BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG HCM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN HỌC
PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
ĐỀ TÀI: GAME CỜ VUA

Giảng viên hướng dẫn: *Thầy Lý Duy Nam*
Lớp: *21CTT2*
Thành viên: *21120041 – Dương Ngọc Thái Bảo*
21120050 – Trương Tấn Đạt
21120058 – Phạm Nhật Duy
21120070 – Nhan Hữu Hiếu

HỌC KỲ II
NĂM HỌC 2022 – 2023

MỤC LỤC

A.THÔNG TIN CHUNG VỀ ĐỒ ÁN

I. Nội dung cần thực hiện:

Sinh viên được yêu cầu lập nhóm từ 4-5 người/nhóm để xây dựng game Cờ Vua với 2 người chơi thông qua các thao tác và hiển thị đơn giản trên màn hình console. Các nhóm được yêu cầu bắt buộc phải tuân thủ phương pháp lập trình hướng đối tượng và sử dụng ngôn ngữ C++ cho đồ án này.

Hình 1: Hình minh họa trò chơi Cờ Vua

a. Yêu cầu cơ bản (10 điểm):
   - Sơ đồ lớp UML (3đ)
   - Chế độ 2 người chơi: (6đ):
     - Cho phép họ chọn quân cờ, nhập vị trí nước đi tiếp theo thông qua bàn phím.
     - Hiển thị bàn cờ và quân cờ trên màn hình console.
     - Cài đặt được logic của trò chơi (điều kiện thắng, nước đi các quân cờ…)
   - Lưu và chơi lại ván cờ trước đó (1đ)

b. Yêu cầu nâng cao (điểm cộng):
   - Thao tác thông qua click chuột trên console (0.5đ)
   - Chế độ chơi với máy (random nước đi 0.5đ, chế độ máy khó 1đ).
   - Undo/Redo (0.5đ)

- Replay: xem lại ván đấu vừa đấu (0.5đ)
- Giao diện người dùng (1đ)
- Âm thanh trò chơi (nhạc nền, di chuyển quân cờ, ăn quân cờ đối phương, kết thúc ván đấu, …) (0.5đ)

II. Thư viện SFML:

Trong đồ án này, thư viện SFML được sử dụng để thiết kế giao diện người dùng và có âm thanh.

1. Giới thiệu về thư viện SFML:

SFML ***(Simple and Fast Multimedia Library)*** là một thư viện đa phương tiện được đóng góp từ nhiều người ở cộng đồng, được viết chủ yếu bằng ngôn ngữ C++.

Hình 2: Logo của thư viện SFML

Thư viện SFML có vài điểm tương đồng với thư viện SDL2 ***(Simpke DirectMedia Layer 2)***, nhưng được viết chủ yếu theo phương pháp hướng đối tượng nên việc tiếp cận cho các phần mềm hướng đối tượng sẽ dễ dàng hơn nhiều so với SDL2.

Sử dụng thư viện SFML giúp ta viết được các chương trình có thể chạy trên nhiều nền tảng.

2. Các modules của thư viện SFML:

Hiện tại thư viện SFML cung cấp cho người dùng 5 modules:

- **Audio:** cung cấp các lớp giúp xử lý về âm thanh như: phát một tập tin nhạc hoặc tập tin ghi âm…
- **Graphics:** cung cấp các lớp giúp xử lý đồ họa như vẽ hình…
- **Network:** cung cấp các lớp giúp xử lý các giao thức mạng như HTTP, FTP, …
- **System:** cung cấp các lớp giúp xử lý các vấn đề hệ thống như thời gian, Unicode, …

- **Window:** cung cấp các lớp giúp xử lý cửa sổ sự kiện.

III. Các trang web về thư viện SFML:

Trang web chính thức của thư viện SFML, nơi người dùng có thể tìm kiếm tài liệu về các lớp của thư viện cũng như đặt các câu hỏi trong quá trình sử dụng: SFML (sfml-dev.org)

GitHub của thư viện: SFML/SFML: Simple and Fast Multimedia Library (github.com).

B. MÔ TẢ THIẾT KẾ PHẦN MỀM

I. Sơ đồ lớp UML của phần mềm:

Do kích thước chiều ngang của giấy có giới hạn nên sơ đồ lớp UML của phần mềm được đặt trong thư mục **UMLDiagram**, đính kèm với bài báo cáo này.

II. Mô tả các lớp trong phần mềm:

1. Namespace `utilities`:

Namespace `utilities` cung cấp các hàm bổ trợ cho quá trình xử lý, ví dụ như hàm cung cấp địa chỉ (location) của hình ảnh, âm thanh hay hàm kiểm tra vị trí hợp lệ, các hàm chuyển đổi tọa độ, …

2. Lớp `Settings`:

Lớp `Settings` cung cấp một vài thông tin cài đặt cơ bản của game.
Sơ đồ lớp UML của lớp Settings như sau:

Hình 3: Sơ đồ lớp UML của lớp `Settings`

Các thuộc tính (attributes) của lớp `Settings` là thông tin về màu của các ô vuông trên bàn cờ, màu của ô vuông lúc được làm nổi bật (highlighted), màu của ô vuông khi bị chiếu. Các màu này được dựa trên mã màu RGBA. Mã màu trong đồ án được lấy từ trang web: RGBA Color Picker. Ngoài ra, còn có các thuộc tính mô tả về tỉ lệ, độ dày mỏng của các nét highlight, cũng như kích thước và offset của ô vuông trên bàn cờ.

Các phương thức (methods) của lớp `Settings` cung cấp các hàm getter và setter cho các thuộc tính của lớp.

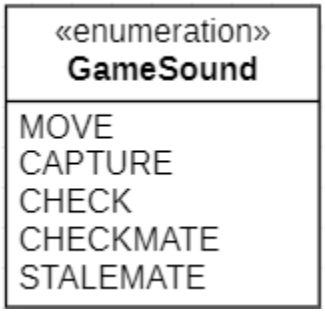
Hầu hết các thuộc tính và phương thức của lớp `Settings` (trừ hàm tạo – constructor và hàm hủy – destructor) đều được khai báo dưới dạng thuộc tính/phương thức tĩnh nhằm tạo ra sự tiện lợi khi ta cần gọi chúng, giúp ta không phải tạo một đối tượng mới mỗi khi muốn sử dụng đến các tính năng này.

3. Các lớp enum:

a. Lớp enum `GameSound`:

Lớp `GameSound` cung cấp các loại âm thanh của game: âm thanh khi di chuyển quân cờ (MOVE), khi bắt quân (CAPTURE), khi chiếu (CHECK), khi chiếu hết (CHECKMATE) hay khi stalemate (STALEMATE – trạng thái mà cả hai bên đều không còn nước nào có thể đi được).

Sơ đồ UML của lớp enum `GameSound` như sau:

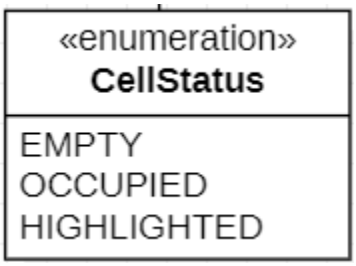


Hình 4: Sơ đồ lớp UML của lớp enum `GameSound`

b. Lớp enum `CellStatus`:

Lớp `CellStatus` cung cấp các trạng thái của một ô vuông trên bàn cờ: đã có quân (OCCUPIED), chưa có quân (EMPTY) hay đang được highlight (HIGHTLIGHTED).

Sơ đồ UML của lớp enum `CellStatus` như sau:

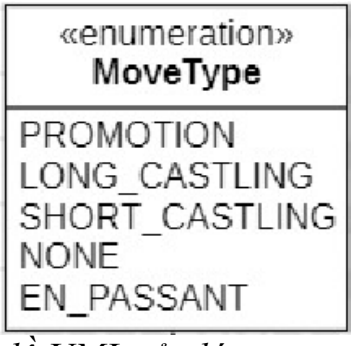


Hình 5: Sơ đồ lớp UML của lớp enum `CellStatus`

c. Lớp enum `MoveType`**:**

Lớp `MoveType` cung cấp thể loại các nước di chuyển trong game: thăng cấp qua cho quân Tốt (PROMOTION), nhập thành ngắn (SHORT_CASTLING), nhập thành dài (LONG_CASTLING), EN_PASSANT (bắt Tốt qua đường) hoặc không phải các nước đi trên (NONE).

Sơ đồ UML của lớp enum `MoveType` như sau:

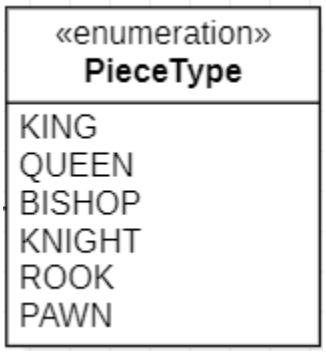


Hình 6: Sơ đồ UML của lớp enum `MoveType`

d. Lớp enum `PieceType`**:**

Lớp `PieceType` cung cấp các loại quân cờ: quân Vua (KING), quân Hậu (QUEEN), quân Tượng (BISHOP), quân Mã (KNIGHT), quân Xe (ROOK) và quân Tốt (PAWN).

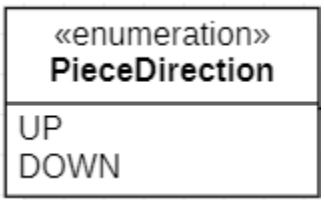
Sơ đồ UML của lớp enum `PieceType` như sau:



Hình 7: Sơ đồ UML của lớp enum `PieceType`

e. Lớp enum `PieceDirection`:

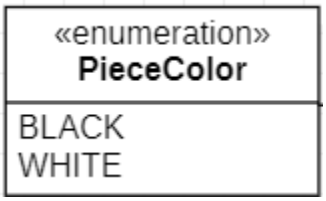
Lớp `PieceDirection` cung cấp hướng di chuyển thẳng về phía trước cho quân Tốt: đi lên (UP) đối với Tốt trắng và đi xuống (DOWN) đối với Tốt đen.
Sơ đồ UML của lớp enum `PieceDirection` như sau:



Hình 8: Sơ đồ UML của lớp enum `PieceDirection`

f. Lớp enum `PieceColor`:

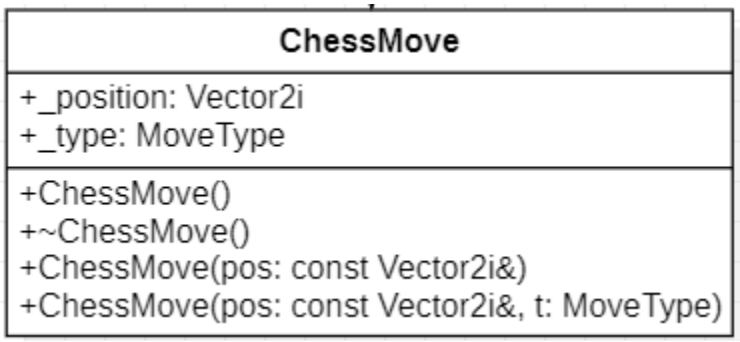
Lớp `PieceColor` cho biết hai màu của người chơi/quân cờ: màu trắng (WHITE) và màu đen (BLACK).
Sơ đồ UML của lớp enum `PieceColor` như sau:



Hình 9: Sơ đồ UML của lớp enum `PieceColor`

4. Lớp `ChessMove`:

Lớp `ChessMove` cung cấp thông tin về nước đi trên bàn cờ
Sơ đồ lớp UML của lớp `ChessMove` như sau:

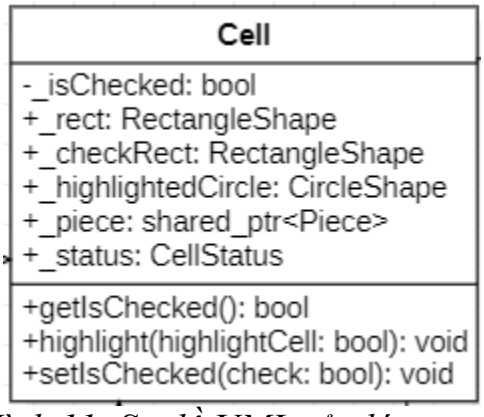


Hình 10: Sơ đồ UML của lớp `ChessMove`

Các thuộc tính của lớp `ChessMove` cho biết vị trí và thể loại của nước đi đó. Các phương thức khởi tạo của lớp `ChessMove` cho phép ta khởi tạo mặc định, khởi tạo với một tham số và khởi tạo với đầy đủ tham số.

5. Lớp `Cell`:

Lớp `Cell` thể hiện mỗi một ô vuông trên bàn cờ vua.
Sơ đồ lớp UML của lớp `Cell` như sau:



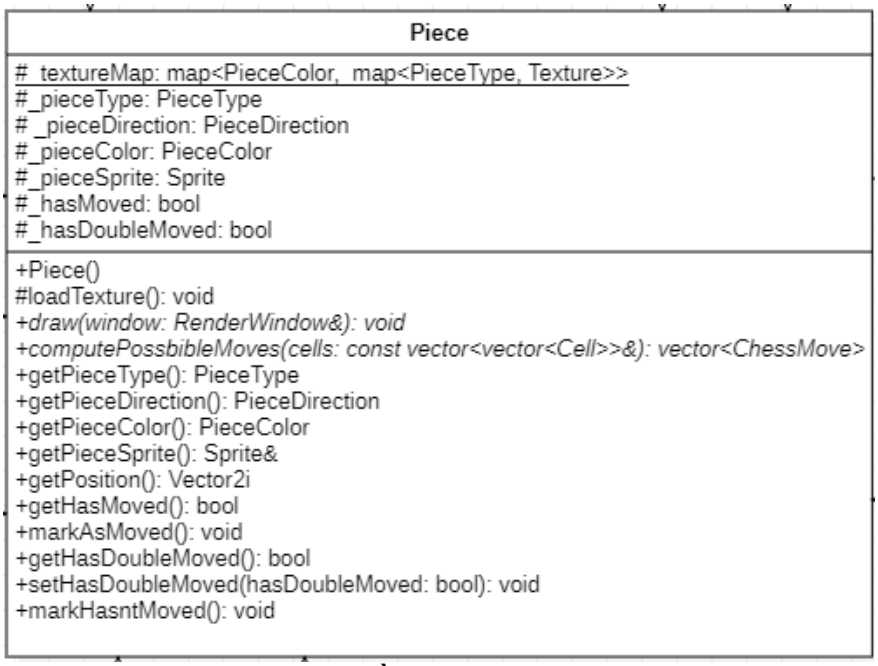
Hình 11: Sơ đồ UML của lớp `Cell`

Các thuộc tính chính của lớp `Cell` gồm có thuộc tính để đánh dấu xem ô này có đang bị chiếu hay không, một con trỏ `share_ptr` trỏ đến một đối tượng thuộc lớp `Piece` (sẽ được trình bày trong phần sau) – quân cờ hiện tại ở ô này (nếu ô này đang không bị chiếm giữ bởi quân cờ nào thì trỏ đến `nullptr`), thuộc tính thể hiện trạng thái hiện tại của ô vuông này. Ngoài ra các đối tượng từ lớp `RectanleShape` và `CircleShape` của thư viện SFML được sử dụng để tạo các thuộc tính nhằm phục vụ cho việc thiết kế giao diện người dùng.

Thuộc tính `_piece` có kiểu con trỏ `share_ptr` của thư viện STL để hạn chế tình trạng rò rỉ bộ nhớ (memory leak).

6. Lớp `Piece`:

Lớp `Piece` là một lớp trừu tượng, để biểu diễn một quân cờ tổng quát trên bàn cờ vua.
Sơ đồ UML của lớp `Piece` như sau:



Hình 12: Sơ đồ UML của lớp `Piece`

Các thuộc tính của lớp `Piece` được khai báo `protected` để cho các lớp kế thừa từ nó có thể truy cập vào các thuộc tính.

Lớp `Piece` có thuộc tính tĩnh `_textureMap` lưu trữ các `Texture` của lần lượt từng quân cờ của mỗi màu. Lớp `Texture` và `Sprite` được sử dụng từ thư viện SFML.

Các thuộc tính khác thể hiện màu sắc, loại quân cờ, hướng di chuyển cùng với hai thuộc tính kiểm tra xem quân cờ này đã di chuyển hay chưa và quân cờ có di chuyển hai bước hay chưa.

Lớp này cung cấp các getter và setter cho các thuộc tính, phương thức dùng để tải (load) các `Texture`. Ngoài ra, lớp `Piece` còn cung cấp các phương thức thuần ảo nhằm tận dụng tính đa hình *(polymorphism)*. Các phương thức thuần ảo này sẽ được cài đặt trong các lớp kế thừa cho phù hợp. Trong đó phương thức `computePossibleMoves` được sử dụng để tính toán các nước đi khả dĩ của từng quân cờ (sẽ được cài đặt lại trong các lớp con), trả về ô `position` mà quân cờ có thể đi mà không bị chặn bởi quân ta và quân địch (trong trường hợp quân ta có thể ăn quân địch) và nước đi này không làm vua bị chiếu (sẽ được kiểm tra ở các lớp sau).

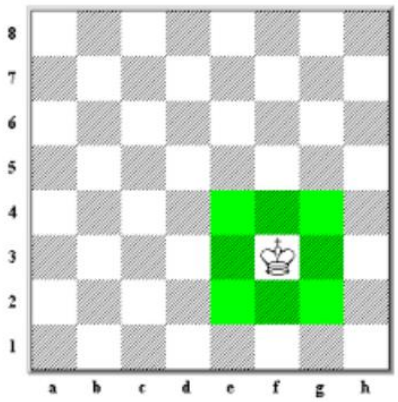
7. Lớp `King`:

Lớp `King` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Vua trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.
Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `King`.
Sơ đồ UML của lớp `King` như sau:

Hình 13.1: Sơ đồ UML của lớp `King`

Trong lớp `King` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Vua. Ngoài ra có phương vẽ quân Vua trên cửa sổ và tính toán các nước đi khả dĩ của quân Vua.

Hình sau mô tả các nước đi có thể của quân Vua, được viết trong phương thức `computePossibleMoves`.



Hình 13.2: Các nước đi của quân Vua.

8. Lớp `Queen`:

Lớp `Queen` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Hậu trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.

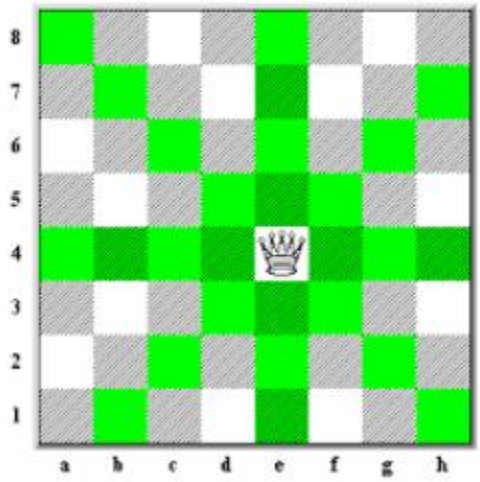
Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `Queen`.

Sơ đồ UML của lớp `Queen` như sau:

Hình 14.1: Sơ đồ UML của lớp `Queen`

Trong lớp `Queen` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Hậu. Ngoài ra có phương thức vẽ quân Hậu trên cửa sổ và tính toán các nước đi khả dĩ của quân Hậu.

Hình sau mô tả các nước đi có thể của quân Hậu, được viết trong phương thức `computePossibleMoves`.



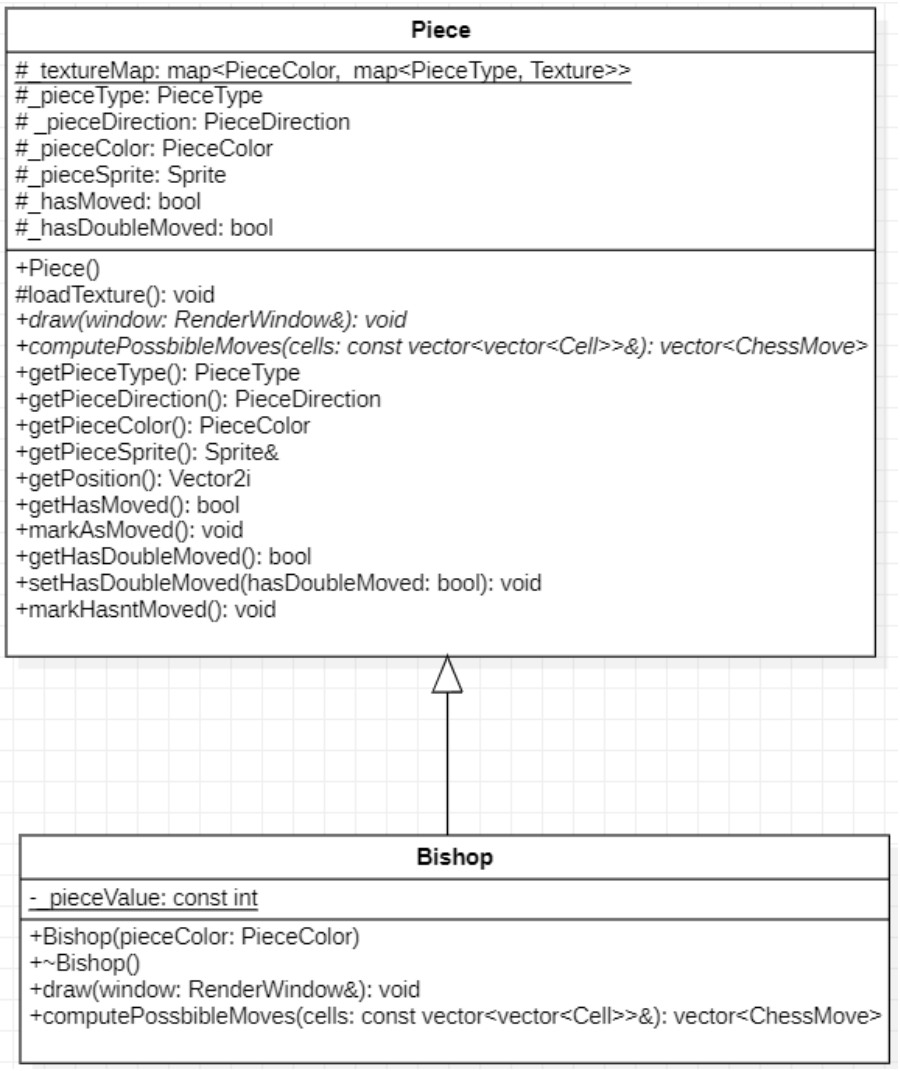
Hình 14.2: Các nước đi của quân Hậu

9. Lớp `Bishop`:

Lớp `Bishop` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Hậu trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.

Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `Bishop`.

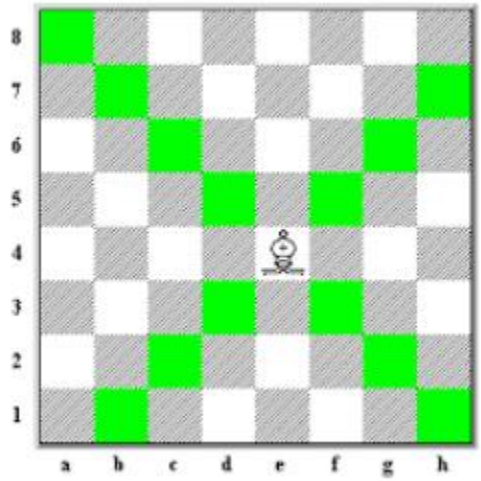
Sơ đồ UML của lớp `Bishop` như sau:



Hình 15.1: Sơ đồ UML của lớp `Bishop`

Trong lớp `Bishop` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Tượng. Ngoài ra có phương vẽ quân Tượng trên cửa sổ và tính toán các nước đi khả dĩ của quân Tượng.

Hình sau mô tả các nước đi có thể của quân Tượng, được viết trong phương thức `computePossibleMoves`.



Hình 15.2: Các nước đi của quân Tượng

10. Lớp `Knight`:

Lớp `Knight` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Mã trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.

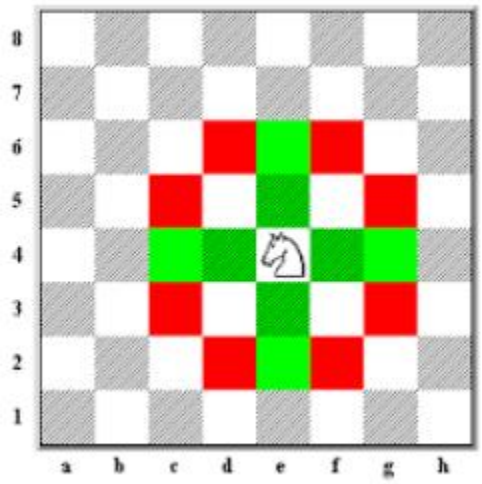
Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `Knight`.

Sơ đồ UML của lớp Knight như sau:

Hình 16.1: Sơ đồ UML của lớp `Knight`

Trong lớp `Knight` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Mã. Ngoài ra có phương thức vẽ quân Mã trên cửa sổ và tính toán các nước đi khả dĩ của quân Mã.

Hình sau (các ô màu đỏ) mô tả các nước đi có thể của quân Mã, được viết trong phương thức `computePossibleMoves`.



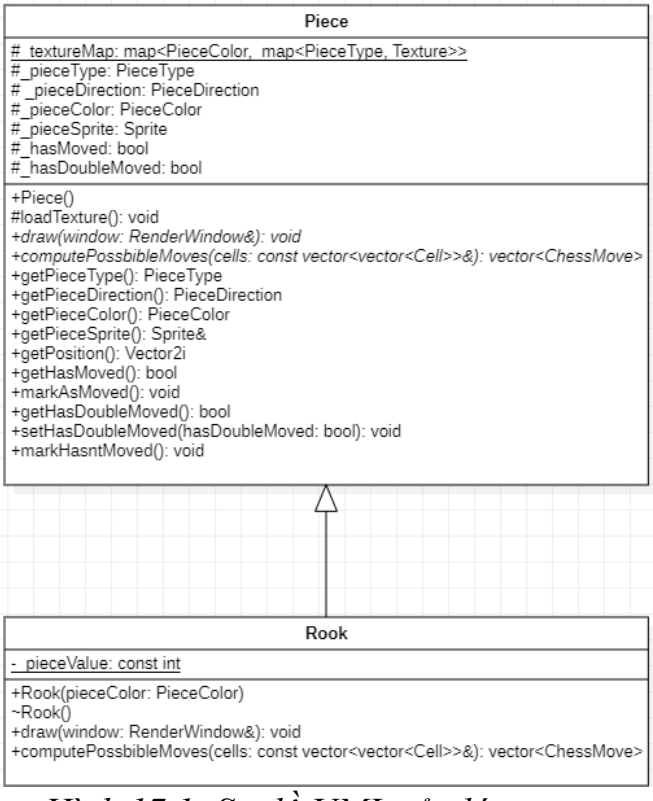
Hình 16.2: Các nước đi của quân Mã

11. Lớp `Rook`:

Lớp `Rook` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Xe trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.

Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `Rook`.

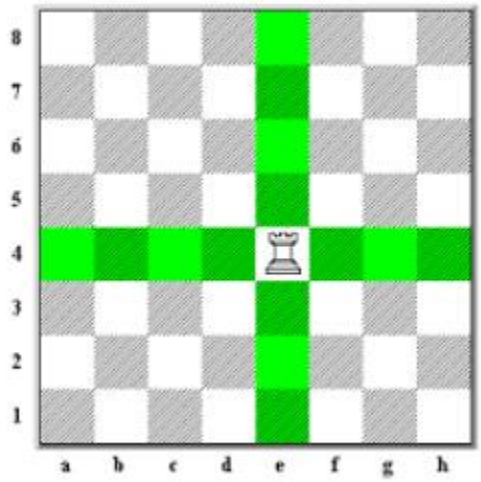
Sơ đồ UML của lớp `Rook` như sau:



Hình 17.1: Sơ đồ UML của lớp `Rook`

Trong lớp `Rook` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Xe. Ngoài ra có phương thức vẽ quân Xe trên cửa sổ và tính toán các nước đi khả dĩ của quân Xe.

Hình sau mô tả các nước đi có thể của quân Xe, được viết trong phương thức `computePossibleMoves`.



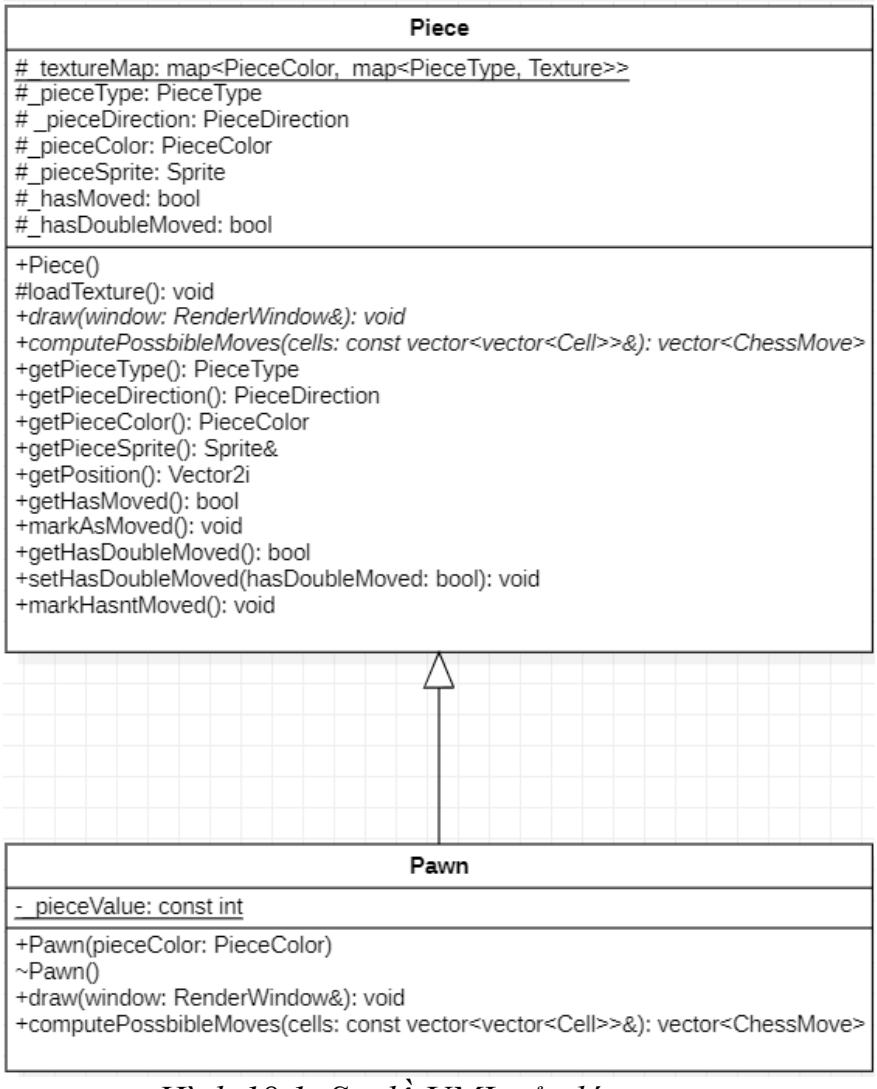
Hình 17.2: Các nước đi của quân Xe

12. Lớp `Pawn`:

Lớp `Pawn` là một lớp con kế thừa từ lớp `Piece`, thể hiện quân Tốt trên bàn cờ. Do là một lớp kế thừa nên nó kế thừa lại tất cả các thuộc tính và phương thức của lớp `Piece`.

Các phương thức thuần ảo của lớp `Piece` được viết lại trong lớp `Pawn`.

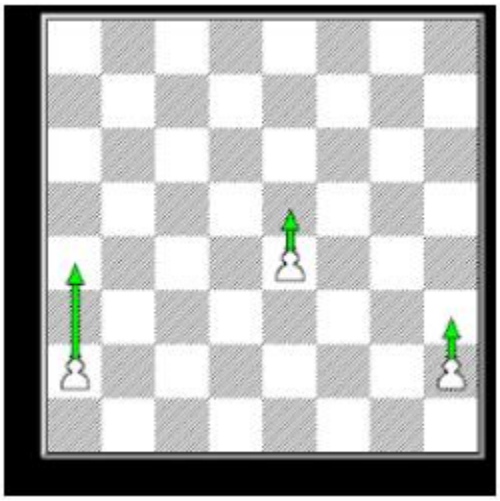
Sơ đồ UML của lớp `Pawn` như sau:



Hình 18.1: Sơ đồ UML của lớp `Pawn`

Trong lớp `Pawn` ta có thêm một thuộc tính tĩnh và hằng để cho biết giá trị của quân Tốt. Ngoài ra có phương thức vẽ quân Tốt trên cửa sổ và tính toán các nước đi khả dĩ của quân Tốt.

Hình sau mô tả các nước đi có thể của quân Tốt, được viết trong phương thức `computePossibleMoves`.

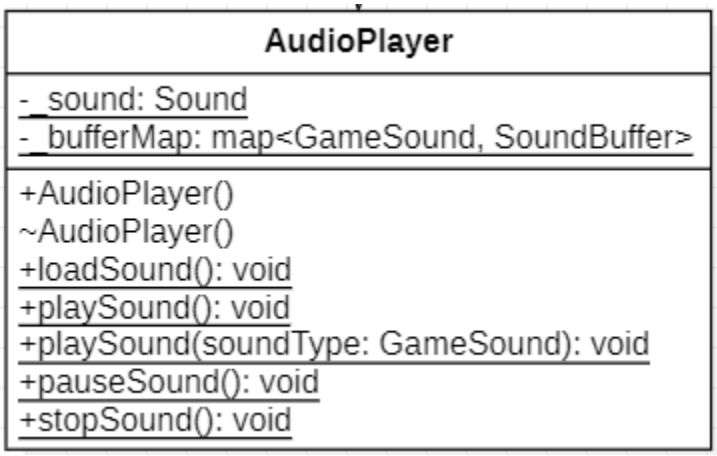


Hình 18.2: Các nước đi của quân Tốt

Việc phong cấp cho quân Tốt sẽ được xử lý ở lớp `ChessBoard`, sẽ được trình bày ở phần sau.

13. Lớp `AudioPlayer`:

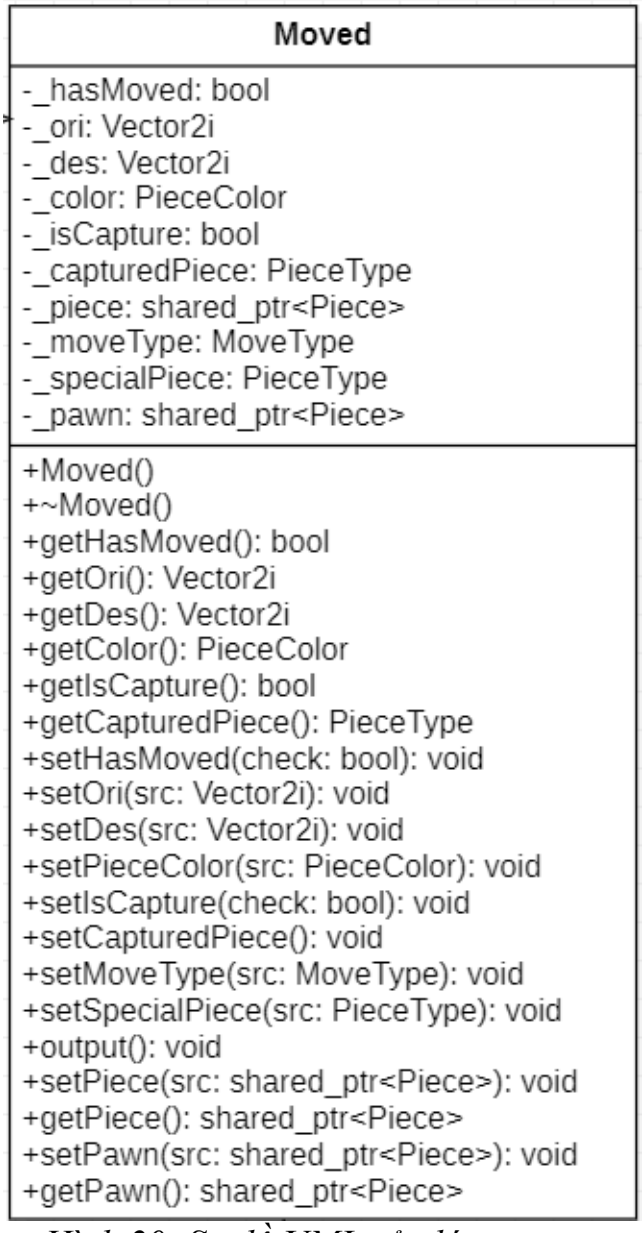
Lớp `AudioPlayer` chủ yếu cung cấp cho ta các phương thức liên quan đến việc xử lý âm thanh của game (load, play, resume, …).
Sơ đồ UML của lớp `AudioPlayer` như sau:



Hình 19: Sơ đồ UML của lớp `AudioPlayer`

14. Lớp `Moved`:

Lớp `Moved` được sử dụng để lưu trữ thông tin về một nước đi của một quân cờ trong trò chơi. Các thuộc tính và phương thức của lớp `Moved` cho phép lưu trữ và truy xuất thông tin về vị trí ban đầu và vị trí đích, màu sắc của quân cờ, trạng thái của nước đi, và các trường hợp đặc biệt hoặc quân cờ đặc biệt – quân Tốt (`Pawn`).
Sơ đồ UML của lớp `Moved` như sau:



Hình 20: Sơ đồ UML của lớp `Moved`

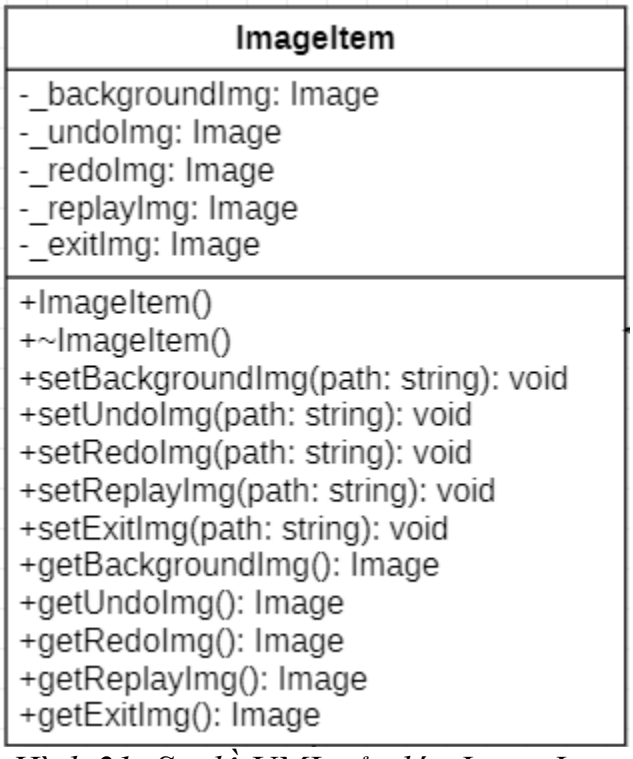
Các thuộc tính của lớp `Moved` gồm có kiểm tra quân cờ đã di chuyển chưa, vị trí đầu và đích, màu của quân cờ, kiểm tra quân cờ đã bị ăn hay chưa, 2 đối tượng enum `PieceType` đại diện cho loại quân cờ bị chiếm và sử

dụng cho các trường hợp đặc biệt như quân cờ được thăng cấp hoặc en_passant, một con trỏ `shared_ptr` để tham chiếm đến các đối tượng của lớp `Piece` đại diện cho quân cờ bị chiếm và một con trỏ `share_ptr` đến các đối tượng của lớp `Piece` đại diện cho quân Tốt, giúp quản lý bộ nhớ tự động và chia sẻ tham chiếu giữa các đối tượng.

Các phương thức getter và setter trong lớp cho phép truy xuất và thiết lập các thuộc tính của lớp `Moved`.

15. Lớp `ImageItem`:

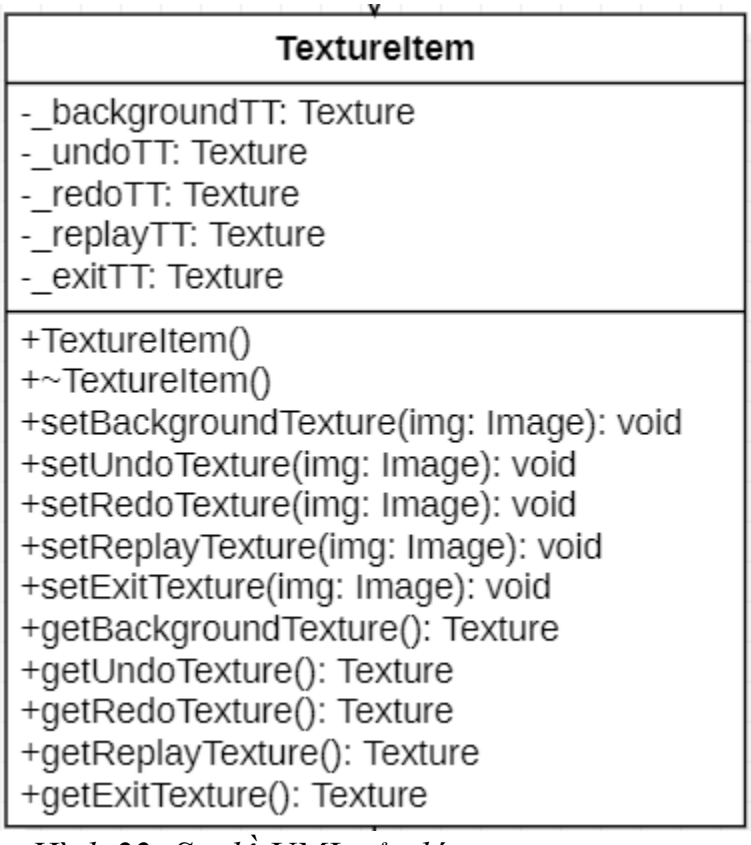
Lớp ImageItem quản lý việc truy xuất và thiết lập hình ảnh của hình nền, nút Undo, nút Redo, nút Replay, nút Exit với các thuộc tính kiểu Image.
Sơ đồ UML của lớp ImageItem như sau:



Hình 21: Sơ đồ UML của lớp ImageItem

16. Lớp `TextureItem`:

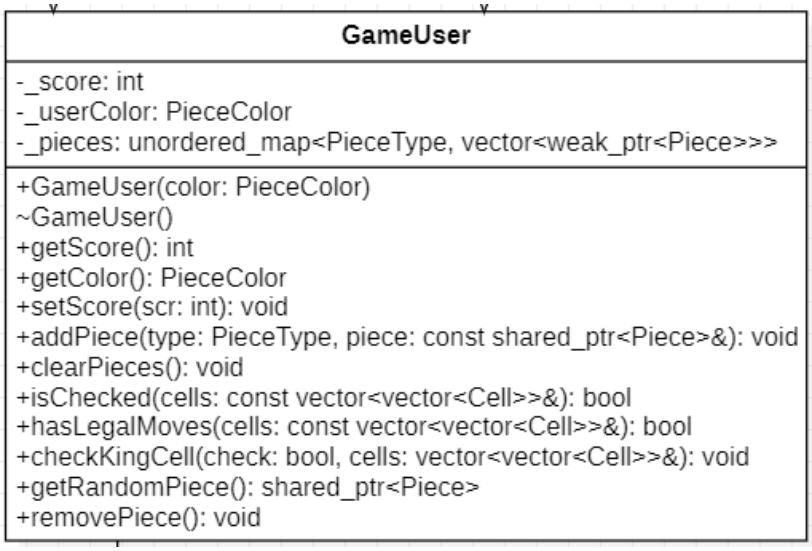
Lớp `TextureItem` lưu trữ các đối tượng `Texture`, cho phép thiết lập và truy xuất đến các đối tượng đại diện cho các hình ảnh như hình nền, hình nút Undo, hình nút Redo, hình nút Replay, hình nút Exit để tải lên giao diện người dùng trong trò chơi. Từ đó giúp quản lý và tương tác với các hình ảnh được sử dụng trong trò chơi một cách thuận tiện.
Sơ đồ UML của lớp `TextureItem` như sau:



Hình 22: Sơ đồ UML của lớp `TextureItem`

17. Lớp `GameUser`:

Lớp `GameUser` thể hiện người chơi game Cờ Vua (bên trắng và bên đen). Sơ đồ UML của lớp `GameUser` như sau:



Hình 23: Sơ đồ UML của lớp `GameUser`

Các thuộc tính của lớp `GameUser` gồm có điểm, màu và các quân cờ. Con trỏ `weak_ptr` của thư viện STL được sử dụng để tham chiếu đến các đối tượng được quản lý bởi `shared_ptr`.

Các phương thức của lớp `GameUser` bao gồm các setter, getter, các hàm giúp ta lấy điểm, màu của người chơi, thêm các quân cờ, kiểm tra xem người dùng có đang bị chiếu hay không, kiểm tra xem người dùng còn nước đi hợp lệ nào hay không, phương thức dùng để chiếu, lấy ra một loại quân cờ ngẫu nhiên (dùng cho chế độ chơi với máy) và xóa quân cờ.

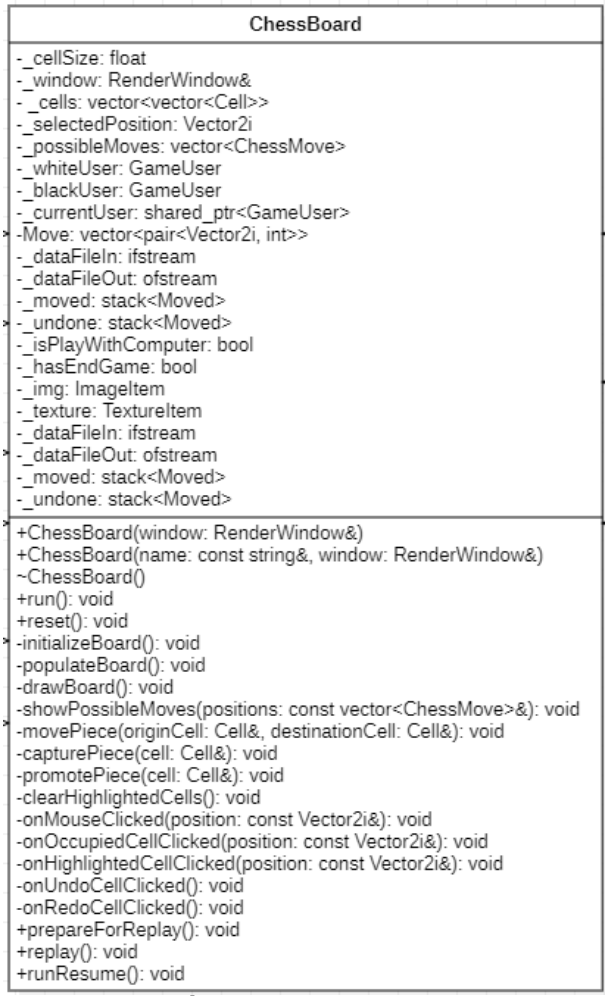
Trong đó:

- `isChecked`: kiểm tra quân Vua có đang bị chiếu hay không bằng cách kiểm tra đường chéo có bị chiếu bởi Tượng/Hậu, đường ngang có bị chiếu bởi Xe/Hậu, kiểm tra xem 8 ô xung quanh có bị chiếu bởi Tốt và hình vuông xung quanh có bị chiếu bởi Mã hay không.
- `hasLegalMove`: kiểm tra xem quân Vua bị chiếu thì có còn nước đi nào có thể đi nữa không, nếu không thì trả về `false` để kết thúc ván đấu, ngược lại trả về `true`.

18. Lớp `ChessBoard`:

Lớp `ChessBoard` thể hiện bàn cờ của game Cờ Vua trên cửa sổ game.

Sơ đồ UML của lớp `ChessBoard` như sau:



Hình 24: Sơ đồ UML của lớp `ChessBoard`

Các thuộc tính của lớp `ChessBoard` bao gồm cửa sổ game, mảng hai chiều các ô vuông trên bàn cờ, các nước đi có thể có, người chơi màu trắng, người chơi màu đen, người chơi hiện tại, `stack` lưu lại các nước đi để phục vụ cho replay, kiểm tra có đang trong chế độ chơi với máy, kiểm tra đã kết thúc ván đấu chưa trong chế độ chơi với máy, lấy hình ảnh, tải hình ảnh lên cửa sổ game, đọc các nước đi được ghi vào file, ghi các nước đi vào file và `stack` lưu lại các nước đi đã được undo.

Các phương thức có tầm vực **`private`** của lớp `ChessBoard` gồm các phương thức:

- `initializeBoard()`: khởi tạo một bàn cờ
- `populateBoard()`: đặt các quân cờ vào vị trí trên bàn cờ
- `drawBoard()`: dùng để hiển thị bàn cờ ra màn hình

- `showPossibleMoves()`: hiển thị những nước đi khả dĩ của quân cờ được chọn hiện tại.
- `movePiece()`: di chuyển một quân cờ và thao tác trên nước đi đã chọn
- `capturePiece()`: bắt (ăn) một quân cờ khác.
- `promotePiece()`: phong cấp cho quân Tốt khi đạt đủ điều kiện
- `onOccupiedCellClicked()`: Gọi hàm `computePossibleMoves` để trả về vector nước đi có thể, sau đó vẽ các chấm mờ thể hiện chúng lên giao diện bàn cờ
- `onHighlightedCellClicked`: Gọi hàm `capturePiece` để bắt (ăn) quân cờ của đối phương (nếu có), sau đó gọi hàm `movePiece` để thực hiện di chuyển quân cờ. Tiến hành kiểm tra xem bên đối phương còn có thể đi nước nào nữa hay không bằng hàm `isChecked` và `checkKingCell`, từ đó dò ra các trạng thái là "White wins", "Black wins" hoặc là "Stalemate". Hàm tiếp tục kiểm tra nếu đang chơi với máy và đến lượt của máy (màu quy định cho máy là màu đen) thì sẽ tiến hành random quân cờ và nước đi khả dĩ của máy để thực hiện. Sau khi di chuyển tiếp tục kiểm tra như trên xem ván đấu kết thúc hay chưa và kết thúc.
- `onUndoCellClicked`: Kiểm tra xem stack `_moved` đã có nước đi hay chưa (kể cả các nước đi đặc biệt như quân Tốt bị ăn bởi en_passant, con Tốt được thăng cấp…), nếu có rồi thì lấy nước đi cuối cùng của _move gán cho `lastmove` và loại bỏ nước đi này ra khỏi stack. Tiếp theo, nước đi đã hoàn tác được đẩy vào stack `_undone`. Di chuyển quân cờ về vị trí ban đầu từ biến `lastmove` còn quân cờ tại tọa độ đích trước đó sẽ được di chuyển đến tọa độ ban đầu và được cập nhật lại vị trí và trạng thái di chuyển của nó. Nếu nước đi cuối cùng có quân cờ bị chặn, tức là nước đi cuối cùng là một nước ăn quân thì quân cờ bị chặn sẽ được khôi phục tại tọa độ đích trước đó và được cập nhật trạng thái của ô đó. Sau đó, tiến hành kiểm tra có phải là các nước đi đặc biệt như en_passant, nhập thành dài, nhập thành ngắn hay không. Cuối cùng kiểm tra các điều kiện thắng như trong phương thức `onHighlightedCellClicked` và kết thúc hàm.
- `onRedoCellClicked`: Kiểm tra stack `_undone`, nếu không rỗng thì lấy nước đi cuối cùng từ `_undone` và gán và `lastUndo`, sau đó loại bỏ nước đi này khỏi stack. Tiếp theo, cách thức hoạt động của chức năng tương tự như trong `onUndoCellClicked`.

Các phương thức có tầm vực **`public`** (trừ hàm khởi tạo và hàm hủy):

- `run()`: biểu diễn bàn cờ lên GUI. Đầu tiên bắt vị trí click chuột của người dùng, trả về một trong 4 phương thức sau dựa theo tọa độ của vị trí click chuột đó
  - Nếu click vào quân cờ bất kỳ trên bàn cờ, gọi phương thức `onOccupiedCellClicked`.
  - Nếu click vào các ô có chấm mờ, gọi phương thức `onHighlightedCellClicked`.
  - Nếu click vào nút Undo, gọi phương thức `onUndoCellClicked`.
  - Nếu click vào nút Redo, gọi phương thức onRedoCellClicked.
  - Nếu click vào nút Exit, đóng cửa sổ game.
- `reset()`: reset bàn cờ
- `runResume()`: biểu diễn ván cờ trước đó. Đọc dữ liệu từ file `data.txt` trước để di chuyển các quân cờ đến trạng thái của bàn cờ trước đó. Sau đó, tiếp tục chạy giống hàm `run`.
- `prepareForReplay()`: chuẩn bị cho replay ván cờ
- `replay()`: replay ván cờ. Bàn cờ được đặt về trạng thái ban đầu bằng phương thức reset. Tạo ra biến fx và fy để lưu tọa độ của click chuột. Mở file data.txt đã có sẵn (nếu có) và đọc các tọa độ đã được lưu vào trước đó, kiểm tra tọa độ của các cú click chuột trong file để gọi đúng phương thức trước đó.

Phần cài đặt của lớp `ChessBoard` sử dụng nhiều các lớp và phương thức của chúng trong thư viện SFML.

C.MÔ TẢ TÍNH NĂNG PHẦN MỀM

Đồ án game Cờ Vua 2 người cung cấp một phần mềm mô phỏng môn thể thao trí tuệ Cờ Vua, với tính năng 2 người chơi và tính năng chơi với máy. Phần mềm cung cấp đầy đủ các tính năng cơ bản của một game Cờ Vua như: các bước di chuyển, tính năng bắt quân cờ, chiếu tướng, phong cấp cho quân Tốt, nhập thành, … và các tính năng nâng cao như: undo, redo, replay ván đấu, tiếp tục ván đấu.

Bài báo cáo được đính kèm với file README hướng dẫn chạy chương trình.

I. Màn hình khởi động của game:

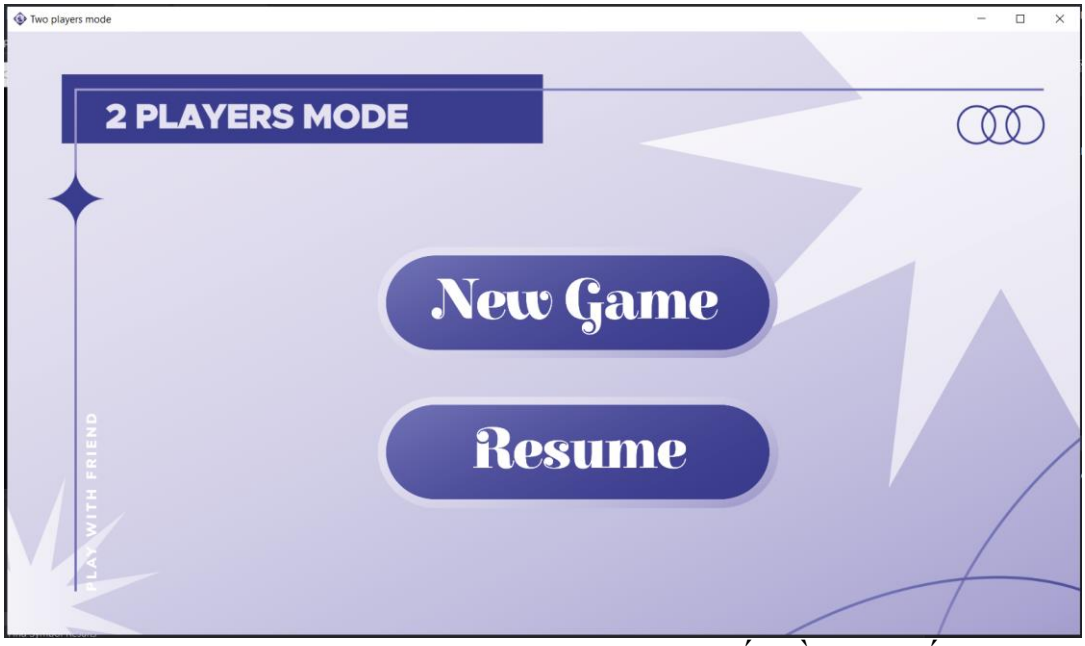
Hình sau đây là màn hình khởi động của game, với các chế độ chơi như chơi với người hoặc chơi với máy và một nút thoát game.

Hình 25: Màn hình khởi động của trò chơi

II. Màn hình bắt đầu ván cờ:

1. Trước khi bắt đầu:

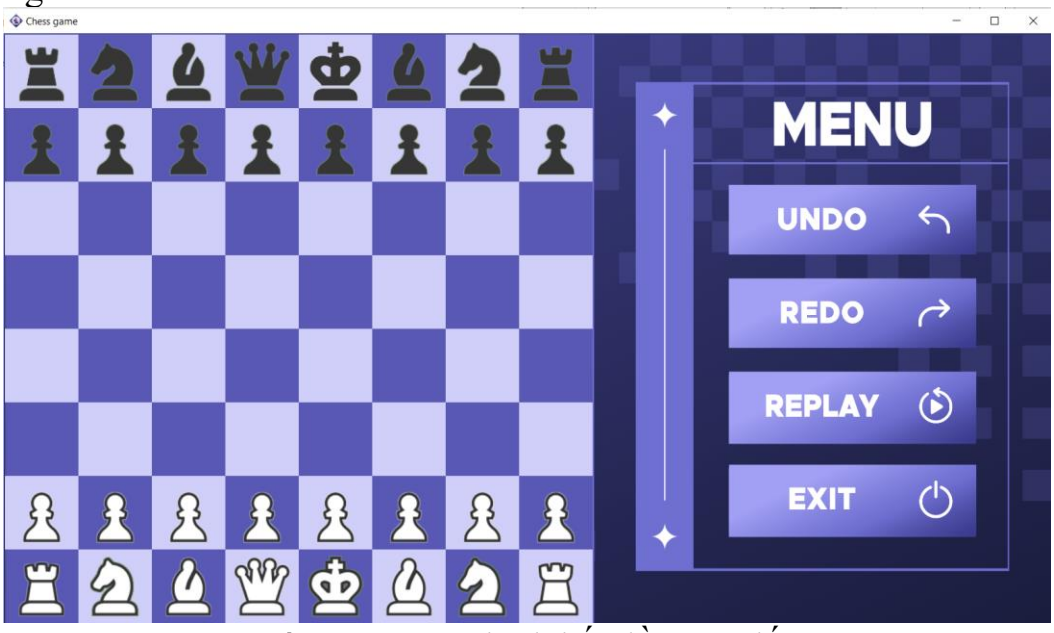
Ở đây chúng ta có hai lựa chọn như hình, bắt đầu một ván đấu mới hoặc tiếp tục ván đấu trước đó.



Hình 26: Màn hình lựa chọn trước khi bắt đầu ván đấu

2. **Sau khi bắt đầu**

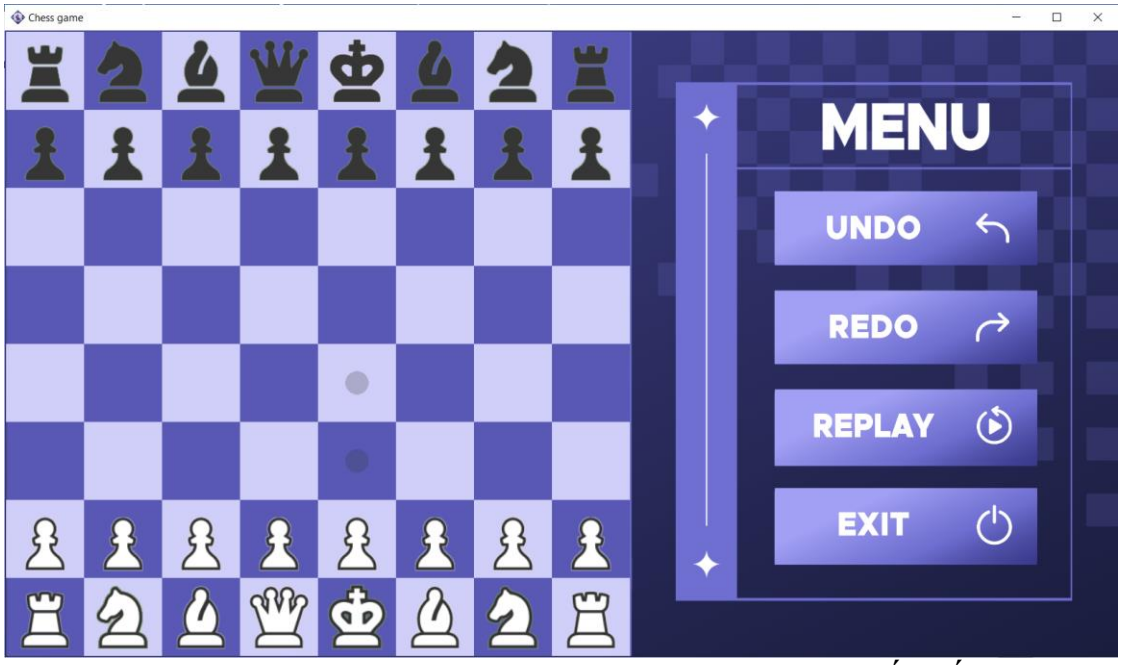
Màn hình ván đấu với đầy đủ các quân cờ đã được sắp xếp vào vị trí, sẵn sàng cho ván đấu:



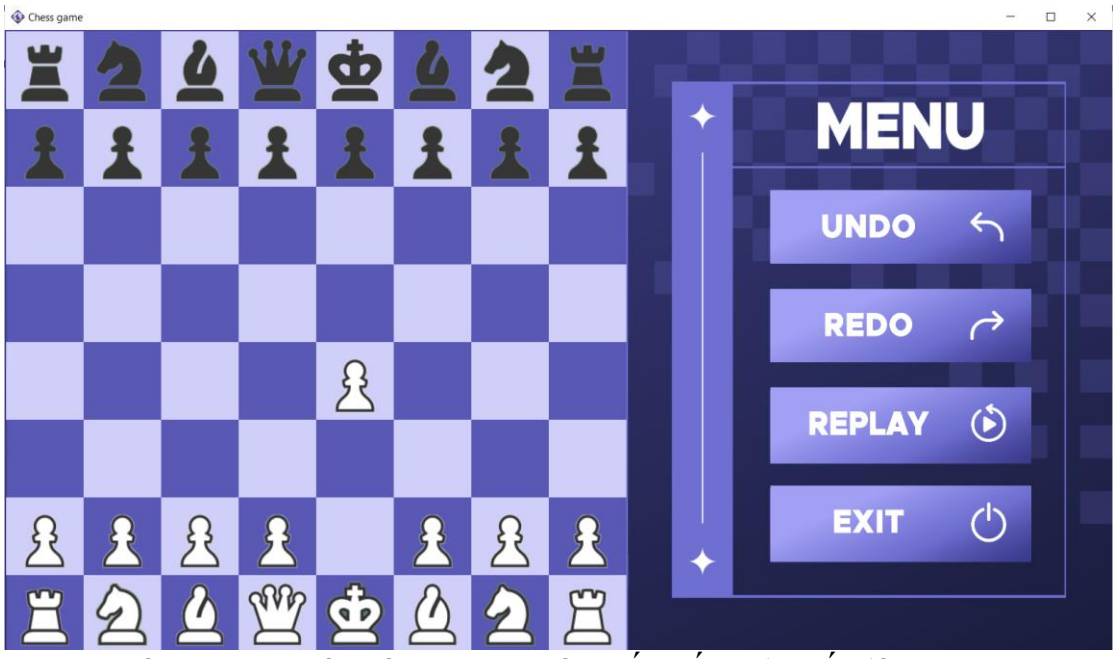
Hình 27: Màn hình bắt đầu ván đấu

III. Di chuyển quân cờ:

Các nước đi đầu tiên của game, với quân Tốt trắng tiến lên hai bước.



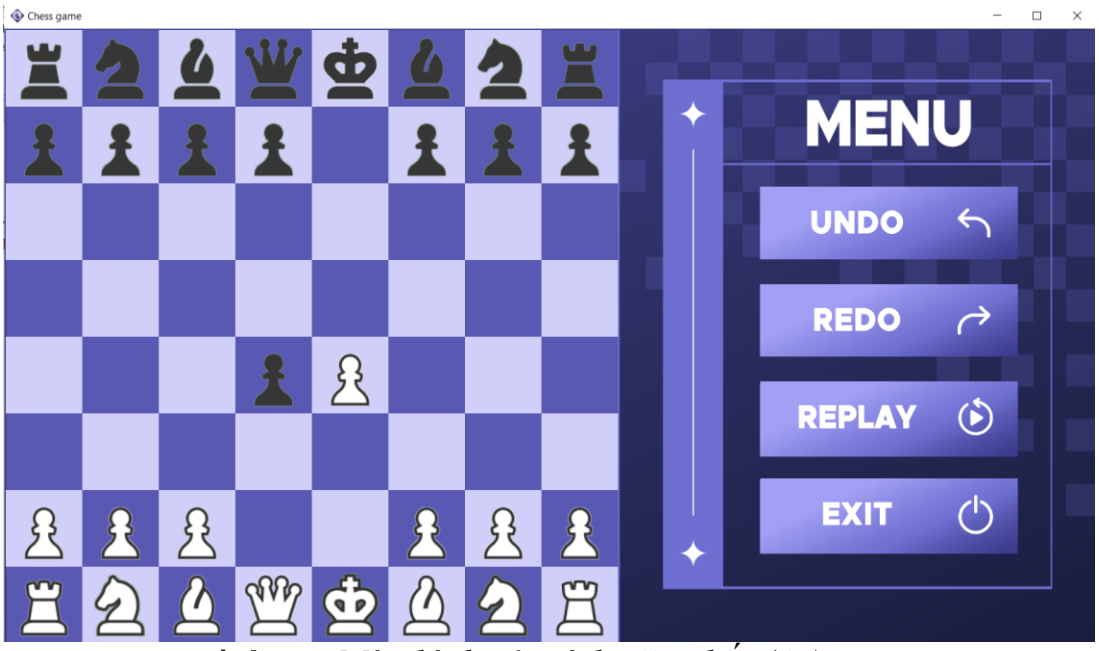
Hình 28.1: Màn hình hiện các nước đi khả dĩ của quân Tốt trắng ở vị trí click chuột



Hình 28.2: Màn hình hiện quân Tốt trắng đã tiến lên 2 bước

IV. Tính năng bắt (ăn quân):

Hình sau đây là ví dụ về tính năng bắt ăn quân theo đường chéo của quân Tốt đen đối với quân Tốt trắng ở ô d4.



Hình 29: Màn hình của tính năng bắt (ăn) quân

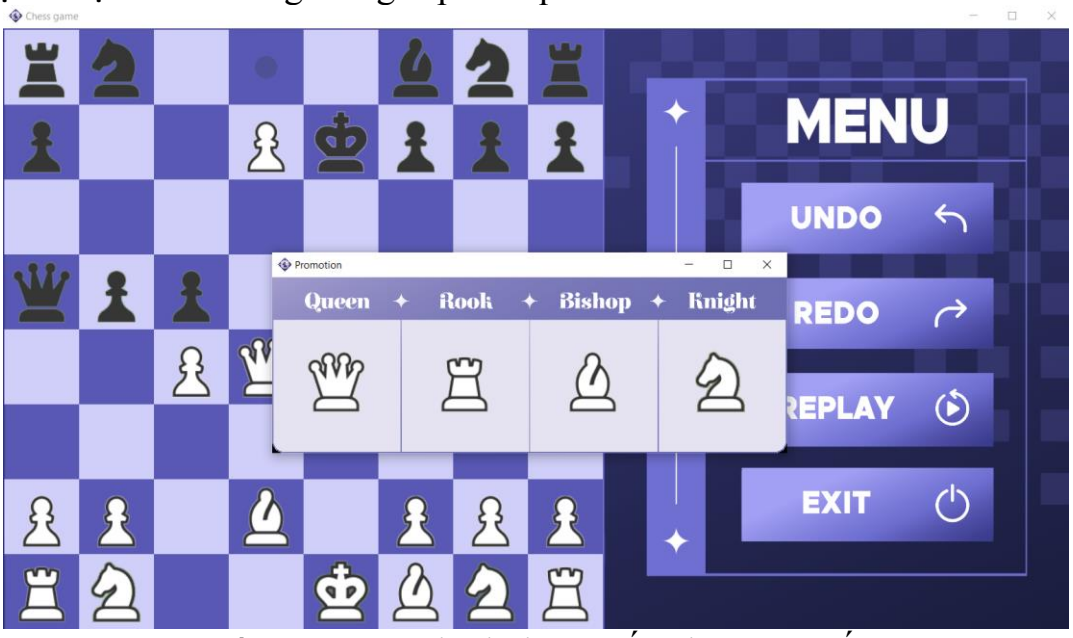
V. Chiếu:

Hình ảnh sau đây mô tả tình huống quân Vua bị chiếu, tức là quân Vua nằm trong một ô có thể bị bắt ở các quân cờ của đối thủ. Ô bị chiếu hiện tại sẽ được tô màu đỏ. Sau khi bị chiếu, quân Vua phải di chuyển đến nơi an toàn hoặc những quân cờ khác phải di chuyển để "ngăn chặn" đường chiếu của quân cờ đối thủ, thậm chí có thể bắt quân cờ này để bảo vệ Vua. Nếu không thể, ván đấu sẽ kết thúc và bên bị chiếu sẽ thua cuộc.

Hình 30: Chiếu

VI. Thăng cấp (Phong cấp) cho quân Tốt:

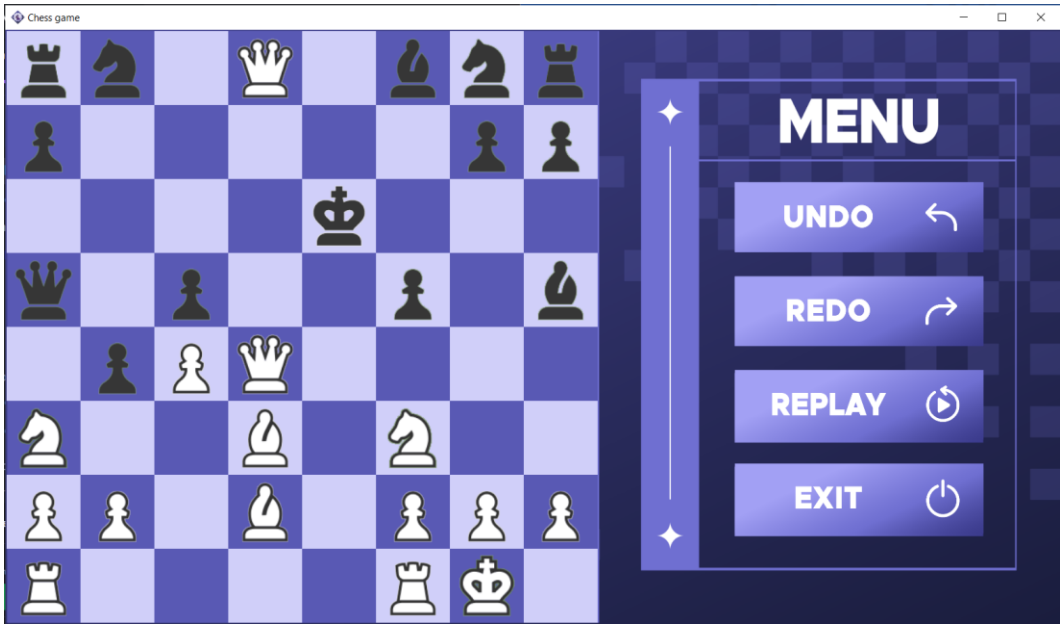
Khi quân Tốt tiến đến hàng cuối cùng trên "địa phận" của đối thủ, nó sẽ được thăng cấp thành một trong 4 quân: Hậu, Xe, Tượng, Mã. Hình sau đây mô tả một ví dụ về tính năng thăng cấp cho quân Tốt.



Hình 31: Màn hình thăng cấp cho quân Tốt

VII. Nhập thành:

Khi quân Vua và quân Xe của một bên chưa di chuyển, các ô giữa chúng đều trở thành ô trống và ô đích đến không bị chiếu thì quân Vua có thể tiến hành nhập thành. Có 2 loại nhập thành, ứng với hai quân Xe, lần lượt là *long castling* ("nhập thành dài") - ứng với quân Xe ở xa Vua hơn và *short castling* ("nhập thành ngắn") - ứng với quân Xe ở gần Vua hơn. Hình sau đây cho ta một ví dụ về nhập thành ngắn bên người chơi trắng.



Hình 32: Màn hình nhập thành

VIII. Undo:

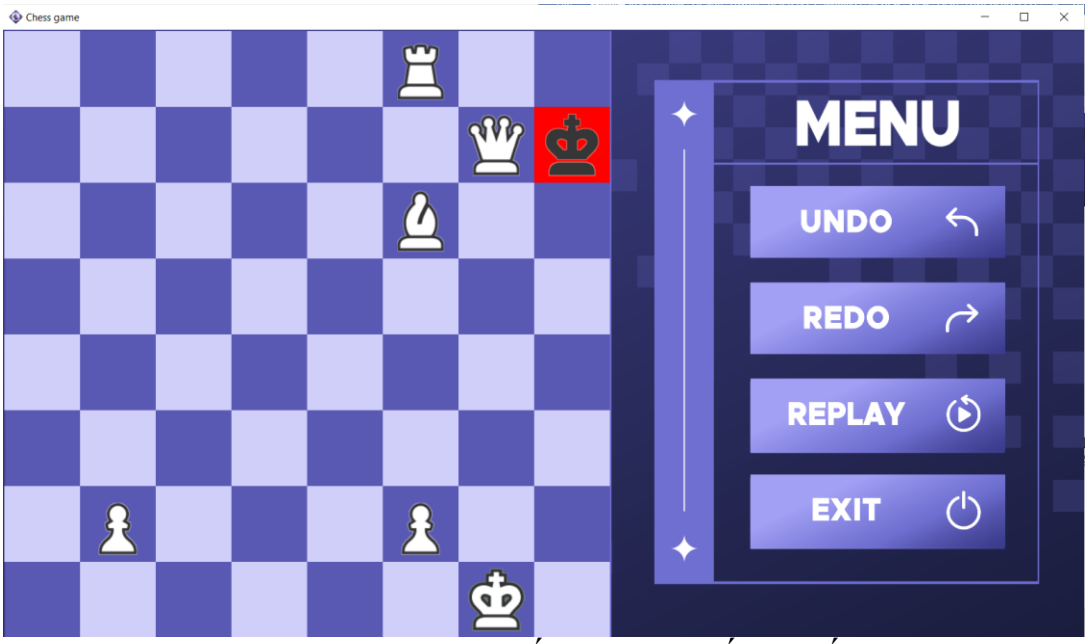
Đi lại nước đi trước đó bằng nút Undo ở Menu bên phải
Minh họa trong video demo ở phần sau.

IX. Redo:

Đi lại nước đi đã Undo trước đó bằng nút Redo ở Menu bên phải
Minh họa trong video demo ở phần sau.

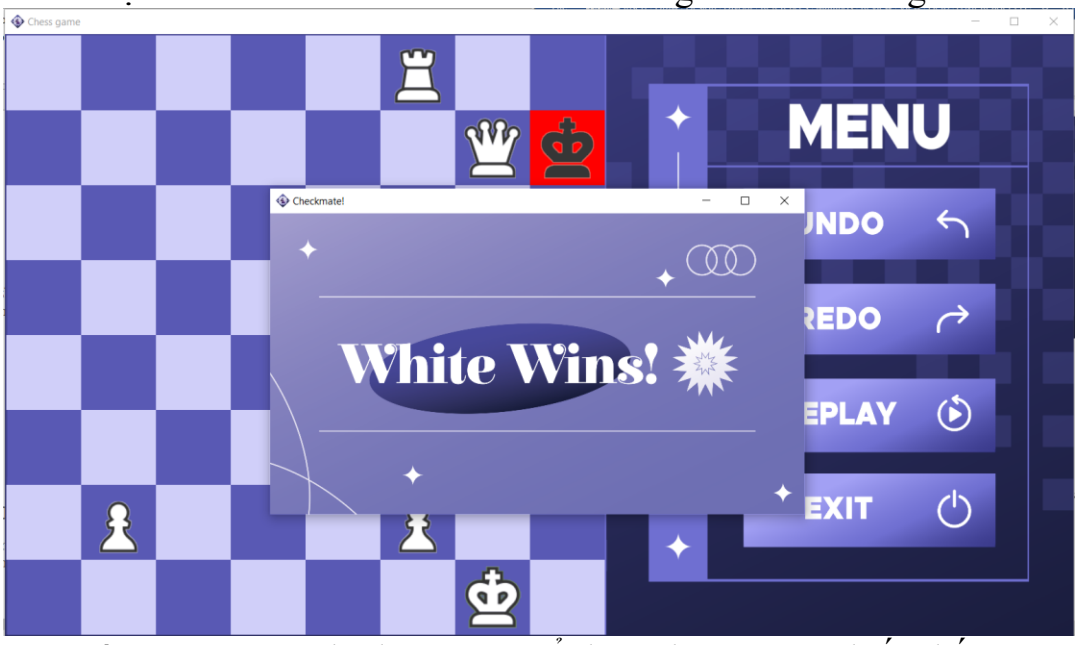
X. Kết thúc ván đấu (chiếu bí):

Khi quân Vua bị chiết và không còn nước đi nào có thể "hộ giá" quân Vua ra khỏi ô bị chiếu này thì ván đấu sẽ kết thúc với chiến thắng dành cho bên chiếu. Hình sau đây cho ta một ví dụ về bàn cờ lúc kết thúc ván đấu.

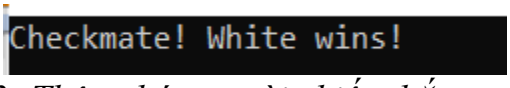


Hình 33.1: Màn hình kết thúc ván đấu (chiếu bí)

Khi tình huống này xảy ra, một cửa sổ thông báo sẽ hiện lên và một thông báo sẽ hiện lên màn hình console cho ta biết người chiến thắng.



Hình 33.2: Màn hình của cửa sổ thông báo người chiến thắng



Hình 33.3: Thông báo người chiến thắng mở console

XI. Thoát game:

Ta có thể thoát game bằng cách bấm vào nút Exit ở Menu bên phải hoặc tắt cửa sổ game.

D. VIDEO DEMO

Nhóm đã tiến hành quay video demo của game Cờ Vua, gồm các chức năng cơ bản, nâng cao và demo 1 ván đấu cờ vua.

Link video demo:

https://drive.google.com/file/d/16mrTAOnfpS7m4DIZFncx579bhoyJfsB6/view?usp=sharing

E. HOẠT ĐỘNG NHÓM ĐỒ ÁN

1. Giới thiệu thành viên nhóm:

Nhóm gồm 4 thành viên của lớp 21CTT2:

STT	Họ và tên	MSSV	Tỉ lệ đóng góp
1	Dương Ngọc Thái Bảo	21120041	30%
2	Trương Tấn Đạt	21120050	28%
3	Phạm Nhật Duy	21120058	27%
4	Nhan Hữu Hiếu	21120070	15%

2. Nguyên tắc hoạt động nhóm:

- Các thành viên phải hoàn thành công việc đúng thời hạn đã giao.
- Các thành viên phải tham gia đúng giờ và đầy đủ các buổi họp (được phép trễ tối đa 15 phút với điều kiện phải thông báo trước cho cả nhóm biết).
- Các source code sau khi hoàn thành phải được up lên GitHub chung của nhóm.

- Khi gặp khó khăn trong quá trình thực hiện công việc được giao, các thành viên chủ động liên lạc với nhau để cùng thảo luận.

3. Đánh giá mức độ hoàn thành chung của đồ án:

- Mức độ hoàn thành của cả đồ án: 100%
- Chi tiết: Hoàn thành, đáp ứng được các yêu cầu cơ bản, thực hiện được các yêu cầu nâng cao như: thao tác thông qua click chuột, chế độ chơi với máy (mức Dễ), Undo, Redo, Replay (xem lại ván vừa đấu), Giao diện người dùng, âm thanh trong trò chơi.

4. Phân công công việc và đánh giá mức độ hoàn thành của từng thành viên:

STT	Họ và tên	MSSV	Nội dung công việc	Đánh giá mức độ hoàn thành
1	Dương Ngọc Thái Bảo	21120041	• Tham gia xây dựng lớp King, quân Rook, lớp Piece và các lớp enum trong PieceData, lớp Moved, hỗ trợ xây dựng lớp ChessBoard • Chức năng Undo, Redo của chế độ chơi với máy và người, lưu và tiếp tục chơi ván đấu trước đó	Hoàn thành tốt
2	Trương Tấn Đạt	21120050	• Tham gia xây dựng lớp Knight, lớp Bishop, lớp Queen, lớp Settings,	Hoàn thành tốt

			namespace utilities, lớp TextureItem, lớp ImageItem, hỗ trợ xây dựng lớp ChessBoard • Chơi với máy dễ (random quân cờ và nước đi của máy dễ) • Viết báo cáo, vẽ sơ đồ UML.	
3	Phạm Nhật Duy	21120058	• Tham gia xây dựng lớp Pawn, lớp AudioPlayer, lớp enum GameSound, lớp GameUser, triển khai hàm main. • Tham gia xây dựng thao tác click chuột, giao diện người dùng, âm thanh trò chơi, cửa sổ thông báo. • Format báo cáo	Hoàn thành tốt
4	Nhan Hữu Hiếu	21120070	• Tham gia xây dựng lớp Cell, lớp ChessBoard, lớp ChessMove • Tham gia xây dựng chức năng replay (xem lại ván đấu trước đó).	Hoàn thành tốt

F. TÀI LIỆU THAM KHẢO

[1] *Simple and fast multimedia library* (no date) *SFML*. Available at: https://www.sfml-dev.org/.

[2] *SFML/SFML: Simple and fast multimedia library*, *GitHub*. Available at: https://github.com/SFML/SFML.

[3] SkizzSkizz, *What does the explicit keyword mean?*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/121162/what-does-the-explicit-keyword-mean.

[4] *Documentation of SFML 2.5.1* (no date) *Main Page (SFML / Learn / 2.5.1 Documentation)*. Available at: https://www.sfml-dev.org/documentation/2.5.1/.

[5] *Dynamic memory management* (no date) *cppreference.com*. Available at: https://en.cppreference.com/w/cpp/memory.