

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
TRÒ CHƠI CỜ CARO

Môn học: Lập trình mạng căn bản

LỚP: NT106.M21.ANTT
GVHD: TH.S TRẦN HỒNG NGHI

Bùi Tấn Hải Đăng	20520173
Phan Hoàng Tuấn	20520847
Nguyễn Thị Thùy Chinh	20521134

TP. HỒ CHÍ MINH, 06/2022

MỤC LỤC

LỜI MỞ ĐẦU	1
CHƯƠNG I.....	2
I. Sơ lược cờ Caro.....	2
II. Luật chơi của cờ Caro.....	2
CHƯƠNG II.....	3
CHƯƠNG III.....	8
1. Sơ đồ chức năng phân rã.....	8
2. Điều khiển luồng giữa các form.....	8
a. Tạo bàn cờ.....	10
c. Xử lý thắng thua.....	12
d. Đếm ngược thời gian.....	13
e. Chức năng Undo.....	14
f. Cài đặt thay đổi biểu tượng quân cờ.....	16
g. Chức năng Restart.....	17
4. Tạo kết nối LAN (Sử dụng giao thức TCP/IP)	18
5. Lập trình SOCKET.....	20
6. Truyền và xử lý dữ liệu database	26
CHƯƠNG IV: PHÂN TÍCH HỆ THỐNG QUẢN LÝ DỮ LIỆU	30
CHƯƠNG V: ÁP DỤNG TRÍ TUỆ NHÂN TẠO VÀO ĐỒ ÁN.....	31
a) Thuật toán AI sử dụng trong đồ án – cơ sở lí thuyết.....	31
b) Tổng quát các thành phần trong thuật toán.....	32
TỔNG KẾT.....	35

LỜI MỞ ĐẦU

Như chúng ta đã biết, cờ Caro là một trò chơi đã có từ rất lâu đời, hiện nay nó rất phổ biến trong cuộc sống hàng ngày nói chung, giới học đường nói riêng. Cờ Caro là một môn thể thao trí tuệ theo lối chơi chiến lược lành mạnh. Ngày nay môn cờ Caro đã được ngành thể thao coi trọng và cũng không ít các bạn trẻ tập chơi với tác dụng giải trí rất tốt cũng như tăng khả năng tư duy.

Để mọi người có thể nâng cao trình độ cờ Caro, cũng như có thể giải trí bất cứ lúc nào mà không phụ thuộc vào người khác, từ đó kéo theo nhu cầu giải trí trong mạng Lan cũng phát triển theo, cũng như xây dựng được game cờ Caro, chúng em đã áp dụng được kiến thức học trên ghế nhà trường như môn lập trình C#, trí tuệ nhân tạo, lập trình ứng dụng socket. Những kiến thức và sự hướng dẫn tận tình của cô Nghi sẽ giúp em hoàn thành đồ án cho môn học này một cách tốt nhất có thể.

CHƯƠNG I

GIỚI THIỆU GAME CỜ CARO

I. Sơ lược cờ Caro

Cờ Caro chính là môn cờ logic lâu đời và cổ xưa nhất trên Trái Đất. Cờ Caro đã được sáng tạo từ nhiều nền văn minh khác nhau một cách độc lập. Nó bắt đầu xuất hiện từ năm 2000 trước CN ở sông Hoàng Hà, Trung Quốc. Cờ Caro du nhập từ Trung Quốc vào Nhật Bản từ khoảng năm 270 trước CN. Nó thường được gọi là Gomoku nhưng cũng có các tên gọi khác tùy theo thời gian và địa phương như Kakugo, gomoku-narabe, Itsutsu-ishi... Người ta đã tìm thấy một trò chơi cổ từ một di tích ở Nhật năm 100 sau CN và thấy nó là một biến thể của Caro. Tuy nhiên, một số nhà khoa học đã tìm thấy bằng chứng chứng minh Caro đã được phát minh ở Hy Lạp cổ đại và ở Châu Mỹ trước thời Colombo.

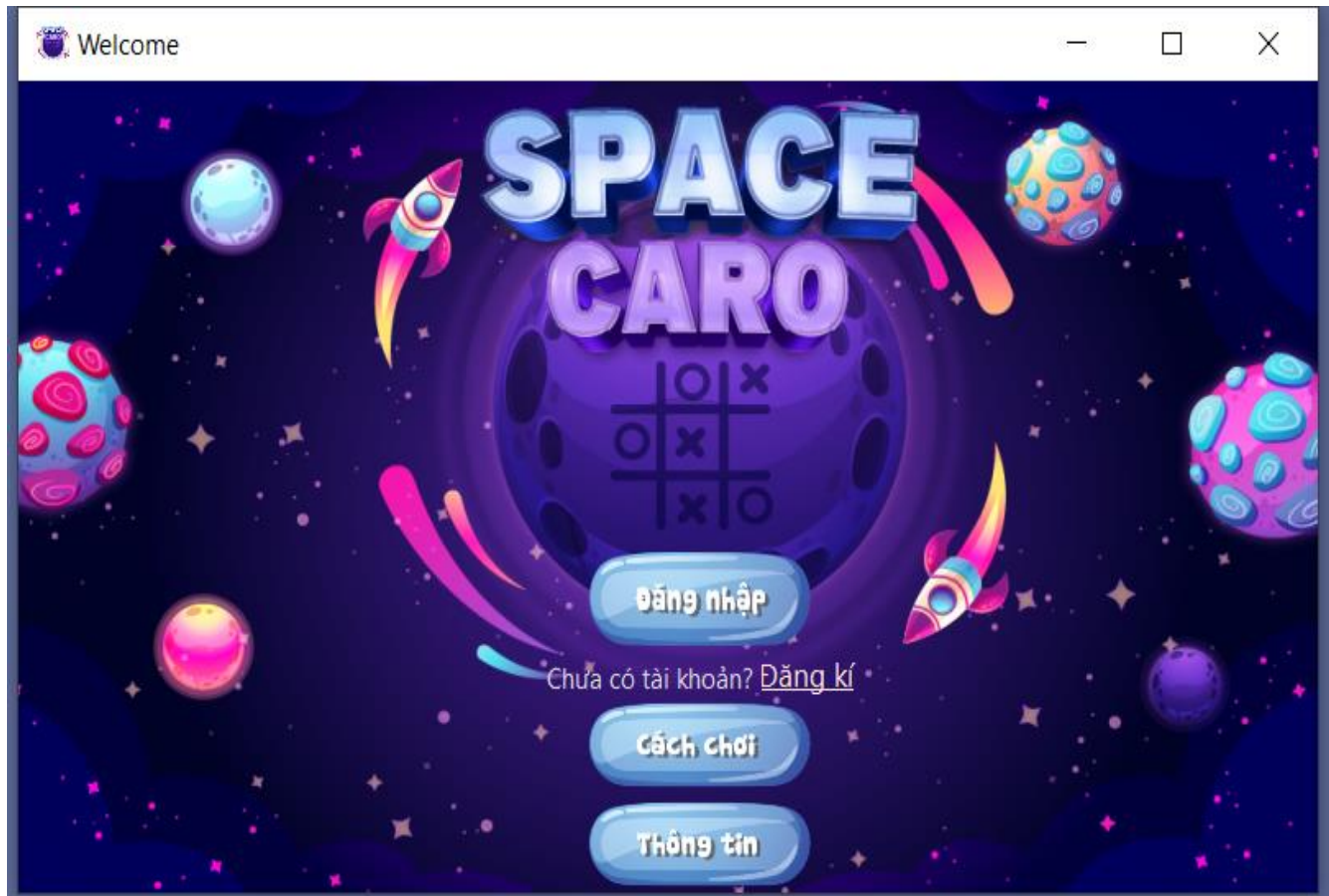
II. Luật chơi của cờ Caro

Là trò chơi đối kháng giữa 2 người, người chơi sẽ dùng kinh nghiệm và chiến thuật của mình để chiến thắng đối thủ bằng cách tạo ra 1 hàng ngang/dọc/chéo có 5 quân (với điều kiện là không bị chặn 2 đầu bởi các quân cờ của đối phương).

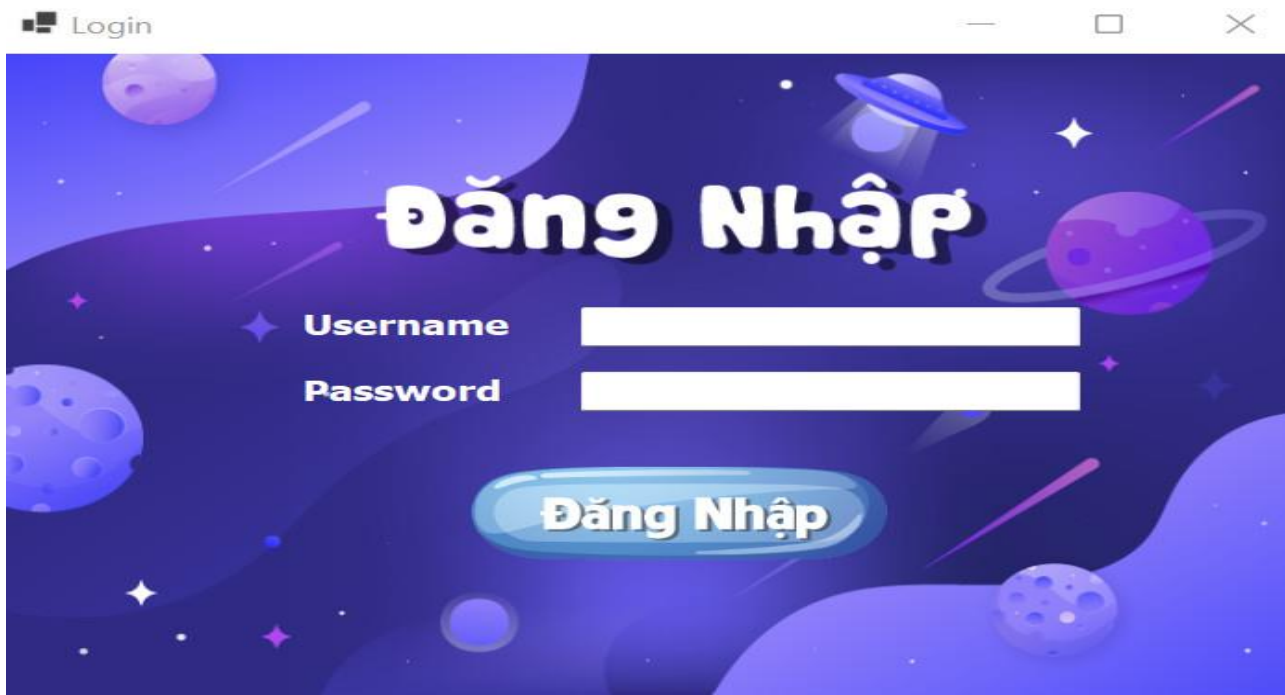
CHƯƠNG II

PHÂN TÍCH THIẾT KẾ GIAO DIỆN

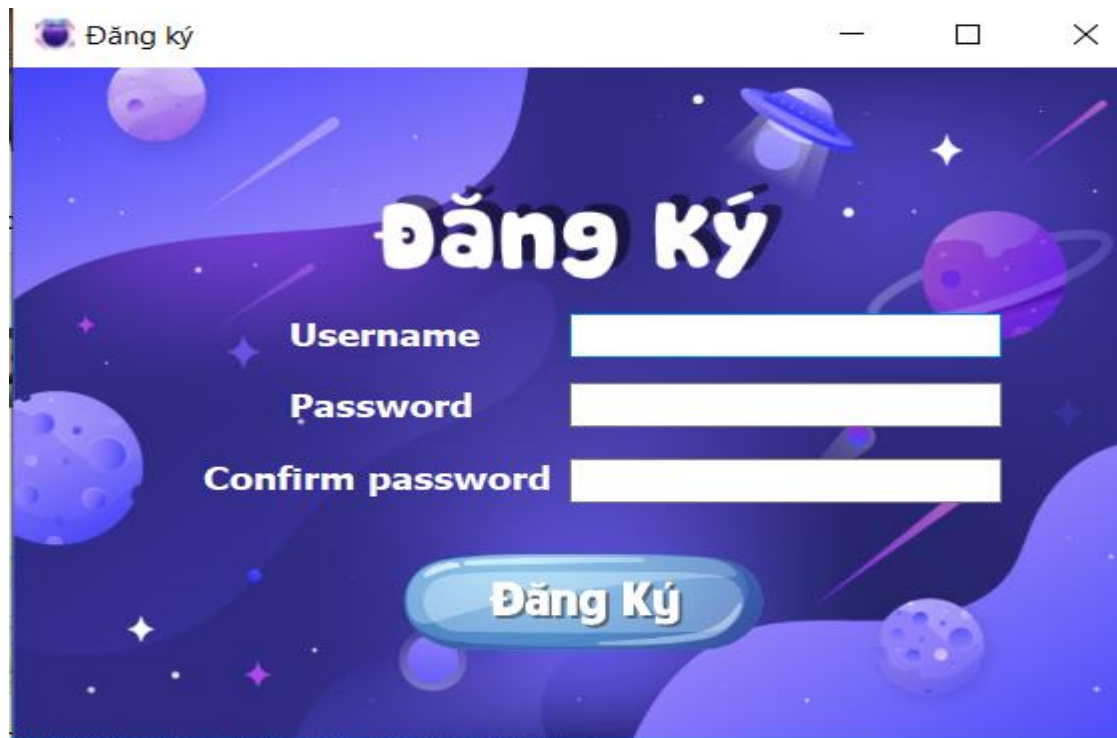
1. Cửa sổ đăng nhập chính



2. Cửa sổ Đăng nhập (có tài khoản)



3. Cửa sổ Đăng ký tài khoản



Đăng ký

Đăng Ký

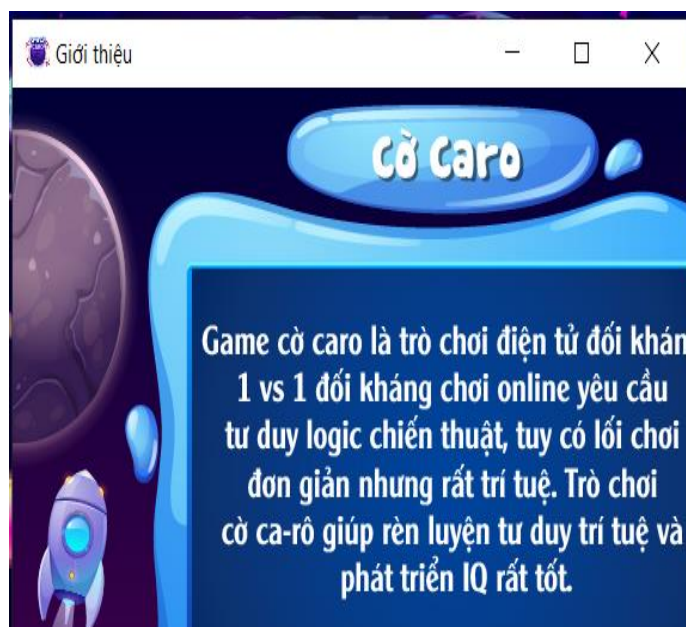
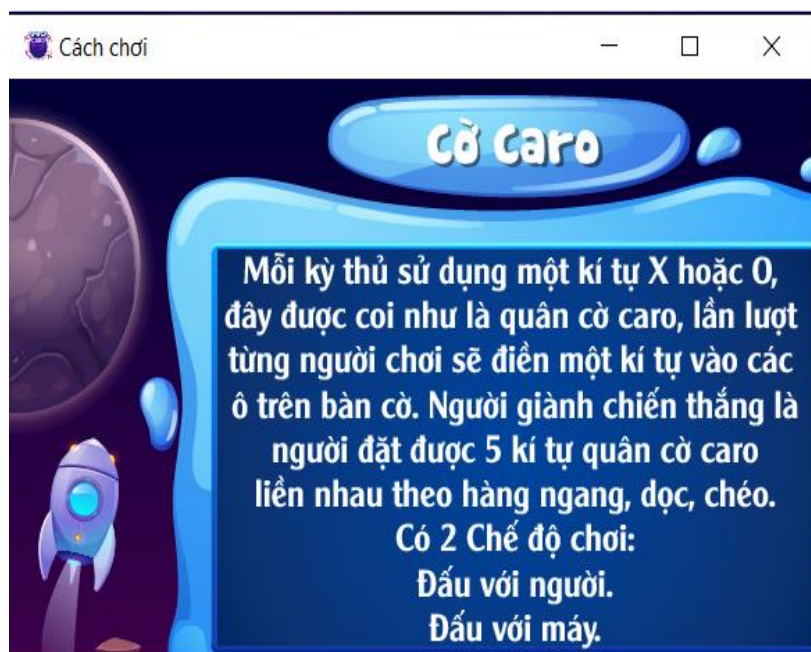
Username

Password

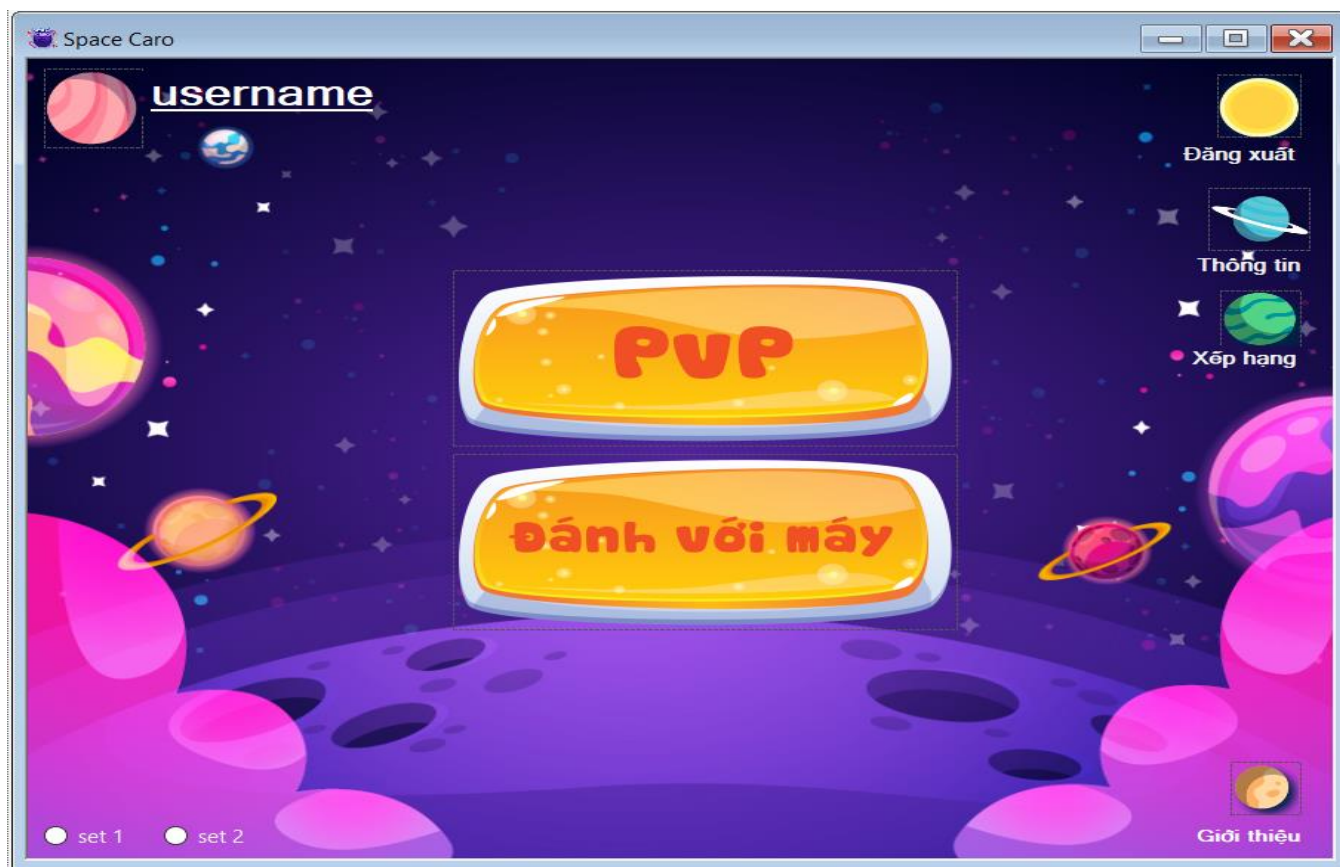
Confirm password

Đăng Ký

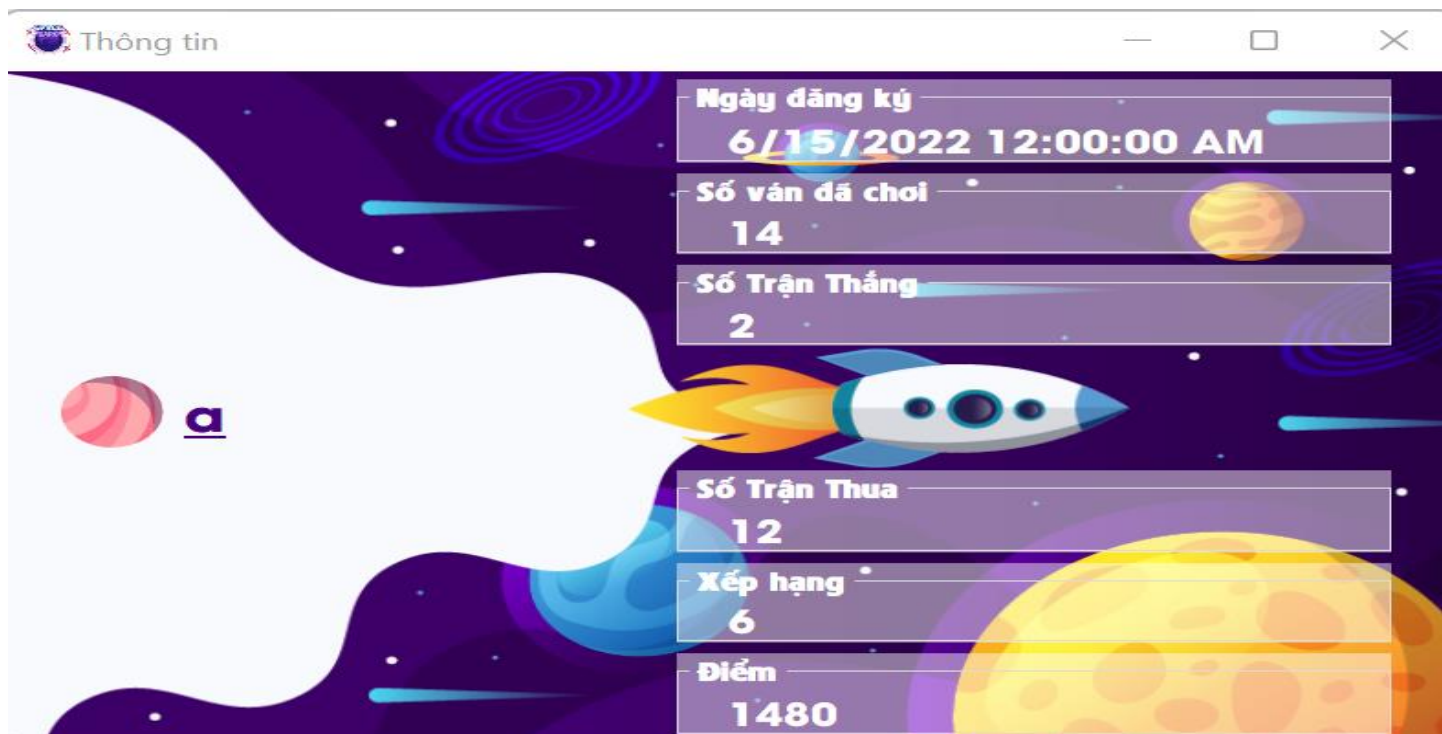
4. Cửa sổ Cách chơi & About game



5. Cửa sổ chính trò chơi



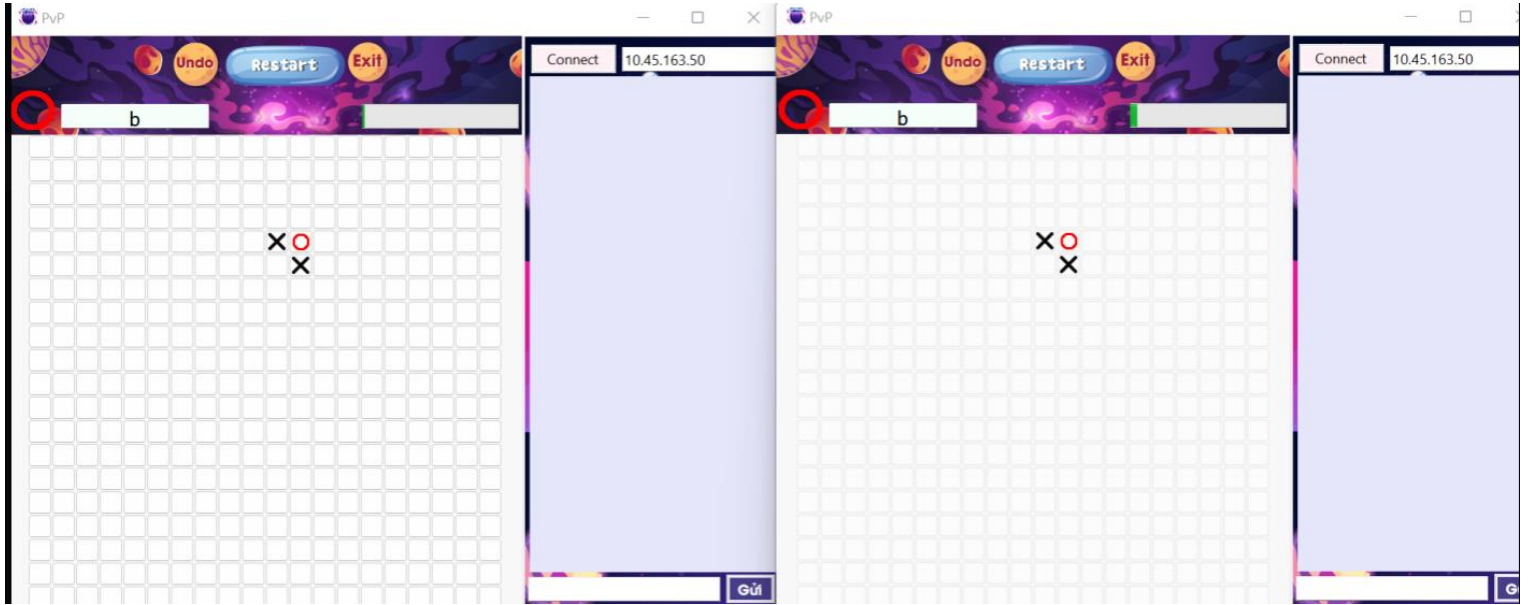
6. Cửa sổ Thông tin người chơi



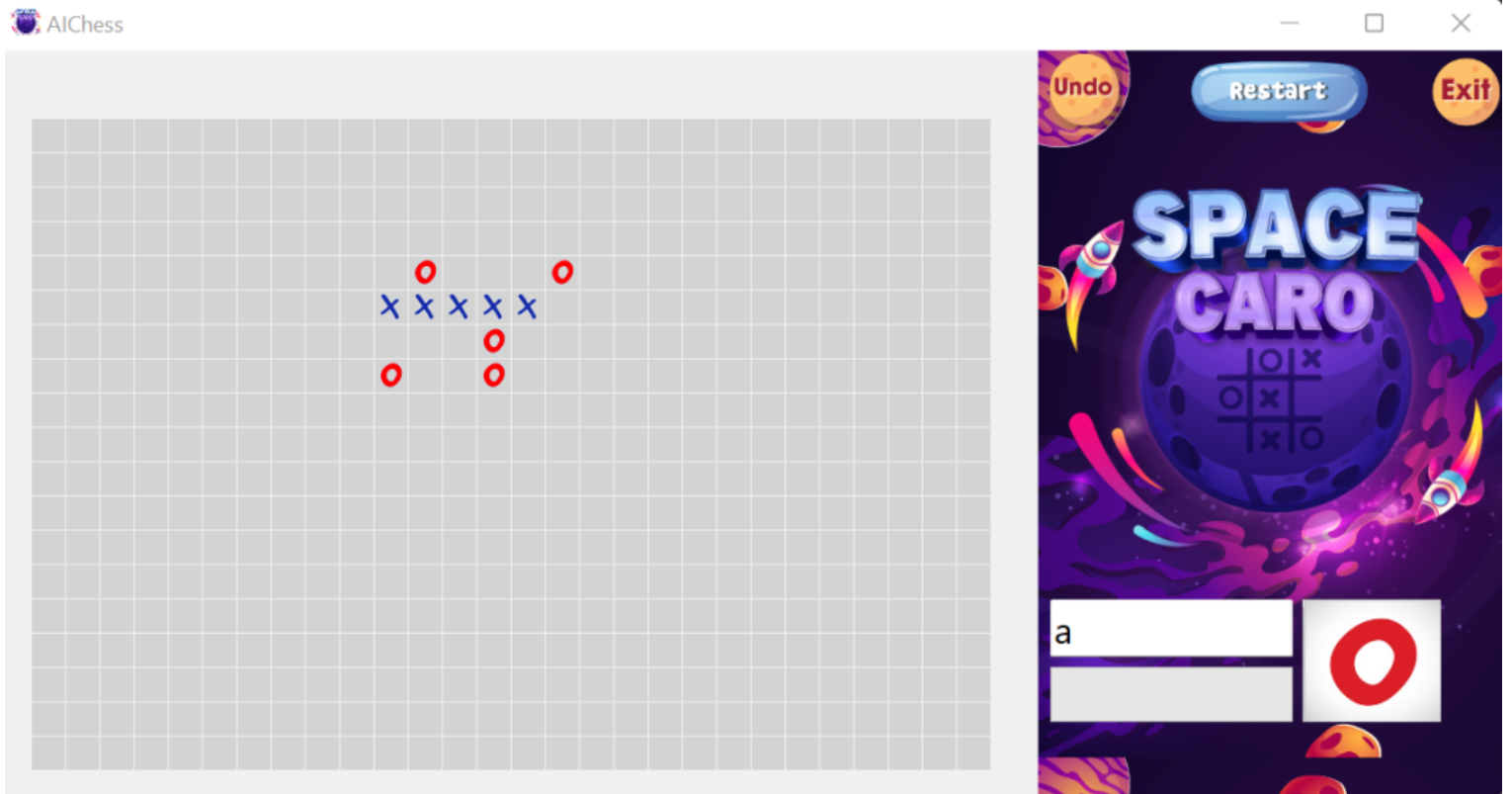
7. Cửa sổ Bảng xếp hạng



8. Cửa sổ Chơi với người (Play Now)



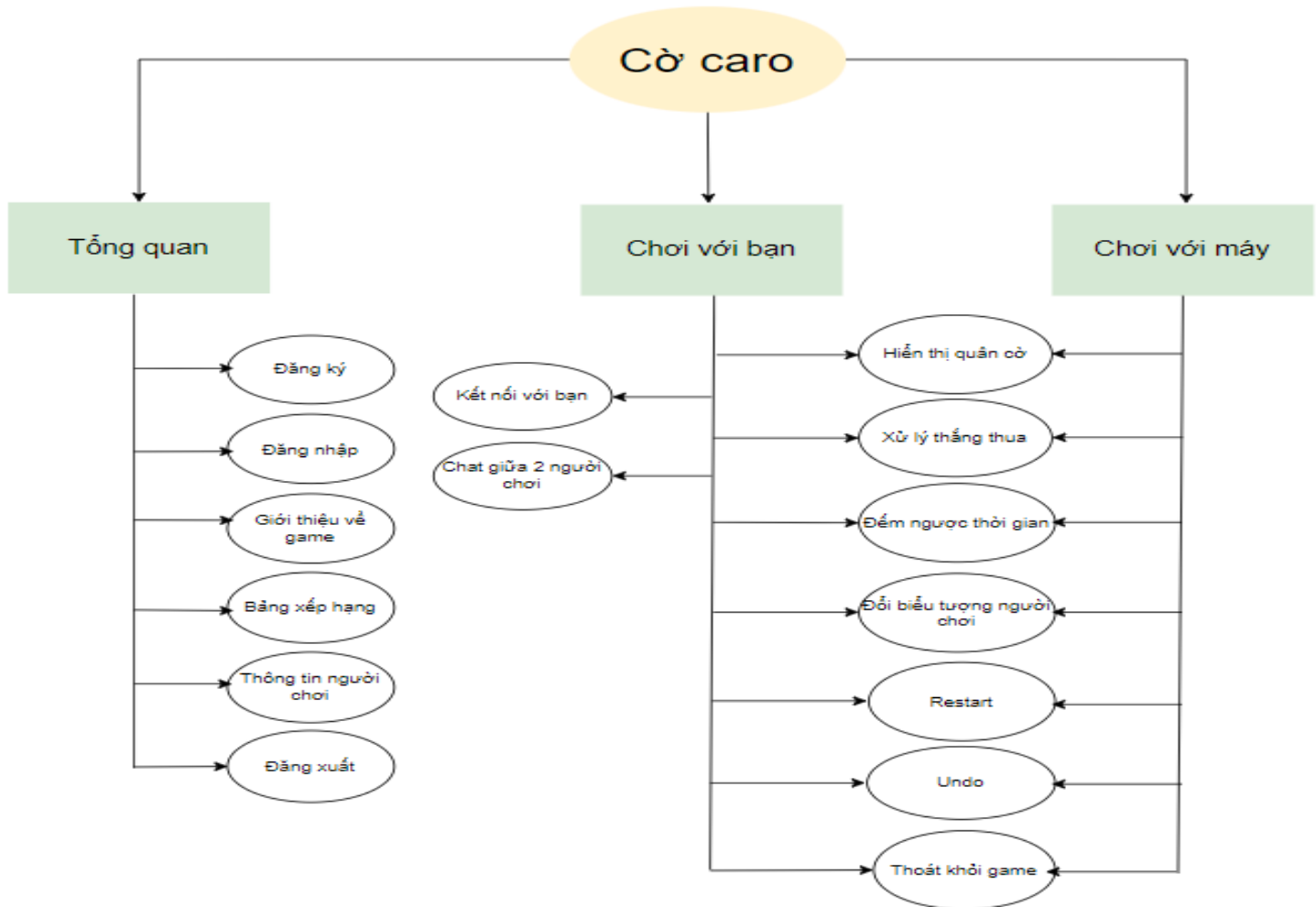
9. Cửa sổ Chơi với máy (vs AI)



CHƯƠNG III

PHÂN TÍCH TÍNH NĂNG HỆ THỐNG

1. Sơ đồ chức năng phân rã



2. Điều khiển luồng giữa các form

- Để thực hiện việc mở một form mới ngay sau khi đóng form hiện tại, ta có thể làm như sau:

```
42  [ ]  
43  [ ]  
44  [ ]  
45  [ ]  
46  [ ]  
47  [ ]  
48  [ ]  
49  [ ]  
  
#region openLogin  
1 reference  
private void button1_Click(object sender, EventArgs e)  
{  
    this.Hide();  
    var login = new Login();  
    login.Closed += (s, args) => this.Close();  
    login.Show();  
}
```

Tuy nhiên, cách làm này có thể gây ra xung đột giữa các luồng nếu trong một form hỗ trợ điều khiển nhiều form khác nhau (VD: Form Cửa Sổ Chính,...).

- Để giải quyết vấn đề trên, Multithread được sử dụng với chức năng giúp xử lý nhiều tác vụ, hỗ trợ tạo ra nhiều luồng giúp tránh xung đột như tự động mở những form không yêu cầu,...



```
32 1 reference  
33 void ReopenCSDN(object obj)  
34 {  
35     Application.Run(new Đăngnhap());  
96 1 reference  
97 private void pictureBox1_Click(object sender, EventArgs e)  
98 {  
99     this.Close();  
100     th = new Thread(ReopenCSDN);  
101     th.SetApartmentState(ApartmentState.STA);  
102     th.Start();  
    }
```

3. Một số tính năng cơ bản của game Caro (class ChessBoardManager)

a. Tạo bàn cờ

- Để có thể tạo bàn cờ như trên ta làm như sau:

```
Button oldbutton = new Button() { Width = 0, Location = new Point(0, 0) };
matrix = new List<List<Button>>(); //tạo ma trận tọa độ bàn cờ
for (int i = 0; i < Cons.DRAW_CHESS_HEIGHT; i++)
{
    matrix.Add(new List<Button>());
    for (int j = 0; j < Cons.DRAW_CHESS_WIDTH; j++)
    {
        Button bnt = new Button()
        {
            Width = Cons.CHESS_WIDTH,
            Height = Cons.CHESS_HIEGHT,
            Location = new Point(oldbutton.Location.X + Cons.CHESS_WIDTH, oldbutton.Location.Y),
            BackgroundImageLayout = ImageLayout.Stretch,
            Tag = i.ToString(), //xác định vị trí button theo hàng
        };
        bnt.Click += Bnt_Click;
        matrix[i].Add(bnt); ///
        chessboard.Controls.Add(bnt);
        oldbutton = bnt;
    }
    oldbutton = new Button() { Location = new Point(0, oldbutton.Location.Y + Cons.CHESS_HIEGHT) };
}
```

- + oldbutton: dùng để lưu tọa độ một button vừa được tạo ra, giá trị đầu tiên được khởi có tọa độ là (0;0)

- + matrix: Dùng để lưu tất cả các button vừa được tạo ra
- + Width, Height: Giá trị lưu kích thước của button(được lưu trong class Cons)
- + DRAW_CHESS_HEIGHT, DRAW_CHESS_WIDTH: kích thước của bàn cờ (được lưu trong class Cons)
- + 1 ô cờ mới được tạo ra bằng cách cộng tọa độ oldbutton với kích thước tương ứng của một button làm tọa độ.

b. Xử lý đối người chơi và hiển thị quân cờ (Chưa kết nối LAN)

Bước 1: ta phải tạo một class **player**, trong đó: **name** dùng để lưu tên người chơi, **Mark** dùng để lưu hình quân cờ của người chơi.

```
public class player
{
    private string name;

    8 references
    public string Name { get => name; set => name = value; }

    private Image mark;
    3 references
    public Image Mark { get => mark; set => mark = value; }
    4 references
}
```

Tại class ChessBoardManager, tạo List chứa các player là quân cờ X-O:

```
53 public void DrawChessBoard(Panel chessboard, TextBox playerName, PictureBox mark)
54 {
55     this.playerName = playerName;
56     this.playermark = mark;
57     this.Chessboard = chessboard;
58
59     chessboard.Controls.Clear();    ///clear bàn cờ -> làm mới
60
61     timeline = new Stack<PlayerInfo>();
62     this.players = new List<player>()
63     {
64         new player(Login.title, Image.FromFile(Application.StartupPath+"\\Resources\\i.png")),    ///X set1
65         new player("", Image.FromFile(Application.StartupPath+"\\Resources\\O.png")),    ///O set1
66         new player(Login.title, Image.FromFile(Application.StartupPath+"\\Resources\\x.png")),    ///X set2
67         new player("", Image.FromFile(Application.StartupPath+"\\Resources\\blackcircle.png")),    ///O set2
68     };
69     Currentplayer = Currentplayer == 0 ? 0 : 2;
70     changeplayer();
```

Người chơi ban đầu được mặc định là X. Sau mỗi lần button trong bàn cờ được click, sẽ hiển thị biểu tượng quân cờ của currentplayer lên bàn cờ. Và người chơi hiện tại lại chính là quân cờ kế tiếp

```

352 void changeplayer()
353 {
354     playerName.Text = players[currentplayer].Name;
355     playermark.Image = players[currentplayer].Mark;
356 }

2 references
104 private void Mark(Button bnt)
105 {
106     bnt.BackgroundImage = players[currentplayer].Mark;
107     bkgnd_btn();
108 }

```

Bước 2: Tạo hàm Bnt_Click cho mỗi khi click chuột vào các button trên bàn cờ

```

1 reference
private void Bnt_Click(object sender, EventArgs e)
{
    Button bnt = sender as Button;

    if (bnt.BackgroundImage != null)
        return;
    Chessboard.Enabled = true;
    Mark(bnt);
    Timeline.Push(new PlayerInfo(Getbuttonaxis(bnt), Currentplayer)); ///
    changeplayer();

    if (playerMark != null)
        playerMark(this, new ButtonclickEvent(Getbuttonaxis(bnt)));

    if (isEndGame(bnt))
    {
        EndGame();
    }
}

```

c. Xử lý thắng thua

Như theo luật chơi, nếu bên nào xếp được 5 quân của mình theo một phương bất kỳ (ngang, dọc, chéo) bất kỳ trước thì bên đó được xem như là thắng cuộc. Khi đó sẽ có một thông báo hiện ra cho người chơi biết kết quả đồng thời dừng ván chơi ngay tại thời điểm đó.

Duyệt bên trái và bên phải khi quân cờ đánh lưu vào một mảng và xét tại vị trí đánh đếm các phía xung quanh

```
private List<List<Button>>>matrix;
```


Mỗi button click sẽ lưu vào trong mảng, sẽ để lấy được tọa độ x hoặc o, nếu điểm click được xác định end game sẽ được nhảy vào hàm EndGame

```
matrix.Add(new List<Button>());

private bool isEndGame(Button btn)
{
    return isEndHorizontal(btn) || isEndVertical(btn) || isEndPrimaryDiagonal(btn) || isEndSubDiagonal(btn);
}
```

Xử lý và lấy tọa độ của điểm X hoặc O đã đánh

```
private Point Getbuttonaxis(Button bnt)    ///lấy giá trị tọa độ nút
{
    int doc = Convert.ToInt32(bnt.Tag);
    int ngang = matrix[doc].IndexOf(bnt);
    Point button = new Point(ngang, doc);
    return button;
}
```

Duyệt vòng lặp đếm (hàng dọc, hàng ngang, chéo trên, chéo dưới) kiểm tra đủ 5 là thắng
Kiểm tra lần lượt hàng dọc, hàng ngang, hàng chéo trên, hàng chéo dưới (chi tiết có trong phần code)

```
1 reference
private bool isEndHorizontal(Button btn)...
1 reference
private bool isEndVertical(Button btn)...
1 reference
private bool isEndPrimaryDiagonal(Button btn)...
1 reference
private bool isEndSubDiagonal(Button btn)...
#endregion
```

d. Đếm ngược thời gian

- Sử dụng ProgressBar (psbCoolDown), Timer trong c#
- Thời gian nhảy là 1s, thời gian tối đa là 20s, thời gian tăng của progressbar 1s

```
///
public static int COOL_DOWN_STEP = 100;
public static int COOL_DOWN_TIME = 20000;
public static int COOL_DOWN_INTERVAL = 100;
```

- Dùng multithread để xét 2 luồng: Đếm thời gian chơi, Đếm thời gian dừng
 - Đếm thời gian chơi

```

prcbCoolDown.Step = Cons.COOL_DOWN_STEP;
prcbCoolDown.Maximum = Cons.COOL_DOWN_TIME;
prcbCoolDown.Value = 0;
timerCoolDown.Interval = Cons.COOL_DOWN_INTERVAL;

prcbCoolDown.PerformStep();

```

- Đếm thời gian dừng

```

182 private void timerCoolDown_Tick(object sender, EventArgs e)
183 {
184     prcbCoolDown.PerformStep();
185     if (checktimeout)
186     {
187
188         if (prcbCoolDown.Value >= prcbCoolDown.Maximum)
189         {
190             EndGame();
191             MessageBox.Show("Bạn đã thua!");
192             Lose();
193             socket.Send(new SocketData((int)SocketCommand.TIME_OUT, "", new Point()));
194         }
195     }
196 }

```

- Reset progressbar = 0 khi đến lượt người chơi tiếp theo hay thực hiện NewGame

e. Chức năng Undo

Bước 1: Tạo class **PlayerInfo** lưu tọa độ quân cờ người chơi mới đánh **Point** và người chơi hiện tại **currentPlayer**

```

7 public class PlayerInfo
8 {
9     private Point point;
10     public Point Point { get => point; set => point = value; }
11
12     private int currentPlayer;
13     public int CurrentPlayer { get => currentPlayer; set => currentPlayer = value; }
14     public PlayerInfo(Point point, int currentPlayer)
15     {
16         this.point = point;
17         this.currentPlayer = currentPlayer;
18     }
19 }
20

```

Bước 2: Tại class ChessBoardManager, tạo Stack<PlayerInfo> timeline để lưu và lấy dữ liệu tọa độ dễ dàng

```
26 | public Stack<PlayerInfo> Timeline { get => timeline; set => timeline = value; } ///
```

Sau mỗi khi button được click, dữ liệu tọa độ người chơi vừa mới đánh được lưu vào stack:

```
1 reference
private void Bnt_Click(object sender, EventArgs e)
{
    Button bnt = sender as Button;

    if (bnt.BackgroundImage != null)
        return;
    Chessboard.Enabled = true;
    Mark(bnt);
    Timeline.Push(new PlayerInfo(Getbuttonaxis(bnt), Currentplayer)); ///
    changeplayer();

    if (playerMark != null)
        playerMark(this, new ButtonclickEvent(Getbuttonaxis(bnt)));

    if (isEndGame(bnt))
    {
        EndGame();
    }
}
```

Bước 3: Khởi tạo hàm Undo có kiểu dữ liệu bool. Nếu số lượng phần tử chứa tọa độ các quân cờ trong stack timeline ≤ 2 thì không thể thực hiện Undo:

```

357 public bool Undo() ////
358 {
359     if (timeline.Count <= 2)
360     {
361         return false;
362     }
363     PlayerInfo oldpos = timeline.Pop();
364     Button btn = matrix[oldpos.Point.Y][oldpos.Point.X];
365     btn.BackgroundImage = null;
366     if (timeline.Count > 1)
367     {
368         if (CuaSoChinh.k == 0)
369         {
370             Currentplayer = Currentplayer == 0 ? 0 : 1;
371         }
372         else if (CuaSoChinh.k==2)
373         {
374             Currentplayer = Currentplayer == 2 ? 2 : 3;
375         }
376     }
377     else
378     {
379         oldpos = timeline.Peek();
380         bkgnd_btn();
381         //currentplayer = currentplayer == 1 ? 0 : 1;
382     }
383     changeplayer();
384     return true;
385 }

```

f. Cài đặt thay đổi biểu tượng quân cờ

Bước 1: Lưu các hình ảnh biểu tượng quân cờ vào List player

```

this.players = new List<player>()
{
    new player(Login.title,Image.FromFile(Application.StartupPath+"\\Resources\\i.png")), ///X set1
    new player("",Image.FromFile(Application.StartupPath+"\\Resources\\0.png")), ///O set1
    new player(Login.title,Image.FromFile(Application.StartupPath+"\\Resources\\x.png")), ///X set2
    new player("",Image.FromFile(Application.StartupPath+"\\Resources\\blackcircle.png")), ///O set2
};

```

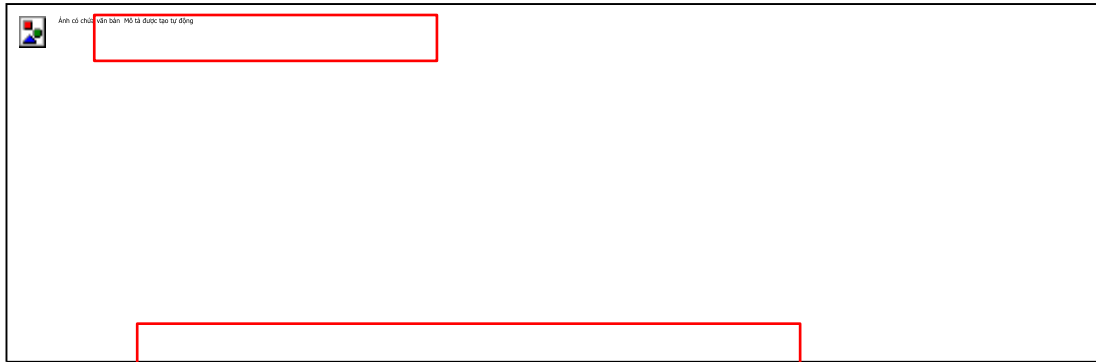
player[0] tương ứng với player[1], player[2] tương ứng [3]

```

109 void bkgnd_btn()
110 {
111     switch (currentplayer)
112     {
113         case 0: case 1:
114             currentplayer = currentplayer == 1 ? 0 : 1;
115             break;
116         case 2: case 3:
117             currentplayer = currentplayer == 3 ? 2 : 3;
118             break;
119     }
120 }

```

Ở form Play, đầu vào của hàm Play sẽ là 1 kiểu dữ liệu int k. k chính là vị trí phần tử tương ứng có trong List player. Giá trị này được nhận vào từ việc lựa chọn radiobutton ở form Cửa Sổ Chính.



(Play.cs)

```

40  -
41  -
42  -
43  -
44  -
45  -
46  -
47  -
48  -
49  -
50  -
51  -
52  -

1 reference
void Play(object obj)
{
    if (radioButton1.Checked)
    {
        k = 0;
        Application.Run(new Play(k));
    }
    else
    {
        k = 2;
        Application.Run(new Play(2));
    }
}

```

(CuaSoChinh.cs)

g. Chức năng Restart

Tại hàm DrawChessBoard bổ sung thêm câu lệnh có chức năng xóa bàn cờ

```

59  | chessboard.Controls.Clear();

```

Ở class của form Play.cs, tạo hàm NEWGAME. Sau khi hàm này được gọi thực hiện, bàn cờ sẽ được xóa và vẽ lại, mọi chức năng sẽ được reset.

```

49 void NewGame()
50 {
51     prcbCoolDown.Value = 0;
52     timerCoolDown.Stop();
53     ChessBoard.Currentplayer = CuaSoChinh.k;
54     ChessBoard.DrawChessBoard(panel1, textBox2, pictureBox1);
55     panel1.Enabled = true;
56     pictureBox2.Enabled = true;
57     socket.Send(new SocketData((int)SocketCommand.SENDNAME, Login.title, new Point()));
58 }

```

4. Tạo kết nối LAN (Sử dụng giao thức TCP/IP)

Bước 1: Tạo class **SocketManager** dùng để quản lý Socket

- Class gồm 2 đối tượng là **client** và **server**.
- Đối với server: giao thức chính là **CreateServer** dùng để khởi tạo server và lắng nghe từ client

```

Socket server;
1 reference
public void CreateServer()
{
    IPEndPoint iep = new IPEndPoint(IPAddress.Parse(IP), PORT);
    server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

    server.Bind(iep);
    server.Listen(10);

    Thread acceptClient = new Thread(() =>
    {
        client = server.Accept();
    });
    acceptClient.IsBackground = true;
    acceptClient.Start();
}

```

- Đối với client: Giao thức **ConnectServer**, trả về true nếu kết nối thành công ngược lại trả về false

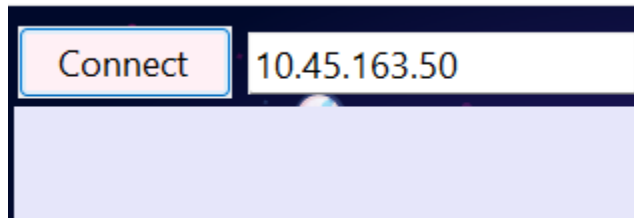
```

1 reference
public bool ConnectServer()
{
    IPEndPoint iep = new IPEndPoint(IPAddress.Parse(IP), PORT);
    client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

    try
    {
        client.Connect(iep);
        return true;
    }
    catch
    {
        return false;
    }
}
#endregion

```


Bước 2: Tạo button LAN dùng để kết nối TCP/IP ở form Play.cs



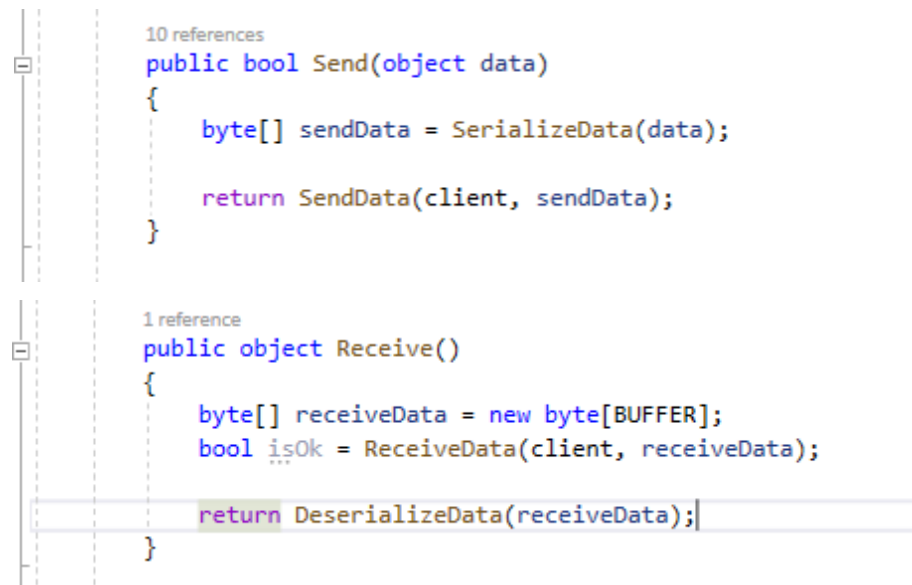
- Nếu chưa có server tức ConnectServer = false \Rightarrow CreateServer()
- Nếu đã có Server tức ConnectServer = true \Rightarrow Listen() */// lắng nghe server*

```
242 void Listen()  
243 {  
244  
245     Thread listenThread = new Thread(() =>  
246     {  
247         try  
248         {  
249             SocketData data = (SocketData)socket.Receive();  
250             ProcessData(data);  
251         }  
252         catch { }  
253     });  
254     listenThread.IsBackground = true;  
255     listenThread.Start();  
256 }  
257
```

5. Lập trình SOCKET

Để có thể truyền quân cờ, tin nhắn, các tín hiệu EndGame, Undo, Win, Quit, ... thực hiện các bước như sau:

Bước 1: Đối với class SocketManager ta viết các hàm gửi và nhận dùng chung cho cả client và server



Các hàm SerializeData và DeserializeData dùng để nén đối tượng trước khi gửi và giải nén sau khi nhận được.

Bước 2: Tạo class **SocketData** dùng làm đối tượng để gửi và nhận

- Command: Dùng để phân tích trường hợp cho dữ liệu
- Point: Chứa tọa độ ô cờ
- Message: chứa thông tin muốn gửi

```
class SocketData
{
    private int command;
    private Point point;
    private String message;

    2 references
    public int Command { get => command; set => command = value; }
    2 references
    public Point Point { get => point; set => point = value; }
    7 references
    public string Message { get => message; set => message = value; }

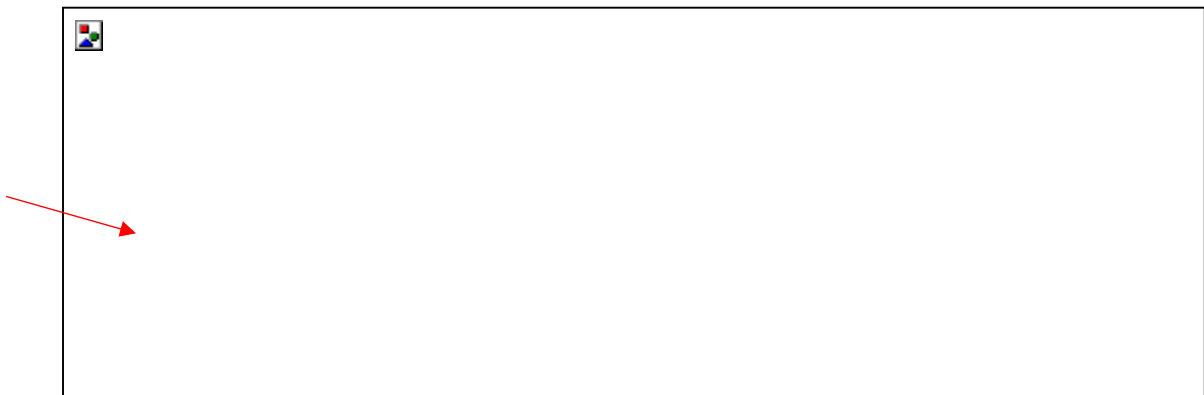
    10 references
    public SocketData(int command, string message, Point point)
    {
        this.Command = command;
        this.Message = message;
        this.Point = point;
    }
}
```

Trong SocketData còn có **SocketCommand** dùng để phân loại các dữ liệu:

```
25 public enum SocketCommand
26 {
27     SEND_POINT,
28     NOTIFY,
29     NEWGAME,
30     ENDGAME,
31     TIME_OUT,
32     SEND_MESS,
33     UNDO,
34     SENDNAME,
35     QUIT,
36     YES,
37     NO
38 }
```

a. Truyền quân cờ

- Sau khi đánh quân cờ: người chơi sẽ gửi một tín hiệu SEND_POINT cho đối thủ



- Đối thủ nhận tín hiệu bằng hàm Listen() và phân tích tín hiệu bằng hàm ProcessData(). Hàm Processdata(): Phân tích tín hiệu là SEND_POINT thực hiện đánh vào ô cờ tương ứng data.point, đối với trường hợp này message không được sử dụng

```
private void ProcessData(SocketData data)
{
    switch(data.Command)
    {
        case (int)SocketCommand.SEND_POINT:
            this.Invoke((MethodInvoker)(() =>
            {
                //checkendgame = true;
                prcbCoolDown.Value = 0;
                panel1.Enabled = true;
                timerCoolDown.Start();
                pictureBox2.Enabled = true;
            }));
            ChessBoard.OtherplayerMark(data.Point);
    }
}
```

b. Truyền tên người chơi

Khi Client kết nối đến server thành công gửi một tín hiệu SENDNAME kèm theo tên tài khoản của mình (Login.title)



Khi nhận được tín hiệu SENDNAME: Nếu người nhận là server: thay thế tên của player thứ hai, Nếu người nhận là client: thay thế tên của player thứ nhất

```
break;
case (int)SocketCommand.SENDNAME:
    if(socket.isServer == true)
    {
        ChessBoard.Players[1].Name = data.Message;
        ChessBoard.Players[3].Name = data.Message;
        socket.Send(new SocketData((int)SocketCommand.SENDNAME, Login.title, new Point()));
    }
    else
    {
        ChessBoard.Players[0].Name = data.Message;
        ChessBoard.Players[1].Name = Login.title;
        ChessBoard.Players[2].Name = data.Message;
        ChessBoard.Players[3].Name = Login.title;
        this.Invoke((MethodInvoker)() =>
        {
            ChessBoard.PlayerName.Text = Login.title;
        }));
    }
    break;
default:
    break;
```

c. Xử lý NewGame

Khi một trong hai người chơi nhấn RESTART:



```
private void pictureBox4_Click(object sender, EventArgs e)
{
    NewGame();
    socket.Send(new SocketData((int)SocketCommand.NEWGAME, "", new Point()));
    panel1.Enabled = true;
}
```

Gửi tín hiệu NEW_GAME cho đối thủ

```
case (int)SocketCommand.NEWGAME:
    this.Invoke((MethodInvoker) (() =>
    {
        NewGame();
        panel1.Enabled = false;
    }));
    break;
```

Đối thủ nhận được tín hiệu và gọi hàm NewGame().

d. Xử lý QuitGame

Khi người chơi nhấn vào button Exit trên giao diện



thì sẽ gửi tín hiệu Thoát cho đối thủ, đồng thời bị xử thua



Khi nhận được tín hiệu QUIT và thông báo người chơi bên kia đã thoát, người bên đây (nhận được tín hiệu QUIT) sẽ được xét là thắng ván game này:

```

case (int)SocketCommand.QUIT:|
if (quitcheck == true)
{
    this.Invoke((MethodInvoker)(() =>
    {
        timerCoolDown.Stop();
        EndGame();
        MessageBox.Show("Đối thủ đã chạy mất dép", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);
        socket.close();
        Win();
    }));
}
socket.close();
break;

```

e. Xử lý Undo

Một trong hai người chơi xin được phép lùi bước



Người chơi đó sẽ gửi một tín hiệu cho đối phương

```

private void pictureBox2_Click(object sender, EventArgs e) ///Undo
{
    socket.Send(new SocketData((int)SocketCommand.UNDO, "", new Point()));
}

```

Khi nhận được tín hiệu Undo gọi hàm request()

```

299 | case (int)SocketCommand.UNDO:
300 |     request();
301 |     break;

```

Hàm request() cho phép đối phương có quyền xác nhận người chơi (click Undo) được phép Undo hay không (Yes or No)

```

void request()
{
    DialogResult Q = MessageBox.Show("Đối thủ muốn Undo?", "Request", MessageBoxButtons.YesNo);
    if(Q==DialogResult.Yes)
    {
        socket.Send(new SocketData((int)SocketCommand.YES, "", new Point()));
        undo();
    }
}

```


Nếu nhận được tín hiệu YES và thông báo “đối thủ đồng ý” thì hàm Undo() được gọi thực hiện; nếu nhận được tín hiệu NO và thông báo “đối thủ không đồng ý” thì quá trình Undo không được thực hiện.

```

302         case (int)SocketCommand.YES:
303             MessageBox.Show("Đối thủ đồng ý.");
304             this.Invoke((MethodInvoker)(() =>
305             {
306                 undo();
307             }));
308             break;
309         case (int)SocketCommand.NO:
310             MessageBox.Show("Đối thủ không đồng ý.");
311             break;

```

f. Xử lý hết thời gian

Khi một trong hai bên hết thời gian trước khi đánh quân cờ của mình

Bên đó sẽ bị xử thua và gửi tín hiệu TIME_OUT cho đối phương:

```

private void timerCoolDown_Tick(object sender, EventArgs e)
{
    prcbCoolDown.PerformStep();
    if (checktimeout)
    {
        if (prcbCoolDown.Value >= prcbCoolDown.Maximum)
        {
            EndGame();
            MessageBox.Show("Bạn đã thua!");
            Lose();
            socket.Send(new SocketData((int)SocketCommand.TIME_OUT, "", new Point()));
        }
    }
}

```

Khi bên còn lại nhận được tín hiệu TIME_OUT, sẽ được xử thắng trận đấu

```

        case (int)SocketCommand.TIME_OUT:
            MessageBox.Show("Bạn đã thắng!");
            Win();
            timerCoolDown.Stop();
            break;

```

g. Xử lý gửi tin nhắn Chat

Khi người chơi điền tin nhắn và nhấn gửi:



Nội dung tin nhắn sẽ được gửi đi kèm theo tên tài khoản của người gửi và gửi tín hiệu SEND_MESS



Khi nhận được tín hiệu SEND_MESS: tin nhắn nhận được dưới dạng data.Message sẽ được thêm vào listview

```
case (int)SocketCommand.SEND_MESS:
    this.Invoke((MethodInvoker)() =>
    {
        listView1.Items.Add(new ListViewItem(data.Message));
    });
    break;
```

6. Truyền và xử lý dữ liệu database

a. Truyền dữ liệu form Đăng ký

Thực hiện việc so sánh dữ liệu để kiểm tra tên tài khoản đã tồn tại hay chưa và lưu vào database

```
//Lưu dữ liệu
string yourSQL = "SELECT TENTK FROM [GAME_CARO].[dbo].[INFO] WHERE TENTK = '" + textBox1.Text + "'";
DataTable checkDuplicates = Tạo_Bàn_Cờ.Connection.SeverConnection.executeSQL(yourSQL);
if(checkDuplicates.Rows.Count>0)
{
    MessageBox.Show("The username already exist. Please try another");
    textBox1.SelectAll();
    return;
}
DialogResult result;
result = MessageBox.Show("Do you want to save this Login.");
if(result==DialogResult.OK)
{
    date = DateTime.Now;

    string mySQL = string.Empty;
    mySQL += "INSERT INTO [GAME_CARO].[dbo].[INFO] (TENTK,PW,NGDK,TONG,WIN,LOSE,SCORE)";
    // mySQL += "VALUES ('" + textBox1.Text + "','" + textBox2.Text + "'" + ",0,0,0,1600)";
    mySQL += "VALUES ('" + textBox1.Text + "','" + textBox2.Text + "','" + date + "','" + ",0,0,0,1600)";
    Tạo_Bàn_Cờ.Connection.SeverConnection.executeSQL(mySQL);
    MessageBox.Show("Succeeded!");
```

b. Truyền dữ liệu form Đăng nhập

Kiểm tra dữ liệu trên textbox có trùng với dữ liệu trong database không, nếu trùng thì mới được quyền truy cập tiếp

```
private void button1_Click(object sender, EventArgs e)
{
    if(!string.IsNullOrEmpty(textBox1.Text)&&!string.IsNullOrEmpty(textBox2.Text))
    {
        string mySQL = string.Empty;

        mySQL += "SELECT *FROM [GAME_CARO].[dbo].[INFO]";
        mySQL += "WHERE TENTK = '" + textBox1.Text + "' ";
        mySQL += "AND PW = '" + textBox2.Text + "'";
        DataTable userData = SeverConnection.executeSQL(mySQL);
        if(userData.Rows.Count>0)
        {
            title = textBox1.Text;
            MyInfo.username = title;
            textBox1.Clear();
            textBox2.Clear();
            close();
            this.textBox1.Select();
        }
        else
        {
            MessageBox.Show("The username or password is incorrect. Try again");
            textBox2.Focus();
            textBox2.SelectAll();
        }
    }
    else
    {
        MessageBox.Show("Please enter username and password.");
        textBox2.Select();
    }
}
```

c. Truyền dữ liệu vào form My Info, Ranking

- Khi nhấn vào biểu tượng My Info

Hàm GetInfo() sẽ được gọi sau khi Loadform. Dữ liệu sẽ được truy xuất dựa theo username của người chơi được lưu trong database

```
void GetInfo()
{
    using (SqlConnection conn = new SqlConnection(constr))
    {
        string query2 = "select * from (select TENTK,SCORE,ROW_NUMBER() over(order by SCORE DESC) as RANK_ from INFO)" +
            " as foo where TENTK = '"+username+"'";
        SqlDataAdapter adapter2 = new SqlDataAdapter(query2, conn);
        DataSet data2 = new DataSet();
        adapter2.Fill(data2);
        Rank.Text = data2.Tables[0].Rows[0]["RANK_"].ToString();
        conn.Open();
        SqlCommand cmd = new SqlCommand(query1, conn);
        using (SqlDataReader reader = cmd.ExecuteReader())
        {
            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    NGDK.Text = reader[2].ToString();
                    SVDC.Text = reader[3].ToString();
                    Win.Text = reader[4].ToString();
                    Lose.Text = reader[5].ToString();
                    DIEM.Text = reader[6].ToString();
                }
            }
        }
        conn.Close();
    }
}
```

- Thực hiện tương tự với Rank

```
1 void getRank()
1 {
    using (SqlConnection conn = new SqlConnection(constr))
    {
        conn.Open();
        DataSet data = new DataSet();
        SqlDataAdapter adapter = new SqlDataAdapter(query1, conn);
        adapter.Fill(data);
        SqlCommand comm = new SqlCommand("SELECT COUNT(*) FROM INFO", conn);
        Int32 count = (Int32)comm.ExecuteScalar();
1 for (int i = 0; i < count; i++)
    { listBox1.Items.Add(" " + (i+1) + ". " + data.Tables[0].Rows[i]["TENTK"].ToString() + "\t\t" +
        data.Tables[0].Rows[i]["SCORE"].ToString() + "\n");}

    conn.Close();
    }
}
```

d. Cập nhật dữ liệu thắng, thua

- Khi người chơi thắng dữ liệu sẽ thay đổi: Số trận thắng +1, Tổng số trận +1, **Điểm +12**
- Khi người chơi thua dữ liệu sẽ thay đổi: Số trận thua +1, Tổng số trận +1, **Điểm -12**

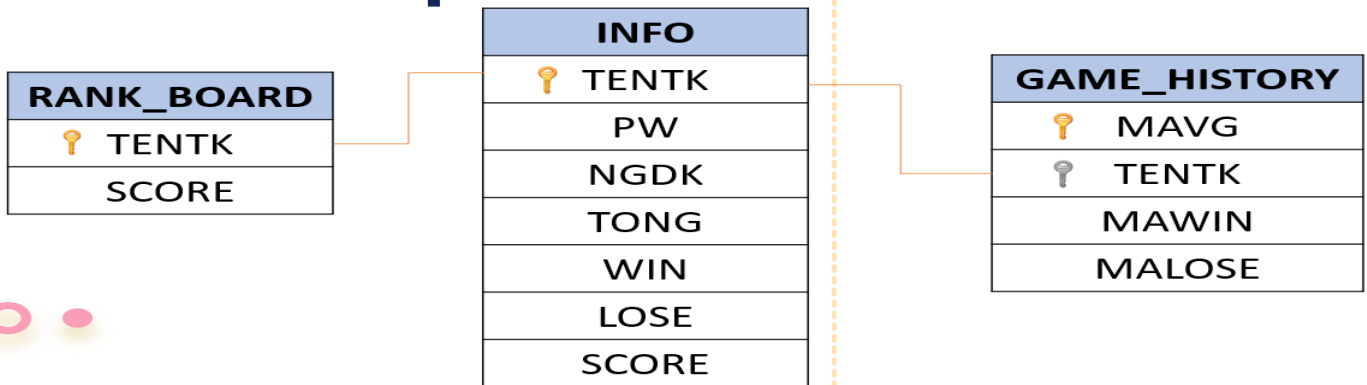
```
373 void Win()
374 {
375     quitcheck = false;
376     string query = "update INFO set TONG+=1,WIN +=1,SCORE+=12 where TENTK = '"+Login.title+"' ";
377     using (SqlConnection conn = new SqlConnection(Connection.ServerConnection.stringConnection))
378     {
379         conn.Open();
380         SqlCommand cmd2 = new SqlCommand(query, conn);
381         using (SqlDataReader dataReader = cmd2.ExecuteReader())
382         {
383         }
384         conn.Close();
385     }
386 }
387 void Lose()
388 {
389     quitcheck = false;
390     string query = "update INFO set TONG+=1,LOSE +=1,SCORE-=12 where TENTK = '"+Login.title+"' ";
391     using (SqlConnection conn = new SqlConnection(Connection.ServerConnection.stringConnection))
392     {
393         conn.Open();
394         SqlCommand cmd2 = new SqlCommand(query, conn);
395         using (SqlDataReader dataReader = cmd2.ExecuteReader())
396         {
397         }
398         conn.Close();
399     }
400 }
401 }
402
403
```

CHƯƠNG IV: PHÂN TÍCH HỆ THỐNG QUẢN LÝ DỮ LIỆU

Game Caro chỉ sử dụng một database để quản lý các thông tin tài khoản người chơi, các thông tin bao gồm:

- Tên tài khoản (TENTK)
- Mật khẩu tài khoản (PW)
- Ngày đăng ký tài khoản (NGDK)
- Tổng số ván game đã chơi (TONG)
- Tổng số trận đã thắng (WIN)
- Tổng số trận đã thua (LOSE)
- Tổng số điểm (SCORE)

Mô tả cơ sở dữ liệu



```
/****** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [TENTK]
      [PW]
      [NGDK]
      [TONG]
      [WIN]
      [LOSE]
      [SCORE]
FROM [GAME_CARO].[dbo].[INFO]
```

	TENTK	PW	NGDK	TONG	WIN	LOSE	SCORE
1	a	a	2022-06-15	15	2	13	1468
2	aa	1	2022-06-16	6	5	1	1648
3	b	b	2022-06-15	7	2	5	1564
4	c	c	2022-06-17	1	1	0	1612
5	tuan	1	2022-06-18	0	0	0	1600
6	w	w	2022-06-18	0	0	0	1600

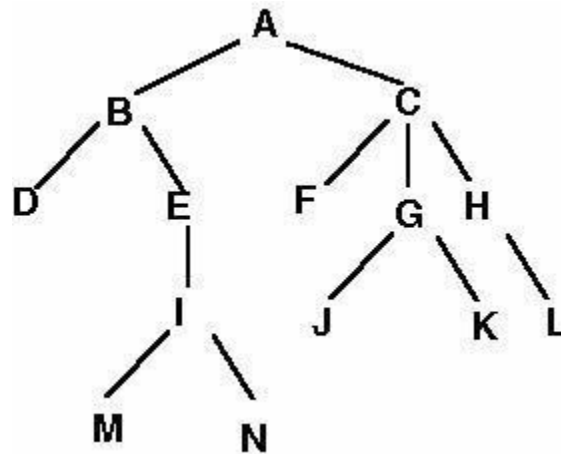
Query executed successfully. | DESKTOP-SFL7UVG\SQLEXPRESS ... | DESKTOP-SFL7UVG\Window... | GAME_CARO | 00:00:00 | 6 rows

CHƯƠNG V: ỨNG DỤNG TRÍ TUỆ NHÂN TẠO VÀO ĐỒ ÁN

a) Thuật toán AI sử dụng trong đồ án – cơ sở lý thuyết

Trong phần này, chỉ sử dụng một thuật toán có tính chất vét cạn, ở mỗi ô đang đánh giá, quét 8 hướng với chiều dài 4 ô (không tính ô hiện tại đang xét) và cộng điểm tấn công với phòng ngự dựa trên mảng điểm đã cho. Ý tưởng này lấy nền tảng từ thuật toán Minimax nhưng không có phần đánh giá về độ sâu để thật sự chỉ ra nước đi nào tối ưu nhất nhưng vì thế sẽ giúp chương trình chạy nhanh hơn.

Sử dụng cấu trúc cây:



Tại lượt đi A, tìm kiếm các nước đi có thể đánh B và C.

Tính trước các nước D, E tương ứng với B và F, G, H tương ứng với C.

Tiếp tục tính các nước đi tương ứng với E và M, N tương ứng với I.

Tương ứng với mỗi nước đi, cần lần lượt quét các hướng:

Lần lượt duyệt các hướng và tính toán cả việc tấn công và phòng ngự có lợi ở nước đó.

Dùng 2 mảng để lưu trữ điểm tấn công và phòng ngự:

- Điểm tấn công = {0, 9, 54, 162, 1458, 13112, 118008,...}
- Điểm phòng ngự = {0, 3, 27, 99, 729, 6561, 59049,...}

Theo hướng đó có bao nhiêu quân thì ta cộng vào bấy nhiêu điểm tấn công hay phòng ngự và ngược lại.

Lặp lại với các nước con có thể đánh ở lượt sau.

b) Tổng quát các thành phần trong thuật toán

Mảng điểm đánh giá:

```
private int[] Attack = new int[7] { 0, 9, 54, 162, 1458, 13112, 118008 };  
private int[] Defense = new int[7] { 0, 3, 27, 99, 729, 6561, 59049 };
```

Hàm tìm kiếm nước đi (Trả về ô cờ nào cần được đánh sau khi quét các ô chưa đánh)

```
1 reference  
private void CptFindChess()  
{  
    if (gameover) return;  
    long max = 0;  
    int imax = 1, jmax = 1;  
    for (int i = 1; i < rows; i++)  
    {  
        for (int j = 1; j < columns; j++)  
            if (vtMap[i, j] == 0)  
            {  
                long temp = Caculate(i, j);  
                if (temp > max)  
                {  
                    max = temp;  
                    imax = i; jmax = j;  
                }  
            }  
    }  
    PutChess(imax, jmax);  
}
```

Công điểm tấn công và phòng thủ để lựa chọn hướng đi tốt nhất

Bằng cách tính điểm cho mỗi bước đi:

- MAX: Tấn công => Chọn nước đi mà máy có điểm cao nhất
- MIN: Phòng thủ => Chọn nước đi sao cho người chơi điểm thấp nhất

Khi đó, lựa chọn tối ưu: ([Tấn công + Phòng thủ]): Máy tấn công thì chọn MAX, còn phòng thủ thì chọn MIN

```

1 reference
private long Caculate(int x, int y)
{
    return EnemyChessess(x, y) + ComputerChessess(x, y);
}
1 reference

```

Các hàm duyệt điểm tấn công – phòng ngự để cộng điểm trong hàm tìm kiếm nước đi theo chiều dọc, chiều ngang, chéo trên, chéo dưới

```

int column = 0, row = 0, mdiagonal = 0, ediagonal = 0;

```

- Chiều dọc

```

while (vtMap[i, j] == 2 && i >= 0)
{
    column++;
    i--;
}
if (vtMap[i, j] == 0) sc_ = 1;
i = x + 1;
while (vtMap[i, j] == 2 && i <= rows)
{
    column++;
    i++;
}
if (vtMap[i, j] == 0) sc = 1;

```

- Chiều ngang

```

while (vtMap[i, j] == 2 && j >= 0)
{
    row++;
    j--;
}
if (vtMap[i, j] == 0) sr_ = 1;
j = y + 1;
while (vtMap[i, j] == 2 && j <= columns)
{
    row++;
    j++;
}
if (vtMap[i, j] == 0) sr = 1;

```

- Chéo trên

```
while (vtMap[i, j] == 2 && i >= 0 && j >= 0)
{
    mdiagonal++;
    i--;
    j--;
}
if (vtMap[i, j] == 0) sm_ = 1;
i = x + 1; j = y + 1;
while (vtMap[i, j] == 2 && i <= rows && j <= columns)
{
    mdiagonal++;
    i++;
    j++;
}
if (vtMap[i, j] == 0) sm = 1;
```

- Chéo dưới

```
while (vtMap[i, j] == 2 && i >= 0 && j <= columns)
{
    ediagonal++;
    i--;
    j++;
}
if (vtMap[i, j] == 0) se_ = 1;
i = x + 1; j = y - 1;
while (vtMap[i, j] == 2 && i <= rows && j >= 0)
{
    ediagonal++;
    i++;
    j--;
}
if (vtMap[i, j] == 0) se = 1;
```

TỔNG KẾT

BẢNG PHÂN CÔNG CÔNG VIỆC

Phân chia công việc

- Thiết kế giao diện trò chơi
- Tạo và quản lý database
- Xử lý đăng nhập, đăng ký, đăng xuất
- Hiển thị quân cờ

Hoàng Tuấn

- Tổng hợp chức năng, bố cục game
- Tạo bàn cờ
- Đếm thời gian
- Bảng xếp hạng
- Tính năng chat
- Thuyết trình
- Làm slide

Thùy Chinh

- Nghiên cứu thuật toán, chơi với máy
- Chức năng Undo
- Đổi người chơi
- Xử lý thắng thua
- Làm báo cáo file word

Hải Đăng