BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

SỬ DỤNG CÔNG CỤ WINFORM TRONG C# MÔ PHỎNG GAME CỜ CARO (2 NGƯỜI CHƠI)

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Ngô Phúc Thịnh

Mã số sinh viên: 62132029

KHÁNH HÒA-2022

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN HỆ THỐNG THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

SỬ DỤNG CÔNG CỤ WINFORM TRONG C# MÔ PHỎNG GAME CỜ CARO (2 NGƯỜI CHƠI)

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Ngô Phúc Thịnh

Mã số sinh viên: 62132029

Khánh Hòa, tháng 12/2022

TRƯỜNG ĐẠI HỌC NHA TRANG
Khoa: Công nghệ Thông tin

PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ BÁO CÁO THỰC TẬP CƠ SỞ

Tên đề tài: Sử dụng công cụ winform trong c# mô phỏng game cờ caro (2 người chơi)

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên được hướng dẫn: Ngô Phúc Thịnh

MSSV: 62132029

Khóa: 62 **Ngành:** Công nghệ Thông tin

Lần	Ngày	Nội dung	Nhận xét của GVHD
1	11/12/2022	Nhận đề tài hướng dẫn và định hướng giải quyết vấn đề. Sinh viên trình bày demo đề tài đã chọn	Sinh viên và GVHD trao đổi nội dung của đề tài. Yêu cầu làm thêm đồng hồ đếm thời gian và làm thêm màn hình menu và màn hình nhập tên người chơi
2	18/12/2022	Sinh viên trình bày tốt những gì mà tuần trước giáo viên yêu cầu tuy nhiên	Sinh viên hiểu được vấn đề cần phải thực hiện thêm. Yêu cầu làm thêm một màn hình about giới thiệu thông tin người làm và làm thêm trong phần điền tên người chơi phần chọn kích cỡ lưới của bàn cờ.
3	28/12/2022	Sinh viên hoàn thiệt yêu cầu tuần trước giáo viên yêu cầu.	Giáo viên hướng dẫn viết báo cáo.
4	4/01/2023	Sinh viên nộp bản thảo của báo cáo thực tập lần thứ 1 và tiến hành chỉnh sửa.	Báo cáo chỉ trình bày chung chung chưa đi vào cụ thể phân tích các yêu cầu của bài toán, hình ảnh, bảng biểu chưa trình bày rõ ràng. Cần hiệu chỉnh theo yêu cầu của GVHD.
5	11/01/2023	Sinh viên nộp bản thảo lần 2 và có minh họa với thư viện của	Báo cáo lần này đã khắc phục được các lỗi của lần trước, tuy nhiên phần

		chuột nhưng chưa thể kết nối với phần trước.	phương pháp và kết quả chưa nổi bật, chưa có sự liên kết giữa các phần.
6	18/1/2023	Sinh viên nộp bản thảo lần cuối sau khi đã chỉnh sửa các yêu cầu như đã đề ra.	Sinh viên nghiêm túc chỉnh sửa báo cáo theo định hướng của GVHD.

Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KL):

Sinh viên thực hiện tốt các yêu cầu của GVHD, trong quá trình thực hiện đề tài có sự liên hệ chặt chẽ với GV. Theo lịch hẹn Sinh viên đều có mặt để trình bày ý tưởng của các nội dung lần trước. Trong quá trình hoàn tất báo cáo đều nỗ lực không ngừng mặc dù đang cao điểm của đợt thi học kỳ nhưng SV vẫn dành thời gian không ít cho TTCS.

Về nội dung báo cáo đã thỏa mãn các yêu cầu của đề tài như trong đề cương. Về kết quả chương trình đã minh họa được thuật toán. Về các yêu cầu cao hơn như sử dụng thư viện của chuột hay minh họa các trường hợp nhược điểm của thuật toán thì chưa thực hiện thành công.

Về hình thức của báo cáo và sản phẩm, báo cáo trình bày rõ ràng các mục tiêu, phương pháp, kết quả và thảo luận cho sản phẩm. Còn về sản phẩm như đã trình bày có phần hạn chế.

Điểm hình thức

Đồng ý cho sinh viên: Được bảo vệ: ☑ Không được bảo vệ: ☐

Khánh Hòa, ngày 31 tháng 12 năm 2022

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để có thể hoàn thành đợt thực tập lần này, chúng em xin chân thành cảm ơn đến quý thầy cô khoa Công nghệ Thông tin đã tạo điều kiện hỗ trợ và giúp đỡ chúng em trong quá trình học tập và nghiên cứu đề tài này.

Qua đây, nhóm xin chân thành cảm ơn thầy Đoàn Vũ Thịnh, người đã trực tiếp quan tâm và hướng dẫn chúng em hoàn thành tốt đợt thực tập trong thời gian qua.

Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của chúng em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

MỤC LỤC

DANH MỤC HÌNH

TÓM TẮT

Windows Forms là thư viện lớp đồ họa mã nguồn mở và miễn phí được bao gồm như một phần của Microsoft.NET Framework hoặc Mono Framework, cung cấp nền tảng để viết các ứng dụng khách phong phú cho máy tính để bàn, máy tính xách tay và máy tính bảng.

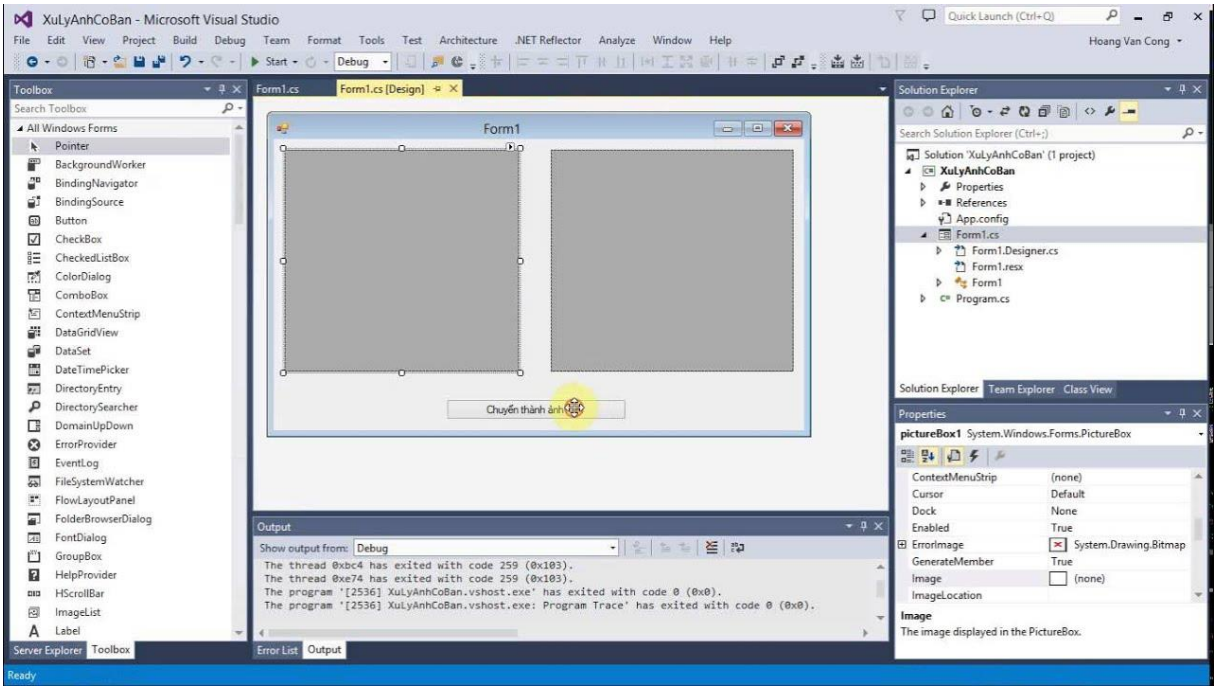
Quy trình thực hiện các bước từ thiết kế và cài đặt thuật toán để tạo ra một game cờ caro hoàn chỉnh đều được thực hiện trên môi trường winform trong c# thông qua ứng dụng Visual studio.

Sản phẩm đã mô phỏng được game cờ caro (2 người chơi) sử dụng tốt và hoàn chỉnh

Toàn bộ mã nguồn của báo cáo được tải lên theo địa chỉ
https://github.com/thinhdoanvu/DOAN_TTCS_TTCN/tree/main/NgoPhucThinh_62132029

GIỚI THIỆU

Winform là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWorrk và có giao diện người dùng Windows Forms.
Mỗi màn hình windows cung cấp một giao diện giúp người dùng giao tiếp với ứng dụng. Giao diện này được gọi là giao diện đồ họa (*GUI*) của ứng dụng.
Là các ứng dụng windows chạy trên máy tính – mã lệnh thực thi ngay trên máy tính: Microsoft, Word, Excel, Access, Calculator, yahoo, Mail… là các ứng dụng **Windows Forms**.
Hiện nay, số lượng các công ty tuyển dụng yêu cầu WinForm cũng không nhiều. Nếu công ty đòi hỏi kiến thức WinForm, có lẽ bạn sẽ phải bảo trì một (hoặc nhiều) dự án WinForm cũ rích. Mà hẳn là không bạn nào muốn làm công việc bảo trì nhàm chán qua ngày này tháng nọ phải không?
Qua bài viết này các bạn đã hiểu Winform là gì rồi và cũng có thể trả lời cho câu hỏi có nên học winform hay không rồi đúng không? Mời bạn tiếp tục tham khảo thêm các khóa học bổ ích khác của chúng tôi:



Hình 1. 1 Ví dụ về winform

Tuy nhiên, hiện nay rất ít người sử dụng Winform trong C# để làm game mà sử dụng những phần mềm như Unreal Engine, Unity, GameMaker Studio 2, BuildBox, …

1.1. Thuật toán Tạo bàn cờ

Tạo một class thông tin thuộc tính

```
namespace Caro_Game
{
    public class MyConst
    {
        static public int _ChessBoard_SizeHeight = 29;
        static public int _ChessBoard_SizeWitdth = 29;
        static public int _NumColumn = 20;
        static public int _NumRow = 20;
        static public float _SizeBoderChessBoard = 1.8f;
        static public Color color_O = Color.Red;
        static public Color color_X = Color.Blue;
        static public string fileName_X = @"images\x.gif";
        static public string fileName_O = @"images\o.gif";

    }
}
```

Tạo một class thông tin của bàn cờ

```
namespace Caro_Game
{
    class Chess_BoardInfo
    {
        int index_Column;
        int index_Row;
        int ownership;
        Point location;
        string fileImage;
        public int Index_Column { get => index_Column; }
        public int Index_Row { get => index_Row; }
        public Point Location { get => location; }
        public int Ownership { get => ownership; set => ownership = value; }
        public string FileImage { get => fileImage; set => fileImage = value; }

        public Chess_BoardInfo()
        {
            this.index_Column = this.index_Row = 0;
            this.location = new Point(0, 0);
        }
        public Chess_BoardInfo(int index_Row,int index_Column, Point location,int currPlayer)
        {
            this.index_Row = index_Row;
            this.index_Column = index_Column;
            this.location = location;
            this.ownership = currPlayer;
            fileImage = "";
        }

    }
}
```

Dùng hàm pen để vẽ lại bàn cờ vẽ cột và hàng của bàng cờ

```
public class Chessboard
{
    Graphics _grsChessBoard;
    public Graphics GrsChessBoard { get => _grsChessBoard; }
    internal List<List<Chess_BoardInfo>> ListChessBoard { get => listChessBoard; }

    List<List<Chess_BoardInfo>> listChessBoard;
    public Chessboard(Graphics grs)
    {
        _grsChessBoard = grs;
        listChessBoard = new List<List<Chess_BoardInfo>>();
        load_CheckBoard();
    }

    public void drawChessboard()
    {
        Pen pen = new Pen(Color.Brown, MyConst._SizeBoderChessBoard);
        // Vẽ các cột
        for (int i = 0; i <= MyConst._NumColumn; i++)
        {
            GrsChessBoard.DrawLine(pen, i* MyConst._ChessBoard_SizeHeight, 0,
                i * MyConst._ChessBoard_SizeHeight,601);

        }
        // vẽ các hàng
        for (int i = 0; i <= MyConst._NumRow; i++)
        {
            GrsChessBoard.DrawLine(pen, 0, i* MyConst._ChessBoard_SizeWitdth,
                691, i * MyConst._ChessBoard_SizeWitdth);
        }

    }
```

1.2. Thuật toán Xử lý thắng thua

1.2.1. Ý tưởng chính của thuật toán

List nằm trong list

Lưu điểm đã đánh vào trong list

Và duyệt theo đường thẳng, ngang, đường chéo chính, đường chéo phụ có đủ 5 điểm chưa nếu đủ thì sẽ tính thắng

1.2.2. thực hiện thuật toán

Tạo hàm kiểm tra thắng trước sau đó tạo các hàm duyệt đường ngang, đường thẳng, đường chéo chính và đường chéo phụ

```
public bool checkWin()
{
    if (StackChess.Count == MyConst._NumRow * MyConst._NumColumn)
    {
        _gameOver = Winner.NOONE;
        turn = 3;
        return true;
    }


    foreach (Chess_BoardInfo item in StackChess)
    {
        if (browseAlong(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseHorizontal(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseMainDiagonal(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseExtraDiagonal(item.Index_Row, item.Index_Column, item.Ownership))
        {
            _gameOver = turn != 1 ? Winner.PLAYER1 : Winner.PLAYER2;
            return true;
        }
    }
    return false;

}
```

Hình 1. 2 hàm kiểm tra thắng thua

```
bool browseAlong(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_R > MyConst._NumRow - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R + cout][currIndex_C].Ownership != currPlayer)
            return false;
    }
    if (currIndex_R == 0 || currIndex_R + cout == MyConst._NumRow)
        return true;
    if (_listChessBoard[currIndex_R + cout][currIndex_C].Ownership == 0 || _listChessBoard[currIndex_R - 1][currIndex_C].Ownership == 0)
        return true;

    return false;
}
bool browseHorizontal(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_C > MyConst._NumColumn - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_C == 0 || currIndex_C + cout == MyConst._NumColumn)
        return true;
    if (_listChessBoard[currIndex_R][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R][currIndex_C - 1].Ownership == 0)
        return true;

    return false;

}
```

Hình 1. 3 hàm duyệt đường ngang và đường thẳng

```
bool browseMainDiagonal(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_C > MyConst._NumColumn - 5 || currIndex_R > MyConst._NumRow - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R + cout][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_C + cout == MyConst._NumColumn || currIndex_R + cout == MyConst._NumRow || currIndex_R == 0 || currIndex_C == 0)
        return true;
    if (_listChessBoard[currIndex_R + cout][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R - 1][currIndex_C - 1].Ownership == 0)
        return true;

    return false;

}
bool browseExtraDiagonal(int currIndex_R, int currIndex_C, int currPlayer)
{

    if (currIndex_R < 4 || currIndex_C > MyConst._NumColumn - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R - cout][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_R == 4 || currIndex_C + cout == MyConst._NumColumn || currIndex_C == 0 || currIndex_R == MyConst._NumRow - 1)
        return true;
    if (_listChessBoard[currIndex_R - cout][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R + 1][currIndex_C - 1].Ownership == 0)
        return true;
    return false;

}
```

Hình 1. 4 duyệt đường chéo chính và đường chéo phụ

CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Tạo bàn cờ

```
namespace Caro_Game
{
    public class MyConst
    {
        static public int _ChessBoard_SizeHeight = 29;
        static public int _ChessBoard_SizeWitdth = 29;
        static public int _NumColumn = 20;
        static public int _NumRow = 20;
        static public float _SizeBoderChessBoard = 1.8f;
        static public Color color_O = Color.Red;
        static public Color color_X = Color.Blue;
        static public string fileName_X = @"images\x.gif";
        static public string fileName_O = @"images\o.gif";

    }
}
```

Hình 2. 1 tạo một class lưu thông tin để dễ dàng khai báo

```
namespace Caro_Game
{
    class Chess_BoardInfo
    {
        int index_Column;
        int index_Row;
        int ownership;
        Point location;
        string fileImage;
        public int Index_Column { get => index_Column; }
        public int Index_Row { get => index_Row; }
        public Point Location { get => location; }
        public int Ownership { get => ownership; set => ownership = value; }
        public string FileImage { get => fileImage; set => fileImage = value; }

        public Chess_BoardInfo()
        {
            this.index_Column = this.index_Row = 0;
            this.location = new Point(0, 0);
        }
        public Chess_BoardInfo(int index_Row,int index_Column, Point location,int currPlayer)
        {
            this.index_Row = index_Row;
            this.index_Column = index_Column;
            this.location = location;
            this.ownership = currPlayer;
            fileImage = "";
        }

    }
}
```

Hình 2. 2 tạo class thông tin bàn cờ

```
public class Chessboard
{
    Graphics _grsChessBoard;
    public Graphics GrsChessBoard { get => _grsChessBoard; }
    internal List<List<Chess_BoardInfo>> ListChessBoard { get => listChessBoard; }

    List<List<Chess_BoardInfo>> listChessBoard;
    public Chessboard(Graphics grs)
    {
        _grsChessBoard = grs;
        listChessBoard = new List<List<Chess_BoardInfo>>();
        load_CheckBoard();
    }

    public void drawChessboard()
    {
        Pen pen = new Pen(Color.Brown, MyConst._SizeBoderChessBoard);
        // Vẽ các cột
        for (int i = 0; i <= MyConst._NumColumn; i++)
        {
            GrsChessBoard.DrawLine(pen, i* MyConst._ChessBoard_SizeHeight, 0,
                i * MyConst._ChessBoard_SizeHeight,601);

        }
        // vẽ các hàng
        for (int i = 0; i <= MyConst._NumRow; i++)
        {
            GrsChessBoard.DrawLine(pen, 0, i* MyConst._ChessBoard_SizeWitdth,
                691, i * MyConst._ChessBoard_SizeWitdth);
        }

    }
```

Hình 2. 3 tạo bàn cờ vẽ cột hàng bằng code

```
public void load_CheckBoard()
{

    listChessBoard.Clear();
    for (int i = 0; i < MyConst._NumRow; i++)
    {
        List<Chess_BoardInfo> tmp = new List<Chess_BoardInfo>();
        listChessBoard.Add(tmp);
        for (int j = 0; j < MyConst._NumColumn; j++)
        {
            Point p = new Point(j * MyConst._ChessBoard_SizeWitdth, i * MyConst._ChessBoard_SizeHeight);
            tmp.Add(new Chess_BoardInfo(i, j, p, 0));
        }
    }
}
public void drawChess_Board(Point p,Bitmap bm)
{
    if (bm == null)
        return;
    Rectangle rte = new Rectangle(p.X+2,p.Y+2, MyConst._ChessBoard_SizeWitdth-3,
        MyConst._ChessBoard_SizeHeight-3);
    _grsChessBoard.DrawImage(bm,rte);

}
```

Hình 2. 4 load lại bàn cờ

2.2. Đổi người chơi

```
bool chess(int Mouse_X, int Mouse_Y)
{
    if (Mouse_X % MyConst._ChessBoard_SizeWitdth == 0 ||
        Mouse_Y % MyConst._ChessBoard_SizeHeight == 0)
    {
        return false;
    }
    int column = Mouse_X / MyConst._ChessBoard_SizeWitdth;
    int row = Mouse_Y / MyConst._ChessBoard_SizeHeight;

    if (_listChessBoard[row][column].Ownership != 0)
        return false;

    _listChessBoard[row][column].Ownership = turn;

    _listChessBoard[row][column].FileImage = _listChessBoard[row][column].Ownership == 1 ? MyConst.fileName_X : MyConst.fileName_O;
    drawChess(_listChessBoard[row][column]);
    StackChess.Push(_listChessBoard[row][column]);
    turn = turn != 1 ? 1 : 2;
    return true;
}
```

Hình 2. 5 Xử lý đổi người chơi

2.3. Xử lý thắng thua

```
enum Winner { PLAYER1, PLAYER2, NOONE, NONE }
public class GameCaroChess
{
    Chessboard _Chesssboard;
    Graphics _grs;
    Stack<Chess_BoardInfo> _stackChess;
    Winner _gameOver;
    int turn;
    int mod;
    string nameP1;
    string nameP2;
    List<List<Chess_BoardInfo>> _listChessBoard;
    public Chessboard chessboard { get => _Chesssboard; set => _Chesssboard = value; }
    public Graphics Grs { get => _grs; }
    public int Turn { get => turn; set => turn = value; }
    internal Winner GameOver { get => _gameOver; }
    public int Mod { get => mod; set => mod = value; }
    public string NameP1 { get => nameP1; set => nameP1 = value; }
    public string NameP2 { get => nameP2; set => nameP2 = value; }
    internal Stack<Chess_BoardInfo> StackChess { get => _stackChess; set => _stackChess = value; }
```

Hình 2. 6 khai báo thông tin để dùng trong các hàm dưới

```
void win()
{
    string nof = "";
    switch (_gameOver)
    {
        case Winner.PLAYER1:
            nof = string.Format("'{0}' Thắng", nameP1);
            break;
        case Winner.PLAYER2:
            nof = string.Format("'{0}' Thắng", nameP2);
            break;
        case Winner.NOONE:
            nof = "Hòa";
            break;

        default:
            return;
    }
    MessageBox.Show(nof, "GameOver");

}
```

Hình 2. 7 tạo hàm thông báo thắng và hòa

```
public bool checkWin()
{
    if (StackChess.Count == MyConst._NumRow * MyConst._NumColumn)
    {
        _gameOver = Winner.NOONE;
        turn = 3;
        return true;
    }


    foreach (Chess_BoardInfo item in StackChess)
    {
        if (browseAlong(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseHorizontal(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseMainDiagonal(item.Index_Column, item.Index_Row, item.Ownership) ||
      browseExtraDiagonal(item.Index_Row, item.Index_Column, item.Ownership))
        {
            _gameOver = turn != 1 ? Winner.PLAYER1 : Winner.PLAYER2;
            return true;
        }
    }
    return false;

}
```

Hình 2. 8 Kiểm tra thắng thua theo các hàm dọc ngang và đường chéo chính và chéo phụ

```
bool browseAlong(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_R > MyConst._NumRow - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R + cout][currIndex_C].Ownership != currPlayer)
            return false;
    }
    if (currIndex_R == 0 || currIndex_R + cout == MyConst._NumRow)
        return true;
    if (_listChessBoard[currIndex_R + cout][currIndex_C].Ownership == 0 || _listChessBoard[currIndex_R - 1][currIndex_C].Ownership == 0)
        return true;

    return false;
}
bool browseHorizontal(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_C > MyConst._NumColumn - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_C == 0 || currIndex_C + cout == MyConst._NumColumn)
        return true;
    if (_listChessBoard[currIndex_R][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R][currIndex_C - 1].Ownership == 0)
        return true;

    return false;

}
```

Hình 2. 9 Duyệt theo đường thẳng và đường ngang

```
bool browseMainDiagonal(int currIndex_C, int currIndex_R, int currPlayer)
{
    if (currIndex_C > MyConst._NumColumn - 5 || currIndex_R > MyConst._NumRow - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R + cout][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_C + cout == MyConst._NumColumn || currIndex_R + cout == MyConst._NumRow || currIndex_R == 0 || currIndex_C == 0)
        return true;
    if (_listChessBoard[currIndex_R + cout][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R - 1][currIndex_C - 1].Ownership == 0)
        return true;

    return false;

}
bool browseExtraDiagonal(int currIndex_R, int currIndex_C, int currPlayer)
{

    if (currIndex_R < 4 || currIndex_C > MyConst._NumColumn - 5)
        return false;
    int cout = 1;
    for (cout = 1; cout < 5; cout++)
    {
        if (_listChessBoard[currIndex_R - cout][currIndex_C + cout].Ownership != currPlayer)
            return false;
    }
    if (currIndex_R == 4 || currIndex_C + cout == MyConst._NumColumn || currIndex_C == 0 || currIndex_R == MyConst._NumRow - 1)
        return true;
    if (_listChessBoard[currIndex_R - cout][currIndex_C + cout].Ownership == 0 || _listChessBoard[currIndex_R + 1][currIndex_C - 1].Ownership == 0)
        return true;
    return false;

}
```

Hình 2. 10 Duyệt theo đường chéo chính và đường chéo phụ

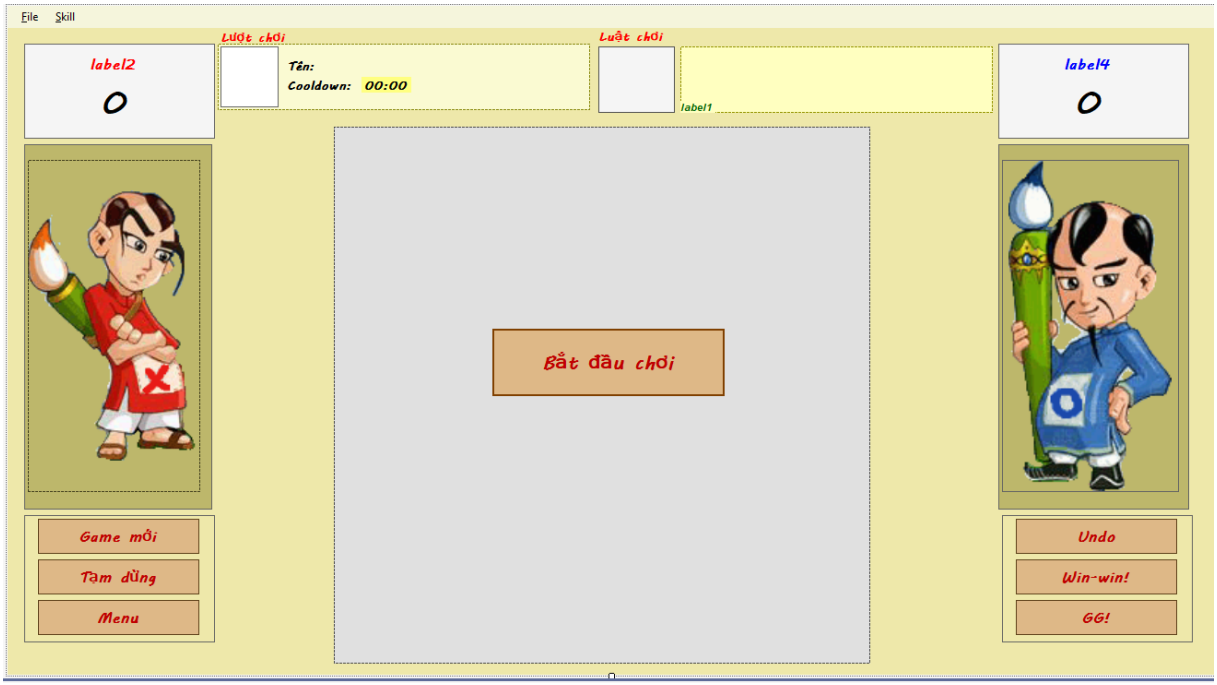
2.4 cooldown

```
#region cooldown
int time = 20;
private void TimerCoolDown_Tick(object sender, EventArgs e)
{
    if (time < 0)
    {
        restartCooldown();
        cooldownEnd();
        return;
    }
    time -= 1;
    if (time >= 0)
        lbCooldown.Text = formatTime(time);
}
string formatTime(int i)
{
    string rsl = "00:";
    if (i < 10)
        rsl += "0" + i.ToString();
    else
        rsl += i.ToString();
    return rsl;
}
void cooldownEnd()
{
    string rsl = PlayGameCaro.Turn == 1 ? nameP2 : nameP1;
    timerCoolDown.Stop();
    MessageBox.Show(string.Format("'{0}' Thắng ", rsl), "GameOver");
    afterGameOver();



}
void pauseGame()
{
    timerCoolDown.Stop();
    DialogResult dialogResult = MessageBox.Show("Bấm \"Ok\" để tiếp game!!!", "Pause Game");
    timerCoolDown.Start();
}
void restartCooldown()
{
    lbCooldown.Text = formatTime(20);
    time = 20;
}
#endregion
```

Hình 2. 11 xử lý cooldown



Hình 2. 12 Tổng quan thiết kế của form playgame

2.5 Thiết kế và xử lý Form menu

Hình 2. 13 thiết kế form menu

2.6 Thiết kế và xử lý Form name

Hình 2. 14 thiết kế form name

```
bool isValidName(string name)
{
    if (name.Length > 10 || name.Length < 2)
        return false;
    char[] arrayChar = name.ToCharArray();
    foreach (char item in arrayChar)
    {
        if (((item < 'a' || item > 'z') && (item < 'A' || item > 'Z')) && item != ' ' && (item < '0' || item > '9'))
            return false;
    }
    return true;
}
```

Hình 2. 15 xử lý cách nhập tên

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
     MyConst._NumColumn = 9;
     MyConst._NumRow = 9;
    MyConst._ChessBoard_SizeHeight = 64;
    MyConst._ChessBoard_SizeWitdth = 64;

}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    MyConst._NumColumn = 19;
    MyConst._NumRow = 19;
    MyConst._ChessBoard_SizeHeight = 32;
    MyConst._ChessBoard_SizeWitdth = 32;

}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    MyConst._NumColumn = 21;
    MyConst._NumRow = 21;
    MyConst._ChessBoard_SizeHeight = 34;
    MyConst._ChessBoard_SizeWitdth = 34;

}
```

Hình 2. 16 Xử lý lựa chọn kích thước lưới

2.7 Thiết kế và xử lý form about

Hình 2. 17 thiết kế form about

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3.1. Bắt đầu game

Hình 3. 1 bắt đầu game

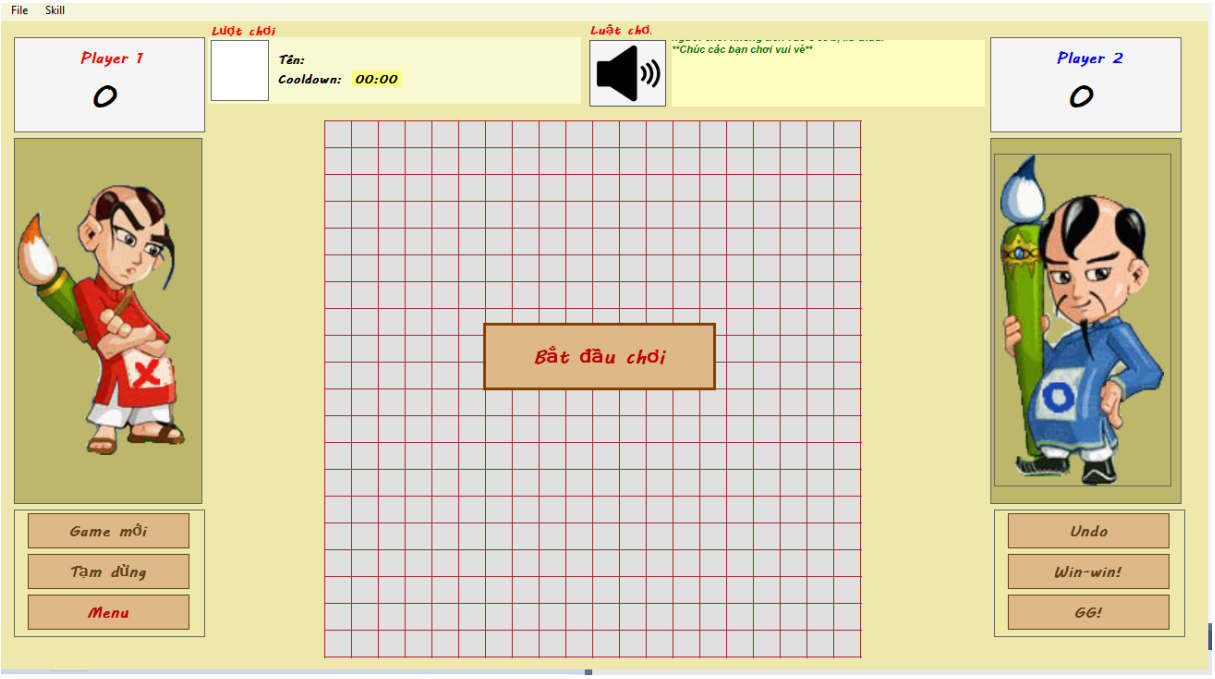
Hình 3.1 nút 2 player là sẽ hiển thị màn hình nhập tên và chọn kích thước, nút About sẽ hiển thị form about và nút thoát game sẽ thoát game

3.2. Vào màn hình nhập tên và chọn kích thước bàn cờ

Hình 3. 2 nhập tên và chọn kích thước

Hình 3.3. Sau khi vào màn hình điền tên thì sẽ có thể đổi tên người chơi và chọn kích thước bàn cờ caro và sẽ chọn nút game mới để vào game hoặc

3.3 vào màn hình chơi game



Hình 3. 3 màn hình chơi game

Hình 3.3 màn hình bắt đầu chơi người chơi có thể click bắt đầu game và chơi phía góc phải sẽ hiện luật chơi góc trái là hiển thị từng lượt người chơi và có thời gian chơi là 20s.

Phía dưới góc trái là nút game mới người chơi có thể out ra màn hình nhập tên để có thể chọn lại kích thước bàn cờ và tên nhân vật và nút tạm dừng game và out ra menu.

Phía dưới góc trái là khi bắt đầu game có nút undo để người chơi có thể đánh lại nút thứ hai là win-win là đầu hàng người chơi còn lại sẽ thắng. và nút GG sẽ là kết quả hòa.

THẢO LUẬN

Đề tài sử dụng winform trong c# mô phỏng game cờ caro đã giúp em có thể thiết kế 1 game cơ bản dễ dàng sử dụng chơi. Giúp em có thể tự tin để lập trình các dự án tiếp theo và môi trường winform giúp em làm quen được với việc thiết kế được một game cơ bản và việc thiết kế các trang web trong tương lai.

Trong tương lai, em sẽ tiếp tục tìm hiểu thêm về thiết kế các website và cách làm game trên các phần mềm khác như Unreal Engine, Unity, GameMaker Studio 2, BuildBox, …

TÀI LIỆU THAM KHẢO

1. Lập trình Winform cơ bản | How Kteam
2. Kỹ thuật lập trình / Bùi Đức Dương
3. Bài giảng lập trình hướng đối tượng / Nguyễn Đình Hưng