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

BÁO CÁO ĐỒ ÁN MÔN HỌC

PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỒ ÁN: GAME CỜ VUA

Giáo viên hướng dẫn	Thầy Lê Khánh Duy
	Thầy Lê Trung Nghĩa
	Thầy Lý Duy Nam
Lớp	21_2
Nhóm thực hiện	09
Sinh viên thực hiện	21120102 – Nguyễn Trúc Nguyên
	21120150 – Nguyễn Song Toàn
	21120179 – Nguyễn Đặng Đăng Khoa
	21120186 – Lê Hữu Trí

Tp. Hồ Chí Minh, tháng 05 năm 2023

Lời cảm ơn

Nhóm chúng em xin cảm ơn thầy Lê Khánh Duy và thầy Lê Trung Nghĩa đã cung cấp cho chúng em những bài học bổ ích, có ý nghĩa thực tiễn để chúng em có thể thực hiện đồ án này. Chúng em cũng xin cảm ơn thầy Lý Duy Nam đã cung cấp những bài thực hành để chúng em vận dụng được những lý thuyết đã học và có những kỹ thuật lập trình phục vụ cho việc lập trình đồ án này.

Mục lục

Danh sách hình

Danh sách bảng

1. Thông tin đồ án

1.1. Môi trường phát triển

- Đồ án được phát triển trên hệ điều hành Windows.
- Đồ án sử dụng thư viện:
  - Các thư viện có sẵn trong C/ C++.
  - Thư viện SFML hỗ trợ làm giao diện người dùng.

1.2. Đánh giá mức độ hoàn thiện của các chức năng

Bảng 1: Đánh giá mức độ hoàn thiện các chức năng

Chức năng	**Mức độ hoàn thiện**
Cho phép họ chọn quân cờ, nhập vị trí nước đi tiếp theo thông qua bàn phím.	100%
Hiển thị bàn cờ và quân cờ lên màn hình console	100%
Cài đặt được logic của trò chơi (điều kiện thắng, nước đi các quân cờ,…)	100%
Lưu và chơi lại ván cờ trước đó	100%
Thao tác thông qua click chuột trên console	100%
Chế độ chơi với máy (random nước đi)	100%
Chế độ chơi với máy (chế độ máy khó)	0%
Replay: xem lại ván đấu vừa đấu	100%
Giao diện người dùng	100%
Âm thanh trò chơi (nhạc nền, di chuyển quân cờ, ăn quân cờ đối thương, kết thúc ván đấu,…)	100%
Undo/Redo	100%

1.3. Thông tin thành viên

Bảng 2: Danh sách thành viên của nhóm

MSSV	**Họ và tên**
21120102	Nguyễn Trúc Nguyên
21120150	Nguyễn Song Toàn
21120179	Nguyễn Đặng Đăng Khoa
21120186	Lê Hữu Trí

1.4. Phân công công việc và mức độ hoàn thành

Bảng 3: Phân công công việc và mức độ hoàn thành

Thành viên	**Công việc**	**Mức độ hoàn thành**
Nguyễn Trúc Nguyên	• Xây dựng lớp Queen, Bishop • Tham gia xây dựng lớp Board • Xây dựng chức năng chơi với máy • Tester • Viết báo cáo	100%
Nguyễn Song Toàn	• Xây dựng lớp King, Rook • Tham gia xây dựng lớp Board • Xây dựng chức năng SaveGame, LoadGame • Tester • Viết báo cáo • Vẽ sơ đồ UML	100%
Nguyễn Đặng Đăng Khoa	• Xây dựng lớp Knight, Pawn • Tham gia xây dựng lớp Board • Tester • Viết báo cáo • Tham gia thiết kế đồ họa	100%
Lê Hữu Trí	• Xây dựng cơ bản lớp Piece, lớp Board • Xây dựng lớp Game • Tìm kiếm âm thanh • Thiết kế đồ họa các nút, icon, giao diện người dùng • Thực hiện chức năng Undo/Redo • Tester	100%

2. Mô tả thiết kế của chương trình

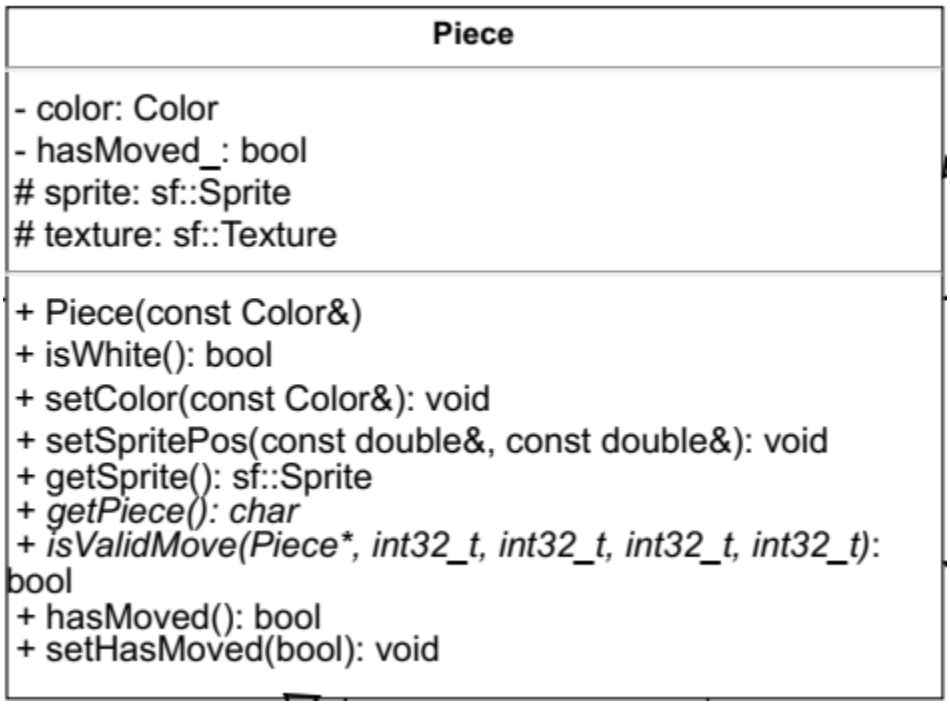
2.1. Sơ đồ UML

Vì kích thước khổ giấy có giới hạn nên nhóm chúng em không thể chèn sơ đồ UML vào báo cáo. Liên kết đến sơ đồ: Group09_UML.pdf

2.2. Các lớp của chương trình

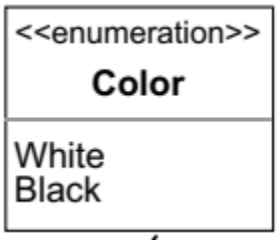
2.2.1. Lớp Piece

Lớp Piece có sơ đồ UML như sau:



Hình 1 - Sơ đồ UML của lớp Piece

Các thuộc tính của lớp Piece là thông tin về màu sắc của quân cờ, quân cờ đã di chuyển hay chưa, và hình ảnh, vị trí của quân cờ trên bàn cờ. Thông tin về màu sắc là class enum Color:



Hình 2 - Class enum Color

Các phương thức của lớp Piece cung cấp các hàm setter cho thuộc tính màu sắc, hình ảnh, đồ họa, đã di chuyển chưa.

Các phương thức của lớp Piece cung cấp:

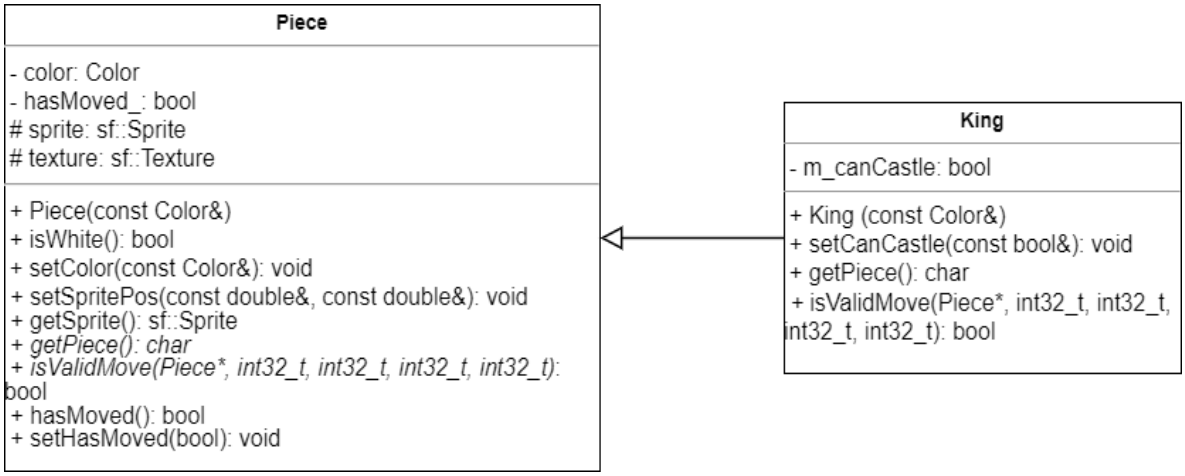
- Hàm kiểm tra quân cờ màu trắng hay không, hàm **isWhite().**

- Hàm lấy thông tin quân cờ **getPiece()**, tức là lấy ký tự đại diện cho quân cờ: quân Vua (K), quân Tốt (P), quân Hậu (Q), quân Mã (N), quân Xe (R), quân Tượng (B).
- Hàm **hasMoved()** - kiểm tra quân cờ đã di chuyển chưa.
- Phương thức thuần ảo ***isValidMove*** nhằm tận dụng tính đa hình. Phương thức này sẽ được cài đặt ở các lớp kế thừa lớp Piece. Phương thức này nhằm kiểm tra xem bước đi của quân cờ đó trên bàn cờ có hợp lệ không.

Các lớp sau kế thừa lớp Piece:

2.2.1.1. Lớp King

Sơ đồ UML của lớp King như sau:



Hình 3 - Sơ đồ UML lớp King

Lớp King kế thừa lớp Piece.

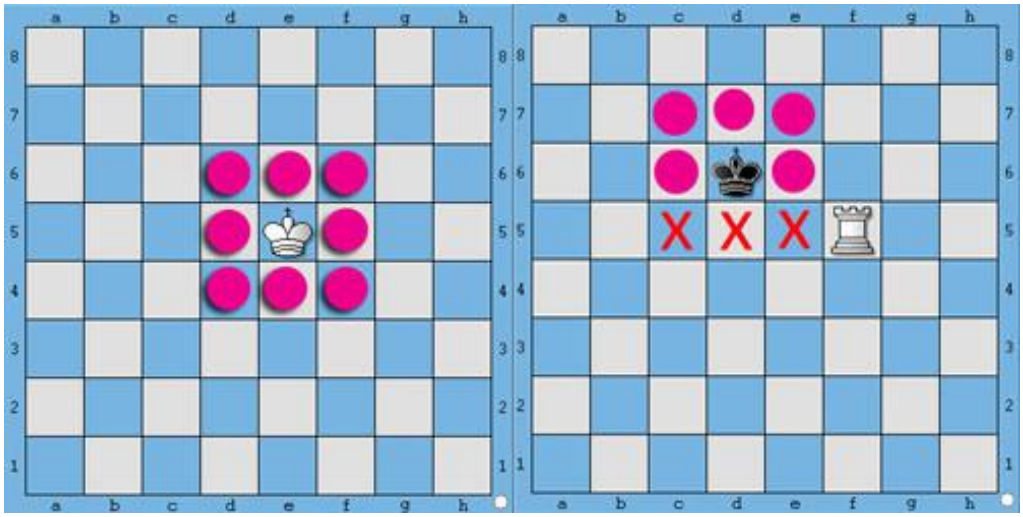
Các thuộc tính của lớp King kế thừa lớp cha Piece và có thêm thuộc tính để kiểm tra quân Vua có nhập thành được không.

Các phương thức của lớp King gồm:

- Hàm setter cho thuộc tính kiểm tra nhập thành
- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Vua là ký tự K.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Vua trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Vua như sau:

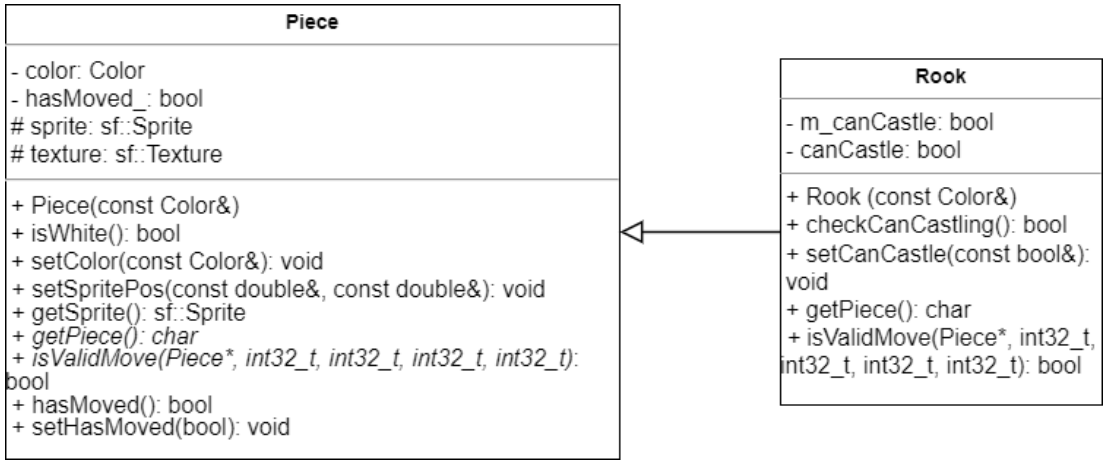
- Trong bàn cờ vua, quân vua là quân quan trọng nhất. Tuy nhiên, phạm vi di chuyển của quân vua lại khá hạn chế. Mỗi lượt đi, quân vua chỉ có thể di chuyển được 1 ô duy nhất về tất cả các hướng: trên, dưới, phải, trái, xéo. Quân vua có thể ăn quân của đối phương đang nằm ở ô mà mình sắp di chuyển tới. Khác với tất cả các quân cờ khác trong bàn cờ vua, quân vua không thể đi đến ô nằm trên phạm vi di chuyển của quân đối phương. Nếu đi sẽ bị tính là một lỗi kỹ thuật và buộc phải đi lại nước cờ đó.
- Các bước đi hợp lệ của quân Vua được thể hiện qua hình ảnh sau:



Hình 4 - Các bước đi hợp lệ của quân Vua.
Nguồn: thethaohcm.com.vn

2.2.1.2. Lớp Rook

Sơ đồ UML của lớp Rook như sau:



Hình 5 - Sơ đồ UML lớp Rook

Lớp Rook kế thừa lớp Piece.

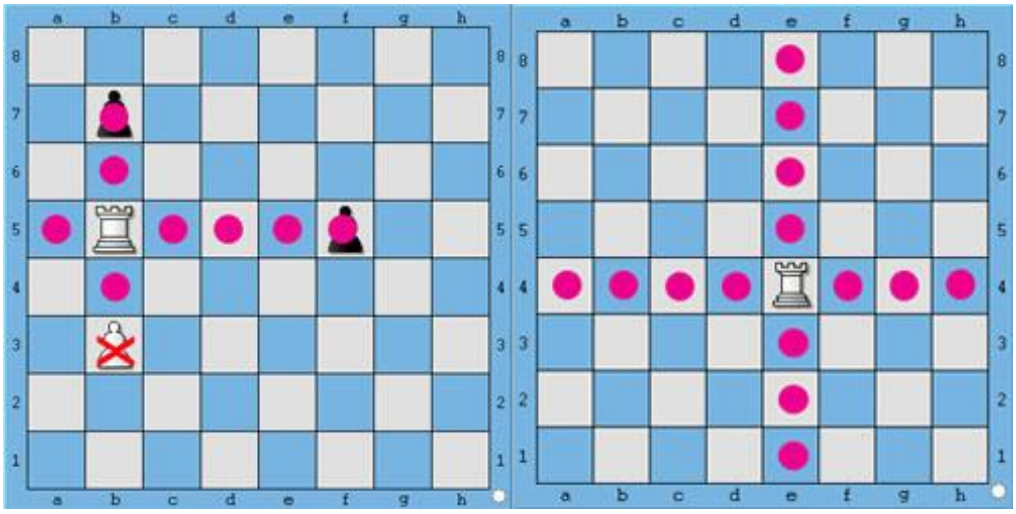
Các thuộc tính của lớp Rook kế thừa lớp cha Piece và có thêm thuộc tính để kiểm tra quân xe có nhập thành được không.

Các phương thức của lớp Rook gồm:

- Hàm setter cho thuộc tính kiểm tra nhập thành.
- Hàm checkCanCastling() để đánh dấu rằng quân xe có nhập thành được không?
- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Xe là ký tự R.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Xe trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Xe như sau:

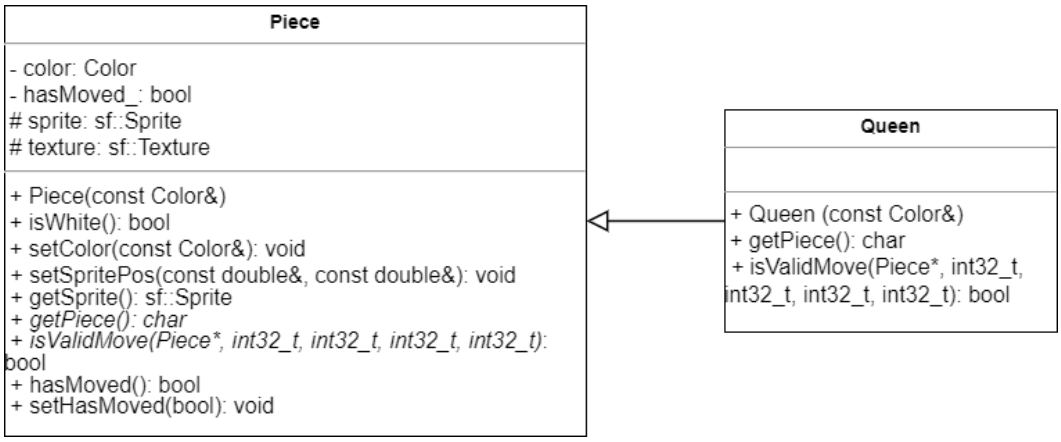
- Xe có quyền đi dọc hoặc đi ngang bao nhiêu ô tùy thích. Tuy nhiên, xe vẫn bị cản bởi quân mình đang nằm trên đường đi. Khi ăn quân, xe sẽ ăn bằng cách di chuyển đến đến vị trí mà quân đối phương đang chiếm giữ. Lấy quân cờ "xấu số" đó ra khỏi bàn cờ và đặt quân xe của mình ở đó.
- Các bước đi hợp lệ của quân Xe được thể hiện qua hình ảnh sau:



Hình 6 - Các bước đi hợp lệ của quân Xe.
Nguồn: thethaohcm.com.vn

2.2.1.3. Lớp Queen

Sơ đồ UML của lớp Queen như sau:



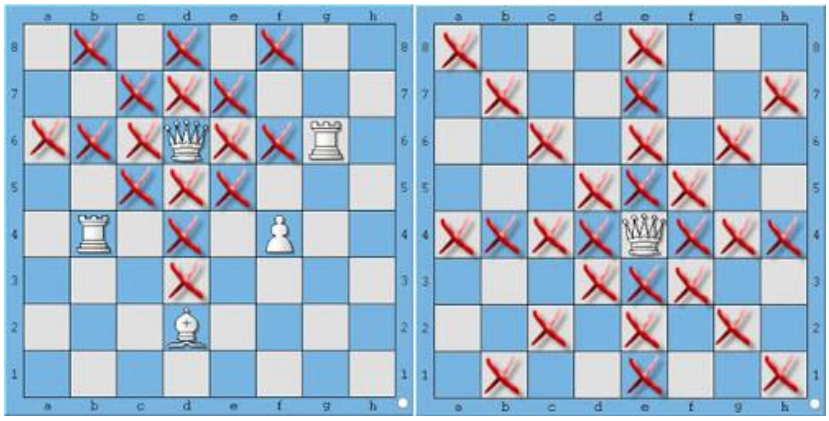
Hình 7 - Sơ đồ UML lớp Queen

Lớp Queen kế thừa lớp Piece.
Các thuộc tính của lớp Queen kế thừa lớp cha Piece.
Các phương thức của lớp Queen gồm:

- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Hậu là ký tự Q.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Hậu trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Hậu như sau:

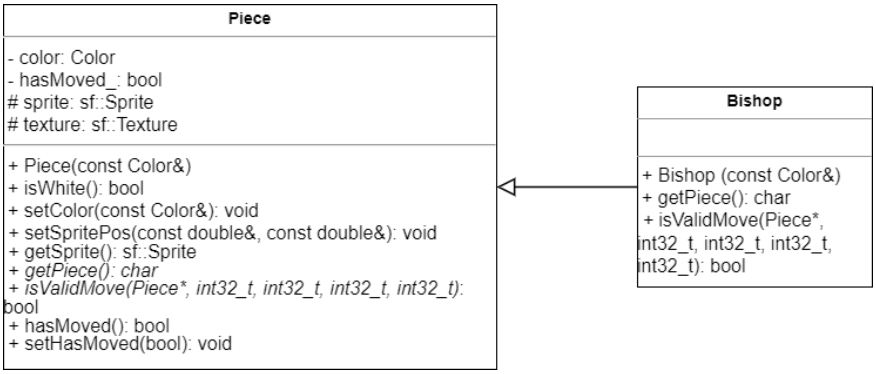
- Trong cờ vua hậu là quân có lối di chuyển linh hoạt nhất, vừa có thể đi như xe, vừa có thể di chuyển như tượng theo cả đường chéo và ngan, dọc. Vì vậy, hậu được xem là quân cờ mạnh nhất. Quân hậu có thể bị cản bởi quân mình nhưng có thể ăn bắt kỳ quân nào của đối phương nếu cản đường bằng các di chuyển đến và thay thế vị trí mà quân đó đang đứng.
- Các bước đi hợp lệ của quân Hậu được thể hiện qua hình ảnh sau:



Hình 8 - Các bước đi hợp lệ của quân Hậu.
Nguồn: thethaohcm.com.vn

2.2.1.4. Lớp Bishop

Sơ đồ UML của lớp Bishop như sau:



Hình 9 - Sơ đồ UML lớp Bishop

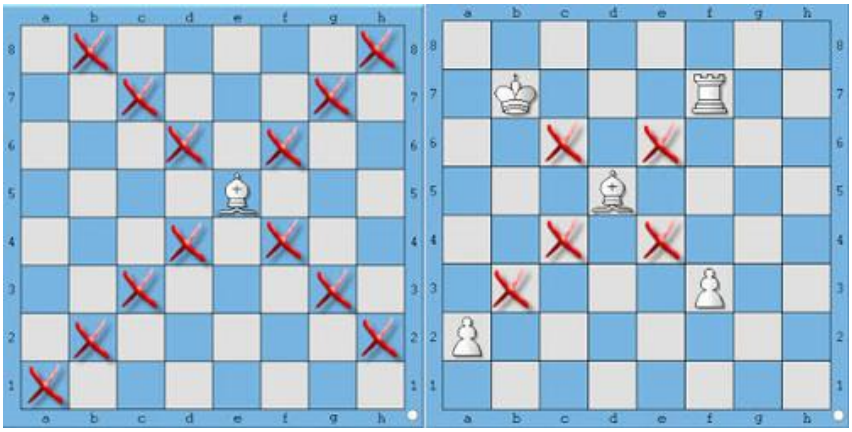
Lớp Bishop kế thừa lớp Piece.
Các thuộc tính của lớp Bishop kế thừa lớp cha Piece.
Các phương thức của lớp Bishop gồm:

- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Tượng là ký tự B.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Tượng trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Tượng như sau:

- Giống như quân xe nhưng khác ở chỗ là tượng di chuyển theo đường chéo.

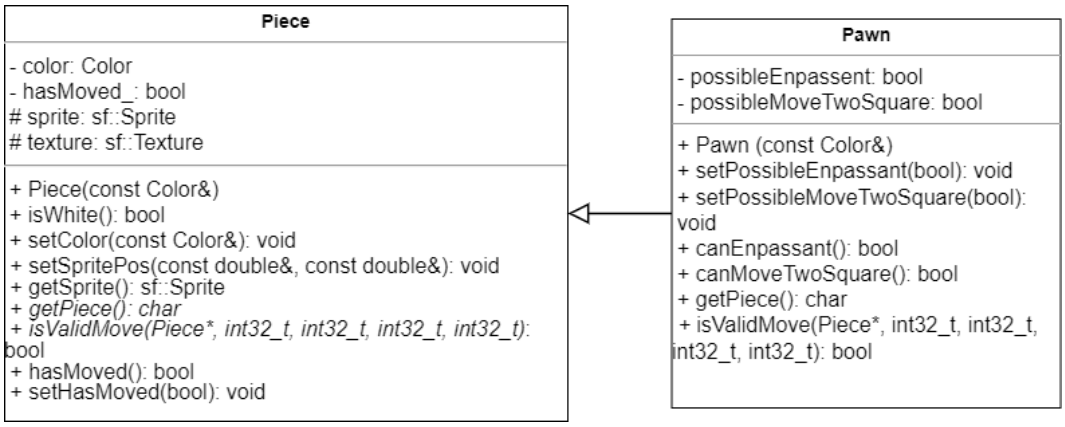
- Các bước đi hợp lệ của quân Tượng được thể hiện qua hình ảnh sau:



Hình 10 - Các bước đi hợp lệ của quân Tượng.
Nguồn: thethaohcm.com.vn

2.2.1.5. Lớp Pawn

Sơ đồ UML của lớp Pawn như sau:



Hình 11 - Sơ đồ UML lớp Pawn

Lớp Pawn kế thừa lớp Piece.

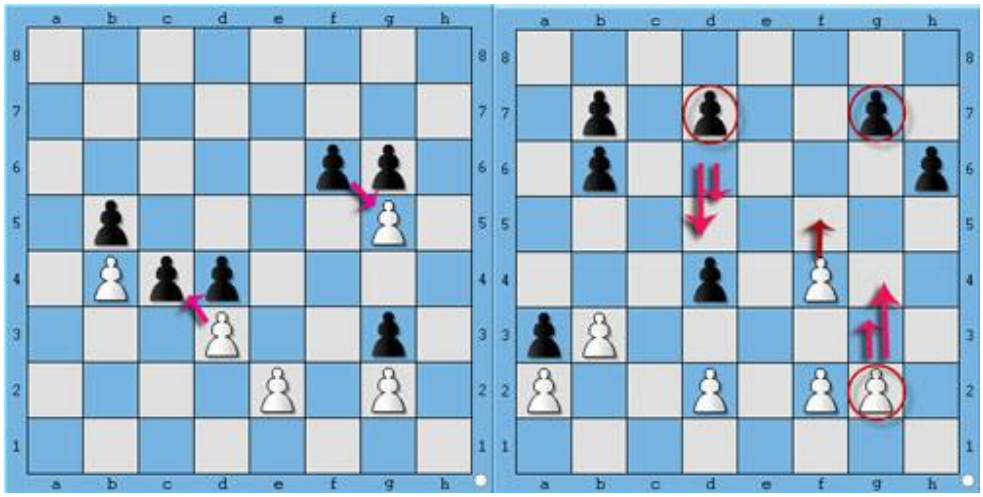
Các thuộc tính của lớp Pawn kế thừa lớp cha Piece và thêm thuộc tính **possibleEnpassant** để đánh dấu có thể bắt Tốt qua đường không và thuộc tính **possibleMoveTwoSquare** để đánh dấu có thể di chuyển 2 ô không.

Các phương thức của lớp Pawn gồm:

- Các hàm setter cho hai thuộc tính mới của lớp Pawn.
- Hàm **canEnpassant()** để kiểm tra quân Tốt có thể bắt Tốt qua đường hay không.
- Hàm **canMoveTwoSquare()** để kiểm tra xem quân Tốt có thể đi một lần hai ô được không.
- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Tốt là ký tự P.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Tốt trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Tốt như sau:

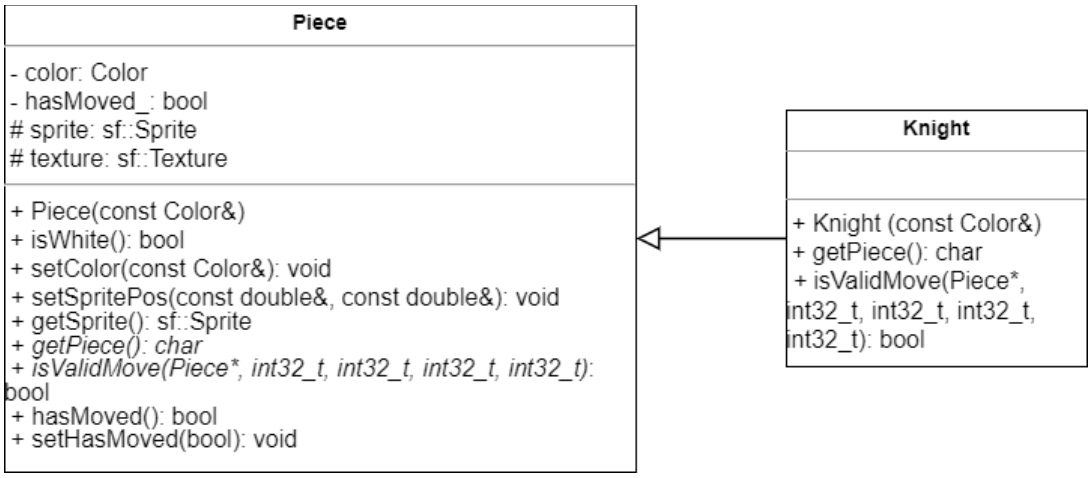
- Quân tốt là quân có vai vế "bé nhất" trong bàn cờ. Tốt chỉ có thể di chuyển về phía trước và không được lùi về phía sau. Ở mỗi lượt đi, tốt chỉ có thể tiến 2 ô ở nước đầu tiền và tiến 1 ô ở tất cả các nước cờ còn lại.
- Khác với các quân cờ khác trong cờ vua, tốt không thể ăn quân trên đường đường đi, mà chỉ có thể ăn chéo về phía trước. Nếu trước mặt quân tốt có quân cờ khác lúc này, tốt sẽ bị chặn và khổng thể tiến thêm về phía trước.
- Đặc biệt, mặc dù là quân yếu nhất về sức mạnh, nhưng nếu tốt đi được tới hàng cuối cùng của bàn cờ bên quân đối phương, tốt sẽ được phong cấp thành một trong các quân: Hậu, Xe, Mã hoặc Tượng tùy ý thích.
- Ngoài ăn chéo về phía trước, tốt còn có một cách ăn khác nữa đó là "bắt Tốt qua đường". Nếu quân tốt đang ở hàng thứ 5 tính từ phía mình, mà tốt đối phương ở 2 ô bên cạnh đi lên 2 ô thì các em có quyền ăn tốt đó. Nước bắt Tốt chỉ có thể thực hiện liền sau nước di chuyển Tốt của đối phương. Nếu thực hiện một nước đi khác thì nước đi tiếp theo sẽ không được bắt tốt quân đó nữa.
- Các bước đi hợp lệ của quân Tốt được thể hiện qua hình ảnh sau:



Hình 12 - Các bước đi hợp lệ của quân Tốt.
Nguồn: thethaohcm.com.vn

2.2.1.6. Lớp Knight

Sơ đồ UML của lớp Knight như sau:



Hình 13 - Sơ đồ UML lớp Knight

Lớp Knight kế thừa lớp Piece.

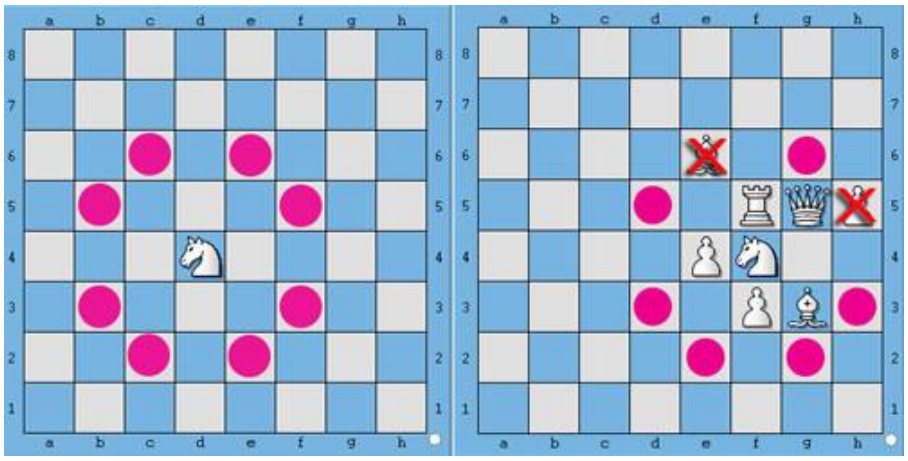
Các thuộc tính của lớp Knight kế thừa lớp cha Piece.

Các phương thức của lớp Knight gồm:

- Hàm **getPiece()** để lấy thông tin của ký tự đại diện quân Mã là ký tự N.
- Hàm **isValidMove** nhằm kiểm tra xem bước đi của quân Mã trên bàn cờ có hợp lệ không.

Luật di chuyển của quân Mã như sau:

- Quân mã di chuyển theo hình chữ L, mỗi nước đi gồm tổng cộn 3 ô: tiến 1 ô rồi quẹo trái hoặc quẹo phải 2 ô và ngược lại; tiến 2 ô rồi quẹo trái hoặc quẹo phải 1 ô và ngược lại.
- Khác với toàn bộ quân cờ trong bàn cờ vua, mã không bị cản bởi bất cứ quân nào và có thể nhảy qua tất cả các quân khác trên đường đi của mình.
- Các bước đi hợp lệ của quân Mã được thể hiện qua hình ảnh sau:



Hình 14 - Các bước đi hợp lệ của quân Mã.

Nguồn: thethaohcm.com.vn

2.2.2. Lớp Board

Hình 15 - Sơ đồ UML của lớp Board

Các vector chứa các loại quân cờ:

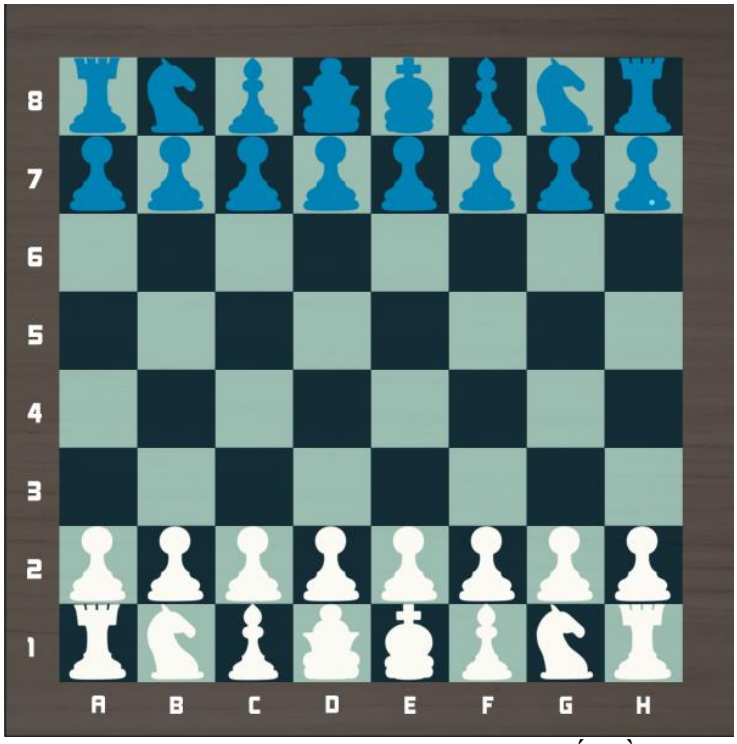
- whitePieces: vector chứa quân trắng
- blackPieces: vector chứa quân đen
- captured: các quân sau khi bị ăn thì được thêm vào vector này
- l_promote: chứa các quân tốt sau khi được phong cấp

Phương thức Board():

- Phương thức khởi tạo mặc định của bàn cờ, setup vị trí, set trạng thái hasMoved=false cho toàn bộ quân cờ.

Phương thức setBoard():

- Xóa vùng nhớ của các quân cờ. Khởi tạo vùng nhớ mới, setup lại vị trí ban đầu để bắt đầu ván đấu khác.



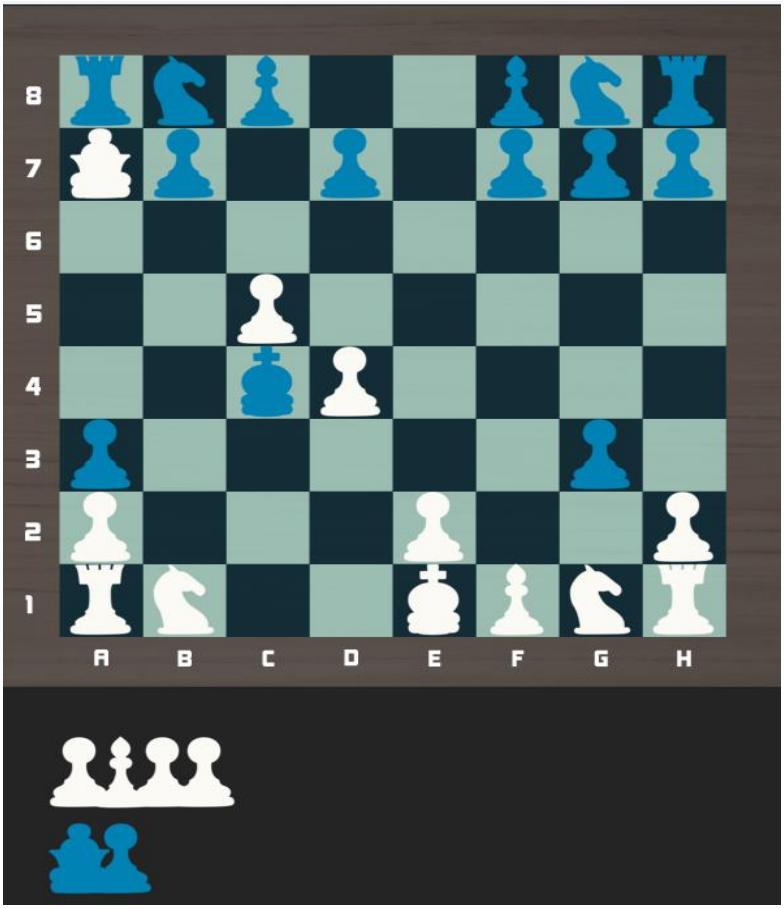
Hình 16 - Bàn cờ khi được set up lúc bắt đầu

Phương thức ~Board():

- Phương thức hủy của bàn cờ.
  - Xóa các quân trong các vector whitePieces, blackPieces, captured, l_promote
  - Set toàn bộ ô trên bàn cờ 8x8 về nullptr.

Phương thức draw:

- Phương thức này sẽ lấy tọa độ của sprite và hiển thị hình ảnh quân cờ trên bàn cờ 8x8, các quân cờ bị ăn sẽ được hiển thị tại khung bên dưới bàn cờ.



Hình 17 - Các quân trên bàn cờ và quân bị ăn

Phương thức getPiece:
- Tham số đầu vào là (x,y).
- Phương thức này sẽ trả về con trỏ Piece* trỏ đến vị trí ô (x,y) trên bàn cờ, nếu ô không chứa quân nào thì sẽ return về nullptr.

Phương thức checkKingSquare() và **checkRookSquare()**:
- Kiểm tra ô chứa vua hoặc xe.

Phương thức canCastle():
- Kiểm tra xem giữa vua và xe có quân nào chặn giữa và có thỏa điều kiện chưa di chuyển lần nào.
  - ❖ **Nếu cả ba phương thức trên đều trả về true thì đó chính là bước đi nhập thành.**

Phương thức getMoveType:
- Xác định MoveType của mỗi nước đi.

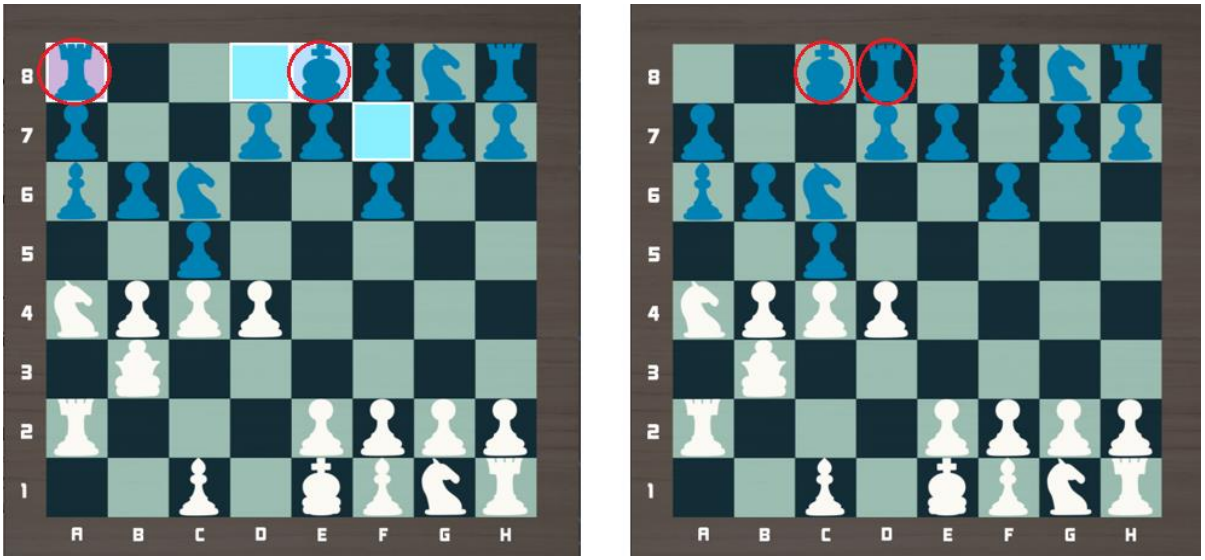
Enum MoveType: bao gồm các kiểu di chuyển trên bàn cờ
- Normal: nước đi bình thường từ ô này đến ô khác (không ăn quân).
- Capture: nước đi ăn quân địch

- Enpassant: bắt tốt qua đường
- LongCastling: nhập thành xa
- ShortCastling: nhập thành gần
- Promote: phong cấp tốt
- Capture_Promote: nước đi ăn quân địch rồi phong cấp.

➢ **Kiểu MoveType tương ứng sẽ được bật lên tại hàm canMove**

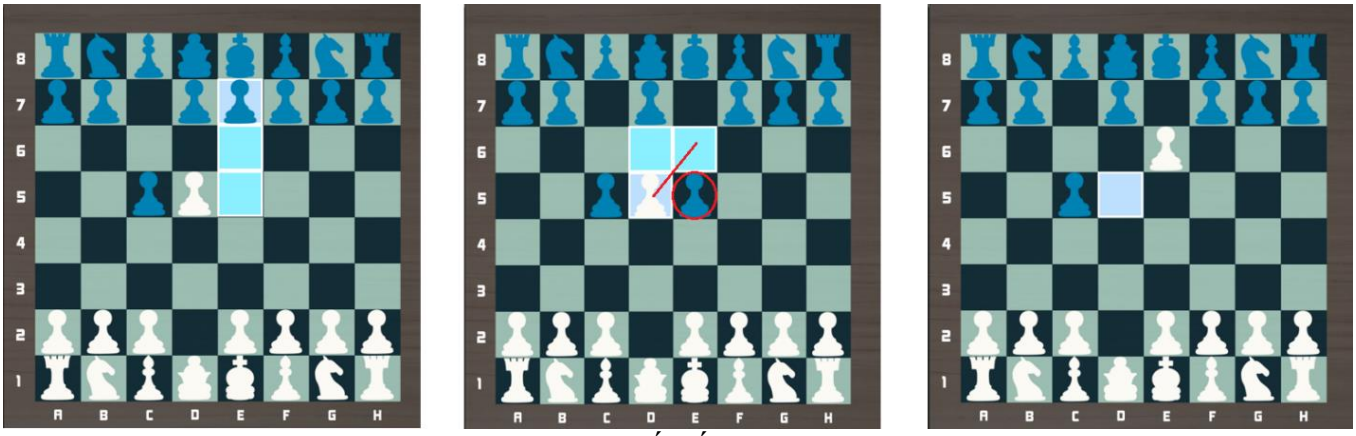
Phương thức canMove():

- Xác định nước đi có hợp lệ hay không, đồng thời bật lên MoveType tương ứng.
- Tham số đầu vào: tọa độ ô đầu, tọa độ ô đích và lượt whiteTurn.
- Bước đi hợp lệ được xác định theo các bước sau:
  - Kiểm tra tọa độ có nằm trong phạm vi bàn cờ (1-8) và có đang đúng lượt của quân mình hay chưa
  - Nếu ô đầu có quân và thỏa điều kiện trên thì tiếp tục xét
    - **Trường hợp nhập thành**: Nếu 2 ô chọn là K, R và 2 quân đều chưa di chuyển thì bật moveType là LongCastling hoặc ShortCastling.



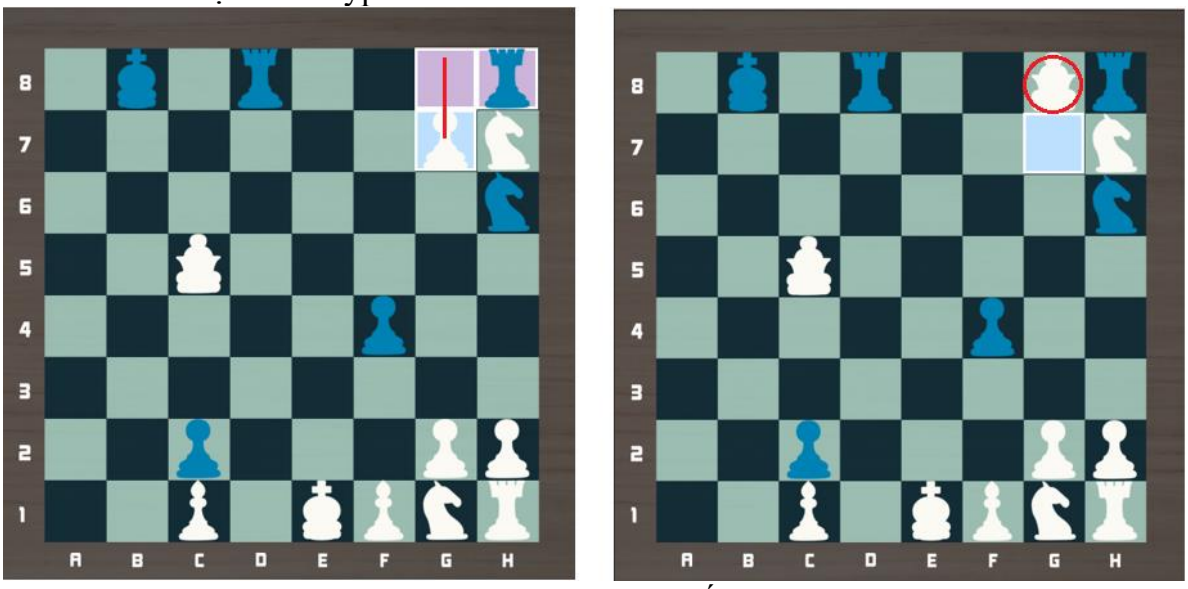
Hình 18 - Nước đi nhập thành

- Các nước đi bên dưới đều phải thông qua phương thức **Piece:: isValidMove** để xác định tính hợp lệ của nước đi đó.
- **Trường hợp bắt tốt qua đường**: nếu ô đầu chứa quân tốt và nó có thể bắt tốt thì bật moveType là Enpassant.



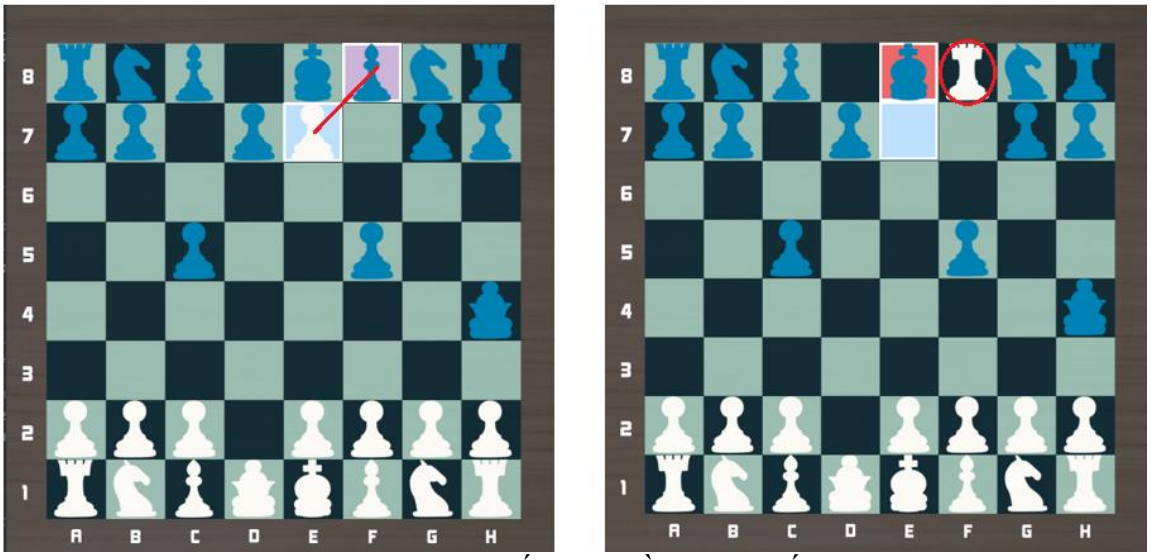
Hình 19 - nước đi bắt tốt qua đường

- **Trường hợp phong cấp:** nếu tốt đi đến dòng cuối bên kia bàn cờ thì bật moveType là Promote.



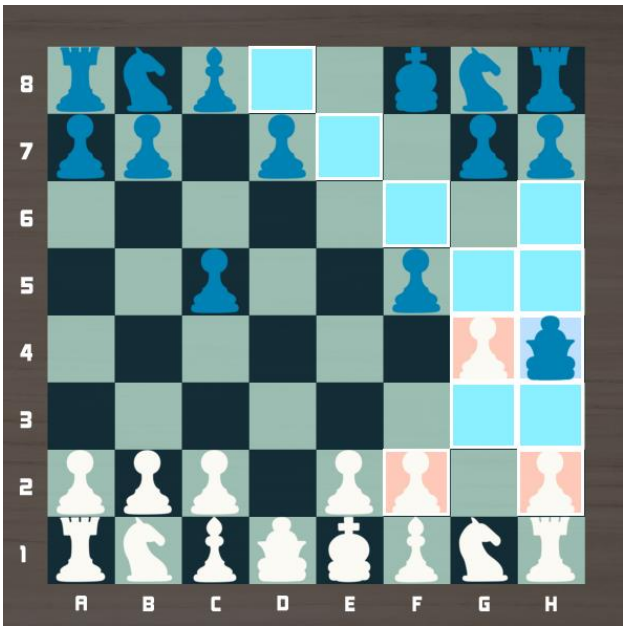
Hình 20 - Nước đi phong cấp

- **Trường hợp ăn quân rồi phong cấp**: nếu con tốt ăn chéo tại dòng cuối bên kia bàn cờ thì bật moveType là Capture_Promote.



Hình 21 - Nước đi bắt quân rồi phong cấp

- **Các trường hợp còn lại**: nếu ô đích có chứa quân thì có nghĩa đó là nước đi ăn quân, bật moveType là Capture. Ngược lại, moveType sẽ là Nornal.



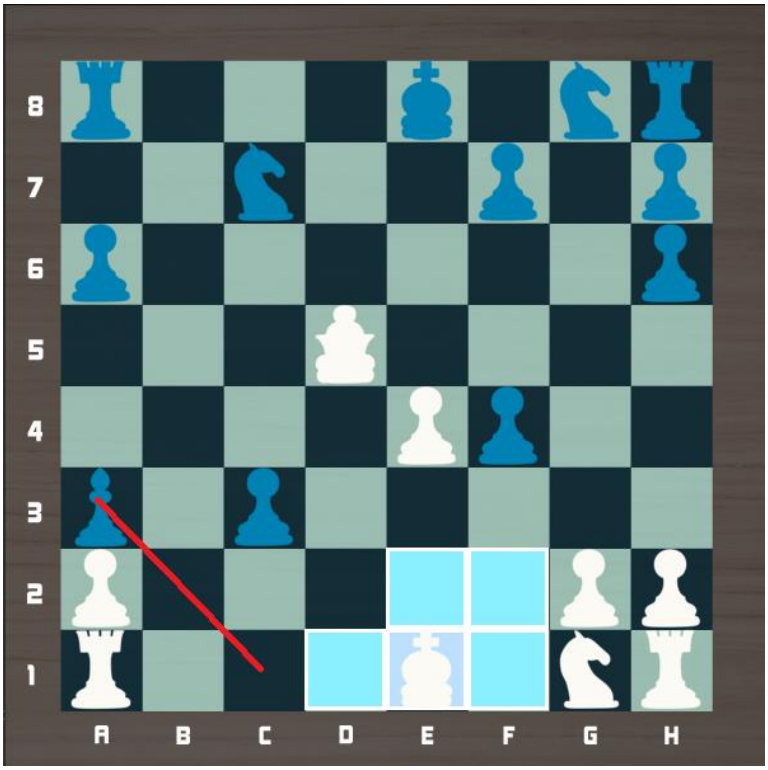
Hình 22 - Nước đi bình thường / ăn quân

- Trong hình trên, các ô tô đỏ là các ô mà Hậu đen ở h5 có thể bắt quân (capture). Các ô tô xanh là các nước đi bình thường (normal)

- Sau khi đã bật được moveType, ta tiến hành bước di chuyển **thử**.
- Nếu sau bước di chuyển thử đó, kiểm tra 'vua' của mình có bị chiếu hay không, sau đó undo lại.
- Kết quả sẽ trả về:

- True: Nếu ‘vua’ không bị chiếu (nước đi này hợp lệ).
- False: Nếu ‘vua’ bị chiếu hoặc không thỏa các điều kiện phía trên (nước đi này không hợp lệ).

❖ Trường hợp nước đi không hợp lệ: Vua trắng muốn nhập thành xa nhưng nếu di chuyển sang ô c1 thì sẽ bị Tượng đen ở ô a3 chiếu (các ô được tô xanh là những ô mà vua có thể đi mà không bị chiếu).



Hình 23 - Nước nhập thành không hợp lệ của Vua

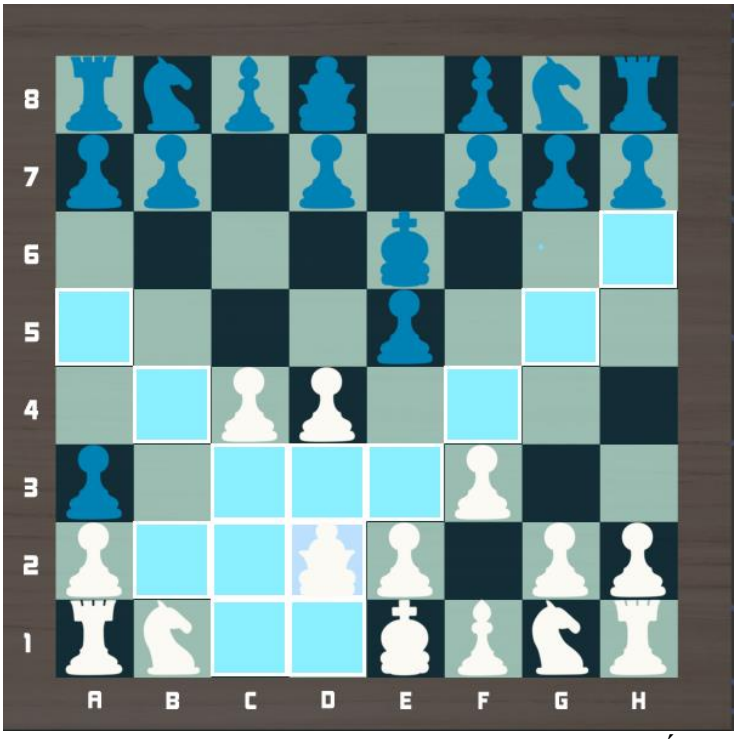
Phương thức makeMove:

- Phương thức di chuyển thật sự của bàn cờ
- Tham số đầu vào: tọa độ ô đầu, tọa độ ô đích và lượt whiteTurn.
- Kiểm tra nước đi hợp lệ bởi phương thức **canMove** nói trên và tiến hành di chuyển dựa vào MoveType tương ứng của nước đi đó.
  - Nếu là vua hoặc xe: sau khi di chuyển thì đánh dấu quân đã di chuyển.
  - Nếu là tốt: nếu tốt vừa đi 2 ô đầu thì đánh dấu lượt sau có thể bắt tốt qua đường.
  - Nếu có quân bị ăn thì thêm vào vector<Piece*> captured và đồng thời xóa quân bị ăn ra khỏi các quân trắng/đen tương ứng.
  - Nếu tốt phong cấp thì xóa quân tốt đó ra khỏi các quân trắng/đen tương ứng. Đồng thời, khởi tạo quân mà tốt phong cấp lên, thêm vào vector whitePieces/blackPieces tương ứng.

Phương thức getPossibleMoves:

- Liệt kê các bước đi hợp lệ của một quân cờ.

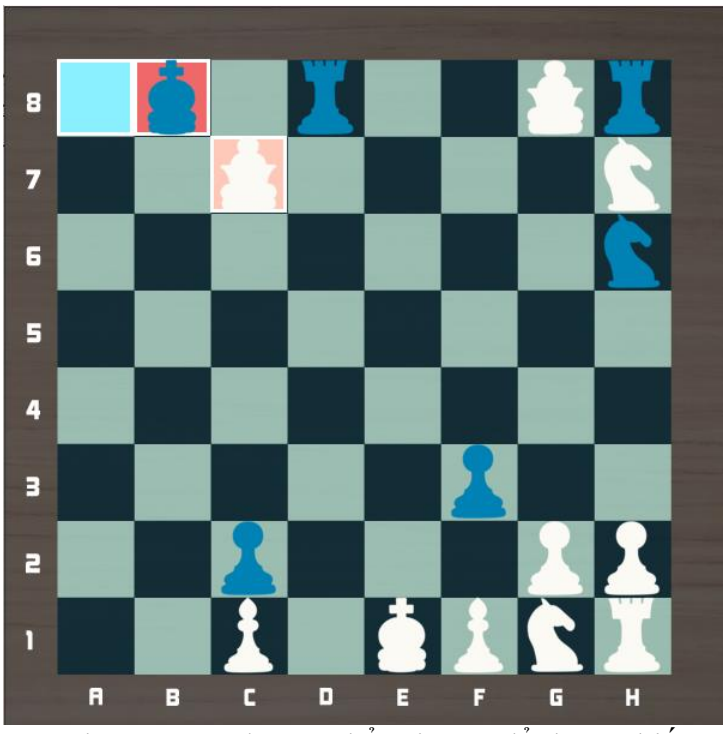
- Phương thức này sẽ giúp người chơi biết được các ô mà quân đó có thể đi đến được, đồng thời phương thức Game::drawPossibleMoves sẽ tô các ô đi được thành màu xanh biển để người chơi dễ dàng lựa chọn nước đi.



Hình 24 - Các nước đi hợp lệ của quân hậu Trắng

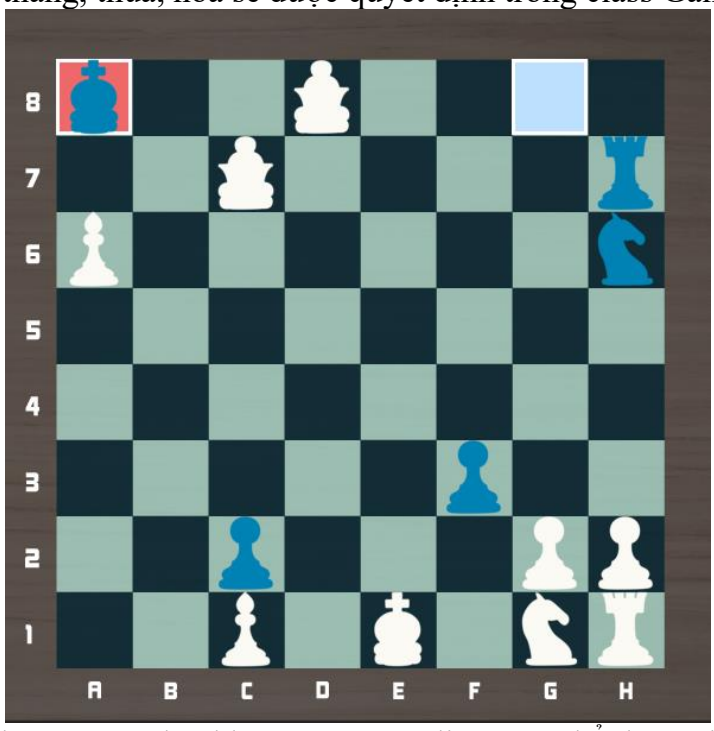
Phương thức isChecked:
- Đầu vào là lượt whiteTurn. Phương thức này xác định vua ở bên lượt của mình có đang bị chiếu hay không.
  - Xác định vị trí của vua.
  - Duyệt trong vector whitePieces/blackPieces (vector quân địch), dùng phương thức Piece::isValidMove xem thử có nước đi hợp lệ nào từ vị trí quân đó đến ô chứa vua của mình hay không. Nếu tồn tại nước đi, nghĩa là vua đang bị chiếu, return về true. Ngược lại, trả về false (vua bên mình đang không bị chiếu).

Phương thức canMoveNext:
- Đầu vào là lượt whiteTurn. Phương thức này xác định tại lượt của mình còn nước đi nào mà khiến vua bên mình không còn bị chiếu hay không.
  - Duyệt lần lượt các quân trong vector whitePieces/blackPiece (vector quân bên mình), dùng phương thức **canMove** để kiểm tra.
  - Nếu canMove() trả về true, nghĩa là lượt whiteTurn có thể đi tiếp và ván cờ vẫn tiếp tục.



Hình 25 - Vua đen có thể tới ô a8 để thoát chiếu

- Nếu trả về false, nghĩa là không có nước đi nào có thể giúp vua không bị chiếu. Lúc này, việc thắng, thua, hòa sẽ được quyết định trong class Game.



Hình 26 - Vua đen không có nước đi nào có thể thoát chiếu

Phương thức isCaptureMove:

- Xác định đó có phải là nước đi ăn quân hay không.
  - Nước đi ăn quân xảy ra khi MoveType thuộc một trong ba kiểu: Enpassant, Capture, Capture_Promote.

Phương thức undo:

- Tham số đầu vào: tọa độ ô mới, tọa độ ô cũ và MoveType .
- Phương thức này xác định các di chuyển dựa vào MoveType.
  - MoveType = LongCastling hoặc ShortCastling: Di chuyển vua và xe lại vị trí ban đầu và đánh dấu 2 quân này chưa di chuyển.
  - MoveType = Promote hoặc Capture_Promote: Di chuyển ngược lại. Thêm quân tốt phong cấp trong vector whitePieces/blackPiece tương ứng với lượt đi và xóa quân tốt đó trong vector l_promote , di chuyển tốt đến ô mới.
    - Nếu MoveType = Capture_Promote thì thêm vào quân đã bị ăn (chính là quân cuối cùng trong vector captured) vào vector whitePieces/ blackPiece tương ứng với lượt hiện tại.
    - Con trỏ của quân bị bắt vừa được thêm vào sẽ trỏ đến ô cũ trên bàn cờ.
  - MoveType = Enpassant: thêm tốt bị ăn (chính là quân cuối cùng trong vector captured) vào vector whitePieces/blackPiece tương ứng với lượt đi và di chuyển ngược lại.
    - ❖ Đánh dấu trạng thái bắt tốt = false (setPosibleEnpassant).
    - ❖ Nếu khoảng cách giữa ô cũ và ô mới là hai ô, đánh dấu quân tốt này có thể di chuyển 2 bước (setPosibleMoveTwoSquare).
  - MoveType = Capture: thêm quân bị ăn (chính là quân cuối cùng trong vector captured) vào vector whitePieces/blackPiece tương ứng với lượt đi và di chuyển ngược lại.
  - MoveType = Normal: di chuyển ngược lại (quân ở ô cũ sẽ di chuyển đến ô mới).

➢ **Phương thức này sẽ được gọi đến bởi phương thức Game::undo.**

2.2.3. Lớp Game

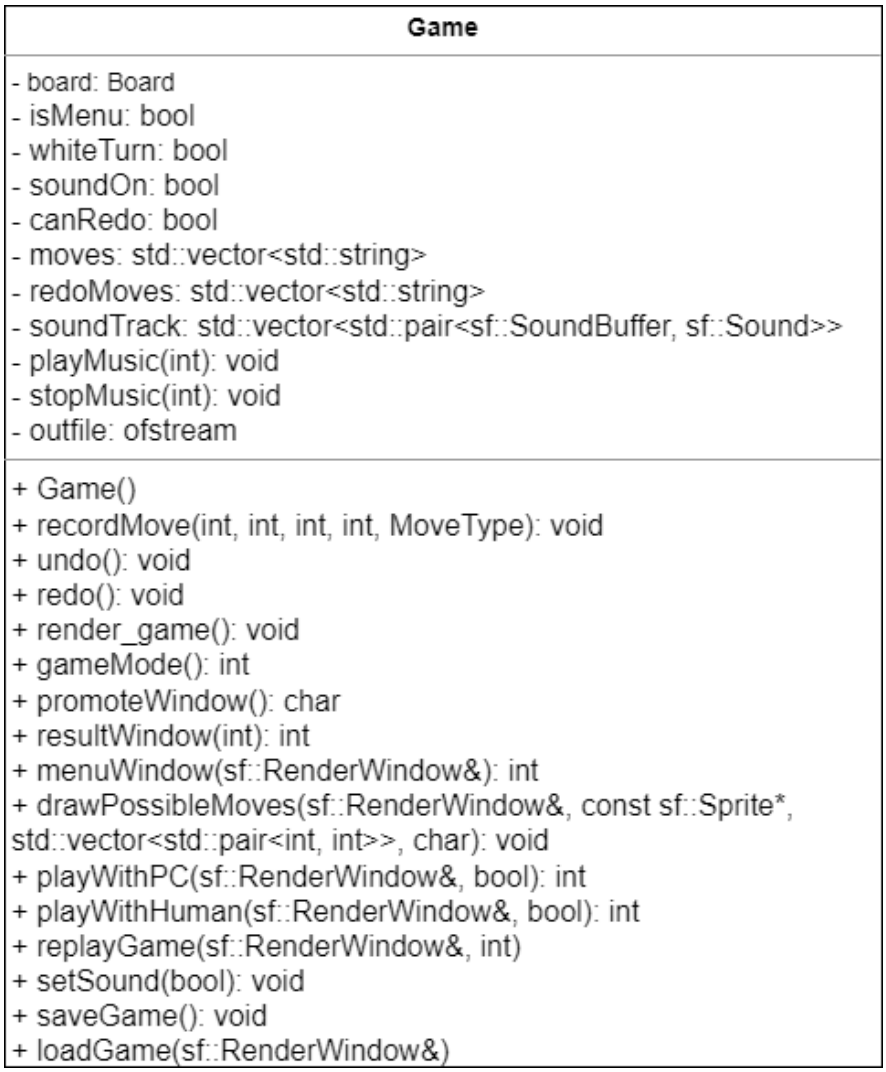
Về giao diện:

Nhóm sử dụng phần mềm Adobe Illustrator để thiết kế giao diện cho ứng dụng. Phần lớn các hình ảnh sửa dụng trong trò chơi đều do nhóm thiết kế. Qua đó, nhóm dễ dàng quản lý kích thước hình ảnh cũng như những dữ liệu cần thiết để thiết kế giao diện bằng SFML.

Để xây dựng giao diện bằng SFML:

- Nhóm xây dựng cửa sổ bằng sf::RenderWindow().
- Các background được thiết kế và canh chỉnh để phù hợp với tình huống.
- Các chi tiết được điều chỉnh size và được sử dụng phương thức setPosition() của lớp sf::Sprite để sắp xếp trên cửa sổ được tạo.

- Việc nhận biết các sự kiện đầu vào được sử lý bằng cách duyệt các sự kiện được lớp sf::Event lọc được. Qua đó sử dụng các phương thức phù hợp để giải quyết sự kiện được đưa vào.



Hình 27 - Sơ đồ UML của lớp Game

Âm thanh được sử dụng được sưu tầm từ các nguồn sau:
Nhạc nền game: "Crises Verse 1" – Honkai Star Rail battle track – Herta Station.
Nhạc nền menu: “Space Walk” - Out of Control - Honkai: Star Rail - Out of Control.
Âm thanh thực hiện bước đi và âm thanh khi thực hiện bước đi có bắt quân: Chess.com.

Phương thức khởi tạo Game():

- Khởi tạo các giá trị ban đầu cho ván cờ mới:
    - Gọi phương thức khởi tạo của class Board (Board::Board()) để khởi tạo một bàn cờ mới cũng như vị trí các quân cờ trên bàn cờ.
    - Gán giá trị false cho các đặc tính isMenu và canRedo:

        - isMenu được sử dụng làm cờ hiệu để chương trình chọn đúng loại nhạc nền.
        - canRedo được sử dụng làm cờ hiệu để chương trình nhận biết rằng việc thực hiện phương thức Game::redo() có khả thi không?
    - Gán giá trị true cho các đặt tính soundOn và whiteTurn:
        - soundOn được sử dụng để đặt cờ hiệu âm thanh của trò chơi có được mở hay không?
        - whiteTurn cho chương trình biết lượt hiện tại là của quân trắng hay quân đen (true: lượt trắng/ false: lượt đen).
    - Tải các file âm thanh vào chương trình bao gồm 4 loại âm thanh từ folder soundtrack:
        - Nhạc nền trong trận: tải soundBuffer vào soundTrack[0].first và setBuffer(soundTrack[0].first) cho soundTrack[0].second.
        - Nhạc nền menu: tải soundBuffer vào soundTrack[1].first và setBuffer(soundTrack[1].first) cho soundTrack[1].second.
        - Âm thanh bước đi bình thường: tải soundBuffer vào soundTrack[2].first và setBuffer(soundTrack[2].first) cho soundTrack[2].second.
        - Âm thanh bước đi khi có quân cờ bị bắt: tải soundBuffer vào soundTrack[3].first và setBuffer(soundTrack[3].first) cho soundTrack[3].second.
        - Ngoài ra, ta tiến hành chỉnh volume phù hợp cho từng âm thanh bằng phương thức setVolume(float) của sf::Sound.

Phương thức playMusic():

- Chơi âm thanh dựa vào số nguyên truyền vào:
    - Nếu cờ hiệu soundOn là false thì âm thanh sẽ không được chơi
    - Số 0 sẽ chơi soundTrack[0].second, số 1 sẽ chơi soundTrack[1].second, số 2 sẽ chơi soundTrack[2].second, số 3 sẽ chơi soundTrack[3].second.

Phương thức stopMusic():

- Dừng âm thanh đang chơi dựa vào số nguyên truyền vào:
    - Số 0 sẽ dừng chơi soundTrack[0].second, số 1 sẽ dừng chơi soundTrack[1].second, số 2 sẽ dừng chơi soundTrack[2].second, số 3 sẽ dừng chơi soundTrack[3].second.

Phương thức setSound():

- Cập nhập cờ hiệu soundOn. Dừng những âm thanh đang được trò chơi mở.

Phương thức recordMove():

- Ghi lại bước đi hợp lệ vào trong vector moves.
- Chuỗi ghi chú bước đi có định dạng như sau:

Bảng 4 - Định dạng chuỗi ghi chú bước đi

0	2	4	6	8	10	12
Toạ độ x cũ	Toạ độ y cũ	Toạ độ x mới	Toạ độ y mới	Loại bước đi	Là bước đi đầu tiên của quân cờ hay không?	Ký hiệu quân thăng cấp

1	3	5	7	9	11
Dấu cách	Dấu cách	Dấu cách	Dấu cách	Dấu cách	Dấu cách

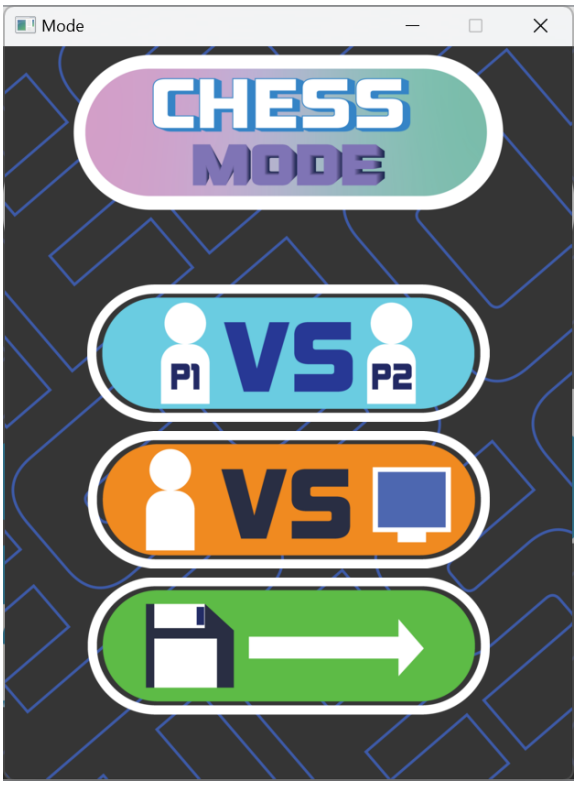
Phương thức undo():

- Đọc bước đi gần nhất từ vector moves, xử lý chuỗi để truyền dữ liệu vào phương thức Board::undo(), đẩy bước đi vừa đọc vào redoMoves, cập nhập cờ hiệu canRedo và xoá bước đi đó khỏi vector moves.

Phương thức redo():

- Đọc bước đi undo gần nhất trong vector redoMoves, xử lý chuỗi để truyền dữ liệu vào phương thức Board::makeMove(), đẩy bước đi vừa đọc vào moves, xoá bước đi đó khỏi vector moves.

Phương thức gameMode():



Hình 28 - Giao diện Game Mode

- Tạo cửa sổ chọn chế độ chơi sử dụng sf::RenderWindow.
- Tạo nền và các nút cần thiết sử dụng sf::Texture và sf::Sprite.
    - Có 3 nút:
        - Chơi với máy: Khởi tạo ván cờ chơi với máy mới.
        - Chơi với người: Khởi tạo ván cờ chơi với người mới.
        - Mở lại game vừa chơi: Tiếp tục ván cờ trước đó.
- Ta kiểm tra lựa chọn của người dùng bằng phương thức getGlobalBound() của sf::Sprite và trả về giá trị nguyên tương ứng với chế độ chơi người dùng chọn.
- Nếu đóng cửa sổ, người dùng sẽ trở về cửa sổ menu của trò chơi.
- Giá trị trả về của phương thức:

Bảng 5 - Giá trị trả về của phương thức gameMode()

GIÁ TRỊ TRẢ VỀ	Ý NGHĨA
-1	Người dùng không chọn lựa chọn nào.
0	Chơi với người.
1	Chơi với máy.
2	Mở lại game vừa chơi.

Phương thức promoteWindow():

Hình 29 - Giao diện cửa sổ Promote

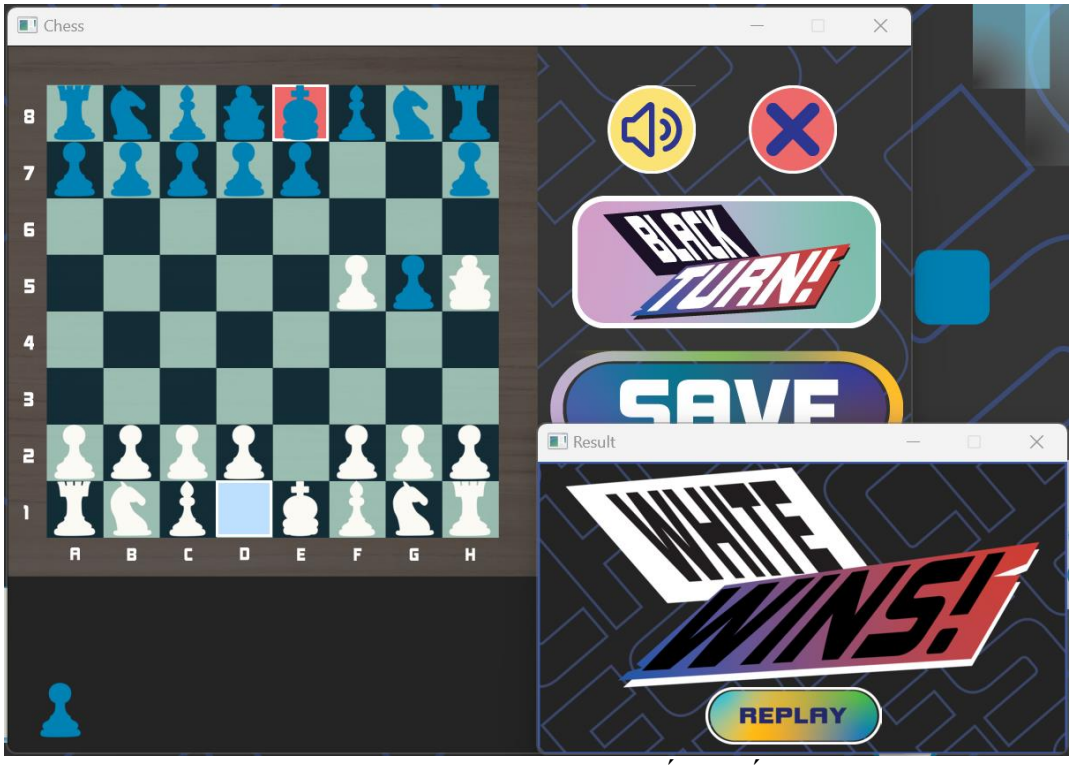
- Tạo cửa sổ thăng cấp cho quân tốt sử dụng sf::RenderWindow.
- Tạo nền và các nút cần thiết sử dụng sf::Texture và sf::Sprite.
    - Có 4 nút:
        - Thăng cấp thành quân mã: khởi tạo một quân mã mới thế chỗ cho quân tốt thăng cấp.
        - Thăng cấp thành quân xe: khởi tạo một quân xe mới thế chỗ cho quân tốt thăng cấp.
        - Thăng cấp thành quân hậu: khởi tạo một quân hậu mới thế chỗ cho quân tốt thăng cấp.
        - Thăng cấp thành quân tượng: khởi tạo một quân tượng mới thế chỗ cho quân tốt thăng cấp.
- Ta kiểm tra lựa chọn của người dùng bằng phương thức getGlobalBound() của sf::Sprite và trả về giá trị ký tự tương ứng với chế độ chơi người dùng chọn.
- Người dùng sẽ không thể tiếp tục trò chơi nếu không chọn quân thăng cấp, vì thế khi đóng cửa sổ, một cửa sổ promoteWindow sẽ mở lên thay thế cho cửa sổ cũ.
- Sau khi đưa ra lựa chọn, người dùng sẽ quay lại cửa sổ trò chơi với quân thăng cấp.
- Giá trị trả về của phương thức:

Bảng 6 - Giá trị trả về của phương thức promoteWindow()

GIÁ TRỊ TRẢ VỀ	Ý NGHĨA
'-1'	Người dùng không chọn lựa chọn nào.
'N'	Quân mã.
'Q'	Quân hậu.
'R'	Quân xe.
'B'	Quân tượng.

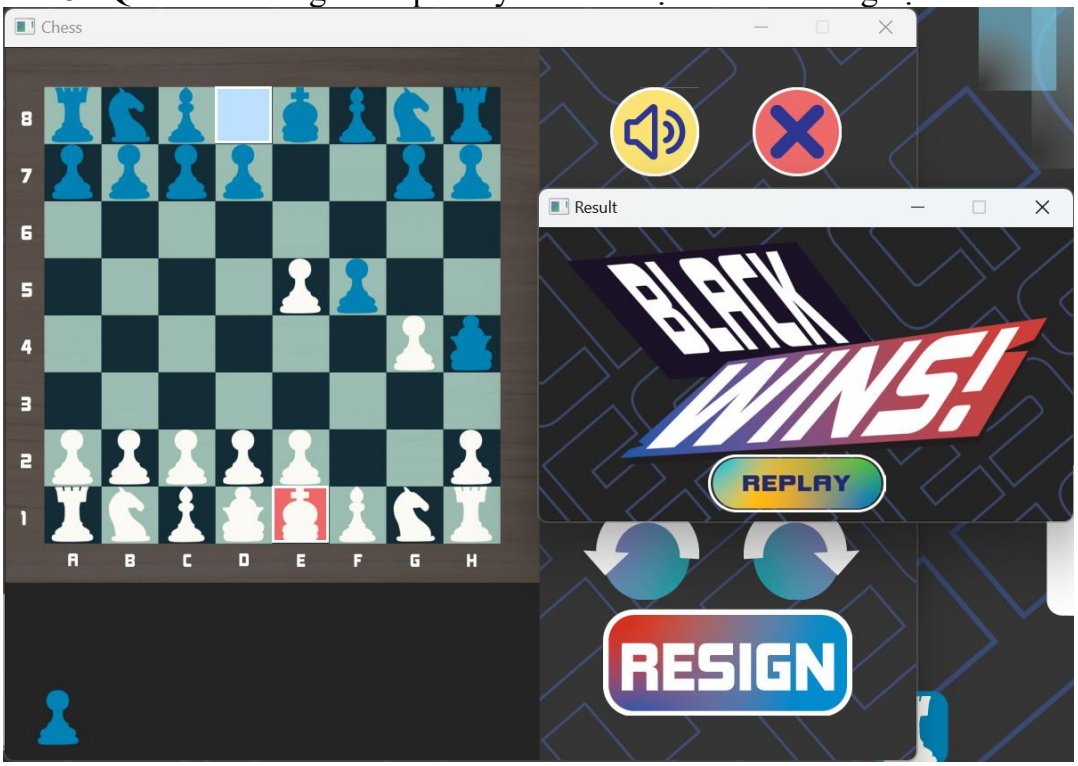
Phương thức resultWindow():

- Mở cửa sổ thông báo kết quả của ván đấu:
    - Quân trắng thắng: Kết quả này sẽ xuất hiện khi vua đen bị chiếu bí.



Hình 30 - Giao diện quân Trắng thắng

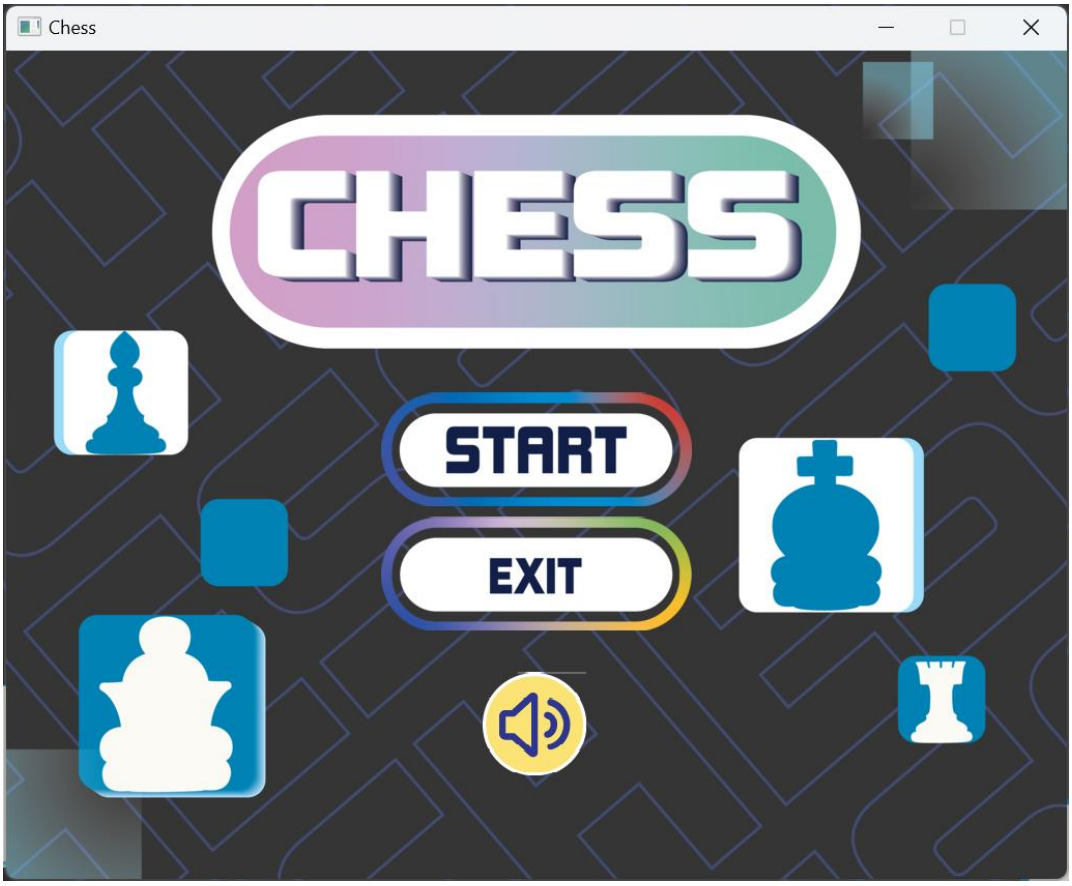
- Quân đen thắng: Kết quả này sẽ xuất hiện khi vua trắng bị chiếu bí.



Hình 31 - Giao diện quân Đen thắng

    - Hoà: Kết quả này sẽ xuất hiện khi quân đen hoặc quân trắng không còn bước đi nào hợp lệ nhưng cả vua đen và vua trắng đều không bị chiếu.
- Người dùng sẽ có lựa chọn xem lại ván đấu nếu muốn.
- Khi tắt cửa sổ, người dùng lập tức quay lại cửa sổ menu.

Phương thức menuWindow():



Hình 32 - Giao diện chính

- Mở cửa sổ menu bằng sf::RenderWindow.
- Tạo nền và các nút cần thiết sử dụng sf::Texture và sf::Sprite.
    - Có 3 nút:
        - Nút mở window game mode.
        - Nút thoát.
        - Nút bật/tắt âm thanh.
- Khi chọn nút mở window game mode, phương thức gameMode() được gọi.
- Khi chọn nút thoát, chương trình kết thúc và người dùng thoát khỏi giao diện ứng dụng cờ vua.
- Khi chọn nút bật/tắt âm thanh, nếu âm thanh bật thì âm thanh sẽ được tắt và texture sẽ thay đổi, nếu âm thanh tắt thì âm thanh sẽ được bật.
- Đây là giao diện chính khi mở trò chơi và quay lại khi hoàn thành ván đấu.

Phương thức drawPossibleMoves():

- Phương thức sử dụng phương thức Board::getPossibleMoves() để lấy những bước đi hợp lệ của quân cờ tại vị trí được truyền vào, qua đó vẽ các ô có thể đi được của quân cờ đó.
- Màu sắc của ô sẽ thay đổi dựa trên loại bước đi để giúp người chơi dễ phân biệt.

Phương thức playWithHuman():

- Phương thức này được gọi khi chế độ chơi với người được chọn.
- Trước tiên, các nút cần thiết sẽ được khởi tạo, cũng như bàn cờ sẽ được vẽ lên cửa sổ nhờ phương thức Board::draw().
- Cách thực hiện bước đi:
    - Để nhận là quân cờ được chọn ta kiểm tra sự kiện chuột trái có được nhấn chưa. Khi chuột trái được nhấn, ta trả về toạ độ đó và kiểm tra xem tại đó có sprite quân cờ không?
    - Sử dụng toạ độ của các sprite quân cờ. Khi phương thức getGlobalBound().contains() trả về true, ta nhận được tín hiệu quân cờ được chọn.
    - Tiến hành kiểm tra quân cờ đó có thể di chuyển được không:
        - Có phải lượt của quân cờ đó.
        - Quân cờ đó có bị quân nào khác chặn.
        - Quân vua hiện tại có đang bị chiếu không?
    - Tiếp tục kiểm tra ô đích được chọn có phải là bước đi hợp lệ bằng phương thức Board::canMove().
    - Nếu bước đi hợp lệ ta gọi hàm Board::makeMove() để thực hiện bước đi và cập nhập cờ hiệu whiteTurn để truyền lượt cho quân màu còn lại.
    - Nếu bước đi đó là loại bước đi thăng cấp thì ta tiến hành gọi phương thức promoteWindow().
    - Lưu bước đi đó lại vào trong vector moves.
    - Nếu bước đi không hợp lệ thì sẽ không thực hiện được bước đi của người chơi.
- Khi quân vua của một màu bị chiếu, màu ô của quân đó sẽ thanh đổi để báo hiệu dễ dàng hơn cho người chơi. Việc kiểm tra chiếu tướng được kiểm tra bởi phương thức Board::isChecked().
- Trong quá trình chơi, ở đầu mỗi lượt ta sẽ kiểm tra xem quân cờ màu đó có còn bước đi hợp lệ không bằng phương thức Board::canMoveNext()
    - Nếu có thì trò chơi tiếp tục.
    - Nếu không thì ta sẽ kiểm tra quân vua của màu đó có bị chiếu không:
        - Nếu vua đang bị chiếu: quân màu còn lại thắng, gọi phương thức resultWindow() cho quân chiến thắng.
        - Nếu vua đang không bị chiếu: kết quả là hoà, gọi phương thức resultWindow() cho kết quả hoà.
- Giao diện bàn cờ cũng có chứa các nút phục vụ cho quá trình chơi:

- Nút bật/tắt âm thanh.
- Nút resign: Sử dụng để nhận thua. Nhận thua khi whiteTurn đang là true thì phần thắng thuộc về quân đen và ngược lại.
- Nút undo/redo: Sử dụng để thu hồi lại bước đi vừa thực hiện. Khi chọn, phương thức undo()/redo() tương ứng sẽ được gọi.
- Nút Save: lưu lại trò chơi để có thể tiếp tục ván đấu vào lần sau.

Phương thức playWithPC():

- Phương thức này được gọi khi chế độ chơi với máy được chọn.
- Lượt quân đen sẽ được xử lý bằng các phương thức để khởi tạo ngẫu nhiên. Từ danh sách các quân đen, ta sẽ chọn ngẫu nhiên 1 quân, lấy danh sách bước đi hợp lệ của quân đó. Chọn ngẫu nhiên, một bước đi hợp lệ và thực hiện bước đi đó. Quân khi thăng chức cũng sẽ được chọn ngẫu nhiên.
- Trước tiên, các nút cần thiết sẽ được khởi tạo, cũng như bàn cờ sẽ được vẽ lên cửa sổ nhờ phương thức Board::draw().
- Cách thực hiện bước đi:
  - Để nhận là quân cờ được chọn ta kiểm tra sự kiện chuột trái có được nhấn chưa. Khi chuột trái được nhấn, ta trả về toạ độ đó và kiểm tra xem tại đó có sprite quân cờ không?
  - Sử dụng toạ độ của các sprite quân cờ. Khi phương thức getGlobalBound().contains() trả về true, ta nhận được tín hiệu quân cờ được chọn.
  - Tiến hành kiểm tra quân cờ đó có thể di chuyển được không:
    - Có phải lượt của quân cờ đó.
    - Quân cờ đó có bị quân nào khác chặn.
    - Quân vua hiện tại có đang bị chiếu không?
  - Tiếp tục kiểm tra ô đích được chọn có phải là bước đi hợp lệ bằng phương thức Board::canMove().
  - Nếu bước đi hợp lệ ta gọi hàm Board::makeMove() để thực hiện bước đi và cập nhập cờ hiệu whiteTurn để truyền lượt cho quân màu còn lại.
  - Nếu bước đi đó là loại bước đi thăng cấp thì ta tiến hành gọi phương thức promoteWindow().
  - Lưu bước đi đó lại vào trong vector moves.
  - Nếu bước đi không hợp lệ thì sẽ không thực hiện được bước đi của người chơi.
- Khi quân vua của một màu bị chiếu, màu ô của quân đó sẽ thanh đổi để báo hiệu dễ dàng hơn cho người chơi. Việc kiểm tra chiếu tướng được kiểm tra bởi phương thức Board::isChecked().
- Trong quá trình chơi, ở đầu mỗi lượt ta sẽ kiểm tra xem quân cờ màu đó có còn bước đi hợp lệ không bằng phương thức Board::canMoveNext()
  - Nếu có thì trò chơi tiếp tục.
  - Nếu không thì ta sẽ kiểm tra quân vua của màu đó có bị chiếu không:

        - Nếu vua đang bị chiếu: quân màu còn lại thắng, gọi phương thức resultWindow() cho quân chiến thắng.
        - Nếu vua đang không bị chiếu: kết quả là hoà, gọi phương thức resultWindow() cho kết quả hoà.
- Giao diện bàn cờ cũng có chứa các nút phục vụ cho quá trình chơi:
    - Nút bật/tắt âm thanh.
    - Nút resign: Sử dụng để nhận thua. Nhận thua khi whiteTurn đang là true thì phần thắng thuộc về quân đen và ngược lại.
    - Nút undo/redo: Sử dụng để thu hồi lại bước đi vừa thực hiện. Khi chọn, phương thức undo()/redo() tương ứng sẽ được gọi.
    - Nút Save: lưu lại trò chơi để có thể tiếp tục ván đấu vào lần sau.

Phương thức render_game():

- Khởi tạo trò chơi: Gọi các phương thức xây dựng ở trên để tạo ván đấu. Đây là phương thức tạo giao diện chính, khi giao diện bị đóng thì ứng dụng sẽ tắt.

Phương thức saveGame():

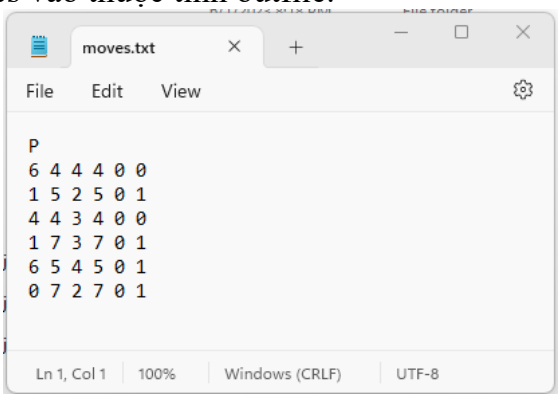
- Đầu file output thì sẽ có thể hiện chơi với máy hay chơi 2 người bằng chữ cái ‘H’ hoặc ‘P’.
- Định dạng mỗi dòng như sau:

Bảng 7 - Định dạng mỗi dòng trong file moves.txt

0	2	4	6	8	10	12
Toạ độ x cũ	Toạ độ y cũ	Toạ độ x mới	Toạ độ y mới	Loại bước đi	Là bước đi đầu tiên của quân cờ hay không?	Ký hiệu quân thăng cấp

1	3	5	7	9	11
Dấu cách	Dấu cách	Dấu cách	Dấu cách	Dấu cách	Dấu cách

- Phương thức dùng để lưu lại ván game đang chơi bằng cách in những phần tử trong vector moves vào thuộc tính outfile.



Hình 33 - File output moves.txt

Phương thức loadGame():

- Phương thức dùng để replay lại ván đấu vừa xong hoặc load lại ván đấu chơi dở vừa lưu. Các bước đi sẽ được đọc lại từ file moves.txt. Sau đó, sẽ set lại bàn cờ như ban đầu để thể hiện từng bước đi cụ thể nếu replay hoặc sẽ di chuyển các quân cờ như đã lưu trong file nếu load lại ván đấu chơi dở.
- Nếu load lại ván đấu chơi dở, sẽ không thể hiện từng bước đi mà sẽ di chuyển nhanh chóng và sẽ xem ván đấu vừa rồi là chơi với máy hay chơi với người bằng cách xem ký tự đầu của file moves.txt là gì để chọn giao diện và tùy chỉnh cho phù hợp.

3. Thư viện SFML

- Thư viện SFML (Simple and Fast Multimedia Library) là thư viện đa phương tiện, chạy trên đa nền tảng như Windows, Linux, MacOS và dùng được bằng nhiều ngôn ngữ như C++, Ruby, Java, Python, …

- Trong đồ án này, thư viện SFML hỗ trợ trong việc tạo ra giao diện người dùng, hành vi di chuyển, âm thanh cho chương trình.

- Phiên bản thư viện SFML được dùng trong đồ án: SFML 2.5.1 Visual C++ 15 (2017) - 64-bit

- Trang web chính thức của thư viện SFML: SFML (sfml-dev.org)

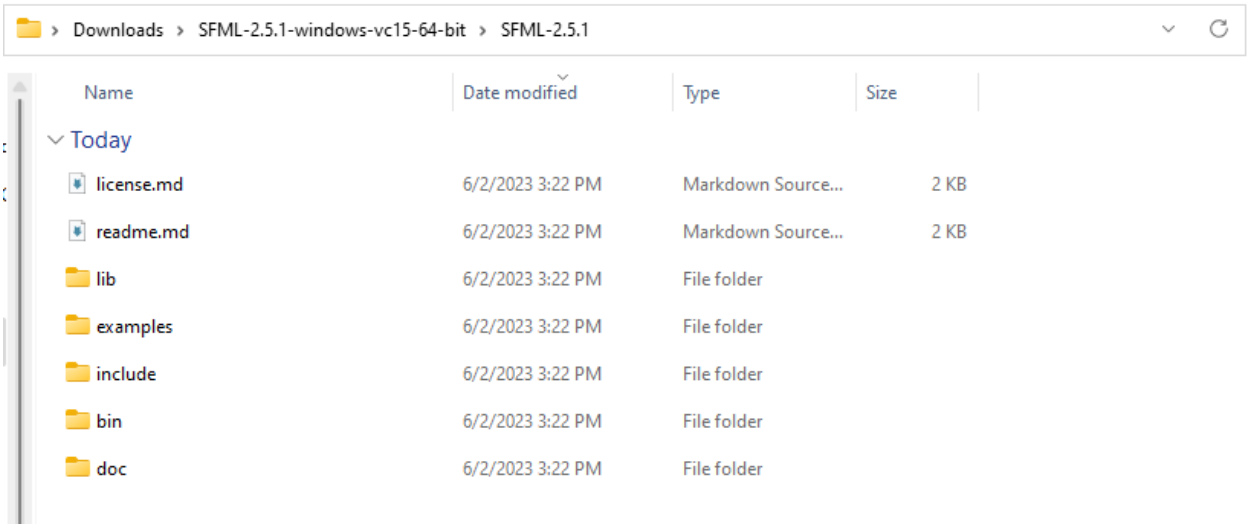
Hình 34 - Logo thư viện SFML

4. Cách build trò chơi

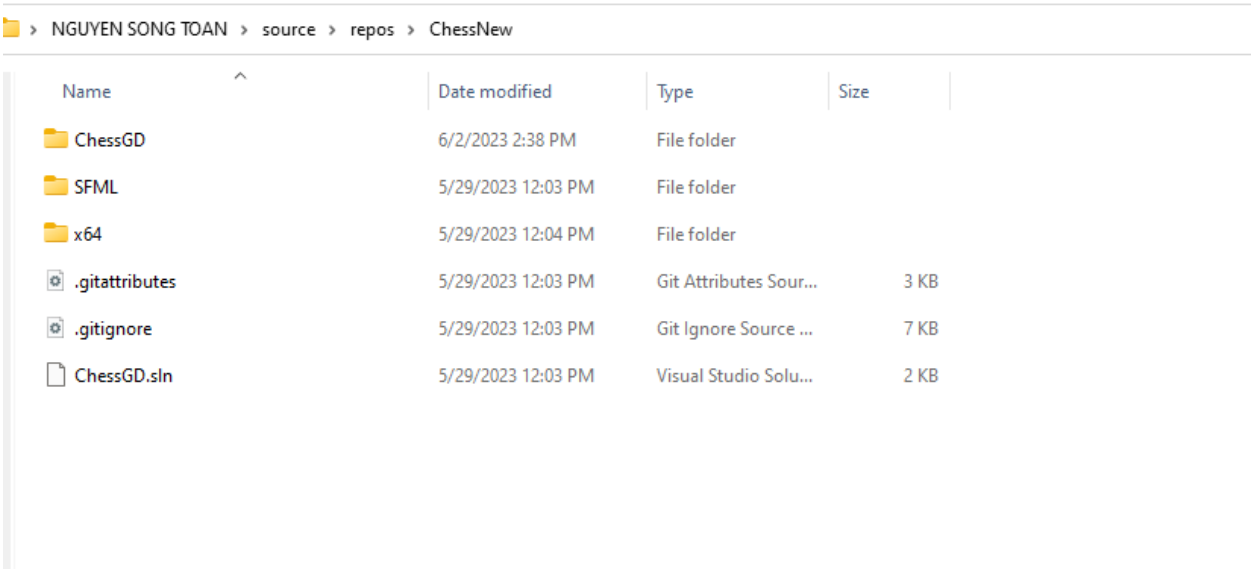
- Chương trình được build trên Visual Studio và ở chế độ ***Debug***.

- Bước 1: Tải file .zip của thư viện phiên bản SFML 2.5.1 Visual C++ 15 (2017) - 64-bit trên trang này về SFML 2.5.1 (SFML / Download) (sfml-dev.org), sau đó giải nén file đó ra.

- Bước 2: Copy thư mục SFML-2.5.1 sang thư mục của project. Bên thư mục của project, tạo thư mục tên SFML và paste những file, sub-folder vừa copy vào thư mục này.



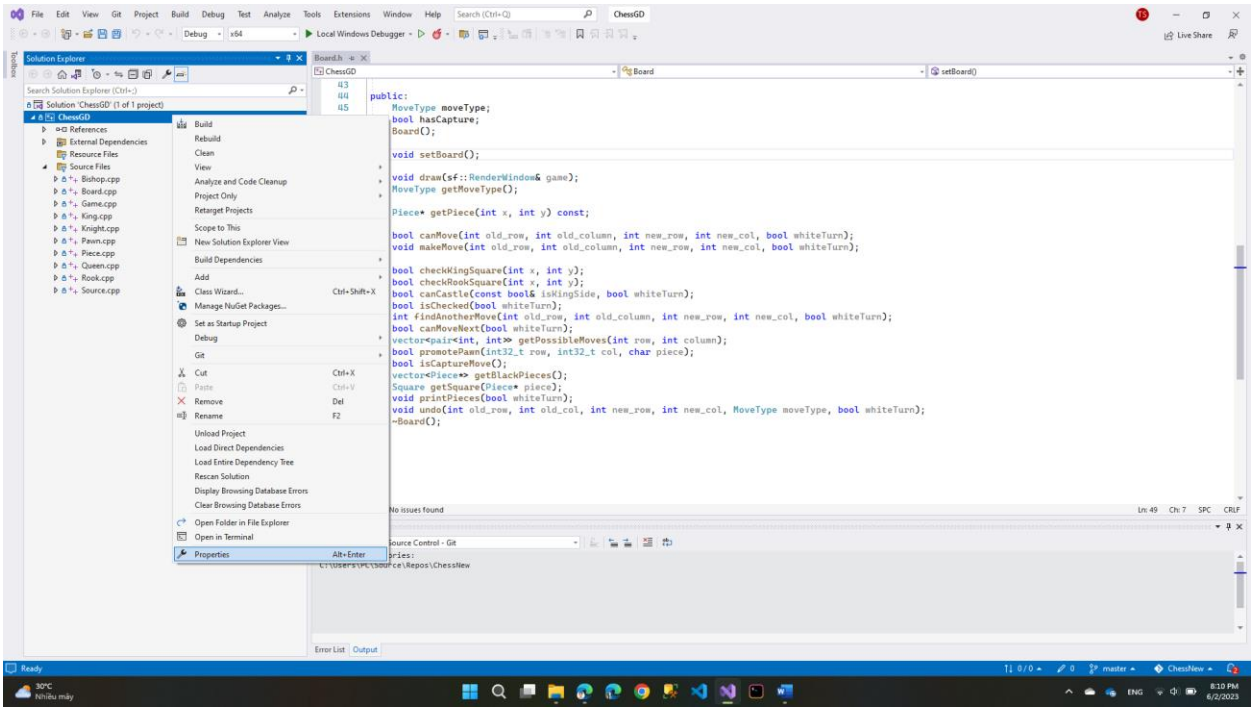
Hình 35 - Thư mục bên Downloads



Hình 36 - Tạo thư mục SFML bên thư mục của project

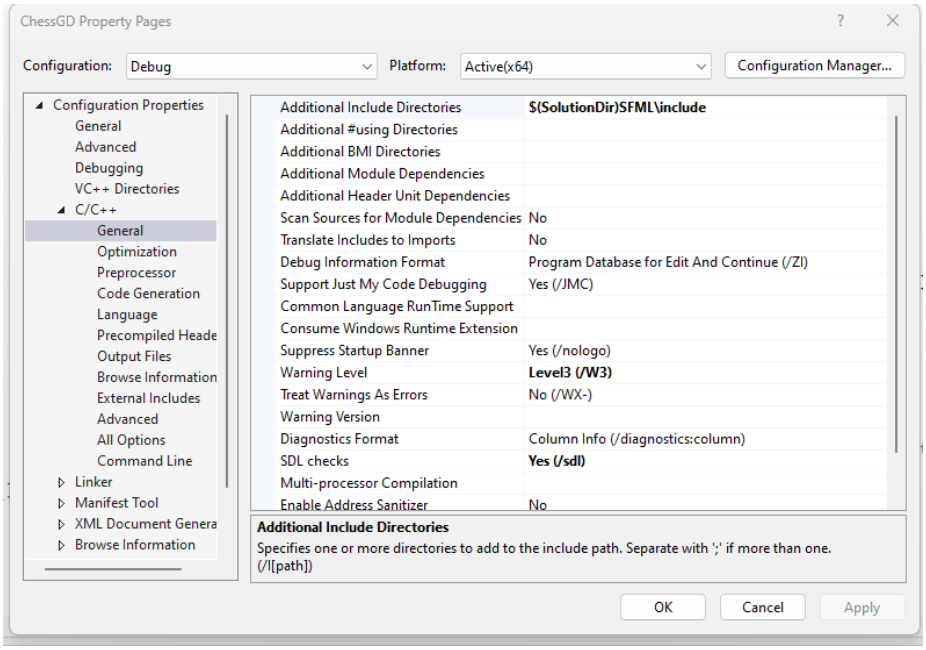
- Bước 3: Build chương trình ở chế độ Debug (x64), sau đó add thư viện vào project như sau:

Click chuột phải vào project và chọn Properties

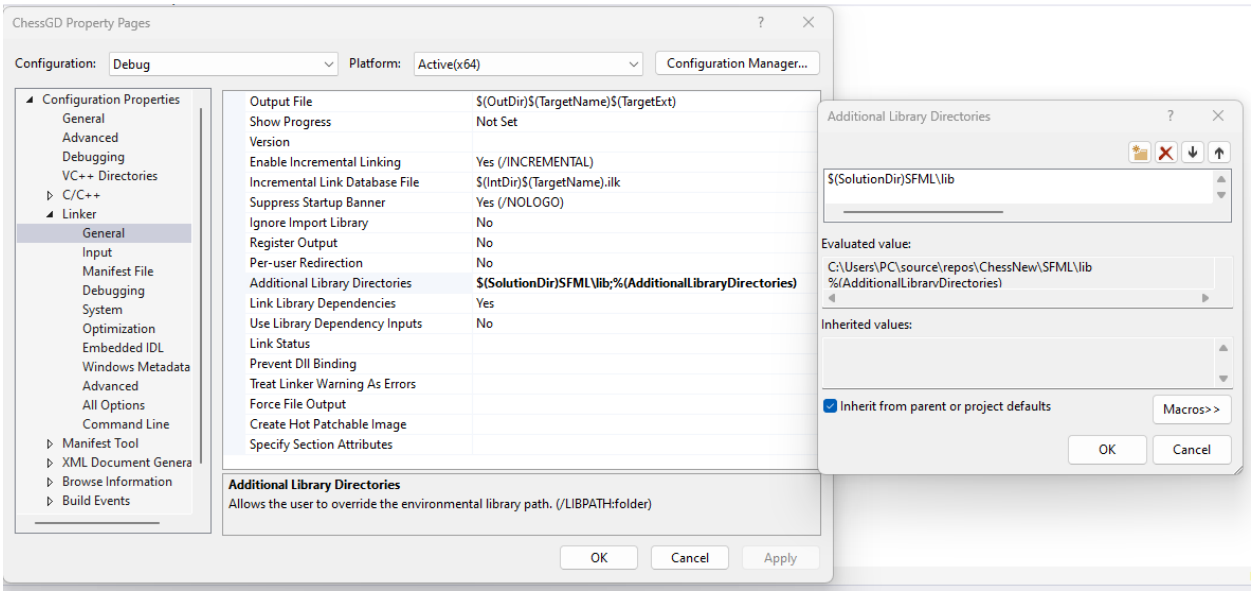


Hình 37 - Properties

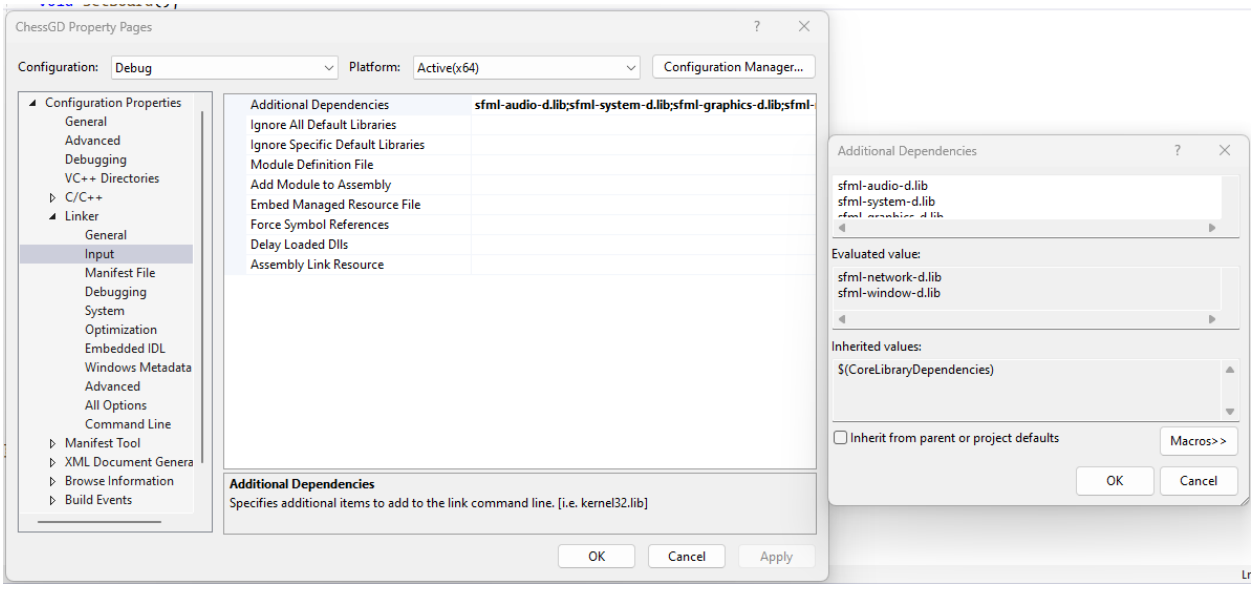
Sau đó, thêm include, lib và các modules (sfml-audio-d.lib, sfml-graphics-d.lib, sfml-network-d.lib, sfml-window-d.lib, sfml-main-d.lib) như sau:



Hình 38 - Thêm include

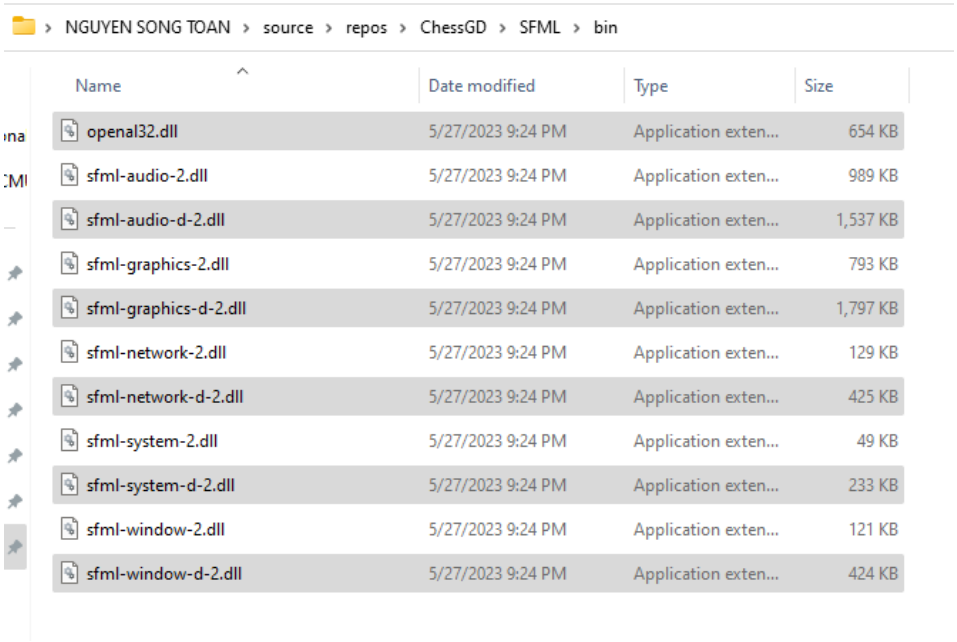


Hình 39 - Thêm library

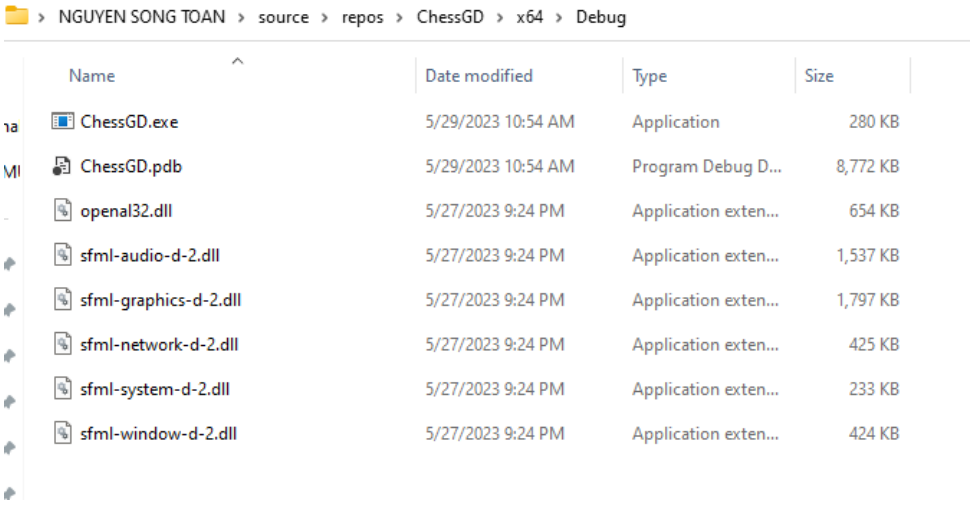


Hình 40 - Thêm modules

Bước 4: Cuối cùng, ta thêm các file này ở thư mục bin của thư mục SFML vào trong thư mục chứa file thực thi exe.



Hình 41 - Thư mục bin



Hình 42 - Thư mục Debug

5. Demo của trò chơi

Giao diện, các chức năng đều được chúng em thể hiện trong demo này. Liên kết dẫn đến demo: https://youtu.be/GsOZxRZBp30

Tài liệu tham khảo

[1] Online. *Chess Game in C++*. URL: Chess Game in C++ | C++ Algorithms | cppsecrets.com

[2] Online. *SFML 2.1 Tutorial Series*. URL: SFML 2.1 Tutorial Series - YouTube

[3] Online. *How to Play Chess*. URL: How to Play Chess: Learn the Rules & 7 Steps To Get You Started - Chess.com