

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------------------------------

**BÁO CÁO BÀI TẬP LỚN**

**Đề tài: Lập trình game cờ vua bằng Python**

**Nhóm: LỚP N10 – NHÓM 09**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên các thành viên** | | | |
| **Tên thành viên** | | **Mã số sinh viên** |
| Nguyễn Anh Tú | | B21DCCN747 |
| Kiều Văn Hiếu | | B21DCCN052 |
| Đặng Minh Đức | | B21DCCN236 |
| Lê Đức Thắng | | B21DCCN664 |
|  | **Giảng viên:** TS. Nguyễn Trọng Khánh | | | |

**Hà Nội 2023**

**MỤC LỤC**

[**PHẦN 1: Giới thiệu 1**](#_Toc150283226)

[**1.1. Tổng quan về game cờ vua 1**](#_Toc150283227)

[**1.2. Giới thiệu về ứng dụng 3**](#_Toc150283228)

[**PHẦN 2: Nội dung 4**](#_Toc150283229)

[**2.1 Xây dựng các chức năng cơ bản của game cờ vua 4**](#_Toc150283230)

[**2.2 Xây dựng ChessBot cho game 8**](#_Toc150283234)

[**2.3 Xây dựng Menu 12**](#_Toc150283235)

[**PHẦN 3: Chạy chương trình 13**](#_Toc150283236)

[**3.1 Giao diện Menu 13**](#_Toc150283237)

[**3.2 Bắt đầu trò chơi 13**](#_Toc150283238)

[**3.3 Nút Info 17**](#_Toc150283239)

[**PHẦN 4: Kết luận 18**](#_Toc150283240)

[**4.1 Đánh giá 18**](#_Toc150283241)

[**4.2 Một số tính năng để cải thiện ứng dụng 18**](#_Toc150283242)

[**Tài liệu tham khảo 18**](#_Toc150283243)

**Bảng phân công công việc**

|  |  |
| --- | --- |
| **Thành viên** | **Nhiệm vụ** |
| Đặng Minh Đức | Lập trình game cờ vua |
| Kiều Văn Hiếu |
| Nguyễn Anh Tú | Xây dựng ChessBot cho game |
| Lê Đức Thắng | Thiết kế Menu cho game |

# PHẦN 1: Giới thiệu

## Tổng quan về game cờ vua

**Cờ vua**, trước kia còn được gọi là **Cờ quốc tế**, là một trò chơi board game cho 2 người chơi. Sau thời gian phát triển từ các trò chơi cổ tương tự có nguồn gốc từ Ấn Độ và Ba Tư, hình thức chơi hiện tại của cờ vua bắt đầu xuất hiện ở Nam Âu ở nửa sau của thế kỷ 15. Ngày nay, cờ vua là một trong những trò chơi trí tuệ phổ biến nhất thế giới; nó được nói đến không chỉ như là một trò chơi mà còn là nghệ thuật, khoa học và thể thao.

Cờ vua được diễn ra trên một bảng hình vuông, gồm 8 hàng (đánh số từ 1 đến 8) và 8 cột (đánh chữ cái từ a đến h), tạo ra 64 ô vuông với các màu đậm nhạt xen kẽ nhau. Mỗi người chơi sẽ bắt đầu ván cờ với 16 quân cờ bao gồm: 8 Tốt, 2 Mã, 2 Tượng, 2 Xe, 1 Hậu và 1 Vua. Người chơi được chia làm 2 bên: quân trắng và quân đen. Người cầm quân trắng là người đi trước.



Hình 1.1. Mô tả bàn cờ vua

Cách xếp bàn cờ được thực hiện tương tự như trong ảnh *1.1*. Như vậy, quân trắng sẽ được xếp vào hàng đầu tiên (hàng 1) theo thứ tự từ trái sang phải (từ a đến h) lần lượt là: xe, mã, tượng, hậu, vua, tượng, mã, xe. Hàng thứ hai được xếp 8 quân tốt trắng. Cách xếp quân đen đối xứng hoàn toàn với quân trắng ở hai hàng cuối cùng.

Các quân cờ có nước đi khác nhau, cụ thể:

**Xe** (ký hiệu quốc tế **R - Rook**) di chuyển theo các đường thẳng dọc theo cột hay hàng tới ô còn trống mà không có quân nào cản trên đường đi hay tới ô bị quân đối phương chiếm giữ (ăn quân) nhưng không thể vượt qua quân đang đứng ở ô đó. Ngoại lệ duy nhất là trường hợp nhập thành. Khi đó nó có thể nhảy qua quân vua của mình để đứng cạnh nó. Chỉ có xe mới có nước đi như thế.

**Tượng** (ký hiệu quốc tế **B - Bishop**) di chuyển theo đường chéo tới ô có cùng màu với nguyên lý tương tự như xe tới ô còn trống hay ô bị quân đối phương chiếm giữ (ăn quân).

**Hậu** (ký hiệu quốc tế **Q - Queen**) có nước đi là tổ hợp đơn giản của chuyển động của xe và tượng. Trong một nước đi nó có thể di chuyển theo đường chéo hoặc đường thẳng dọc theo cột hay hàng, với nguyên lý đi và ăn quân giống như tượng và xe.

**Mã** (ký hiệu quốc tế **N - Knight**) có thể di chuyển tới ô còn trống hay ô bị quân đối phương chiếm giữ (ăn quân) theo dạng hình chữ L (hình chữ nhật 3×2 hay 2×3). Quân mã không bị cản như trong cờ tướng.

**Tốt** (không cần ký hiệu) có thể di chuyển thẳng về phía trước chỉ một ô một lần tới ô còn trống (đi mà không ăn quân), nhưng khi di chuyển quân để ăn quân đối phương thì đi chéo. Nó có thể di chuyển 1 hoặc 2 ô nếu nó đi từ vị trí xuất phát ban đầu tới ô chưa bị chiếm giữ, nhưng không thể nhảy qua một quân khác để tới ô đó. Tốt còn một đặc điểm nữa là khi nó di chuyển đến hàng cuối cùng thì người chơi có quyền phong cấp cho nó thành bất kỳ quân cờ nào (hậu, xe, tượng, mã). Một nước đi đặc biệt khác của quân tốt là “Bắt Tốt qua đường” (En passant), chỉ xảy ra khi quân Tốt bắt quân Tốt vừa di chuyển hai ô trong lượt đầu tiên theo hướng chéo lên. Quân Tốt sau khi thực hiện en passant sẽ ở vị trí mà nếu như con Tốt bị bắt ban đầu chỉ di chuyển một ô. Luật này sinh ra nhằm đảm bảo rằng một quân Tốt sẽ không thể thoát khỏi Tốt của đối phương một cách dễ dàng bằng việc di chuyển hai ô ngay lập tức.

**Vua** (ký hiệu quốc tế là **K - King**) là quân quan trọng nhất, nếu mất vua thì người chơi thua cuộc. Mỗi lần đi nó có thể ăn quân hoặc di chuyển sang các ô bao quanh ô mà nó hiện tại đang chiếm giữ, nhưng không thể tới ô mà quân của mình đang chiếm giữ hay các ô bị quân đối phương kiểm soát. Ngoại lệ duy nhất là trường hợp nhập thành. Khi đó nó có thể di chuyển qua hai ô đồng thời với việc di chuyển quân xe của mình để quân xe đó đứng bên cạnh nó về phía cột trung tâm.

Khi ăn quân đối phương, quân tấn công sẽ di chuyển tới ô đó và thay thế cho quân đối phương tại vị trí này. Quân bị ăn được loại ra khỏi bàn cờ. Vua không thể không bảo vệ khỏi nước chiếu, do đó khi bị chiếu thì người chơi phải thực hiện các biện pháp nhằm cứu vua (di chuyển vua khỏi vị trí bị chiếu, ăn quân đang chiếu hay dùng quân khác của mình cản đường chiếu nếu có thể). Nếu không thể có nước đi để cứu vua thì người chơi bị chiếu bí và thua cuộc.

Trò chơi kết thúc khi một người chơi ăn được vua của đối phương hoặc rơi vào trạng thái hết nước đi, nghĩa là tình thế người chơi đến lượt đi không bị chiếu, nhưng cũng không thể thực hiện bất kì nước đi nào đúng luật. Trong trường hợp này thì kết quả được tính là hòa.

## Giới thiệu về ứng dụng

Ứng dụng chơi game cờ vua trên máy tính với các chức năng cơ bản. Khi khởi động ứng dụng, giao diện menu hiện ra gồm các chức năng chính:

- Chế độ 2 người chơi: 2 người chơi lần lượt thực hiện các nước đi tuân thủ theo các luật cờ vua đã nêu ở trên.

- Chế độ chơi với máy: Tương tự chế độ 2 người chơi nhưng người chơi quân đen sẽ do máy điều khiển.

- Xem một số hướng dẫn về các phím trong trò chơi, thông tin các thành viên trong nhóm

- Tắt chương trình

Trong lượt chơi của mình, người chơi có thể nhấn chọn một quân cờ, các nước đi hợp lệ sẽ được tô màu xanh lá, nhấn chọn ô màu xanh để thực hiện nước đi. Trò chơi kết thúc khi một bên bị chiếu hết cờ (checkmate) hoặc một bên không thể thực hiện nước đi nào hợp lệ (stalemate). Người chơi có thể nhấn phím “z” để quay lại nước đi trước đó, phím “r” để cài đặt lại bàn cờ về trạng thái bắt đầu, phím “m” để quay lại menu chính.

**Thư viện Pygame**

Pygame là một thư viện mã nguồn mở viết bằng Python, cung cấp các chức năng để tạo và phát triển các trò chơi. Pygame hỗ trợ nhiều nền tảng, hệ diều hành. Trong bài tập này, Pygame được sử dụng để hiển thị giao diện và xử lý các tương tác của người dùng trong game cờ vua.

# PHẦN 2: Nội dung

Mã nguồn chương trình được tổ chức thành 4 file:

- ChessEngine.py: chứa các lớp, hàm để lưu trữ trạng thái bàn cờ, kiểm tra và thực hiện các bước di chuyển

- chessmain.py: chứa hàm chính để chạy trò chơi, xử lí các sự kiện trong quá trình chơi và các hàm thiết kế giao diện, hiệu ứng của trò chơi.

- ChessBot: chứa hàm để tìm ra nước đi cho máy.

- Menu: chứa menu chính của chương trình, được chạy khi khởi động ứng dụng.

## 2.1 Xây dựng các chức năng cơ bản của game cờ vua

Nội dung phần này sẽ trình bày về các lớp, hàm cần thiết để xây dựng nên một game cờ vua tiêu chuẩn cho 2 người chơi.

**a) Cách lưu trữ bàn cờ**

Bàn cờ được lưu trữ dưới dạng một ma trận. Mỗi ô trong ma trận đại diện cho một ô vuông trên bàn cờ và chứa thông tin về quân cờ hiện đang ở ô đó. Mỗi quân cờ là một xâu kí tự với chữ cái bắt đầu đại diện cho phe trắng “w” và đen “b”; chữ cái thứ 2 sẽ thể hiện chức năng của quân cờ (“K”: vua, “Q”: hậu, “R”: xe, “K”: mã, “B”: tượng, “p”: tốt). Các ô trống sẽ chứa xâu “--".

***A screenshot of a computer screen

Description automatically generated***

Hình 2.1. Mô tả cách lưu trữ bàn cờ

**b) Lớp Move**

Class Move đại diện cho các nước đi với thuộc tính:

**ranksToRows**, **rowsToRanks**, **filesToCols**, **colsToFiles** dùng để xác định vị trí trên bàn cờ.

**startSq**: Đây là một tuple chứa tọa độ (hàng, cột) của ô xuất phát (nơi quân cờ bắt đầu di chuyển).

**endSq**: Đây cũng là một tuple chứa tọa độ (hàng, cột) của ô đích (nơi quân cờ di chuyển đến).

**board**: Biến này là bảng cờ chứa thông tin về vị trí của các quân cờ trên bàn cờ.

**isEnpassantMove**: Một biến kiểu boolean xác định nếu di chuyển này là một nước ăn tốt qua đường (en passant capture).

**isCastleMove**: Một cờ boolean xác định nếu di chuyển này là một nước nhập thành (castling move).

**startRow**, **startCol**: vị trí hàng và cột của ô xuất phát.

**endRow**, **endCol**: vị trí hàng và cột của ô đích đến.

**pieceMoved**: Biến này lưu trữ thông tin về quân cờ đã di chuyển từ ô xuất phát.

**pieceCaptured**: Biến này lưu trữ thông tin về quân cờ bị ăn nếu có.

Nếu **isEnpassantMove** là True, thì **pieceCaptured** sẽ là quân trắng (đen) nếu **pieceMoved** là quân đen (trắng).

**isPawnPromotion**: Một cờ boolean xác định nếu di chuyển này là một nước phong cấp tốt (pawn promotion), trả về True nếu thỏa mãn và ngược lại.

**isCapture**: nhận giá trị True nếu nước di chuyển này là 1 nước ăn quân đối phương.

**moveID**: Một số nguyên duy nhất được tạo ra từ tọa độ của ô xuất phát và ô đích, được sử dụng để xác định một di chuyển cụ thể.

Trong lớp Move có hàm **“\_\_eq\_\_”** so sánh hai đối tượng di chuyển dựa trên giá trị của thuộc tính **moveID**, mà là một số nguyên duy nhất được tạo ra từ tọa độ của ô xuất phát và ô đích của nước đi. Nếu hai đối tượng di chuyển có cùng moveID, tức là chúng đại diện cho cùng một nước đi trên bàn cờ, thì trả về True, ngược lại trả về False.

**c) Lớp GameState**

Class “GameState” là đối tượng đại diện cho trạng thái hiện tại của bàn cờ. Các thuộc tính:

**board**: Thuộc tính này lưu trữ trạng thái hiện tại của bàn cờ.

**whiteToMove**: Đây là một biến boolean (True hoặc False) dùng để chỉ ra đến lượt của quân trắng hay quân đen. Ban đầu, nó được thiết lập là True, cho biết lượt của quân trắng diễn ra trước.

**moveLog**: Đây là một danh sách được sử dụng để lưu trữ lịch sử các nước cờ đã di chuyển trong trò chơi. Khi một nước cờ được thực hiện, nó sẽ được thêm vào danh sách này để có thể hoàn tác nếu cần.

**whiteKingLocation** và **blackKingLocation**: Đây là biến để lưu trữ tọa độ của vua trắng và vua đen trên bàn cờ. Điều này cần thiết để kiểm tra tính chiếu cho vua.

**checkmate** và **stalemate**: Đây là biến boolean được sử dụng để chỉ ra trạng thái của trò chơi, nếu có chiếu hết (checkmate) hoặc hoà (stalemate). Ban đầu, cả hai biến đều được thiết lập là False.

**enpassantPossible:** Đây là biến dùng để chỉ ra tọa độ của ô vuông khi "en passant" (ăn tốt qua đường) là hợp lệ. Ban đầu, nó được thiết lập là None, vì ở trạng thái ban đầu, không có nước "en passant" nào có thể.

**enpassantPossibleLog**: Đây là danh sách được sử dụng để lưu trữ lịch sử các ô vuông đã từng được đánh "en passant" cho việc hoàn tác nước cờ. Ban đầu, danh sách này rỗng.

**currentCastlingRight**: Đây là một đối tượng của lớp CastleRights để biểu diễn quyền nhập thành hiện tại của cả hai bên. Đối tượng này chứa thông tin về quyền nhập thành cho từng bên và sẽ được cập nhật khi có nước cờ nhập thành được thực hiện.

**castleRightsLog**: Đây là danh sách được sử dụng để lưu trữ lịch sử quyền nhập thành của cả hai bên, cho việc hoàn tác nước cờ nếu cần.

Mô tả các hàm trong lớp GameState:

**Hàm makeMoved:** nhận vào một nước đi. Nó sẽ kiểm tra đặc điểm của nước đi và cập nhật lại vị trí cho các quân cờ.

## Hàm undoMoved: Được gọi đến để quay trở lại nước đi trước đó.

**Hàm squareUnderAttack:** nhận vào vị trí của một ô bao gồm hàng và cột. Nó sẽ kiểm tra xem ô đó có bị tấn công bởi đối phương hay không.

## Hàm inCheck: kiểm tra xem vua có đang bị chiếu hay không

## Hàm getAllossibleMoves: trả về danh sách bao gồm các nước đi có thể thực hiện được trong trạng thái hiện tại của bàn cờ. Để thực hiện được thì hàm cần gọi đến các hàm khác trả về nước đi hợp lệ của từng quân cờ, cụ thể: getPawnMoves: Cho quân tốt (pawn), getKnightMoves: Cho quân mã (knight), getBishopMoves: Cho quân tượng (bishop), getRookMoves: Cho quân xe (rook), getQueenMoves: Cho quân hậu (queen), getKingMoves: Cho quân vua (king).

**Hàm updateCastleRights:** sử dụng để cập nhật quyền nhập thành sau mỗi nước đi.

**Hàm getCastleMoves:** kiểm tra xem nước đi nhập thành có thể thực hiện không, nếu có sẽ thêm vào danh sách, sử dụng **getQueesideCastleMoves** để kiểm tra với xe phía bên hậu và **getKingsideCastleMoves** cho bên còn lại.

**Hàm getValidMoves:** trả về danh sách tất cả các nước đi hợp lệ có thể thực hiện được bao gồm các nước đi từhàm **getAllossibleMoves** và hàm **getCastleMoves**. Nếu một bên thực hiện nước đi mà làm vua của bên mình bị chiếu thì nước đi sau chắc chắn sẽ thua. Vì vậy đây cũng không được coi là nước đi hợp lệ. Hàm **getValidMoves** sẽ kiểm tra cả điều này và loại bỏ các nước đi. Đồng thời nếu số nước đi hợp lệ bằng 0, nó cũng sẽ kiểm tra để cập nhật lại trạng thái checkmate và stalemate. Nếu vua đang bị chiếu checkmate được gán True, nghĩa là bên đó bị thua. Nếu vua không bị chiếu thì stalemate chuyển thành True, trò chơi kết thúc với kết quả hòa.

**d) Hàm main**

Hàm **main** được sử dụng để chạy một ván cờ. Nó sẽ nhận vào giá trị biến singlePlayer. Khi biến này có giá trị True thì trò chơi sẽ ở chế độ chơi với máy. Ngược lại trò chơi ở chế độ 2 người chơi. Mô tả hàm **main**:

**screen = p.display.setmode((WIDTH, HEIGHT))**: Khởi tạo một màn hình trò chơi Pygame với kích thước được định nghĩa bởi hai biến WIDTH và HEIGHT. Đây là màn hình trò chơi mà trò chơi cờ vua sẽ được hiển thị.

**clock = p.time.Clock():** Khởi tạo một đối tượng đồng hồ để điều chỉnh tốc độ của trò chơi. Đối tượng này sẽ được sử dụng để kiểm soát số lượng khung hình trên mỗi giây, để trò chơi chạy với tốc độ cố định.

**gs = ChessEngine.GameState():** Khởi tạo một đối tượng GameState từ một lớp (class) ChessEngine. Đối tượng này biểu thị trạng thái ban đầu của trò chơi cờ vua.

**validMoves = gs.getValidMoves():** Gọi phương thức getValidMoves() của đối tượng gs để lấy danh sách các nước đi hợp lệ trong trạng thái ban đầu của trò chơi. Danh sách này sẽ được sử dụng để kiểm tra tính hợp lệ của các nước đi người chơi.

**moveMade:** Biến này được sử dụng để theo dõi xem nước đi hiện tại đã được thực hiện hay chưa. Ban đầu, không có nước đi nào được thực hiện, nên giá trị ban đầu là False.

**animate:** Biến này được sử dụng để kiểm soát xem có cần thực hiện hiệu ứng di chuyển cho nước đi hay không. Ban đầu, không có hiệu ứng nào được kích hoạt, nên giá trị ban đầu là False.

**loadImages():** Gọi một hàm loadImages() để nạp hình ảnh quân cờ và biểu đồ bàn cờ. Điều này sẽ cung cấp các hình ảnh cần thiết để vẽ quân cờ và bàn cờ lên màn hình. running = True: Biến này được sử dụng để kiểm soát việc trò chơi cờ vua đang chạy hay không. Ban đầu, trò chơi bắt đầu và biến này được đặt thành True.

**sqSelect:** Biến này lưu trữ tọa độ của ô trên bàn cờ mà người chơi đã chọn (nếu có). Ban đầu, chưa có ô nào được chọn, nên giá trị ban đầu là một tuple rỗng ().

**playerClicks:** Mảng lưu trữ vị trí ban đầu và vị trí muốn di chuyển tới của quân cờ trên bàn cờ. Mảng này có tối đa hai phần tử để lưu giá trị ban đầu và giá trị muốn di chuyển đến. Ban đầu, mảng này rỗng.

**gameOver = False:** Biến này được sử dụng để kiểm tra xem trò chơi đã kết thúc chưa. Ban đầu, trò chơi chưa kết thúc, nên giá trị ban đầu là False.

**PlayerOne** và **PlayerTwo:** lần lượt đại diện cho bên trắng và bên đen. Nếu giá trị bằng True thì là người còn bằng False thì là máy.

Vòng lặp **while running**: theo dõi sự kiện và trạng thái trò chơi. Biến **HumanTurn** được thêm vào để xác định đây là lượt của người hay máy. Vòng lặp sẽ thực hiện các công việc:

- Nếu người dùng thoát thì tắt chương trình. Nếu nguời dùng ấn phím “r” thì reset lại game. Nếu người dùng nhấn “z” thì quay lại trạng thái trước đó.

- Nếu đang trong lượt của người chơi **HumanTurn = True,** trò chơi chưa kết thúc **GameOver = False** đồng thời có các sự kiện click chuột thỏa mãn một nước (click vào 2 ô khác nhau) đi thì kiểm tra nước đi đó hợp lệ hay không. Nếu hợp lệ sẽ thực hiện nước đi, cập nhật nước đi đã được thực hiện **moveMade = True**.

- Nếu đang trong lượt của máy **HumanTurn = False** và trò chơi chưa kết thúc **GameOver = False** thì cho máy thực hiện nước đi và cập nhật nước đi đã được thực hiện **moveMade = True.**

- Nếu nước đi đã được thực hiện **moveMade = True** thì cập nhật lại danh sách các nước đi hợp lệ của trạng thái hiện tại **validMoves**, cập nhật lại **moveMade = False**

- Kiểm tra trạng thái của trò chơi. Nếu trò chơi đã kết thúc (**gs.checkmate** hoặc **gs.stalemate** có giá trị True) thì cập nhật gameOver = True, trò chơi dừng lại.

**e****) Các hàm khác**

**Hàm loadImage:** tải hình ảnh các quân cờ.

**Hàm animateMove:** Thêm hiệu ứng hình ảnh khi di chuyển quân cờ.

**Hàm highlightSquares:** làm nổi bật vị trí và các nước đi có thể thực hiện của quân cờ được chọn.

**Hàm drawGameBoard:** hiển thị bàn cờ.

**Hàm drawPieces:** hiển thị các quân cờ.

**Hàm drawText:** hiển thị nội dung văn bản được truyền vào lên màn hình game.

**Hàm drawGameState:** vẽ trạng thái hiện tại bao gồm bàn cờ , các quân cờ, quân cờ đang được chọn.

## 2.2 Xây dựng ChessBot cho game

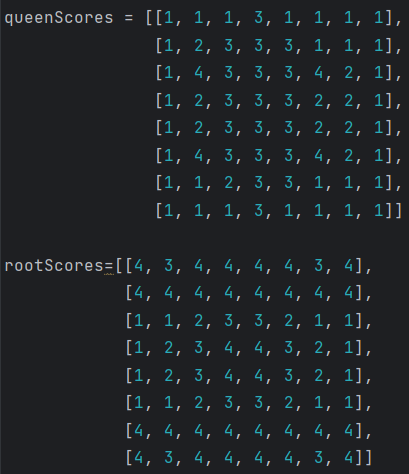
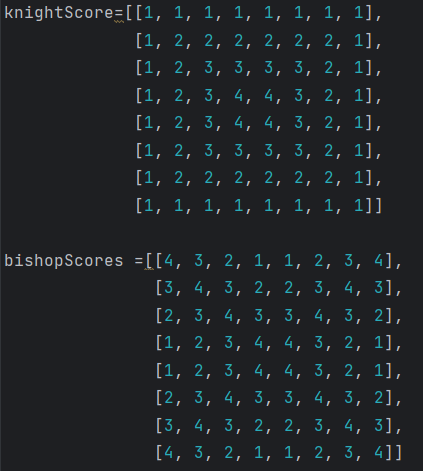
**a) Lượng giá bàn cờ**

Ta cần xây dựng một chức năng đánh giá để ChessBot có thể hiểu được trạng thái của bàn cờ hiện tại. Vì thế ta sẽ gán một điểm số cho từng trường hợp của bàn cờ, từ đó ChessBot có thể đưa ra quyết định về nước đi nào sẽ tốt hơn các nước đi khác.

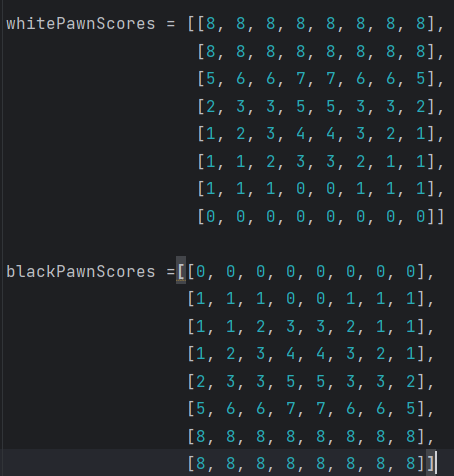
Khía cạnh đầu tiên ta xét đến là sức mạnh của từng quân cờ. Ta gán cho mỗi quân cờ giá trị theo bảng sau:

|  |  |
| --- | --- |
| Tốt | 10 |
| Mã | 30 |
| Tượng | 30 |
| Xe | 50 |
| Hậu | 90 |
| Vua | 900 |

Khía cạnh tiếp theo ta xét đến là vị trí cụ thể của một quân cờ. Ví dụ một con mã nằm ở giữa bàn cờ có thể thực hiện nhiều nước đi hơn đồng thời ngăn cản nước đi của quân đối thủ vào khu trung tâm; vì thế nó sẽ được xem là mạnh hơn chính nó khi ở góc bàn cờ. Các giá trị sau đây sẽ thể hiện giá trị cộng thêm của một quân cờ ở vị trí cụ thể:



Hình 2.3. Mã và Tượng Hình 2.4. Hậu và Xe



Hình 2.4. Tốt trắng và tốt đen

Đã có giá trị các quân cờ trên các vị trí cụ thể, tiếp theo, ta sẽ xây dựng hàm để đánh giá nước đi của ChessBot có tên **scoreBoard**. Ta sẽ truyền cho hàm trạng thái hiện tại của bàn cờ và hàm sẽ tính toán để trả về số điểm của trạng thái đó. Mặc định ban đầu số điểm là 0, khi gặp quân trắng điểm số sẽ được cộng thêm tương ứng với giá trị quân cờ và vị trí của nó, ngược lại khi gặp quân đen thì điểm số sẽ bị trừ đi. Điều này có nghĩa là giá trị này càng lớn thì bên trắng càng có lợi còn với bên đen thì giá trị này càng nhỏ càng tốt. Theo đó thì bất kì lợi thế nào do một bên tạo ra sẽ là thiệt hại cho bên còn lại.

Có một vài trường hợp đặc biệt khi thực hiện một nước đi, đối với những trường hợp này thì có thể trả về luôn giá trị:

- Chiếu hết: trả về giá trị 10000 cho bên vừa thực hiện chiếu hết.

- Hoà: trả về giá trị bằng 0.

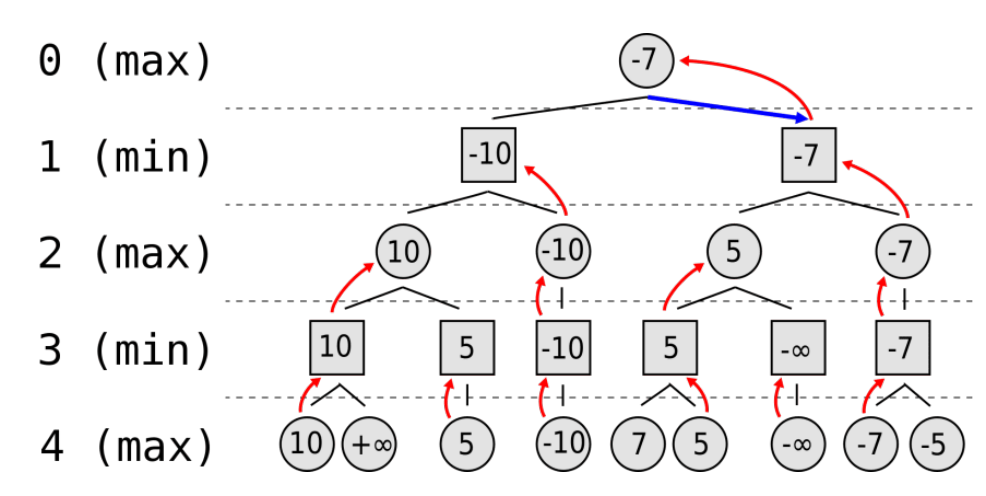
**b) Cây trò chơi**

Cây trò chơi (game tree) là một cách biểu diễn trạng thái và các lựa chọn có thể có trong một trò chơi. Trong trường hợp của cờ vua, cây trò chơi biểu diễn tất cả các nước đi có thể được thực hiện từ trạng thái hiện tại của bàn cờ. Mỗi Node trong cây trò chơi đại diện cho một trạng thái của bàn cờ, và các cạnh đại diện cho các nước đi có thể được thực hiện từ trạng thái đó. Từ trạng thái hiện tại, có thể có rất nhiều lựa chọn khác nhau về các nước đi, do đó cây trò chơi sẽ có rất nhiều nhánh. Các Node lá của cây trò chơi sẽ đại diện cho các trạng thái mà một bên đã chiến thắng hoặc là hòa. Mục tiêu của việc tìm kiếm cây trò chơi là tìm ra nước đi tốt nhất từ trạng thái hiện tại đến một trạng thái chiến thắng hoặc hòa.

**c) Thuật toán Minimax**

Nhờ có hàm đánh giá, ChessBot có thể dễ dàng so sánh và chọn ra nước đi tốt hơn. Tuy nhiên, khi xét đến các trường hợp phân nhánh trong tương lại ở cây trò chơi nêu trên, một nước đi có thể tốt ở hiện tại nhưng lại tạo ra cơ hội tấn công nguy hiểm cho đối thủ. Để xử lí vấn đề này, ChessBot cần tính cả đến những nước đi trong tương lai. Ta sẽ áp dụng thuật toán Minimax để thực hiện việc này.

Hai người chơi trong game được đại diện là MAX và MIN. MAX đại diện cho người chơi luôn muốn chiến thắng (quân trắng) và cố gắng tối ưu hóa ưu thế của mình còn MIN (quân đen) đại diện cho người chơi cố gắng cho người MAX giành số điểm càng thấp càng tốt. Giải thuật Minimax thể hiện bằng cách định trị các Node trên cây trò chơi: Node thuộc lớp MAX thì gán cho nó giá trị lớn nhất của con Node đó. Node thuộc lớp MIN thì gán cho nó giá trị nhỏ nhất của con Node đó. Từ các giá trị này người chơi sẽ lựa chọn cho mình nước đi tiếp theo hợp lý nhất.



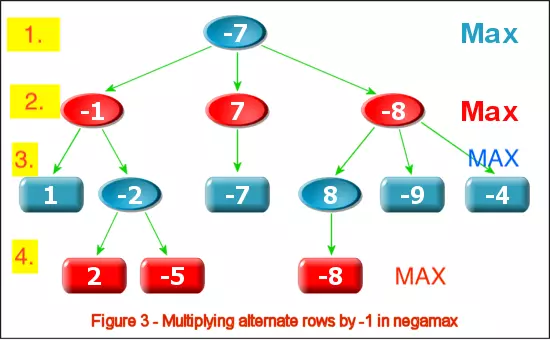
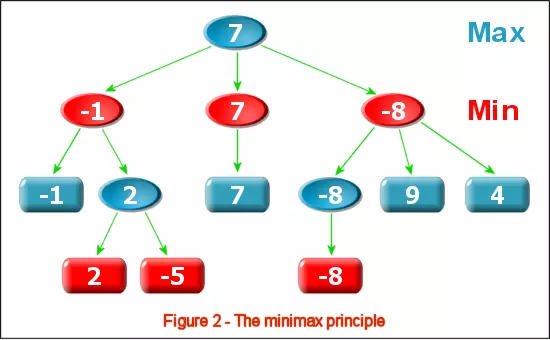
Hình 2.4. Ví dụ thuật toán Minimax

Trong thực tế, chúng ta chỉ có thể dự đoán trước một vài lượt - không khả thi về mặt tính toán để dự đoán trước các trạng thái thắng hoặc thua cuối cùng. Ta sẽ phải đưa ra giới hạn độ sâu tương ứng và sử dụng chức năng đánh giá để xác định mức độ thuận lợi của các trạng thái trò chơi khi cây trò chơi đạt đến giới hạn độ sâu.

NegaMax là một biến thể tối giản của thuật toán Minimax dựa theo nguyên tắc:

max(a, b) = -min(-a,-b).

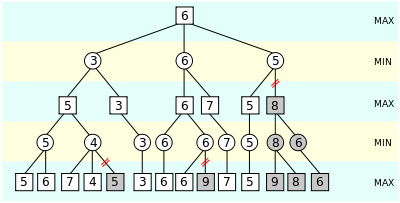
Node sẽ được gán giá trị là số đối của giá trị lớn nhất các Node con.



Hình 2.5. Ví dụ về Minimax và NegaMax

**d) Cắt tỉa Alpha Beta**

Không gian trạng thái của trò chơi cờ vua là rất lớn, vì thế việc thuật toán Minimax sinh ra cây trò chơi dựa vào tất cả các nước đi có thể tốn rất nhiều thời gian. Thuật toán cắt tỉa Alpha Beta sẽ giúp loại bỏ các nhánh không cần thiết của một đỉnh đang xét.



Hình 2.6. Ví dụ về cắt tỉa alpha-beta

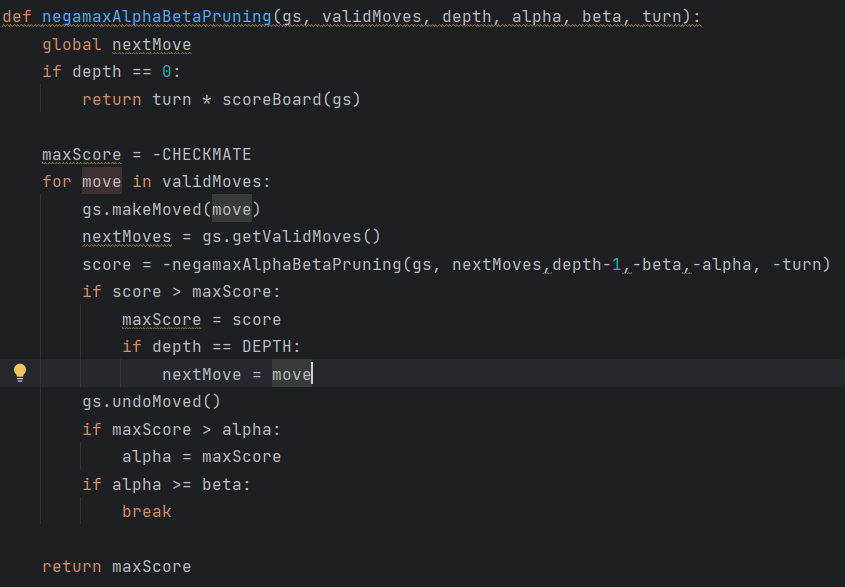
Hai tham số có thể được định nghĩa là:

* Alpha: Sự lựa chọn tốt nhất (giá trị cao nhất) cho đến thời điểm xét tại bất kỳ điểm nào trên con đường của MAX. Giá trị ban đầu của alpha là -∞.
* Beta: Lựa chọn tốt nhất (giá trị thấp nhất) cho đến thời điểm xét bất kỳ điểm nào dọc theo đường dẫn của MIN. Giá trị ban đầu của beta là +∞.

Khi giá trị Alpha >= Beta, thuật toán thực hiện cắt bỏ nhánh.

Nói cách khác, các trạng thái trò chơi của cây tìm kiếm được phân tích để tìm kiếm nước đi tối ưu cho người chơi MAX và MIN. Vì vậy, nếu giải pháp được tìm thấy ở một nút con bên dưới là tốt hơn so với giải pháp tại nút cha, nút cha sẽ chọn giải pháp tốt nhất từ giải pháp của nút con đó. Điều này được thực hiện bằng cách theo dõi giá trị tốt nhất đã tìm thấy cho người MAX hoặc người chơi MIN tại mỗi nút. Tuy nhiên, với Alpha-Beta pruning, một số nút không cần phân tích bởi vì chúng không ảnh hưởng đến giá trị tốt nhất đã tìm thấy của nút cha. Vì vậy, nếu ta đã tìm được một giá trị Beta mà bé hơn hoặc bằng giá trị Alpha, chúng ta sẽ không phải phân tích các nút con khác trong cây tìm kiếm, vì chúng ta sẽ không sử dụng kết quả của các nút con đó.

Trong chương trình ta sẽ cài đặt thuật toán Minimax dưới dạng NegaMax có áp dụng cắt tỉa AlphaBeta.

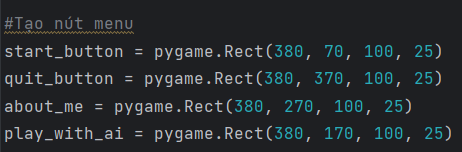


Hình 2.7. Cài đặt thuật toán Minimax trong chương trình

## 2.3 Xây dựng Menu

**a) Xây dựng giao diện Menu**

Định nghĩa các hình chữ nhật cho các nút trong menu trò chơi (x, y, chiều rộng, chiều cao). Các hình chữ nhật này sẽ được sử dụng để tương tác với menu.



Hình 2.8. Các nút chức năng

**Hàm Draw:** Thực hiện việc tạo front, màu sắc, vẽ các nút được đinh nghĩa ở trên.

**Hàm Background:** Vẽ cửa sổ game, tải hình nền lên, đặt tiêu đề trên cửa sổ.

**Hàm ShowInfo:** Hàm hiển thị văn bản lên màn hình game, phục vụ cho chức năng Info sẽ hiển thị tên thành viên và một số hướng dẫn khi chơi game.

**b) Hàm menu**

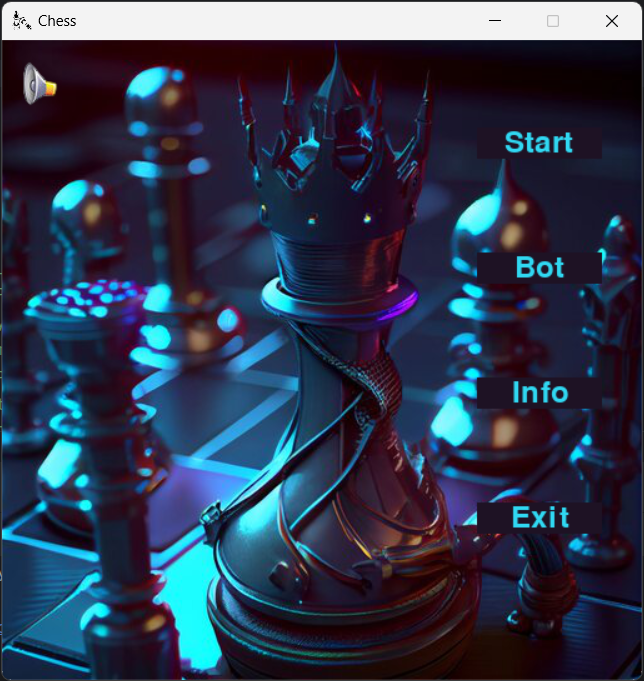
Hàm menu là hàm được gọi đến khi chạy chương trình. Tương tự với hàm main, hàm menu sẽ có nhiệm vụ quản lý các sự kiện, tương tác của người dùng với menu, đảm bảo các chức năng của menu được thực hiện. Cụ thể:

* Start: khởi động chế độ chơi hai người bằng cách gọi đến hàm Main với tham số singlePlayer = False.
* Bot: Khởi động chế độ chơi với máy người bằng cách gọi đến hàm Main với tham số singlePlayer = True.
* Info: Hiển thị thông tin các thành viên nhóm, một số phím chức năng của game bằng hàm ShowInfo.
* Exit: Thoát khỏi chương trình

# PHẦN 3: Chạy chương trình

## 3.1 Giao diện Menu

Giao diện menu với các chức năng đã trình bày ở trên.



Hình 3.1. Menu chương trình

## 3.2 Bắt đầu trò chơi

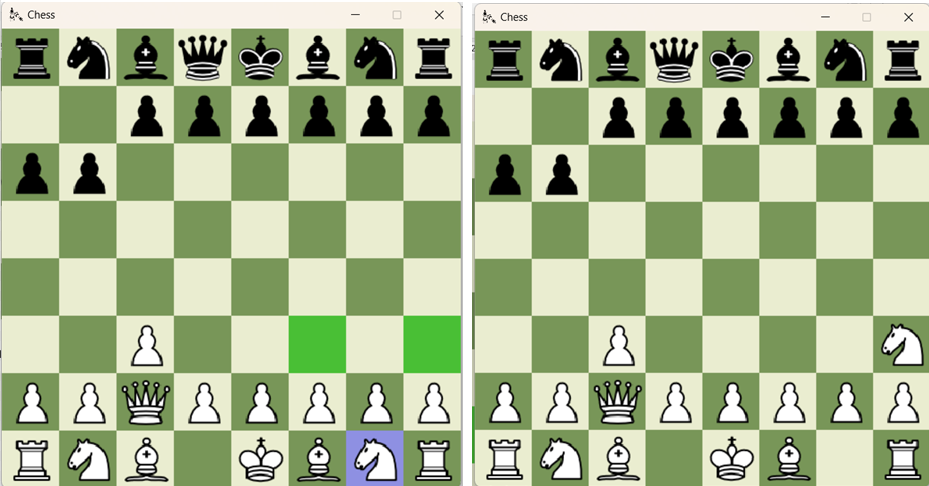
Trạng thái ban đầu của bàn cờ



Hình 3.2. Bàn cờ ở trạng thái bắt đầu

Chọn và di chuyển một quân cờ:

 Hình 3.3. Di chuyển quân tốt

 Hình 3.4. Di chuyển quân mã

Nước đi nhập thành:

*Hình 3.5. Nhập thành bên trái*

*Hình 3.6. Nhập thành bên phải*

Nước đi “Ăn tốt qua đường”:

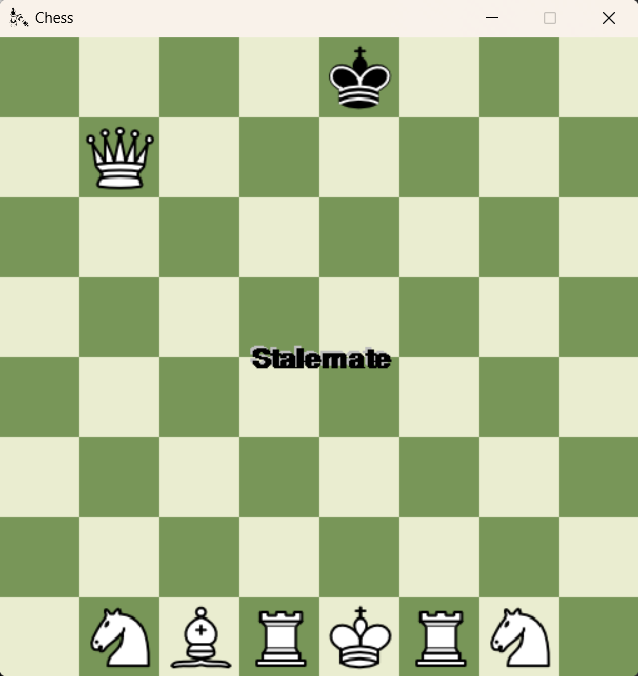
 *Hình 3.7. Ăn tốt qua đường*

Khi vua đang bị chiếu:

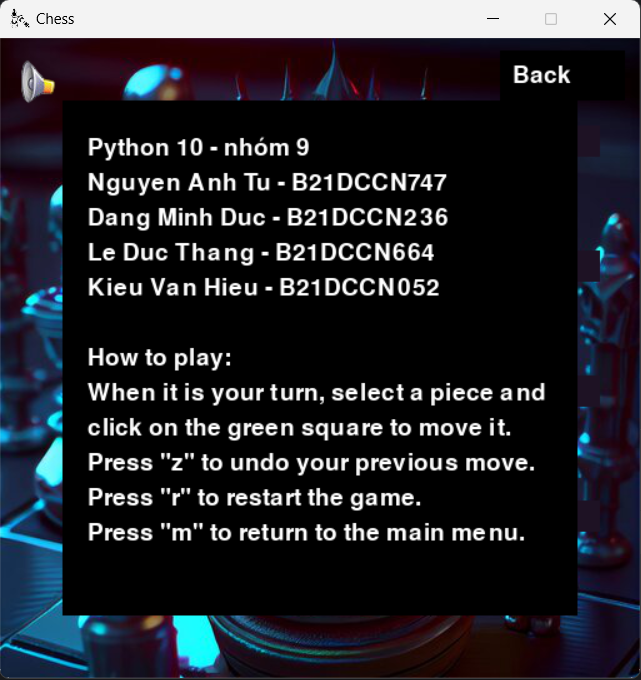


*Hình 3.8. Vua bị chiếu*

Khi trò chơi kết thúc:

  *Hình 3.9. Hòa do hết nước đi Hình 3.10. Bên trắng chiến thắng*

## 3.3 Nút Info



*Hình 3.10. Hướng dẫn chơi và thông tin nhóm*

## 

# PHẦN 4: Kết luận

## 4.1 Đánh giá

Về cơ bản nhóm đã thành công xây dựng một ứng dụng chơi game cờ vua đơn giản đáp ứng được các chức năng đặt ra ban đầu. Thông qua quá trình thực hiện bài tập lớn cũng như học tập trên lớp, nhóm đã củng cố được các kiến thức nền tảng của môn học lập trình Python đồng thời tích lũy được kinh nghiệm trong việc sử dụng Python và thư viện Pygame vào một dự án thực tế.

## 4.2 Một số tính năng để cải thiện ứng dụng

- Thêm chức năng đặt thế cờ trước khi bắt đầu ván đấu

- Cho phép lựa chọn trong chức năng phong cấp.

- Thêm giới hạn thời gian cho mỗi lượt

- Thêm các nút thực hiện chức năng như quay lại menu, chơi lại, quay lại nước đi … vào màn hình ván đấu.

- Cải thiện ChessBot: thay đổi giá trị đánh giá quân cờ, vị trí; tìm cách sắp xếp các nước đi để cắt tỉa hiệu quả hơn.

Trong quá trình thiết kế và thực hiện bài tập không tránh khỏi những sai sót, mong thầy thông cảm và có những góp ý giúp nhóm hoàn thiện hơn cho sản phẩm.

# Tài liệu tham khảo

# Dự án tham khảo: <https://www.youtube.com/playlist?list=PLBwF487qi8MGU81nDGaeNE1EnNEPYWKY>

# Algorithms Explained – minimax and alpha-beta pruning | Sebastian Lague.

1. <https://www.pygame.org/wiki/tutorials>