جامعة بنها
BENHA UNIVERSITY

كلية الحاسبات والذكاء الاصطناعي
FACULTY OF COMPUTERS & ARTIFICIAL INTELLIGENCE

Connect 4 Game System

Professional Project Documentation

Project Team Members

This project was developed by a collaborative team of students demonstrating expertise in software engineering, artificial intelligence, and system design:

- **Mohamed Sobhy Mohamed Awad**
- **Mohamed Amir Abouzaid Ahmed**
- **Mohamed Ayman Yehia Abdelsalam**
- **Yusuf Hesham Abdel-Fattah Abdelazim**

Ai Department

December 2025

Abstract

This documentation presents the design and implementation of a Connect 4 Game System that integrates classical game theory with modern software engineering principles. The system supports three operational modes: Player vs Player, Player vs Artificial Intelligence, and Artificial Intelligence vs Artificial Intelligence, serving both educational and recreational objectives.

The application is developed using Python, utilizing NumPy for efficient board representation and Pygame for graphical rendering and user interaction. The artificial intelligence component is based on the Minimax algorithm, enhanced with Alpha-Beta pruning to improve computational efficiency while maintaining optimal decision-making.

The system adopts a modular architecture featuring a centralized menu that launches independent game modes as separate processes, demonstrating effective process management and modular design practices. Performance evaluation shows that Alpha-Beta pruning reduces node exploration by approximately 60–70% compared to standard Minimax, enabling real-time gameplay with search depths of up to five or six moves on typical hardware.

Table of Contents

1. Executive Summary

2. Introduction

    2.1 Background and Context

    2.2 Project Motivation

    2.3 Scope and Objectives

3. System Architecture

    3.1 Architectural Overview

    3.2 Component Structure

    3.3 Process Management Framework

4. Game Modes and Functionality

    4.1 Player versus Player Mode

    4.2 Player versus AI Mode

    4.3 AI versus AI Mode

5. Menu System Design

    5.1 User Interface Architecture

    5.2 Visual Design Principles

6. Artificial Intelligence Implementation

    6.1 Minimax Algorithm

    6.2 Alpha-Beta Pruning Optimization

    6.3 Heuristic Evaluation Function

    6.4 Strategic Decision Making

7. Technical Specifications

    7.1 Configuration Parameters

    7.2 Performance Metrics

7.3 Data Structures

8. Installation and Configuration

8.1 System Requirements

8.2 Installation Procedures

8.3 Troubleshooting Guide

9. User Documentation

9.1 Application Launch

9.2 Gameplay Instructions

9.3 Game Rules

10. Code Organization

10.1 Function Reference

10.2 Algorithm Complexity

11. Testing and Validation

12. Development Team

13. Conclusion

14. References and Bibliography

1. Executive Summary

1.1 Project Overview

The Connect 4 Game System is a complete software project that combines several important computer science areas, such as artificial intelligence, software architecture, graphical user interface (GUI) design, and basic concurrent programming.
The project provides a fully working game with three different playing modes. Each mode is designed for a specific purpose, including normal entertainment and educational use to demonstrate how AI algorithms work in games.

1.2 Technical Foundation

Programming Language: Python 3.12.1 or higher, selected for its extensive library ecosystem, cross-platform compatibility, and suitability for rapid application development.

Core Libraries: NumPy for numerical array operations ,Pygame for graphics rendering, event handling, and multimedia capabilities.

1.3 Key Achievements

- Successfully implemented a complete game system with three fully functional modes of operation
- Developed sophisticated AI using Minimax algorithm achieving optimal play through Alpha-Beta pruning

2. Introduction

2.1 Project Motivation

The development of this Connect 4 Game System was motivated by several interconnected objectives:

- **Educational Value:** The project provides a practical demonstration of fundamental concepts in artificial intelligence, particularly adversarial search algorithms and game tree exploration. Students and practitioners can examine a complete, working implementation of Minimax with Alpha-Beta pruning, gaining insights into algorithm design and optimization techniques.
- **Algorithm Optimization:** The implementation of Alpha-Beta pruning showcases practical optimization techniques that reduce computational complexity without sacrificing solution quality. Performance analysis demonstrates the dramatic improvements achievable through intelligent algorithmic enhancements.

2.3 Scope and Objectives

The primary objectives of this project encompass both technical and educational goals:

1. Develop a complete, functional implementation of Connect 4 with multiple play modes (Player vs Player , Player vs Ai and Ai vs Ai).
2. Implement artificial intelligence using industry-standard algorithms and optimization techniques
3. Create a professional-grade user interface with menu system and process management
4. Demonstrate software engineering best practices in code organization and documentation
5. Provide educational value through clear implementation of AI concepts and algorithms
6. Achieve optimal performance through algorithmic optimization and efficient data structures

3. System Architecture

3.1 Architectural Overview

The Connect 4 Game System uses a launcher–worker architecture that separates the main menu interface from the game execution logic. This design improves stability by isolating each game mode in its own process and allows better resource management.

The system includes a main launcher (main.py) that acts as the user interface. When a game mode is selected, the launcher starts a separate Python process for that game. Each game runs independently with its own game state and event loop, preventing interference between modes and ensuring smooth execution.

3.2 Component Structure

The system consists of four main executable files, each with a specific role:

- main.py (Launcher):
  Provides the main menu interface and controls game mode selection. It launches game modes as separate Python processes using the subprocess module, ensuring the menu remains responsive while games run independently.

- connect4.py (Player vs Player):
  Implements local two-player gameplay with mouse-based input, real-time move validation, win detection, and visual feedback for turns and victories.

- connect4_with_ai.py (Player vs AI):
  Allows a human player to compete against an AI using the Minimax algorithm with Alpha-Beta pruning. The AI difficulty can be adjusted by changing the search depth, supporting beginner to advanced gameplay levels.

- connect4_ai_vs_ai.py (AI vs AI):
  Runs a fully automated match between two AI agents using the same decision-making algorithm. A short delay between moves enables observation, making this mode useful for learning, testing, and demonstration purposes.

3.3 Process Management Framework

The menu system uses Python's subprocess module to manage game execution. When a user selects a game mode, the launcher starts the corresponding game file as a separate process using subprocess.Popen.

This approach allows non-blocking execution, keeping the menu responsive while games are running. Each game runs in an isolated process with its own memory and resources, ensuring stability and preventing interference between sessions.

The launcher monitors active processes using polling. When a game window closes, the process ends naturally, and the launcher releases its resources. This design supports safe termination, crash handling, and efficient resource management.

4. Game Modes and Functionality

4.1 Player versus Player Mode

The Player versus Player mode provides a classic local multiplayer experience where two human participants compete on a single computer system. This mode implements the fundamental Connect 4 gameplay mechanics without artificial intelligence components, focusing instead on robust input handling, clear visual feedback, and comprehensive game logic implementation.

4.1.1 Technical Implementation

The game uses Pygame’s event system to handle mouse input. When a player clicks, the x-coordinate is converted into a column index, which is then validated to ensure it is within bounds and not full.

If valid, the system finds the lowest available row in that column and places the player’s piece in the NumPy board array. The graphical interface updates immediately, displaying the new piece at the correct position.

4.1.2 Features and Capabilities

- Mouse-based column selection with real-time visual highlighting
- Turn indication system displaying which player should move next
- Immediate validation of move legality with visual feedback for invalid attempts
- Comprehensive win detection checking horizontal, vertical, and both diagonal directions
- Draw condition detection when board fills without achieving connection
- Winner announcement with prominent text display and color-coded messaging

4.2 Player versus AI Mode

The Player versus AI mode introduces artificial intelligence as an opponent, providing single-player gameplay with adjustable difficulty levels. This mode combines human input handling from the Player versus Player implementation with sophisticated AI decision-making powered by the Minimax algorithm and Alpha-Beta pruning optimization.

4.2.1 AI Configuration and Difficulty

The artificial intelligence difficulty level is determined primarily by the search depth parameter passed to the Minimax algorithm. This parameter specifies how many moves ahead the AI will analyze when evaluating possible moves. The relationship between search depth and playing strength is exponential, as each additional level of depth multiplies the number of positions evaluated by the branching factor (approximately 4-7 valid moves per position in typical mid-game situations).

Depth Range	**Skill Level**	**Characteristics**
3-4	Beginner	Makes tactical errors, misses winning opportunities, suitable for learning
5-6	Intermediate	Solid strategic play, rarely makes mistakes, challenging for most players
7+	Expert	Near-optimal moves, extremely difficult to defeat, slower computation

4.2.2 Gameplay Dynamics

Gameplay alternates between human input phases and AI computation phases. During the human player's turn, the system operates identically to Player versus Player mode, accepting mouse input and validating column selection. Once the human player makes a valid move, control transfers to the AI component.

The AI invokes the Minimax algorithm with the current board state, configured search depth, and appropriate piece identifier. The algorithm explores the game tree, evaluating positions using the heuristic function and pruning non-promising branches through Alpha-Beta optimization. Upon completion, Minimax returns the optimal column index, which the system validates (redundantly, for safety) before executing the AI's move and updating the display.

4.3 AI versus AI Mode

The AI versus AI mode represents a fully automated demonstration where two artificial intelligence agents compete without human intervention. This mode serves multiple purposes including educational demonstration of AI capabilities, algorithm performance testing and validation, and entertainment through observation of strategic machine play.

4.3.1 Implementation Details

Both AI agents utilize identical Minimax implementations with Alpha-Beta pruning, typically configured to the same search depth to ensure fair competition. The game loop alternates control between agents, with each invoking its Minimax algorithm to determine the optimal move. A configurable delay (default 500 milliseconds) is introduced between moves to allow human observers to follow the progression of play.

The first player is selected randomly at game initialization to eliminate any systematic advantage from move order. All moves are logged to the console in addition to graphical display, providing a complete record of the game progression suitable for analysis and debugging purposes.

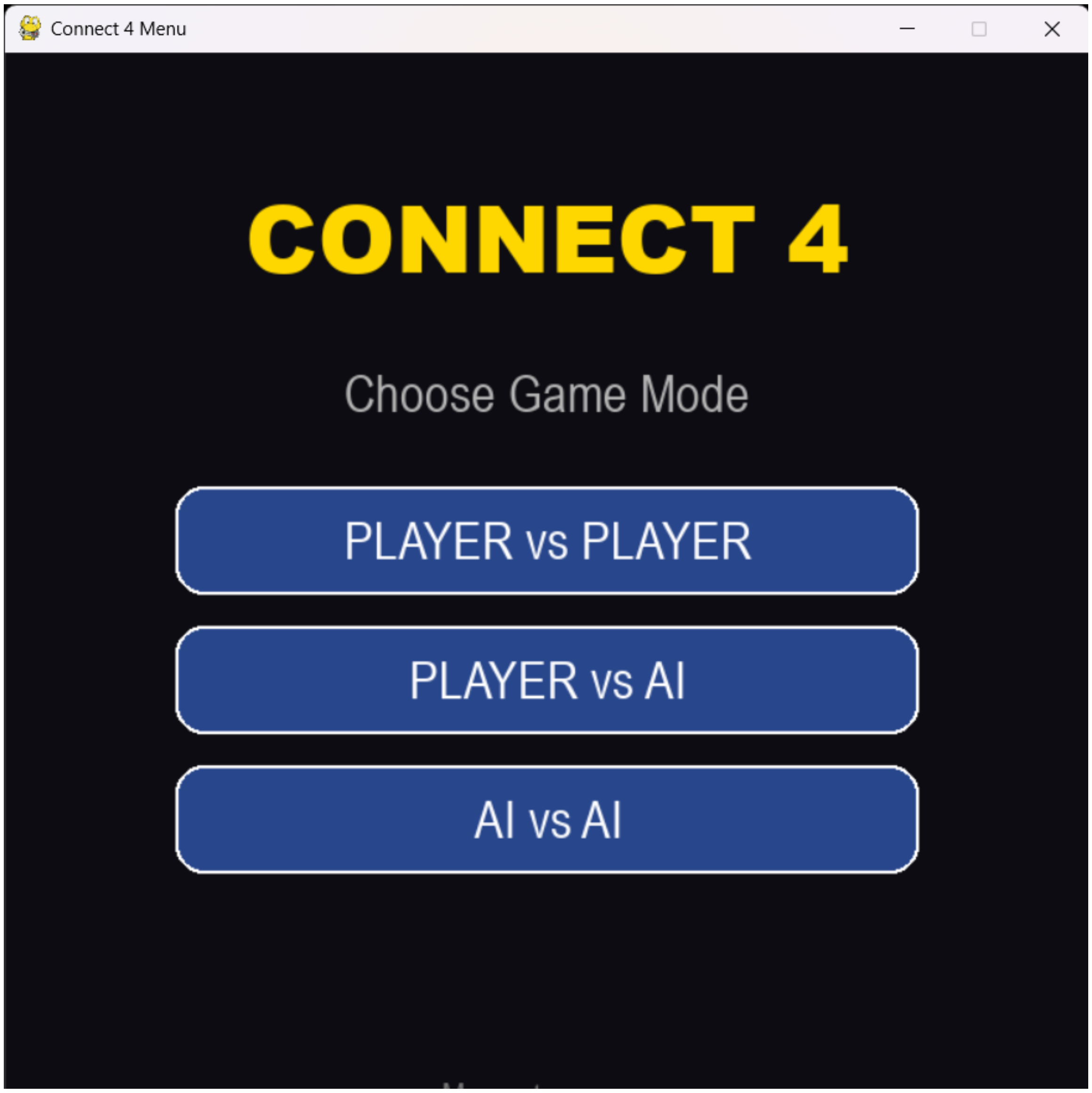
4.3.2 Applications and Use Cases

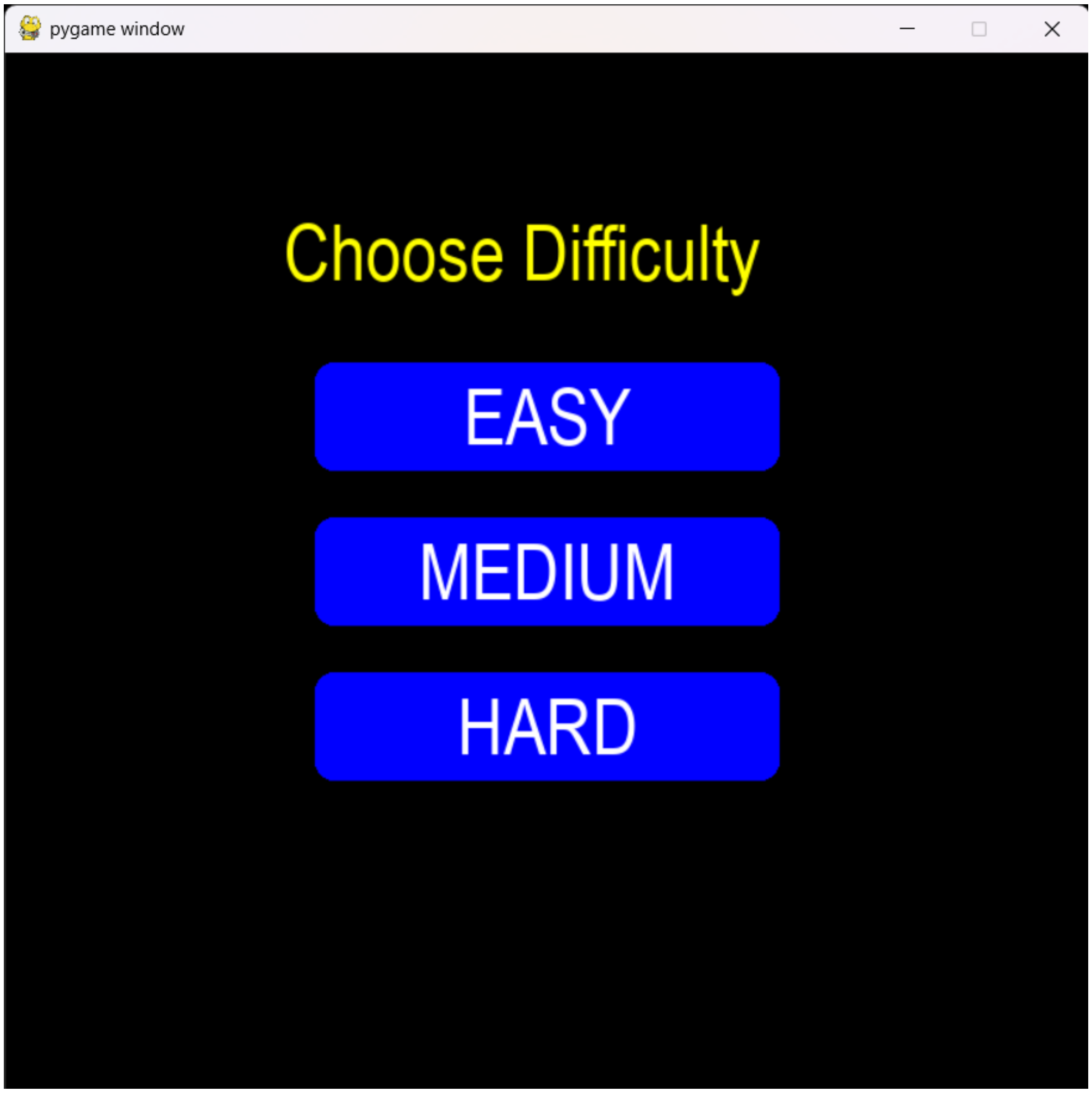
- Algorithm performance benchmarking and computational efficiency analysis
- Validation of AI correctness through observation of strategic decision-making
- Educational demonstrations for teaching game theory and AI concepts
- Entertainment value through observation of automated strategic play
- Testing of heuristic evaluation function effectiveness
- Debugging and refinement of AI algorithms in controlled conditions

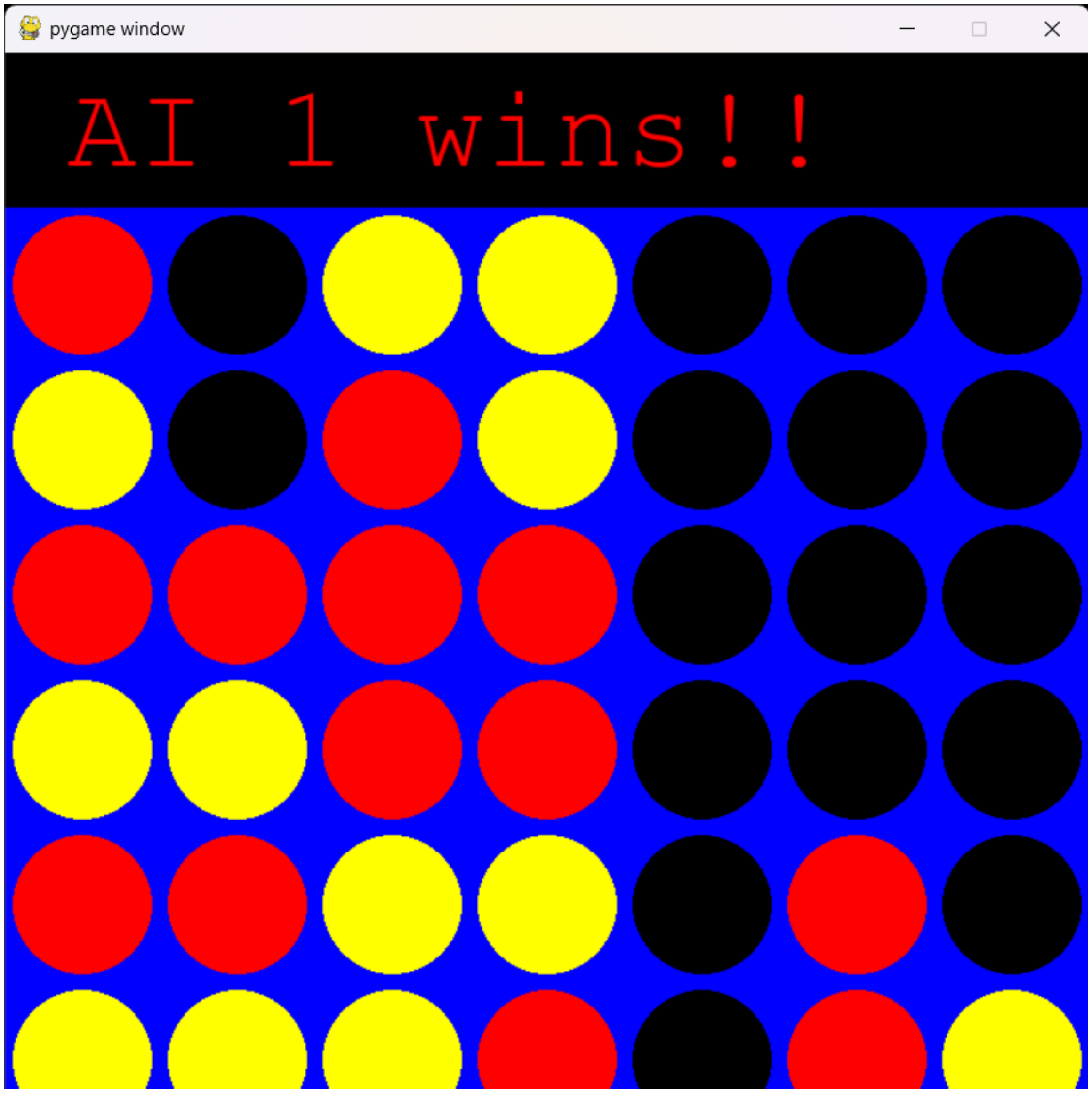
5. Menu System Design

5.1 User Interface Architecture

The menu system functions as the primary user interface and entry point for the application. Implemented using Pygame's graphics and event handling capabilities, the menu presents a clean, intuitive interface that enables users to launch game modes through simple mouse interaction. The design prioritizes clarity, immediate feedback, and visual appeal while maintaining professional aesthetics appropriate for both educational and recreational contexts.


Connect 4 Menu
CONNECT 4
Choose Game Mode
PLAYER vs PLAYER
PLAYER vs AI
AI vs AI


pygame window
Choose Difficulty
EASY
MEDIUM
HARD



5.2 Visual Design Principles

The visual design adheres to established user interface design principles including visual hierarchy, consistency, feedback, and affordance. The color palette was carefully selected to provide sufficient contrast for readability while maintaining modern aesthetic appeal. The typography employs clear, legible fonts with appropriate sizing to establish information hierarchy.

- **Background:** Deep space blue (#0C0C12) provides a professional, non-distracting foundation that enhances the visibility of interface elements.
- **Title Element:** Gold color (#FFD700) applied to the "CONNECT 4" title establishes clear visual hierarchy and creates an attractive focal point.

- **Interactive Buttons:** Bright blue (#5A96FF) provides clear affordance for clickable elements, with white borders and text ensuring maximum readability.
- **Hover Effects:** Color shifts on mouse hover provide immediate visual feedback, confirming interactivity and improving user confidence in navigation.

6. Artificial Intelligence Implementation

6.1 Minimax Algorithm

The Minimax algorithm represents a cornerstone of adversarial game-playing artificial intelligence. Originally formulated by John von Neumann in 1928 as part of his work on game theory, Minimax provides a formal framework for decision-making in zero-sum games where one player's gain directly corresponds to the opponent's loss. The algorithm's fundamental premise assumes that both players will play optimally, seeking to maximize their own advantage while minimizing the opponent's potential gains.

6.1.1 Theoretical Foundation

Minimax operates by constructing and exploring a game tree—a hierarchical data structure where each node represents a possible board state and edges represent legal moves. The tree's root node corresponds to the current game position, while leaf nodes represent terminal states (wins, losses, or draws). The algorithm recursively evaluates this tree by alternating between maximizing and minimizing layers.

At maximizing layers (representing the AI's turns), the algorithm seeks the move that produces the highest evaluation score, assuming the opponent will subsequently make their best defensive move. Conversely, at minimizing layers (representing the opponent's turns), the algorithm assumes the opponent will select the move that minimizes the AI's advantage. This alternating process continues until reaching the specified search depth or encountering a terminal game state.

6.1.2 Implementation Details

The implementation accepts several key parameters that control its behavior:

- board: Current game state represented as a 6x7 NumPy array
- depth: Number of moves ahead to analyze (search depth)
- alpha: Best value guaranteed to maximizer along path to root
- beta: Best value guaranteed to minimizer along path to root
- maximizingPlayer: Boolean indicating current layer type
- piece: AI player identifier (1 or 2)

6.2 Alpha-Beta Pruning Optimization

Alpha-Beta pruning represents a critical optimization technique that dramatically improves Minimax performance without sacrificing solution quality. Developed independently by multiple researchers in the late 1950s and early 1960s, this optimization eliminates evaluation of branches that cannot possibly influence the final decision, reducing computational complexity while guaranteeing identical results to exhaustive Minimax search.

6.2.1 Pruning Mechanism

The optimization maintains two values throughout the tree traversal: alpha (α) representing the best score achievable by the maximizing player, and beta (β) representing the best score achievable by the minimizing player. As the algorithm explores the tree, it updates these bounds based on actual position evaluations. When a node's evaluation indicates that one player can force a better outcome through a different branch already explored, the current branch can be safely pruned without further analysis.

The pruning condition occurs when alpha becomes greater than or equal to beta ($\alpha \geq \beta$). This condition indicates that the maximizing player has already found a move that guarantees a better outcome than the minimizing player will allow in the current branch. Mathematically, this represents a guarantee that the current branch cannot produce a better result than alternatives already discovered.

6.2.2 Performance Analysis

Theoretical analysis demonstrates that Alpha-Beta pruning can, in the best case, reduce the effective branching factor from b to $\sqrt{b}$, achieving a time complexity of O(b^(d/2)) compared to O(b^d) for naive Minimax. In practical Connect 4 implementation, empirical testing reveals:

Search Depth	Without Pruning	With Alpha-Beta
Depth 4	~10,000 nodes	~2,500 nodes
Depth 5	~50,000 nodes	~8,000 nodes
Depth 6	~250,000 nodes	~35,000 nodes

6.3 Heuristic Evaluation Function

The heuristic evaluation function assigns numerical scores to non-terminal board positions, providing the Minimax algorithm with guidance when search depth limits prevent complete game tree exploration. An effective heuristic must balance accuracy (correctly predicting position strength) with computational efficiency (evaluating quickly to minimize search overhead).

6.3.1 Scoring Methodology

The evaluation function examines all possible four-cell windows (consecutive sequences) across the board in horizontal, vertical, and both diagonal directions. Each window receives a score based on its contents, with higher values assigned to configurations that indicate stronger positions. The total board evaluation represents the sum of all window scores, adjusted for positional considerations such as center column control.

- Four pieces aligned: +100 points (terminal winning position)
- Three pieces with one empty: +5 points (strong immediate threat)
- Two pieces with two empty: +2 points (potential future threat)
- Opponent three pieces with empty: -4 points (defensive imperative)
- Center column occupation: +3 points per piece (strategic advantage)

12. Development Team

Project Timeline: December 2025

Academic Context: Computer Science Department, Artificial Intelligence Course Project

12.2 Contributions

The development process involved collaborative efforts across all project phases including requirements analysis, architectural design, algorithm implementation, user interface development, testing and validation, and comprehensive documentation. Each team member contributed expertise in specific technical domains while maintaining awareness of the overall system architecture and project objectives.

13. Conclusion

13.1 Project Summary

The Connect 4 Game System successfully demonstrates the integration of classical artificial intelligence algorithms with modern software engineering practices to create a comprehensive, fully functional gaming platform. The project achieves all stated objectives including implementation of multiple game modes, sophisticated AI with algorithmic optimization, professional user interface design, and robust system architecture supporting concurrent operations.

13.2 Technical Achievements

From a technical perspective, the project makes several notable contributions. The Minimax implementation with Alpha-Beta pruning achieves a 60-70% reduction in node evaluations compared to naive search, enabling real-time gameplay with search depths sufficient for strong play. The launcher-worker architecture demonstrates advanced concepts in process management and system design. The heuristic evaluation function effectively balances accuracy with computational efficiency, producing strategically sound AI decision-making.

13.3 Educational Value

Beyond its functional capabilities, the project serves significant educational purposes. The codebase provides a complete, working example of game AI implementation suitable for study by students and practitioners. The modular architecture illustrates software engineering best practices in large-scale system design. The comprehensive documentation facilitates understanding of both high-level concepts and implementation details, making the project valuable as a teaching resource and reference implementation.

13.4 Future Directions

Several opportunities exist for future enhancement and expansion. Network multiplayer support would enable remote gameplay across internet connections. Machine learning approaches could supplement or replace the heuristic evaluation function, potentially discovering novel strategic patterns. Additional game variants and board sizes would increase flexibility and replay value. Performance profiling and optimization could further improve computational efficiency, enabling deeper search depths and stronger play.

--- End of Document ---