**Báo cáo tổng quan**

Phụ lục

[**I. Thông tin chung 1**](#_Toc6318)

[**II. Nội dung chuyên môn 1**](#_Toc3372)

[1. Lý do chọn đề tài: 1](#_Toc10116)

[2. Mô tả kiến thức sử dụng 2](#_Toc1207)

[3. Thuật toán 2](#_Toc27993)

[4. Thiết kế 3](#_Toc25587)

[5. Cách thức cài đặt 4](#_Toc6931)

[6. Đánh giá kết quả 5](#_Toc24470)

[7. Kết luận 6](#_Toc25685)

1. **Thông tin chung**
2. **Tên đề tài:** Cờ vua (Chess)
3. **Phân công công việc:**

* Phó Viết Tiến Anh - 22022568
* Xử lý nước đi, bàn cờ, tính toán nước đi
* Xử lý các luật trong cờ vua
* Nguyễn Xuân Hiệp - 22022591
* Giao diện người dùng, quản lý luồng trò chơi
* Xử lý hiển thị hình ảnh, hoạt ảnh di chuyển
* Nguyễn Phương Đông - 22022593
* Tìm kiếm và đề xuất các nước đi thông minh cho máy tính
* Sử dụng các thuật toán để chọn nước đi

1. **Nội dung chuyên môn**
2. **Lý do chọn đề tài:**

Lĩnh vực trí tuệ nhân tạo đang trở thành 1 vấn đề quan trọng trên thế giới. Sự hình thành và phát triển của trí tuệ nhân tạo đang trở thành xu hướng toàn cầu. Việc hiểu rõ trí tuệ máy tính suy nghĩ và hành động như thế nào không chỉ giúp cải thiện hiệu suất làm việc của các doanh nghiệp mà còn có thể mang lại những ảnh hương tích cực cho cộng đồng và xã hội.

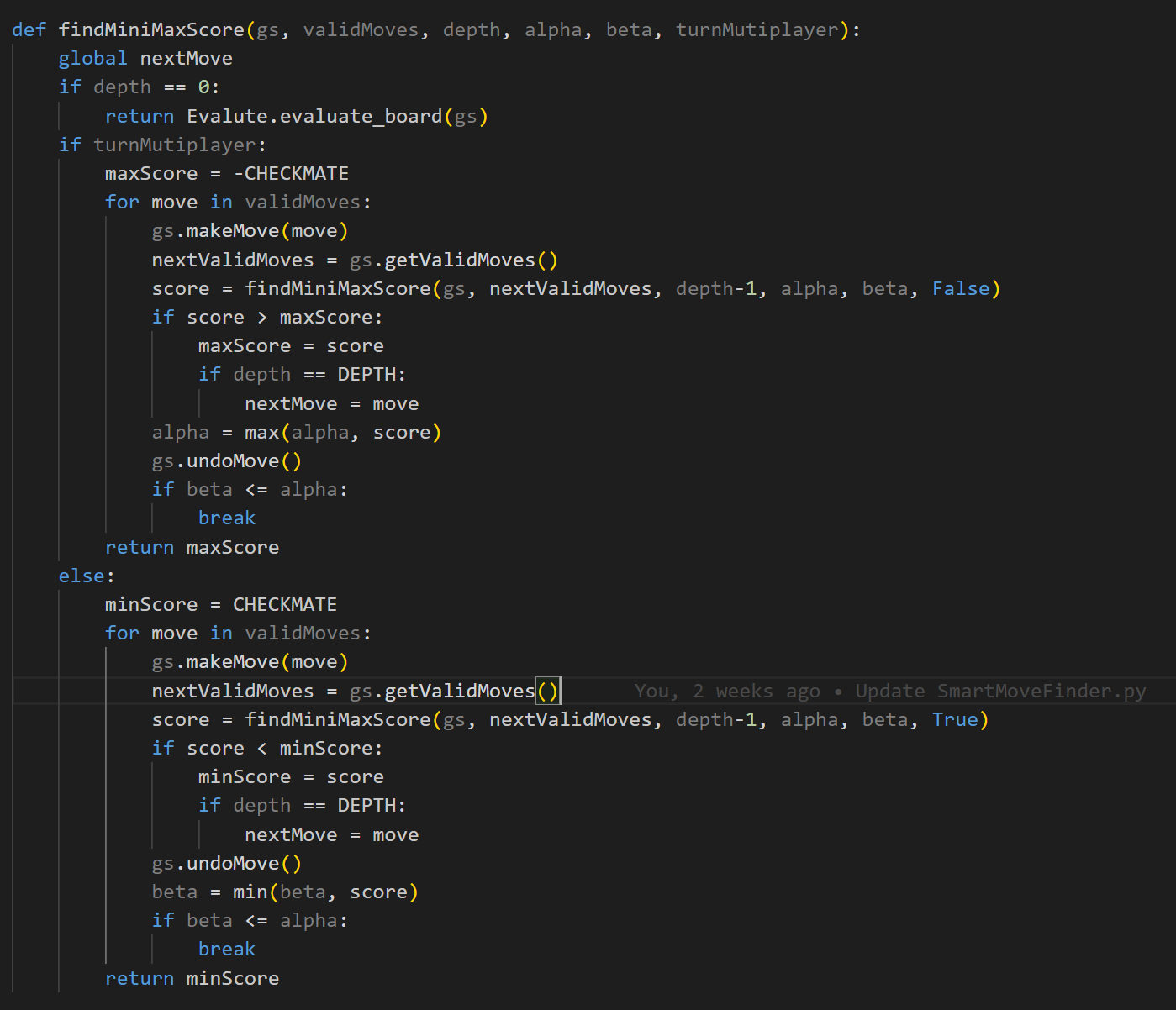
Cờ vua là trò chơi phức tạp nhưng phổ biến cho đến ngày nay, nó đòi hỏi sự suy luận, chiến lược và dự đoán. Trong trò chơi cờ vua được ứng dụng AI để giúp ta dễ tiếp cận, thân thiện với mỗi người, ta có thể dễ dàng đào tạo máy tính suy nghĩ và hành động một cách hợp lý, đánh giá được cách máy tính nghĩ và dự đoán nước đi. Cờ vua cung cấp một môi trường thử nghiệm lý tưởng cho các nhà nghiên cứu trong lĩnh vực trí tuệ nhân tạo. Nghiên cứu Trí tuệ Nhân tạo trong Cờ vua không chỉ giúp cải thiện hiệu suất của các chương trình chơi Cờ vua mà còn đóng góp vào sự phát triển của các phương pháp và công nghệ AI.

Tuy nhiên, việc áp dụng và đào tạo suy nghĩ máy tính trong trò chơi cờ vua vẫn có những bất cập và hạn chế.Cờ vua là một trò chơi có tính phức tạp cao với số lượng lớn các tình huống và biến thể có thể xảy ra. Điều này làm cho việc phát triển thuật toán AI có khả năng chơi cờ vua hiệu quả trở nên khó khăn. Thêm nữa một số tình huống có thể dẫn đến hàng loạt các tình huống tiếp theo, và dự đoán chính xác những diễn biến này đòi hỏi một lượng tài nguyên tính toán rộng lớn và khả năng dự đoán tương lai cao.

1. **Mô tả kiến thức sử dụng**

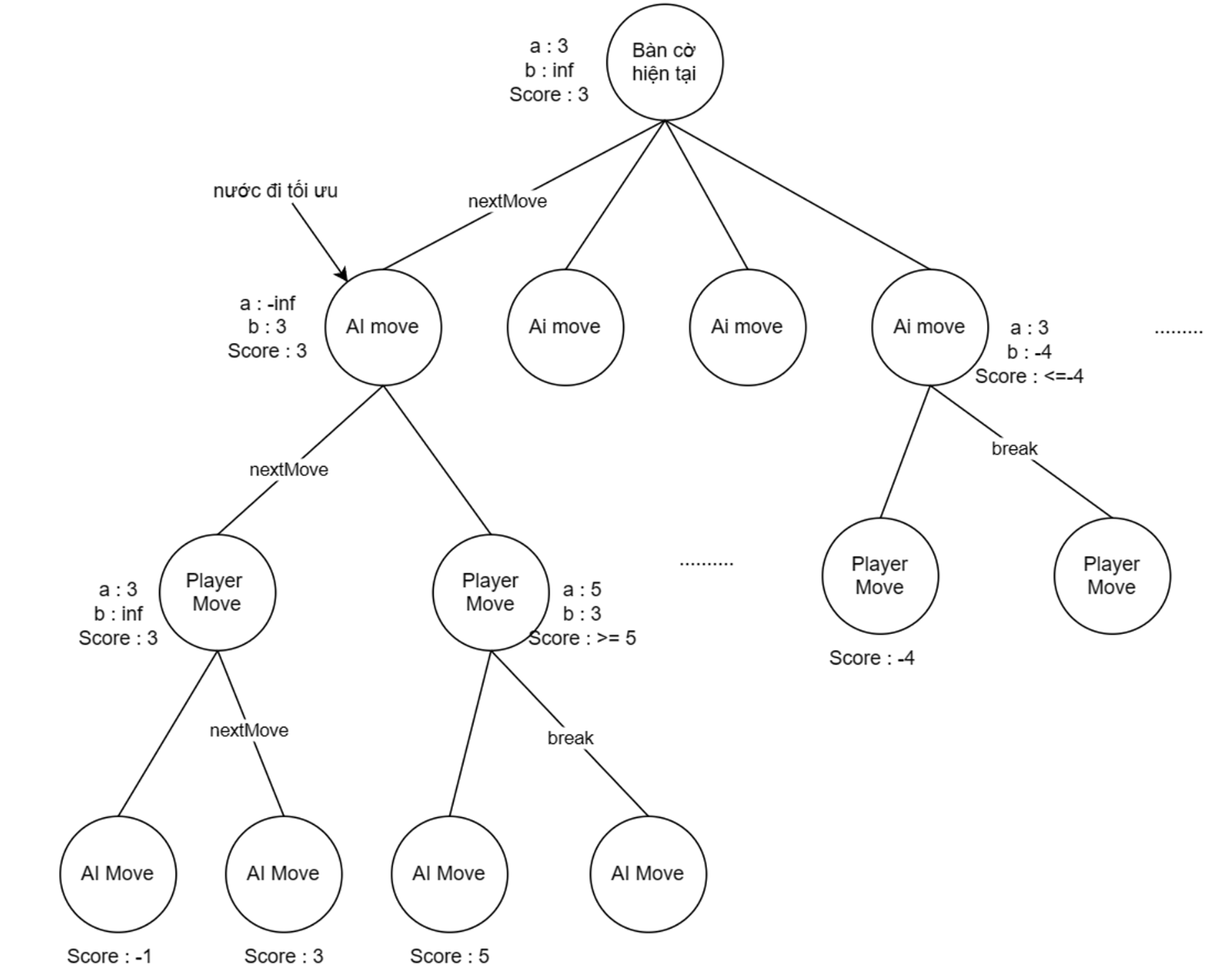
* Sử dụng cấu trúc dữ liệu như danh sách, từ điển để lưu trữ và quản lý thông tin về bàn cờ, nước đi, và trạng thái của trò chơi. Sử dụng các thuật toán để xác định nước đi hợp lệ, kiểm tra chiếu, chiếu hết, hòa cơ.
* Kiến thức về lập trình hướng đối tượng để đóng gói, tổ chức mã nguồn vào các lớp, phương thức, thuộc tính để tạo thành một hệ thống có cấu trúc và dễ bảo trì.
* Kiến thức về quy tắc và luật chơi của cờ vua được thể hiện trong việc xác định nước đi hợp lệ cho từng loại quân và các trạng thái của trò chơi như chiếu, chiếu hết, hòa cơ, cách theo dõi và cập nhật trạng thái của trò chơi như vị trí của quân cờ, lịch sử nước đi, trạng thái của vua, quyền nhập thành.
* Các thư viện hỗ trợ như **Pygame** để tạo của sổ vẽ đồ họa trò chơi, xử lý sự kiện bên người dùng như tương tác với bàn phím, chuột. Thư viện **random** sử dụng để tạo số ngẫu nhiên hoặc lựa chọn ngẫu nhiên từ danh sách các phần tử áp dụng cho việc tính toán nước đi của AI.
* Sử dụng các lớp từ các module khác trong dự án như class **Move** và **GameState** từ module **ChessEngine** và class **Evalute** từ module **Evalute**.
* Sử dụng ma trận đánh giá cho mỗi quân cờ, đánh giá giá trị của mỗi quân cờ dựa trên vị trí của chúng trên bàn cờ và giai đoạn của trò chơi (cận kề về cuối trò chơi). Tính toán điểm số của bàn cờ dựa trên giá trị của từng quân cờ và vị trí của chúng trên bàn cờ.
* Kiến thức về các thuật toán ứng dụng trong trí tuệ nhân tạo minimax, alpha-beta

1. **Thuật toán**

****

* **Minimax** :
* Hàm **findBestMinimaxMove(gs, validMoves)** là hàm để tìm nước đi tốt nhất cho máy tính. Trong hàm này, sử dụng biến toàn cục **nextMove** để có thể lưu trữ nước đi tiếp theo ở mọi hàm phục vụ cho hàm **findMiniMaxScore**, danh sách các nước đi hợp lệ được xáo trộn để tăng tính ngẫu nhiên, sau đó hàm **findMiniMaxScore** được gọi để tìm kiếm đệ quy nước đi tốt nhất.
* Trong **findMiniMaxScore(gs, validMoves, depth, alpha, beta, turnMutiplayer)**, thuật toán **Minimax với cắt tỉa Alpha-Beta** được thực hiện. Nó đánh giá trạng thái của bảng bằng cách sử dụng một phương pháp tìm kiếm đệ quy theo chiều sâu. Nếu đạt đến độ sâu cụ thể (**bằng 0**) hoặc một nút kết thúc, nó sẽ trả về giá trị đánh giá của trạng thái bảng bằng cách sử dụng hàm **Evalute.evaluate\_board(gs)-Đây là 1 hàm đánh giá chúng ta sẽ bàn ở phần dưới của báo cáo**.
* Biến **turnMutiplayer** là biến kiểm tra xem đang là lượt của ai **turnMutiplayer==True** Nếu là lượt của người chơi còn AI là **False**
* Thuật toán lặp qua các **validMoves** - Các nước đi hợp lệ trên bàn cờ . Sau đó nó thực hiện nước đi đó và tính điểm nó đạt được cho nước đi tiếp theo thông qua hàm **Evalute.evaluate\_board(gs) .** Sau đó nó tính toán xem điểm đạt được ở nước đi đó là tối ưu không **:**
* Nếu là AI nó cần cực đại hóa điểm số
* Nếu là người chơi nó cần cực tiểu hóa điểm số
* **Cắt tỉa Alpha-Beta**:
* **Alpha-Beta** là một kỹ thuật tối ưu hóa được sử dụng trong **Minimax** để giảm bớt số lượng nút cần xem xét trong quá trình tìm kiếm. Kỹ thuật này hoạt động bằng cách sử dụng hai tham số là alpha và beta để giới hạn phạm vi các giá trị có thể đạt được trong quá trình tìm kiếm.
* Thuật toán cũng đồng thời duy trì 2 giá trị **alpha** **beta** để cắt tỉa cây
* Nếu là người chơi nó cần cực đại hóa điểm số **alpha** với điểm AI
* Nếu là AI nó cần cực tiểu hóa điểm số **beta** với người chơi
* nếu **alpha** lớn hơn hoặc bằng **beta** nó sẽ cắt tỉa nhánh đó

**Dưới đây là ví dụ về cách thuật toán hoạt động:**

****

* **Hàm đánh giá**
* Có 2 đầu điểm chúng ta cần chú ý là: **ĐIểm số của các quân** **cờ** và **Điểm vị trí các quân cờ:**
* **ĐIểm số các quân cờ** : Mỗi quân cờ được gán giá trị nhất đính tùy thuộc vào sức mạnh và mức độ quan trọng của từng con. Càng quan trọng thì điểm càng cao
* **Điểm số cho vị trí từng quân trên bàn cờ** : được tính bằng hàm **evaluate\_piece(piece, square, location, end\_game)** .Mỗi quân cờ đều có 1 list mảng 2 chiều tương ứng với bàn cờ chứa các điểm ở các vị trí khác nhau của 1 quân cờ
* Hàm dánh giá sẽ duyệt qua toàn bộ bàn cờ và tính tổng **ĐIểm số các quân cờ + Điểm số cho vị trí từng quân trên bàn cờ.** Nếu là quân trắng thì cộng điểm, quân đen thì trừ điểm. Điểu này cho thấy điểm số cao thì càng có lợi cho quân trắng và ngược lại
* **Một số cài đặt khác**

Để tăng sự thông minh cho AI thì chúng ta cần tăng độ sâu thuật toán Minimax nhưng đồng thời điều này sẽ tốn rất nhiều tài nguyên và thời gian của máy tính vì nó sẽ phải duyệt qua nhiều trường hợp hơn. Chính vì vậy tôi sẽ để độ sâu bằng 3 ở các giai đoạn đầu của ván cờ và tăng độ sâu lên 4 ở giai đoạn sau của ván cờ (Tức là khi số quân cờ ít dần đi và việc tính toán và duyệt nước đi trở nên ít phức tạp hơn) bằng cách sử dụng hàm **Evalute.check\_mid\_game(gs)** khi số lượng quân cờ nhỏ hơn 20 nó sẽ trả về **True** và tăng độ sâu lên 4.

1. **Thiết kế**

**Mục tiêu:**

* Mục tiêu của dự án là xây dựng một ứng dụng cờ vua giải trí, cho phép người chơi tham gia trò chơi với nhau. Ứng dụng sẽ cung cấp một giao diện người dùng thân thiện, cho phép người dùng dễ dàng thực hiện các nước đi và tương tác với bảng cờ. Phạm vi của dự án bao gồm việc phát triển các tính năng như chơi chế độ 1 người và 2 người chơi hoặc có thể xem 2 máy chơi.

**Kiến trúc tổng quan:**

* Giao diện người dùng và thao tác của người dùng: được tổ chức trong file **ChessMain.py**. File này chịu trách nhiệm hiển bị bàn cờ, các quân cờ, giao diện tương tác cho người dùng.
* Logic trò chơi: file **ChessEngine.py** chịu trách nhiệm cho việc điều khiển luồng trò chơi, xác định các quy tắc, các luật, các ràng buộc của trò chơi cờ vua. Quản lý trạng thái của bảng cờ, quân cờ, lượt chơi. Kiểm tra các điều kiện kết thúc trò chơi như **checkmate**, hòa.
* Sử dụng thuật toán để ứng dụng AI: file **SmartMoveFinder.py** chịu trách nhiệm cho suy nghĩ của máy tính, sử dụng các thuật toán như Minimax, Alpha-Beta Pruning để đánh giá các nước đi và chọn nước đi tốt nhất cho máy tính, đưa ra nước đi hợp lý và tối ưu.

**Thiết Kế Giao diện Người dùng:**

* Bảng cờ vua được hiển thị trên màn hình với kích thước 8x8 ô
* Mỗi ô trên bàn cờ được biểu diễn bằng một hình vuông có kích thước phù hợp
* Các quân cờ được hiển thị trên bảng cờ tại vị trí tương ứng.
* Mỗi loại quân cờ được biểu diễn bằng một biểu tượng hình ảnh tương ứng, như quân tốt, quân mã, quân tượng,…
* Khi người dùng nhấp chuột vào một ô trên bảng cờ, ô đó sẽ được chọn và hiển thị bằng màu sắc khác biệt để chỉ ra rằng người dùng đã chọn nó.
* Sau khi chọn một ô, người dùng có thể nhấp chuột vào ô khác để chọn nước đi hoặc nhấn phím để thực hiện các hành động như undo là phím z hoặc reset là phím r.
* Khi người dùng thực hiện một nước đi, các quân cờ sẽ di chuyển trên bảng cờ theo hình thức hoạt ảnh, giúp người dùng theo dõi các bước di chuyển một cách dễ dàng.

**Logic:**

* Đánh giá vị trí của các quân cờ: Mỗi vị trí của quân cờ được đánh giá dựa trên các yếu tố như giá trị tương ứng với loại quân, vị trí trên bàn cờ, hoạt động của quân cờ, và các yếu tố chiến thuật như kiểm soát bàn cờ và phòng thủ. Điều này quyết định nên di chuyển quân cờ như thế nào trong mỗi tình huống.
* Kiểm tra trạng thái trò chơi: kiểm tra trạng thái của trò chơi, bao gồm chiếu mạng (checkmate) và hòa cơ (stalemate). Chiếu mạng xảy ra khi vua của một người chơi bị chiếu và không có nước đi nào để thoát khỏi chiếu. Hòa cơ xảy ra khi không có nước đi hợp lệ nào và trò chơi kết thúc với kết quả hòa hoặc hòa cờ theo luật.

1. **Cách thức cài đặt**

**Thuộc tính:**

* **board**: Mảng 2D biểu diễn bàn cờ.
* **moveFunctions**: Từ điển ánh xạ các loại quân cờ tới hàm xác định các nước đi hợp lệ của chúng.
* Các thuộc tính khác như **whiteToMove, moveLog, whiteKingLocation, blackKingLocation, castleRightsLog...**
* **Hàm makeMove**: Thực hiện một nước đi trên bàn cờ và cập nhật trạng thái.
* **Hàm undoMove**: Hoàn tác nước đi trước đó.

**Các hàm xác định nước đi hợp lệ cho từng loại quân cờ:**

* **getPawnMoves, getRookMoves, getKnightMoves, getBishopMoves, getQueenMoves, getKingMoves**.

**Các hàm hỗ trợ:**

* **checkForPinsAndChecks**: Kiểm tra xem có quân bị ghim hoặc chiếu không.
* **updateCastleRights**: Cập nhật quyền nhập thành sau mỗi nước đi.
* **squareUnderAttack**: Kiểm tra ô trên bàn cờ có bị tấn công không.
* **getAllPossibleMoves**: Lấy tất cả các nước đi có thể của người chơi hiện tại.
* **incheck**: Kiểm tra xem trò chơi có trong tình trạng hòa cơ không.
* Import thư viện và thiết lập các hằng số như kích thước cửa sổ, kích thước ô trên bàn cờ, số lần lặp trên mỗi giây để cập nhật trạng thái trò chơi.
* Khởi tạo biến và tải hình ảnh của các quân cờ từ thư mục "**images**".
* Hàm main() quản lý giao diện người dùng và trạng thái trò chơi bao gồm khởi tạo cửa sổ, xử lý sự kiện chuột và bàn phím, cập nhật và vẽ trạng thái trò chơi lên cửa sổ.
* Các hàm vẽ đồ họa bao gồm vẽ bàn cờ, vẽ quân cờ, vẽ chữ cái và số, tô màu ô được chọn và ô đích của một nước đi hợp lệ, vẽ bảng chọn quân khi tốt đến cuối bàn cờ, và hoạt ảnh di chuyển của quân cờ.
* Hàm xử lý sự kiện chuột và bàn phím để xác định vị trí chuột được nhấp, xử lý lựa chọn quân cờ và nước đi, hoàn tác nước đi và khởi tạo lại trò chơi.
* Hàm **drawButton()** và **drawButtonDown()** để vẽ các nút điều khiển trò chơi như "AI Play", "One Player", "Two Player", "White", "Black".
* Hàm **findRandomMove(validMoves**): Trả về một nước đi ngẫu nhiên từ danh sách các nước đi hợp lệ.
* Hàm **findBestMinimaxMove(gs, validMoves)**: Sử dụng **minimax** và **alpha-beta pruning** để tìm nước đi tốt nhất cho máy tính.
* Hàm **findMiniMaxScore(gs, validMoves, depth, alpha, beta, turnMutiplayer)**: Đây là hàm đệ quy để tính toán giá trị **minimax**.

1. **Đánh giá kết quả**

* Code được tổ chức thành các hàm và phương thức riêng biệt để thực hiện các nhiệm vụ cụ thể, giúp dễ dàng quản lý và hiểu code hơn.
* Có sử dụng các comment để giải thích ý nghĩa của từng phần trong code, giúp người đọc hiểu rõ hơn về cách hoạt động của từng phương thức.
* Có khả năng xử lý nhiều tình huống khác nhau trong trò chơi cờ vua như nhập thành, bắt quân qua đường, phong hậu, và các quy tắc khác. Ngoài ra còn có giao diện bắt mắt, dễ dàng tương tác. AI thân thiện, tính toán nước đi hợp lý, thông minh.
* Do được tính đóng gói đóng gói nên dự án có tính linh hoạt, dễ bảo trì, dễ mở rộng và cải tiến thêm chức năng hoặc các thuật toán.

1. **Kết luận**

* Việc phát triển giao diện người dùng thân thiện và dễ sử dụng cũng là một kỹ năng quan trọng. Có thể học được cách tạo ra trải nghiệm người dùng tốt thông qua việc phản hồi từ người dùng và thử nghiệm.
* Phát triển một trò chơi cờ vua không chỉ là về việc thiết kế giao diện đẹp mắt. Ngoài ra cần xử lý logic trò chơi, kiểm soát luồng, và tối ưu hóa hiệu suất. Việc này sẽ giúp cải thiện hơn trong các ngôn ngữ lập trình và cả trong việc quản lý dự án phức tạp.
* Bằng cách lập trình AI để chơi cờ vua, ta sẽ phải nắm vững các thuật toán như Minimax, Alpha-Beta Pruning, hoặc Reinforcement Learning. Qua quá trình này sẽ hiểu được cách AI suy luận và ra quyết định trong môi trường có tính chất lớn và phức tạp.
* Hơn nữa, phát triển một trò chơi cờ vua với AI cho phép thử nghiệm và cải tiến các phương pháp trí tuệ nhân tạo. Ta có thể thử nghiệm các biến thể của thuật toán hoặc thử nghiệm các phương pháp học tăng cường để cải thiện hiệu suất của AI.