Rock-Paper-Scissors Board Game – Python-Based Turn-Based Strategy Game

Submitted By: Taha Sharif (22k-4145).Asadullah (22k-4138), Shozab Mehdi (22k-4522)

Course: AI

Instructor: Sir Shafique Ur Rehman

Submission Date: May 11, 2025

1. Executive Summary

This project aimed to reimagine the classic Rock-Paper-Scissors game into a strategic turn-based board game. We implemented AI opponents with varying difficulty levels (RandomAI, BasicAI, AdvancedAI, and MinimaxAI with alpha-beta pruning) to provide a progressive challenge. Our version supports 2-3 players, introduces strategic movement on a 6x6 grid, and leverages heuristic and decision-tree AI techniques.

2. Introduction

Background

Rock-Paper-Scissors is a simple two-player hand game involving three possible moves with circular dominance. We transformed it into a grid-based tactical board game to integrate strategic planning and AI behavior in a multi-agent setting.

Objectives of the Project

- Design and implement a 6x6 board game version of Rock-Paper-Scissors.
- Integrate four levels of AI using increasingly complex decision-making algorithms.
- Evaluate AI performance and compare techniques.
- Provide a polished GUI and game experience with animations and sound.

3. Game Description

Original Game Rules

The original Rock-Paper-Scissors is a two-player game where each player chooses one of three options: rock, paper, or scissors. Rock beats scissors, scissors beats paper, and paper beats rock.

Innovations and Modifications

- Turn-based board game on a 6x6 grid.
- Support for up to 3 players with randomized starting positions.
- Rock, Paper, and Scissors act as unique movable pieces.
- AI integration with 4 levels of difficulty.
- Strategic movement, combat rules, animations, and sound.

4. AI Approach and Methodology

AI Techniques Used

The game includes four AI techniques:
- RandomAI: Selects legal moves randomly.
- BasicAI: Rule-based selection using safe/capturing/risky priorities.
- AdvancedAI: Uses heuristic evaluation functions to rank moves.
- MinimaxAI: Implements minimax with alpha-beta pruning for lookahead-based decision making.

Algorithm and Heuristic Design

Heuristics evaluate:
- Material advantage
- Center control
- Type distribution (R/P/S balance)
- Threat/opportunity patterns

AI Performance Evaluation

Performance was assessed by measuring win rates against human players and decision latency:
- BasicAI: ~0.5s
- AdvancedAI: ~0.8s
- MinimaxAI: ~1.2s

MinimaxAI consistently outperformed others in decision quality and survival rate.

5. Game Mechanics and Rules

Modified Game Rules

- 6x6 grid with random initial piece placement.
- Players have 1 Rock, 1 Paper, 1 Scissors.
- Pieces move one cell horizontally or vertically.
- Combat follows classic RPS rules.

Turn-based Mechanics

Players take turns selecting a piece and making a move. Each turn allows one move per player. Combat occurs when a piece moves onto an opponent.

Winning Conditions

The game ends when only one player has pieces left. If no players have pieces left, the game is declared a draw.

6. Implementation and Development

Development Process

The game was developed in Python using Pygame. The design followed an object-oriented architecture. AI modules were developed and tested individually before full integration into the main game loop.

Programming Languages and Tools

- Language: Python
- Libraries: Pygame, random
- Tools: GitHub, VSCode

Challenges Encountered

- Handling AI decision complexity at higher depths.
- Balancing AI difficulty levels.
- Managing GUI responsiveness alongside AI computation.
- Ensuring smooth transitions and bug-free combat resolution.

7. Team Contributions

- Asadullah (22k-4138): Implemented multiplayer modes, BasicAI, and core board mechanics.
- Shozab Mehdi (22k-4522): Designed GUI, animations, And advanced GUI with heuristics.
- Taha Sharif (22k-4145): Developed ExpertAi and MinimaxAI with alpha-beta pruning, Also contributed in GUI.
- All team members contributed to testing, debugging, and design refinement.

8. Results and Discussion

The game successfully demonstrated AI behavior differences:
- BasicAI won ~40% vs. RandomAI
- AdvancedAI won ~60% vs. BasicAI
- MinimaxAI won ~75% across different configurations

AI performance scaled well with decision logic, with MinimaxAI offering competitive strategic play.

9. References

- Pygame Documentation (https://www.pygame.org/docs/)
- AI Game Programming Wisdom (Steve Rabin)
- Python Docs (https://docs.python.org/3/)
- GeeksforGeeks and Stack Overflow community answers