, 13x13, 19x19).
- 💾 Save and load game functionality.
- 📊 Real-time score updates.
- 🤖 AI difficulty levels (Easy, Medium, Hard).

Applied Algorithms

Minimax Algorithm:

- 🧐 Evaluates all possible moves.
- 🏆 Chooses the move with the highest score for the player.

Alpha-Beta Pruning:

- 🚀 Optimizes Minimax by ignoring branches that don’t affect the outcome.
- 🔄 Improves computational efficiency.

Heuristic Evaluation:

- 🧠 Scores board states based on territory, captured stones, and potential moves.

Timmerman Industries

Board State Management

- Board Representation:
  - 🗂️ 2D Array: Implemented using NumPy for efficient state tracking.
- States Tracked:
  - 🔍 Current board state.
  - ↩️ Previous states (for undo functionality).
  - ⚡ Captured stones (dynamic score updates).
- Save/Load Feature: 💾 Uses JSON serialization for state storage.

Experiments and Results

- Experiments:
  - ⚙️ Tested AI performance at different difficulty levels.
  - 📊 Benchmarked computational efficiency of Minimax with and without Alpha-Beta Pruning.
- Results:
  - 🟢 Easy AI: Random moves, low computational cost.
  - 🟡 Medium AI: Minimax with shallow depth, moderate difficulty.
  - 🔴 Hard AI: Minimax with deep evaluation and Alpha-Beta Pruning, high difficulty.

Timmerman Industries

Analysis and Discussion

- **Insights:**
  - 🚀 **Alpha-Beta Pruning significantly reduces computation time.**
  - ✅ **Heuristic evaluation ensures the AI selects optimal moves without exhaustive search.**
- **Advantages:**
  - 🛠️ **Customizable difficulty levels.**
  - ✨ **Responsive and intuitive GUI.**
- **Limitations:**
  - 🕒 **High computational cost on larger boards (19x19).**
  - 🎨 **GUI limitations with Tkinter for advanced visualizations.**
- **Future Work:**
  - 🌌 **Implement Monte Carlo Tree Search (MCTS) for improved AI.**
  - 🌐 **Add online multiplayer functionality.**
  - 🖌️ **Enhance GUI design using advanced frameworks like PyQt or Kivy.**

Documentation for Users and Developers

- User Guide:
  a. 🖥️ Install Python and required libraries.
  b. ▶ Run main.py to start the application.
  c. 🛠️ Use menu options to:
    - 🆕 Start a new game.
    - 🏞️ Change board size.
    - 💾 Save/load game states.

  d. 🤖 Choose AI difficulty level and interact with the board by clicking intersections.
- Developer Guide:
  - Code Structure:
    - 🧩 game_logic.py: Handles rules, scoring, and move validation.
    - 🧠 ai.py: Implements Minimax and Alpha-Beta Pruning.
    - 🎨 gui.py: Manages the Tkinter interface.
    - ⚙️ utils.py: Includes helper functions for state management.
  - Instructions:
    - 🔧 Extend AI by modifying ai.py.
    - ✏️ Customize GUI elements in gui.py.

Conclusion Summary: 🏁✨🤝

Summary:

- 🎮 Developed a Go game with AI and GUI in Python.
- 🧠 Implemented Minimax with Alpha-Beta Pruning for optimized decision-making.
- ✨ Provided interactive gameplay with real-time updates.

Closing Note:

- 🔗 The project demonstrates the integration of AI, game logic, and GUI design.
- 🚀 Future enhancements aim to expand functionality and improve user experience.

Acknowledgments and References

Acknowledgments:

1. 👥 Team members and mentors.
2. 🐍 Python community for libraries and documentation.

References:

- 📖 “The Elements of Go Strategy” by Bozulich and Davies.
- 🧠 “AlphaGo Zero: Mastering the Game of Go” by Silver et al.
- 🛠️ Python Tkinter Documentation.
- 🗂️ NumPy Documentation.
- 🔬 “Heuristic Evaluation in AI”.

Timmerman Industries

Home About Contact

Thank You

FOR YOUR ATTENTION

Contact Information:

GitHub: https://GitHub