TRƯỜNG ĐẠI HỌC TÀI CHÍNH – MARKETING

KHOA KHOA HỌC DỮ LIỆU

ĐỒ ÁN MÔN HỌC

LẬP TRÌNH JAVA

Đề tài:

GAME CỜ VUA

Giảng viên hướng dẫn: Ths. Nguyễn Thanh Trường

Sinh viên thực hiện 1: Nguyễn Hoài Việt Chương

Lớp học phần: 2511101058502

TP.HCM, ngày 14 tháng 4 năm 2025

TRƯỜNG ĐẠI HỌC TÀI CHÍNH – MARKETING

KHOA KHOA HỌC DỮ LIỆU

ĐỒ ÁN MÔN HỌC

LẬP TRÌNH JAVA

Đề tài:

GAME CỜ VUA

Giảng viên hướng dẫn: Ths. Nguyễn Thanh Trường

Sinh viên thực hiện 1: Nguyễn Hoài Việt Chương

Mssv thực hiện 1: 2221004148

Lớp học phần: 2511101058502

TP.HCM, ngày14 tháng 4 năm 2025

LỜI CẢM ƠN

Đồ án: “Chương trình cờ vua” cơ bản đã được hoàn thành. Dù việc thực hiện bước đầu còn nhiều khó khăn, thách thức nhưng bằng sự kiên trì, nỗ lực không ngừng của bản thân cùng những kiến thức được tiếp thu từ trường lớp đã giúp chúng em xây dựng, phát triển một trò chơi đầy hấp dẫn. Trong quá trình trau dồi, phân tích, nghiên cứu và hoàn thiện, chúng em đã nhận được rất nhiều sự hướng dẫn, giúp sức nhiệt tình từ thầy giảng viên, bạn bè cùng các anh chị khóa trên

Chúng em xin chân thành cảm ơn và tri ân sâu sắc đến quý thầy cô Trường Đại học Tài Chính – Marketing , đặc biệt là thầy Nguyễn Thanh Trường đã dạy dỗ tận tình cho chúng em những tháng ngày vừa qua, cô luôn quan tâm và hướng dẫn chúng em từng ý nhỏ

Với vốn kinh nghiệm còn rất khiêm tốn, ít ỏi và đây cũng là bước đầu chúng em làm quen với công việc sáng tạo, thiết kế ra một chương trình game thì chắc rằng kết quả thu hoạch được cũng không tránh khỏi những hạn chế. Em rất mong nhận được những lời góp ý từ thầy giảng viên, những bạn sinh viên đi trước hay bất kỳ độc giả nào quan tâm để chúng em có thể hoàn thiện hơn cho các đồ án cũng như nghiên cứu sau này mình.

Chúng em xin kính chúc thầy Nguyễn Thanh Trường cùng tất cả những người đã hỗ trợ, đồng hành cùng em trên chặng đường hoàn thiện bài đồ án này có thật nhiều sức khỏe, niềm vui, niềm hân hoan và gặt hái thật nhiều thành công trong cuộc sống. Chúng em tin đây mới chỉ là bước khởi đầu để trường đại học Tài chính-Marketing của chúng ta nói chung cùng chính bản thân chúng em nói riêng có thể giúp nâng tầm vị thế của ngành Hệ thống thông tin quản lý trong thời đại mới và xa hơn nữa là mục tiêu xây dựng đất nước ngày càng tiến bộ, vững mạnh..

NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Nhận xét của giảng viên phản biện (chấm 1)	**Nhận xét của giảng viên phản biện (chấm 2)**
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
Điểm số:	Điểm số: .....
Điểm chữ:	Điểm chữ: .....
Ngày…/…/202…	Ngày…/…/202…
Ký tên (ghi rõ họ tên)	*Ký tên (ghi rõ họ tên)*

MỤC LỤC

DANH MỤC BẢNG, BIỂU ĐỒ

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

A.I -	Artificial Intelligence
IDE	Integrated Development Environment
TH	Trường hợp

DANH MỤC THUẬT NGỮ ANH - VIỆT

Integrated Development Environment	Môi trường phát triển tích hợp
Castling	Hậu vị
En passant	Đi qua
Pawn	Tốt
Knight	Mã
Bishop	Tịnh (hoặc Tượng)
Rook	Xe
Queen	Hậu
King	Vua

CHƯƠNG 1 TỔNG QUAN ĐỀ TÀI

1.1 Lý do hình thành đề tài

Cờ vua là một trò chơi trí tuệ cổ điển có lịch sử lâu đời dành cho hai người chơi, với mục tiêu chính là đặt "vua" của đối phương vào thế không thể di chuyển mà không bị bắt, gọi là "chiếu bí." Trò chơi được chơi trên một bàn cờ 8x8 ô vuông, với tổng cộng 64 ô xen kẽ màu sáng và tối.

Cờ vua không chỉ mang tính chất giải trí mà còn rèn luyện tư duy chiến thuật, sự kiên nhẫn và khả năng ra quyết định. Đây là một trò chơi không biên giới, được yêu thích trên khắp thế giới bởi cả người mới bắt đầu lẫn các đại kiện tướng cờ vua chuyên nghiệp.

Trong thế kỷ 21 ngày nay, lĩnh vực Công nghệ thông tin đang phát triển mạnh mẽ và mang đến sự ảnh hưởng lớn tới mọi khía cạnh của cuộc sống, xã hội, đặc biệt là trong lĩnh vực giải trí. Với sự phát triển nhanh chóng của thị trường đòi hỏi quy mô lớn, việc thiết kế một trò chơi điện tử đã trở nên phức tạp và mang đến nhiều thách thức khó khăn. Công nghệ thông tin phát triển giúp mang lại nhiều công cụ, phương tiện, phần mềm để thiết kế các tính năng, giao diện, đồ họa cho trò chơi điện tử.

Đặc biệt với sự loan tỏa của cơn sốt trí tuệ nhân tạo AI (Artificial Intelligence), việc lựa chọn một trong những trò chơi chiến thuật cổ xưa nhất của là người là một quyết định khá rõ ràng. Vì thế em quyết định lựa chọn đề tài ‘Chương trình cờ vua’ để trau dồi kinh nghiệm, và thử thách chính mình.

1.2 Giới thiệu sơ lược về cờ vua

Cờ vua có một lịch sử lâu dài và nhiều giai thoại. Trò chơi đã thay đổi khá nhiều so với hình thức ban đầu của nó ở Ấn Độ. Phiên bản hiện đại mà chúng ta thưởng thức ngày nay không được biết đến cho đến thế kỷ 16. Không có đồng hồ và các quân cờ không được chuẩn hóa cho đến thế kỷ 19.

Danh hiệu vô địch thế giới chính thức ra đời vào cuối thế kỷ 19, ngay sau khi các giải đấu lớn đầu tiên được tổ chức và nhiều phong cách chơi đã bắt đầu phát triển đầy đủ. Mặc dù cuốn sách đầu tiên về khai cuộc đã được xuất bản vào đầu năm 1843, nhưng lý thuyết mà chúng ta biết vẫn chưa thực sự phát triển cho đến đầu/giữa thế kỷ 20. Các công cụ máy tính và cơ sở dữ liệu không được đưa vào sử dụng cho đến tận cuối thế kỷ 20..

- Nguồn gốc cờ vua:

Cờ vua, như chúng ta biết ngày nay, ra đời từ trò chơi chaturanga của Ấn Độ trước những năm 600 sau Công nguyên. Trò chơi này lan rộng khắp Châu Á và Châu Âu trong những thế kỷ tiếp theo, và cuối cùng phát triển thành trò chơi mà chúng ta biết đến là cờ vua vào khoảng thế kỷ 16. Một trong những bậc thầy đầu tiên của trò chơi này là một linh mục người Tây Ban Nha tên là Ruy Lopez. Mặc dù ông không phát minh ra khai cuộc mang tên mình, nhưng ông đã phân tích nó trong một cuốn sách mà ông xuất bản năm 1561. Lý thuyết cờ vua khi đó còn rất thô sơ.

1.3 Mục tiêu và nội dung nghiên cứu

1.3.1 Mục tiêu đề tài:

Xây dựng một chương trình cờ vua hoàn chỉnh, bao gồm các yếu tố cơ bản như nhân vật, môi trường, gameplay, âm thanh, hình ảnh.

Phân tích và đánh giá các thuật toán, chiến thuật trước đây để học hỏi kinh nghiệm.

Áp dụng các kiến thức về lập trình, thiết kế, đồ họa để tạo ra một sản phẩm chất lượng cao.

1.3.2 Nội dung nghiên cứu:

Tìm hiểu về lịch sử và cách chơi của cờ vua.

Triển khai các thuật toán cần thiết cho các quân cờ.

Thiết kế giao diện đơn giản, dễ hiểu.

Triển khai thuật toán Min-Max với sự hỗ trợ của thuật toán tìm kiếm Alpha-Beta

1.4 Đối tượng và phạm vi đề tài

1.4.1 Đối tượng nghiên cứu

Chương trình cờ vua

1.4.2 Phạm vi đề tài

- Cách thức hoạt động của của cờ vua
- Quân cờ
- Bàn cờ
- Hình ảnh

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 Thuật toán Min Max, cắt tỉa Alpha-Beta

Trong các trò chơi hai người, hai đối thủ thường được gọi là MIN và MAX, luân phiên thay thế nhau thực hiện lượt đi. Người chơi MAX đại diện cho mục tiêu tối đa hóa lợi thế của mình, tức là cố gắng đạt điểm số cao nhất có thể, trong khi người chơi MIN lại cố gắng giảm điểm số của MAX hoặc làm cho lợi thế của MAX bị giới hạn, nhằm đạt kết quả tốt nhất cho mình. Giả định rằng MIN và MAX có cùng mức độ kiến thức về không gian trạng thái của trò chơi và cả hai đều cố gắng tối ưu hóa nước đi của mình.

Mỗi Node trên cây trò chơi biểu diễn một trạng thái của trò chơi. Các Node lá là những trạng thái kết thúc, nơi trò chơi đã đạt đến kết quả cuối cùng. Giải thuật Minimax được dùng để định trị các Node này nhằm tìm nước đi tối ưu nhất:

Đối với Node thuộc lớp MAX, nó sẽ nhận giá trị lớn nhất từ các con của nó (các trạng thái tiếp theo có thể xảy ra).

Đối với Node thuộc lớp MIN, nó sẽ nhận giá trị nhỏ nhất từ các con của nó.

Dựa trên các giá trị này, người chơi sẽ chọn nước đi tiếp theo hợp lý nhất để tối ưu hóa kết quả.

2.1.1 Về thuật toán Minimax:

Minimax quét qua toàn bộ các trạng thái có thể để tìm trạng thái có lợi nhất. Tuy nhiên, nhược điểm lớn của nó là thời gian xử lý rất lâu, đặc biệt đối với trò chơi phức tạp như cờ vua, nơi không gian trạng thái rất lớn (bàn cờ vua có 64 ô và hàng triệu khả năng nước đi).

Kỹ thuật cắt tỉa Alpha-beta:

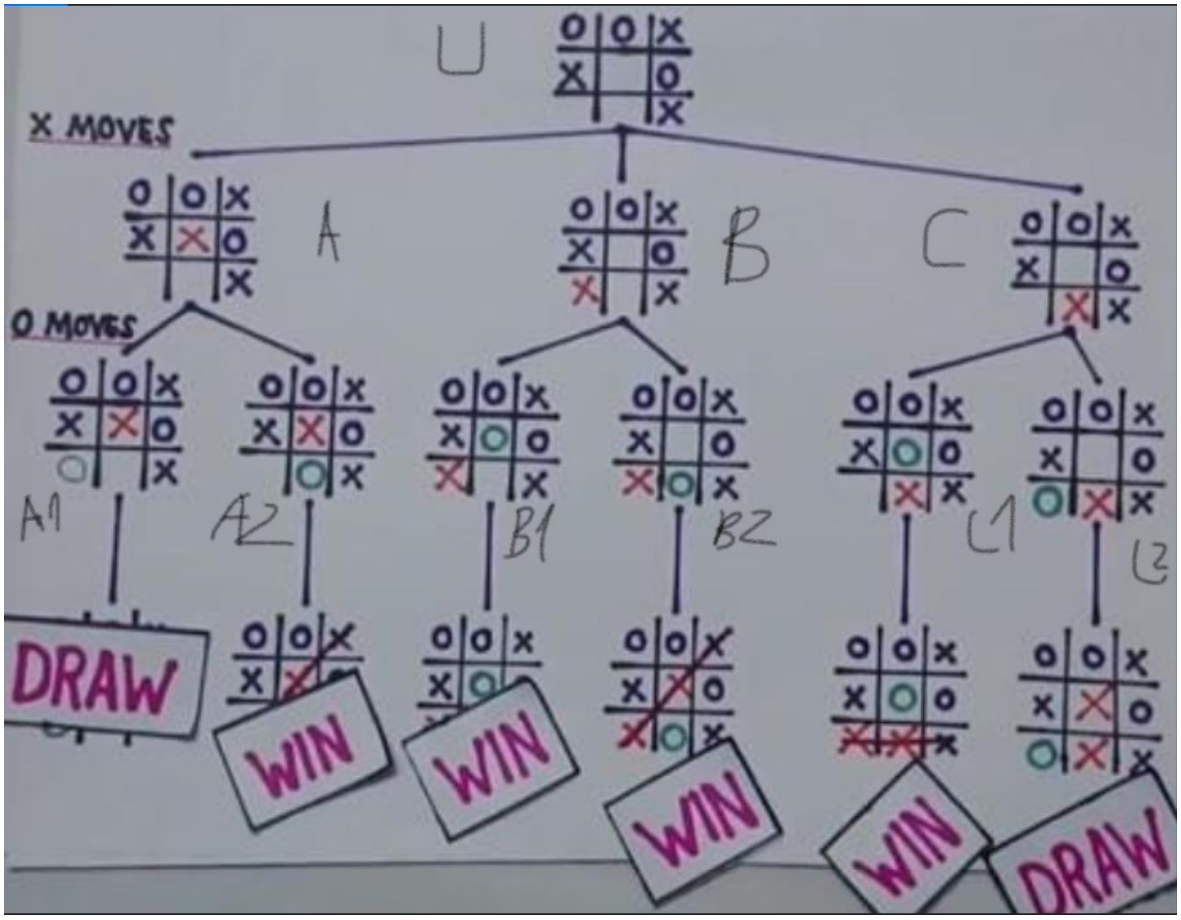
Để khắc phục nhược điểm của Minimax, kỹ thuật cắt tỉa Alpha-beta được áp dụng nhằm giảm số lượng trạng thái cần xét:

Thay vì duyệt toàn bộ không gian trạng thái, Alpha-beta chỉ xét các nhánh cần thiết và loại bỏ những nhánh không có tiềm năng cải thiện kết quả.

Khi đạt độ sâu tính toán nhất định hoặc xác định rằng một nhánh không hữu ích (giá trị Alpha ≥ Beta), thuật toán sẽ cắt tỉa nhánh đó.

Điều này giúp tiết kiệm thời gian và tài nguyên xử lý, mà vẫn đạt được nước đi tối ưu trong giới hạn tính toán.

Trong cờ vua, người chơi (hoặc AI) thường chỉ tính toán tới một độ sâu nhất định trong cây trò chơi, bởi việc xét toàn bộ trạng thái là không khả thi. Kỹ thuật Alpha-beta giúp AI đánh giá thế cờ nhanh chóng và đưa ra nước đi hợp lý, mô phỏng lối chơi thông minh.



Hình 2-1 Thuật toán Min-Max thông qua cờ cờ ca-rô

2.1.1.1 Trường hợp 1: X là máy tính (computer)

Thuật toán Minimax sẽ quét toàn bộ các trạng thái tiếp diễn có thể xảy ra từ trạng thái U và xác định giá trị tối ưu nhất cho máy tính.

Sau khi phân tích, máy tính nhận thấy rằng sẽ đạt được chiến thắng nếu đi theo trạng thái B1 hoặc B2 (đây là 2 trạng thái tối ưu nhất trong toàn bộ các trạng thái tiếp diễn).

Do đó, máy tính (X) sẽ chọn trạng thái B để đi, nhằm đảm bảo kết quả thuận lợi nhất cho mình.

2.1.1.2 Trường hợp 2: O là máy tính và X là người chơi không giỏi

Người chơi (X): Đi ở trạng thái A.

Máy tính (O): Xét trạng thái A1 và các trạng thái tiếp diễn từ đó.

Máy tính nhận thấy rằng mình không thể đạt được chiến thắng trong các trạng thái tiếp diễn. Tuy nhiên, nó vẫn chọn trạng thái thuận lợi nhất cho mình (giảm rủi ro tối đa). Do đó, máy tính sẽ đi trạng thái A1.

2.1.1.3 Trường hợp 3: O là máy tính và X là người chơi bình thường

Người chơi (X): Đi ở trạng thái B.

Máy tính (O): Sau khi xét các trạng thái tiếp diễn, nhận thấy cả hai trạng thái khả thi đều không mang lại lợi ích hoặc chiến thắng. Trong trường hợp này, máy tính sẽ thực hiện lựa chọn ngẫu nhiên (random) một trạng thái.

Ưu điểm của thuật toán Minimax

Tìm kiếm được tất cả các trạng thái có thể diễn ra, đảm bảo không bỏ sót bất kỳ lựa chọn nào.

Cung cấp kết quả chiến thuật tối ưu trong các trò chơi không gian trạng thái nhỏ.

Hữu ích cho các trò chơi đơn giản như cờ Ca-rô, nơi số lượng trạng thái có thể dễ dàng kiểm soát.

Nhược điểm của thuật toán Minimax

Khi áp dụng cho các trò chơi với không gian trạng thái lớn (như cờ caro hoặc cờ tướng), Minimax không còn hiệu quả do sự bùng nổ tổ hợp trạng thái.

Thuật toán sử dụng nguyên lý vét cạn, không tận dụng được thông tin hiện tại để tăng tốc quá trình tìm kiếm.

2.1.2 Thuật toán cắt tỉa Alpha-beta

Thuật toán cắt tỉa Alpha-beta là kỹ thuật tối ưu hóa dựa trên thuật toán Minimax. Nó giúp giảm số lượng trạng thái cần xét mà vẫn đảm bảo tìm ra kết quả chính xác cho nước đi tối ưu. Cơ chế cốt lõi của Alpha-beta là loại bỏ các trạng thái không cần thiết trong quá trình tìm kiếm, dựa trên các ngưỡng Alpha và Beta.

Nguyên lý hoạt động:

Đỉnh MAX: Luôn nhận giá trị lớn nhất từ các nút con của nó. Mục tiêu là tối đa hóa lợi thế của người chơi.

Đỉnh MIN: Luôn nhận giá trị nhỏ nhất từ các nút con của nó. Mục tiêu là giảm lợi thế của đối thủ.

Cắt tỉa tại các đỉnh:

Đỉnh C (thuộc MAX):

Để chọn giá trị lớn nhất, thuật toán cần so sánh giá trị hiện tại của đỉnh u với giá trị của nút con v.

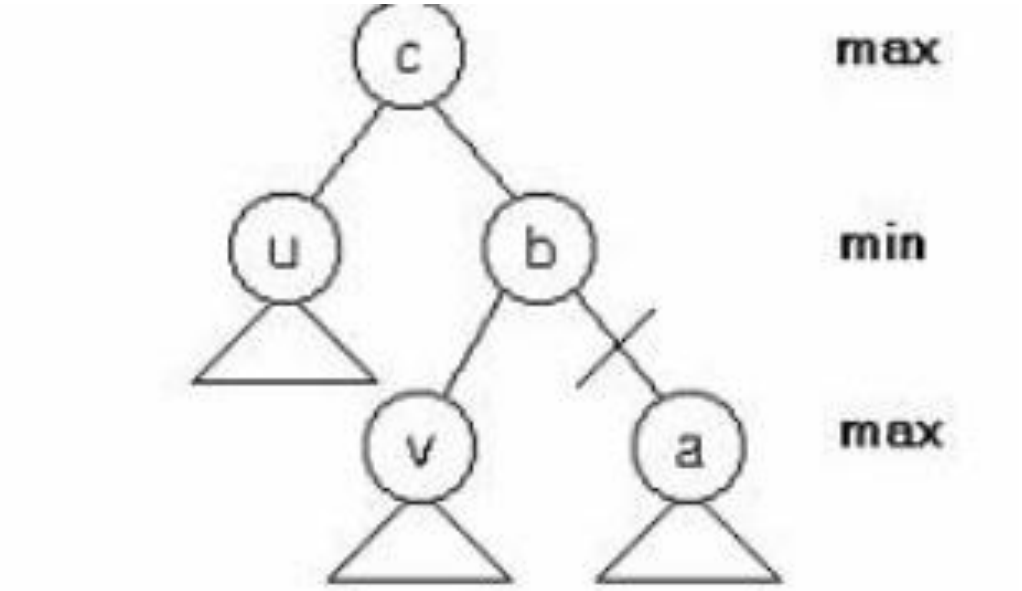
Nếu u > v, việc kiểm tra a là không cần thiết (các giá trị còn lại không thể thay đổi kết quả cuối cùng).

Do đó, cắt tỉa ở đây giúp giảm thời gian xử lý mà vẫn đảm bảo kết quả tại đỉnh C là chính xác.

Đỉnh B (thuộc MIN):

Mục tiêu là chọn giá trị nhỏ nhất từ các nút con. Giả sử, nếu v > a, thì giá trị tại đỉnh B không còn hợp lệ.

Tuy nhiên, nếu u > v, giá trị tại đỉnh C (thuộc MAX) vẫn đúng. Điều này chứng minh rằng ta chỉ cần quan tâm giá trị cuối cùng, thay vì kiểm tra toàn bộ các giá trị trung gian như a.



Hình 2-2 Thuật toán Alpha-Beta

2.2 Thuật toán tiềm kiếm bước đi

Thuật toán tìm kiếm nước đi là một phần quan trọng trong các chương trình mô phỏng cờ vua và AI chơi cờ. Nó có nhiệm vụ xác định tất cả các nước đi hợp lệ từ trạng thái hiện tại của bàn cờ, tuân theo các quy tắc và luật chơi cờ vua. Dưới đây là giải thích và phân tích chi tiết về thuật toán này:

2.2.1 Mục đích

Xác định các nước đi hợp lệ: Thuật toán đảm bảo rằng tất cả các nước đi được liệt kê đều đúng theo luật cờ vua, bao gồm các quy tắc đặc biệt như nhập thành (castling), ăn quân , ăn chéo tốt (En Passant), và thăng cấp tốt (promotion).

Loại bỏ nước đi không hợp lệ: Ví dụ, nếu một nước đi khiến vua bị chiếu, nước đó sẽ bị loại bỏ khỏi danh sách các nước đi hợp lệ.

Hỗ trợ chiến thuật: Sau khi liệt kê các nước đi, AI có thể đánh giá từng nước đi để tìm ra lựa chọn tốt nhất dựa trên trạng thái hiện tại của bàn cờ.

2.2.2 Cách hoạt động

Thuật toán tìm kiếm nước đi thường được triển khai theo các bước sau:

Bước 1: Duyệt qua các quân cờ

Thuật toán sẽ kiểm tra tất cả các quân cờ trên bàn cờ của người chơi đang có lượt đi.

Với mỗi quân cờ, nó sẽ xác định các nước đi hợp lệ dựa trên loại quân và vị trí hiện tại.

Bước 2: Kiểm tra quy tắc di chuyển của từng loại quân

Tốt (Pawn):

Di chuyển thẳng 1 hoặc 2 ô nếu ở lượt đi đầu tiên.

Ăn quân theo đường chéo.

Kiểm tra nước đi đặc biệt En Passant.

Thăng cấp khi di chuyển đến hàng cuối.

Mã (Knight): Di chuyển theo hình chữ "L", và có thể nhảy qua các quân khác.

Xe (Rook): Di chuyển theo hàng ngang hoặc dọc, miễn là không bị chặn bởi quân khác.

Tịnh (Bishop): Di chuyển theo đường chéo, miễn là không bị chặn.

Hậu (Queen): Kết hợp cả nước đi của Xe và Tịnh, có thể di chuyển thẳng và chéo.

Vua (King): Di chuyển 1 ô xung quanh, ngoại trừ các ô bị kiểm soát bởi quân đối phương. Kiểm tra nhập thành nếu vua chưa di chuyển.

Bước 3: Kiểm tra trạng thái chiếu

Mỗi nước đi sẽ được kiểm tra xem nó có khiến vua của người chơi bị chiếu hay không. Nếu có, nước đi đó sẽ bị loại bỏ.

Bước 4: Tạo danh sách các nước đi hợp lệ

Sau khi duyệt qua tất cả các quân cờ và kiểm tra các điều kiện, thuật toán sẽ trả về một danh sách các nước đi hợp lệ.

2.2.3 Ưu điểm và nhược điểm:

Ưu điểm:

Tuân thủ luật chơi: Đảm bảo rằng mọi nước đi được liệt kê đều hợp lệ theo quy tắc cờ vua.

Hỗ trợ chiến thuật: Cung cấp danh sách nước đi để AI hoặc người chơi có thể lựa chọn chiến thuật tốt nhất.

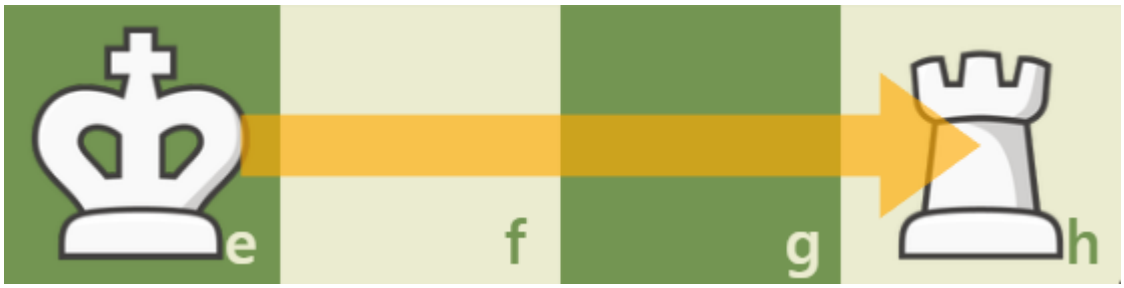
Kết hợp linh hoạt với các thuật toán khác: Các nước đi tìm được có thể được đánh giá bởi thuật toán Minimax hoặc Alpha-Beta để chọn nước đi tối ưu.

Nhược điểm:

Độ phức tạp cao: Trong các trạng thái phức tạp, việc kiểm tra từng nước đi hợp lệ có thể tiêu tốn nhiều tài nguyên.

Phụ thuộc vào thiết kế: Nếu thuật toán không được viết chính xác, có thể xuất hiện nước đi sai hoặc bị thiếu.

Ví dụ



Hình 2-3 Ví dụ minh họa Thuật toán tìm kiếm bước đi

Giả sử trạng thái bàn cờ:

Vua trắng ở ô E1, Xe trắng ở H1.

Vua trắng chưa bị chiếu và có thể thực hiện nhập thành.

Thuật toán sẽ:

Kiểm tra nước đi của vua trắng (E1 → G1).

Kiểm tra các điều kiện nhập thành:

Xe trắng chưa di chuyển.

Không có quân cờ chắn giữa E1 và G1.

Không có ô nào trên đường đi (E1, F1, G1) bị kiểm soát bởi quân đối phương.

Nếu tất cả điều kiện đều hợp lệ, nước đi nhập thành sẽ được thêm vào danh sách.

2.3 Thuật toán đánh giá trạng thái

Thuật toán đánh giá trạng thái là một thành phần quan trọng trong chương trình cờ vua, đặc biệt khi tích hợp các thuật toán tìm kiếm như Minimax hoặc cắt tỉa Alpha-beta. Chức năng chính của thuật toán này là cung cấp một giá trị số đại diện cho mức độ tốt hoặc xấu của trạng thái bàn cờ tại một thời điểm cụ thể. Dựa trên giá trị này, AI có thể đưa ra quyết định về nước đi tiếp theo một cách tối ưu.

Thuật toán đánh giá trạng thái thường dựa trên nhiều yếu tố khác nhau. Một trong những yếu tố cơ bản nhất là giá trị của các quân cờ, trong đó mỗi quân cờ được gán một điểm số phản ánh tầm quan trọng của nó. Ví dụ, quân tốt (Pawn) có giá trị 1 điểm, quân mã (Knight) và tịnh (Bishop) mỗi quân 3 điểm, quân xe (Rook) 5 điểm, quân hậu (Queen) 9 điểm, và quân vua (King) có giá trị rất cao, thường được coi là vô cùng. Tổng giá trị các quân cờ của mỗi bên được tính toán để xác định lợi thế hoặc bất lợi của trạng thái hiện tại. Ngoài ra, thuật toán cũng xét đến vị trí của quân cờ trên bàn; những quân cờ ở trung tâm bàn cờ thường có giá trị cao hơn vì chúng kiểm soát nhiều ô hơn và có thể triển khai các chiến thuật hiệu quả hơn.

Giả sử trạng thái bàn cờ như sau:

AI : Queen, Rook, và Pawn → Tổng giá trị: $9+5+1\times3=17$

Đối thủ : Queen và Bisho→ Tổng giá trị: $9+3=12$

$score=17-12=+5$

Điểm dương (+5) cho thấy trạng thái bàn cờ đang có lợi cho AI.

2.4 Thuật toán quả lý sự kiện

Quản lý sự kiện (Event Handling) là một khía cạnh quan trọng trong phát triển phần mềm, đặc biệt trong các ứng dụng có giao diện đồ họa, hệ thống tương tác, hoặc trò chơi. Thuật toán quản lý sự kiện được thiết kế để xử lý các hành động hoặc sự kiện xảy ra do người dùng hoặc hệ thống tạo ra, chẳng hạn như nhấn phím, di chuyển chuột, kéo thả, hoặc thậm chí các sự kiện tự động từ phần mềm. Việc quản lý sự kiện đảm bảo rằng mỗi hành động sẽ được nhận diện, xử lý đúng cách và có phản hồi phù hợp.

2.4.1 Mục đích

Theo dõi sự kiện từ đầu vào: Nhận biết các hành động từ người dùng, như nhấp chuột, kéo thả, hoặc nhập văn bản.

Phản hồi sự kiện kịp thời: Đảm bảo ứng dụng thực hiện hành động phù hợp ngay khi sự kiện xảy ra.

Xử lý các quy tắc và logic ứng dụng: Liên kết sự kiện với các chức năng tương ứng, ví dụ, khi người chơi kéo một quân cờ, kiểm tra xem nước đi có hợp lệ không.

Đảm bảo luồng chương trình ổn định: Ngăn ngừa lỗi khi xử lý các sự kiện song song hoặc trong các trạng thái phức tạp.

2.4.2 Các Bước Cơ Bản của Thuật Toán Quản Lý Sự Kiện

Thuật toán quản lý sự kiện thường tuân theo các bước sau:

Bước 1: Phát hiện sự kiện (Event Detection)

Sử dụng bộ lắng nghe sự kiện (Event Listeners) để phát hiện sự kiện từ người dùng hoặc hệ thống.

Trong Java, có các MouseListener để theo dõi các hành động liên quan đến chuột, hoặc KeyListener để theo dõi các phím bấm.

Ví dụ thực tế: Khi người dùng nhấp chuột trái vào màn hình, sự kiện "Mouse Clicked" sẽ được kích hoạt và chuyển sang bước xử lý.

Bước 2: Xác định loại sự kiện (Event Identification)

Mỗi sự kiện sẽ được gán một loại cụ thể, chẳng hạn như "Mouse Pressed," "Mouse Released,",...

Xác định loại sự kiện giúp định tuyến đến hàm xử lý phù hợp.

Ví dụ thực tế: Nếu sự kiện là "Mouse Dragged," thuật toán sẽ liên kết nó với hành vi kéo thả (drag-and-drop).

Bước 3: Xử lý sự kiện (Event Handling)

Khi một sự kiện được phát hiện và xác định, thuật toán sẽ gọi các hàm hoặc phương thức phù hợp để xử lý sự kiện.

Quy trình xử lý có thể bao gồm:

Cập nhật giao diện người dùng (ví dụ: tô sáng ô cờ khi di chuột qua).

Thực hiện logic ứng dụng (ví dụ: kiểm tra tính hợp lệ của nước đi cờ vua).

Tương tác với dữ liệu hoặc trạng thái (ví dụ: lưu trạng thái quân cờ đang được kéo).

Bước 4: Cập nhật và phản hồi (Update and Respond)

Sau khi xử lý sự kiện, thuật toán sẽ cập nhật trạng thái chương trình hoặc giao diện.

Gửi phản hồi trực quan hoặc logic cho người dùng, ví dụ:

Hiển thị thông báo nếu có lỗi xảy ra (như nước đi không hợp lệ).

Di chuyển quân cờ trên bàn cờ nếu nước đi hợp lệ.

2.4.3 Ưu điểm và nhược điểm

Ưu Điểm

Tương tác động: Đảm bảo chương trình có thể phản hồi ngay lập tức mọi hành động của người dùng, giúp tạo trải nghiệm trực quan.

Phân tách trách nhiệm:

Sử dụng các Event Listeners giúp cô lập logic xử lý sự kiện khỏi các phần khác của chương trình, dễ bảo trì.

Tính mở rộng: Dễ dàng thêm hoặc sửa đổi các sự kiện mới mà không ảnh hưởng đến cấu trúc hiện tại.

Hiệu quả: Thay vì kiểm tra liên tục, thuật toán chỉ xử lý khi có sự kiện xảy ra, tiết kiệm tài nguyên hệ thống.

Nhược Điểm và Thách Thức

Phức tạp khi xử lý sự kiện song song:

Trong các ứng dụng lớn, nhiều sự kiện có thể xảy ra đồng thời, đòi hỏi xử lý đồng bộ (synchronization) để tránh xung đột.

Ràng buộc vào GUI Frameworks:Đa phần thuật toán quản lý sự kiện phụ thuộc vào framework hoặc thư viện cụ thể (như Swing, JavaFX trong Java).

Lỗi tiềm ẩn do trạng thái:Nếu trạng thái ứng dụng không được cập nhật đúng cách, có thể dẫn đến lỗi logic trong quá trình xử lý sự kiện.

2.4.4 Ví Dụ

Giả định tình huống: Trò chơi kéo thả (Drag-and-Drop) quân cờ

Tình huống: Người chơi nhấn chuột trái vào quân cờ, kéo nó đến vị trí mới, và thả ra.

Cách quản lý sự kiện:

Mouse Pressed (Nhấn chuột):

Phát hiện xem người chơi đang chọn quân cờ nào.

Lưu tọa độ ban đầu của quân cờ.

Mouse Dragged (Kéo chuột):

Khi người chơi kéo chuột, cập nhật vị trí tạm thời của quân cờ theo tọa độ chuột.

Mouse Released (Thả chuột):

Kiểm tra xem vị trí thả chuột có hợp lệ không (tuân theo quy tắc trò chơi, ví dụ: quân cờ không được "đè" lên vua).

Nếu hợp lệ, cập nhật vị trí chính thức của quân cờ và làm mới bàn cờ.

Nếu không hợp lệ, trả quân cờ về vị trí ban đầu.

2.4.5 Ứng Dụng Cụ Thể trong Cờ Vua

Trong cờ vua:

Di chuyển quân cờ: Khi người chơi nhấp và kéo thả quân cờ, thuật toán xử lý sự kiện sẽ kiểm tra tính hợp lệ của nước đi (dựa trên quy tắc).

Hiển thị thông báo lỗi: Nếu người chơi thực hiện nước đi không hợp lệ (ví dụ: vua bị chiếu), thuật toán sẽ gửi phản hồi bằng thông báo trên màn hình.

Cập nhật trạng thái: Sau mỗi nước đi, thuật toán sẽ làm mới bàn cờ và chuyển lượt cho đối thủ.

2.5 Công cụ sử dụng

2.5.1 Java

Java là một trong những ngôn ngữ lập trình hướng đối tượng. Ngôn ngữ Java được sử dụng phổ biến trong phát triển phần mềm, trang web, game hay ứng dụng trên các thiết bị di động.

Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun MicroSystem năm 1991. Ban đầu Java được tạo ra nhằm mục đích viết phần mềm cho các sản phẩm gia dụng, và có tên là Oak. Java được chính thức phát hành năm 1994, đến năm 2010 được Oracle mua lại từ Sun MicroSystem

Hình 2-4 Java

Ưu điểm:

- Độ tin cậy cao

- Tính đa nền tảng

- Quản lý bộ nhớ tự động

- Công cụ phát triển phong phú

- Hỗ trợ đa luồng

Nhược điểm

Hiệu suất: Java sử dụng máy ảo Java (JVM), điều này giúp Java trở nên linh hoạt nhưng có thể dẫn đến hiệu suất chậm hơn so với các ngôn ngữ lập trình gần sát phần cứng như C hoặc C++. Các ứng dụng Java thường yêu cầu tối ưu hóa để đạt được hiệu quả cao.

Quản lý bộ nhớ: Mặc dù Java có trình dọn rác (garbage collector) để quản lý bộ nhớ, nhưng điều này đôi khi khiến việc kiểm soát bộ nhớ trở nên khó khăn hơn, đặc biệt đối với các ứng dụng yêu cầu hiệu suất cao.

Cú pháp: Java có cú pháp tương đối phức tạp và dài dòng hơn so với các ngôn ngữ hiện đại như Python hoặc Ruby. Điều này có thể làm giảm tốc độ lập trình và đôi khi gây khó khăn cho người mới học.

Phụ thuộc vào JVM: Để chạy được ứng dụng Java, thiết bị cần phải cài đặt JVM. Điều này có thể gây phiền phức trong một số trường hợp, đặc biệt là khi triển khai trên các thiết bị mà JVM không phổ biến.

Chi phí bộ nhớ: Ứng dụng Java thường tiêu thụ bộ nhớ nhiều hơn so với một số ngôn ngữ lập trình khác. Điều này có thể làm giảm hiệu suất trên các thiết bị có cấu hình phần cứng thấp.

Phát triển ứng dụng giao diện người dùng: Mặc dù Java có các thư viện như Swing và JavaFX, nhưng việc phát triển giao diện người dùng không được coi là thuận lợi và hiệu quả như một số framework hoặc ngôn ngữ chuyên dụng khác.

2.5.2 Apache NetBean

NetBeans là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) mã nguồn mở, ra đời từ năm 1996 dưới tên gọi Xelfi, ban đầu là dự án IDE Java được phát triển bởi sinh viên dưới sự hướng dẫn của Khoa Toán - Lý tại Đại học Charles, Prague. Sau đó, vào năm 1997, Roman Staněk đã thành lập một công ty quanh dự án này, và sau đó bán phiên bản thương mại của NetBeans cho đến khi Sun Microsystems mua lại vào năm 1999.

Hình 2-5 Apache NetBeans

Sau khi mua lại, Sun Microsystems mở mã nguồn của NetBeans vào tháng 6 năm 2000, từ đó NetBeans trở thành một dự án mã nguồn mở phát triển dưới sự hỗ trợ của cộng đồng người dùng. Oracle Corporation mua lại Sun vào năm 2010, khiến NetBeans trở thành một phần của Oracle. Tuy nhiên, vào tháng 9/2016, Oracle đã đề xuất quyên tặng dự án NetBeans cho Apache Software Foundation, để mở ra mô hình quản trị mới.

NetBeans không chỉ hỗ trợ việc phát triển ứng dụng bằng nhiều ngôn ngữ lập trình như Java, PHP, C/C++, JavaScript và các ngôn ngữ khác, mà còn cung cấp một loạt các tính năng như gỡ lỗi, tự động hoàn thành mã, quản lý phiên bản và tích hợp với các công cụ hỗ trợ phát triển phổ biến. IDE này cũng cho phép người dùng mở rộng chức năng, thông qua việc cài đặt các plugin từ cộng đồng lập trình viên.

Nhờ khả năng hỗ trợ đa ngôn ngữ và giao diện người dùng thân thiện, NetBeans đã trở thành một công cụ lý tưởng cho việc phát triển ứng dụng đa nền tảng, và hỗ trợ nhu cầu của nhiều dự án khác nhau. Từ người mới bắt đầu đến những lập trình viên

có kinh nghiệm, NetBeans đều cung cấp cho họ sự tiện ích, hỗ trợ rất nhiều trong quá trình phát triển phần mềm.

Một số tính năng của phần mềm NetBean:

- Tích hợp trình biên dịch: Tích hợp trình biên dịch cho nhiều ngôn ngữ, giúp kiểm tra và chạy ứng dụng mà không cần thoát khỏi môi trường IDE.\

- Tích hợp với Apache: Được phát triển và hỗ trợ bởi Apache Software Foundation, NetBeans cho phép dễ dàng tích hợp các công nghệ và plugin từ Apache, tăng hiệu suất làm việc của người dùng.

- Hỗ trợ Git: Tích hợp sẵn hệ thống quản lý phiên bản Git, giúp các lập trình viên quản lý mã nguồn và làm việc nhóm một cách dễ dàng.

- Tích hợp Plugin: Hỗ trợ một loạt các plugin từ cộng đồng lập trình viên, mở rộng tính năng và tùy chỉnh IDE theo nhu cầu cụ thể của dự án.

- Hỗ trợ phát triển web: IDE hỗ trợ các framework phổ biến cho phát triển ứng dụng web như JavaServer Faces (JSF), Struts và Spring MVC.

- Tích hợp server ứng dụng: NetBeans có tích hợp các server ứng dụng phổ biến như Apache Tomcat, GlassFish và WildFly, giúp người dùng triển khai ứng dụng dễ dàng.

- Debug và Profiling: Cung cấp tính năng gỡ lỗi và đánh giá hiệu năng, giúp người dùng xác định, sửa lỗi trong mã nguồn nhanh chóng và hiệu quả.

- Xác định mã nguồn thông minh: Cung cấp tính năng gợi ý mã, hoàn thành mã tự động và xác định lỗi cú pháp, tăng năng suất, giảm lỗi trong quá trình phát triển ứng dụng.

- Có thiết kế giao diện người dùng: NetBeans đi kèm với trình thiết kế giao diện Swing và JavaFX, để thiết kế giao diện người dùng một cách dễ dàng.

- Hỗ trợ dự án lớn: Có khả năng hỗ trợ các dự án lớn có cấu trúc phức tạp, quản lý mã nguồn hiệu quả.

- Hỗ trợ đa ngôn ngữ: Hỗ trợ phát triển ứng dụng trong nhiều ngôn ngữ lập trình như Java, PHP, C/C++, JavaScript, HTML, CSS và các ngôn ngữ khác.

- Giao diện người dùng thân thiện: Giao diện người dùng dễ sử dụng, phù hợp cho cả người mới bắt đầu và người có kinh nghiệm, giúp tăng hiệu suất làm việc.

CHƯƠNG 3 XÂY DỰNG CHƯƠNG TRÌNH

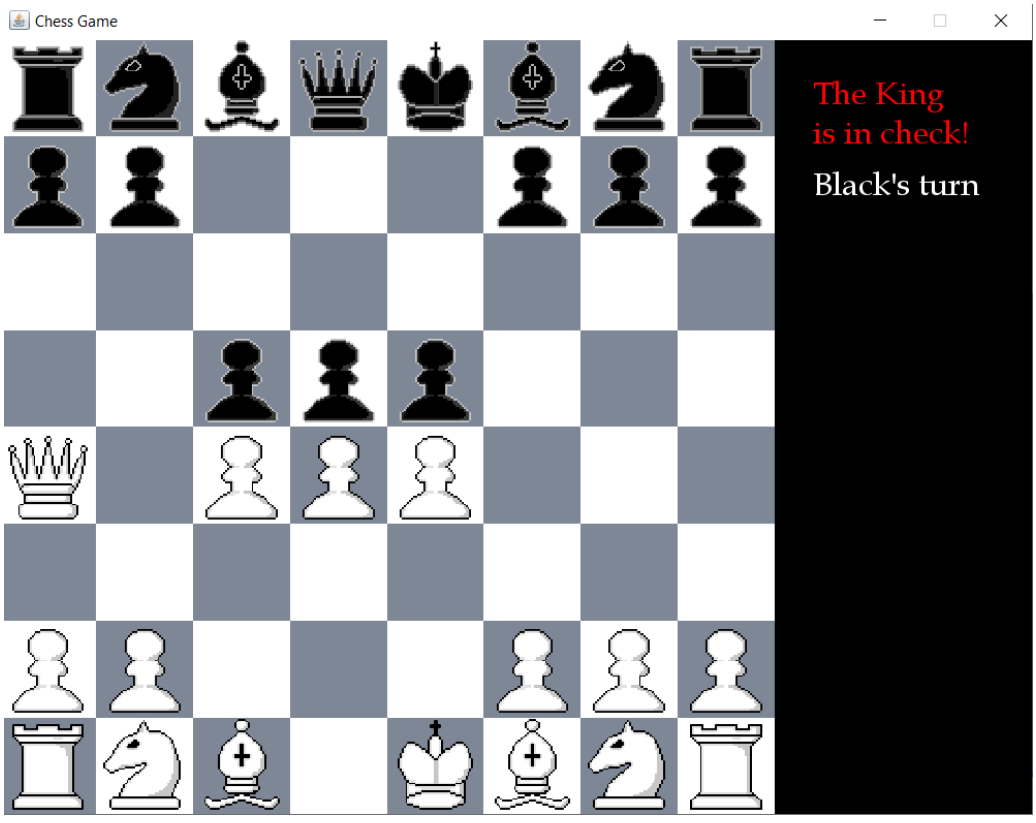
3.1 Giao diện

3.1.1 paintComponent(Graphics g)

Mục đích: Hiển thị toàn bộ trạng thái bàn cờ và trò chơi.

Hoạt động:

- Vẽ bàn cờ.
- Vẽ quân cờ.
- Tô màu ô mục tiêu tùy vào trạng thái hợp lệ của nước đi.
- Hiển thị các thông báo liên quan đến trạng thái trò chơi: lượt chơi hiện tại, chiếu vua, kết thúc, hoặc hòa cờ.



Hình 3-1 Giao diện thông báo

Hình 3-2 Giao diện kết thúc

3.2 Các thuật toán hỗ trợ

3.2.1 changePlayer()

Mục đích: Chuyển lượt chơi giữa hai bên, đặt lại trạng thái của quân Tốt, và kiểm tra điều kiện kết thúc trò chơi.

H động:

- Thay đổi người chơi hiện tại (trắng ↔ đen).
- Đặt lại trạng thái quân Tốt.
- Nếu chơi với AI, gọi logic của AI để thực hiện nước đi.
- Kiểm tra điều kiện kết thúc.

3.2.2 updatePosition()

Mục đích:

Cập nhật vị trí hiện tại của quân cờ sau khi thực hiện một nước đi.

Kiểm tra điều kiện để kích hoạt trạng thái En Passant khi quân Tốt di chuyển 2 ô trong lượt đi.

En Passant:

Nếu quân cờ là Tốt (type == Type.PAWN) và di chuyển 2 hàng (Math.abs(row - preRow) == 2), trạng thái twoStepped được kích hoạt. Điều này sẽ hỗ trợ kiểm tra nước đi đặc biệt En Passant.

Cập nhật vị trí:

Tọa độ x, y của quân cờ được tính lại dựa trên cột và hàng mới (getX(col) và getY(row)).

Vị trí trước đó (preCol, preRow) được đồng bộ với vị trí hiện tại.

Đánh dấu quân cờ đã di chuyển (moved = true).

3.2.3 isWithThinBoard(targetCol, targetRow)

Mục đích:

Kiểm tra xem ô mục tiêu (targetCol, targetRow) có nằm trong phạm vi hợp lệ của bàn cờ (8x8).

- Ô hợp lệ nếu targetCol nằm trong khoảng [0, 7] và targetRow nằm trong khoảng [0, 7].
- Trả về true nếu điều kiện thỏa mãn, ngược lại trả về false.

3.2.4 resetPosition()

Mục đích:

Khôi phục quân cờ về vị trí trước đó nếu nước đi không hợp lệ hoặc bị hủy.

- Cột và hàng của quân cờ được đặt về giá trị trước đó (preCol, preRow).
- Tọa độ x, y cũng được tính lại dựa trên vị trí cột và hàng trước đó.

3.2.5 isValidSquare(targetCol, targetRow)

Mục đích:

Xác định xem ô mục tiêu có hợp lệ không, dựa trên trạng thái của ô (trống hoặc chứa quân đối thủ).

- Nếu ô mục tiêu trống: Trả về true.
- Nếu ô mục tiêu chứa quân đối thủ: Trả về true vì quân đó có thể bị bắt.
- Nếu ô mục tiêu chứa quân cùng màu: Không hợp lệ, đặt hittingP = null và trả về false.

3.2.6 isSameSquare(targetCol, targetRow)

Mục đích:

Kiểm tra xem ô mục tiêu (targetCol, targetRow) có trùng với vị trí hiện tại của quân cờ hay không.

- So sánh giá trị cột và hàng hiện tại (preCol, preRow) với giá trị của ô mục tiêu.
- Trả về true nếu trùng, ngược lại trả về false.

3.2.7 pieceIsOnStraightLine(targetCol, targetRow)

Mục đích:

Kiểm tra xem có quân cờ nào nằm trên đường thẳng giữa vị trí hiện tại và ô mục tiêu.

- Di chuyển sang trái: Duyệt qua từng ô giữa vị trí hiện tại và mục tiêu, kiểm tra xem có quân cờ tại mỗi ô không.
- Di chuyển sang phải: Tương tự, kiểm tra từ vị trí hiện tại tới mục tiêu ở hướng phải.
- Di chuyển lên: Kiểm tra các ô phía trên, theo hàng dọc.
- Di chuyển xuống: Kiểm tra các ô phía dưới, theo hàng dọc.
- Nếu phát hiện quân cờ, đặt hittingP và trả về true. Ngược lại, trả về false.

3.2.8 pieceIsOnDiagonalLine(targetCol, targetRow)

Mục đích:

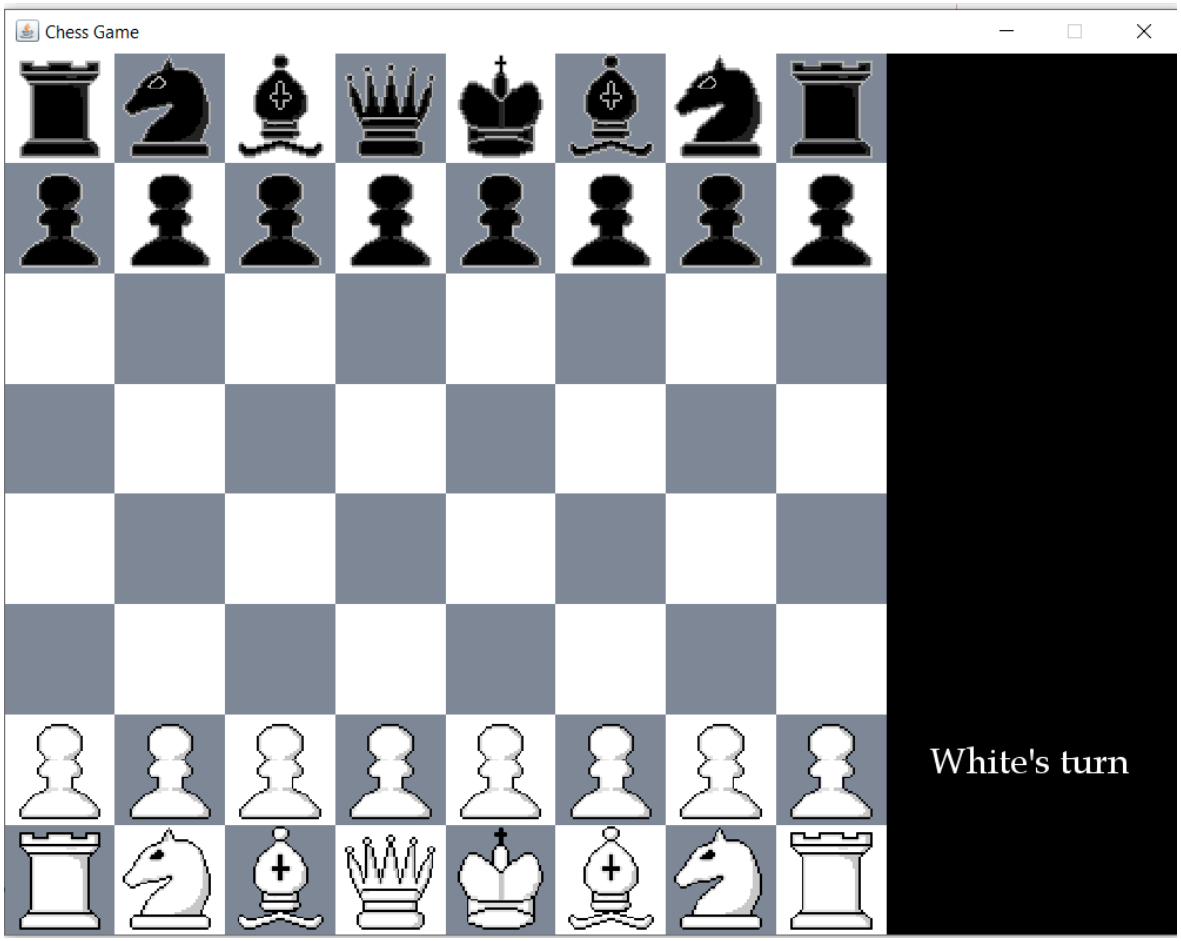
Kiểm tra xem có quân cờ nào nằm trên đường chéo giữa vị trí hiện tại và ô mục tiêu.

- Di chuyển lên trái: Kiểm tra các ô đường chéo phía trên bên trái.
- Di chuyển lên phải: Kiểm tra các ô đường chéo phía trên bên phải.
- Di chuyển xuống trái: Kiểm tra các ô đường chéo phía dưới bên trái.
- Di chuyển xuống phải: Kiểm tra các ô đường chéo phía dưới bên phải.
- Sử dụng độ chênh lệch (diff) giữa cột để tính hàng tương ứng theo đường chéo.
- Nếu phát hiện quân cờ, đặt hittingP và trả về true. Ngược lại, trả về false.

3.3 Mô tả trò chơi Cờ vua

Giới thiệu cơ bản:

Cờ vua là một trò chơi trí tuệ mang tính chiến thuật, được chơi theo lượt giữa hai người, chia thành hai bên quân: trắng và đen.



Hình 3-3 Giao diện bàn cờ

Bàn cờ: Gồm 64 ô vuông xen kẽ trắng và đen, xếp thành lưới 8x8.

Quân cờ: Mỗi bên sẽ có 16 quân, bao gồm:

- 2 Xe (Rook)
- 2 Mã (Knight)
- 2 Tịnh hay Tượng (Bishop)
- 1 Hậu (Queen)
- 1 Vua (King)
- 8 Tốt (Pawn)
- Quy tắc cơ bản:
- Bố trí quân cờ:

Hậu trắng đứng bên ô trắng, Hậu đen đứng bên ô đen (đảm bảo đúng vị trí đầu ván cờ).

Các quân cờ khác được xếp theo cấu trúc mặc định trên bàn.

3.3.1 Tốt (Pawn):

Đi thẳng về phía trước.

Ăn quân theo đường chéo.

Không được di chuyển lùi.

Khi Tốt đến hàng cuối cùng của bàn cờ (hàng 8), nó sẽ thăng cấp thành bất kỳ quân nào (trừ vua).

3.3.1.1 Mã giả

// Hàm kiểm tra tính hợp lệ của nước đi

function canMove(targetCol, targetRow):

// Kiểm tra xem nước đi có nằm trong bàn cờ và không phải ô hiện tại

if isWithinBoard(targetCol, targetRow) AND isNotSameSquare(targetCol, targetRow):

// Xác định hướng di chuyển dựa trên màu quân (trắng hoặc đen)

if color == WHITE:

moveValue = -1

else:

moveValue = 1

// Kiểm tra xem có quân cờ nào bị ăn tại ô mục tiêu

hittingPiece = getHittingPiece(targetCol, targetRow)

// Kiểm tra di chuyển một ô

if targetCol == currentCol AND targetRow == currentRow + moveValue AND hittingPiece == NULL:

return true

// Kiểm tra di chuyển hai ô (chỉ khi quân Tốt chưa từng di chuyển)

if targetCol == currentCol AND targetRow == currentRow + 2 * moveValue AND hittingPiece == NULL AND moved == false AND isStraightLineClear(targetCol, targetRow):

return true

// Kiểm tra ăn quân theo đường chéo

if ABS(targetCol - currentCol) == 1 AND targetRow == currentRow + moveValue AND hittingPiece != NULL AND hittingPiece.color != color:

return true

// Kiểm tra nước đi đặc biệt En Passant

if ABS(targetCol - currentCol) == 1 AND targetRow == currentRow + moveValue:

for piece in simulatedPieces:

if piece.col == targetCol AND piece.row == currentRow AND piece.twoStepped == true:

hittingPiece = piece

return true

// Nếu không thỏa mãn bất kỳ điều kiện nào, nước đi không hợp lệ

return false

3.3.1.2 Giải thích các bước

Kiểm tra điều kiện cơ bản

-Xác nhận ô mục tiêu (targetCol, targetRow) nằm trong bàn cờ (8x8).

-Xác nhận rằng ô mục tiêu không phải là ô hiện tại của quân Tốt.

Xác định hướng di chuyển

-Dựa vào màu quân (trắng hoặc đen), thiết lập giá trị hướng di chuyển (moveValue). Tốt trắng di chuyển lên trên bàn cờ (giảm hàng), còn Tốt đen di chuyển xuống dưới (tăng hàng).

Kiểm tra các kiểu di chuyển

Di chuyển một ô về phía trước:

-Quân Tốt di chuyển thẳng tới ô tiếp theo trên cùng cột, nếu ô đó không bị chặn bởi quân cờ nào.

Di chuyển hai ô về phía trước:

-Quân Tốt chỉ có thể đi hai ô nếu đây là nước đi đầu tiên của nó (biến moved == false), và không có quân cờ nào cản trở trên đường đi.

Hình 3-4 Pawn di chuyển 1 hoặc 2 ô về phía trước

Ăn quân theo đường chéo:

-Quân Tốt có thể ăn quân đối phương nếu di chuyển tới một ô chéo (chênh lệch cột là 1), và ô đó chứa quân cờ đối lập màu (hittingPiece.color != color).

Hình 3-5 Pawn ăn quân theo đường chéo

En Passant:

-Kiểm tra nước đi đặc biệt En Passant khi quân Tốt có thể ăn quân đối phương sau nước đi hai ô của quân đó ở lượt trước. Duyệt qua danh sách các quân trong trạng thái mô phỏng (simPieces) để kiểm tra điều kiện.

Hình 3-6 En Passant giai đoạn 1

Hình 3-7 En Passant giai đoạn 2

Bước 4: Kết thúc

-Nếu nước đi thỏa mãn một trong các điều kiện, trả về true (hợp lệ).

-Ngược lại, trả về false (không hợp lệ).

3.3.2 Thăng cấp (Promotion)

3.3.2.1 Mục đích

- Kiểm tra quy tắc thăng cấp quân Tốt trong cờ vua:
- Quân Tốt trắng thăng cấp khi đến hàng cuối cùng (row = 0).

- Quân Tốt đen thăng cấp khi đến hàng cuối cùng (row = 7).
- Tạo danh sách quân cờ có thể chọn để thăng cấp.

3.3.2.2 Phân tích logic

-Điều kiện kiểm tra:

- Xác định quân cờ hiện tại:
- Phương thức chỉ áp dụng cho quân Tốt (activeP.type == Type.PAWN).

-Kiểm tra hàng cuối cùng của bàn cờ:

- Tốt trắng (White) phải ở hàng 0 (activeP.row == 0).
- Tốt đen (Black) phải ở hàng 7 (activeP.row == 7).

-Hành động khi đủ điều kiện thăng cấp:

- Xóa danh sách thăng cấp hiện tại: promoPieces.clear();
- Danh sách promoPieces được làm trống để chuẩn bị thêm các quân cờ mới.
- Thêm quân cờ có thể chọn để thăng cấp:

Các quân cờ được thêm vào danh sách thăng cấp:

- Xe (Rook): Vị trí ở cột 9, hàng 2.
- Mã (Knight): Vị trí ở cột 9, hàng 3.
- Tịnh (Bishop): Vị trí ở cột 9, hàng 4.
- Hậu (Queen): Vị trí ở cột 9, hàng 5.

Tất cả quân cờ này có cùng màu với quân Tốt đang thăng cấp (currentColor).

Trả về kết quả:

Nếu quân Tốt đủ điều kiện, trả về true. Nếu không, trả về false.

3.3.2.3 Mã giả

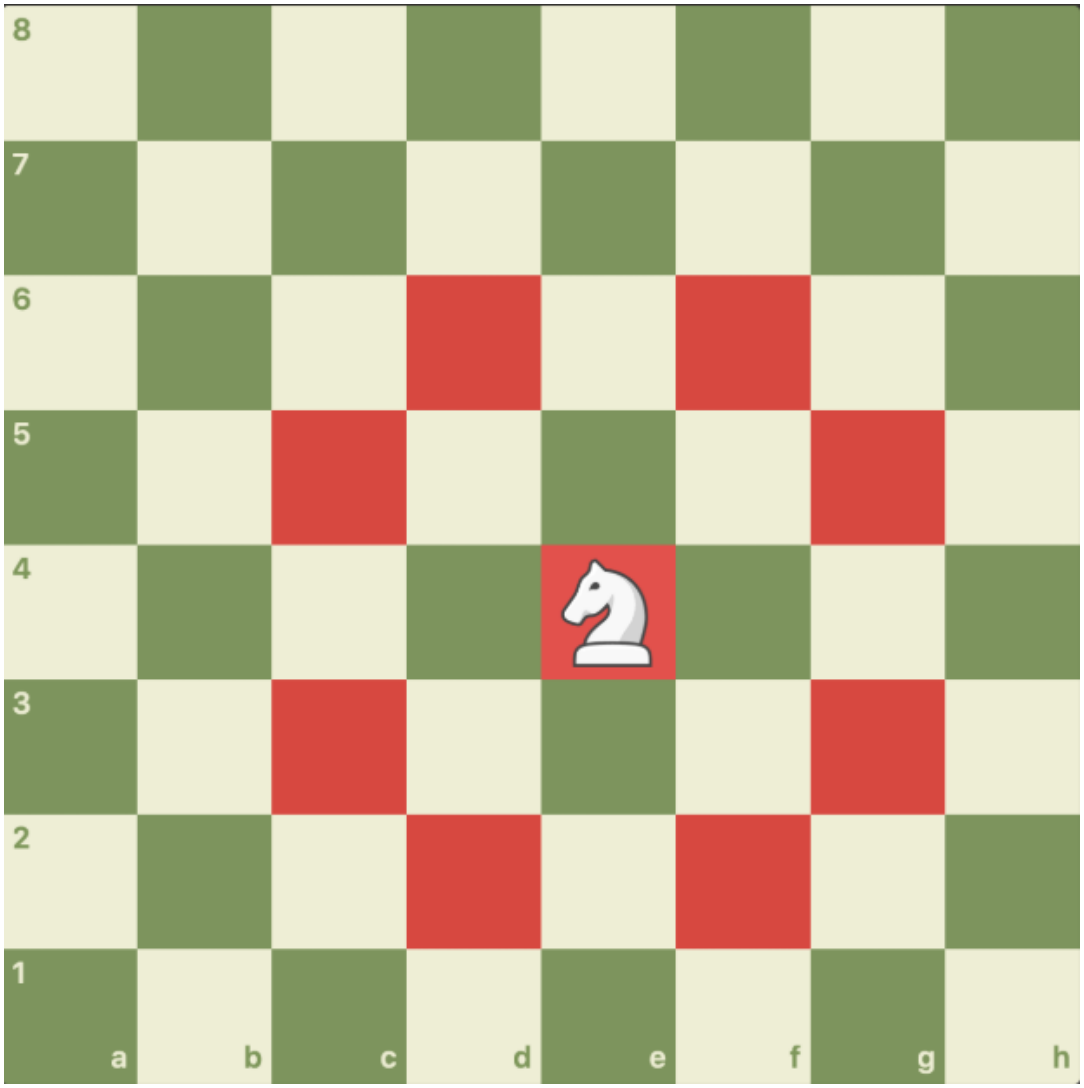
```
function canPromote():
   if activePiece.type == PAWN:
```

```
            if (currentColor == WHITE AND activePiece.row == 0) OR
              (currentColor == BLACK AND activePiece.row == 7):
                promoPieces.clear()
                promoPieces.add(new Rook(9, 2, currentColor))
                promoPieces.add(new Knight(9, 3, currentColor))
                promoPieces.add(new Bishop(9, 4, currentColor))
                promoPieces.add(new Queen(9, 5, currentColor))
                return true
        return false
```

3.3.3 Mã (Knight):

-Đi theo hình chữ "L" (2 ô theo một hướng và 1 ô theo hướng vuông góc).

-Là quân cờ duy nhất có thể nhảy qua các quân khác.



Hình 3-8 Cách thức di chuyển của Knight

3.3.3.1 Mã giả

// Hàm kiểm tra tính hợp lệ của nước đi của quân Mã

function canMove(targetCol, targetRow):

// Kiểm tra xem ô mục tiêu có nằm trong bàn cờ hay không

if !isWithinBoard(targetCol, targetRow):

return false

// Kiểm tra tỷ lệ di chuyển của quân Mã (chữ "L": 1:2 hoặc 2:1)

```
if ABS(targetCol - currentCol) * ABS(targetRow - currentRow) != 2:
    return false
// Kiểm tra ô mục tiêu có hợp lệ không
if !isValidSquare(targetCol, targetRow):
    return false
// Nếu tất cả điều kiện đều thỏa mãn, nước đi hợp lệ
return true
```

3.3.3.2 Giải thích các bước:

Kiểm tra giới hạn bàn cờ:

-Quân Mã chỉ có thể di chuyển trong phạm vi bàn cờ 8x8. Nếu ô đích nằm ngoài phạm vi này, nước đi không hợp lệ.

Kiểm tra tỷ lệ di chuyển (hình chữ "L"):

-Tỷ lệ sản phẩm khoảng cách của cột và hàng (1:2 hoặc 2:1) đảm bảo nước đi tuân theo quy tắc của quân Mã.

Kiểm tra ô đích hợp lệ:

-Gọi hàm isValidSquare để kiểm tra rằng:

-Ô đích không bị chặn bởi quân cờ cùng màu.

-Ô đích có thể chứa quân cờ đối thủ để thực hiện nước ăn quân.

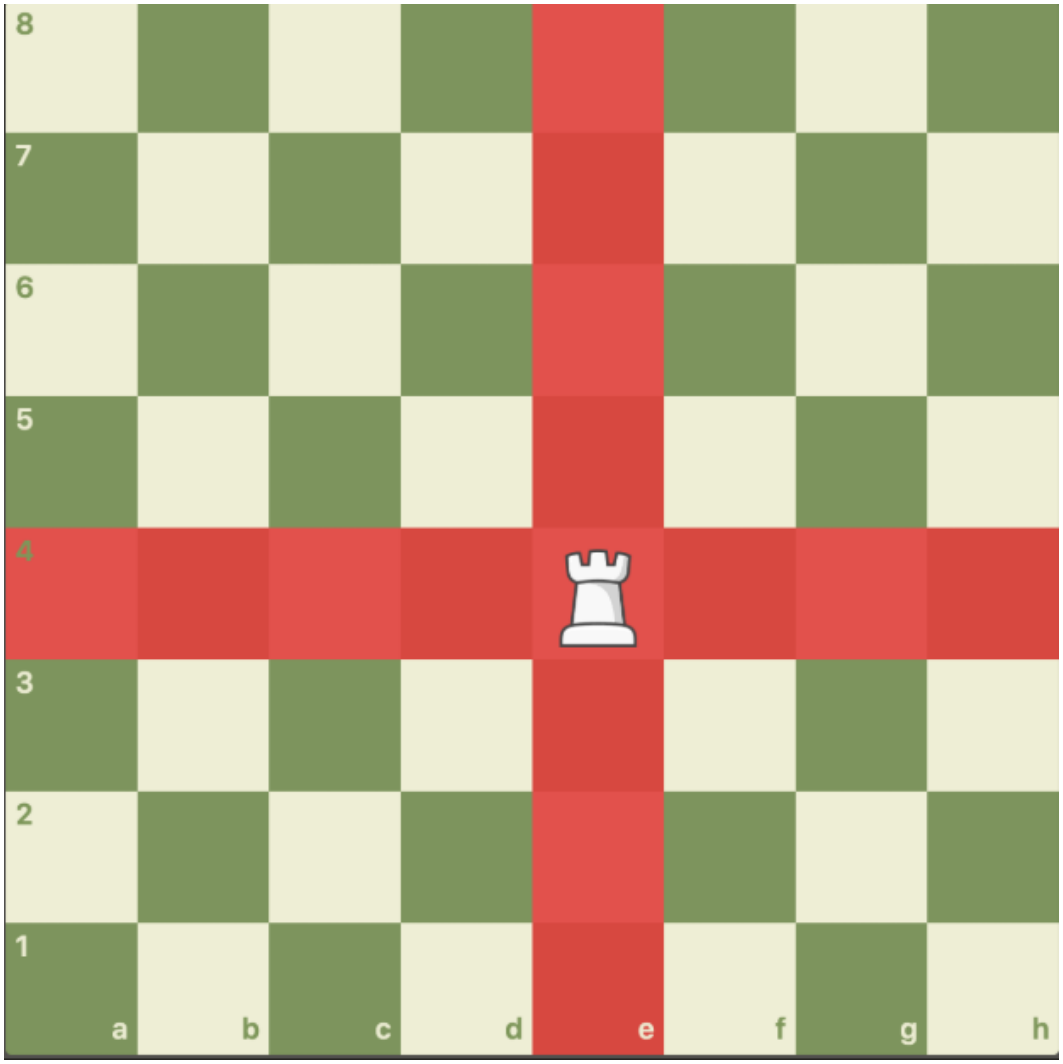
Trả về kết quả:

-Nếu tất cả các điều kiện được đáp ứng, nước đi được coi là hợp lệ.

3.3.4 Xe (Rook):

-Di chuyển thẳng theo hàng ngang hoặc dọc.

-Ăn quân cũng trên đường thẳng.



Hình 3-9 Phương thức di chuyển của Rook

3.3.4.1 Mã giả

// Hàm kiểm tra tính hợp lệ của nước đi của quân Xe

function canMove(targetCol, targetRow):

  // Kiểm tra ô mục tiêu có nằm trong bàn cờ

  if !isWithinBoard(targetCol, targetRow):

    return false

  // Kiểm tra không di chuyển tại ô hiện tại

```
if isSameSquare(targetCol, targetRow):
    return false
// Kiểm tra hướng di chuyển: hàng ngang hoặc hàng dọc
if targetCol != currentCol AND targetRow != currentRow:
    return false
// Kiểm tra tính hợp lệ của ô mục tiêu
if !isValidSquare(targetCol, targetRow):
    return false
// Kiểm tra không có quân cờ chắn đường di chuyển
if pieceIsOnStraightLine(targetCol, targetRow):
    return false
// Nếu thỏa mãn tất cả điều kiện, nước đi hợp lệ
return true
```

3.3.4.2 Giải thích các bước

Kiểm tra giới hạn bàn cờ:

-Quân Xe chỉ có thể di chuyển đến các ô nằm trong bàn cờ.

Kiểm tra ô mục tiêu khác ô hiện tại:

-Nước đi phải thay đổi vị trí của quân Xe.

Kiểm tra hướng di chuyển:

-Quân Xe chỉ di chuyển theo chiều ngang (cột giữ nguyên) hoặc chiều dọc (hàng giữ nguyên).

Kiểm tra tính hợp lệ của ô mục tiêu:

-Ô mục tiêu phải hợp lệ, không chứa quân cờ cùng màu.

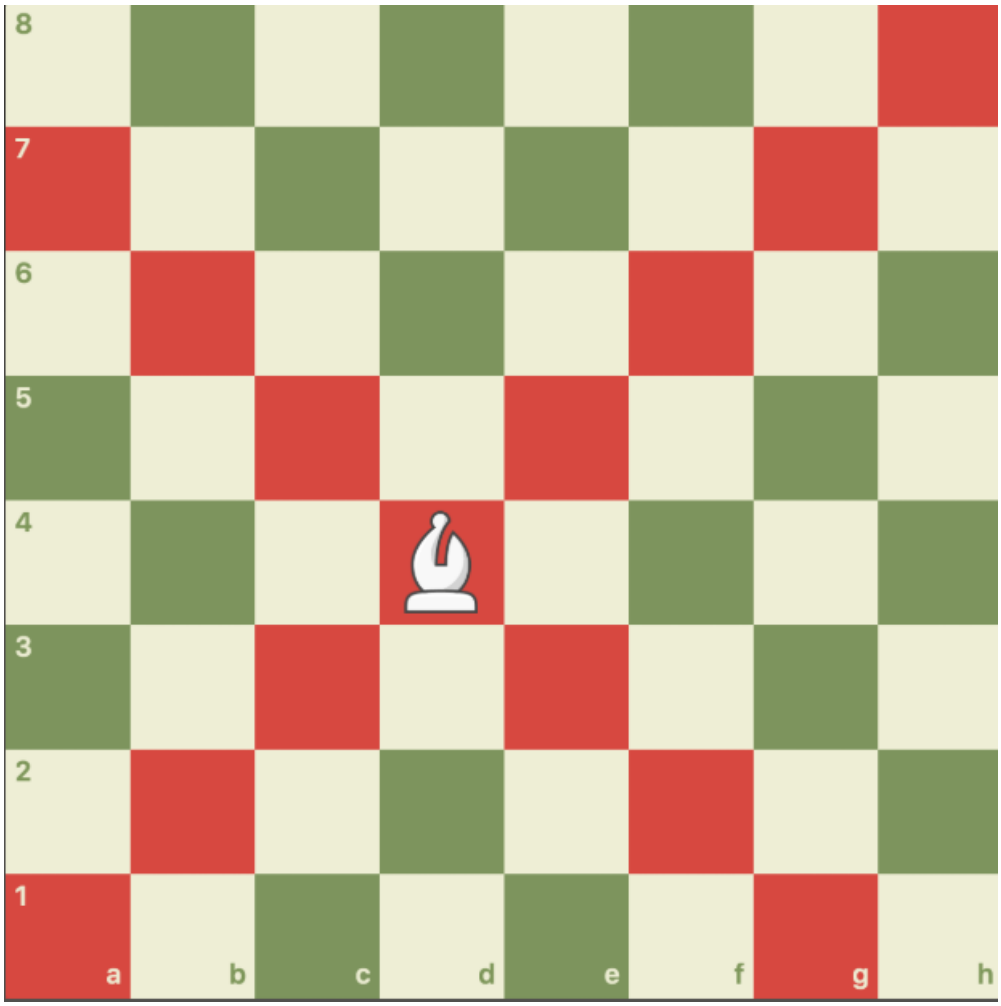
Kiểm tra đường đi không bị chặn:

-Đảm bảo rằng không có quân cờ nào trên đường giữa ô hiện tại và ô mục tiêu.

3.3.5 Tượng (Bishop):

Đi chéo trên bàn cờ, không giới hạn số ô.

Ăn quân theo đường chéo.



Hình 3-10 Cách thức di chuyển của Bishop

3.3.5.1 Mã giả

// Hàm kiểm tra tính hợp lệ của nước đi của quân Tượng

function canMove(targetCol, targetRow):

// Kiểm tra ô mục tiêu có nằm trong bàn cờ

```
if !isWithinBoard(targetCol, targetRow):
    return false
// Kiểm tra nước đi không trùng vị trí hiện tại
if isSameSquare(targetCol, targetRow):
    return false
// Kiểm tra di chuyển trên đường chéo
if ABS(targetCol - currentCol) != ABS(targetRow - currentRow):
    return false
// Kiểm tra tính hợp lệ của ô mục tiêu
if !isValidSquare(targetCol, targetRow):
    return false
// Kiểm tra không có quân cờ chắn trên đường chéo
if pieceIsOnDiagonalLine(targetCol, targetRow):
    return false
// Nếu tất cả điều kiện đều thỏa mãn, nước đi hợp lệ
return true
```

3.3.5.2 Giải thích các bước

Kiểm tra giới hạn bàn cờ:

Quân Tượng chỉ có thể di chuyển tới các ô nằm trong phạm vi bàn cờ 8x8.

Kiểm tra nước đi khác ô hiện tại:

Nước đi phải dẫn tới ô mục tiêu khác với vị trí hiện tại.

Kiểm tra đường chéo:

Đường chéo được xác định dựa trên sự thay đổi bằng nhau về cột và hàng. Nếu điều kiện không thỏa mãn, nước đi không hợp lệ.

Kiểm tra ô mục tiêu:

Ô mục tiêu phải hợp lệ, không chứa quân cờ cùng màu.

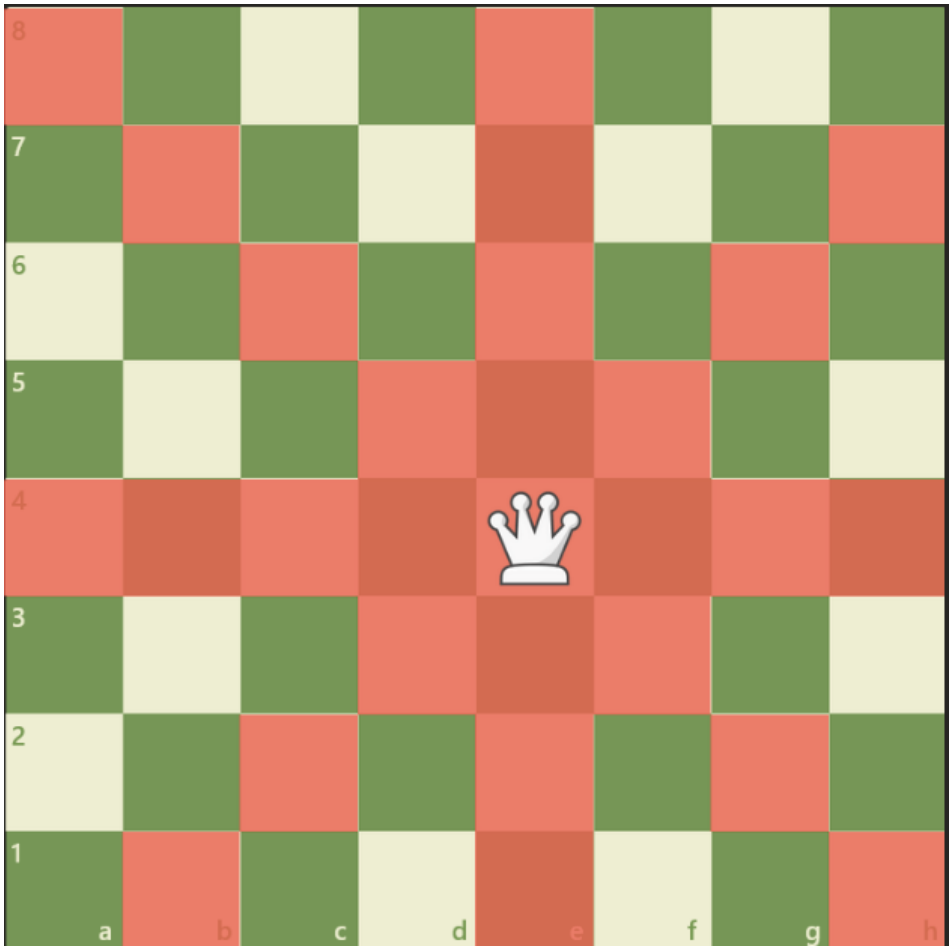
Kiểm tra không có quân cờ chắn:

Đảm bảo rằng không có quân nào nằm trên đường chéo giữa vị trí hiện tại và ô mục tiêu.

3.3.6 Hậu (Queen):

Là quân mạnh nhất, kết hợp nước đi của Xe (Rook) và Tịnh (Bishop).

Có thể đi thẳng hoặc chéo, không giới hạn số ô.



Hình 3-11 Cách thức di chuyển của Queen

3.3.6.1 Mã giả

```
// Hàm kiểm tra tính hợp lệ của nước đi của quân Hậu
function canMove(targetCol, targetRow):
  // Kiểm tra ô mục tiêu nằm trong bàn cờ
  if !isWithinBoard(targetCol, targetRow):
    return false
  // Kiểm tra nước đi không trùng vị trí hiện tại
  if isSameSquare(targetCol, targetRow):
    return false
  // Kiểm tra di chuyển theo hàng ngang hoặc dọc
  if targetCol == currentCol OR targetRow == currentRow:
    if isValidSquare(targetCol, targetRow) AND !pieceIsOnStraightLine(targetCol, targetRow):
      return true
  // Kiểm tra di chuyển theo đường chéo
  if ABS(targetCol - currentCol) == ABS(targetRow - currentRow):
    if isValidSquare(targetCol, targetRow) AND !pieceIsOnDiagonalLine(targetCol, targetRow):
      return true
  // Nếu không thỏa mãn bất kỳ điều kiện nào, nước đi không hợp lệ
  return false
```

3.3.6.2 Giải thích các bước

Kiểm tra ô mục tiêu nằm trong bàn cờ:

Nước đi phải nằm trong phạm vi hợp lệ của bàn cờ (8x8).

Kiểm tra nước đi khác ô hiện tại:

-Quân Hậu không được đứng yên tại vị trí hiện tại, nước đi phải dẫn tới ô mới.

Kiểm tra di chuyển theo hàng ngang hoặc dọc:

-Nếu giữ nguyên hàng hoặc giữ nguyên cột, kiểm tra:

Ô mục tiêu không chứa quân cùng màu (isValidSquare).

-Đường đi không bị chặn bởi quân cờ khác (!pieceIsOnStraightLine).

Kiểm tra di chuyển theo đường chéo:

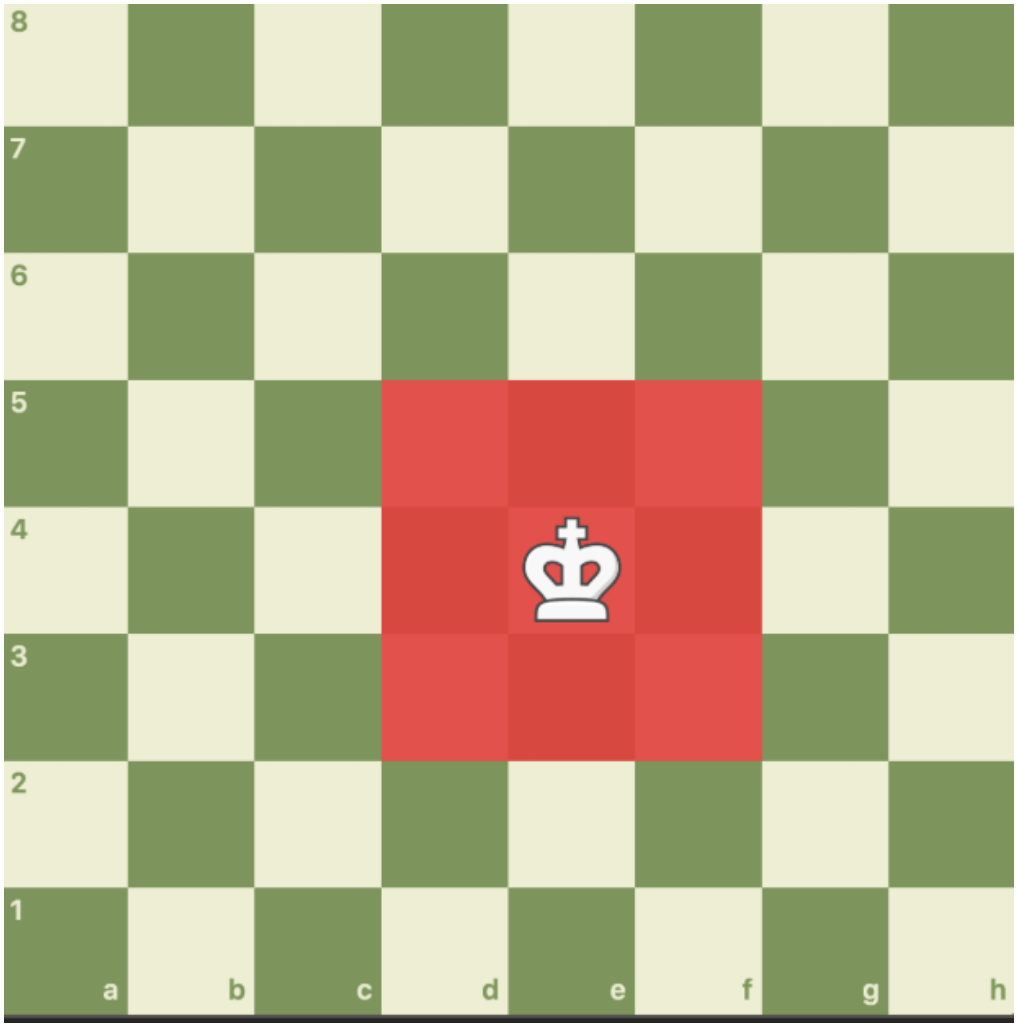
-Đường chéo được xác định bởi sự thay đổi bằng nhau giữa cột và hàng. Kiểm tra:

- Ô mục tiêu hợp lệ (isValidSquare).
- Đường đi không bị chắn bởi quân cờ khác (!pieceIsOnDiagonalLine).

3.3.7 Vua (King):

Đi và ăn trong phạm vi 1 ô xung quanh vị trí hiện tại.

Có thể nhập thành với Xe nếu thỏa mãn điều kiện.



Hình 3-12 Cách thức di chuyển của vua

3.3.7.1 Mã giả

// Hàm kiểm tra tính hợp lệ của nước đi của quân Vua

function canMove(targetCol, targetRow):

// Kiểm tra ô mục tiêu có nằm trong bàn cờ

if !isWithinBoard(targetCol, targetRow):

return false

// Di chuyển bình thường (1 ô quanh vị trí hiện tại)

if ABS(targetCol - currentCol) + ABS(targetRow - currentRow) == 1 OR

```
            ABS(targetCol - currentCol) * ABS(targetRow - currentRow) == 1:
            if isValidSquare(targetCol, targetRow):
                return true
        // Nhập thành (Castling)
        if !hasMoved:
            // Nhập thành bên phải
            if targetCol == currentCol + 2 AND targetRow == currentRow:
                if !pieceIsOnStraightLine(targetCol, targetRow):
                    for piece in simulatedPieces:
                        if piece.col == currentCol + 3 AND piece.row == currentRow AND !piece.hasMoved:
                            castlingPiece = piece
                            return true
            // Nhập thành bên trái
            if targetCol == currentCol - 2 AND targetRow == currentRow:
                if !pieceIsOnStraightLine(targetCol, targetRow):
                    for piece in simulatedPieces:
                        if piece.col == currentCol - 4 AND piece.row == currentRow AND !piece.hasMoved:
                            castlingPiece = piece
                            return true
        // Nếu không thỏa mãn bất kỳ điều kiện nào, nước đi không hợp lệ
        return false
```

3.3.7.2 Giải thích các bước

Kiểm tra ô mục tiêu nằm trong bàn cờ:

-Nước đi phải dẫn tới một ô nằm trong phạm vi bàn cờ (8x8).

Di chuyển bình thường (1 ô):

-Kiểm tra khoảng cách cột và hàng giữa vị trí hiện tại và ô mục tiêu để đảm bảo nước đi nằm trong phạm vi 1 ô quanh Vua.

-Xác nhận ô mục tiêu hợp lệ bằng cách kiểm tra quân cờ tại đó.

Nhập thành bên phải:

-Vua di chuyển 2 ô sang phải, đảm bảo không có quân nào chắn giữa Vua và Xe.

-Kiểm tra quân Xe ở vị trí hợp lệ, chưa di chuyển.

Nhập thành bên trái:

-Tương tự, kiểm tra điều kiện cho việc nhập thành sang trái.

3.3.8 Trạng thái đặc biệt của quân vua

Tập hợp các thuật toán để kiểm tra các trạng thái đặc biệt trong cờ vua như chiếu tướng (Check), chiếu bí (Checkmate), vua còn sống (King Alive), hòa cờ (Stalemate), và cách mô phỏng các nước đi hợp lệ.

3.3.8.1 isKingAlive()

Mục đích:

Kiểm tra xem quân Vua của người chơi hiện tại có còn tồn tại trên bàn cờ hay không.

Phân tích logic:

- Lấy quân Vua hiện tại bằng hàm getKing(true).
- Nếu quân Vua không tồn tại (king == null), trả về false.
- Nếu quân Vua tồn tại, trả về true.

Mã giả (Pseudo Code):

// Kiểm tra Vua có còn trên bàn cờ không

```
function isKingAlive():
   king = getKing(currentPlayer)
   if king == null:
      return false
   return true
```

3.3.8.2 isIllegal(Piece king)

Mục đích:

Kiểm tra xem trạng thái hiện tại của bàn cờ có phải là trạng thái bất hợp lệ (Vua bị tấn công) hay không.

Phân tích logic:

- Nếu quân Vua (king) bị bất kỳ quân cờ đối thủ nào tấn công (quân đối thủ có thể di chuyển tới vị trí của Vua), trạng thái bàn cờ được xem là bất hợp lệ và trả về true.
- Ngược lại, trả về false.

Mã giả (Pseudo Code):

// Kiểm tra trạng thái bàn cờ có bất hợp lệ không

```
function isIllegal(king):
   for each piece in simPieces:
      if piece != king AND piece.color != king.color AND piece.canMove(king.col, king.row):
         return true
   return false
```

3.3.8.3 opponentCanCaptureKing()

Mục đích:

Kiểm tra xem đối thủ có thể tấn công quân Vua của người chơi hay không.

Phân tích logic:

- Lấy vị trí quân Vua của đối thủ bằng hàm getKing(false).
- Nếu bất kỳ quân nào của người chơi có thể di chuyển tới vị trí của quân Vua đối thủ, trả về true.
- Ngược lại, trả về false.

Mã giả (Pseudo Code):

```
// Kiểm tra xem đối thủ có thể tấn công Vua không
function opponentCanCaptureKing():
    king = getKing(opponentPlayer)
    for each piece in simPieces:
        if piece.color != king.color AND piece.canMove(king.col, king.row):
            return true
    return false
```

3.3.8.4 isCheckMate()

Mục đích:

Kiểm tra xem người chơi hiện tại có đang bị chiếu bí (Checkmate) hay không.

Phân tích logic:

- Kiểm tra xem quân Vua có thể di chuyển thoát khỏi trạng thái chiếu hay không (kingCanMove).
- Nếu có thể, không phải chiếu bí → Trả về false.
- Nếu Vua không thể thoát:

- Kiểm tra xem có quân cờ nào của người chơi có thể chặn nước đi chiếu bằng cách:
- Xác định các ô trên đường tấn công của quân đang chiếu.
- Duyệt qua tất cả các quân của người chơi xem có quân nào di chuyển tới các ô này hay không.

Mã giả (Pseudo Code):

// Kiểm tra xem có phải chiếu bí không

```
function isCheckMate():
    king = getKing(currentPlayer)
    if kingCanMove(king):
        return false
    // Kiểm tra các ô trên đường tấn công của quân chiếu
    for each square in pathBetween(checkingPiece, king):
        for each piece in simPieces:
            if piece.color == currentPlayer AND piece.canMove(square.col, square.row):
                return false
    return true
```

3.3.8.5 kingCanMove(Piece king)

Mục đích:

Kiểm tra xem quân Vua có thể di chuyển tới bất kỳ ô nào an toàn hay không.

Phân tích logic:

Giả lập các nước đi tiềm năng của quân Vua tới các ô xung quanh.

Kiểm tra:

- Ô mục tiêu nằm trong bàn cờ.

- Quân Vua có thể di chuyển tới đó.
- Trạng thái sau khi di chuyển không gây nguy hiểm (bị chiếu).

Mã giả (Pseudo Code):

// Kiểm tra nếu Vua có thể di chuyển

```
function kingCanMove(king):
    for each (colOffset, rowOffset) in allDirections:
        if isValidMove(king, colOffset, rowOffset):
            return true
    return false
```

3.3.8.6 isValidMove(Piece king, int colPlus, int rowPlus)

Mục đích:

Kiểm tra xem nước đi của quân Vua tới một ô cụ thể có hợp lệ và an toàn hay không.

Phân tích logic:

- Cập nhật vị trí tạm thời của Vua.
- Kiểm tra tính hợp lệ của nước đi và đảm bảo sau khi di chuyển, Vua không bị chiếu.
- Khôi phục lại vị trí ban đầu của Vua và trạng thái bàn cờ.

Mã giả (Pseudo Code):

// Kiểm tra nếu nước đi hợp lệ và an toàn

```
function isValidMove(king, colPlus, rowPlus):
    king.col += colPlus
    king.row += rowPlus
    if king.canMove(king.col, king.row) AND !isIllegal(king):
```

```
        resetPosition(king)
        return true
    resetPosition(king)
    return false
```

3.3.8.7 isStalemate()

Mục đích:

Kiểm tra trạng thái hòa cờ (Stalemate) khi một bên không thể thực hiện nước đi hợp lệ.

Phân tích logic:

- Đếm số lượng quân cờ còn lại trên bàn.
- Nếu chỉ còn quân Vua và Vua không thể di chuyển, đó là hòa cờ.
- Ngược lại, kiểm tra tất cả các nước đi hợp lệ khác.

Mã giả (Pseudo Code):

```
// Kiểm tra nếu hòa cờ
function isStalemate():
    if numberOfPieces(opponentPlayer) == 1:
        king = getKing(currentPlayer)
        return !kingCanMove(king)
    return false
```

3.3.9 Điều kiện kết thúc ván cờ:

Thắng cuộc:

Khi một bên chiếu bí vua của đối phương (Checkmate).

Thua cuộc:

Khi vua của bên mình bị chiếu bí.

Hòa (Draw):

Một bên không còn quân nào có thể di chuyển hợp lệ nhưng cũng không bị chiếu (vua không bị kiểm soát).

3.4 Đối thủ máy

3.4.1 Thuật toán hỗ trợ

3.4.1.1 getAllValidMoves()

Mục đích:

Xác định tất cả các nước đi hợp lệ của quân cờ hiện tại (dựa trên loại quân cờ như Tốt, Mã, Xe, Tịnh, Hậu hoặc Vua).

Phân loại logic di chuyển cho từng loại quân.

Phân tích logic:

- Tạo danh sách validMoves để lưu trữ các nước đi hợp lệ.
- Sử dụng cấu trúc switch-case để gọi hàm xử lý tương ứng với từng loại quân.
- Tốt: handlePawnMoves(validMoves)
- Mã: handleKnightMoves(validMoves)
- Xe: handleRookMoves(validMoves)
- Tịnh: handleBishopMoves(validMoves)
- Hậu: handleQueenMoves(validMoves)
- Vua: handleKingMoves(validMoves)

Mã giả:

```
function getAllValidMoves():
    validMoves = []
    switch (type):
        case PAWN:
```

```
            handlePawnMoves(validMoves)
        case KNIGHT:
            handleKnightMoves(validMoves)
        case ROOK:
            handleRookMoves(validMoves)
        case BISHOP:
            handleBishopMoves(validMoves)
        case QUEEN:
            handleQueenMoves(validMoves)
        case KING:
            handleKingMoves(validMoves)
    return validMoves
```

3.4.1.2 handleLinearMoves()

Mục đích:

Xử lý nước đi theo đường thẳng hoặc đường chéo (dành cho Xe, Tịnh và Hậu).

Kiểm tra các ô mục tiêu trong các hướng được cung cấp (dọc, ngang hoặc chéo) cho đến khi gặp giới hạn bàn cờ hoặc quân cờ khác.

Phân tích logic:

- Lặp qua từng hướng di chuyển trong directions.
- Duyệt qua các ô trong từng hướng:
- Kiểm tra nếu ô ngoài phạm vi bàn cờ (isWithThinBoard), dừng kiểm tra hướng hiện tại.
- Nếu ô chứa quân đối thủ, thêm vào danh sách nước đi hợp lệ, rồi dừng hướng đó.
- Nếu ô trống, thêm vào danh sách nước đi hợp lệ.

- Ngừng kiểm tra nếu gặp quân cờ cùng màu hoặc quân bất kỳ trên đường đi.

Mã giả:

```
function handleLinearMoves(validMoves, directions):
  for each direction in directions:
    targetCol = col
    targetRow = row
    while true:
      targetCol += direction[0]
      targetRow += direction[1]
      if !isWithinBoard(targetCol, targetRow):
        break
      piece = getPieceAt(targetCol, targetRow)
      if piece != null:
        if piece.color != this.color:
          validMoves.add([targetCol, targetRow])
        break
      validMoves.add([targetCol, targetRow])
```

3.4.1.3 Hàm xử lý cho từng loại quân

a/Tốt (handlePawnMoves)

Di chuyển thẳng: Tốt di chuyển thẳng về phía trước 1 hoặc 2 ô tùy trạng thái (đi 2 ô khi chưa di chuyển lần nào).

Bắt quân: Tốt ăn quân theo đường chéo.

En Passant: Kiểm tra và xử lý nước đi đặc biệt En Passant.

Thăng cấp: Tốt được tự động thăng cấp thành Hậu nếu đạt hàng cuối.

Mã giả:

```
function handlePawnMoves(validMoves):
    direction = (color == WHITE) ? -1 : 1
    // Di chuyển thẳng (1 ô)
    forwardRow = row + direction
    if isWithinBoard(col, forwardRow) AND getPieceAt(col, forwardRow) == null:
        validMoves.add([col, forwardRow])
        // Di chuyển thẳng (2 ô)
        if !moved:
            doubleForwardRow = row + (2 * direction)
            if isWithinBoard(col, doubleForwardRow) AND getPieceAt(col, doubleForwardRow) == null:
                validMoves.add([col, doubleForwardRow])
    // Bắt quân chéo
    leftDiagCol = col - 1
    rightDiagCol = col + 1
    if isWithinBoard(leftDiagCol, forwardRow):
        targetPiece = getPieceAt(leftDiagCol, forwardRow)
        if targetPiece != null AND targetPiece.color != this.color:
            validMoves.add([leftDiagCol, forwardRow])
    if isWithinBoard(rightDiagCol, forwardRow):
        targetPiece = getPieceAt(rightDiagCol, forwardRow)
```

```
            if targetPiece != null AND targetPiece.color != this.color:
                validMoves.add([rightDiagCol, forwardRow]
        // En Passant
    if (row == (color == GamePanel.WHITE ? 3 : 4)) { // Check if the pawn is in the correct row for en passant
                // En passant Trái
                if (isWithThinBoard(leftDiagCol, forwardRow)) {
                    Piece adjacentPiece = getPieceAt(leftDiagCol, row);
                    if (adjacentPiece != null && adjacentPiece.type == Type.PAWN &&
                        adjacentPiece.color != color && adjacentPiece.twoStepped) {
                        validMoves.add(new int[]{leftDiagCol, forwardRow});
                    }
                }
                // En passant Phải
                if (isWithThinBoard(rightDiagCol, forwardRow)) {
                    Piece adjacentPiece = getPieceAt(rightDiagCol, row);
                    if (adjacentPiece != null && adjacentPiece.type == Type.PAWN
                        adjacentPiece.color != color && adjacentPiece.twoStepped) {
                        validMoves.add(new int[]{rightDiagCol, forwardRow});
                    }
                }
            }
```

b/Mã (handleKnightMoves)

Mã có thể nhảy tới tối đa 8 ô theo hình chữ "L".

Không bị cản bởi quân khác.

Mã giả:

```
function handleKnightMoves(validMoves):
    deltas = [
        [-2, -1], [-1, -2], [1, -2], [2, -1],
        [2, 1], [1, 2], [-1, 2], [-2, 1]
    ]
    for each delta in deltas:
        targetCol = col + delta[0]
        targetRow = row + delta[1]
        if isWithinBoard(targetCol, targetRow) AND isValidSquare(targetCol, targetRow):
            validMoves.add([targetCol, targetRow])
```

c/Xe (handleRookMoves)

Di chuyển trên hàng ngang và dọc.

Sử dụng handleLinearMoves() với các hướng {[1, 0], [-1, 0], [0, 1], [0, -1]}.

d/Tịnh (handleBishopMoves)

Di chuyển trên đường chéo.

Sử dụng handleLinearMoves() với các hướng {[1, 1], [-1, 1], [1, -1], [-1, -1]}.

e/Hậu (handleQueenMoves)

Kết hợp cách di chuyển của Xe và Tịnh.

Sử dụng handleLinearMoves() với cả 8 hướng.

f/Vua (handleKingMoves)

Di chuyển 1 ô quanh vị trí hiện tại.

Kiểm tra tính hợp lệ của từng ô mục tiêu.

Mã giả:

```
function handleKingMoves(validMoves):
    deltas = [
        [-1, -1], [-1, 0], [-1, 1],
        [0, -1], [0, 1],
        [1, -1], [1, 0], [1, 1]
    ]
    for each delta in deltas:
        targetCol = col + delta[0]
        targetRow = row + delta[1]
        if isWithinBoard(targetCol, targetRow) AND isValidSquare(targetCol, targetRow):
            validMoves.add([targetCol, targetRow])
```

3.4.1.4 undoMoveOnBoard

Mục đích:

Khôi phục trạng thái bàn cờ sau khi giả lập một nước đi (hoặc thực hiện nước đi không hợp lệ).

Cần thiết cho các thuật toán như Minimax khi thử nghiệm các nước đi.

Logic hoạt động:

- Khôi phục vị trí gốc của quân cờ:
- Quân cờ được đưa trở về vị trí trước khi thực hiện nước đi.
- Khôi phục quân cờ bị bắt (nếu có):

- Nếu có quân cờ bị bắt, nó được thêm lại vào danh sách quân cờ trên bàn cờ.

3.4.1.5 deepCopyBoard

Mục đích:

Tạo một bản sao sâu của trạng thái bàn cờ.

Bản sao được sử dụng để giả lập trạng thái trong các thuật toán như Minimax, mà không làm ảnh hưởng đến trạng thái thực tế của bàn cờ.

Logic hoạt động:

- Duyệt qua từng quân cờ trên bàn cờ gốc (originalBoard):
- Tạo một đối tượng mới cho mỗi quân cờ bằng constructor sao chép (copy constructor).
- Trả về bản sao bàn cờ:
- Danh sách boardCopy chứa các quân cờ giống hệt như trạng thái hiện tại nhưng độc lập.

3.4.1.6 evaluateBoard

Mục đích:

Đánh giá trạng thái bàn cờ và trả về một điểm số để xác định lợi thế cho AI hoặc người chơi.

Điểm số dương có lợi cho AI (BLACK), điểm số âm có lợi cho người chơi (WHITE).

Logic hoạt động:

Khởi tạo điểm số:

- Biến score dùng để tích lũy giá trị của các quân cờ trên bàn.
- Lặp qua tất cả các quân cờ:
- Giá trị quân cờ được tính dựa trên hàm getPieceValue.
- Quân đen (AI) tăng điểm số, quân trắng (người chơi) giảm điểm số.

3.4.1.7 getPieceValue

Mục đích:

Trả về giá trị số tương ứng với từng loại quân cờ.

Logic hoạt động:

- Xử lý các trường hợp null:
- Nếu quân cờ không tồn tại, trả về giá trị 0.
- Trả về giá trị dựa trên loại quân:

Giá trị các quân cờ phản ánh tầm quan trọng của chúng trong trò chơi:

- Tốt (Pawn): 1
- Mã (Knight) và Tịnh (Bishop): 3
- Xe (Rook): 5
- Hậu (Queen): 9
- Vua (King): 100 (vì vua là quân quan trọng nhất).

3.4.1.8 synchronizePieces

Mục đích:

Đồng bộ danh sách quân cờ mô phỏng (simPieces) với danh sách thực tế (pieces).

Loại bỏ quân cờ không hợp lệ hoặc nằm ngoài bàn cờ.

Logic hoạt động:

- Duyệt qua danh sách pieces:
- Chỉ thêm các quân cờ có vị trí hợp lệ (nằm trong bàn cờ).
- Thay thế danh sách simPieces:
- Cập nhật simPieces với danh sách mới.

3.4.1.9 getPieceAt

Mục đích:

Tìm và trả về quân cờ tại vị trí cụ thể trên bàn cờ.

Logic hoạt động:

- Lặp qua danh sách quân cờ:
- So sánh cột và hàng của mỗi quân cờ với vị trí mục tiêu.
- Trả về quân cờ nếu tìm thấy:
- Nếu tìm thấy quân cờ, trả về quân cờ đó.
- Nếu không tìm thấy, trả về null.

3.4.2 Thực hiện bước đi sử dụng Min-Max

3.4.2.1 Phương thức performAIMove()

Mục đích:

Xác định nước đi tốt nhất cho AI dựa trên thuật toán Minimax.

Thực hiện nước đi đó trên bàn cờ thực.

Các bước chính:

Đồng bộ trạng thái bàn cờ (synchronizePieces()):

Đảm bảo danh sách simPieces (bàn cờ mô phỏng) và pieces (bàn cờ thực) đồng nhất.

Kiểm tra nước đi hợp lệ:

Nếu không có nước đi hợp lệ, trò chơi kết thúc và người chơi thắng.

Tìm nước đi tốt nhất:

- Sử dụng thuật toán Minimax với độ sâu (depth) cố định (mặc định là 3).
- Lặp qua tất cả các quân cờ của AI (màu đen) và tính điểm cho từng nước đi có thể.
- Cập nhật nước đi tốt nhất dựa trên điểm trả về từ Minimax.
- Thực hiện nước đi tốt nhất:
- Nếu xác định được nước đi tốt nhất, AI thực hiện nước đi và đồng bộ trạng thái bàn cờ.

- Nếu không thể tìm thấy nước đi hợp lệ, kết thúc trò chơi.

Phương thức thực hiện

- Deep Copy: Sử dụng deepCopyBoard() để mô phỏng trạng thái bàn cờ khi thử nghiệm các nước đi.
- Undo Move: Dùng undoMoveOnBoard() để khôi phục trạng thái bàn cờ sau khi giả lập.

3.4.2.2 Phương thức minimax()

Mục đích:

Tìm điểm số tối ưu cho AI (màu đen) bằng cách duyệt qua các trạng thái bàn cờ có thể xảy ra.

Minimax giúp AI quyết định nước đi dựa trên giả lập:

AI cố gắng tối đa hóa (maximize) lợi thế của mình.

Người chơi cố gắng tối thiểu hóa (minimize) lợi thế của AI.

Cắt tỉa Alpha-Beta (Alpha-Beta Pruning):

Alpha: Giá trị điểm lớn nhất mà AI có thể đảm bảo.

Beta: Giá trị điểm nhỏ nhất mà người chơi có thể đảm bảo.

Cắt tỉa : Nếu tại bất kỳ điểm nào, Beta ≤ Alpha, dừng việc tiếp tục duyệt nhánh.

Các bước chính:

Trường hợp cơ sở (Base Case):

- Dừng đệ quy khi đạt độ sâu bằng 0 hoặc trò chơi kết thúc (gameOver).
- Trả về điểm đánh giá trạng thái bàn cờ bằng hàm evaluateBoard().

Trường hợp Maximizing (AI - BLACK):

- AI cố gắng tối đa hóa giá trị điểm.
- Thử tất cả các nước đi hợp lệ của quân cờ màu đen.
- Cập nhật giá trị lớn nhất (maxEval) và giá trị Alpha.

- Dừng cắt tỉa: Nếu Beta ≤ Alpha, dừng duyệt.

Trường hợp Minimizing (Player - WHITE):

- Người chơi cố gắng tối thiểu hóa giá trị điểm.
- Thử tất cả các nước đi hợp lệ của quân cờ màu trắng.
- Cập nhật giá trị nhỏ nhất (minEval) và giá trị Beta.
- Dừng cắt tỉa: Nếu Beta ≤ Alpha, dừng duyệt.

CHƯƠNG 4 Kết luận

4.1 Kết quả đạt được

- Đã hoàn thành việc thiết kế và xây dựng game cờ vua bằng ngôn ngữ lập trình java.

- Game có đầy đủ các tính năng cơ bản như di chuyển quân cờ, xét chiếu tướng và hiển thị người thắng cuộc.

- Tích hợp được trí tuệ nhân tạo đơn giản sử dụng thuật toán MinMax kèm với thuật toán cắt tỉa Alpha-Beta.

4.2 Hạn chế

Thời gian xử lý của AI đôi khi rất lâu.

Còn nhiều lỗi vẫn chưa phát hiện kịp thời do số lượng các bước đi của các quân cờ là khá lớn.

Giao diện còn thiếu nhiều tiện nghi.

4.3 Hướng phát triển

Tiếp tục dò soát các lỗi.

Cải thiển AI, kèm với chức năng lựa chọn độ khó,

Cải thiện giao diện người dùng, thêm âm thanh cho các hành động

TÀI LIỆU THAM KHẢO

RyiSnow, https://www.youtube.com/@RyiSnow

Chess, https://www.chess.com/lessons