

**Application of Minimax Algorithm in Dots and Boxes Game**

**BY**

**TAN AH HWA**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEMS**

**ENGINEERING**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**FEBRUARY 2025**

## COPYRIGHT STATEMENT

© 2024 Tan Ah Hwa. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of **Bachelor of Information Systems (Honours) Information Systems Engineering** at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

## **ACKNOWLEDGEMENTS**

I would like to extend my heartfelt gratitude to my supervisor, Mr. Lee Heng Yew and my moderator, Dr Ramesh Kumar Ayyasamy, for providing me with this invaluable opportunity to work on an Application of Minimax Algorithm in Dots and Boxes Game project. His guidance and insights have been instrumental in my growth and have set the foundation for my journey into the Application of Algorithms. A million thanks for your support and mentorship.

Lastly, I am deeply thankful to my parents and family for their love, encouragement, and continuous support throughout this journey.

## ABSTRACT

This project addresses the common issue of deficient artificial intelligence (AI) opponents in digital versions of the classic strategy game Dots and Boxes, which often limits gameplay engagement and strategic depth. The core problem lies in developing a strategically competent AI capable of navigating the game's large decision space and computational demands. The methodology involved implementing the Minimax algorithm as the primary decision-making engine for the AI. This was enhanced with several optimization techniques, including Alpha-Beta pruning, transposition tables, killer move heuristics, and quiescence search. For the highest difficulty setting, an iterative deepening Minimax approach was utilized alongside a heuristic evaluation function that considers score difference and strategic board positions like chains and potential opponent scoring opportunities. The research process included designing and developing a functional Dots and Boxes game prototype using C# and the Universal Windows Platform (UWP), featuring a user-friendly interface and customizable settings. Rigorous testing confirmed the application's functionality and the AI's progressive difficulty, with human players winning 70% of games on "Easy," the AI winning 50% on "Medium," and the AI achieving an 80%-win rate on "Hard". AI response times remained acceptable even on larger boards. The project successfully demonstrates the application of an enhanced Minimax algorithm to create a challenging and engaging AI opponent, effectively revitalizing the classic game by offering significant strategic depth through its advanced AI implementation and varied difficulty levels.

Area of Study (Minimum 1 and Maximum 2): **Artificial Intelligence in Gaming, Game Theory**

Keywords (Minimum 5 and Maximum 10): **Minimax Algorithm, Dots and Boxes, Artificial Intelligence, Game AI, Alpha-Beta Pruning, Heuristic Evaluation, Game Development, Strategic Games, Universal Windows Platform (UWP)**

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>COPYRIGHT STATEMENT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF SYMBOLS</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement	2
1.2 Motivation	4
1.3 Project Objectives	5
1.4 Project Scope	7
1.5 Contributions	8
1.6 Report Organization	9
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>10</b>
2.1 Review of the Technologies	10
2.1.1 Programming Language	10
2.1.2 Minimax Algorithm	12
2.1.3 Minimax Algorithm in Dots and Boxes	13
2.1.4 Summary of the Technologies Review	14
2.2 Critical Remarks of Previous Works	14
2.2.1 Strengths and Weaknesses of the Linja Game Project	14
2.2.2 Strengths and Weaknesses of the Tic-Tac-Toe Algorithmic Analysis	17
2.2.3 Compare them with my proposed solutions	20
2.3 Review of the Existing Systems/Applications	21
2.3.1 Website Dots and Boxes Games A	21

2.3.2	Website Dots and Boxes Games B	22
2.3.3	Website Dots and Boxes Games C	22
2.3.4	Summary of the Existing Systems	23
<b>CHAPTER 3</b>	<b>SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)</b>	<b>24</b>
3.1	System Design Diagram/Equation	24
3.1.1	System Architecture Diagram	26
3.1.2	Use Case Diagram and Description	28
3.1.3	Activity Diagram	31
<b>CHAPTER 4</b>	<b>SYSTEM DESIGN</b>	<b>34</b>
4.1	System Block Diagram	34
4.2	System Components Interaction Operations	40
<b>CHAPTER 5</b>	<b>SYSTEM IMPLEMENTATION (FOR DEVELOPMENT-BASED PROJECT)</b>	<b>42</b>
5.1	Software Setup	42
3.1.3	Visual Studio 2022	42
3.1.3	C# Programming Language	43
3.1.3	Universal Windows Platform (UWP)	43
3.1.3	XAML (Extensible Application Markup Language)	44
5.2	Setting and Configuration	44
5.3	System Operation (with Screenshot)	46
5.4	Implementation Issues and Challenges	54
5.5	Concluding Remark	56
<b>CHAPTER 6</b>	<b>SYSTEM EVALUATION AND DISCUSSION</b>	<b>58</b>
6.1	System Testing and Performance Metrics	58
6.2	Testing Setup and Result	60
6.3	Project Challenges	62
6.4	Objectives Evaluation	64

6.5 Concluding Remark	66
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>68</b>
7.1 Conclusion	68
7.2 Recommendation	69
<b>REFERENCES</b>	<b>71</b>
<b>POSTER</b>	<b>74</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 3.1	Game overall diagram	24
Figure 3.2	Game Architecture Diagram	26
Figure 3.3	Game use case diagram	28
Figure 3.4	Game Activity Diagram	31
Figure 4.1	Simple Overall diagram	34
Figure 4.2	Settings page diagram	35
Figure 4.3	Main Menu diagram	36
Figure 4.4	Game page diagram	38
Figure 5.1	Start Page	46
Figure 5.2	Rules Page	47
Figure 5.3	Settings Page	48
Figure 5.4	Main Menu Page	49
Figure 5.5	Gameplay and AI Operation	50
Figure 5.6	Easy Level Play	51
Figure 5.7	Medium Level Play	52
Figure 5.8	Hard Level Play	53



## LIST OF SYMBOLS

$\beta$	beta
$\alpha$	Alpha
$\infty$	Infinity
$\geq$	Less Than or Equal to
$\leq$	Greater than or equal to

## LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>GUI</i>	Graphical User Interface
<i>CPU</i>	Central Processing Unit
<i>LINQ</i>	Language Integrated Query
<i>IL</i>	Intermediate Language
<i>CLR</i>	Common Language Runtime
<i>RAM</i>	Random Access Memory
<i>GC</i>	Garbage Collection
<i>LOH</i>	Large Object Heap
<i>DFS</i>	Depth-First Search
<i>XML</i>	Extensible Markup Language
<i>XAML</i>	Extensible Application Markup Language
<i>UI</i>	User Interface
<i>2D</i>	Two-Dimensional

# Chapter 1

## Introduction

Dots and Boxes, a classic strategy game played on a grid of dots, challenges players to draw lines connecting adjacent dots, aiming to complete squares [1]. While its rules are simple – the player completing the most squares wins – the game possesses a surprising strategic depth, making it an excellent testbed for artificial intelligence (AI). The requirement for foresight and tactical planning, despite the straightforward mechanics, provides a compelling environment to explore AI decision-making algorithms.

This project focuses on enhancing the traditional Dots and Boxes experience by implementing the Minimax algorithm, a fundamental technique in AI and game theory designed for two-player, zero-sum games where players have opposing goals [2]. Minimax operates by systematically exploring the game tree, which represents all possible sequences of moves and game states [2]. By evaluating these potential outcomes, the algorithm selects the move that maximizes the AI's potential score while simultaneously minimizing the score achievable by the opponent, effectively simulating optimal play from both sides [2].

Applying Minimax to Dots and Boxes empowers the computer opponent to play strategically. By recursively evaluating future board configurations and anticipating the human player's likely optimal responses, the AI can make informed decisions [2]. This includes identifying opportunities to complete squares, strategically sacrificing moves to set up future gains (traps), blocking the opponent from completing squares, and balancing offensive and defensive maneuvers. The result is an AI capable of competing effectively, offering a significantly more engaging and challenging experience for the human player.

The main goal of this project is to develop a functional prototype of Dots and Boxes featuring an AI opponent driven by the Minimax algorithm. This prototype will serve as a practical demonstration of how AI can elevate strategic gameplay in traditional games, allowing players to interact directly with an intelligent, adaptable adversary. Furthermore, the development process will offer valuable insights into the practical challenges and opportunities involved in integrating such AI techniques into game environments.

Ultimately, this project aims to deliver a playable game that not only showcases the effectiveness of the Minimax algorithm in a strategic context but also establishes a foundation for potential future enhancements, such as incorporating more advanced AI techniques or optimizations. By successfully creating these intelligent Dots and Boxes opponent, the project contributes to the understanding of AI in gaming, demonstrating how algorithms can be used to craft more sophisticated and human-like adversaries, with potential applications extending to other strategic games and AI research domains.

### 1.1 Problem Statement

#### 1. Deficient AI Opponents Limit Gameplay Engagement and Strategic Depth.

Many readily available digital versions of the classic game Dots and Boxes suffer from underdeveloped artificial intelligence opponents. These AI players often employ simplistic strategies, such as only focusing on immediate square captures (greedy approach) or failing to anticipate more than one or two moves ahead. This lack of sophistication means the AI frequently makes strategically naive errors, fails to recognize or set up complex traps (like controlling long chains of potential squares), and doesn't adequately balance offensive opportunities with defensive necessities. Consequently, the gameplay experience often becomes predictable and insufficiently challenging, particularly for players who have moved beyond the basics and are seeking to explore the game's deeper strategic nuances. This deficiency significantly limits the game's replay ability in a single-player context, as the AI ceases to be a compelling opponent, hindering the player's ability to learn advanced tactics and ultimately failing to showcase Dots and Boxes' full potential as a rich strategic exercise. This creates a clear need for an AI that can offer a persistent and adaptive challenge.

#### 2. The Algorithmic Complexity of Developing a Strategically Competent Dots and Boxes AI.

Designing and implementing an AI agent capable of playing Dots and Boxes at a high level presents a substantial algorithmic challenge inherent to the game's structure. While the rules are straightforward, the number of possible moves (placing a line between two adjacent dots) can be considerable, leading to a large branching factor. This results in a game tree that grows

exponentially with the number of turns, making exhaustive analysis complex. Creating an AI that performs effectively requires moving far beyond simplistic approaches like random move selection or purely greedy strategies. A competent AI must exhibit genuine foresight, capable of evaluating the long-term consequences of its moves. This includes sophisticated reasoning such as balancing the immediate gain of completing a square against the potential disadvantage of opening multiple squares for the opponent, understanding the critical concept of chain control (forcing the opponent to 'open' chains of potential boxes), strategically sacrificing moves to gain positional advantage, and accurately assessing complex endgame positions. Successfully embedding this level of strategic understanding into an AI necessitates the careful application, implementation, and tuning of robust game-playing algorithms, like Minimax, capable of navigating this complex decision space.

### 3. Computational Performance Challenges in Implementing Game Tree Search for Dots and Boxes.

The practical implementation of game tree search algorithms, such as the Minimax algorithm, for Dots and Boxes is often constrained by computational performance limitations, particularly as the game progresses or when played on larger board sizes (e.g., 9x9 or greater). A naive, unoptimized Minimax implementation attempts to recursively explore every possible sequence of moves down to a certain depth or until the end of the game. Given the game's combinatorial nature, the number of game states to evaluate can quickly become astronomically large, potentially requiring significant processing time and memory resources. This computational burden can manifest as unacceptably long delays between the player's move and the AI's response, severely disrupting the natural flow of gameplay and leading to a frustrating user experience. Therefore, a critical problem is not just selecting an appropriate algorithm like Minimax, but also addressing its inherent computational cost. There is a crucial need to investigate, implement, and fine-tune optimization techniques—most notably Alpha-Beta pruning, which intelligently eliminates the need to evaluate large portions of the game tree [3], and potentially heuristic evaluation functions to estimate board value without full searches—to ensure the AI operates efficiently, provides timely responses, and remains viable even for more complex game scenarios.

### 1.2 Motivation

The fundamental motivation driving this project is the compelling opportunity to significantly enhance the classic game of Dots and Boxes by developing and integrating a sophisticated artificial intelligence opponent based on the Minimax algorithm. While Dots and Boxes is renowned for its simple rules and accessibility, its underlying strategic depth is often underserved in digital versions that feature rudimentary AI. This project is born from the desire to bridge that gap, moving beyond predictable or easily exploitable opponents to create a genuinely engaging and intellectually stimulating challenge for players of all skill levels. The goal is to transform the single-player experience from a basic pastime into a dynamic contest of wits, where the AI opponent demonstrates foresight, tactical awareness, and the ability to execute complex strategies like setting up sacrifices and controlling critical chains, thereby increasing player engagement and promoting a deeper appreciation for the game's subtleties.

Furthermore, this endeavor provides a valuable platform to explore the practical application of core AI game-playing principles within the unique constraints and tactical landscape of Dots and Boxes. It offers a chance to investigate how a well-established algorithm like Minimax, typically discussed in the context of games like Chess or Tic-Tac-Toe, adapts to the specific mechanics of completing squares and managing line placements. This exploration involves delving into how to best represent the game state, designing effective evaluation functions that capture positional advantages beyond simple box counts, and observing the emergent strategic behaviors produced by the algorithm. Successfully applying Minimax in this context serves as a practical case study in AI problem-solving within combinatorial games.

Intrinsic to this exploration is the motivation derived from tackling the inherent technical challenges associated with implementing game tree search algorithms efficiently. The project confronts the computational demands of Minimax, particularly the potential for state-space explosion on larger grids, which can render a naive implementation impractically slow. Overcoming this involves the intellectually stimulating task of researching, implementing, and fine-tuning optimization techniques such as Alpha-Beta pruning and potentially developing effective heuristics. Successfully navigating these performance hurdles is key to demonstrating how theoretical AI concepts can be translated into a responsive, functional, and genuinely enjoyable interactive system, showcasing practical AI engineering alongside algorithmic understanding.

Ultimately, the project aims to culminate in a fully playable Dots and Boxes prototype featuring a competent Minimax-driven AI. This tangible outcome serves multiple purposes: it acts as a clear demonstration of the algorithm's effectiveness in enhancing strategic gameplay, provides users with a challenging and rewarding experience, and validates the design and implementation choices made. Beyond its immediate function, this prototype is envisioned as a foundation—a robust starting point for potential future explorations, such as incorporating more advanced AI techniques (like machine learning enhancements), expanding game features, or adapting the framework for educational purposes or application to other strategic board games. This project, therefore, is motivated not just by improving a single game but by contributing a well-realized example of AI in action within the accessible and engaging domain of classic strategy games.

### 1.3 Project Objectives

1. To develop and implement the core Minimax algorithm as the decision-making engine for an artificial intelligence opponent within a Dots and Boxes game.

This objective involves the detailed design and coding of the fundamental AI logic. It requires creating a robust internal representation of the Dots and Boxes game state, capable of accurately tracking the grid configuration, which lines have been drawn, the ownership of completed squares, the current scores, and whose turn it is. Central to this objective is the implementation of the recursive Minimax function itself, which must correctly alternate between maximizing the AI's score on its turn and minimizing the human opponent's score on their simulated turns [2]. This function needs to effectively explore the game tree by generating all valid successor states (possible line placements) from any given state, evaluating terminal states (win, loss, draw, or end-of-game based on completed squares), and propagating the calculated utility values back up the tree. The ultimate output of this core component will be the reliable selection of the move deemed most advantageous according to the Minimax principle, forming the strategic heart of the AI player and enabling it to play with foresight rather than just reacting greedily.

2. To integrate performance optimization techniques and variable difficulty levels into the Minimax AI.

## CHAPTER 1

Building upon the core Minimax implementation, this objective addresses the practical challenges of computational complexity and user experience. Recognizing that a full Minimax search can become excessively slow on larger boards or at greater depths, this involves implementing crucial optimization techniques. Primarily, this includes integrating Alpha-Beta pruning, a method designed to significantly reduce the number of nodes evaluated in the game tree by eliminating branches that cannot possibly influence the final decision [4], thereby drastically speeding up move calculation without sacrificing the accuracy of the outcome. Additionally, this objective may involve developing and incorporating heuristic evaluation functions to assess non-terminal game states, allowing the AI to make informed decisions even when a full search to the end of the game is infeasible. Furthermore, this objective explicitly includes implementing the user-selectable "Computer Difficulty" setting. This will likely involve mechanisms such as varying the maximum search depth of the Minimax algorithm (deeper searches lead to stronger, but potentially slower, play), adjusting the complexity or accuracy of heuristic functions, or potentially introducing controlled randomness at lower difficulty settings, ensuring the game provides an appropriate and adaptable level of challenge for a diverse range of players.

3. To create a complete, functional, and user-friendly prototype of the Dots and Boxes application incorporating the Minimax AI.

This objective focuses on delivering the tangible product: a fully operational game application that seamlessly integrates the AI opponent developed in the previous objectives. It encompasses the entire scope of the user-facing software, starting with the development of an intuitive graphical user interface (GUI) that visually represents the game board, scores, and available moves clearly. A key aspect is implementing the described navigation structure, ensuring users can move effortlessly and logically between the Start Page, Main Menu, Game Page, Settings, and Rules sections. This objective also mandates the functional implementation of all specified user customization options: toggling sound effects and background music, selecting from available music tracks, applying different visual themes (checkerboard colours), choosing various board sizes, and configuring the game setup, including the number of human players and the AI's difficulty level. Crucially, this objective ensures the successful integration of the optimized Minimax AI (from Objectives 1 and 2) into the game loop, allowing a human



player to interactively play a complete game against the computer opponent through the developed interface, from initial setup to the final determination and display of the game's outcome.

### 1.4 Project Scope

The scope of this project centers on the development of a functional, playable prototype of the Dots and Boxes game featuring a competent artificial intelligence opponent powered by the Minimax algorithm. This includes the complete implementation of the Minimax logic for AI decision-making, incorporating state evaluation, recursive game tree exploration, and move selection based on maximizing the AI's score while minimizing the opponent's [2]. Furthermore, the scope explicitly includes the integration of performance optimizations, such as Alpha-Beta pruning or heuristic evaluations, to ensure efficient AI operation, along with the implementation of variable computer difficulty levels accessible to the user.

The project scope also encompasses the creation of a complete user application framework. This involves developing a graphical user interface with intuitive navigation between distinct sections: a Start Page, Main Menu, Game Page, Settings screen, and a Rules display area, following the specific interaction flows described. Essential game setup and customization features fall within scope, including user selection of board size, player configuration, AI difficulty, and visual theme (choice of three checkerboard colors). User-specific preferences, namely the ability to enable/disable sound effects and background music (with three track options), are also included within the Settings section. The implementation will cover all core Dots and Boxes game mechanics, such as turn-based line drawing, square completion detection, scoring, and handling of game end conditions.

Conversely, the project scope is strictly defined to exclude elements not central to the core AI and gameplay demonstration. Specifically, the development of advanced or high-fidelity graphics is outside the scope; the focus will remain on functional representation rather than aesthetic polish. The project will not implement any online multiplayer capabilities, confining gameplay to local sessions on a single device. Additionally, the exploration and implementation of artificial intelligence algorithms other than Minimax (and its direct optimizations like Alpha-Beta pruning or heuristics) are explicitly excluded, as the primary

goal is to demonstrate the application and effectiveness of the Minimax algorithm within the Dots and Boxes context.

### 1.5 Contributions

A Functional and Optimized Minimax AI for Dots and Boxes is the primary contribution of this project, involving the successful implementation and adaptation of the Minimax algorithm, potentially enhanced with Alpha-Beta pruning or heuristics, specifically tailored for the strategic complexities of Dots and Boxes. This results in an AI opponent capable of demonstrating genuine foresight, balancing offensive and defensive tactics, recognizing traps, and providing a significantly more challenging and engaging gameplay experience compared to simpler AI implementations commonly found in this game. The AI serves as a practical realization of game theory principles within this specific domain.

This project also contributes an enhanced and Configurable Dots and Boxes Game Prototype, which is a complete, playable software application integrating the Minimax AI within a user-friendly framework. Beyond basic gameplay, this prototype offers tangible enhancements including variable AI difficulty levels, user customization options (such as board size, visual themes, sound effects, and background music), and a clear, navigable user interface connecting the start screen, main menu, game board, settings, and rules. This integrated system serves as a polished demonstration platform showcasing the AI in action and providing a richer user experience.

Furthermore, the project provides Practical Insights and a Demonstration of AI applications in a Classic Game. It serves as a valuable case study on the practical application of a fundamental AI algorithm (Minimax) to a classic strategy game. It demonstrates how theoretical AI concepts can be translated into a working interactive system, highlighting both the effectiveness of the algorithm in creating intelligent behavior and the practical challenges faced during implementation, particularly regarding computational performance and the need for optimization. The resulting prototype and the knowledge gained during its development can serve educational purposes, illustrate AI principles to a wider audience, and potentially act as a foundation for future research or extensions involving more advanced AI techniques or game features.

### 1.6 Report Organization

Chapter 1 serves as the Introduction. This chapter likely details the background of the Dots and Boxes game, introduces the Minimax algorithm, states the problem the project aims to solve, outlines the motivation behind the project, lists the project objectives, defines the scope of the work, and highlights the contributions of the project. It concludes with the report's organization.

Chapter 2 is the Literature Review. This section is expected to cover a review of the technologies used, such as the C# programming language and the Minimax algorithm, including its application in Dots and Boxes. It also provides critical remarks on previous works, comparing them with the proposed solutions, and reviews existing Dots and Boxes systems or applications.

Chapter 3 details the System Methodology or Approach for this development-based project. This would typically include system design diagrams like flowcharts or architecture diagrams to explain how the system is structured and how its components interact.

Chapter 4 focuses on the System Design. This chapter likely elaborates on the system block diagram and the interaction operations between different system components.

Chapter 5 covers System Implementation. This section would describe the software setup, system settings and configurations, and the system's operation, potentially with screenshots. It also discusses implementation issues and challenges faced during development and offers a concluding remark on the implementation phase.

Chapter 6 is dedicated to System Evaluation and Discussion. This chapter would outline the system testing procedures and performance metrics used, detail the testing setup and results, discuss project challenges, and evaluate how well the project objectives were met. It concludes with a summary of the evaluation.

Finally, Chapter 7 provides the Conclusion and Recommendations. This chapter summarizes the project's achievements and offers recommendations for future work or enhancements. The report also includes a list of References.

## Chapter 2

### Literature Review

#### 2.1 Review of the Technologies

##### 2.1.1 Programming Language

C# is a modern, versatile programming language developed by Microsoft, designed with developer productivity and application scalability in mind. It features static and strong typing, alongside support for object-oriented and component-oriented programming paradigms [5]. Running primarily on the .NET framework, C# is employed across a wide spectrum of applications, including full-stack desktop applications, mobile apps, game development and others [6]. Its extensive libraries, cross-platform capabilities enabled by .NET, and features like exception handling make it a popular choice, consistently ranking among the top programming languages [5]. Key C# features contributing to its utility include Language Integrated Query (LINQ) for data manipulation, a structured approach to exception handling, and functional programming elements like lambda expressions [5]. C# code is typically compiled into an intermediate language (IL) and executed within the .NET Common Language Runtime (CLR) [7]. Code executed under the CLR's management is referred to as "managed code," benefiting from services such as security checks, type safety, and, critically for this review, automatic memory management through Garbage Collection (GC) [7].

A cornerstone feature of the .NET framework and C# is its automatic memory management system, the Garbage Collector (GC) [7]. The GC serves as an automatic memory manager within the CLR, relieving developers of manually allocating and deallocating memory [7]. This automation frees developers from writing explicit memory release code, allocates objects efficiently on the managed heap, and automatically reclaims memory from objects no longer in use [8]. When a .NET application starts, the CLR reserves a contiguous block of virtual memory known as the managed heap, where all reference-type objects are allocated [9]. Allocation is generally fast, comparable to stack allocation, as the runtime maintains a pointer to the next available address and advances it upon object creation, often resulting in contiguous storage, which can improve data locality [7]. The GC determines object liveness by examining application "roots" (static fields, local variables, CPU registers, GC handles, finalization queue

objects) and constructing a graph of all reachable objects [7]. Objects not part of this graph are considered garbage [7]. The collection process typically involves marking reachable objects, relocating/compacting the heap by moving live objects together to reduce fragmentation (updating references accordingly), and sweeping/releasing the memory of unreachable objects [9]. Objects exceeding 85,000 bytes are allocated on the Large Object Heap (LOH), which is usually not compacted due to the cost of moving large blocks, though compaction mechanisms exist in later .NET versions [8].

To optimize performance, the GC employs a generational approach based on the observation that most objects are short-lived [7]. The managed heap is logically divided into three generations: Generation 0 holds the youngest, typically short-lived objects and is collected most frequently [7]. Generation 1 holds objects that survived a Gen 0 collection, acting as a buffer and collected less frequently and Generation 2 holds long-lived objects that survived Gen 1, collected least frequently via a "full garbage collection" which includes all generations and the LOH [7]. Objects surviving a collection in a lower generation are promoted to the next higher one, and a collection of a higher generation always includes objects from lower generations [7]. Garbage collection is triggered automatically by the CLR based on conditions like low physical memory signalled by the OS, allocated memory on the managed heap exceeding a dynamically adjusted threshold, or an explicit call to `GC.Collect()` (though manual triggering is generally discouraged) [8].

For game development, automatic memory management via GC offers benefits in productivity and code safety, allowing developers to focus on game logic rather than manual memory management, thus reducing development time and eliminating bugs like memory leaks and dangling pointers [7]. However, GC operations can introduce pauses or "hiccups" as they may require suspending application threads, which can be disruptive in real-time games needing smooth frame rates [10]. The non-deterministic timing of GC adds to this challenge [8]. Developers using C# for games must adopt strategies to minimize GC impact, such as minimizing allocations within game loops, using object pooling (reusing objects instead of creating new ones), preferring value types (structs) where appropriate to avoid heap allocations, avoiding known garbage-generating operations like excessive string concatenation, strategically using `GC.Collect()` during non-critical times (e.g., loading screens) and utilize memory profiling tools in Visual Studio [11]. Modern .NET GCs have optimizations like

generational and background collection to mitigate pauses, but proactive memory management remains crucial for optimal game performance [10].

### 2.1.2 Minimax Algorithm

The Minimax algorithm is a fundamental decision-making algorithm from game theory, applied in AI for programming game-playing agents, especially for two-player, turn-based games with perfect information and opposing goals (zero-sum games) like Tic-Tac-Toe, Chess, and Dots and Boxes [12]. Its core principle is to find the optimal move by assuming the opponent will also play optimally to counter [12]. It seeks to maximize the player's minimum guaranteed outcome, minimizing the maximum possible loss the opponent can inflict [12]. Minimax involves two conceptual players: the Maximizer (Max), typically the AI, aiming for the highest score, and the Minimizer (Min), the opponent, aiming for the lowest score for Max [12]. The game is modeled as a game tree where nodes represent game states and edges represent moves [12]. The algorithm explores this tree using recursive depth-first search (DFS), going down paths until a terminal state or a predefined depth limit is reached [2]. A utility function assigns a definitive score to terminal states (e.g., +10 for Max win, -10 for Min win, 0 for draw), while a heuristic evaluation function estimates the desirability of non-terminal states reached at the search depth limit, based on game features [2]. The core calculation involves propagating these values up the tree: Min nodes take the minimum value of their children, and Max nodes take the maximum value of their children [12]. The value calculated for the root node represents the best score Max can guarantee, and the optimal move leads to the child node with this value [12]. However, Minimax has exponential time complexity, approximately  $O(b^d)$  (where  $b$  is the branching factor,  $d$  is the depth), making full searches infeasible for complex games [2]. Practical implementations must limit search depth, relying on the heuristic function's accuracy at the limit, making the chosen move an approximation of the true optimal move.

Given Minimax's computational cost, Alpha-Beta pruning is an essential optimization technique applied to the Minimax search [13]. It significantly reduces the number of nodes evaluated without changing the final move chosen by Minimax [13]. It works by maintaining bounds on achievable scores: Alpha ( $\alpha$ ), the best (highest) score found so far for the Maximizer (initially  $-\infty$ ), and Beta ( $\beta$ ), the best (lowest) score found so far for the Minimizer

(initially  $+\infty$ ) [13]. These values are passed down the tree and updated [13]. Pruning occurs when beta becomes less than or equal to alpha ( $\beta \leq \alpha$ ) [13]. At a Min node, if the current best score for Min ( $\beta$ ) becomes  $\leq$  the alpha inherited from an ancestor Max node, the remaining children are pruned because Max already has a better option (guaranteeing at least  $\alpha$ ) [13]. At a Max node, if the current best score for Max ( $\alpha$ ) becomes  $\geq$  the beta inherited from an ancestor Min node, the remaining children are pruned because Min already has a better option (guaranteeing at most  $\beta$ ) [13]. In the best case (optimal move ordering), Alpha-Beta can reduce complexity towards  $O(b^{d/2})$ , allowing significantly deeper searches [14]. However, its efficiency heavily depends on exploring the best moves first; poor move ordering can degrade performance close to basic Minimax [14]. Therefore, implementing effective move ordering heuristics is crucial [14]. Alpha-Beta pruning finds the same optimal move as Minimax (at a given depth  $d$ ) but explores only a subset of nodes, making it faster [14].

### 2.1.3 Minimax Algorithm in Dots and Boxes

Applying Minimax/Alpha-Beta to Dots and Boxes is suitable as it's a two-player, deterministic, perfect-information, zero-sum game [15]. The AI (Maximizer) aims to maximize its captured boxes minus the opponent's (Minimizer) [15]. The main challenge is the enormous state space. Even a 3x3 grid has a vast number of configurations (estimated around  $10^{15}$ ), making a full search impossible and necessitating depth-limited search and optimizations [15]. Efficient state representation is critical. Options include a 2D grid/list of lists representing dots, links, and boxes, custom data structures like Edge and Box classes, bitboards for fast updates and hashing, or the conceptual "strings-and-coins" analogy [16]. The choice impacts the feasibility of optimizations like transposition tables (requiring fast hashing) and symmetry detection [16]. The evaluation function is pivotal due to the depth limit. Terminal state evaluation is simple (AI boxes - Human boxes). The heuristic function for non-terminal states must estimate the final score difference, considering strategic nuances beyond immediate captures [16]. Factors might include current score difference (though potentially limited early on), immediate captures, control over "chains" (sequences of boxes with two open sides, crucial for endgame strategy), number of available safe moves, and number of boxes with 2 lines (potential chain elements) [16]. A common technique tracks a single zero-sum score (incrementing for AI captures, decrementing for human captures) [15]. The "extra turn" rule (player captures a box,

moves again) must be handled correctly in the state transition logic and the simulation continues with the same player until a move doesn't capture a box [15]. Essential optimizations beyond Alpha-Beta include depth limiting, transposition tables (caching results of previously seen states using hashing), symmetry handling (treating rotationally/reflectionally equivalent states identically), and move ordering heuristics (prioritizing likely strong moves like captures or chain-avoiding moves) [16].

### **2.1.4 Summary of the Technologies Review**

In summary, this review covered C# with its automatic garbage collection, the Minimax algorithm optimized by Alpha-Beta pruning, and the specific challenges of applying these to Dots and Boxes (state space, representation, heuristics, extra turns, optimizations). C# offers productivity via features like GC, but GC pauses require mitigation in games [8]. Minimax provides a theoretical basis for optimal play, but its complexity necessitates depth limits and Alpha-Beta pruning [12]. Implementing this for Dots and Boxes demands careful handling of its large state space, strategic heuristics (especially chain control), state representation supporting optimizations like transposition tables, and the extra turn rule. Next, the key takeaways for implementation include managing memory consciously to minimize GC impact, implementing optimized Minimax with Alpha-Beta and depth limiting, prioritizing heuristic design focusing on strategic elements like chains, and choosing state representation wisely for efficiency and optimizations.

## **2.2 Critical Remarks of Previous Works**

### **2.2.1 Strengths and Weaknesses of the Linja Game Project**

Using the Minimax algorithm to increase the strategic depth of gaming, the Linja game project is a noteworthy application of game theory concepts [20]. This report will look at the project's advantages and disadvantages, giving a fair assessment of its accomplishments and potential for development.



### **Strengths**

The Linja game project demonstrates several key strengths, notably its impressive application of game theory concepts, particularly the minimax algorithm [20]. This algorithm is fundamental to the development of an intelligent system capable of selecting optimal moves within the game. By enabling the AI to minimize potential losses, the minimax algorithm facilitates strategic play, thereby enhancing its performance against human opponents and showcasing a profound understanding of strategic decision-making [20].

Another significant strength is the project's comprehensive framework [20]. The documentation thoroughly outlines the game's rules, the data structures employed, and the specifics of the algorithm's implementation. This detailed introduction ensures that the project's scope and technical components are clearly understood, which contributes to a better comprehension of the AI's functionality and the overall effectiveness of the system [20].

The project also excels in its focus on user interface design [20]. The inclusion of a graphical user interface (GUI) significantly improves user interaction and makes the game more accessible. Furthermore, the project considers varying difficulty levels for the AI, allowing for a more personalized and enjoyable gaming experience tailored to different player skill levels. This emphasis on user interface design reflects a commitment to creating a dynamic and user-friendly application [20].

Moreover, the project incorporates an extensive experimental validation method, which is crucial for assessing the AI's efficacy [20]. Through rigorous testing and analysis of outcomes, the project demonstrates the AI's performance against human players. This empirical approach not only validates the AI's capabilities but also supports the project's overall conclusions. The detailed analysis of results offers valuable insights into the AI's performance and identifies areas for potential improvement [20].

Finally, the document displays a forward-thinking approach by considering future work [20]. By outlining potential enhancements and modifications for broader platforms, the project demonstrates a proactive stance on software development. This willingness to explore further improvements signifies a dedication to continuous innovation and the ongoing development of the game beyond its initial release [20].

### **Weaknesses**

Bachelor of Information Systems (Honours) Information Systems Engineering  
Faculty of Information and Communication Technology (Kampar Campus), UTAR

## CHAPTER 2

Despite its strengths, the Linja game project faces challenges, primarily due to the complexity of its implementation [20]. Games like Linja, with a vast number of potential moves, can be computationally intensive and strain system resources. The intricacy of managing and accessing a wide range of game states can present significant programming hurdles and affect the AI's overall efficiency [20].

Another weakness lies in the potentially limited scope of testing scenarios [20]. Although the project's testing is described as extensive, it might not encompass all conceivable game dynamics. This limitation could mean the AI's learning capacity is not fully robust, as its performance might vary when confronted with unexpected or diverse game situations. Broadening the range of tests could yield a more comprehensive assessment of the AI's performance [20].

The project's reliance on heuristic evaluations for decision-making presents both advantages and disadvantages [20]. While heuristics simplify the analysis of complex game states, they can sometimes lead to suboptimal decisions. Heuristic-based methods may oversimplify game dynamics, potentially limiting the AI's ability to handle more intricate game scenarios effectively [20].

Furthermore, user experience variability poses another challenge [20]. Differences in player skill levels and strategic approaches can impact the AI's learning process, potentially leading to inconsistent gameplay experiences. This unpredictability might affect the AI's capacity to adapt and perform optimally across a diverse range of player interactions [20].

Finally, scalability concerns may arise as the game evolves or new features are introduced [20]. Maintaining performance and managing new complexities might necessitate considerable modifications to the existing architecture. Ensuring the system's ability to scale effectively while preserving usability and performance is a critical consideration for future development [20].

The application of game theory, extensive framework, user interface design, experimental validation, and consideration of future work are only a few of the strengths that the Linja game project highlights [20]. It also has issues with scalability, user experience unpredictability, reliance on heuristic evaluation, limited testing scenarios, and complexity of implementation

[20]. Progressing the project and guaranteeing its sustained success in providing an interesting and strategic gaming experience would need to address these shortcomings while capitalizing on its strengths [20].

### **2.2.2 Strengths and Weaknesses of the Tic-Tac-Toe Algorithmic Analysis**

Significant insights into how computational strategies might improve gameplay and decision-making processes are gained from the study of algorithms in game artificial intelligence [21]. The reviewed study provides a thorough comparative examination of multiple algorithms, with a particular emphasis on the optimization of the minimax algorithm using Alpha-Beta pruning [21]. This essay looks at the paper's advantages and disadvantages, noting its contributions to the field of game AI research and suggesting avenues for development [21].

#### **Strengths**

A salient feature of the study is its comprehensive analysis of several algorithms, with the thorough examination of the minimax algorithm and its optimization using Alpha-Beta pruning standing out as a particularly insightful contribution [21]. This detailed comparison allows readers to understand the respective benefits and drawbacks of each algorithm concerning decision-making and operational efficiency. The research offers valuable insights into the functionality of each algorithm within the Tic-Tac-Toe context, and this understanding can be extrapolated to a broader range of strategic games [21].

The decision to utilize Tic-Tac-Toe as the primary platform for algorithmic experimentation is commendable due to its clarity [21]. As a well-known and straightforward game, Tic-Tac-Toe enables readers, irrespective of their background in game theory or artificial intelligence, to grasp the fundamental concepts more easily. This familiar context facilitates the paper's ability to clearly demonstrate AI algorithm performance without the obfuscation of complex game rules, allowing readers to focus on the efficacy of the strategies rather than the intricacies of the game itself [21].

The work also excels in situating its findings within the broader context of current AI research [21]. By referencing prior studies and methodologies, the paper demonstrates a comprehensive

understanding of the prevailing state of AI and game development. This contextualization helps validate the study by placing its conclusions in relation to existing literature and illustrating how the findings contribute to the ongoing discourse on game AI, making the work significant not only for its algorithmic insights but also for its contribution to advancing knowledge on AI applications in gaming [21].

Furthermore, the article's thorough analysis of the intricacies of different algorithms is a significant asset [21]. The comparison between brute force methods and Alpha-Beta pruning is particularly useful, as it clearly demonstrates the differences in operational efficiency between various tactics. These insights are valuable for both researchers and developers, offering precise guidance on selecting appropriate algorithms based on the specific requirements of a game. This focus on algorithmic efficiency positions the study as a useful resource for anyone aiming to enhance AI performance in gaming scenarios [21].

Finally, the paper's conclusions carry important practical implications, especially for game developers seeking to integrate AI into their creations [21]. The thorough examination of how AI can enhance user experience and influence gameplay is crucial for understanding the real-world applications of these algorithms. The research underscores how AI algorithms improve strategic play and decision-making, thereby highlighting the relevance of its findings to actual game development situations [21].

### **Weaknesses**

While Tic-Tac-Toe serves as a useful tool for clarity, its use also limits the applicability of the study's results [21]. Although the relatively simple game of Tic-Tac-Toe is excellent for illustrating fundamental AI tactics, it may not adequately represent the complexities encountered in more intricate gaming environments. The simplicity of the game means that the findings might not readily apply to games with more extensive decision trees or deeper strategic elements. Consequently, while the paper's conclusions are valuable, their applicability to more complex AI applications may be constrained [21].

The extensive technical analysis presented in the paper could pose a challenge for individuals not well-versed in algorithms or computer science [21]. The technical jargon and intricate mathematical explanations of the minimax algorithm and Alpha-Beta pruning might be

difficult to follow for those unfamiliar with these subjects. As a result, the paper's accessibility to a broader audience may be limited, potentially alienating readers who lack the necessary technical background. Incorporating clearer explanations or illustrative examples could help bridge this gap [21].

Although the work provides a solid theoretical framework, it would benefit from the inclusion of more real-world examples or empirical data [21]. While a theoretical understanding of algorithmic performance is useful, the paper's findings could be strengthened by incorporating actual tests or case studies. Demonstrating the algorithms in action through simulations or gaming scenarios would offer concrete proof of their efficacy, thereby reinforcing the arguments and connecting them to practical applications [21].

The paper's apparent preference for specific methods, such as Alpha-Beta pruning, raises concerns about potential bias [21]. While this algorithm has recognized advantages, the criteria for judging it against other algorithms are not always clearly defined. A more comprehensive comparison that impartially evaluates a wider range of algorithms would provide a more balanced view of the AI landscape. Without this, the study risks presenting a skewed perspective that might overlook the merits of other viable alternatives [21].

Furthermore, the analysis in this work primarily focuses on static algorithmic performance, neglecting dynamic elements like evolving strategies or player behaviour [21]. In real-world game scenarios, player behaviour can significantly influence outcomes, potentially requiring AI tactics to adapt. By concentrating solely on the technical aspects of algorithmic efficiency, the research overlooks the importance of adaptive AI capable of adjusting to shifting game dynamics. A discussion of how these algorithms could account for player behaviour would deepen the research and enhance its relevance to practical scenarios [21].

In the context of Tic-Tac-Toe, the study provides a thorough and perceptive analysis of AI algorithms, with a focus on the minimax algorithm and its Alpha-Beta pruning optimization [21]. Its thorough analysis, understandable application, context for the research, algorithmic insights, and useful consequences are its strongest points [21]. The article does, however, have many drawbacks, such as its limited scope, technical difficulty, and absence of empirical data, potential bias, and emphasis on static analysis [21]. The article may gain greater traction and

provide more insightful information about the dynamic and changing nature of AI in game development if these flaws were fixed [21].

### **2.2.3 Compare them with my proposed solutions**

#### **1. Comparison with the Linja Game Project**

The Minimax algorithm was used by the Linja game project to generate a clever AI opponent [20]. The application of game theory, a thorough framework, an intuitive interface, and experimental validation are among its main advantages [20]. Unfortunately, scalability issues, reliance on heuristic judgments, limited testing scenarios, and technical complexity all hindered it [20].

#### **Comparison**

Both the Linja project and the proposed Dots and Boxes solution focus on integrating the minimax algorithm to enhance AI decision-making [20]. However, my project adds a layer of game adaptability, where the AI adjusts its strategy dynamically based on the player's actions. This addresses Linja's limitations in testing scenarios and adaptation, offering a more responsive AI.

Linja's reliance on heuristics could lead to suboptimal decisions [20], which is mitigated in Dots and Boxes through the implementation of Alpha-Beta pruning to optimize the Minimax algorithm, reducing unnecessary calculations without sacrificing accuracy. The proposed project also places additional emphasis on balancing AI competency to avoid overly simplistic or unbeatable opponents, improving user experience.

#### **2. Comparison with Tic-Tac-Toe Algorithmic Analysis**

The Tic-Tac-Toe analysis concentrated on contrasting optimizations like Alpha-Beta pruning with algorithms like Minimax [21]. Detailed algorithmic comparison, useful ramifications for game production, and algorithmic insights are some of its strong points [21]. However, it was

devoid of empirical evidence, offered a static analysis, and seemed to be biased in favour of some techniques over others, such as Alpha-Beta pruning [21].

### **Comparison**

The suggested Dots and Boxes approach has the same ability to optimize performance by utilizing Alpha-Beta pruning as Tic-Tac-Toe [21]. It goes one step further by putting the AI to the test in a variety of challenging scenarios and dynamic game situations to make sure it can adjust to changing player tactics [21]. This offers a more flexible, adaptive AI experience, addressing the static analysis weakness of Tic-Tac-Toe analysis [21].

In contrast to Tic-Tac-Toe, which emphasizes a more straightforward game with fewer strategic facets, Dots and Boxes offers a more intricate choice space, demanding a more thorough analysis of move selection. The drawback of limited empirical data in the Tic-Tac-Toe analysis is overcome by thoroughly testing a variety of game scenarios to address this complexity [21].

To sum up, while other projects have effectively shown the Minimax algorithm's potential in games, the suggested Dots and Boxes approach improves these by emphasizing flexibility, balanced difficulty, and strategic complexity, giving players a more engaging experience.

## **2.3 Review of the Existing Systems/Applications**

### **2.3.1 Website Dots and Boxes Games A [17]**

Existing Games A represents a robust and feature-rich online implementation of the classic Dots and Boxes game. Its primary strength, as noted by users, lies in the significant challenge presented by its artificial intelligence, particularly when players engage with the highest difficulty setting. This suggests a well-developed game engine, potentially utilizing sophisticated algorithms to drive computer play. The system thoughtfully caters to a diverse audience by offering a granular selection of four distinct difficulty levels. This range allows both novices seeking a gentle introduction and experienced strategists looking for a demanding

contest to find an appropriate level of challenge, thereby enhancing replay ability. Beyond the core single-player experience against the computer, System A acknowledges the social aspect of gaming by providing functionality for users to compete directly against friends. Furthermore, it empowers users with significant control over their gaming environment through multiple customization options. Players can precisely define the complexity and length of a match by selecting the dimensions of the game board. They can also personalize the visual experience by choosing a preferred theme color for the interface and configure the game for different group sizes by specifying the number of players, making it a highly adaptable and user-centric platform.

### **2.3.2 Website Dots and Boxes Games B [18]**

Existing Games B is another iteration of Dots and Boxes available as a web-based application. User feedback highlights that engaging with this version's computer opponent provides a stimulating and challenging experience, implying the integration of a competent AI that requires thoughtful strategic play to overcome. However, System B diverges significantly from others in its approach to difficulty configuration. A key characteristic and notable limitation of this system is the complete absence of any user-selectable difficulty levels. The game appears to operate on a single, fixed level of AI competence. While this fixed level is perceived as challenging, this lack of adjustability means the game may not be suitable for beginners, finding it too difficult, or experts seeking an even greater challenge or varied gameplay. Consequently, user control over the game's parameters is restricted primarily to the ability to customize the size of the playing grid. While board size selection allows some control over game dynamics, the overall flexibility and adaptability of the experience are considerably less than systems offering explicit difficulty scaling and other customization features.

### **2.3.3 Website Dots and Boxes Games C [19]**

This third online Dots and Boxes game, Game C, offers players a simplified choice regarding game difficulty, providing only two explicit options: an 'Easy' level and a 'Hard' level. While offering a choice is beneficial, user experience indicates a potential weakness in its AI implementation. Specifically, players have reported that triumphing over the computer opponent, even when set to the designated 'Hard' level, feels relatively straightforward and less



demanding compared to the higher difficulty settings found in System A or the default challenge presented by System B. This suggests that the algorithms governing the AI's strategy in System C might be less advanced or perhaps deliberately tuned to offer a more casual, less intimidating gameplay experience. Despite this perceived limitation in the AI challenge, System C incorporates essential features for usability and social play. Like System A, it supports matches against the computer as well as contests between human players (friends). Additionally, it provides the standard, yet crucial, functionality for users to select the desired dimensions of the game board, allowing for variation in game length and complexity.

### **2.3.4 Summary of the Existing Systems**

In evaluating these three existing Dots and Boxes systems, a clear hierarchy in terms of feature depth, customization, and AI challenge becomes apparent. Game A emerges as the most comprehensive offering. It successfully combines a challenging AI, particularly at its peak setting, with extensive user control, featuring four difficulty levels and options to modify board size, theme colour, and player numbers, alongside multiplayer capabilities. This makes it suitable for a wide range of players seeking both challenge and personalization. Game B presents a somewhat paradoxical experience: it delivers a notably challenging AI opponent but severely restricts user control by omitting any means to select or adjust difficulty. Its customization is essentially limited to board size, potentially alienating players who prefer graduated difficulty or find the fixed level unsuitable. Game C offers a basic structure with two difficulty levels ('Easy', 'Hard'), multiplayer support, and board size selection. However, its key drawback is the perceived lack of genuine challenge even on its 'Hard' setting, positioning it more towards casual players or those new to the game rather than users seeking a demanding strategic duel. While all three permit board size adjustments, and Systems A and C facilitate multiplayer games, the crucial differences lie in the sophistication and scalability of the AI opponent and the overall degree of user customization afforded.

## Chapter 3

### System Methodology/Approach

#### 3.1 System Design Diagram

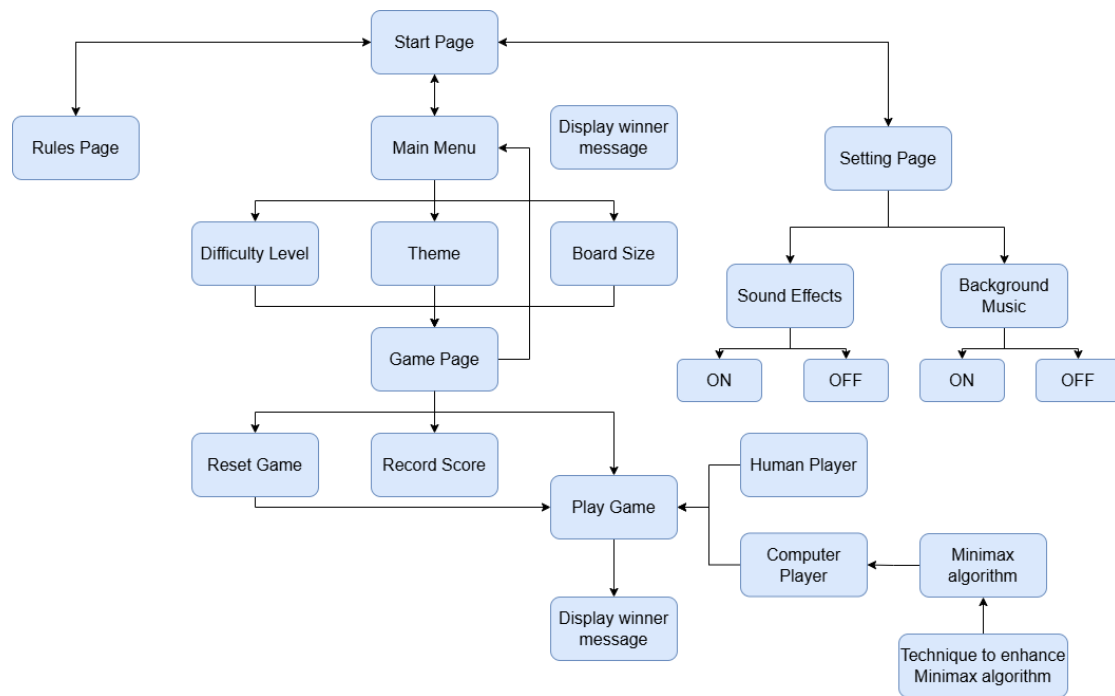


Figure 3.1 Game overall diagram

In Figure 3.1, the application's system design centers around a user interface that allows seamless navigation between several key sections. The entry point is the **Start Page**, which acts as a hub providing access to the **Main Menu**, **Rules page**, and **Settings page**. The **Rules page** provides game instructions and allows navigation back to the **Start Page**. The **Settings page** offers customization options and permits users to return to the **Start Page** or **Main Menu**.

The **Main Menu** serves as a central navigation point, allowing users to go back to the **Start Page**, proceed to the **Game Page**, or access the **Setting Page**. Within the **Main Menu**, users can configure game parameters such as select from three theme colours (default, light, blue), set the computer's difficulty level, and choose the board size. These selections directly influence the gameplay experience on the **Game Page**.

## CHAPTER 3

The Settings page provides controls for audio customization. Users can toggle sound effects on or off, affecting auditory feedback during gameplay and operations. Additionally, background music can be enabled or disabled, with three different background music options available for selection. These settings allow users to tailor the game's audio environment to their preferences.

The core gameplay occurs on the Game Page. This component manages the user interface for the Dots and Boxes game, dynamically generating the game board (dots, lines, and squares) based on parameters received from the Main Menu (grid size, difficulty, theme). It manages human player input via pointer events for drawing lines and provides visual feedback. The game page also initializes and interacts with the artificial intelligence module for computer opponents. It updates the visual state of the board based on moves from both human and AI players, plays sound effects, calculates scores, and manages turn transitions. If a player completes a square, they get another turn. The page continuously checks for game end conditions (win, loss, or draw based on total score versus available squares) and displays the outcome, disabling further board interaction. A menu button on the Game Page allows resetting the current game or returning to the Main Menu. It also handles background video playback and applies selected visual themes.

The AI's logic is encapsulated within a dedicated module. This component implements an AI opponent with "easy", "medium", and "hard" difficulty levels, which adjust the search depth of its core Minimax algorithm. The Minimax algorithm is enhanced with alpha-beta pruning for efficient game tree exploration. To further optimize performance and decision-making, the AI utilizes a transposition table (caching previously evaluated board states), a killer moves heuristic (prioritizing moves that caused pruning), and a quiescence search (extending search for capture sequences to mitigate the horizon effect). The AI evaluates board positions using a heuristic function that considers score difference, penalizes moves creating immediate scoring opportunities for the opponent (three sides of a box open), and discourages moves creating chains of boxes with two sides. For the "hard" difficulty, iterative deepening is employed, progressively increasing search depth within a time limit (which adjusts based on grid size) to find the best possible move. The AI simulates moves on internal board copies to determine consequences. The game page retrieves the current visual board state, converts it to an internal representation of the board state, and asynchronously calls the AI module's function to determine the best move. The AI's chosen move is then rendered on the UI by the game page. The system design emphasizes a clear separation between the UI/game flow management and

the AI's decision-making logic, facilitating a customizable and challenging gameplay experience.

### 3.1.1 System Architecture Diagram

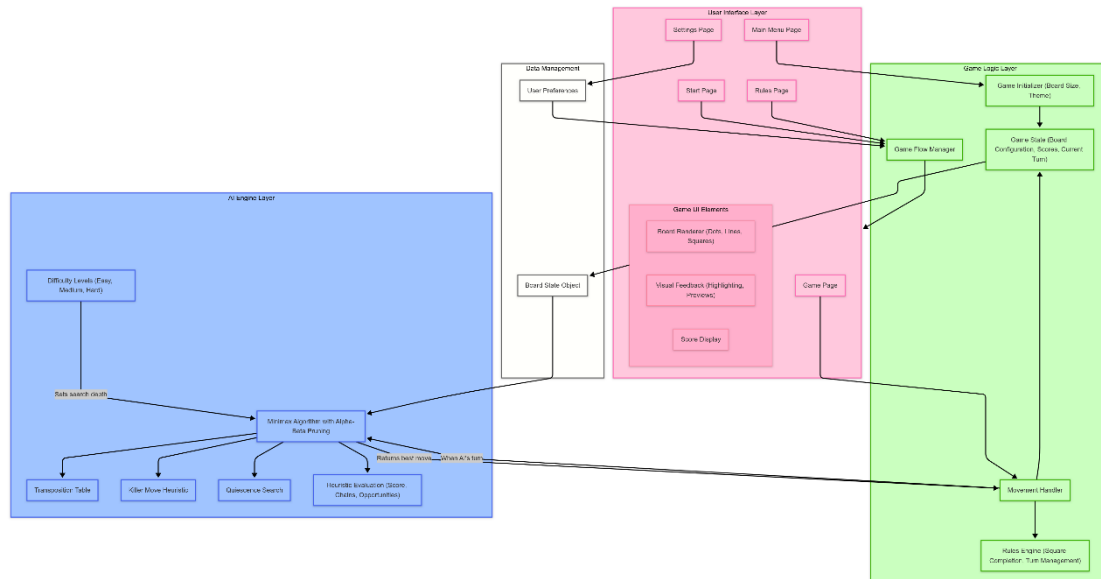


Figure 3.2 Game Architecture Diagram

The system architecture in Figure 3.2 for the Dots and Boxes game application is designed as a component-based structure, likely following a layered approach to separate concerns and enhance modularity. This architecture ensures a clear distinction between user interaction, game management, and artificial intelligence, facilitating an intuitive and engaging gameplay experience.

At the forefront is the User Interface (UI) Layer, which serves as the primary point of interaction for the user. This layer is responsible for presenting all visual elements and handling user input across various sections of the application. It comprises several distinct views or pages: the Start Page (initial entry point), the Rules Page (displaying game instructions), the Settings Page (for audio and other preferences), the Main Menu Page (for game configuration), and the Game Page (where actual gameplay occurs). The Game Page component is particularly dynamic, responsible for rendering the game board with its dots, lines, and squares, and providing real-time visual feedback like dot highlighting and line previews during a player's move.

## CHAPTER 3

Interfacing with the UI Layer is the Game Logic Layer (or Application Layer). This layer acts as the central coordinator of the application. It manages the overall game flow, including navigation between the different UI pages based on user actions. Critically, it maintains the Game State, which includes the current configuration of the board (lines drawn, squares completed), player scores, and whose turn it is. This layer implements the fundamental rules of Dots and Boxes, such as awarding points and an extra turn upon completing a square and switching turns. When a game starts, this layer uses parameters selected by the user in the Main Menu (like board size, AI difficulty, and visual theme) to initialize the game environment. The component managing the game screen and player interaction within this layer is also responsible for orchestrating communication between the user's actions on the UI and the AI engine when it's the computer's turn. It handles events like valid moves, updates scores, and checks for game termination conditions.

The AI Engine Layer provides the intelligence for the computer opponent. This component is centred around a Minimax search algorithm, which is enhanced with alpha-beta pruning for efficiency. To further refine its decision-making and performance, the AI engine incorporates several advanced techniques: a transposition table to cache results of previously evaluated board states, a killer move heuristic to prioritize promising moves, and a quiescence search to stabilize evaluations in volatile (capture-heavy) positions. The AI's behaviour is scalable through different difficulty levels ("easy," "medium," "hard"). These levels primarily adjust the depth of its search algorithm: "easy" and "medium" use fixed, shallower search depths, while "hard" employs iterative deepening up to a greater depth, constrained by time management based on grid size. The AI evaluates board positions using a sophisticated heuristic function that considers score differences, potential opponent scoring opportunities, and the strategic implications of creating chains. It simulates potential moves on internal copies of the board state before selecting its optimal move.

Underpinning these layers are Data Management aspects. The Game Logic Layer manages a representation of the current game state (referred to as a BoardState object when interacting with the AI), which includes all details about drawn lines and completed squares. User preferences from the Settings page are also implicitly managed. The AI Engine maintains its own data structures, most notably the transposition table for caching board evaluations.

The interactions between these layers are well-defined. User input from the UI Layer is processed by the Game Logic Layer, which updates the game state and/or navigates the UI. When it's the AI's turn, the Game Logic Layer provides the current game state to the AI Engine. The AI Engine then asynchronously calculates its best move and returns it to the Game Logic Layer, which in turn updates the game state and directs the UI Layer to reflect the AI's move on the board. This architecture ensures that the game's presentation, core logic, and AI decision-making are decoupled, leading to a robust and maintainable system.

### 3.1.2 Use Case Diagram and Description

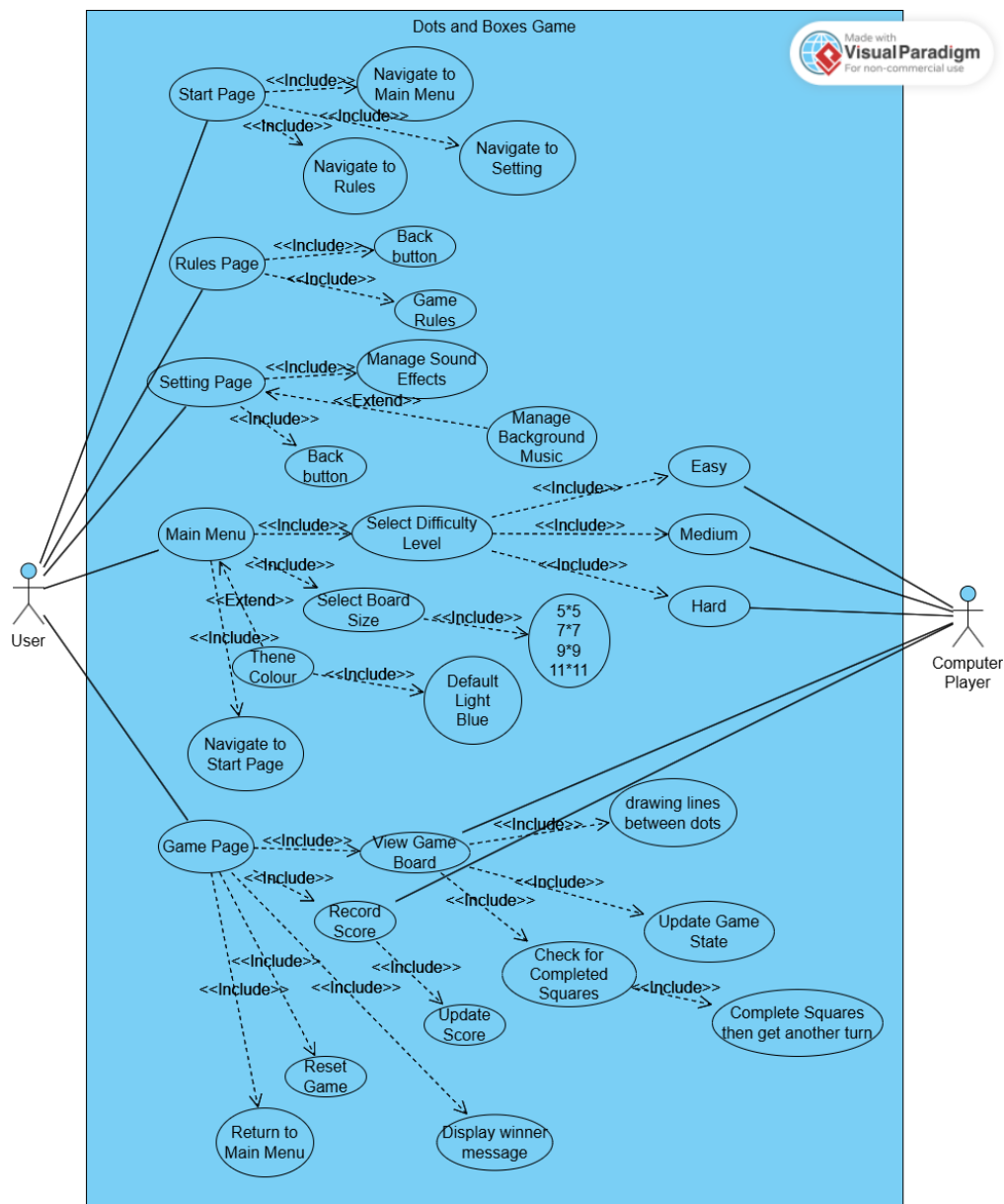


Figure 3.3 Game use case diagram

## CHAPTER 3

In Figure 3.3, the system facilitates several interactions for the User, who is the primary actor. The User begins at the Start Page, from which they can initiate several use cases: Navigate to Main Menu, Navigate to Rules, and Navigate to Settings. The Navigate to Rules use case allows the User to view game instructions and then Navigate to Start Page.

The Navigate to Settings use case allows the User to access and modify application preferences. Within the Settings, the User can Manage Sound Effects (Enable/Disable) and Manage Background Music (Enable/Disable). If enabling background music, the User can also Select Background Music Track from three available options. From Settings, the User can Navigate to Start Page or Navigate to Main Menu.

The Navigate to Main Menu use case, accessible from the Start Page, allows the User to configure a new game. In the Main Menu, the User can Select Number of Players, Select Computer Difficulty (with options for Easy, Medium, or Hard, which influences how the AI Actor plays), Select Board Size, and Select Theme Colour (Default, Light, or Blue). After configuring, the User can Navigate to Game Page to start playing. The Main Menu also allows the User to Navigate to Start Page.

Once on the Game Page, the User can View Game Board. The primary gameplay use case for the User is Make a Move, which involves drawing lines between dots. The System (or a Game Management component) will then Update Game State, Check for Completed Squares, Update Score accordingly, and Play Sound Effects (if enabled by the User in Settings). If the User completes a square, they get another turn; otherwise, the System will Switch Turns. The User can also, from a menu on the Game Page, Reset Game (restarting with the same configurations) or Return to Main Menu.

The AI Player, acting as a secondary actor or system component, primarily engages within the Game Page. When it is the AI's turn, its central task is to "Determine AI Move." The sophistication of this process is directly influenced by the difficulty level selected by the User.

For the Easy Difficulty setting, the AI employs a Fixed-Depth Minimax search algorithm, exploring the game tree to a shallow depth of 2 plies. This search is optimized through several techniques, including Alpha-Beta Pruning to cut off branches that won't influence the outcome, a Transposition Table to cache and retrieve evaluations of previously seen game states, Move Ordering to prioritize more promising moves, the Killer Moves heuristic to try moves that have

## CHAPTER 3

been effective in other parts of the search tree, and a Quiescence Search to ensure stability in tactical situations by extending the search for capture sequences.

At the Medium Difficulty level, the AI's foresight is extended. It utilizes a Fixed-Depth Minimax search to 4 plies, allowing for a more comprehensive evaluation of potential moves. This deeper search continues to benefit from the same suite of enhancements employed in the Easy difficulty: Alpha-Beta Pruning, a Transposition Table, Move Ordering, Killer Moves, and Quiescence Search, all of which become even more crucial for managing the larger search space.

The Hard Difficulty presents the most formidable AI opponent. Here, the AI shifts to an Iterative Deepening Minimax strategy, starting with a shallow search and progressively increasing the depth up to a maximum of 10 plies. This approach is dynamically constrained by Time Management, ensuring the AI decides within a set timeframe. Throughout each iteration of its search, the AI heavily relies on Alpha-Beta Pruning, a Transposition Table, powerful Move Ordering techniques, the Killer Moves heuristic, and a Quiescence Search at the leaf nodes to refine its evaluations.

Once the AI has determined its move, the System takes over to integrate it into the game. This involves updating the Game State to reflect the AI's action, checking if the move resulted in any Completed Squares, updating the AI's Score accordingly, and playing any relevant Sound Effects if enabled. If the AI's move successfully completes one or more squares, the AI is granted another turn. If no square is completed, the turn switches back to the User.

Throughout the game, the System is responsible for Defining Game State initially and after each move. It also handles Evaluating Actions based on the Minimax algorithm (for the AI), which involves maximizing the AI's score and minimizing the opponent's score. Finally, the System will Handle Game End Conditions, determining if the game is a win, loss, or draw based on final scores, and then Display Game Outcome to the User.



## 3.1.3 Activity Diagram

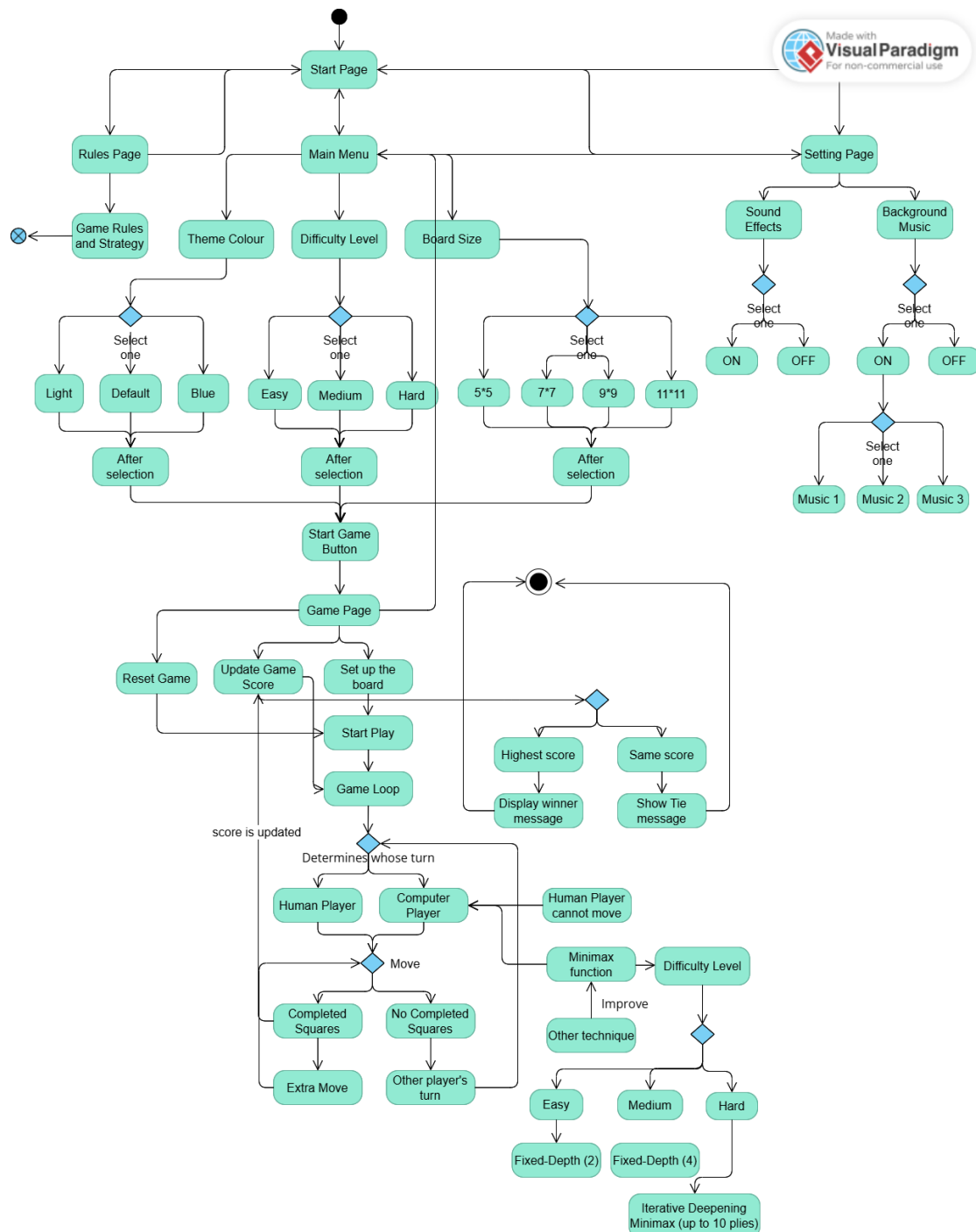


Figure 3.4 Game Activity Diagram

In Figure 3.4, the flow begins when the user launches the application, landing on the Start Page. From this initial activity, a decision node allows the user to navigate to different sections: the Main Menu, the Rules page, or the Settings page. If the user selects Rules, the system displays the game rules, after which the user typically returns to the Start Page. If Settings is chosen, the user enters a settings configuration flow. Here, they can perform actions such as enabling

## CHAPTER 3

or disabling sound effects and background music. A conditional flow exists for background music: if enabled, the user is presented with three options to choose from. After adjusting settings, the user can navigate back to the Start Page or Main Menu.

Should the user navigate to the Main Menu (either from the Start Page or Settings), they engage in game setup activities. These include selecting the Computer Difficulty (Easy, Medium, or Hard – a critical input that alters subsequent AI behaviour), selecting the Board Size, and picking a Theme Colour (default, light, or blue). Once these parameters are set, the flow transitions to initializing and starting the game on the Game Page. Options to return to the Start Page or access Settings directly from the Main Menu would also be available.

The core Gameplay activity begins with setting up the board and defining the initial game state. This leads into a loop that continues until game-end conditions are met. Inside the loop, a decision point determines whose turn it is: Human Player or AI Player.

If it's the Human Player's turn, the UI enables input, allowing the player to select dots to draw a line. Visual feedback is provided during this action. Upon completion of a valid move, the system updates the visual state of the board and plays a sound effect (if enabled). Following this, the system "Checks for Completed Squares." If a square is completed, the human player's score is updated, and they get another turn (looping back to the start of their turn). If no square is completed, the turn switches to the AI Player.

The process for the AI Player's turn begins with the disabling of human input. The system then retrieves the current state of the board and initiates the AI's move determination process. This is a sophisticated sub-activity where the Minimax algorithm plays a central role. The AI defines and utilizes a Minimax function, which is enhanced by Alpha-Beta pruning and various heuristics. This function evaluates potential actions with the dual goals of maximizing the AI's score while simultaneously minimizing the opponent's score. The precise manner in which this evaluation and move selection occurs varies significantly based on the difficulty level chosen by the user.

For the Easy Difficulty setting, the AI employs a Fixed-Depth Minimax search, exploring the game tree to a relatively shallow depth of 2 plies. This search is augmented by several techniques to improve efficiency and decision quality, including a transposition table to store and recall evaluations of previously encountered board states, a killer move heuristic to

## CHAPTER 3

prioritize moves that have proven effective elsewhere in the search, and a quiescence search to ensure more stable evaluations in volatile positions. Notably, at this level, iterative deepening and primary time management strategies are not utilized.

When set to Medium Difficulty, the AI performs a more extensive Fixed-Depth Minimax search, extending its analysis to 4 plies. It continues to use the same set of enhancements as the Easy level, such as the transposition table, killer move heuristic, and quiescence search. Similar to the Easy setting, iterative deepening and primary time management are not key components of its decision-making process at this difficulty.

At the Hard Difficulty level, the AI utilizes a more advanced Iterative Deepening Minimax approach, allowing it to search progressively deeper, potentially up to 10 plies, within the constraints of an allotted time for its turn. This strategy is heavily supported by a transposition table, the killer move heuristic, and a quiescence search. The AI also simulates potential moves on internal copies of the board state to assess their consequences before committing to the best one. Once the AI's move is selected, the system updates the user interface to reflect this action, plays a sound effect (if enabled), and then proceeds to update the game state and check for any completed squares. In a manner identical to the human player's turn, if the AI successfully completes one or more squares, its score is updated, and it is granted another turn. If no square is completed, the turn transitions back to the Human Player.

After each turn, regardless of the player, the system will check for game end conditions. If the game is not over, the gameplay loop continues. If game end conditions are met (e.g., all squares are filled), the system determines the winner (Win, Loss, or Tie), displays a game over message, and disables further interaction with the game board. From this end state, the user can select options such as "Reset Game" (restarting with current settings) or "Return to Main Menu."

## Chapter 4: System Design

### 4.1 System Block Diagram

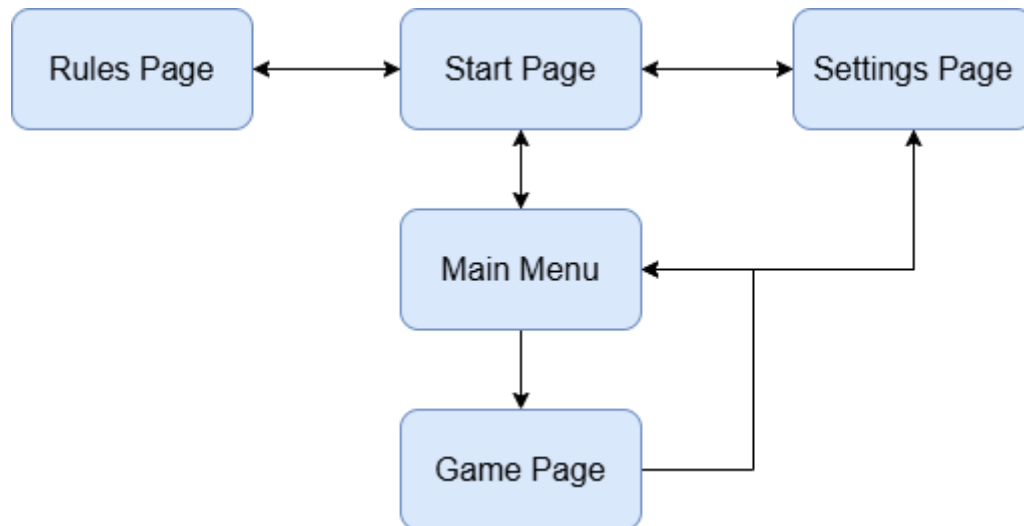


Figure 4.1 Simple Overall diagram

In Figure 4.1, the user interface of the application is designed to facilitate seamless navigation between various sections, ensuring an intuitive experience for users. At the forefront is the Start Page, which serves as the initial entry point. From this page, users can access three key areas: the Main Menu, Rules, and Settings.

Navigating to the Main Menu allows users to explore further options within the application. This menu not only provides a pathway back to the Start Page but also leads to the Game Page, where the core gameplay takes place. Additionally, users can return to the Settings from the Main Menu, enabling them to adjust their preferences before diving into the game.

The Rules section is accessible directly from the Start Page, providing users with essential information about how to play the game. Once users have familiarized themselves with the rules, they can easily return to the Start Page to continue exploring the application.

The Settings area offers flexibility, allowing users to navigate back to the Start Page or Main Menu. This ensures that users can make adjustments to their preferences without losing their place within the application.

Finally, the Game Page is designed for active gameplay, but users can also access the Settings from here or return to the Start Page. This interconnected structure enhances user experience by providing multiple pathways for navigation, making it easy for users to switch between different sections based on their needs.

In summary, the application's user interface is thoughtfully structured to provide a fluid navigation experience, allowing users to move effortlessly between the Start Page, Main Menu, Rules, Settings, and Game Page. This design not only promotes ease of use but also encourages users to engage fully with the application.

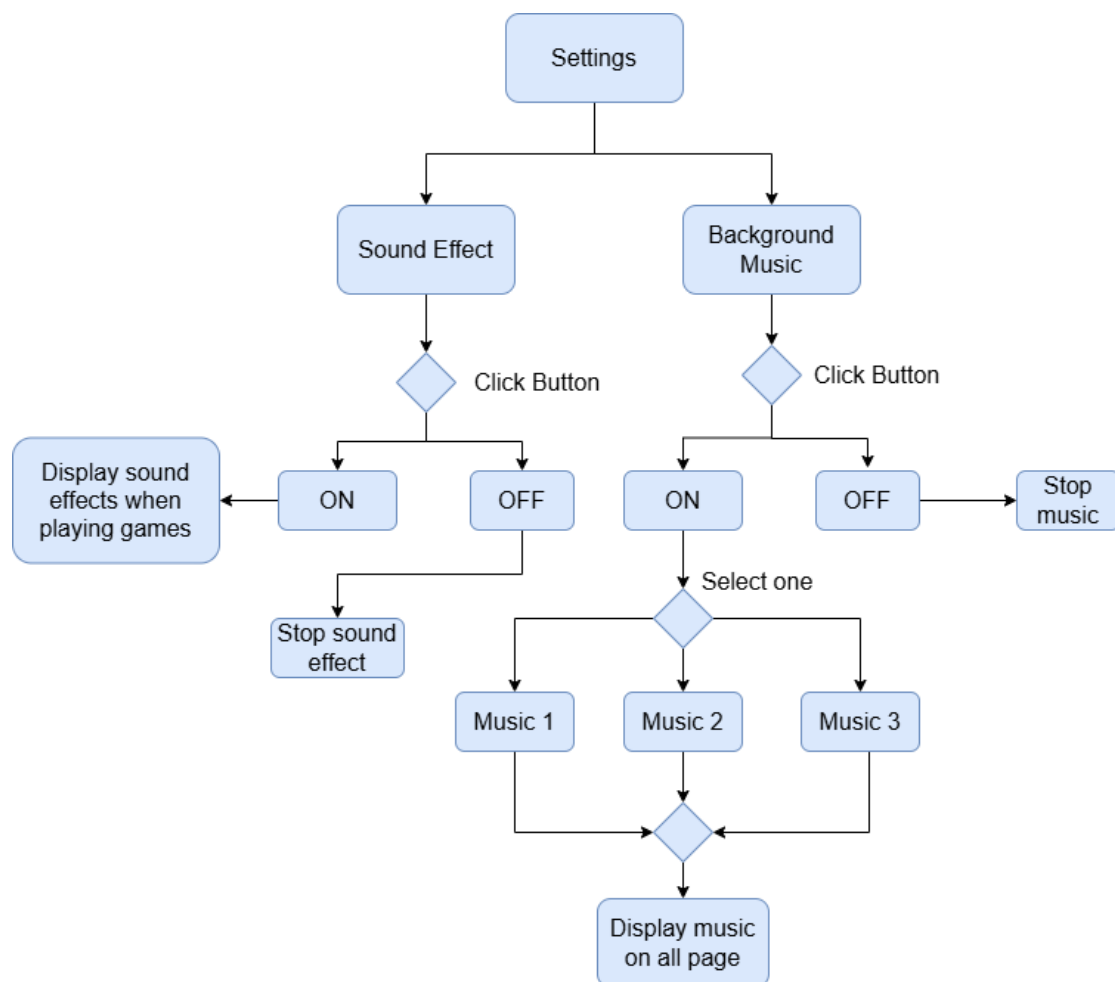


Figure 4.2 Settings page diagram

In Figure 4.2, one of the primary features in Settings is the option to enable or disable sound effects. When the sound effects are turned on, users will hear auditory feedback during

gameplay and operations, enriching their interaction with the game. Conversely, if the sound effects are turned off, these auditory cues will not play, allowing for a quieter experience that some users may prefer.

In addition to sound effects, the Settings also allow users to control background music. Users can choose to have the background music play when it is enabled, providing an engaging atmosphere that complements the gameplay. However, if a user opts to turn off the background music, it will not play, ensuring that the gaming experience remains tailored to their preferences.

In conclusion, the Settings section plays a crucial role in personalizing the user experience by offering flexible audio options. Whether users prefer a fully immersive soundscape or a more subdued atmosphere, the ability to customize sound settings ensures that every player can enjoy the game in their unique way.

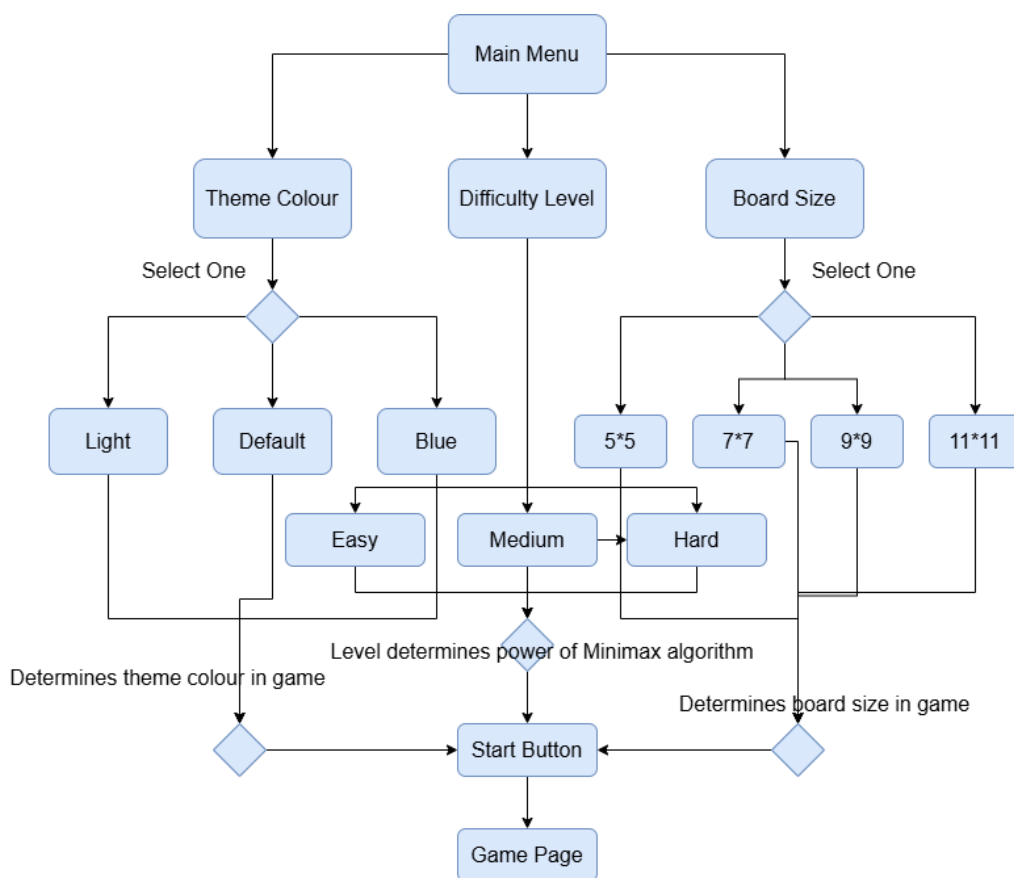


Figure 4.3 Main Menu diagram

## CHAPTER 4

In Figure 4.3, the Main Menu of the application features several interactive elements designed to enhance gameplay by allowing users to customize their gaming experience. Among these elements are the combo boxes for selecting the theme colour, computer difficulty, and board size, each of which plays a crucial role in shaping how the game unfolds.

The Theme option enables users to determine the colour of the board in the game. Users can select from various configurations, including default, light and blue colour. This flexibility allows for a tailored gaming environment that can accommodate different preferences.

In conjunction with player selection, users can also set the Computer Difficulty. This feature is particularly important for those who opt to include computer players in their games. By choosing the appropriate difficulty level, users can ensure that the computer opponents provide a suitable challenge. Whether a user is a novice seeking a more relaxed experience or an experienced player looking for a formidable adversary, this setting allows for a balanced and enjoyable gameplay experience.

Additionally, the Board Size selection allows users to customize the scale of the game. Depending on the complexity and length of the game they desire, users can choose from various board sizes. A larger board may offer more strategic possibilities and a longer gameplay duration, while a smaller board can lead to quicker matches and more immediate action.

In summary, the Main Menu's options for theme colour, computer difficulty, and board size are integral to creating a tailored gameplay experience. By providing these choices, the application ensures that every player can engage with the game in a way that suits their style and preferences, ultimately leading to a more satisfying and immersive gaming experience.

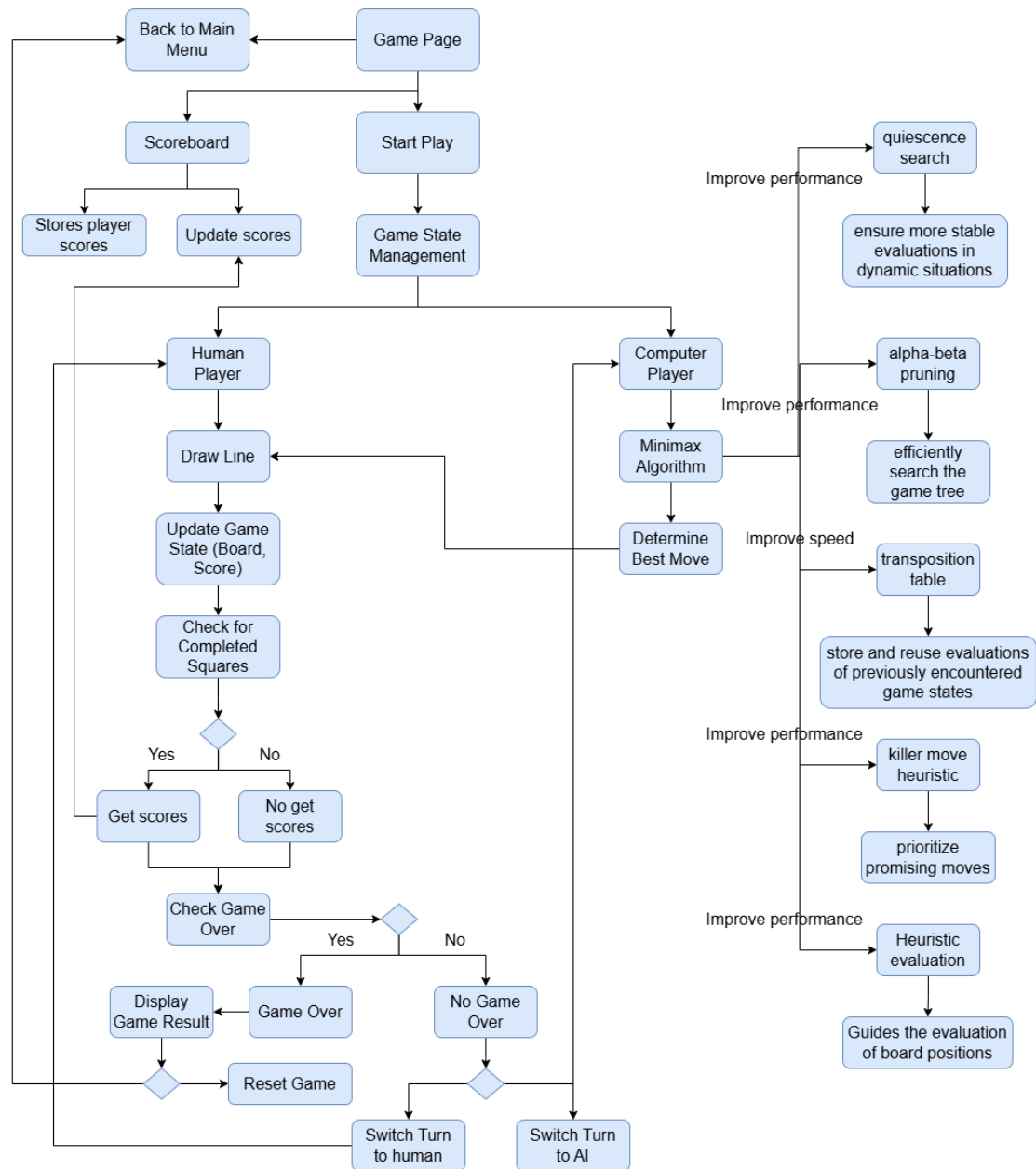


Figure 4.4 Game page diagram

In Figure 4.4, the Game Page System Block Diagram illustrates the flow of gameplay and the interaction between the human player and the computer AI. The process initiates with the user navigating to the "Game Page," which triggers the "Start Play" state. This involves "Game State Management," where the initial board configuration is set up, and the game begins. The diagram clearly delineates the turn-based nature of the game, branching into "Human Player" and "Computer Player" turns.



## CHAPTER 4

During the "Human Player" turn, the user interacts with the UI to "Draw Line." This action leads to "Update Game State (Board, Score)," where the game board is visually updated with the drawn line, and the scoreboard is potentially updated. Following this, the system "Check for Completed Squares." If squares are completed ("Yes" branch), the "Get scores" block is activated to award points to the human player, and the flow returns to "Check for Completed Squares" to see if any further squares were completed by the same move, adhering to the game rule that a player continues their turn if they complete a square. If no squares are completed ("No" branch), the turn switches to the "Computer Player." After checking for completed squares (regardless of whether any were completed), the system proceeds to "Check Game Over." If the game is over ("Yes" branch), the "Game Over" state is reached, and the "Display Game Result" is shown before offering the option to "Reset Game," which likely takes the user back to a state where a new game can be started, possibly from the Main Menu. If the game is not over ("No" branch), the turn switches to the appropriate player ("Switch Turn to human" or "Switch Turn to AI").

The "Computer Player" turn involves the "Minimax Algorithm" to "Determine Best Move." The diagram highlights several techniques that enhance the Minimax algorithm's performance and decision-making. "Alpha-beta pruning" improves performance by efficiently searching the game tree, reducing the number of nodes to evaluate. A "transposition table" enhances speed by storing and reusing evaluations of previously encountered game states, avoiding redundant computations. The "killer move heuristic" further improves performance by prioritizing moves that have historically led to cutoffs in the search tree, making the search more efficient. "Quiescence search" ensures more stable evaluations in dynamic situations, particularly near the end of capture sequences, by extending the search beyond the regular depth limit for potentially volatile moves. Finally, "Heuristic evaluation" guides the evaluation of board positions, providing a score for non-terminal states that the Minimax algorithm uses to compare different moves. After the AI determines its best move, the game state is updated ("Update Game State (Board, Score)"), and the system checks for completed squares, similar to the human player's turn, before checking for game over and switching turns if necessary. The "Scoreboard" element is shown to interact with both human and computer turns by "Update scores" and also "Stores player scores." The "Back to Main Menu" option provides a way to exit the current game and return to the application's main navigation.

### 4.2 System Components Interaction Operations

The Dots and Boxes application features several interconnected components that work together to deliver the game experience. The user's journey begins at the Start Page, which acts as a central hub, providing access to the Main Menu, Rules, and Settings. Navigation between these sections is seamless, allowing users to move back and forth intuitively. From the Start Page, a user can delve into the Rules to understand the game mechanics and then return to the Start Page. Alternatively, they can enter the Settings to customize their preferences, such as enabling or disabling sound effects and background music, and selecting from three background music options. Within the Settings, users can navigate to the Game Page, the Start Page, or the Main Menu, offering flexibility in adjusting configurations. The Main Menu itself presents options to return to the Start Page, proceed to the Game Page to start playing, or revisit the Settings. Crucially, the Main Menu also allows users to customize their gameplay experience by selecting the computer's difficulty level (easy, medium, hard), choosing the board size, and setting the theme colour from default, light, and blue.

The core gameplay unfolds on the Game Page. Initially, the game board is set up, and the game enters the turn-based phase. The `GamesPage.xaml.cs` file is instrumental in managing the user interface and game flow on this page. It dynamically generates the visual representation of the board, including dots, initially hidden lines, and transparent squares, based on the grid size received from the Main Menu. It also initializes an instance of the `AIPlayer.cs` based on the selected difficulty level. Human players interact with the board by clicking and dragging between adjacent dots to draw lines. The `GamesPage.xaml.cs` handles these pointer events, providing visual feedback during the drag operation. Upon completing a valid move, the corresponding line becomes visible, its colour changes, and a sound effect plays. The system then checks if any squares have been completed. If so, the square's appearance is updated, the human player's score increases, and they get another turn. If no square is completed, the turn switches to the AI player.

When it's the AI's turn, the `GamesPage.xaml.cs` disables human input and retrieves the current visual state of the board, passing it to the `AIPlayer.cs`. The `AIPlayer.cs` then employs its internal logic, based on the selected difficulty, to determine the best move using the Minimax algorithm, potentially enhanced with alpha-beta pruning, a transposition table, a killer move heuristic, and quiescence search. For the "hard" difficulty, it also uses iterative deepening with time

management. This calculation is performed asynchronously to prevent UI freezes. Once the `AIPlayer.cs` returns the chosen move, the `GamesPage.xaml.cs` updates the UI, accordingly, drawing the AI's line, playing a sound, and again checking for completed squares, updating scores and turns as necessary. The game continuously monitors for game end conditions by comparing the total score to the number of available squares. Upon game completion, a message indicating the winner (or a tie) is displayed, and further interaction with the board is disabled. Additionally, the Game Page includes a menu button that allows the player to reset the current game or return to the Main Menu, and it also manages background video playback and applies visual themes based on the settings.

The `AIPlayer.cs` component encapsulates the artificial intelligence logic. It receives the current board state from the `GamesPage.xaml.cs` and returns the AI's chosen move. The complexity of the AI's decision-making process varies significantly based on the difficulty level. On the "easy" setting, the AI performs a shallow, fixed-depth (2 plies) Minimax search with alpha-beta pruning, a transposition table, move ordering, killer moves, and a quiescence search for immediate captures. The "medium" difficulty extends the fixed-depth search to 4 plies, leveraging the same optimization techniques more extensively. The "hard" difficulty employs a more sophisticated iterative deepening Minimax approach, progressively increasing the search depth within a given time limit (time management), and utilizes alpha-beta pruning, a transposition table, move ordering, killer moves, and quiescence search. The AI's evaluation function considers the score difference, penalizes moves that create immediate opportunities for the opponent, and discourages moves that lead to chains of two-sided boxes. Throughout all difficulty levels, the AI simulates potential moves on internal copies of the board to predict their outcomes without affecting the actual game state displayed on the Game Page.

## Chapter 5: System Implementation

### 5.1 Software Setup

The foundation of this project rests upon a carefully selected suite of software tools and platforms, each chosen for its specific strengths in facilitating modern application development. This setup is designed to ensure a streamlined workflow, robust functionality, and a high-quality end-user experience. The core components include a state-of-the-art integrated development environment, a versatile and powerful programming language, a comprehensive development framework for Windows applications, and a declarative markup language for crafting the user interface.

#### 5.1.1 Visual Studio 2022

Visual Studio 2022 that furnishes the necessary tools for developing, debugging, and testing Universal Windows Platform (UWP) applications using C# [22]. It is widely recognized for its extensive array of features specifically designed for Windows app development, which include a user-friendly interface, sophisticated debugging tools, and integrated support for multiple programming languages [22].

Visual Studio 2022 is relevant to projects as it facilitates the seamless integration of diverse development components, encompassing both front-end (UI) and back-end elements like game logic and AI [22]. Its integrated XAML designer is crucial for crafting the graphical interface for applications such as a "Dots and Boxes" game, offering drag-and-drop capabilities for arranging the layout and visual components [22]. Furthermore, the debugging feature is instrumental in identifying and resolving errors during runtime, which ensures the proper functioning of AI logic and that the game operates as anticipated [22]. The IDE also incorporates version control and collaboration tools, simplifying the process of tracking code modifications and providing rollback options [22]. Additionally, Visual Studio offers robust testing environments for UWP applications, enabling the game to be evaluated across various device form factors and screen dimensions, such as desktops and tablets [22].

### 5.1.2 C# Programming Language

C# is a contemporary, object-oriented programming language developed for constructing a diverse range of applications, such as games, desktop applications, and web services, among others [24]. As a component of the Microsoft .NET ecosystem, it is particularly well-suited for Windows app development [24].

The relevance of C# to a project, such as developing a "Dots and Boxes" game, is multifaceted. It is utilized for scripting the game logic, including the rules of "Dots and Boxes," and for implementing algorithms like minimax. The language's structured methodology and inherent support for recursion make it an excellent choice for managing the recursive characteristics of the minimax algorithm [25]. Furthermore, C#'s extensive graphics libraries, for instance, DirectX for UWP, are instrumental in generating fluid animations, drawing lines between dots, and visually emphasizing completed squares [23]. The event-driven programming paradigm of C# is vital for processing user inputs, like clicks on dots, and for initiating AI responses in real-time [26]. Additionally, C#'s memory management capabilities ensure that the game operates efficiently, thereby preventing memory leaks or performance degradation as the game unfolds [23].

### 5.1.3 Universal Windows Platform (UWP)

The Universal Windows Platform (UWP) serves as a development framework that empowers developers to build applications capable of running on all Windows devices, such as desktops, tablets, Xbox, and HoloLens, using a singular codebase [27]. This framework streamlines cross-device compatibility, guaranteeing that the application can scale and operate optimally on any device [27].

UWP's relevance to a project is significant as it facilitates the creation of a responsive interface that adjusts to varying screen sizes and resolutions, thereby maintaining a consistent user experience across different devices [27]. It offers native API access to Windows functionalities like graphics, touch input, and device capabilities, which are utilized for managing user interactions such as clicking on dots and drawing lines [27]. The platform also supports XAML, a declarative language for defining UI elements, simplifying the design and modification of the game's visual interface [27]. Moreover, its inherent cross-device compatibility allows the game

to be played on diverse Windows devices without the necessity for separate codebases for each platform [27]. UWP also includes built-in support for animations and transitions, which are crucial for delivering smooth visual feedback when lines are drawn, or squares are completed [23].

### **5.1.4 XAML (Extensible Application Markup Language)**

XAML, an XML-based declarative language, is utilized in the Universal Windows Platform (UWP) to design and define the user interface (UI) of applications [28]. It facilitates the separation of UI design from business logic, thereby simplifying the management of intricate UI layouts and the customization of visual elements [28].

The relevance of XAML to a project is demonstrated through its capacity to enable developers to construct interactive and visually engaging interfaces without the need for complex coding for layout and design [28]. Specifically, it will be employed to delineate the layout of the "Dots and Boxes" grid, display player scores, and indicate AI status [28]. Furthermore, XAML's support for data binding permits real-time updates of the game's state, such as the count of completed squares, to be mirrored in the UI automatically [28]. The animations and visual transitions inherent in XAML will also serve to enrich the user experience by delivering fluid feedback during gameplay, including highlighting completed squares or animating the moves made by the AI [28].

## **5.2 Setting and Configuration**

This section details the user-configurable aspects of the "Dots and Boxes" application, illustrating how players can personalize their gaming environment and tailor the gameplay experience to their preferences. The options available are primarily managed through the dedicated Settings area and the Main Menu, and visual representations of these interfaces would be integral to this discussion.

The application offers several audio customization features, accessible through the Settings interface. A primary feature here is the ability to toggle sound effects. When enabled, users

receive auditory feedback during gameplay actions and operational interactions, which can enrich the interactive experience. Conversely, users who prefer a quieter environment can disable these sound effects, and the game will operate without these auditory cues. Complementing this, users also have control over the background music. They can choose to enable background music, which provides an engaging atmosphere to accompany gameplay. There are three distinct background music options for the user to select from, allowing for some variety. If a user opts to disable the background music, it will not play, ensuring the gaming experience aligns with their individual auditory preferences. In essence, these flexible audio options allow users to create either a fully immersive soundscape or a more subdued atmosphere, ensuring that every player can enjoy the game in a way that is most comfortable for them.

Further customization is available through the Main Menu, where users can adjust several interactive elements crucial for shaping the gameplay. The "Theme" options allow players to alter the game's visual appearance, specifically by choosing from three different checkerboard color schemes: a default option, a light colour scheme, and a blue color scheme. This allows users to select a visual style that they find most appealing. A significant gameplay configuration is the "Computer Difficulty" setting. This allows users, particularly when playing against an AI opponent, to select an appropriate challenge level. Options typically range from easy, suitable for novices or those seeking a relaxed game, to more formidable levels for experienced players desiring a significant challenge. This ensures a balanced and enjoyable experience for a wide range of skill levels. Additionally, users can customize the "Board Size." This selection directly influences the scale and complexity of the game. Players can choose from various board dimensions, with larger boards generally offering more strategic depth and longer gameplay, while smaller boards lead to quicker matches and more immediate tactical decisions.

In summary, the Setting and Configuration options for theme colour, computer difficulty, board size, sound effects, and background music are integral to providing a tailored and user-centric gameplay experience. By offering these choices, the application empowers players to engage with the "Dots and Boxes" game in a manner that best suits their individual style, preferences, and desired level of challenge, ultimately fostering a more satisfying and immersive interaction.

### **5.3 System Operation (with Screenshot)**

This section provides a comprehensive walkthrough of the Dots and Boxes application's operation to illustrate the user's journey from launching the game to engaging in gameplay against the AI. The operational flow is designed to be intuitive, supported by a clear user interface and a robust backend implementing the Minimax algorithm. Screenshots will be referenced throughout to visually guide the description.

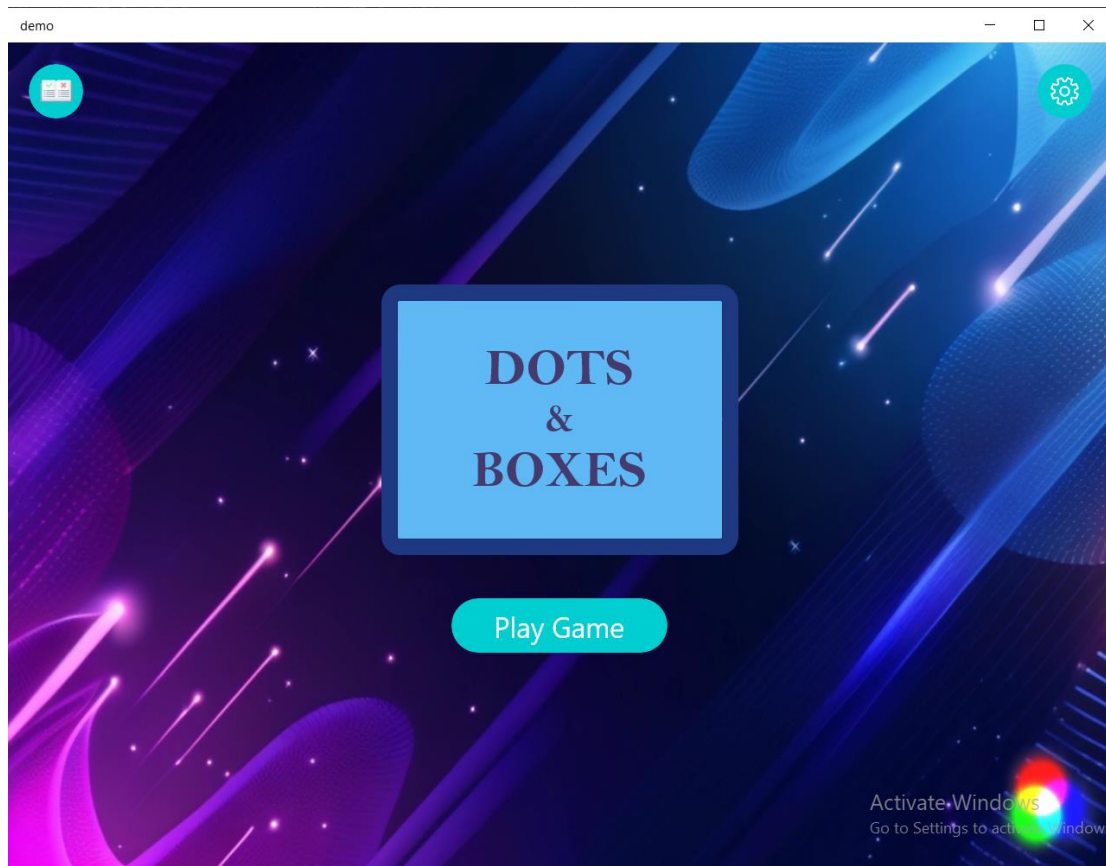


Figure 5.1 Start Page

Upon launching the application, the user is greeted by the Start Page, as depicted in Figure 5.1. This page serves as the initial entry point and presents a clean, welcoming interface. Prominently displayed is the game's title, "DOTS & BOXES," establishing the application's identity. Below the title, a "Play Game" button invites the user to proceed to the main game setup. Additionally, the icons typically representing "Rules" and "Settings" (as suggested by the book icon in the top left and a settings icon in the top right of the provided screenshot for Figure 5.1) offer direct access to these respective sections. This design ensures that users can immediately understand the primary action (to play the game) or choose to learn more about



the game or configure preferences before starting. The overall aesthetic is shown in the screenshot with its vibrant background, which aims to be engaging.

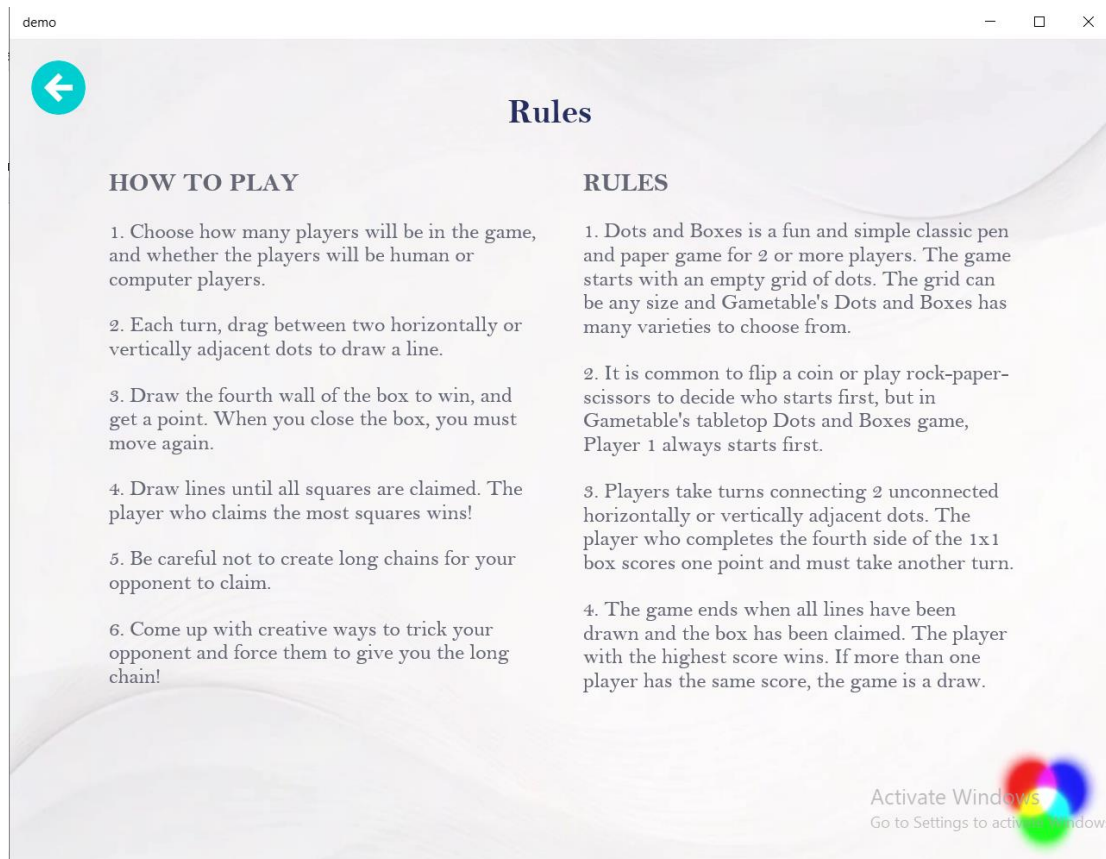


Figure 5.2 Rules Page

If the user navigates to the Rules section, likely by clicking a dedicated icon on the Start Page, they are presented with the Rules Page, illustrated in Figure 5.2. This page is designed to provide players with all the necessary information regarding how to play Dots and Boxes. The screenshot shows two main sections: "HOW TO PLAY," offering a concise list of steps and a more detailed "RULES" section, which elaborates on aspects like game setup, turn-taking, scoring (completing a box to earn a point and another turn), and the end-game condition (all lines drawn, player with the most squares wins). The layout is clear and legible, facilitating easy understanding. A back arrow icon visible in the top-left corner of the screenshot allows the user to conveniently return to the previous page, likely the Start Page, after familiarizing themselves with the game's mechanics.

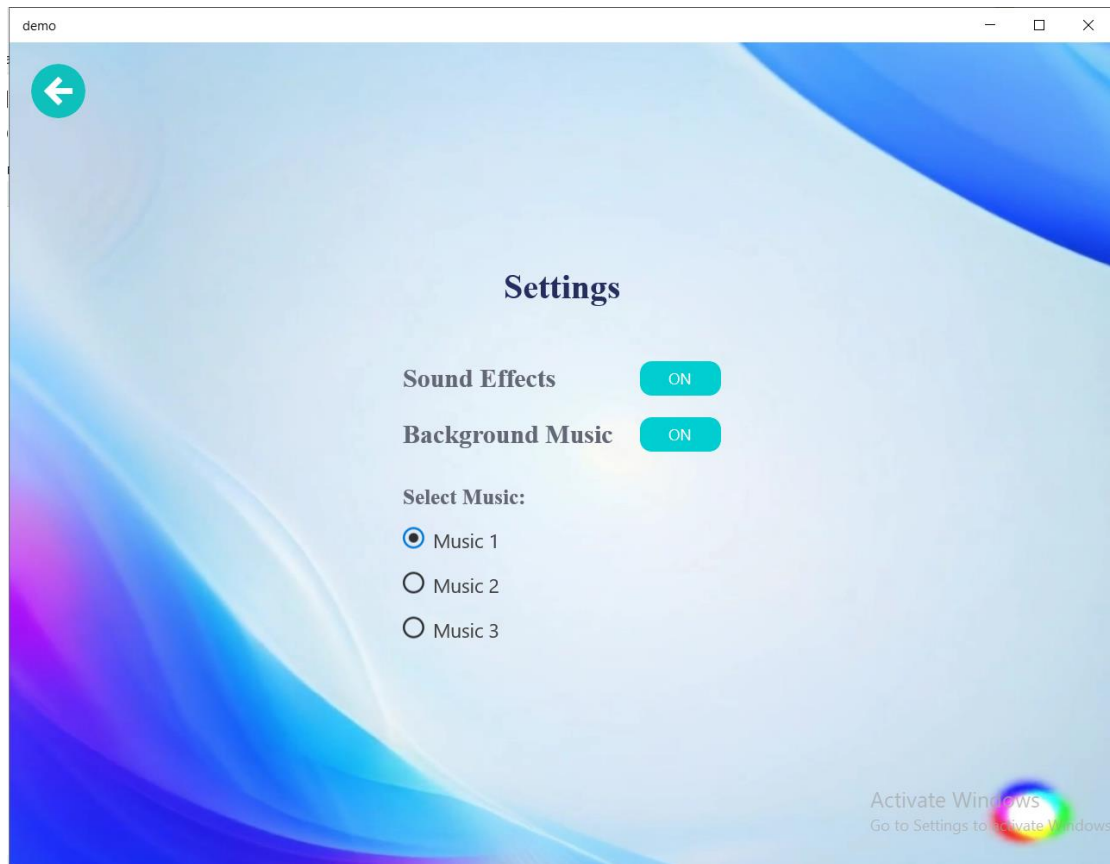


Figure 5.3 Settings Page

The Settings Page, shown in Figure 5.3, is accessible from the Start Page and potentially the Main Menu, allowing users to customize their audio experience. This page plays a crucial role in personalizing the application. As seen in the screenshot, users are presented with clear options: "Sound Effects" can be toggled "ON" or "OFF," enabling or disabling auditory feedback during gameplay and operations. Similarly, "Background Music" can be turned "ON" or "OFF." Below these toggles, if background music is enabled, users can "Select Music" from a list of options (Music 1, Music 2 and Music 3 are shown as radio button choices). This allows users to tailor the game's ambiance to their preference, whether they enjoy an immersive soundscape or a quieter session. A back arrow icon is again present for easy navigation.

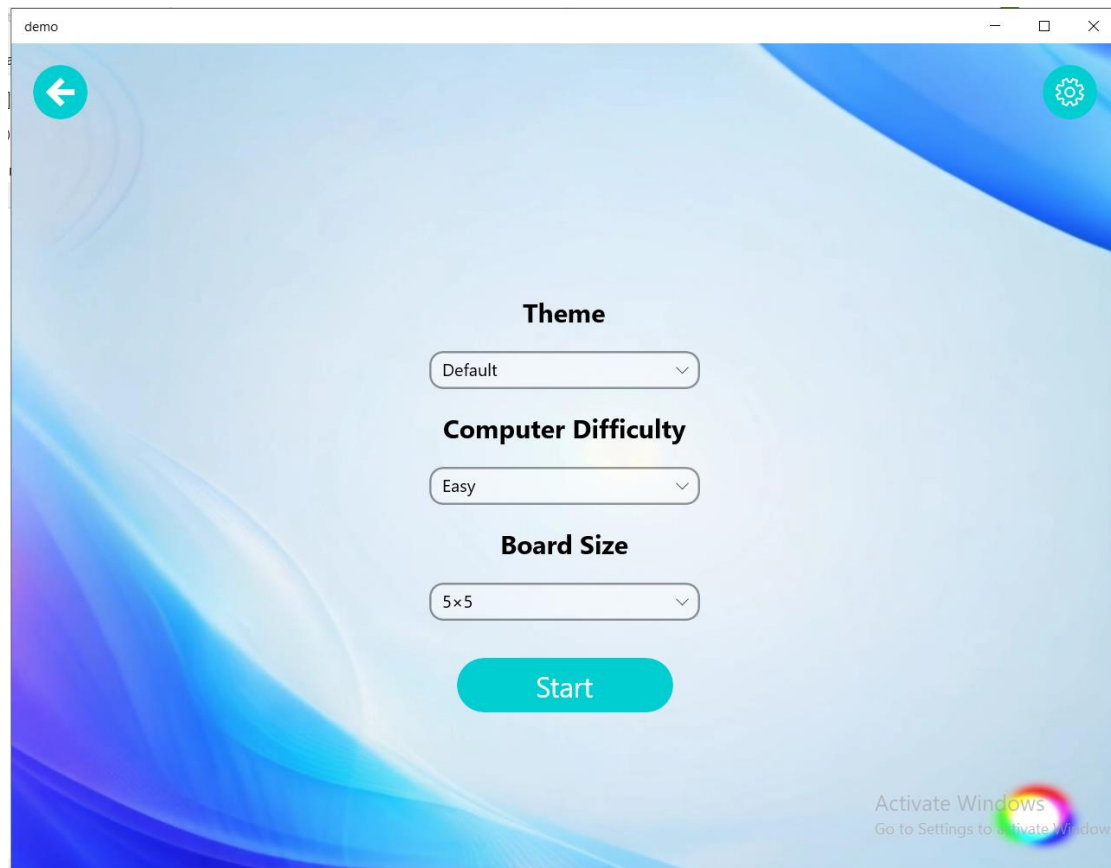


Figure 5.4 Main Menu Page

Navigating from the Start Page (e.g., by clicking "Play Game") or returning from other sections like Settings (if accessed from the Main Menu) leads the user to the Main Menu Page depicted in Figure 5.4. This page is central to configuring the upcoming game. The screenshot shows several interactive elements: a "Theme" dropdown allows users to select from different visual styles for the game board (e.g., "Default," with other options like Light and Blue as per your description). Crucially, users can set the "Computer Difficulty" via another dropdown (shown as "Easy," with options for Medium and Hard implied). This setting directly influences the AI's playing strength. A "Board Size" dropdown allows users to customize the dimensions of the game grid, affecting the game's length and complexity. Once these preferences are set, the user clicks the prominent "Start" button to commence gameplay. Icons for returning (back arrow) and accessing settings are also visible, providing flexible navigation.



Figure 5.5 Gameplay and AI Operation

Proceeding to the gameplay screenshots (Figure 5.5), it's important to outline the core operations that occur once the "Start" button on the Main Menu is pressed. The application transitions to the Game Page. Here, the initial steps involve setting up the game board according to the selected size and entering the game phase where players take turns. The system maintains a clear definition of the game state, which includes all drawn lines, the current player's turn, and completed squares – this is crucial for the Minimax algorithm.

The AI's decision-making, managed by `AIPlayer.cs`, then comes into play. This involves implementing the Minimax algorithm to evaluate potential moves by simulating outcomes. This core algorithm is enhanced with Alpha-Beta pruning and heuristics to improve efficiency. The process involves defining the minimax function to recursively evaluate game states, checking for terminal conditions (win/lose/draw). The AI evaluates actions to maximize its own score and minimize the opponent's score. Based on this evaluation, an action is generated, and the game state is updated. The `GamesPage.xaml.cs` file manages this user interface, game flow, player interaction (handling human clicks/drags to draw lines, providing visual

feedback), dynamically generating the UWP UI elements for the board (dots, lines, squares), and managing turns between the human and AI. It invokes the AI's `GetBestMove` method asynchronously to keep the UI responsive and updates the visual state based on the AI's or human's move, including scores and highlighting completed boxes. The game continuously checks for end conditions.

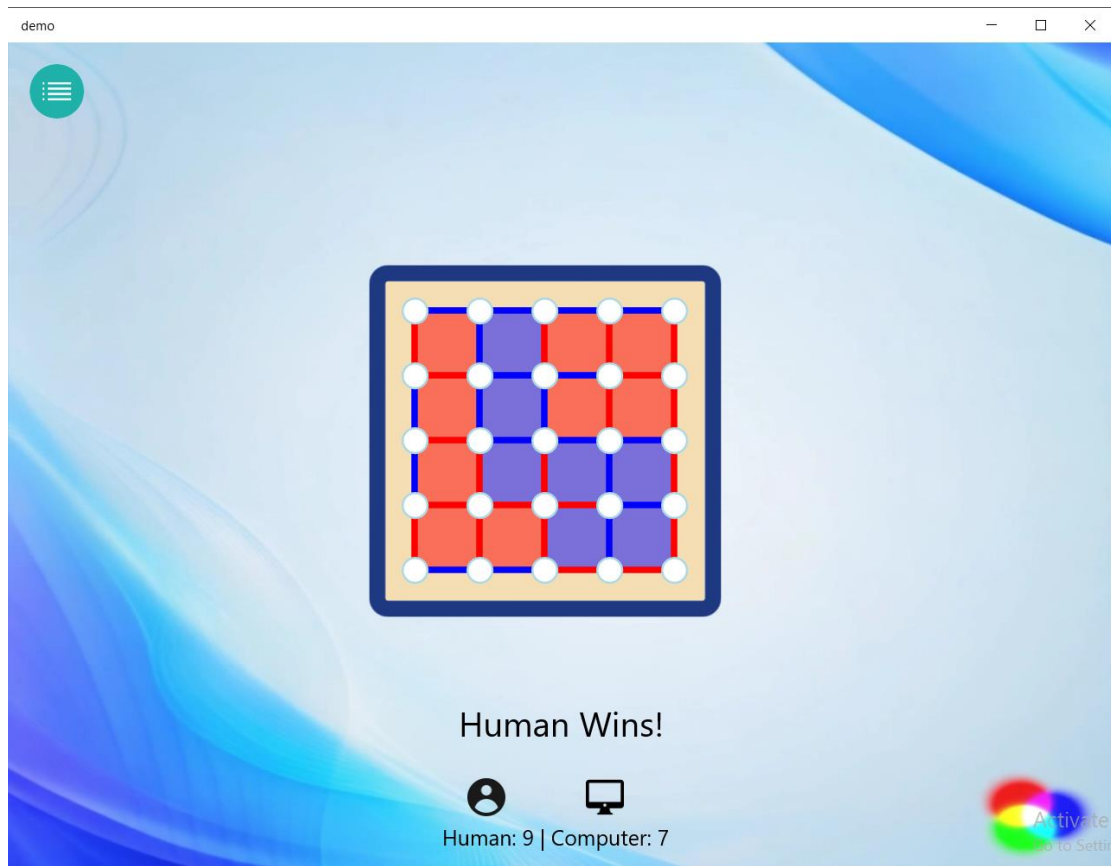


Figure 5.6 Easy Level Play

Figure 5.6 showcases an example of gameplay at the "Easy" difficulty level. The screenshot displays the game board, likely at or near the end of a game, with dots, lines drawn by both players (distinguished by colour, e.g., red and blue), and completed squares filled in. Player scores are visible at the bottom ("Human: 9 | Computer: 7" in the example screenshot), along with a game outcome message ("Human Wins!"). During gameplay on Easy, the AI uses a Fixed-Depth Minimax search, exploring only 2 plies ahead. While it employs Alpha-Beta Pruning, a Transposition Table, Move Ordering, Killer Moves, and Quiescence Search for efficiency and basic tactical stability, its limited foresight makes it a more predictable and less

challenging opponent, suitable for novice players. Iterative deepening is not used, and time management is not a primary constraint due to the shallow search.

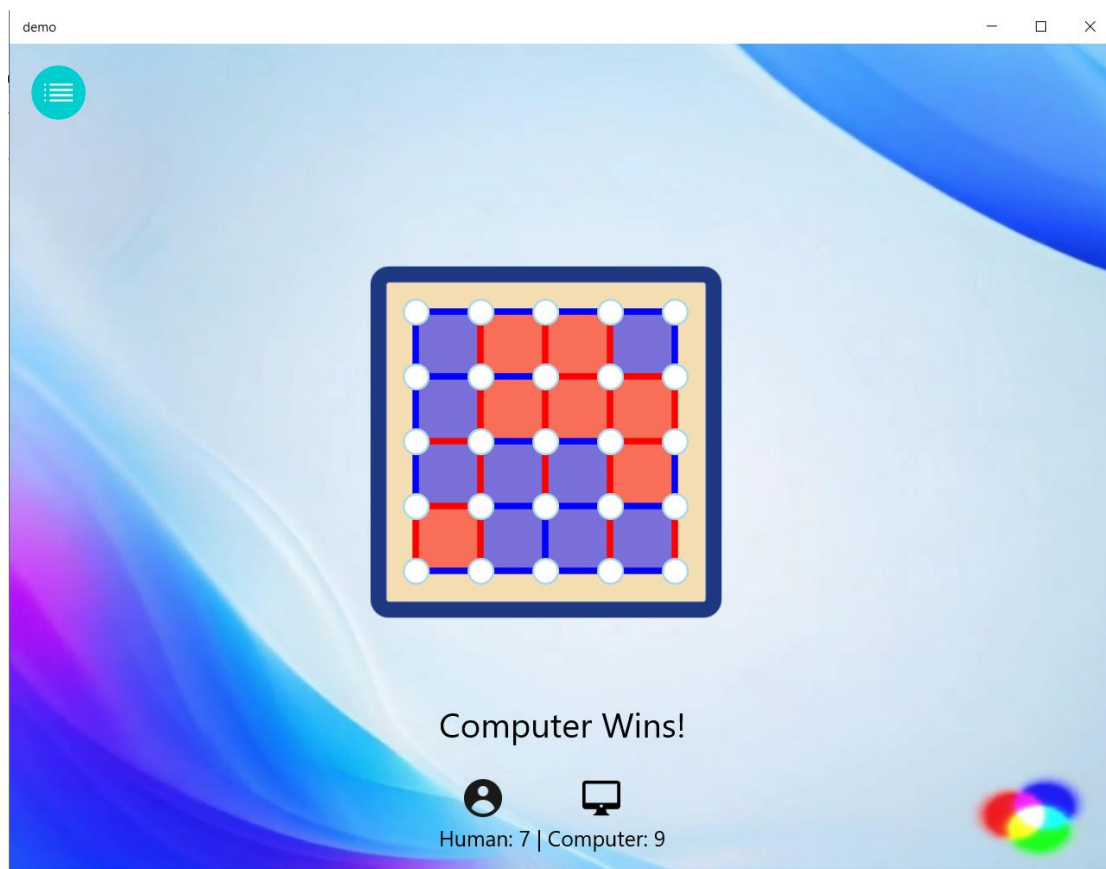


Figure 5.7 Medium Level Play

Figure 5.7 presents a scenario from a game played on "Medium" difficulty. Like the previous figure, it shows the game board, scores ("Human: 7 | Computer: 9" in this example), and a game outcome ("Computer Wins!"). The AI's operation at this level is more sophisticated. It uses a Fixed-Depth Minimax search extended to 4 plies. This deeper lookahead allows the AI to understand more complex tactical situations and make more deliberate short-term plans. Alpha-Beta Pruning becomes more critical here to manage the larger search space. The Transposition Table, Move Ordering, and Killer Moves continue to enhance efficiency and decision quality. The Quiescence Search remains crucial for stable evaluations at the search horizon. However, like the Easy level, Iterative Deepening is not employed, and the search depth is fixed. This AI provides a noticeably greater challenge than the Easy level.



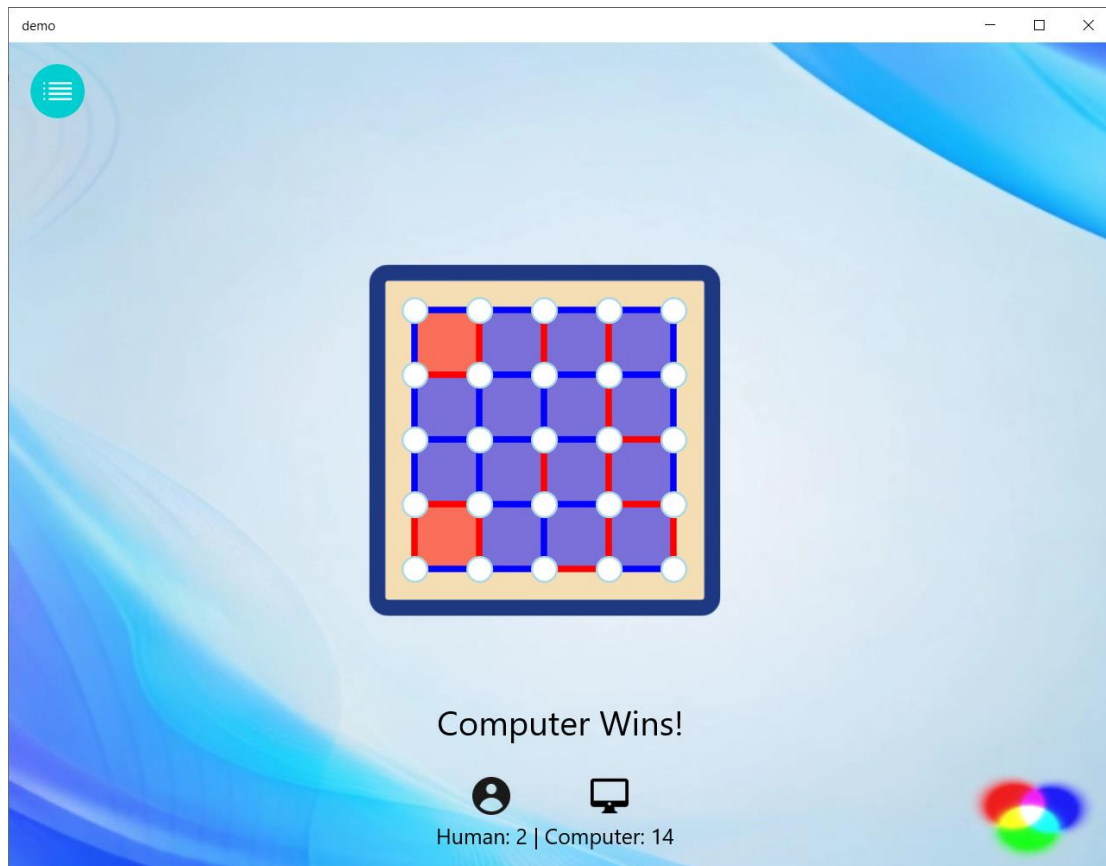


Figure 5.8 Hard Level Play

Figure 5.8 illustrates gameplay against the AI at its most challenging "Hard" difficulty setting. The screenshot again shows the game board, scores ("Human: 2 | Computer: 14"), and a game outcome ("Computer Wins!"), potentially with a more decisive victory for the AI. At this level, the AI employs its most advanced strategy: Iterative Deepening Minimax. It starts with a shallow search and progressively increases the depth (up to a maximum of 10 plies), using the best move found from the deepest completed iteration. This adaptive search is governed by Time Management, ensuring the AI makes a move within a reasonable timeframe. Alpha-Beta Pruning is indispensable, and the Transposition Table becomes extremely powerful, leveraging information from shallower searches. High-quality Move Ordering and Killer Moves are crucial for maximizing the search depth achievable within the time limit. A Quiescence Search is applied at the leaf nodes of each iteration. This dynamic and potentially much deeper analytical approach allows the AI to understand complex long-term strategies, making it a formidable opponent.

Throughout these gameplay stages (Figures 5.6-5.8), the `GamesPage.xaml.cs` ensures that player inputs are handled, the board is updated visually (lines drawn, squares coloured, scores incremented), sound effects are played (if enabled) and turns alternate correctly. The menu icon visible in the top-left corner of the game screen likely provides options to reset the game or return to the Main Menu.

In summary, the system's operation is characterized by a well-structured user interface that guides the user from initial setup to engaging gameplay. The core of the application lies in its AI, driven by the Minimax algorithm and its enhancements, providing varying levels of challenge corresponding to the selected difficulty. All is visually represented on the game board and through clear feedback to the user.

### **5.4 Implementation Issues and Challenges**

This part reflects on the practical difficulties and obstacles encountered during the development of the Dots and Boxes game, alongside the strategies and solutions employed to overcome them. The implementation of a game featuring a sophisticated AI, a dynamic user interface and various customization options invariably presents a unique set of challenges.

A primary area of technical challenge revolved around the correct and efficient implementation of the Minimax algorithm, particularly with its enhancements. Ensuring the logical correctness of the recursive Minimax function and the conditions for Alpha-Beta pruning was paramount, as subtle errors here could lead to significantly suboptimal AI play. Debugging these recursive algorithms required meticulous attention to detail, possibly involving logging AI decision paths or visualizing parts of the search tree. Developing an effective heuristic evaluation function for the AI also posed a considerable challenge. This involved carefully balancing various factors, such as the current score difference, penalizing moves that create immediate scoring opportunities for the opponent (like leaving three sides of a box open) and discouraging the creation of long, unfavorable chains of boxes with two sides. Fine-tuning the weights and logic of this heuristic function to produce intelligent and human-like behavior across different game situations was an iterative process. Furthermore, implementing advanced techniques like transposition tables (managing cache efficiently and handling hash collisions), killer move heuristics, and quiescence search (defining "volatile" positions accurately) added layers of



## CHAPTER 5

complexity that needed careful design and testing. For the "Hard" difficulty, integrating iterative deepening with effective time management that can ensure the AI made strong moves within an acceptable timeframe, which adjusted based on grid size, was a specific optimization challenge.

Managing the game state efficiently and accurately was another significant hurdle. The Dots and Boxes game requires frequent updates to the board, scores and turn information. For the AI to simulate potential moves, game states often need to be cloned. Doing this without incurring substantial performance overhead, especially with larger board sizes or deeper search depths, was a critical consideration. Correctly implementing the turn-taking logic, especially the rule that allows a player to take another turn after completing a square, required careful state management to prevent errors in game flow.

From a User Interface and User Experience perspective, developing the game within the UWP framework using C# and XAML presented its own set of challenges. Dynamically generating the game board on a XAML canvas, including the dots, lines (initially hidden and then made visible), and rectangles for completed squares and ensuring it scaled correctly for different board sizes demanded precise layout and rendering code. A major concern was maintaining a responsive UI, especially when the AI was performing its computations. This was addressed by invoking the AIPlayer's `GetBestMove` method asynchronously, as detailed in the `GamesPage.xaml.cs` description, preventing the UI from freezing. Handling human player input smoothly, such as clicking and dragging to draw lines between dots and providing immediate visual feedback like dot highlighting and preview lines, also required careful event handling and UI updates. Implementing the different visual themes and ensuring consistent application across game elements, as well as integrating and managing the state of sound effects and background music (including selection persistence), added to the UI development workload.

The debugging process for a game involving AI can be particularly intricate. Beyond typical code errors, identifying why an AI makes a suboptimal move or behaves unexpectedly requires insight into its decision-making process. This might have involved creating specialized debugging tools or extensive logging to trace the AI's evaluation of different game branches. Testing the AI thoroughly across a multitude of game scenarios and for each difficulty level

was essential to validate its effectiveness and to ensure that the difficulty scaling felt appropriate and fair to the player.

Finally, achieving the right balance for the AI difficulty levels ("Easy," "Medium," and "Hard") was a significant design and implementation challenge. This involved not just setting different search depths (2 plies for Easy, 4 for Medium, iterative deepening up to 10 for Hard) but also potentially adjusting heuristic evaluation parameters or the application of certain advanced techniques (like the stringency of time limits for Hard) to ensure a distinct and progressively challenging experience for the player. This often requires extensive play-testing and iterative refinement.

Overcoming these challenges involved a combination of careful algorithm design, robust coding practices, thorough testing and an iterative approach to development, particularly in refining the AI's behaviour and the overall user experience.

### **5.5 Concluding Remark**

Process of transforming the conceptual design of the "Application of Minimax Algorithm in Dots and Boxes Game" into a functional and interactive software system. It has covered the meticulous software setup, the array of user-configurable settings and options, a walkthrough of the system's operational flow from the user's perspective and a candid discussion of the implementation issues and challenges encountered along with their resolutions. The journey from design to a tangible product has been both challenging and rewarding, culminating in a system that effectively brings the classic game of Dots and Boxes to life with an intelligent AI opponent.

The key achievements of this implementation phase are noteworthy. A fully playable Dots and Boxes game has been successfully developed, featuring a robust AI opponent powered by the Minimax algorithm. This AI is not monolithic, it incorporates varying difficulty levels (Easy, Medium, and Hard), achieved through sophisticated techniques such as adjustable search depths, Alpha-Beta pruning, transposition tables for caching results, killer move heuristics for prioritizing promising moves, quiescence search for more accurate evaluation in volatile positions, and even iterative deepening with time management for the highest difficulty.

Furthermore, the application boasts a user-friendly interface, designed with intuitive navigation

between the Start Page, Main Menu, Rules, Settings and the Game Page itself. Users are provided with significant control to personalize their experience through customizable game board themes, board sizes and comprehensive audio settings, including sound effects and a selection of background music tracks. The successful integration of these features underscores the practical realization of the project's design specifications.

Reflecting on the initial objectives, the implemented system effectively demonstrates the application and potency of the Minimax algorithm within the context of a strategic board game like Dots and Boxes. The varying AI behaviours across different difficulty settings clearly showcase the algorithm's capability to make intelligent decisions and provide a challenging experience. The development of the AI, from its core logic in `AIPlayer.cs` to its interaction with the game environment managed by `GamesPage.xaml.cs` serves as a practical testament to the algorithm's utility in game development.

The overall implementation process can be regarded as a success, having navigated numerous technical and design challenges to deliver a complete and engaging application. The development journey has provided invaluable insights into game AI development, user interface design for interactive applications within the UWP framework and the practical aspects of integrating complex algorithms with a user-facing front end. Key lessons learned include the critical importance of modular design for managing complexity, the necessity of asynchronous programming for maintaining UI responsiveness during intensive computations like AI decision-making and the iterative nature of refining AI heuristics and balancing difficulty levels through rigorous testing and observation. These takeaways will undoubtedly prove beneficial for future software development endeavours, particularly those involving artificial intelligence or interactive game design. This concluding remark signifies the completion of the system implementation phase, with a functional Dots and Boxes game ready for evaluation and user engagement.

## Chapter 6: System Evaluation And Discussion

### 6.1 System Testing and Performance Metrics

In this initial section of the evaluation, the comprehensive strategy employed for testing the "Application of Minimax Algorithm in Dots and Boxes Game" is clearly defined alongside the specific metrics used to rigorously measure its operational performance, the efficacy of its artificial intelligence and the overall quality of the user experience. The primary goal of this testing phase is to validate the application's functionality against its design specifications, assess the intelligence and challenge posed by its AI across different difficulty levels, determine its usability from a player's perspective and quantify its technical performance characteristics.

A multifaceted testing approach was adopted, beginning with Functional Testing. This was meticulously carried out to ensure all aspects of the game operate as intended. This included verifying the seamless UI navigation between the Start Page, Main Menu, Rules, Settings and the Game Page, including all back-and-forth transitions. The correct application of Dots and Boxes game rules—such as valid line placement, accurate box completion and scoring (one point per box and an extra turn upon scoring) and adherence to turn-based play—was thoroughly checked. All customization options available in the settings, including the application of visual themes, the toggling and selection of sound effects and background music, the effective switching of AI computer difficulty levels and the correct rendering of different board sizes were validated. It was also confirmed that the AI makes strategically valid moves in all game situations and that game termination conditions (win, lose or draw) are detected and displayed accurately. The functionality of in-game options, such as resetting the current game or returning to the main menu, was also affirmed.

AI Performance Testing formed a critical component of the evaluation, given the project's central focus on the Minimax algorithm. This testing was designed to objectively assess the AI's playing strength, the quality of its decision-making and the distinctness of the challenge presented by its "Easy", "Medium" and "Hard" difficulty settings. Evaluation methods included conducting a series of games against human testers with varying levels of experience in Dots and Boxes. The impact of the different AI techniques implemented such as varied search depths for fixed-depth Minimax, the efficiency of Alpha-Beta pruning, the utility of transposition

## CHAPTER 6

tables and killer move heuristics, the strategic considerations of the quiescence search and the adaptive nature of iterative deepening with time management on the "Hard" level such as was observed through gameplay outcomes and AI response patterns.

Usability Testing was conducted to evaluate the ease of use and overall user-friendliness of the application. This involved assessing the intuitiveness of the UI navigation and the logical flow of interaction from launching the game to completing a match. The clarity of the game rules as presented on the Rules Page and the ease with which users could understand and operate the various options in the Settings and Main Menu were key focus areas. The overall user satisfaction and engagement with the game were also considered, often gathered through direct observation and feedback from testers.

Finally, technical performance testing addressed non-functional aspects of the application. A primary focus was measuring the AI's move calculation time, especially for the more computationally intensive "Medium" and "Hard" difficulty levels and observing how these times scaled with different board sizes. UI responsiveness was monitored to ensure smooth visual updates on the game board and to confirm that the application remained interactive and did not freeze during AI turns, thereby validating the effectiveness of asynchronous AI processing. Application load times for critical screens and transitions were also noted to ensure a swift user experience. While not a primary focus for this type of game, general resource usage (CPU and memory) might be observed during demanding AI calculations to ensure no excessive consumption patterns.

To quantitatively and qualitatively assess these areas, a precise set of Performance Metrics was defined. For evaluating AI Strength and Performance, the primary metrics included win/loss/tie ratios derived from games played against human testers across all three difficulty levels. Average score differentials in these games were also considered to gauge the decisiveness of wins or losses. Qualitative feedback from testers regarding the perceived intelligence of the AI's moves, the challenge posed by each difficulty level and any observable strategic patterns formed an important part of this assessment. For Technical Performance, metrics included the average and maximum AI move calculation time (measured in milliseconds or seconds) for each difficulty level and selected board sizes. UI responsiveness was gauged by ensuring no noticeable lag or stuttering during gameplay and AI thinking periods. Application load times for key screens (e.g., from Main Menu to Game Page) were also measured. For Usability,

metrics encompass task completion rates for essential user actions (e.g., successfully starting a game with specific custom settings, changing audio preferences), user satisfaction ratings (potentially gathered via a simple post-test questionnaire) and the frequency and nature of any errors or points of confusion encountered by testers. The justification for these testing methodologies and metrics lies in their collective ability to provide a holistic and robust evaluation of the Dots and Boxes game, thoroughly covering its core functionality, the intelligence of its central AI component, its overall user-friendliness and its technical stability and efficiency.

### **6.2 Testing Setup and Result**

The Testing Setup was designed to ensure a thorough evaluation of the application. Tests were conducted on standard Windows 10 personal computers, equipped with processors and RAM typical for running UWP applications, and utilizing the Visual Studio 2022 environment for debugging and performance monitoring where applicable. The testers were run by developers and covered a wide range of experience with Dots & Boxes games, from novice players unfamiliar with the game's deep strategy to veteran players who could provide a more challenging benchmark for the AI. For Functional Testing, a comprehensive checklist approach was adopted. This covered all UI navigation paths (Start Page, Rules Page, Settings Page, Main Menu Page, and transitions to/from the Game Page), the correct implementation of all game rules (line drawing, box completion logic, scoring one point per box, awarding an extra turn upon scoring), the successful application of all user-configurable settings (theme color changes, sound effect and background music toggling/selection, AI difficulty adjustments and board size variations) and the accurate detection and display of game termination conditions (win, loss, or draw). For AI Performance Testing, a key method involved human testers playing a series of 10 games against the AI at each of its three difficulty levels (Easy, Medium, Hard) on a standard board size (e.g., 5x5). Additionally, specific tests were conducted on a larger 11x11 board to evaluate AI response times under more computationally intensive conditions. Usability testing is conducted by testers while interacting with the application, subsequently evaluating the experience in terms of UI intuitiveness, clarity of game rules, and ease of configuration settings. Manual testing was the primary method for functional and usability

## CHAPTER 6

evaluations, while Visual Studio's diagnostic tools and manual timing were utilized for assessing AI response times.

The results obtained from this multi-faceted testing approach are detailed as follows. Functional testing verified that all core functionalities of the Dots and Boxes application operate correctly and align with the design specifications. UI navigation between all screens was found to be seamless and intuitive. The implementation of game rules, including accurate line placement, correct box capture and scoring, and the proper awarding of extra turns, was confirmed. Furthermore, all settings related to themes, audio controls, AI difficulty selection, and board size customization were fully functional and correctly applied during gameplay. The game also reliably detected and announced win, loss, or draw conditions.

Regarding AI performance in terms of playing strength, the AI was tested against human players over 10 games on a standard board size for each difficulty level. On the Easy Level, human testers achieved 7 wins, experienced 2 losses, and 1 game resulted in a tie. This 70%-win rate for humans indicates that the Easy AI offers a suitable challenge for beginners, enabling them to learn game mechanics without being overwhelmed and providing ample opportunities for success. When facing the Medium AI, human testers secured 4 wins, while the AI won 5 games, with 1 game ending in a tie. This nearly 50-50 split, where humans won 40% of games and the AI won 50%, suggests a well-balanced difficulty level that provides an engaging and competitive experience for players with some familiarity with Dots and Boxes. The Hard AI proved to be a formidable opponent, as human testers won only 2 games while the AI achieved 8 wins. This 20%-win rate for humans demonstrates the significantly increased playing strength of the Hard AI, validating the effectiveness of its advanced search techniques, such as iterative deepening Minimax and its heuristics. Collectively, these results demonstrate a clear and effective progression in AI playing strength across the three difficulty levels, successfully aligning with the project's objective of providing varied and appropriate challenges.

In terms of AI performance related to response time, specific measurements were taken on a larger 11x11 board to assess how the system performed under a greater computational load. For the Easy Level on this 11x11 board, the AI's response was consistently "fast", which implies near-instantaneous moves, ensuring a fluid and uninterrupted experience even on a larger grid for this introductory difficulty. When the Medium Level AI was tested on the 11x11

## CHAPTER 6

board, it took "around 2.5 seconds" on average to make a move. This duration is considered acceptable, maintaining player engagement despite the increased complexity of the board and the moderately challenging AI. For the Hard Level on the 11x11 board, the AI's move calculation time averaged "around 4.5 seconds." This response time is reasonable given the significant increase in the search space on such a large board and the depth of analysis performed by the iterative deepening Minimax algorithm. It also indicates that the time management heuristics implemented for the Hard AI are effective in keeping the game playable within satisfactory time limits. Overall, the observed response times scale appropriately with the AI's increasing complexity and the board size, ensuring that the game remains interactive and enjoyable across all configurations.

The usability testing results and feedback from usability testing were predominantly positive. Testers found the application's navigation to be intuitive and straightforward. The game rules presented on the Rules Page were deemed clear and easy to understand. Appreciated the range of customization options in the Settings and Main Menu, particularly the ability to change themes, control audio, and select difficulty and board size, finding these options easy to access and configure. Visual feedback during gameplay, such as the drawing of lines, highlighting of completed squares, and score updates, was well-received and contributed to a positive user experience. No significant usability issues were reported, suggesting the design effectively meets user needs for ease of use and engagement.

In analysing these results, it is evident that the Dots and Boxes application has successfully met its core design and functional requirements. The AI demonstrates a clear differentiation in playing strength across its difficulty settings, providing both accessible gameplay for novices and a significant challenge for more experienced players. The system performs efficiently, with AI response times remaining within acceptable limits for an engaging user experience, even on larger board configurations.

### 6.3 Project Challenges

One of the primary strategic challenges was defining the appropriate scope and sophistication for the Computer Player. While the goal was to implement a robust Minimax algorithm, determining the optimal set of advanced AI techniques such as transposition tables, killer move



heuristics, quiescence search, and iterative deepening for the "Hard" level required a balance between the ambition for a highly intelligent opponent and the practical constraints of development time and resources available for a project of this nature. Another significant strategic consideration was ensuring a distinct and progressive difficulty curve across the "Easy", "Medium" and "Hard" levels. This was not merely a case of implementing different search depths but involved considerable iterative tuning of heuristic parameters and AI search strategies to provide a gameplay experience that was appropriately challenging and engaging for different user skill levels. The provided win/loss data (Human: 7 Win - 2 loss -1Tie on Easy, 4 Win – 5 Loss – 1 Tie on Medium, 2 Win – 8 Loss on Hard) suggests this was largely successful, but achieving this precise balance was a persistent focus.

Resource limitations, primarily in terms of time, were an overarching project challenge. The available timeframe inevitably influenced the extent to which certain features could be explored or refined. For instance, while a robust heuristic function was developed (considering score difference, penalizing opponent's immediate scoring opportunities, and discouraging two-sided chain creation), further experimentation with more complex heuristic components or machine learning-assisted tuning might have been pursued with more time.

The learning curve associated with mastering both the theoretical and practical aspects of advanced game AI development and the intricacies of the UWP platform for creating a polished application also represented a significant project challenge. Moving beyond a basic Minimax implementation to effectively integrate and debug techniques like iterative deepening, sophisticated move ordering and quiescence search required a deep understanding of game tree search algorithms. Similarly, leveraging the full capabilities of C# and XAML for dynamic UI generation, asynchronous operations for responsive AI and cross-device compatibility within UWP demanded a continuous learning effort throughout the project.

Finally, the iterative refinement of the AI's core logic, particularly its heuristic evaluation function and the specific parameters governing search depth or time limits for different difficulty levels, was a substantial undertaking. Knowing when a heuristic was "good enough" or when the balance between AI strength and response time (e.g., Easy level being "fast", Medium "around 2.5 seconds" and Hard "around 4.5 seconds" on an 11x11 board) was optimal often involved subjective judgment informed by repeated testing and observation. This iterative process, while crucial for quality, also had to be managed within the project's scope constraints.

These challenges were managed through a combination of careful initial planning, adopting an agile and iterative development methodology, prioritizing core project objectives and diligently applying learned knowledge. While most challenges were successfully addressed to deliver a functional and effective application, the process highlighted the complexities inherent in AI-driven game development and evaluation.

### **6.4 Objectives Evaluation**

The first project objective was "To develop and implement the core Minimax algorithm as the decision-making engine for an AI opponent within a Dots and Boxes game." This objective has been fully met. The core logic of the artificial intelligence is based on the Minimax search algorithm. The AI opponent demonstrably makes decisions and actively participates in the game, indicating that the Minimax engine is not only implemented but also functional. The ability of the AI to play complete games and achieve wins, particularly evident in the Medium and Hard difficulty levels (where the AI won 50% and 80% of games against human testers, respectively) provides strong evidence that Minimax serves as an effective decision-making engine. Furthermore, functional testing confirmed that the AI consistently makes valid moves within the rules of Dots and Boxes, reinforcing the successful implementation of this core algorithmic objective.

The second objective was "To integrate performance optimization techniques and variable difficulty levels into the Minimax AI." This objective has also been comprehensively achieved. Regarding performance optimization, Alpha-Beta pruning, a significant enhancement to the basic Minimax algorithm was explicitly implemented. Beyond this, several other advanced techniques were successfully integrated to improve both the efficiency and the strategic capabilities of the AI. These include the use of transposition tables to cache results of previously evaluated board states, killer move heuristics to prioritize strategically relevant moves, and quiescence search to ensure more stable evaluations in volatile capture sequences. The AI's response times, even on a large 11x11 board (Easy: "fast", Medium: "around 2.5 seconds", Hard: "around 4.5 seconds"), demonstrate that these optimization techniques allow the AI to perform complex searches within acceptable timeframes, making the game interactive and enjoyable. In terms of variable difficulty levels, the application successfully implements "Easy", "Medium" and "Hard" settings, which are user-selectable via the Main Menu. These

levels are not superficial. They employ distinct AI strategies, including different fixed search depths (2-ply for Easy, 4-ply for Medium) and the use of Iterative Deepening Minimax (up to 10 plies with time management) for the Hard level. The AI performance results from testing validate the effectiveness of this implementation, with win/loss ratios showing a distinct and progressive increase in AI playing strength from Easy (human win rate 70%) to Medium (AI win rate 50%) and further to Hard (AI win rate 80%). This confirms that variable and meaningful difficulty levels, supported by optimized AI, were successfully integrated.

The third objective was "To create a complete, functional and user-friendly prototype of the Dots and Boxes application incorporating the Minimax AI". This objective has been fully realized. The developed application constitutes a complete prototype, encompassing all essential features for a Dots and Boxes game. This includes dynamic board setup based on user-selected sizes, intuitive mechanics for players to draw lines and complete boxes, accurate scorekeeping, correct turn management (including awarding extra turns upon scoring), and reliable detection and announcement of win, loss, or draw outcomes. The functional testing results, as detailed in section 6.2, confirmed that all these core game functionalities, along with UI navigation and the application of settings, operate correctly as designed. The Minimax AI is seamlessly integrated as an opponent. In terms of user-friendliness, the application's interface with its distinct Start Page, Main Menu, Rules section, and Settings area was designed to facilitate intuitive navigation. Feedback from usability testing indicated that users found the UI easy to navigate, the game rules clear, and the available settings (for themes, audio, difficulty and board size) straightforward to understand and use. Positive reception of visual feedback during gameplay further attests to the application's user-friendly design. The provision of these customization options enhances the user experience, allowing players to tailor the game to their preferences.

In conclusion, based on the comprehensive implementation of features and the supporting evidence from system testing. All stated project objectives for the "Application of Minimax Algorithm in Dots and Boxes Game" have been successfully met. The project has delivered a functional game with an intelligent AI opponent that employs the Minimax algorithm, incorporates performance optimizations and variable difficulty levels, and is wrapped in a user-friendly and customizable interface.

### 6.5 Concluding Remark

In concluding this chapter on System Evaluation and Discussion, it is evident that the "Application of Minimax Algorithm in Dots and Boxes Game" has undergone a thorough assessment process, yielding significant insights into its functionality, performance and overall quality. The key findings derived from comprehensive system testing, encompassing functional verification, AI performance analysis across its varied difficulty levels, usability assessments, and technical performance measurements, along with the critical evaluation of project objectives and reflection on inherent project challenges, provide a solid basis for an overall appraisal of the developed application. The system has been proven to be functionally complete with all core game mechanics and user interface elements operating as designed. The AI, powered by the Minimax algorithm and its associated optimizations, demonstrably offers a progressively challenging experience, as evidenced by the win/loss ratios against human testers (Human 70% win rate on Easy, AI 50% win rate on Medium, AI 80% win rate on Hard) and maintains acceptable response times even on larger boards (Easy: fast, Medium: ~2.5s, Hard: ~4.5s on an 11x11 board). Furthermore, usability feedback has affirmed the intuitive nature of the application's navigation and the clarity of its options.

Considering these comprehensive evaluation outcomes, the project can be confidently assessed as a success. It has effectively achieved its central aim of developing and implementing a Dots and Boxes game that not only incorporates a functional Minimax-based AI but also demonstrates the algorithm's capabilities through varied difficulty levels and optimization techniques. The main strengths of the developed application lie in its robust AI implementation, which leverages techniques like Alpha-Beta pruning, transposition tables, killer move heuristics, quiescence search and iterative deepening to provide a challenging opponent, particularly at the "Hard" difficulty. The user-friendly interface, characterized by intuitive navigation across the Start Page, Main Menu, Rules, Settings and Game Page, coupled with a good degree of customizability through options for themes, audio controls, AI difficulty and board sizes, significantly enhances the player experience. The functional completeness ensures that users have access to all essential features of the Dots and Boxes game.

While the evaluation has been largely positive, and no major weaknesses were identified that impede the core functionality or user experience, it is acknowledged that the pursuit of artificial intelligence offers limitless scope for advancement. For instance, while the current AI is

## CHAPTER 6

effective, future iterations could explore even more sophisticated heuristic evaluation functions or adaptive learning capabilities for a more nuanced and human-like opponent. Performance on exceptionally large board sizes or with search depths significantly exceeding the current "Hard" level's 10 plies might also present increasing computational demands, though current testing has shown robust performance within the designed parameters.

This chapter's evaluation, therefore, encapsulates that the developed Dots and Boxes application stands as a well-implemented system that successfully meets its stated objectives. The findings from this evaluation might naturally lead to considerations for future enhancements, such as the exploration of more advanced AI paradigms, the potential addition of features like online multiplayer capabilities, an expanded range of customization options, or even a dedicated tutorial mode to help new players grasp the strategic depths of Dots and Boxes. Such future work could build upon the solid foundation established by this project.

## Chapter 7: Conclusion and Recommendation

### 7.1 Conclusion

This project successfully achieved its central aim, the development and implementation of a functional Dots and Boxes game featuring an intelligent Artificial Intelligence (AI) opponent driven by the Minimax algorithm. The journey involved meticulous design and development, culminating in a comprehensive application that not only provides engaging gameplay but also serves as a practical demonstration of AI principles in a classic strategy game context.

Key achievements of this project are multifaceted. A fully playable Dots and Boxes game was developed, complete with intuitive user navigation across various sections, including the Start Page, Main Menu, Rules, Settings, and the Game Page. The core of the AI opponent lies in the robust implementation of the Minimax algorithm. This was significantly enhanced with performance optimization techniques such as Alpha-Beta pruning, transposition tables to cache previously evaluated game states, killer move heuristics to prioritize strategically advantageous moves, and quiescence search to ensure more stable evaluations in dynamic game situations.

Furthermore, the project successfully delivered variable AI difficulty levels—Easy, Medium and Hard. These levels are not merely superficial adjustments but are characterized by distinct AI behaviors and search strategies. The Easy level employs a fixed-depth Minimax search of 2 plies, offering an accessible challenge. The Medium level increases this to a 4-ply search, providing a more thoughtful opponent. The Hard level utilizes a sophisticated Iterative Deepening Minimax approach, searching up to 10 plies and incorporating time management, presenting a formidable challenge to experienced players. This progressive difficulty ensures an engaging experience for a wide range of user skill levels.

The application also boasts a user-friendly interface and a suite of customization options that enhance the player experience. Users can personalize their game by selecting different visual themes (default, light and blue), choosing from various board sizes to alter game complexity and duration, and managing audio settings, including toggling sound effects and selecting from three background music tracks. The seamless integration of these features, managed by components like `GamesPage.xaml.cs` for the UI and game flow, and `AIPlayer.cs` for the AI logic, underscores the successful realization of the project's design specifications.

Drawing upon the system evaluations, it is evident that the initial objectives of the project have been comprehensively met. The Minimax algorithm serves as an effective decision-making engine, the optimization techniques ensure efficient AI operation and the variable difficulty levels provide a graduated challenge. The creation of a complete, functional and user-friendly prototype incorporating this AI has been fully realized. The development journey has also provided valuable insights into the practical application of AI algorithms in game development, the intricacies of user interface design for interactive experiences, and the balance between AI sophistication and computational performance. This project stands as a testament to how classic games can be revitalized and enhanced through the thoughtful application of artificial intelligence.

### **7.2 Recommendation**

The following recommendations aim to expand the game's capabilities, further refine the artificial intelligence and enrich the overall user experience.

Further advancements in Artificial Intelligence are a key area for future development. While the Minimax algorithm with its current heuristics and optimizations provides a strong AI, future work could involve integrating machine learning. Reinforcement learning, for instance, could enable the AI to learn and evolve its strategies through self-play, potentially discovering novel tactics and adapting more dynamically to different opponent styles beyond predefined heuristics. The current heuristic function, which considers critical factors like score difference and chain prevention, could also be made more sophisticated. Future iterations could explore more nuanced heuristic evaluations, possibly incorporating dynamic weighting of game elements based on the game phase (early, mid, endgame) or more complex pattern recognition to better assess board positions. To create an even more human-like and challenging AI, opponent modelling could be introduced, involving the AI attempting to identify patterns or weaknesses in the human player's strategy during a game and adapting its approach accordingly.

Expanding features and enriching gameplay also offer significant potential. A major enhancement would be the introduction of online multiplayer capabilities, allowing users to compete against friends or other players remotely, thereby greatly increasing the game's replay

ability and community engagement. Ensuring a polished local two-player mode, if not already a core feature, where two humans can play against each other on the same device, would also be beneficial. Developing an interactive tutorial mode could significantly benefit new players by explaining basic rules, demonstrating simple tactics, and perhaps even offering insights into the AI's decision-making process at lower difficulty levels. An advanced strategy guide section could also be added to help players understand deeper concepts like chain control and sacrifices. Furthermore, introducing diverse game modes, such as "Challenge Puzzles" with pre-set board configurations, "Timed Games", or variations on scoring rules, could add substantial variety.

Enhancing the user experience and customization options is another avenue for improvement. While the current theme and audio options are good, expanding these further could include a wider array of visual themes, board styles, or even allowing users to import their background music. Beyond the existing Easy, Medium and Hard levels, allowing users to create custom AI profiles by fine-tuning parameters like search depth or specific heuristic weights could appeal to advanced players seeking a very specific challenge. Implementing game analysis tools, such as a move history viewer, the ability to undo moves in practice modes, or an option to request hints from the AI, could improve the learning experience and allow players to analyse their games more effectively.

Performance and technical refinements should also be considered for future iterations. For players who enjoy very large board sizes, further optimization of the AI's search algorithms could be explored, possibly by investigating techniques for more aggressive pruning or even parallelizing parts of the search process on multi-core processors. While the game is developed as a UWP application, exploring possibilities for porting it to other platforms, such as the web or mobile devices, could significantly broaden its reach and accessibility.

By considering these recommendations, the "Application of Minimax Algorithm in Dots and Boxes Game" can continue to evolve, offering an even more intelligent, engaging, and feature-rich experience for its users, while also serving as a valuable platform for further exploration in the domain of game AI.



## REFERENCES

- [1] Wikipedia contributors, "Dots and boxes," Wikipedia, May 12, 2024. [https://en.wikipedia.org/wiki/Dots\\_and\\_boxes#:~:text=Dots%20and%20boxes%20is%20a,an d%20pigs%20in%20a%20pen](https://en.wikipedia.org/wiki/Dots_and_boxes#:~:text=Dots%20and%20boxes%20is%20a,an d%20pigs%20in%20a%20pen). (Accessed: 04 May 2025)
- [2] GeeksforGeeks, "Minimax Algorithm in Game Theory | Set 1 (Introduction)," GeeksforGeeks, Jun. 13, 2022. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/> (Accessed: 04 May 2025)
- [3] GeeksforGeeks, "Minimax Algorithm in Game Theory | Set 4 (AlphaBeta Pruning)," GeeksforGeeks, Jan. 16, 2023. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/> (Accessed: 04 May 2025)
- [4] S. Hossain, "Key Problem-Solving Algorithms and Strategies in AI," Medium, Nov. 23, 2024. [Online]. Available: <https://machinelearning4all.com/key-problem-solving-algorithms-and-strategies-in-ai-d64e665fc1de> (Accessed: 04 May 2025)
- [5] Wikipedia contributors, "C Sharp (programming language)," Wikipedia, May 04, 2025. [https://en.wikipedia.org/wiki/C\\_Sharp\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) (Accessed: 04 May 2025)
- [6] "W3Schools.com." [https://www.w3schools.com/cs/cs\\_intro.php#:~:text=C%23%20Introduction&text=C%23%20is%20pronounced%20%22C%2DSharp,And%20much%2C%20much%20more!](https://www.w3schools.com/cs/cs_intro.php#:~:text=C%23%20Introduction&text=C%23%20is%20pronounced%20%22C%2DSharp,And%20much%2C%20much%20more!) (Accessed: 04 May 2025)
- [7] Gewarren, "Fundamentals of garbage collection - .NET," Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals> (Accessed: 04 May 2025)
- [8] GeeksforGeeks, "Garbage collection in C# | .NET Framework," *GeeksforGeeks*, Mar. 11, 2025. <https://www.geeksforgeeks.org/garbage-collection-in-c-sharp-dot-net-framework/> (Accessed: 04 May 2025)
- [9] Gewarren, ".NET garbage collection - .NET," Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/> (Accessed: 04 May 2025)


## REFERENCES

- [10] “As a low-level game dev, I consider C# a nearly perfect, wonderful language, tra... | Hacker News.” <https://news.ycombinator.com/item?id=14169152> (Accessed: 04 May 2025)
- [11] Stackexchange.com. [Online]. Available: <https://gamedev.stackexchange.com/questions/7/what-are-typical-pitfalls-when-writing-games-with-a-managed-language-like-c>. [Accessed: 04 May 2025].
- [12] GeeksforGeeks, “MiniMax Algorithm in Artificial Intelligence,” GeeksforGeeks, Apr. 07, 2025. <https://www.geeksforgeeks.org/mini-max-algorithm-in-artificial-intelligence/> (Accessed: 04 May 2025)
- [13] GeeksforGeeks, “Minimax Algorithm in Game Theory | Set 4 (AlphaBeta Pruning),” GeeksforGeeks, Jan. 16, 2023. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/> (Accessed: 04 May 2025)
- [14] Wikipedia contributors, “Alpha–beta pruning,” Wikipedia, Apr. 04, 2025. [https://en.wikipedia.org/wiki/Alpha%E2%80%93beta\\_pruning](https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning) (Accessed: 04 May 2025)
- [15] Researchgate.net. [Online]. Available: [https://www.researchgate.net/publication/338855656\\_INTELLIGENT\\_AGENTS\\_FOR\\_SOLVING\\_THE\\_GAME\\_DOTS\\_AND\\_BOXES](https://www.researchgate.net/publication/338855656_INTELLIGENT_AGENTS_FOR_SOLVING_THE_GAME_DOTS_AND_BOXES) (Accessed: 05 May 2025)
- [16] Aaai.org. [Online]. Available: <https://cdn.aaai.org/ojs/8144/8144-13-11671-1-2-20201228.pdf> (Accessed: 04 May 2025)
- [17] “Dots and Boxes | Gametable.org,” Gametable.org. <https://gametable.org/games/dots-and-boxes/> (Accessed: 04 May 2025)
- [18] O. Manners, “Dots and boxes.” <https://dotsandboxes.org/> (Accessed: 04 May 2025)
- [19] Coolmath Games, “Dots and Boxes - Play it Online at Coolmath Games,” Coolmath Games, May 02, 2025. <https://www.coolmathgames.com/0-dots-and-boxes> (Accessed: 04 May 2025)
- [20] M.-J. Suárez-Barón, H.-J. Rincón-Díaz, C.-D. González-Rodríguez, and J.-S. Gonzalez-Sanabria, “Linja: a mobile application based on Minimax strategy and game theory,” 2022. <https://www.redalyc.org/journal/4139/413971146003/html/> (Accessed: 05 May 2025)

## REFERENCES

- [21] Analysis of Minimax algorithm using tic-tac-toe. Available at: [https://www.researchgate.net/publication/346813363\\_Analysis\\_of\\_Minimax\\_Algorithm\\_Using\\_Tic-Tac-Toe](https://www.researchgate.net/publication/346813363_Analysis_of_Minimax_Algorithm_Using_Tic-Tac-Toe) (Accessed: 05 May 2025)
- [22] “Universal Windows Platform | Visual Studio,” Visual Studio, May 02, 2023. <https://visualstudio.microsoft.com/vs/features/universal-windows-platform/> (Accessed: 06 May 2025)
- [23] Stevewhims, “Windows game development guide - UWP applications,” Microsoft Learn, Aug. 22, 2024. <https://learn.microsoft.com/en-us/windows/uwp/gaming/e2e> (Accessed: 06 May 2025)
- [24] “What is C# Programming? A Beginner’s Guide | Pluralsight,” May 13, 2024. <https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c-#:~:text=What%20is%20C%23> (Accessed: 06 May 2025)
- [25] S. Singh, “Recursion in C# - Scaler topics,” Scaler Topics, Nov. 06, 2023. <https://www.scaler.com/topics/csharp/recursion-in-csharp/> (Accessed: 06 May 2025)
- [26] Codex, A.C. (2023) Using the event-driven model in .NET C#, Reintech media. Available at: <https://reintech.io/blog/event-driven-model-in-net-c-sharp-tutorial> (Accessed: 06 May 2025)
- [27] QuinnRadich, “What’s a Universal Windows Platform (UWP) app? - UWP applications,” Microsoft Learn, Aug. 21, 2024. <https://learn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide> (Accessed: 06 May 2025)
- [28] Wikipedia contributors, “Extensible application markup language,” Wikipedia, May 16, 2024, [https://en.wikipedia.org/wiki/Extensible\\_Application\\_Markup\\_Language](https://en.wikipedia.org/wiki/Extensible_Application_Markup_Language) (Accessed: 06 May 2025)

## POSTER



**UTAR**  
UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION AND  
COMMUNICATION TECHNOLOGY**

***Application of Minimax Algorithm in  
Dots and Boxes Game***

**DOTS  
&  
BOXES**

---

***Introduction***

The project integrates the Minimax algorithm to enhance the AI opponent, improving strategic depth and gameplay experience.

***Objective***

Develop an AI opponent that can adapt and compete strategically with human players.

---

***Proposed Method***

1

The project uses the Minimax algorithm, a game theory-based approach, for decision-making.

2

Alpha-Beta pruning is applied to minimize unnecessary calculations, improving the efficiency of the AI.

---


***Why is this system better than the existing one?***

1. Better Strategy
2. AI Adaptation
3. Optimized AI Performance
4. Balanced Difficulty

***Conclusion***

The project wants to develop an AI opponent that can make strategic decisions, offering a more engaging and competitive gaming experience.

---



Project Developer: Tan Ah Hwa  
Project Supervisor: Mr Lee Heng Yew

