Shape, square

Description automatically generated

TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

-----🙞🙜🕮🙞🙜-----

  
  
  
  
  
  
  
ĐỒ ÁN

**THIẾT KẾ KIẾN TRÚC PHẦN MỀM**

NGÀNH CÔNG NGHỆ THÔNG TIN

**XÂY DỰNG ỨNG DỤNG CHƠI TRÒ CHƠI TIC-TAC-TOE**

|  |  |
| --- | --- |
| Mã đồ án: | **K66** |
| Sinh viên thực hiện: | **Bùi Văn Đức**  **Nguyễn Tuấn Đạt** |
| Mã sinh viên: | **0187066 - 0185866** |
| Lớp: | **66PM3** |
| Giảng viên hướng dẫn: | **ThS. Phạm Hữu Tùng** |

Hà Nội, 11/2024

1. **Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành ĐAMH**

*Hà Nội, ngày tháng 05 năm 2025*

Giảng viên hướng dẫn

*ThS. Phạm Hữu Tùng*

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành đến thầy Phạm Hữu Tùng – người đã tận tâm hướng dẫn, chỉ bảo và truyền cảm hứng suốt quá trình thực hiện đề tài đồ án Xây dựng ứng dụng trò chơi XO. Nhờ sự hỗ trợ quý báu của thầy, chúng em đã học hỏi được nhiều kiến thức bổ ích, kỹ năng lập trình và tư duy hệ thống bài bản. Chúng em rất biết ơn sự kiên nhẫn, tận tâm và những góp ý quý giá giúp chúng em hoàn thành đề tài một cách tốt nhất.

*Tháng 5 Năm 2025*

MỤC LỤC

[**1.** **Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành ĐAMH** 2](#_Toc199232939)

[CHƯƠNG I. ĐẶT VẤN ĐỀ 6](#_Toc199232940)

[**Đặt vấn đề** 6](#_Toc199232941)

[**Mục tiêu** 6](#_Toc199232942)

[**Phạm vi và giới hạn** 7](#_Toc199232943)

[**Phương pháp thực hiện** 7](#_Toc199232944)

[**1.1. *Ngôn ngữ lập trình*** 7](#_Toc199232945)

[**1.2. *Công cụ sử dụng*** 7](#_Toc199232946)

[CHƯƠNG II: CƠ SỞ LÝ THUYẾT VÀ KHOA HỌC 8](#_Toc199232947)

[**1.** **Khái niệm** 8](#_Toc199232948)

[**1.1. Thuật toán Minimax 8**](#_Toc199232949)

[**1.2. Giới hạn độ sâu 9**](#_Toc199232952)

[**1.3. Cắt tỉa alpha-beta 10**](#_Toc199232953)

[**1.4. Hàm Heuristic 12**](#_Toc199232954)

[**1.5. Các hàm khác 13**](#_Toc199232955)

[CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 19](#_Toc199232956)

[**1. Yêu cầu chức năng** 19](#_Toc199232957)

[**2. Yêu cầu phi chức năng** 19](#_Toc199232958)

[**3.** **Sơ đồ Usecase** 20](#_Toc199232961)

[**3.1. *Sơ đồ tổng quát*** 20](#_Toc199232963)

[**4. Sơ đồ Activity** 20](#_Toc199232964)

[**4.1. Chức năng Chơi người – người** 20](#_Toc199232965)

[CHƯƠNG III. THIẾT KẾ HỆ THỐNG 22](#_Toc199232966)

[**1.** **Lựa chọn mô hình kiến trúc** 22](#_Toc199232967)

[**1.** **Biểu đồ tuần tự** 22](#_Toc199232968)

[2. Thiết kế giao diện người dùng (Mockup / Wireframe) 23](#_Toc199232969)

[***1.*** ***Môi trường triển khai*** 26](#_Toc199232970)

[**2.** **Cài đặt hệ thống** 26](#_Toc199232971)

[**2.1. Yêu cầu hệ thống** 26](#_Toc199232974)

[**2.2. Các bước cài đặt chương trình** 26](#_Toc199232975)

[CHƯƠNG V: KẾT QUẢ VÀ ĐÁNH GIÁ 27](#_Toc199232976)

[1**. Kết quả thử nghiệm hệ thống** 27](#_Toc199232977)

[**2. Đánh giá hiệu quả hệ thống** 27](#_Toc199232978)

# 

# CHƯƠNG I. ĐẶT VẤN ĐỀ

## **Đặt vấn đề**

Trong thế giới số hóa đầy biến động, ứng dụng giải trí không đơn thuần là một công cụ để giải tỏa căng thẳng mà còn trở thành phương tiện thúc đẩy sự phát triển tư duy logic, khả năng xử lý tình huống và rèn luyện sự kiên nhẫn của con người. Trò chơi XO – với luật chơi đơn giản nhưng ẩn chứa những chiến thuật sâu sắc – đã trở thành biểu tượng của sự kết hợp giữa giải trí và trí tuệ, được đông đảo mọi thế hệ ưa chuộng.

Việc xây dựng ứng dụng trò chơi XO trên nền tảng phần mềm hiện nay không chỉ nhằm mục đích tái hiện lại trải nghiệm truyền thống mà còn mở rộng ra nhiều hướng phát triển hiện đại, giúp người chơi có thể tương tác một cách mượt mà, trực quan trên các thiết bị di động hoặc máy tính. Đây cũng là cơ hội để người học và phát triển công nghệ tiếp cận sâu hơn với các kiến thức lập trình cơ bản như cấu trúc dữ liệu, thuật toán, xử lý sự kiện và thiết kế giao diện người dùng thân thiện.

Hơn nữa, trong bối cảnh mạng Internet phủ sóng rộng rãi, việc phát triển thêm các tính năng như chơi đa người trực tuyến sẽ tăng tính tương tác, kết nối cộng đồng, đồng thời mang lại trải nghiệm chơi game phong phú hơn, phù hợp với xu thế phát triển xã hội số hiện nay. Đó chính là lý do vì sao bài toán xây dựng ứng dụng chơi XO không chỉ mang tính giải trí mà còn trở thành một đề tài đồ án lý tưởng giúp sinh viên rèn luyện kỹ năng tư duy hệ thống, nâng cao năng lực giải quyết vấn đề thực tế.

Cuối cùng, dự án này còn phản ánh rõ nét tinh thần truyền thống “học đi đôi với hành”, tạo nền tảng vững chắc để người học có thể phát triển sự nghiệp công nghệ trong tương lai, đồng thời góp phần bảo tồn và phát huy giá trị trò chơi dân gian dưới dạng công nghệ số hiện đại.

## **Mục tiêu**

Trò chơi Tic Tac Toe hướng đến việc tạo ra một sân chơi trí tuệ đơn giản nhưng đầy thử thách, giúp người chơi rèn luyện tư duy chiến thuật, khả năng dự đoán và phản xạ nhanh nhạy. Mục tiêu chính là người chơi phải sắp xếp thành công chuỗi ba, bốn hoặc năm ký hiệu giống nhau (O hoặc X) theo hàng ngang, hàng dọc hoặc đường chéo trước đối thủ để giành chiến thắng. Trò chơi hướng đến chức năng sau:

* Cho phép người chơi đánh dấu X hoặc O trên bảng 3x3, 5x5 hay 7x7 theo lượt.
* Quản lý lượt chơi tự động, đảm bảo mỗi người chỉ được đánh một lần trong một lượt.
* Kiểm tra và xác định kết quả thắng, thua hoặc hòa ngay khi đủ điều kiện.
* Thông báo kết quả trò chơi rõ ràng, trực quan đến người chơi.
* Cung cấp tính năng đặt lại bàn chơi để bắt đầu ván mới dễ dàng.
* Hỗ trợ chơi hai người trên cùng một thiết bị hoặc chơi với máy (AI đơn giản).

## **Phạm vi và giới hạn**

**Phạm vi**:

* **Đối tượng người chơi:** Ứng dụng được thiết kế để chơi cho 2 người trên cùng một thiết bị hoặc người chơi có thể đối đầu với máy tính (AI) trong các ván đơn.
* **Chế độ chơi**: Bao gồm chế độ chơi đơn (1 người vs máy) và chế độ chơi đôi (2 người cùng chơi trên cùng một thiết bị).
* **Kết quả:** Ứng dụng sẽ tự động xác định kết quả thắng, thua hoặc hòa và thông báo ngay khi có người chiến thắng hoặc ván đấu kết thúc.

**Giới hạn:**

* **Đối tượng**: Ứng dụng chỉ hỗ trợ chơi với hai người cùng một lúc. Chế độ chơi nhiều người trực tuyến hoặc với nhiều người hơn không được hỗ trợ trong phiên bản này.
* **Chế độ AI:** Trò chơi với máy chỉ sử dụng một thuật toán đơn giản để đối phó.
* **Chức năng mạng:** Phiên bản này không hỗ trợ chơi trực tuyến hoặc kết nối nhiều người chơi từ xa.

## **Phương pháp thực hiện**

* 1. ***Ngôn ngữ lập trình***

Dự án game Tic Tac Toe sử dụng C# làm ngôn ngữ lập trình chính, phát triển trên nền tảng Unity.

* 1. ***Công cụ sử dụng***
     1. ***Phần mềm hỗ trợ thiết kế hệ thống***
* Microsoft Word 365 : Viết tài liệu đặc tả hệ thống.
* Visual Paradigm 17.1: Vẽ các biểu đồ Usecase, Sequence.
  + 1. ***Công cụ xây dựng ứng dụng***
* Unity: Xây dựng giao diện trò chơi.
* Visual Studio : Xây dựng logic trò chơi.

# CHƯƠNG II: CƠ SỞ LÝ THUYẾT VÀ KHOA HỌC

## **Khái niệm**

### Thuật toán Minimax

**Khái niệm**

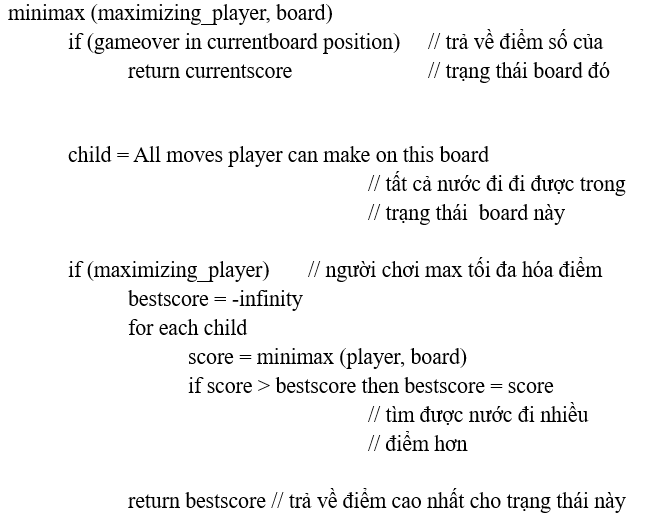
Minimax là một loại thuật toán quay lui (backtracking) được sử dụng trong quá trình ra quyết định và lý thuyết trò chơi để tìm ra nước đi tối ưu cho người chơi, giả sử rằng đối thủ cũng chơi những nước đi tối ưu nhất.

Trong thuật toán Minimax, hai người chơi được gọi là **Maximizer** và **Minimizer**. Maximizer cố gắng đạt điểm cao nhất có thể trong khi Minimizer cố gắng làm ngược lại và đạt điểm thấp nhất có thể.

Mỗi trạng thái trên bàn cờ đều có một giá trị liên quan đến nó. Trong một trạng thái nhất định, nếu Maximizer có lợi thế thì điểm của bàn cờ sẽ có xu hướng là một giá trị dương. Nếu Minimizer có lợi thế trong trạng thái bàn cờ đó thì điểm của bàn cờ sẽ có xu hướng là một giá trị âm.

* + 1. **Thuật toán**

**1.1.2.1. Minimax thuần**



A white background with black text

AI-generated content may be incorrect.

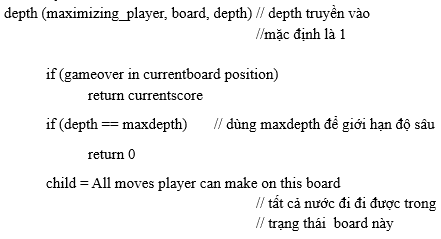


### Giới hạn độ sâu

depth: độ sâu hiện tại của cây trò chơi.

* Nếu kích cỡ không phải 3x3 mà nhiều hơn nữa (5x5, 7x7,…) thì không gian mẫu sẽ lớn hơn rất nhiều nên cần giới hạn độ sâu để giới hạn lại phạm vi dự đoán.
* Ngoài ra dùng depth có thể tăng khả năng dự đoán chính xác hơn. Giả sử có nước thắng ở độ sâu thấp hơn (nước đi gần hơn) thì sẽ ưu tiên nước đi đó.
* Nhược điểm là không thể dự đoán hết các nước đi nên giới hạn maxdepth càng thấp thì các nước đi sẽ càng không tối ưu => phải dùng thêm 1 số hàm heuristic để cải thiện nhược điểm này (ví dụ như hàm heuristic tính toán lượng điểm dựa trên chuỗi ô cờ của người chơi liên tiếp)

**Thuật toán**

****

**A white paper with black text

AI-generated content may be incorrect.**

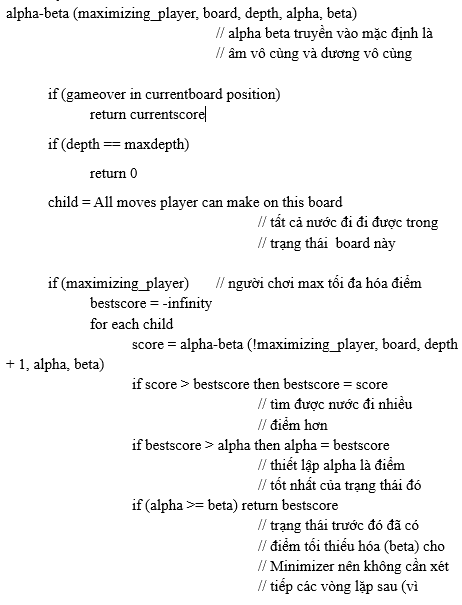
### Cắt tỉa alpha-beta

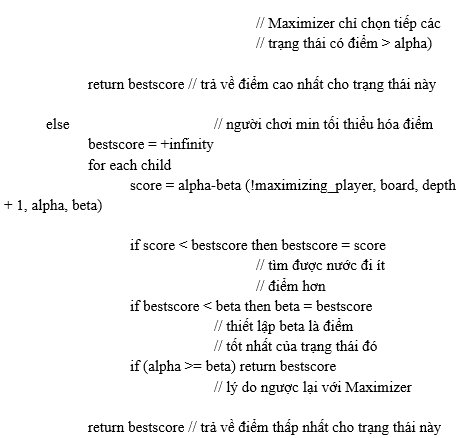
Cắt tỉa alpha-beta là kĩ thuật cải thiện thuật toán Min-Max bằng cách loại bỏ các nhánh không ảnh hưởng đến quyết định cuối cùng.

Với **alpha** là giá trị tốt nhất mà Maximizer có thể đảm bảo cho đến trạng thái đó, **beta** giá trị tốt nhất mà Minimizer có thể đảm bảo cho đến trạng thái đó.

Trong quá trình duyệt, khi alpha >= beta thì sẽ cắt các cành còn lại (hay trong vòng lặp thì bỏ các vòng lặp còn lại không cần duyệt qua nữa)

**Thuật toán**

****

****

### Hàm Heuristic

* **Lý do:** Sử dụng hàm heuristic giúp AI đánh giá mức độ được của một trạng thái mà không cần duyệt toàn bộ cây trò chơi (khi mà đã đạt đến độ sâu tối đa).
* **Mục đích:** Đưa ra số điểm đại diện cho lợi thế của AI so với người chơi. Điểm càng cao thì càng có lợi cho AI.
* **Cách tính điểm:**
  + Chuỗi 5 quân liên tiếp: 100000 điểm.
  + Chuỗi ngắn hơn (1, 2, 3, 4 quân): Điểm dựa trên số quân liên tiếp và trạng thái hai đầu mút (mở cả hai đầu, mở một đầu, hoặc chặn cả hai đầu – được thể hiện ở hàm ScorePattern phía dưới).
  + Chuỗi mở được ưu tiên vì có tiềm năng mở rộng.
  + Điểm của AI được cộng, điểm của người chơi được trừ để cân bằng.

**1.4.1. Hàm ScorePattern**

private int ScorePattern(int count, bool isOpenStart, bool isOpenEnd)

{

if (count == 4)

return isOpenStart && isOpenEnd ? 10000 : (isOpenStart || isOpenEnd ? 5000 : 0);

if (count == 3)

return isOpenStart && isOpenEnd ? 1000 : (isOpenStart || isOpenEnd ? 500 : 0);

if (count == 2)

return isOpenStart && isOpenEnd ? 100 : (isOpenStart || isOpenEnd ? 50 : 0);

if (count == 1)

return isOpenStart && isOpenEnd ? 10 : (isOpenStart || isOpenEnd ? 5 : 0);

return 0;

}

Tính điểm cho một chuỗi liên tiếp dựa trên:

* + Số quân (count).
  + Trạng thái đầu mút (isOpenStart, isOpenEnd): Chuỗi có mở (ô còn trống) hay bị chặn.
    1. **Hàm EvaluateLine**
* **Mục đích:**
  + Đánh giá một dòng cụ thể (hàng hoặc cột) trên bảng, tính điểm cho các chuỗi liên tiếp của player và opponent.
  + Dòng được xác định bởi điểm bắt đầu (startRow, startCol) và hướng di chuyển (dRow, dCol) (ví dụ, (0, 1) cho hàng ngang, (1, 0) cho cột dọc).
  + Cách duyệt thì bắt đầu mỗi hàng (cột) thì isOpenStart sẽ là false, lý do là vì không có ô trước đó nên coi là bị chặn 1 đầu mút. Sau thì cứ đếm xem đó là chuỗi cross (X) hay nought (O) cho đến khi bị cắt chuỗi (tức ô đó trống hoặc là 1 nước của đối thủ) thì sẽ tính điểm của chuỗi đó dựa trên EvaluatePattern. Kết thúc vòng lặp mà còn chuỗi thì tính điểm nốt chuỗi đó với isOpenEnd là false, đơn giản là vì ô cuối cùng của hàng (cột) nên không có ô sau.
* **Thuật toán:**

private int EvaluateLine(Player[,] matrix, int startRow, int startCol, int dRow, int dCol, Player player, Player opponent)

{

int score = 0;

int playerCount = 0;

int opponentCount = 0;

bool isOpenStart = false;

for (int i = 0; i < boardSize; i++)

{

int row = startRow + i \* dRow;

int col = startCol + i \* dCol;

if (row < 0 || row >= boardSize || col < 0 || col >= boardSize)

break;

if (matrix[row, col] == player)

{

if (opponentCount > 0)

{

score -= ScorePattern(opponentCount, isOpenStart, false);

isOpenStart = false;

opponentCount = 0;

}

playerCount++;

}

else if (matrix[row, col] == opponent)

{

if (playerCount > 0)

{

score += ScorePattern(playerCount, isOpenStart, false);

isOpenStart = false;

playerCount = 0;

}

opponentCount++;

}

else

{

if (playerCount > 0)

{

score += ScorePattern(playerCount, isOpenStart, true);

isOpenStart = true;

playerCount = 0;

}

else if (opponentCount > 0)

{

score -= ScorePattern(opponentCount, isOpenStart, true);

isOpenStart = true;

opponentCount = 0;

}

}

}

if (playerCount > 0)

score += ScorePattern(playerCount, isOpenStart, false);

if (opponentCount > 0)

score -= ScorePattern(opponentCount, isOpenStart, false);

return score;

}

* + 1. **Hàm Evaluate Diagonal**
* **Mục đích:**
  + Đánh giá một đường chéo (chéo chính hoặc chéo phụ) trên bảng, tương tự EvaluateLine nhưng cho các đường chéo.
  + offset: Quyết định đường chéo nào được đánh giá (từ -boardSize + 1 đến boardSize - 1).
  + isMainDiagonal:
    - true: Chéo chính (trái trên → phải dưới, như (0,0), (1,1), (2,2)).
    - false: Chéo phụ (phải trên → trái dưới).
  + Cách duyệt giống với EvaluateLine em đã trình bày phía trên.
* **Thuật toán**

private int EvaluateDiagonal(Player[,] matrix, int offset, bool isMainDiagonal, Player player, Player opponent)

{

int score = 0;

int startRow = offset >= 0 ? offset : 0;

int startCol = offset >= 0 ? 0 : -offset;

int count = 0;

int playerCount = 0;

int opponentCount = 0;

bool isOpenStart = false;

while (startRow < boardSize && startCol < boardSize)

{

int row = isMainDiagonal ? startRow : startRow;

int col = isMainDiagonal ? startCol : boardSize - 1 - startRow + offset;

if (row >= boardSize || col >= boardSize || col < 0)

break;

if (matrix[row, col] == player)

{

if (opponentCount > 0)

{

score -= ScorePattern(opponentCount, isOpenStart, false);

isOpenStart = opponentCount == 0;

opponentCount = 0;

}

playerCount++;

}

else if (matrix[row, col] == opponent)

{

if (playerCount > 0)

{

score += ScorePattern(playerCount, isOpenStart, false);

isOpenStart = playerCount == 0;

playerCount = 0;

}

opponentCount++;

}

else

{

if (playerCount > 0)

{

score += ScorePattern(playerCount, isOpenStart, true);

isOpenStart = true;

playerCount = 0;

}

else if (opponentCount > 0)

{

score -= ScorePattern(opponentCount, isOpenStart, true);

isOpenStart = true;

opponentCount = 0;

}

}

startRow++;

startCol++;

count++;

}

if (playerCount > 0)

score += ScorePattern(playerCount, isOpenStart, false);

if (opponentCount > 0)

score -= ScorePattern(opponentCount, isOpenStart, false);

return score;

}

* + 1. **Hàm EvaluateBoard**
* **Mục đích:** Xét điểm từng hàng, từng cột, từng đường chéo thông qua các hàm trên đã trình bày.
* **Thuật toán**

private int EvaluateBoard(Player[,] matrix, bool isMaximizing)

{

Player player = isMaximizing ? Player.NOUGHT : Player.CROSS;

Player opponent = isMaximizing ? Player.CROSS : Player.NOUGHT;

int score = 0;

// Đánh giá các hàng, cột, chéo

for (int i = 0; i < boardSize; i++)

{

score += EvaluateLine(matrix, i, 0, 0, 1, player, opponent); // Hàng

score += EvaluateLine(matrix, 0, i, 1, 0, player, opponent); // Cột

}

// Chéo chính và chéo phụ

for (int i = -boardSize + 1; i < boardSize; i++)

{

score += EvaluateDiagonal(matrix, i, true, player, opponent); // Chéo chính

score += EvaluateDiagonal(matrix, i, false, player, opponent); // Chéo phụ

}

return score;

}

# CHƯƠNG III: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## **Yêu cầu chức năng**

* 1. **Chế độ chơi:**
* Chơi đơn (1 người vs máy): Người chơi có thể đối đầu với máy tính trong một ván chơi đơn giản. Máy sẽ tự động chọn ô trống để đi.
* Chơi đôi (2 người chơi trên cùng thiết bị): Hai người chơi có thể thay phiên nhau đánh dấu X và O trên bảng chơi.
  1. **Bảng chơi:**
* Bảng 3x3; 5x5; 7x7: Ứng dụng sử dụng bảng kích thước tùy chọn mỗi ván đấu.
* Đánh dấu X và O: Mỗi người chơi sẽ chọn X hoặc O và đánh dấu lần lượt vào ô trống.
* Cập nhật ô đã chọn: Khi người chơi đánh dấu vào ô, bảng sẽ được cập nhật ngay lập tức và không thể thay đổi khi đã có dấu.
  1. **Kiểm tra kết quả:**
* Thắng cuộc: Ứng dụng tự động kiểm tra nếu có ba dấu X hoặc O liên tiếp theo hàng ngang, dọc hoặc chéo.
* Hòa: Nếu tất cả các ô đã được điền mà không có người chiến thắng, trò chơi sẽ kết thúc với kết quả hòa.
* Thông báo kết quả: Sau mỗi ván đấu, ứng dụng sẽ thông báo người thắng cuộc, hòa, hoặc ván đấu chưa kết thúc.
  1. **Khởi động lại ván mới:**
* Reset bàn chơi: Sau khi kết thúc ván đấu, người chơi có thể khởi động lại trò chơi để bắt đầu một ván mới mà không cần phải thoát ra khỏi ứng dụng.
  1. **Giao diện người dùng:**
* Giao diện trực quan: Bảng chơi sẽ được hiển thị rõ ràng với các ô vuông có kích thước đều nhau. Người chơi chỉ cần chạm vào ô để đánh dấu X hoặc O.

## **Yêu cầu phi chức năng**

* 1. **Hiệu suất:**

Ứng dụng phải có tốc độ phản hồi nhanh, không có độ trễ trong các thao tác chơi, và tối ưu bộ nhớ.

* 1. **Tính ổn định:**

Ứng dụng cần duy trì trạng thái ổn định, khôi phục sau sự cố mà không làm gián đoạn trò chơi.

* 1. **Tính khả dụng:**

Giao diện dễ sử dụng, trực quan, hỗ trợ người chơi ngay từ lần đầu sử dụng.

* 1. **Dễ bảo trì:**

Mã nguồn dễ đọc, dễ bảo trì và có tài liệu hướng dẫn rõ ràng.



## **Sơ đồ Usecase**

1. 1. **A diagram of a diagram

      AI-generated content may be incorrect.*Sơ đồ tổng quát***

## **Sơ đồ Activity**

* 1. **Chức năng Chơi người – người**

A diagram of a company

AI-generated content may be incorrect.

# CHƯƠNG III. THIẾT KẾ HỆ THỐNG

## **Lựa chọn mô hình kiến trúc**

* Chương trình được thiết kế dựa trên mô hình **Monolithic.**
* **Đặc điểm mô hình Monolithic:**
  + **Triển khai tập trung**: tất cả code được build, chạy, deploy như một ứng dụng duy nhất.
  + **Không phân tán module**: không có API hay microservice giao tiếp qua mạng.
  + **Mọi chức năng nội tại**: xử lý logic, UI, trạng thái... đều nội bộ.
* **Sử dụng mô hình trong dự án**

|  |  |
| --- | --- |
| Thành phần | Vai trò |
| Board.cs, Cell.cs | Xử lý logic thắng, thua |
| SceneGame.unity | Dàn dựng không gian chơi |
| InputSystem\_Actions | Nhận thao tác người chơi |
| PR\_Cell | Giao diện |

* + Tất cả thành phần đều **liên kết trực tiếp với nhau**, chia sẻ tài nguyên chung và **không giao tiếp thông qua API, Queue hoặc Module phân tán**.**Biểu đồ tuần tự (*các chức năng chính*)**



## **Biểu đồ tuần tự**

A diagram of a diagram

AI-generated content may be incorrect.

Biểu đồ tuần tự - Chơi với máy

## **Thiết kế giao diện người dùng (Mockup / Wireframe)**

A screenshot of a game

AI-generated content may be incorrect.

Màn hình chính – chọn chế độ chơi

A screenshot of a screen

AI-generated content may be incorrect.

Chọn bảng chơi

A screenshot of a computer game

AI-generated content may be incorrect.

Giao diện trò chơi 7x7

A screenshot of a game

AI-generated content may be incorrect.

Giao diện kết quả trò chơi – Hòa

A screenshot of a computer

AI-generated content may be incorrect.

Giao diện kết quả trò chơi – Bot chiến thắng

**CHƯƠNG IV: CÀI ĐẶT VÀ TRIỂN KHAI HỆ THỐNG**

## ***Môi trường triển khai***

* Visual Studio
* Unity

1. **Cài đặt hệ thống**
3. 1. **Yêu cầu hệ thống**

* Windows 10 trở lên
  1. **Các bước cài đặt chương trình**
* Biên dịch chương trình

# CHƯƠNG V: KẾT QUẢ VÀ ĐÁNH GIÁ

## 1**. Kết quả thử nghiệm hệ thống**

* Ứng dụng **Tic Tac Toe** đã triển khai thành công trên nền tảng Unity, chạy ổn định trên cả máy tính và thiết bị di động (nếu build).
* Hệ thống đảm bảo đầy đủ các chức năng cốt lõi:
  + Khởi tạo bàn chơi với kích thước linh hoạt (3x3, 5x5, 7x7)
  + Chọn chế độ PvP hoặc đánh với máy (AI)
  + Kiểm tra thắng/thua/hòa theo thời gian thực
  + Giao diện thân thiện, thao tác đơn giản phù hợp cho mọi lứa tuổi

## **2. Đánh giá hiệu quả hệ thống**

* **Ứng dụng đáp ứng đúng mục tiêu ban đầu**: hỗ trợ người dùng giải trí, luyện tư duy logic thông qua trò chơi trí tuệ kinh điển.
* **Giảm thiểu thao tác thủ công**: toàn bộ luật chơi, kiểm tra thắng, xử lý lượt đi của máy đều được hệ thống tự động hóa hoàn toàn.
* **Tích hợp trí tuệ nhân tạo (AI)** thông qua thuật toán **Minimax**, giúp tăng độ thử thách khi chơi với máy.
* **Gợi ý cải tiến tương lai**:
  + Thêm hiệu ứng âm thanh, animation để tăng trải nghiệm người dùng
  + Mở rộng hỗ trợ nhiều nền tảng: Android, WebGL,...
  + Tích hợp bảng xếp hạng điểm số hoặc lưu lịch sử trận đấu
  + Tùy biến thêm các chế độ chơi nâng cao như 2v2, chơi thời gian thực online