TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

LẬP TRÌNH TRÒ CHƠI CỜ CARO

Giảng viên hướng dẫn : *TS. Đinh Tuấn Long*

Nhóm sinh viên thực hiện : 7C2038.17-2.20232024.2 - N17

Nguyễn Hoàng Anh - 2010A01

Cao Xuân Biên - 2010A05

Nguyễn Hợp Trường - 2010A05

MÔN HỌC : CHUYÊN ĐỀ THỰC TẬP NGÀNH

Hà Nội – 2024

TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

LẬP TRÌNH TRÒ CHƠI CỜ CARO

Giảng viên hướng dẫn : *TS. Đinh Tuấn Long*

Nhóm sinh viên thực hiện : 7C2038.17-2.20232024.2 - N17

Nguyễn Hoàng Anh - 2010A01

Cao Xuân Biên - 2010A05

Nguyễn Hợp Trường - 2010A05

MÔN HỌC : CHUYÊN ĐỀ THỰC TẬP NGÀNH

Hà Nội – 2024

LỜI CẢM ƠN

Lời đầu tiên Nhóm 17 xin chân thành cảm ơn đến Thầy Đinh Tuấn Long - người đã trực tiếp giảng dạy, truyền đạt những kiến thức thật bổ ích cho em trong quá trình học, đó sẽ là nền tảng cơ bản, là hành trang vô cùng quý giá, là bước đầu tiên cho nhóm em bước vào sự nghiệp sau này trong tương lai.

Dù đã có sự hỗ trợ và chỉ dẫn từ Thầy, nhưng đôi khi trong quá trình học, nhóm em có thể gặp phải những thách thức và khó khăn khi không thực sự hài lòng của thầy. Tuy nhiên, điều quan trọng là nhóm em vẫn tích cực học hỏi cố gắng thực hiện theo yêu cầu của Thầy.

Ngoài việc thực hiện hoàn thành những mục tiêu mà nhóm đã đặt ra trong quá trình học. Sau đây là những ý tưởng, yêu cầu của Thầy và mức độ hoàn thành các yêu cầu trong 3 buổi báo cáo mà Nhóm 17 đã tổng hợp lại.

Buổi	Yêu cầu	Tiến độ thực hiện
1	Thực hiện xử lý thêm chế độ chơi giữa người với máy	Hoàn thành
2	Ở chế độ chơi với máy, với ý tưởng đơn giản : • Máy thực hiện đi trong phạm vi xung quanh nước cờ mà người chơi đi (8 ô xung quanh) • Thực hiện tính điểm, để máy có thể lựa chọn nước đi cao nhất để làm nước đi	Hoàn thành
3	Thực hiện xử lý Logic, khi có 1 ô chặn đầu chuỗi 5 ô liên tiếp thì không được tính là thắng cuộc Ý tưởng thực hiện, mở rộng phạm vi tìm kiếm, tăng độ khó của máy • Khi người chơi đạt 4 ô liên tiếp, máy sẽ tự tìm kiếm và chặn đầu không cho người chơi dành thắng cuộc	Hoàn thành Đang tiến hành

MỤC LỤC

BẢNG PHÂN CÔNG CÔNG VIỆC			
Công việc	Nguyễn Hoàng Anh	Cao Xuân Biên	Nguyễn Hợp Trường
Viết báo cáo	P		
Khảo sát hệ thống trò chơi	P	S	
Thiết kế giao diện trò chơi	P		S
Lập trình logic trò chơi	S	P	S
Thêm các chức năng nâng cao	S	S	P
Chạy thử nghiệm và kiểm tra ràng buộc	S	P	
Thực hiện sửa lỗi		P	S
Hoàn thiện báo cáo	P		

I. Giới thiệu đề tài

1.1. Tổng quan

- Đề tài lập trình trò chơi cờ Caro là một đề tài phù hợp với sinh viên ngành công nghệ thông tin. Đề tài giúp sinh viên nắm vững các kiến thức về lập trình game và vận dụng được các kiến thức đó để tạo ra một trò chơi đơn giản nhưng hấp dẫn.
- Trò chơi cờ Caro là một trò chơi chiến thuật đơn giản nhưng hấp dẫn, phù hợp với mọi lứa tuổi. Trò chơi có luật chơi đơn giản, dễ hiểu, nhưng đòi hỏi người chơi phải có tư duy chiến thuật và tính toán logic.

1.2. Mục tiêu

- Tạo được giao diện màn hình trò chơi cờ Caro với các thành phần cần thiết, bao gồm:
    - Bảng cờ
    - Hiển thị 2 quân cờ của người chơi
    - Hiển thị thời gian đếm ngược
- Lập trình vòng lặp chơi game, cho phép người chơi nhập quân cờ và kiểm tra người thắng cuộc.
- Thêm các tính năng nâng cao cho trò chơi, bao gồm:
    - Thêm chế độ chơi với máy
    - Hiệu ứng hình ảnh

II. Lịch trình thực hiện

STT	Công việc	Tuần 1	Tuần 2	Tuần 3	Tuần 4	Tuần 5	Tuần 6	Tuần 7	Tuần 8
1	Viết báo cáo	x	x	x					
2	Khảo sát hệ thống trò chơi		x	x					
3	Thiết kế giao diện trò chơi		x	x					
4	Lập trình logic trò chơi				x	x	x		
5	Thêm các chức năng nâng cao				x	x	x		
6	Chạy thử nghiệm và kiểm tra ràng buộc							x	
7	Thực hiện sửa lỗi							x	
8	Hoàn thiện báo cáo								x

III. Phương pháp thực hiện

3.1. Ngôn ngữ sử dụng : C#

- **C#** là ngôn ngữ lập trình hướng đối tượng, hiện đại và mạnh mẽ được phát triển bởi Microsoft. Nó được sử dụng để xây dựng nhiều ứng dụng khác nhau, bao gồm:
    - Ứng dụng Windows: C# là ngôn ngữ chính để phát triển các ứng dụng Windows Forms và WPF.
    - Ứng dụng web: C# có thể được sử dụng để tạo các ứng dụng web ASP.NET MVC và Web API.
    - Trò chơi: C# là ngôn ngữ phổ biến để phát triển trò chơi với Unity.
    - Ứng dụng di động: C# có thể được sử dụng để tạo các ứng dụng di động Xamarin.
    - Phần mềm doanh nghiệp: C# được sử dụng để xây dựng các ứng dụng web doanh nghiệp và các dịch vụ web.
- Đặc điểm nổi bật của C#:
    - Hướng đối tượng: C# là ngôn ngữ hướng đối tượng, giúp cho việc phát triển và bảo trì phần mềm trở nên dễ dàng hơn.
    - Có cú pháp đơn giản: C# có cú pháp dễ học và dễ sử dụng, tương tự như Java.
    - An toàn và bảo mật: C# được thiết kế để an toàn và bảo mật, giúp giảm thiểu nguy cơ lỗi và bảo vệ dữ liệu.
    - Có hiệu suất cao: C# được biên dịch sang mã máy, giúp cho nó có hiệu suất cao.

- Hỗ trợ đa nền tảng: C# có thể chạy trên nhiều nền tảng khác nhau, bao gồm Windows, macOS, Linux và Android.

3.2. Công nghệ sử dụng : Visual Studio 2022

- **Visual studio 2022:** là một môi trường phát triển tích hợp (IDE) từ Microsoft. Microsoft Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới ", được dùng để lập trình C++ và C# là chính. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.
- Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.
- Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C,[4] C++ và C++/CLI (thông qua Visual C++), VB.NET (thông qua Visual Basic.NET), C# (thông qua Visual C#) và F# (như của Visual Studio 2010[5]). Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng rẽ.

3.3. Framework được sử dụng (Window Forms)

- **Windows Forms** là thư viện lớp đồ họa (GUI) mã nguồn mở và miễn phí được tích hợp trong Microsoft .NET Framework hoặc Mono Framework. Nó cung cấp nền tảng để phát triển các ứng dụng khách phong phú cho máy tính để bàn, máy tính xách tay và máy tính bảng.
- Đặc điểm nổi bật của Windows Forms:
    - Dễ sử dụng: Windows Forms cung cấp giao diện trực quan và dễ sử dụng, giúp cho việc phát triển ứng dụng GUI trở nên dễ dàng hơn.
    - Hỗ trợ đa dạng: Windows Forms hỗ trợ nhiều loại điều khiển GUI khác nhau, bao gồm nút bấm, hộp văn bản, hộp kiểm, menu, v.v.
    - Mạnh mẽ: Windows Forms cho phép bạn tạo các ứng dụng GUI phức tạp với nhiều tính năng.
    - Có thể mở rộng: Windows Forms có thể được mở rộng bằng cách sử dụng các thư viện bên thứ ba.
- Ứng dụng của Windows Forms:
    - Ứng dụng kinh doanh: Windows Forms được sử dụng để phát triển các ứng dụng kinh doanh như phần mềm kế toán, phần mềm quản lý bán hàng, v.v.
    - Ứng dụng giáo dục: Windows Forms được sử dụng để phát triển các ứng dụng giáo dục như phần mềm học tập, phần mềm thi cử, v.v.
    - Ứng dụng giải trí: Windows Forms được sử dụng để phát triển các ứng dụng giải trí như trò chơi, phần mềm nghe nhạc, v.v

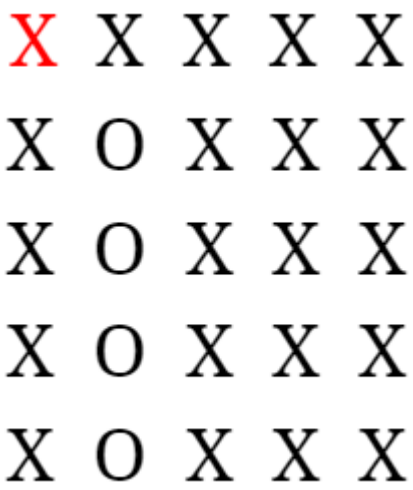
IV. Nội dung

4.1. Chương 1: Tổng quan về trò chơi cờ Caro

- Quy tắc chơi cờ Caro
    - Người chơi có quân X luôn chơi đầu tiên. Họ phải đặt nước đầu tiên tại trung tâm của bàn cờ. Người chơi quân cờ O di chuyển bằng cách đặt quân cờ tại một trong tám nút giao liền kề với quân cờ đen đã đi trước.
    - Người chơi cờ X sau đó phản ứng với nước đi vừa rồi của quân O và cứ thế cho đến khi có năm quân cờ cùng màu xuất hiện theo các hướng: dọc, ngang hoặc đường chéo. Khi một người chơi có 5 quân cờ cùng chữ theo các hướng này, ván đấu sẽ kết thúc ngay cả khi vẫn còn các thế cờ khác.
    - Trong trò chơi cờ caro, người bắt đầu có một chút lợi thế hơn so với đối thủ: Do đó, tốt nhất là nên được sắp xếp thứ tự đi trước xen kẽ để không có người chơi nào được ưu tiên. Mỗi người chơi chỉ có một lần di chuyển ở mỗi lượt. Sau khi đã đi, không thể đi lại hoặc xóa nước đi khỏi bàn đấu.
- Thuật toán tìm người thắng cuộc
    - Thuật toán tìm người thắng cuộc trong cờ Caro là một thuật toán đơn giản, và có nhiều cách tiếp cận khác nhau. Dưới đây là một phương pháp cơ bản sử dụng kiểm tra trong từng hàng, cột và đường chéo trên bảng cờ:
    - Kiểm Tra Theo Chiều Ngang, Dọc và Đường Chéo:
        - ➢ Duyệt qua từng hàng, cột và đường chéo trên bảng cờ.
        - ➢ Đối với mỗi ô trên dòng, cột hoặc đường chéo, kiểm tra xem có một chuỗi liên tiếp các quân cờ của cùng một người chơi hay không.

➢ Nếu có một chuỗi gồm n quân cờ của một người chơi, nếu n đủ lớn (thường là 5), người chơi đó là người chiến thắng.

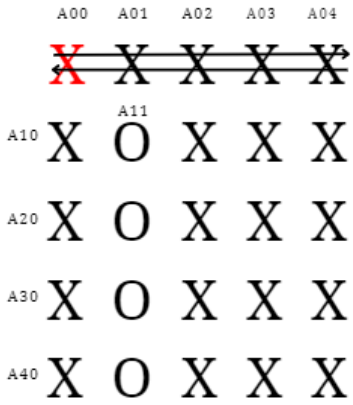
- Thuật toán duyệt ma trận hoạt động bằng cách duyệt qua từng ô trong ma trận và kiểm tra xem ô đó có phải là ô đầu tiên của một chuỗi 5 ô liên tiếp hay không. Nếu có, thì thuật toán sẽ tiếp tục duyệt qua các ô lân cận của ô đó để tìm thêm các ô cùng màu. Nếu tìm thấy đủ 5 ô cùng màu, thì thuật toán đã tìm thấy một chuỗi 5 ô liên tiếp.
- Cụ thể, trong trường hợp của Caro, bạn có thể sử dụng ma trận 2 chiều để đại diện cho bàn cờ. Mỗi ô trong ma trận sẽ đại diện cho một ô trên bàn cờ. Sau đó, bạn có thể sử dụng vòng lặp for để duyệt qua từng ô trong ma trận. Tại mỗi ô, bạn có thể kiểm tra xem ô đó có cùng màu với màu cờ mà bạn đang tìm kiếm hay không. Nếu có, thì bạn có thể tiếp tục duyệt qua các ô lân cận của ô đó.
- Giả sử chúng ta có bàn cờ Caro sau:
    - A00 là vị trí đầu tiên cho chuỗi 5 ô X liên tiếp



- Thuật toán áp dụng
    - Đầu tiên duyệt tất cả các ô trên bàn cờ

- Tạo 2 biến đếm theo 2 hướng , sao cho kết quả trả về của 2 biến thành 1 chuỗi liên tiếp có tổng >=5
- Thuật toán sẽ duyệt các ô lân cận ô A00 xem có cùng màu với A00 hay không ?
- Nếu cùng màu, khi này biến đếm sẽ tăng lên 1 đơn vị. (Dừng lại khi tổng biến đếm >=5)
- Nếu khác màu, thuật toán sẽ tiếp tục duyệt sang các ô khác trên bàn cờ.

- Ví dụ :



- Trong trường hợp trên :
  - Nếu A00 là ô đầu tiên của chuỗi 5 ô liên tiếp
  - Thuật toán sẽ duyệt qua A01 (Trái sang phải) , A10(Trên xuống dưới) , A11(Trên xuống dưới của đường chéo chính)
  - Trường hợp theo chiều ngang A01 :
    - Khi thuật toán duyệt qua A01 biến đếm được nào sẽ tăng lên 1 đơn vị; sau đó, người chơi tiếp tục đánh tiếp sang các ô A02, A03 , A04 (Cùng màu). Khi đó thực toán đã tìm được 5 ô cùng màu đồng nghĩa với việc người chơi đó dành thắng cuộc.

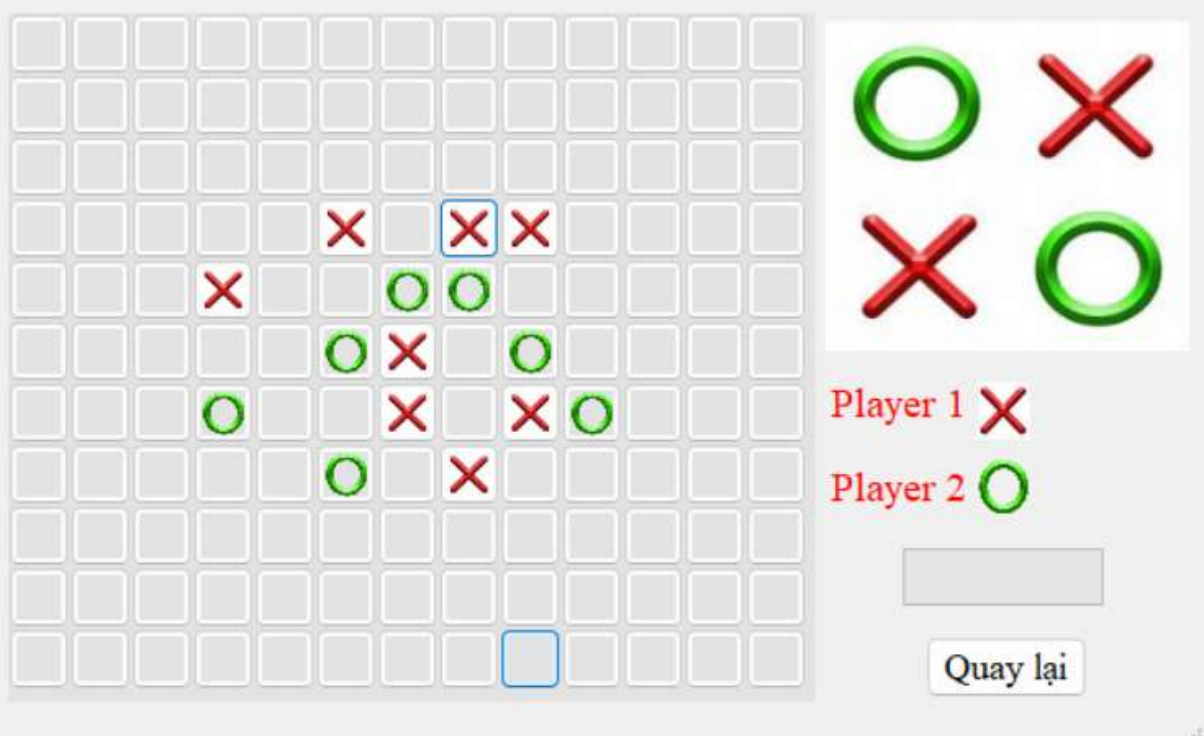
        - Tuy nhiên, nếu chỉ có 1 biến đếm thì sẽ không tối ưu cho thuật toán này.
        - Ví dụ : Người chơi chỉ đánh tại nước cờ A00 và A01, Sau đó đánh ô A04 , A03 thì biến đếm sẽ không được thực thi. Vì thế sẽ tạo thêm 1 biến từ phải sang trái để khi trong trường hợp trên tổng 2 biến đếm sẽ >=5.
- Sau đó, tương tự với các trường hợp còn lại (Kiểm tra theo chiều dọc, Kiểm tra theo đường chéo chính, Kiểm tra theo đường chéo phụ)

4.2. Chương 2: Lập trình giao diện trò chơi cờ Caro

- Tạo giao diện màn hình trò chơi

Hình 1. Màn hình chính

Hình 2. Màn hình chế độ chơi



Hình 3. Màn hình trò chơi

- Tạo các thành phần giao diện cần thiết
  - Giao diện bàn cờ
  - Một số hình ảnh quân cờ người chơi
  - Thanh thời gian đếm ngược
  - Một số nút " Làm mới, Quay lại " khi trong màn hình trò chơi

4.3. Chương 3: Lập trình logic trò chơi cờ Caro

- Lập trình vòng lặp cho trò chơi và logic kiểm tra người thắng cuộc
  - Khai báo 1 hàm lấy vị trí của phần tử trong Ma trận

```
// Hàm lấy phần tử của Ma trận M(x,y)
4 references
private Point GetChessPoint(Button btn)
{
    int doc = Convert.ToInt32(btn.Tag);
    int ngang = Matrix[doc].IndexOf(btn);
    Point point = new Point(ngang,doc);
    return point;
}
```

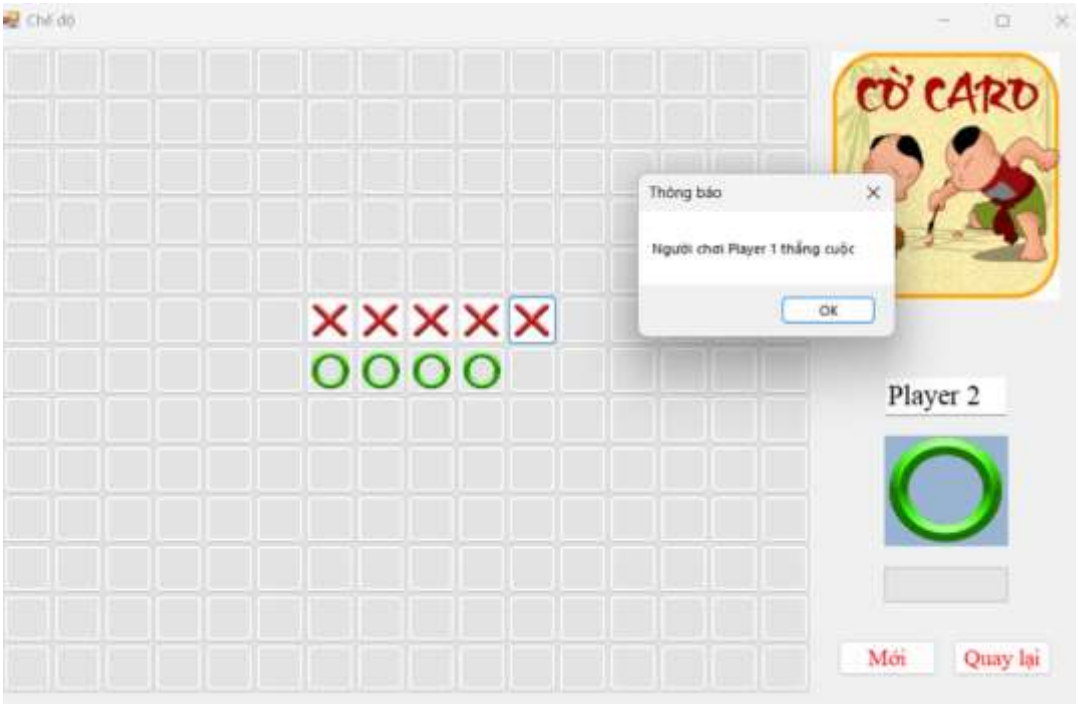
Hình 4. Hàm lấy vị trí các phần tử trong ma trận

- Đoạn mã sử dụng (Kiểm tra theo hàng ngang)

```
private bool isNgang(Button btn)
{
    Point point = GetChessPoint(btn);

    int demTrai = 0;
    int demPhai = 0;

    for (int i = point.X - 1; i >= 0; i--)
    {
        if (Matrix[point.Y][i].BackgroundImage == btn.BackgroundImage)
        {
            demTrai++;
        }
        else
            break;
    }

    for (int i = point.X + 1; i < Cons.CHESS_Board_WIDTH; i++)
    {
        if (Matrix[point.Y][i].BackgroundImage == btn.BackgroundImage)
        {
            demPhai++;
        }
        else
            break;
    }
    return demTrai + demPhai >= 4;
}
```

Hình 5. Thực hiện kiểm tra theo hàng ngang

- Kết quả



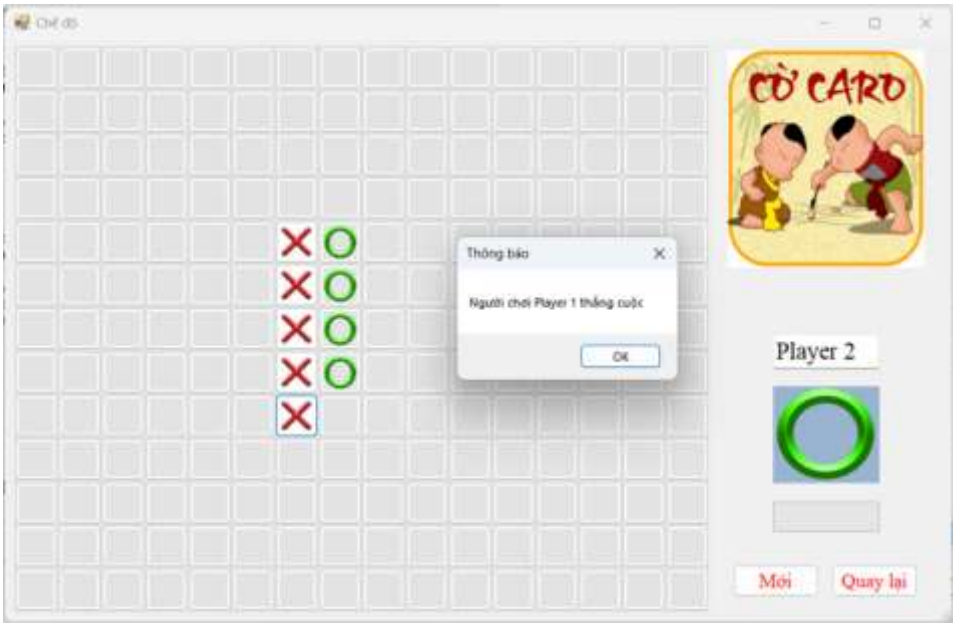
Hình 6. Kết quả thắng theo hàng ngang

- Kiểm tra theo hàng dọc

```
1 reference
private bool isDoc(Button btn)
{
    Point point = GetChessPoint(btn);

    int demTren = 0;
    int demDuoi = 0;

    for (int i = point.Y - 1; i >= 0; i--)
    {
        if (Matrix[i][point.X].BackgroundImage == btn.BackgroundImage)
        {
            demTren++;
        }
        else
            break;
    }


    for (int i = point.Y + 1; i < Cons.CHESS_Board_HEIGHT; i++)
    {
        if (Matrix[i][point.X].BackgroundImage == btn.BackgroundImage)
        {
            demDuoi++;
        }
        else
            break;
    }

    return demTren + demDuoi >= 4;
}
```

Hình 7. Kiểm tra theo hàng dọc

- Kết quả



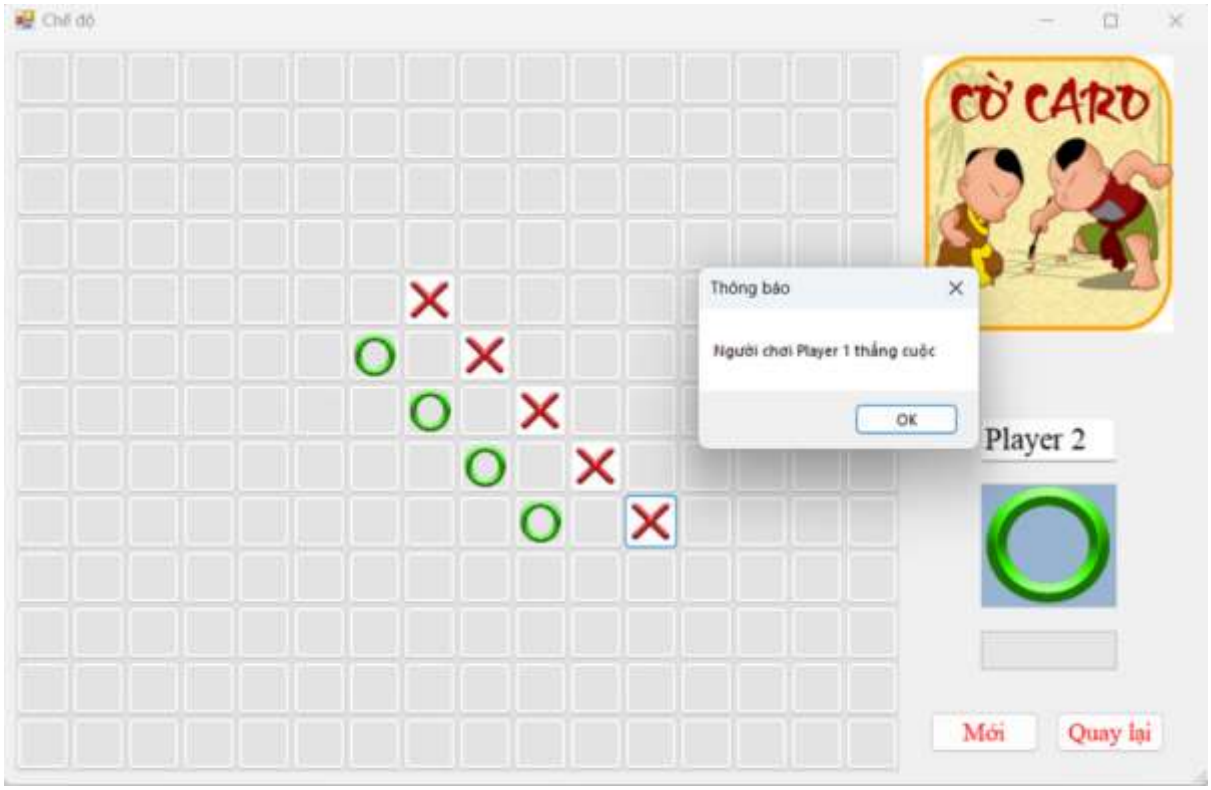
Hình 8.Kết quả thắng theo hàng dọc

- Kiểm tra theo đường chéo chính

```
1 reference
private bool isCheoChinh(Button btn)
{
    Point point = GetChessPoint(btn);

    int demCheoCT = 0;
    int demCheoCD = 0;


    for (int i = 0; i <= Math.Min(point.X, point.Y); i++)
    {
        if (Matrix[point.Y - i][point.X - i].BackgroundImage == btn.BackgroundImage)
        {
            demCheoCT++;
        }
        else
            break;
    }

    for (int i = 1; i < Math.Min(Cons.CHESS_Board_WIDTH - point.X, Cons.CHESS_Board_HEIGHT - point.Y); i++)
    {
        if (Matrix[point.Y + i][point.X + i].BackgroundImage == btn.BackgroundImage)
        {
            demCheoCD++;
        }
        else
            break;
    }

    return demCheoCT + demCheoCD >= 5;
```

Hình 9. Kiểm tra theo đường chéo chính

- Kết quả



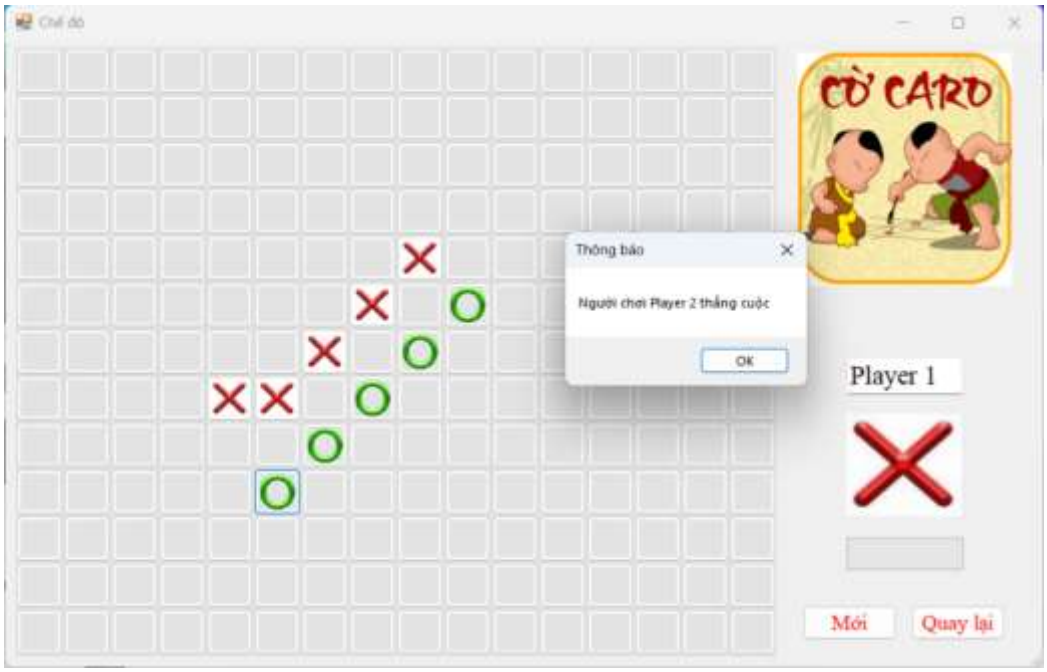
Hình 10. Kết quả thắng theo đường chéo chính

- Kiểm tra theo đường chéo phụ

```
private bool isCheoPhu(Button btn)
{
    Point point = GetChessPoint(btn);

    int demCheoPT = 0;
    int demCheoPD = 0;


    for (int i = 0; i <= Math.Min(point.X, Cons.CHESS_Board_HEIGHT - point.Y - 1); i++)
    {
        if (Matrix[point.Y + i][point.X - i].BackgroundImage == btn.BackgroundImage)
        {
            demCheoPT++;
        }
        else
            break;
    }

    for (int i = 1; i <= Math.Min(Cons.CHESS_Board_WIDTH - point.X - 1, point.Y); i++)
    {
        if (Matrix[point.Y - i][point.X + i].BackgroundImage == btn.BackgroundImage)
        {
            demCheoPD++;
        }
        else
            break;
    }

    return demCheoPD + demCheoPT >= 5 ;
}
```

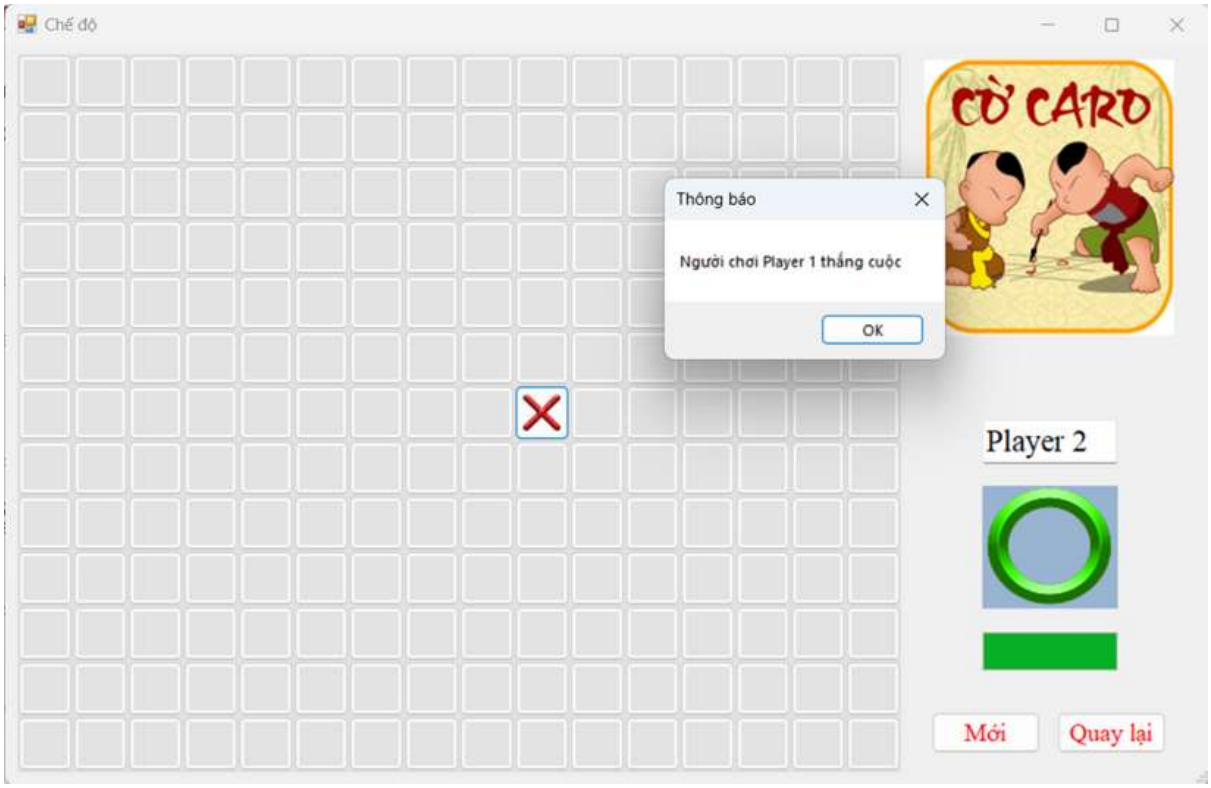
Hình 11. Kiểm tra theo đường chéo phụ

- Kết quả



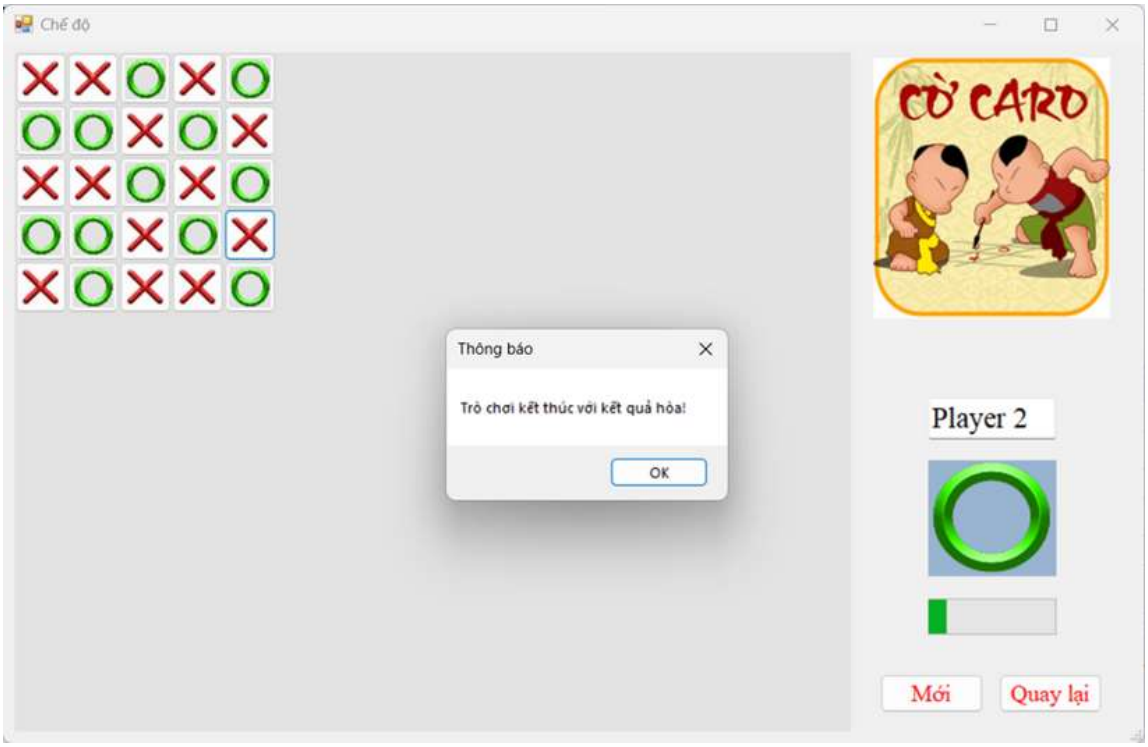
Hình 12. Thắng theo đường chéo phụ

- Kiểm tra điều kiện hết thời gian đếm ngược



Hình 13. Hết thời gian

- Kiểm tra điều kiện bàn cờ đã được đánh hết mà không tìm thấy người thắng cuộc



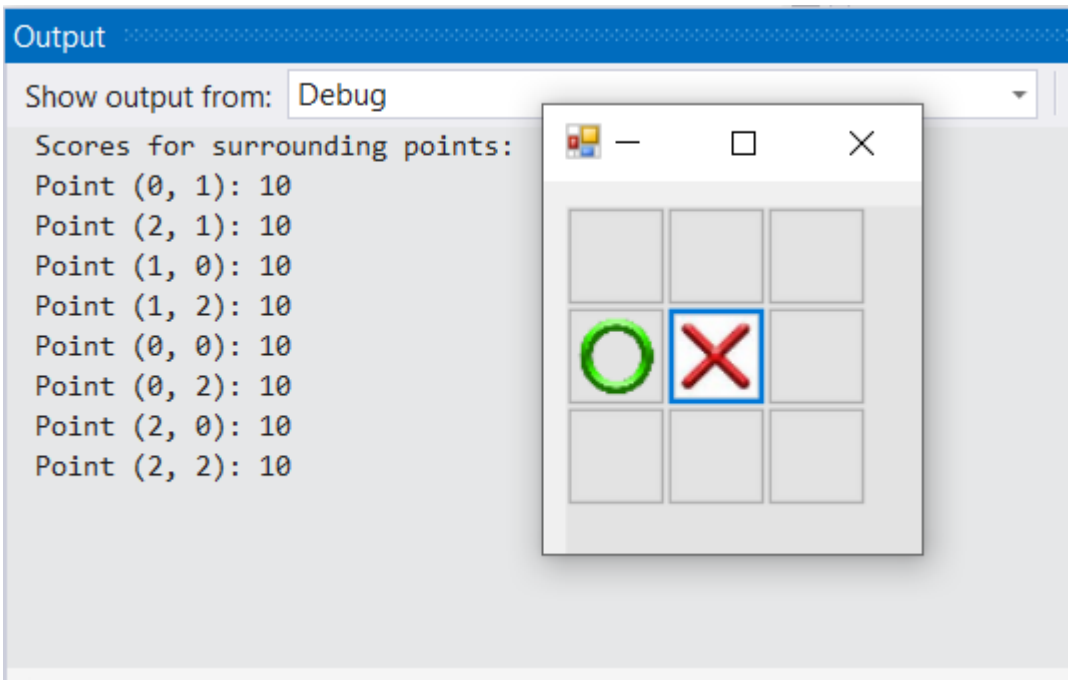
Hình 14. Hòa

4.4. Chương 4: Thêm các tính năng nâng cao cho trò chơi cờ Caro

- Thêm chế độ chơi với máy
- Ý tưởng thực hiện :
- TH1 : Trường hợp khi người chơi đánh vào 1 ô bất kỳ trên bàn cờ

1	2	3
8	X	4
7	6	5

- Máy sẽ theo số điểm cao nhất và đánh xung quanh ô mà người chơi đánh (8 vị trí như hình phía trên)
- Trong trường hợp có nhiều hơn 2 vị trí có điểm cao như nhau thì sẽ ưu tiên vị trí được tìm thấy và tính điểm đầu tiên.
- Ví dụ :



Hình 15. Ví dụ tính điểm

- **MakeEasyComputerMoveAround(Button btn):** Phương thức này được gọi khi đến lượt của máy đánh. Nó sẽ thực hiện việc đánh một nước đi của máy dựa trên một số chiến lược đơn giản.
- Các thành phần trong hàm MakeEasyComputerMoveAround:

```
// Lấy tọa độ của ô được chọn
Point selectedPoint = GetChessPoint(btn);

// Tạo danh sách các ô xung quanh không bao gồm ô đã được chọn trước đó
List<Point> surroundingPoints = new List<Point>()
{
    new Point(selectedPoint.X - 1, selectedPoint.Y), // Bên trái
    new Point(selectedPoint.X + 1, selectedPoint.Y), // Bên phải
    new Point(selectedPoint.X, selectedPoint.Y - 1), // Phía trên
    new Point(selectedPoint.X, selectedPoint.Y + 1), // Phía dưới
    new Point(selectedPoint.X - 1, selectedPoint.Y - 1), // Đường chéo trên trái
    new Point(selectedPoint.X - 1, selectedPoint.Y + 1), // Đường chéo dưới trái
    new Point(selectedPoint.X + 1, selectedPoint.Y - 1), // Đường chéo trên phải
    new Point(selectedPoint.X + 1, selectedPoint.Y + 1) // Đường chéo dưới phải
};
```

```
// Tính điểm cho mỗi ô xung quanh
Dictionary<Point, int> pointScores = new Dictionary<Point, int>();

foreach (Point point in surroundingPoints)
{
    if (point.X >= 0 && point.X < Cons.CHESS_Board_WIDTH &&
        point.Y >= 0 && point.Y < Cons.CHESS_Board_HEIGHT &&
        Matrix[point.Y][point.X].BackgroundImage == null &&
        !point.Equals(selectedPoint))
    {
        // Đánh vào ô và tính điểm
        Matrix[point.Y][point.X].BackgroundImage = playerList[CurrentPlayer].Mark;
        int score = CalculateScoreForPoint(point);
        pointScores.Add(point, score);
        // Đặt lại trạng thái ban đầu
        Matrix[point.Y][point.X].BackgroundImage = null;
    }
}
```

```
// Chọn ô có điểm cao nhất làm nước đi
if (pointScores.Count > 0)
{
    Point bestPoint = pointScores.OrderByDescending(x => x.Value).First().Key;
    Matrix[bestPoint.Y][bestPoint.X].BackgroundImage = playerList[CurrentPlayer].Mark;

    // Kiểm tra điều kiện kết thúc trò chơi sau nước đi của máy
    if (isEndGame(Matrix[bestPoint.Y][bestPoint.X]))
    {
        EndGame();
    }
}
```

Hình 16. Đoạn mã thực hiện

- **CalculateScoreForPoint(Point point):** Phương thức này tính điểm cho một ô cụ thể dựa trên các chuỗi ô cùng màu và số lượng ô trống xung quanh ô đó. Điểm số được tính dựa trên số lượng ô trống xung quanh và độ dài của các chuỗi ô cùng màu.

```
// Phương thức tính điểm cho một ô cụ thể
1 reference
private int CalculateScoreForPoint(Point point)
{
    int emptyNeighbors = 0;
    int sameColorChainScore = 0;

    // Tính điểm cho các chuỗi cùng màu theo hướng ngang, dọc, đường chéo chính và đường chéo phụ
    for (int dx = -1; dx <= 1; dx++)
    {
        for (int dy = -1; dy <= 1; dy++)
        {
            if (dx != 0 || dy != 0)
            {
                int chainLength = GetChainLength(point, new Point(dx, dy));
                if (chainLength >= 4)
                {
                    sameColorChainScore += chainLength;
                }
            }
        }
    }
```

```
// Tính điểm cho số lượng ô trống xung quanh

foreach (var delta in new[] { new Point(1, 0), new Point(-1, 0), new Point(0, 1),
                              new Point(0, -1) , new Point(-1,-1),new Point( 1, 1) })
{
    Point neighbor = new Point(point.X + delta.X, point.Y + delta.Y);
    if (IsInBoard(neighbor) && Matrix[neighbor.Y][neighbor.X].BackgroundImage == null)
    {
        emptyNeighbors++;
    }
}

// Tổng hợp điểm số từ số lượng ô trống xung quanh và chuỗi cùng màu
int totalScore = emptyNeighbors + sameColorChainScore;
return totalScore;
```

- Trong lần thực hiện này, thay vì để máy thực hiện các nước cờ có số điểm cao nhất xung quanh quân cờ của người chơi (8 vị trí như đã liệt kê ở TH1). Nhóm đã cải thiện để máy ưu tiên đánh các ô cùng màu trong phạm vi 8 ô. Trong trường hợp trong phạm vi 8 ô mà máy không tìm được ô cùng màu thì máy không đánh theo chuỗi mà đánh theo ô có điểm cao nhất

- **GetChainLength(Point point, Point delta)**: Phương thức này tính độ dài của một chuỗi các ô cùng màu, bắt đầu từ một ô cụ thể và theo một hướng delta xác định. Nó sẽ kiểm tra các ô liên tiếp cùng màu trong hướng delta cho đến khi gặp ô không cùng màu hoặc ra khỏi bàn cờ.
- **IsInBoard(Point point):** Phương thức này kiểm tra xem một điểm có nằm trong phạm vi của bàn cờ hay không.

```
private int GetChainLength(Point point, Point delta)
{
    int chainLength = 1;
    Point currentPoint = new Point(point.X + delta.X, point.Y + delta.Y);
    while (IsInBoard(currentPoint) && Matrix[currentPoint.Y][currentPoint.X].BackgroundImage
           == Matrix[point.Y][point.X].BackgroundImage)
    {
        chainLength++;
        currentPoint = new Point(currentPoint.X + delta.X, currentPoint.Y + delta.Y);
    }
    currentPoint = new Point(point.X - delta.X, point.Y - delta.Y);
    while (IsInBoard(currentPoint) && Matrix[currentPoint.Y][currentPoint.X].BackgroundImage
           == Matrix[point.Y][point.X].BackgroundImage)
    {
        chainLength++;
        currentPoint = new Point(currentPoint.X - delta.X, currentPoint.Y - delta.Y);
    }
    return chainLength;
}

private bool IsInBoard(Point point)
{
    return point.X >= 0 && point.X < Cons.CHESS_Board_WIDTH && point.Y >= 0 && point.Y < Cons.CHESS_Board_HEIGHT;
}
```

- Trong phương thức **MakeEasyComputerMoveAround**, máy tính sẽ thử đánh vào các ô xung quanh ô được chọn trước đó, tính điểm cho mỗi ô và chọn ô có điểm cao nhất làm nước đi. Nếu không có ô trống xung quanh, nó sẽ mở rộng phạm vi tìm kiếm và đánh vào ô trống gần nhất trong phạm vi mở rộng đó.
- Cuối cùng, nếu nước đi của máy tính dẫn đến kết thúc trò chơi, phương thức sẽ gọi hàm kết thúc trò chơi.
    - TH3 : Khi xung quanh phạm vi ô Player sắp đánh, đã được đánh hết

	X	X	O	
	O		X	
	X	O	X	
			O	

- Trong trường hợp trên nếu áp dụng Máy đánh xung quanh ô mà người chơi đánh thì sẽ xảy ra trường hợp trùng ô. Khi đó ô đánh của người chơi sẽ hơn máy 1 nước cờ.
- Để khắc phục, nhóm đã thực hiện khi trường hợp này xảy ra thì sẽ đánh tiếp trong phạm vi gần nhất.
- Tuy nhiên, nếu tối ưu hơn thì nên mở rộng phạm vi toàn bàn cờ. Trong đoạn code dưới đây chỉ thực hiện trong phạm vi 2 ô.
- Đoạn mã thực hiện :

```
if (pointScores.Count == 0)
{
    // Mở rộng phạm vi tìm kiếm
    for (int i = 2; i <= 3; i++) // Điều chỉnh giá trị này tùy thuộc vào mức độ mở rộng mong muốn
    {
        for (int row = selectedPoint.Y - i; row <= selectedPoint.Y + i; row++)
        {
            for (int col = selectedPoint.X - i; col <= selectedPoint.X + i; col++)
            {
                // Kiểm tra ô có nằm trong phạm vi bàn cờ không
                if (row >= 0 && row < Cons.CHESS_Board_HEIGHT && col >= 0 && col < Cons.CHESS_Board_WIDTH)
                {
                    // Kiểm tra ô có trống không
                    if (Matrix[row][col].BackgroundImage == null)
                    {
                        // Đánh vào ô trống gần nhất
                        Matrix[row][col].BackgroundImage = playerList[CurrentPlayer].Mark;

                        // Kiểm tra điều kiện kết thúc trò chơi sau nước đi của máy
                        if (isEndGame(Matrix[row][col]))
                        {
                            EndGame();
                        }
                        return;
                    }
                }
            }
        }
    }
}
```

PHỤ LỤC

TÀI LIỆU THAM KHẢO

[1] https://www.playcaro.com/news/luat-choi-co-caro/

[2] https://text.xemtailieu.net/tai-lieu/mot-so-thuat-toan-trong-tro-choi-doi-khang-va-xay-dung-game-co-caro-1217233.html

[3] [Lập trình game Caro C# Winform] - Bài 1: Demo game Caro mạng LAN | HowKteam