**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PBL2: DỰ ÁN CƠ SỞ LẬP TRÌNH**

**Đề tài: Trò Chơi Cờ Vua**

**GIẢNG VIÊN HƯỚNG DẪN: TS.Đặng Thiên Bình**

**SINH VIÊN THỰC HIỆN:**

**Tên sinh viên: Nguyễn Bảo Ngọc LỚP: 24T\_KHDL NHÓM: 24.Nh15B**

**Tên sinh viên: Dương Tấn Khôi LỚP: 24T\_KHDL NHÓM: 24.Nh15B**

**Đà Nẵng 11/2025**

LỜI MỞ ĐẦU

Trong thời đại mà khoa học và công nghệ phát triển vượt bậc và ngày càng thăng tiến thì đi đôi với nó là nhu cầu của con người đối với những sản phẩm từ khoa học và công nghệ ngày càng tăng cao. Con người hiện nay vẫn đang tìm hiểu, nghiên cứu lĩnh vực này để đáp ứng được những nhu cầu liên tục tăng ấy, đặc biệt là trong lĩnh vực giải trí - nơi thu hút đông đảo người dùng trên toàn cầu.

Những đột phá trong công nghệ đã tạo ra những ứng dụng và trò chơi với những tính năng cùng đồ hoạ mà ở những giai đoạn trước được xem là những điều rất khó khăn vì trình độ còn hạn hẹp. Đề tài là cơ sở để nhóm có thể rèn luyện tư duy thiết kế hệ thống, giải thuật hợp lý dựa vào việc ứng dụng kiến thức đã học từ 2 môn lập trình Hướng đối tượng và Cấu trúc dữ liệu, bên cạnh đó là làm quen, tiếp cận với các công cụ hỗ trợ mạnh mẽ cho việc lập trình như thư viện đồ hoạ.

Qua việc thực hiện và phát triển đề tài “Trò chơi Cờ Vua” đã giúp nhóm có cái nhìn sâu sắc hơn trong lập trình hướng đối tượng, thiết kế cấu trúc dữ liệu cũng như việc thiết kế một hệ thống, quản lý bộ nhớ và tài nguyên máy một cách tối ưu và an toàn để lấy đó làm nền tảng để có thể phát triển những dự án phức tạp hơn trong tương lai.

Lập trình hướng đối tượng (Object Oriented Programming) và Cấu trúc dữ liệu (Data Structure) là hai trụ cột cực kỳ quan trọng của mọi hệ thống phần mềm từ nhỏ cho đến lớn. Trong đó, Lập trình hướng đối tư là nền móng về kiến trúc để thiết kế một hệ thống, dễ dàng mở rộng và tái sử dụng lại mã nguồn có sẵn. Còn Cấu trúc dữ liệu đóng vai trò là nền móng về tối ưu hoá và quản lý dữ liệu, nhờ tổ chức dữ liệu hợp lý và các giải thuật xử lý hiệu quả, chương trình đạt được hiệu năng cao, ổn định và hạn chế phát sinh lỗi trong quá trình vận hành. Cả hai kết hợp và bổ trợ cho nhau, trở thành “xương sống” của hầu hết các ứng dụng, phần mềm hiện nay. Chính vì vậy mà nhóm đã chọn cờ vua làm đề tài để phát triển dựa trên lập trình hướng đối tượng và cấu trúc dữ liệu.

MỤC LỤC [1](#_m79l0hhemewp)

[LỜI MỞ ĐẦU 2](#_jdivr6ttsutj)

[MỤC LỤC 3](#_h0nrty4g5xbs)

[DANH MỤC HÌNH VẼ 4](#_mqyc4me1zzay)

1. [GIỚI THIỆU ĐỀ TÀI 5](#_ad03827wpzmo)
   1. Lý do chọn đề tài 5
   2. Phạm vi và đối tượng nghiên cứu 5
2. CƠ SỞ LÝ THUYẾT 5
   1. Giới thiệu về C++ 5
   2. Thư viện đồ hoạ Raylib 5
   3. Ưu điểm của Raylib khi làm game và khi kết hợp với C++ 5
3. PHÂN TÍCH CHỨC NĂNG HỆ THỐNG 5
   1. Chức năng người dùng 5
   2. Các chức năng xử lý của hệ thống 5
   3. Các chức năng AI 5
   4. Giao diện người dùng 5
4. THIẾT KẾ CẤU TRÚC DỮ LIỆU 5
   1. Stack 5
   2. Vector 5
5. MÔ TẢ THIẾT KẾ GAME 5
6. PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG 5
   1. Cấu trúc hệ thống hướng đối tượng (class, object, relation) 5
   2. Kết quả 5
      1. Giao diện chính của chương trình 5
      2. Kết quả thực thi của chương trình 5
      3. Nhận xét 5
7. [KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 5](#_fyu3xsf6dags)
   1. Kết luận 5
   2. Hướng phát triển 5

[TÀI LIỆU THAM KHẢO 6](#_33ov3jh2lu52)

[[1] Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản 6](#_25j21f8v3nc2)

[[2] Tên chủ sở hữu, Tên bài viết, url, ngày truy cập 6](#_ctu6gzhmqjx3)

DANH MỤC HÌNH VẼ

No table of figures entries found.

# GIỚI THIỆU ĐỀ TÀI

## Lý do chọn đề tài

Từ rất lâu về trước cờ Vua đã được cả thế giới biết đến và dần trở nên phổ biến trên toàn thế giới, đây là một trò chơi trí tuệ và mang tính chiến thuật cao. Cờ Vua không chỉ là một trò chơi mà nó còn 1 cuộc đấu trí đầy căng thẳng giữa 2 cờ thủ với nhau vì chỉ cần mắc phải 1 sai lầm thôi cũng có thể đủ để thua một ván cờ. Chính vì những yếu tố đó đã giúp cờ vua ngày càng được ưa chuộng trên thế giới kể cả Việt Nam.Cờ Vua là một trò chơi có luật lệ rõ ràng, không quá phức tạp,dễ dàng tiếp cận cũng như việc mổ phỏng lại dựa vào lập trình. Ngoài ra, game cờ Vua có chế độ chơi với máy (AI) và hướng đối tương nên nhóm đã chọn đây làm đề tài để có thể làm quen với thuật toán tìm kiếm cơ bản cũng như lập trình hướng đối tượng, hiểu sâu hơn về ngôn ngữ lập trình C++.

## Phạm vi và đối tượng nghiên cứu

Phạm vi nghiên cứu là lập trình hướng đối tượng, cấu trúc dữ liệu và thuật toán tìm kiếm đơn giản (minimax).

Đối tượng nghiên cứu: ngôn ngữ lập trình C++, thư viện đồ hoạ Raylib (một thư viên đồ hoạ 2D đơn giản và hỗ trợ cho C++).

# CƠ SỞ LÝ THUYẾT

## Giới thiệu về C++

C++ là một ngôn ngữ lập trình mạnh mẽ, linh hoạt và hiệu suất cao. Nó được phát triển bởi Bjarne Stroustrup vào năm 1983 và là sự mở rộng của ngôn ngữ C, tích hợp các tính năng lập trình hướng đối tượng (OOP).

C++ thường được sử dụng trong các ứng dụng yêu cầu tốc độ xử lý cao như:

* + - Lập trình hệ thống (hệ điều hành, trình biên dịch).
    - Phần mềm đồ họa và trò chơi.
    - Ứng dụng nhúng, thiết bị IoT.
    - Phần mềm tài chính, thời gian thực.

C++ là một ngôn ngữ lập trình cho phép truy cập “cấp thấp” vào phần cứng, kiểm soát chặt chẽ bộ nhớ và tài nguyên máy tính. Chính vì thế mà C++ đã được sử dụng làm ngôn ngữ chính trong ngành lập trình game đặc biệt là các game lớn (AAA) nơi mà sự tối ưu được đặt lên hàng đầu.

## Thư viện đồ hoạ Raylib

Raylib là một **thư viện lập trình đồ họa mã nguồn mở (open-source) được viết bằng ngôn ngữ C**, được thiết kế để giúp việc học lập trình game và đồ họa trở nên dễ dàng, trực quan và thân thiện với người mới.

* + 1. **Một số điểm chính về Raylib**

**Ngôn ngữ chính:** C, nhưng có các binding cho C++, Python, Go, Lua, Java, C#, … nên có thể sử dụng trên nhiều ngôn ngữ.

**Mục đích:** Hỗ trợ tạo các ứng dụng đồ họa 2D, 3D và game mà không cần phải học những thư viện đồ họa phức tạp như OpenGL hay DirectX từ đầu.

**Tính đơn giản:** API của raylib được thiết kế trực quan, dễ nhớ, giúp người mới học lập trình game nhanh chóng làm được các trò chơi cơ bản.

**Đa nền tảng:** Hỗ trợ Windows, Linux, macOS, và cả các nền tảng nhúng như Raspberry Pi.

**Tính năng:**

Vẽ hình 2D, 3D cơ bản (hình chữ nhật, hình tròn, hình hộp, mesh,…).

Quản lý cửa sổ, input (bàn phím, chuột, gamepad).

Xử lý âm thanh và nhạc nền

Tích hợp font chữ, hình ảnh, texture, sprite, và các hiệu ứng cơ bản.

Nói ngắn gọn: **Raylib là công cụ “nhẹ, đơn giản, dễ học” giúp lập trình viên nhanh chóng tạo ra game hoặc ứng dụng đồ họa mà không bị rối bởi những kiến thức sâu về đồ họa máy tính.**

## ****Ưu điểm của Raylib khi làm game và khi kết hợp với C++****

* + 1. **Ưu điểm của Raylib khi làm game**

**Dễ học, dễ dùng:**

API trực quan, đơn giản, giảm bớt độ phức tạp so với các thư viện đồ họa khác (OpenGL, DirectX).

Thích hợp cho người mới học lập trình game.

**Hỗ trợ 2D và 3D:**

Có thể vẽ hình 2D cơ bản, sprite, animation.

Hỗ trợ 3D cơ bản như mesh, camera, ánh sáng.

**Đa nền tảng:**

Chạy trên Windows, Linux, macOS, và cả thiết bị nhúng như Raspberry Pi.

**Quản lý input và âm thanh sẵn:**

Raylib hỗ trợ bàn phím, chuột, gamepad, và âm thanh mà không cần cài thêm thư viện khác.

**Hiệu suất tốt:**

Nhẹ, tối ưu, phù hợp với game 2D/3D nhỏ và vừa.

**Hỗ trợ cộng đồng:**

Mã nguồn mở, cộng đồng lớn, nhiều ví dụ và tutorial.

* + 1. **Khi kết hợp với C++**

**Kết hợp C++ OOP:**

Có thể tận dụng lập trình hướng đối tượng (class, inheritance, polymorphism) để quản lý game dễ hơn.

Ví dụ: tạo class Player, Enemy, GameObject để quản lý logic game gọn gàng.

**Tăng tính linh hoạt:**

C++ cho phép quản lý bộ nhớ thủ công, template, STL,… giúp tối ưu hiệu suất game.

**Mở rộng dễ dàng:**

Có thể tích hợp các thư viện C++ khác (Physics, AI, GUI…) với raylib mà không gặp trở ngại.

**Hiệu suất cao hơn:**

Raylib vốn viết bằng C, khi dùng với C++ vẫn giữ hiệu suất cao, đồng thời tận dụng các tính năng tối ưu của C++ (inline, move semantics…).

**Tạo game phức tạp hơn:**

Khi dự án lớn, sử dụng C++ với raylib giúp tổ chức code rõ ràng, dễ bảo trì hơn so với viết thuần C

# PHÂN TÍCH CHỨC NĂNG HỆ THỐNG

## Chức năng người dùng

* + 1. **Khởi tạo**

Khi vừa chạy game thì sẽ xuất hiện màn hình tiêu đề và người chơi có thế click chuột vào nút “Start Game” để bắt đầu chơi

Sau khi click vào “Start Game” thì sẽ được chuyển tới màn hình chọn chế độ chơi, ở đây có các chế độ chơi là cờ thường và cờ chớp, PvP và PvC sau đó mới bắt đầu vào ván cờ

* + 1. **Các thao tác quản lý ván cờ**

Người chơi có thể chọn các quân cờ và di chuyển quân cờ bằng cách click chuột vaof quân cờ theo đúng lượt chơi và huỷ chọn quân bằng cách click vào lại lần nữa

## Các chức năng xử lý của hệ thống

* + 1. **Sinh và kiểm tra nước đi**

Khi click chọn quân cờ thì hệ thống sẽ xét các nước đi đúng với luật di chuyển của quân đó, sau đó sẽ kiểu tra xem sau nước đó vua cùng màu với quân đó có bị chiếu không, nếu không thì sẽ đc thêm vào danh sách nước đi hợp lệ của quân cờ đó.

Mỗi khi chọn một quân cờ khác với quân cờ trước đó thì danh sách sẽ được tạo lại tử đầu.

* + 1. **Xử lý các luật đặc biệt**

Gồm các nước phong hậu và nhập thành. Khi di chuyển quân tốt của mình lên ô cuối bên địch thì hệ thống sẽ thay vào đó quân hậu và lấy quân tốt ra lưu lại ở stack trạng thái của bản cờ đề undo lại. Đối với nhập thành thì sẽ được kiểm tra vị trí vua và đường vua đi có bị chiếu hay không, kiếm tra vua và xe đã di chuyển chưa, nếu không bị chiếu và vua xe đều chưa di chuyển thì cho phép nhập thành.

* + 1. **Quản lý trạng thái bàn cờ**

Mỗi khi xong một nước đi thì hệ thống sẽ lưu lại trạng thái trước đó vào stack (quân nào di chuyển, đi từ đâu tới đâu, quân nào bị ăn, có nhập thành hay phong hậu hay không) để có thể undo lại các nước đi trước đó nếu muốn, stack này khi restart hoặc thoát khỏi ván cờ thì sẽ được dọn dẹp.

## Các chức năng AI

* + 1. **Sinh nước đi hợp lệ**
    2. **Tìm kiếm nước đi tốt nhất**

## Giao diện người dùng

Các nút thao tác trong màn hình như U(undo) – quay lại trạng thái cúa bàn cờ của nước trước đó, R(Restart) – khôi phục lại trạng thái ban đầu bàn cờ và chơi lại từ đầu, B(Back) – quay lại màn hình menu để chọn giữa 2 chế độ chơi PvP và PvC

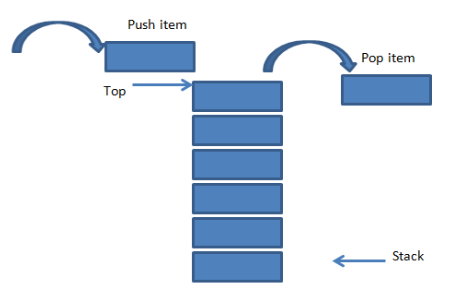
Khi click chọn quân cờ, bàn cờ sẽ tô viền những ô mà quân cờ được chọn có thể đi tới. Nếu quân vua của lượt chơi hiện tại đang bị chiếu thì sẽ được tô đỏ lên để người chơi chú ý

# THIẾT KẾ CẤU TRÚC DỮ LIỆU

## Stack

* + 1. **Khái niệm:**  
       Stack là một danh sách được bổ sung 2 thao tác: **thêm một phần tử vào cuối danh sách**, và **loại bỏ một phần tử ở cuối danh sách**. Ví trí cuối của Stack được gọi là đỉnh (**top**).

Có thể hình dung Stack như một chồng sách. Việc đặt một quyển sách lên trên cùng chính là thao tác thêm phần tử, và lấy ra quyển sách ở trên đầu là thao tác loại bỏ phần tử. Như vậy, quyển sách được đặt vào sau cùng sẽ luôn được lấy ra trước tiên. Vì tính chất này, Stack còn được gọi là danh sách **LIFO** (Last In - First Out, hay vào sau - ra trước).

Hình ảnh mô phỏng:  


Công dụng:

- Trong game cờ vua, việc sử dụng **stack** mang lại nhiều lợi ích quan trọng, đặc biệt trong quản lý nước đi và thao tác Undo/Redo. Stack hoạt động theo cơ chế **LIFO (Last In, First Out)**, nghĩa là nước đi được thực hiện gần đây nhất sẽ được lưu trên cùng và được truy xuất đầu tiên khi cần quay lại. Điều này rất phù hợp với cơ chế **Undo** trong cờ vua: khi người chơi hoặc AI muốn hủy bỏ nước đi cuối cùng, stack cho phép truy xuất nhanh nước đi vừa thực hiện mà không cần duyệt toàn bộ lịch sử.

Một số lợi ích cụ thể:

**+ Hiệu quả và đơn giản**: Stack lưu trữ các nước đi một cách tuyến tính, không cần phức tạp như danh sách liên kết hay mảng hai chiều, giúp thao tác push/pop nhanh chóng.

**+ Quản lý lịch sử di chuyển**: Giữ toàn bộ thông tin quan trọng của nước đi (vị trí quân, loại quân, trạng thái đặc biệt như phong cấp hay nhập thành) để dễ dàng phục hồi.

**+ Hỗ trợ Undo/Redo**: Việc Undo nước đi được thực hiện chỉ với thao tác pop và đọc top, không cần tính toán hay tìm kiếm phức tạp.

**+ Tiết kiệm bộ nhớ và logic rõ ràng**: Stack chỉ giữ những nước đi cần thiết, loại bỏ nước đi cũ sau khi Undo, giúp quản lý bộ nhớ đơn giản hơn.

**Vì sao nên ưu tiên dùng stack trong game cờ vua:**

+ Cơ chế LIFO hoàn toàn phù hợp với bản chất Undo/Redo trong cờ vua.

+ Giảm độ phức tạp thuật toán, giúp code gọn gàng, dễ bảo trì.

+ Cho phép thực hiện thao tác nhanh, ổn định ngay cả với nhiều nước đi được thực hiện liên tục.

Nói tóm lại, stack không chỉ **tiện lợi** mà còn **tối ưu hiệu năng**, là lựa chọn tự nhiên cho quản lý lịch sử nước đi trong game cờ vua.

* + 1. **Công dụng của các hàm trong class Stack**
       1. Stack()

Khởi tạo một stack rỗng với data\_ = nullptr,

capacity\_ = 0, count\_ = 0.

Gọi hàm reserve(4) để cấp phát bộ nhớ khởi đầu cho 4 phần tử.

**Công dụng:** Chuẩn bị stack để lưu trữ các phần tử mà không cần phải cấp phát bộ nhớ ngay lập tức mỗi khi push.

* + - 1. ~Stack()

Giải phóng bộ nhớ đã cấp phát cho data\_.

**Công dụng:** Ngăn rò rỉ bộ nhớ khi stack không còn dùng nữa.

* + - 1. Stack (const Stack& other)

Tạo bản sao của một stack khác.

Cấp phát bộ nhớ đủ cho other.capacity\_ và sao chép tất cả các phần tử.

**Công dụng:** Cho phép copy stack mà không bị trỏ chéo dữ liệu, tránh lỗi khi stack bị xóa

* + - 1. **operator=(const Stack& other) (Copy Assignment)**

Gán một stack cho stack khác.

Xóa dữ liệu cũ, cấp phát mới, copy toàn bộ phần tử.

**Công dụng:** Cho phép gán stack mà vẫn giữ tính an toàn bộ nhớ.

* + - 1. **void push (const T& value)**

Thêm một phần tử vào đỉnh stack.

Nếu stack đầy (count\_ >= capacity\_), gọi reserve để tăng gấp đôi dung lượng.

**Công dụng:** Lưu trữ một nước đi mới (Move) vào stack t rong game cờ vua.

* + - 1. **void pop()**

Xóa phần tử trên cùng của stack (không xóa thực thể, chỉ giảm count\_).

**Công dụng:** Hoàn tác nước đi cuối cùng khi người chơi undo.

* + - 1. **T& top() / const T& top() const**

Trả về phần tử trên cùng của stack.

Nếu stack rỗng, trả về một phần tử tĩnh mặc định dummy.

**Công dụng:** Lấy nước đi gần nhất để xử lý undo hoặc kiểm tra nước đi cuối.

* + - 1. **bool empty() const**

Kiểm tra stack có rỗng không (count\_ == 0).

**Công dụng:** Dùng để biết stack còn nước đi để undo hay không.

* + - 1. int size() const

Trả về số phần tử hiện có trong stack (count\_).

**Công dụng:** Biết có bao nhiêu nước đi đã thực hiện.

* + - 1. void clear()

Xóa toàn bộ stack (count\_ = 0) mà không giải phóng bộ nhớ.

**Công dụng:** Reset stack khi bắt đầu game mới.

* + - 1. void reserve(int new\_capacity)

Cấp phát thêm bộ nhớ cho stack khi cần.

Copy dữ liệu cũ sang mảng mới và xóa mảng cũ.

**Công dụng:** Tự động mở rộng stack để không bị đầy khi push nhiều nước đi.

## Vector

* + 1. **Khái niệm:**  
        Trong C++, vector là một **container động** trong thư viện STL (Standard Template Library), hoạt động giống như **mảng động (dynamic array)**.

Nó có thể **tự động thay đổi kích thước** khi thêm hoặc xóa phần tử.

Nếu mảng truyền thống (array) có kích thước cố định ngay khi khai báo, thì **vector có khả năng tự mở rộng hoặc thu nhỏ** tùy theo số lượng phần tử hiện có.

Vector được dùng rất phổ biến trong lập trình vì:

**\*Lưu trữ danh sách dữ liệu linh hoạt**

- Dùng vector để lưu:

+ danh sách số nguyên

+ danh sách ký tự

+ danh sách đối tượng (class, struct)

+ danh sách chuỗi, …

### **\* Thêm, xoá, truy cập phần tử thuận tiện**

- Vector hỗ trợ nhiều thao tác quan trọng:

+ push\_back() – thêm cuối nhanh

+ pop\_back() – xóa cuối nhanh

+ size() – lấy số lượng phần tử

+ clear() – xoá tất cả

+ truy cập chỉ số như mảng: v[I]

### **\* Làm việc tốt với vòng lặp**

- Kết hợp dễ dàng với:

+ for,

+ range-based for,

+ iterator (begin(), end())

### **\* Tương thích tốt với STL algorithms**

- Vector hoạt động tốt với:

+ sort()

+ reverse()

+ count()

+ binary\_search()

+ find()

* + 1. **Công dụng của Vector trong game**

Trong dự án cờ vua của nhóm, Vector được dùng để lưu những cấu trúc dữ liệu có kích thước động và cần truy cập ngẫu nhiên (index) nhanh:

Vector<Move> / Vector<MoveHint>: lưu danh sách **nước đi hợp lệ** do AI/Player sinh ra và danh sách gợi ý để vẽ highlight.

Vector dùng trong AI để chứa tất cả nước đi khả dĩ khi duyệt Minimax.

Khi cần sort, traverse, copy danh sách nước đi (ví dụ trước khi sắp xếp move để alpha-beta hiệu quả) — Vector cung cấp con trỏ liên tiếp nên rất phù hợp cho std::sort (begin/end trả về T\*).

Dùng mỗi khi cần bộ nhớ liên tiếp (contiguous) để tận dụng locality của CPU — giúp duyệt / so sánh nhanh trong vòng lặp lớn (AI).

Tóm lại: Vector dùng để **lưu trữ tạm thời danh sách có kích thước thay đổi** như tập nước đi, gợi ý, hoặc các danh sách phục vụ thuật toán AI.

# Tại sao nên dùng Vector (thay vì mảng thường hoặc linked list) trong cờ vua

**Truy cập O(1)** theo chỉ số — thuận tiện khi lặp qua các nước đi (for i = 0..n-1).

**Bộ nhớ liên tiếp (contiguous)** → cache-friendly → quan trọng cho hiệu năng ở những vòng lặp lồng nhau (AI).

**Amortized O(1) push\_back**: thêm nước đi nhanh, khi cần mở rộng thì đoạn code nhân đôi capacity (trong bạn: reserve doubling logic).

**Tương thích với STL**: con trỏ T\* từ begin()/end() cho phép dùng std::sort, std::for\_each, v.v.

**An toàn hơn raw array**: tự cấp phát và giải phóng, tránh phải quản lý new[]/delete[] thủ công mọi nơi.

**Hỗ trợ move semantics**: bản của bạn hỗ trợ move ctor/assign nên tránh copy nặng khi trả về Vector tạm.

Trong cờ vua, AI cần tạo & huỷ nhiều danh sách nước đi rất nhanh — Vector tối ưu cho workflow này.

* + 1. **Các chức năng chính của Vector**

**Quản lý bộ nhớ:**

reallocate(size\_t newCap): cấp phát vùng nhớ mới lớn hơn, di chuyển các phần tử sang vùng mới, giải phóng vùng cũ.

reserve(size\_t newCap): đảm bảo dung lượng tối thiểu, tránh cấp phát nhiều lần khi thêm phần tử.

**Khởi tạo và giải phóng:**

Vector(): khởi tạo vector rỗng.

~Vector(): giải phóng toàn bộ phần tử và bộ nhớ.

**Copy & Move:**

Vector(const Vector& other),

operator=(const Vector& other): tạo và gán bản sao vector khác.

Vector(Vector&& other),

operator=(Vector&& other): di chuyển dữ liệu từ vector khác, tối ưu hiệu năng.

**Thao tác thêm/xóa phần tử:**

push\_back(const T& value), push\_back(T&& value): thêm phần tử vào cuối vector.

pop\_back(): xóa phần tử cuối.

clear(): xóa toàn bộ phần tử nhưng giữ capacity để dùng lại.

**Truy cập phần tử và thông tin trạng thái:**

operator[](size\_t index): truy cập phần tử theo chỉ số.

size(): số phần tử hiện có.

capacity(): dung lượng đã cấp phát.

empty(): kiểm tra vector rỗng hay không.

**Iterator (con trỏ):**

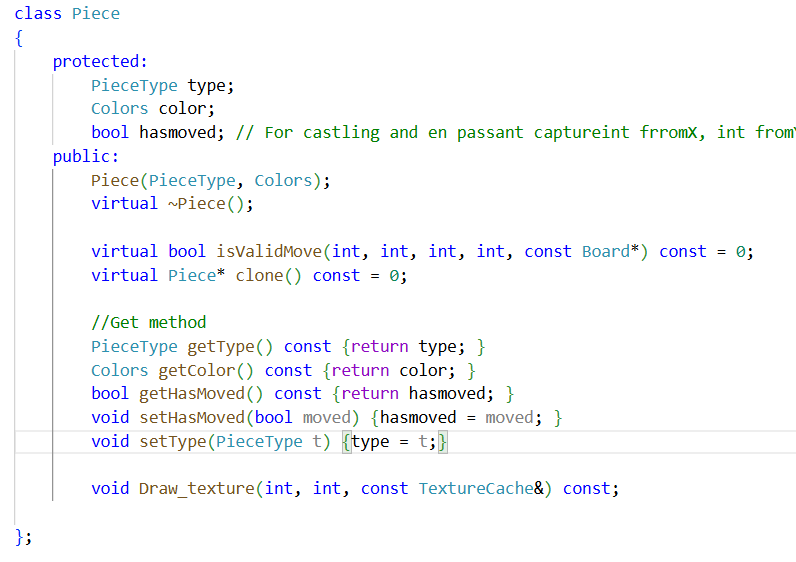
begin() / end(): trả về con trỏ đầu và sau phần tử cuối, phiên bản const cũng có để dùng với vector không chỉnh sửa.

**Ý nghĩa trong game cờ vua:**

Vector được dùng để lưu trữ danh sách nước đi hợp lệ, lịch sử di chuyển, các gợi ý nước đi cho người chơi hoặc AI. Sử dụng vector giúp code linh hoạt, dễ quản lý, tránh lỗi tràn mảng, và tối ưu hiệu năng nhờ khả năng mở rộng động và hỗ trợ move semantics.

# XÂY DỰNG CHI TIẾT HỆ THỐNG

## Trừu tượng hoá quân cờ



Các đối tượng quân cờ đều sẽ được quản lý qua con trỏ của lớp cơ sở ảo Piece. Lớp này sẽ chứa các biến thể hiện cho loại quân cờ, màu quân cờ, và trạng thái di chuyển của quân cờ. Việc trừu tượng hoá lớp cơ sở Piece sẽ giúp cho việc truy cập vào từng quân cờ sẽ dễ dàng hơn. Vì mọi quân cờ đều được lưu địa chỉ tại con trỏ kiểu Piece nên dễ đàng để thiết kế mã nguồn và sửa đổi khi cần.

Hình ảnh của quân cờ sẽ được lưu tại TextureCache (1 class dùng để lưu hình ảnh của tất cả quân cờ để có thể tái sử dụng nhiều lần) và được vẽ lên màn hinh trò chơi thông qua Draw\_texture().

## Xử lý input từ chuột và di chuyển quân cờ

Để xử lý được thao tác của người chơi thì hệ thống sẽ dùng hai biến xVal, yVal để lưu lại vị trí click chuột và kiểm tra vị trí có quân hay không và kiểm tra xem có chọn đúng theo lượt không, nếu thoả mãn thì bắt đầu tạo ra danh sách các nước đi hợp lệ của quân cờ đó. Nếu lần click chuột tiếp theo là ô hợp lệ thì sẽ bắt đầu thực hiện di chuyển quân cờ tới vị trí đó.Nếu click chọn quân cùng màu với quân đã chọn trước đó thì danh sách các nước đi hợp lệ sẽ được xây dựng lại từ đầu theo quân vừa được chọn và tiếp tục xử lý lại như trên.

Danh sách nước đi hợp lệ được khởi tạo bằng cách kiểm tra từ vị trí quân cờ được chọn tới tất cả vị trí trên bàn cờ, nếu ô nào hợp lệ thì sẽ được thêm vào danh sách, một ô hợp lệ phải đồng thời thoả mãn hai điều kiện: ô đó là nước đi đúng luật của quân cở được chọn hoặc sau khi đi tới ô đó thì Vua của bên mình không bị chiếu, để làm được thì sẽ cho giả lập nước đi rồi kiếm tra vị trí vua có bị quân nào của đối thủ chiếu hay không.



*ảnh*

**Các thao tác xử lý click chuột đều được xử lý thông qua hàm HandleInput():

*ảnh*

Các luật di chuyển đặc biệt sẽ được xử lý ngay tại hàm Makemove:

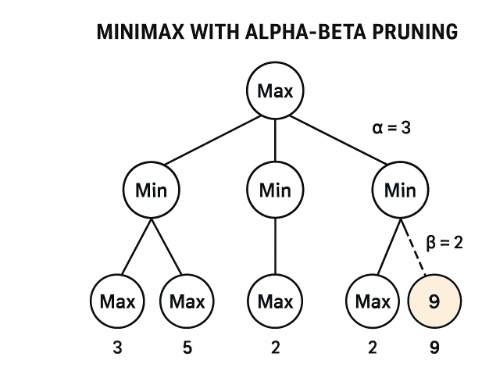
ảnh

*ảnh*

*ảnh*

## Thuật toán sinh nước đi khi chơi với máy

Khi chơi ở chế độ PvC, để máy có di chuyển quân cờ sau lượt của người chơi thì nhóm đã tự sinh ra các nước đi hợp lệ sau đó tìm kiếm nước đi có giá trị cao dựa vào thuật toán tìm kiếm, ở đề tài này nhóm đã chọn minimax làm thuật toán để tìm kiếm nước đi.



*ảnh*

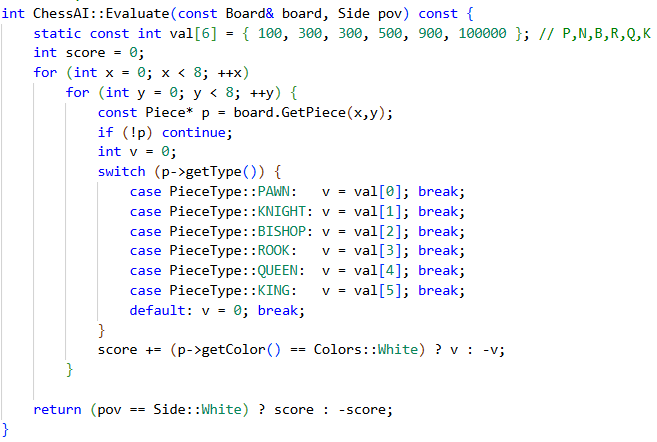
**Minimax** là một thuật toán dùng trong các trò chơi hai người chơi, có tổng bằng 0, như cờ vua. Ý tưởng chính:

**MAX** (người chơi AI) cố gắng **tối đa hóa** điểm số.

**MIN** (đối thủ) cố gắng **tối thiểu hóa** điểm số.

Thuật toán tạo một **cây trò chơi**, đánh giá tất cả các nước đi có thể để chọn nước đi tốt nhất dựa trên việc giả lập phản ứng của đối thủ.

Khi chạy thì thuật toán sẽ duyệt DFS tới độ sâu d được chỉ định và bắt đầu tính giá trị của bàn cờ tại đó, việc tính giá trị sẽ được hàm Evaluate() xử lý



*ảnh*

Sau khi đã tính xong giá trị thì sẽ được chuyển lên các nút trên để so sánh giá trị giữa các nút con để chọn kết quả tổt hơn và cuối cùng là chọn kết quả tốt nhất tìm được cho nút gốc cũng là nước đi tiếp theo của máy.

Thuật toán sẽ duyệt tất cả các nước đi hợp lệ theo từng độ sâu từ gốc tới độ sâu d, nên độ phức tạp của Minimax sẽ là O(bd)

Trong đó : b – trung bình sổ nước đi hợp lệ ở mỗi lượt

d – độ sâu tìm kiếm

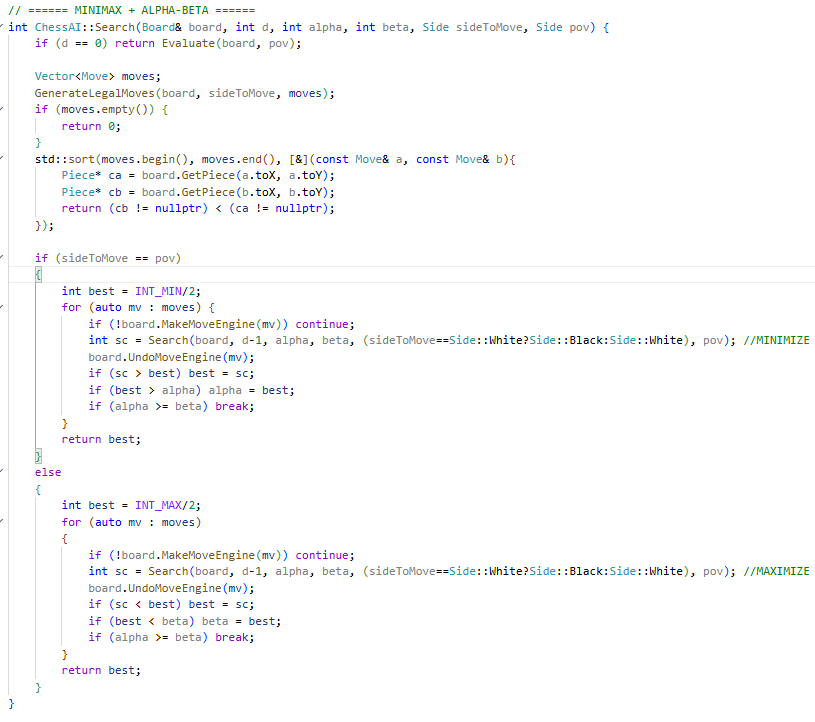
Như vậy ta có thể thấy thuật toán sẽ càng nhanh hơn khi trên bàn cờ có càng ít quân và thuật toán sẽ càng lâu nếu d lớn. Để giảm bớt số trường hợp phải tính để thời gian tính toán của máy nhanh hơn thì ta có thể áp dụng thêm thuật toán cắt tỉa các nhành không cần thiết – Alphabeta pruning:

**Cắt tỉa Alpha-Beta (Alpha-Beta Pruning)** là một cải tiến của Minimax:

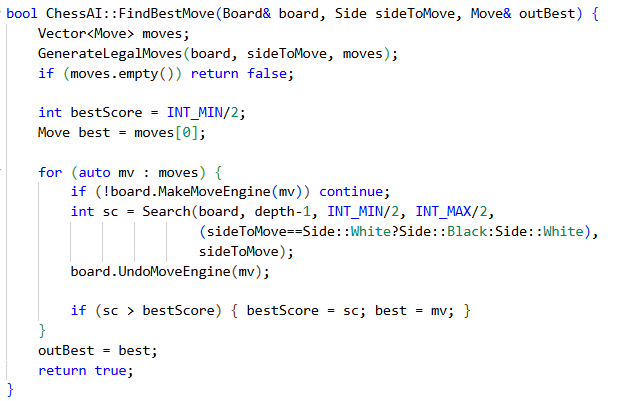
**Alpha (α):** giá trị tốt nhất mà MAX có thể đảm bảo.

**Beta (β):** giá trị tốt nhất mà MIN có thể đảm bảo.

Khi giá trị tạm thời vượt ra ngoài alpha hoặc beta, nhánh đó **không cần xét tiếp**, giúp giảm số nút cần duyệt, tăng tốc thuật toán.

Khi có thêm cắt tỉa Alphabeta thì trường hợp tệ nhất của thuật toán vẫn là O(bd) nhưng trường hợp tốt nhất sẽ gần như là O(bd/2) – giảm một nửa độ sâu.

*ảnh*

Hàm FindBestMove() sẽ có nhiệm vụ gọi tới hàm Search() để tạo ra danh sách các nước có thể đi và chọn ra nước đi tốt nhất vừa tìm được

*ảnh*

# PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG VÀ TRIỂN KHAI HỆ THỐNG

## Cấu trúc hệ thống hướng đối tượng (Ngọc)

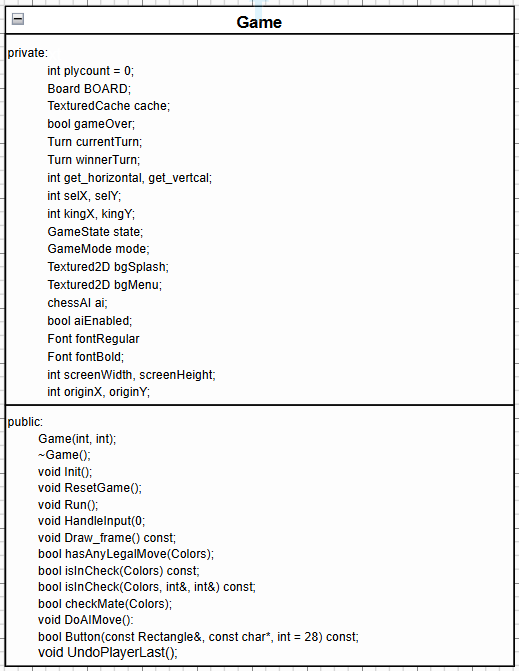
* + 1. **Các lớp và đối tượng trong hệ thống trò chơi**

Class Game:

Đây là class chính, đối tượng thuộc Game sẽ đảm nhận vai trò khởi tạo trò chơi cũng như quản lý mọi thử của game, đây là trung tâm điều hàn và là “hệ thần kinh” của cả hệ thống, nó phân tính input từ người chơi và bắt đầu gọi tới các hàm khác nhau để game hoạt động

.

Trong Game sẽ quán lý việc lựa chọn chế độ chơi, sinh nước đi của máy khi chơi PvC, kiểm tra check và checkmate, render màn hình, gọi các hàm vẽ bàn cờ quân cờ background, Game có các biến lưu các giá trị như kích thước màn hình, toạ độ trên bàn cờ.

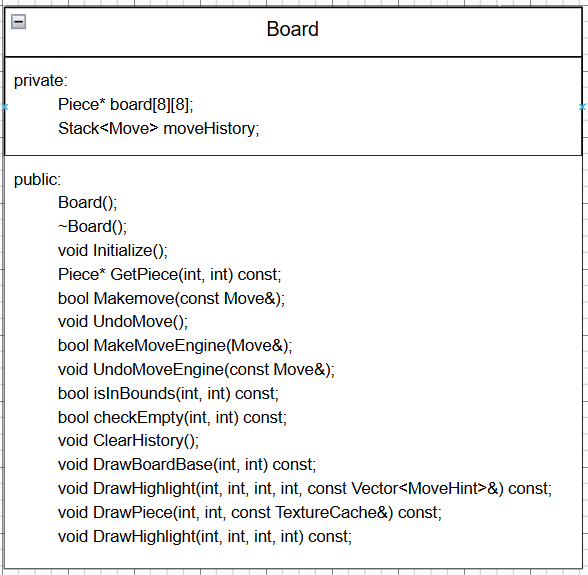


*ảnh*

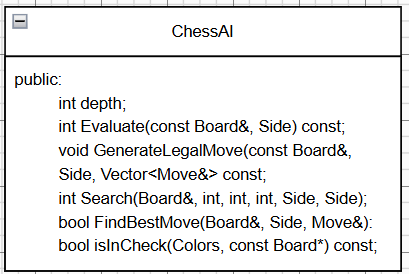
Class Board:

Đối tượng thuộc lớp Board sẽ là đối tượng quán lý toàn bộ bàn cờ dựa trên mảng con trỏ Piece với kích thước 8x8, đây sẽ là nơi trò chơi thực sự được xử lý.

Trong Board có các hàm để trò chơi vận hành, Board chứa thêm 1 thuộc tính moveHistory, đây là một cấu trúc dữ liệu ngăn xếp (stack) có tác dụng dùng để lưu các nước đi trước đó của bàn cờ để phục vụ việc undo. Đối tượng Board sẽ là một thuôc tính trong Game, Game sẽ điều hành trò chơi và gọi tới các phương thức của Board.



*ảnh*

Class ChessAI:

*ảnh*

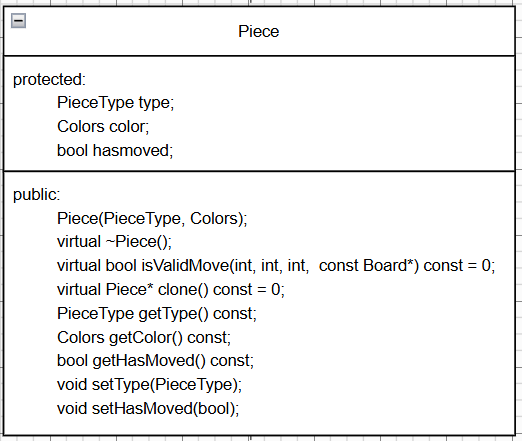
ChessAI là lớp chứa các hàm quan trọng để máy có thể tính toán được nước đi cho lượt của mình. ChessAI bao gồm các hàm: tính toán giá trị bàn cờ, sinh nước đi hợp lệ, đệ quy minimax để thực hiện tìm kiếm nước đi và chọn ra lựa chọn tốt nhất tìm được.

ChessAI cũng là một đối tượng thuộc Game và sẽ được gọi tới khi người chơi chọn chế độ chơi với máy

Class Piece:

Piece là lớp cơ sở cho toàn bộ các quân cờ, lớp này được trừu tượng hoá để thể hiện rõ Piece chỉ là lớp cơ sở với các thuộc tính cơ bản của 1 quân cờ như loại quân cờ, màu, trạng thái di chuyển và các hàm get set cơ bản và phương thức thuần ảo là luật di chuyển của quân cờ và sao chép quân cờ để các lớp quân cờ cụ thể kế thứa từ Piece phải tự đa hình lại các phương thức đó để có thể sử dụng.

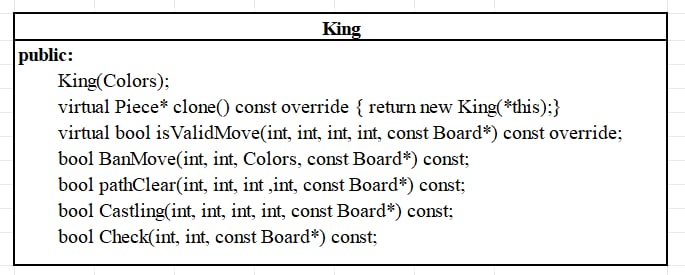
Piece cũng là lớp đảm nhiệm việc vẽ quân cờ từ asset, Board chỉ có thể gọi tới Piece để vẽ được quân cờ.



*ảnh*

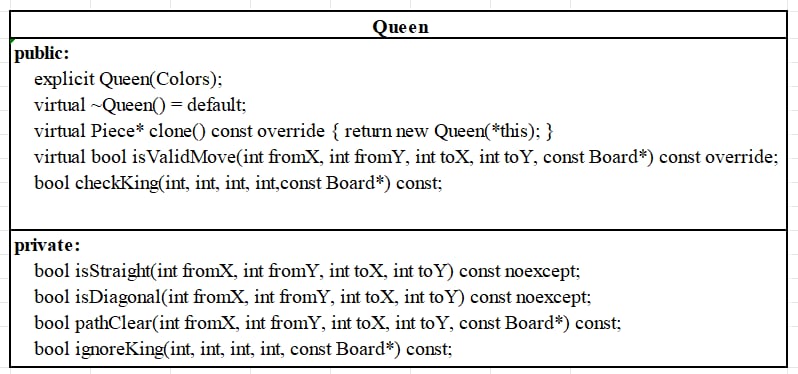
Những class được kế thừa từ Piece ngoài việc đa hình lại hàm isValidMove() theo luật của mình và hàm clone thì còn có các hàm phục các luâtj lệ và chức năng riêng của mình

* + - 1. Class King

King kế thứa Piece và đa hình lại phương thức của Piece để có thể sử dụng và các hàm để thực hiện luật đặc biệt, trong King sẽ có các hàm giúp quân vua tránh đi vào những ô khiến bản thân bị chiếu và kiểm tra các nước đủ điều kiện để thực hiện nhập thành

*ảnh*

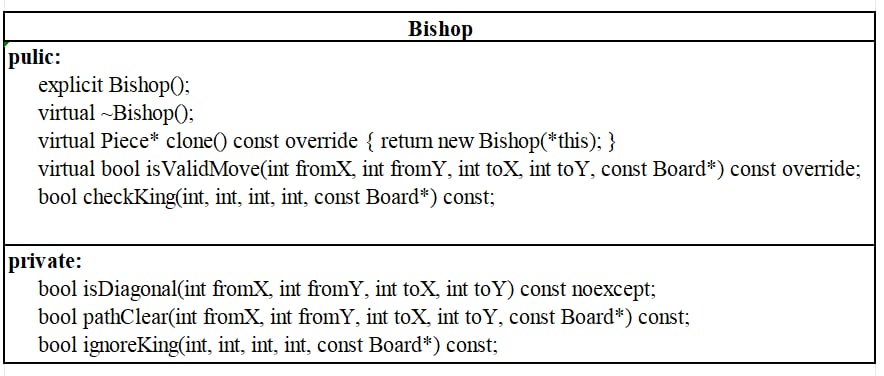
* + - 1. Class Queen

Class Queen chỉ chứa các hàm phục vụ cho luật di chuyển của bản thân vì quân hậu là một quân cờ có nhiều hướng đi và có thể đi rát xa nhưng không có luật đặc biệt.

*ảnh*

* + - 1. Class Bishop

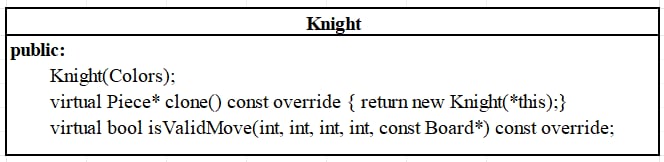
Bishop là lớp dẫn xuất từ lớp Piece, đại diện cho quân tượng và được cài đặt luật di chuyển theo đường chéo



*ảnh*

* + - 1. Class Knight

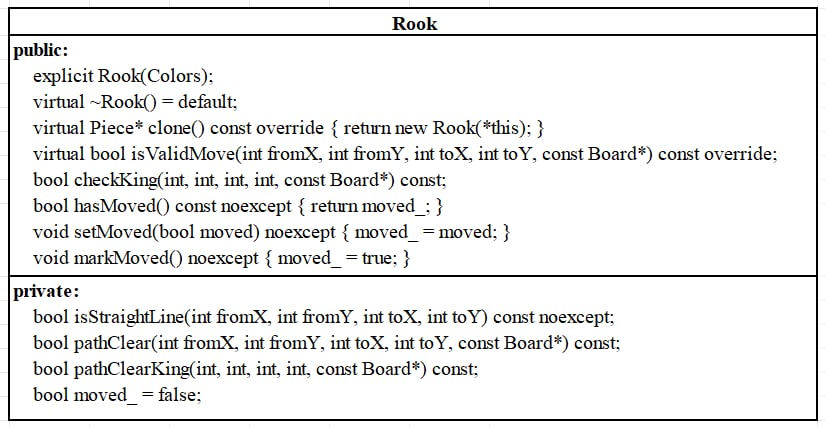
Knight là lớp dẫn xuất của Piece, đại diện cho quân tượng và được cài đặt luật di chuyển theo hình chữ L, quân ngựa di chuyển theo kiểu “nhảy” nên không cần kiểm tra dường đi trống hay không



*ảnh*

* + - 1. Class Rook

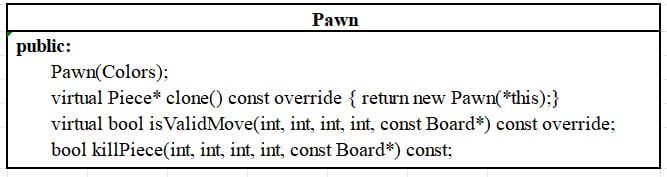
Rook là lớp dẫn xuất của Piece, đại diện cho quân xe, lớp được cài đặt với luật di chuyển theo hướng dọc và ngang trên bàn cờ



*ảnh*

* + - 1. Class Pawn

Pawn là lớp dẫn xuất của Piece, đại diện cho quân tốt , được đa hình lại để cài đặt theo luật di chuyển và ăn quân của quân tốt

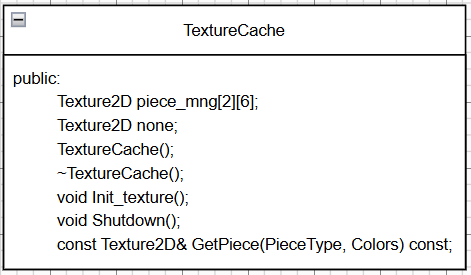


*ảnh*

Class TextureCache

TextureCache được xây dựng để khi bắt đầu chạy chương trình thì texture của các quân cờ được tải lên GPU sẵn, khi nào cần thì chỉ cần gọi tới, việc xây dựng TextureCach để tải texture lên GPU thay vì mỗi quân tự tải lên, điều này giúp tránh được việc xoá nhầm ảnh của quân cờ sau khi quân cờ đó bị ăn mất nhưng quân cờ khác cùng loại vần còn.

Khi cần vẽ thì TextureCache chỉ cần gửi đi đúng ảnh theo loại quân cờ cần vễ là được. Toàn bộ dữ liệu của TextureCache chỉ thực sự được giải phóng sau khi đóng cửa sổ và thoát game.

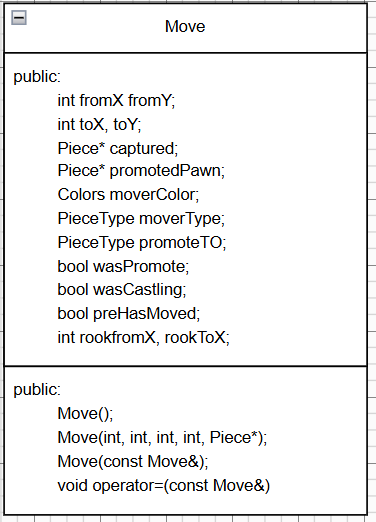


*ảnh*

Class Move

Move được cài đặt để lưu giữ trạng thái của bàn cờ tại một nước đi nào đó của ván đấu, Move lưu các giá trị fromX/fromY và toX/toY là vị trí của quân cờ di chuyển trong lượt đó cùng con trỏ giữ quân đã bị ăn (nếu có) để có thể undo chính xác khi cần, trong Move chưa các cờ dùng để chương trình nhận biết có nhập thành hay phong hậu hay không để undo được chính xác đối với các luật đặc biệt này.

Move còn được sử dụng trong việc giả lập nước đi, một tính năng quan trọng để thuật toán AI có thể sinh nươc đi mà không làm “bẩn” bàn cờ cùng như hệ thống có thể sinh ra danh sách các nước đi hợp lệ và gợi ý trên màn hình trò chơi cho người chơi.



*ảnh*

* + 1. **Quan hệ giữa các đối tượng**

Game sẽ là đối tượng trung tâm, điều hành mọi hoạt đồng của game, Game sẽ sở hữu các thuộc tính là đối tượng của Board, TextureCache, ChessAI để có thể quản lý bàn cờ, sử dụng ảnh các quân cờ, gọi tới các thuật toán tìm kiếm sinh nước đi cho máy khi chơi PvC

Các class quân cờ cụ thể sẽ là lớp dẫn xuất thừa kế từ class Piece

ảnh sơ đồ

* + 1. **Lợi ích khi sử dụng hướng đối tượng**

## Trong quá trình phát triển trò chơi cờ vua, việc áp dụng lập trình hướng đối tượng (Object-Oriented Programming – OOP) mang lại nhiều lợi ích quan trọng. Các thành phần trong cờ vua như quân cờ, bàn cờ hay nước đi đều có đặc điểm riêng, hành vi riêng và có mối quan hệ chặt chẽ với nhau. OOP cho phép mô hình hóa những đối tượng này một cách tự nhiên, rõ ràng và dễ quản lý, nhờ đó giúp hệ thống trở nên linh hoạt, mở rộng và dễ bảo trì hơn trong suốt vòng đời phát triển trò chơi.

\*Các lợi ích của hướng đối tượng đối với game cờ vua:

- Trừu tượng hoá (Abstraction):

OOP cho phép mô hình hóa các thành phần phức tạp như “quân cờ”, “bàn cờ”, “nước đi” thành các đối tượng rõ ràng với thuộc tính và hành vi riêng.

Trong trò chơi của nhóm: Piece là lớp cơ sở được trừu tượng hoá, trong Piece chỉ lưu loại quân cờ, màu cờ, trạng thái di chuyển và các hàm get set cơ bản còn các lớp dẫn xuất kế thừa từ Piece như King, Queen,v.v.. sẽ phải đa hình lại để cài đặt luật đi riêng của mình.

-Kế thừa (Inheritance):

Vì mỗi quân cờ trong trò chơi đề có chung các biến đăc điểm (loại quân cờ, màu, trạng thái di chuyển) nên việc kế thứa sẽ tái sử dụng những đoạn mã nguồn liên quan tới các thuộc tính này trở nên dễ dàng hơn, giúp mã nguồn gọn và đỡ phức tạp hơn, giảm trùng lập những đoạn mã nguồn có sẵn và có thế thêm các loại quân cờ mới một cách dễ dàng/

-Đa hình (Polymorphism):

Các quân cơ đều là Piece nhưng mỗi quân lại có những luật di chuyển riêng biệt nên việt đa hình lại hàm isValidMove() trong Piece để cài đặt riêng các luật di chuyển theo từng quân cờ sẽ khiến việc xử lý các nước đi của các quân cờ trở nên thống nhất, dễ để gọi trong các thuật toán tìm kiếm của máy và sinh nước đi, tránh được việc phải tạo ra nhiều câu lệnh điều kiện if- else khiến mã nguồn trở nên phức tạp, khó đọc.

-Tính đóng gói (Encapsulation):

Vì mỗi đối tượng và mỗi lớp sẽ quản lý dữ liệu của riêng nó nên sẽ tránh được việc dữ liệu bi thay đổi không mong muốn, ngăn không cho các đoạn lệnh bên ngoài tác động trực tiếp và dữ liệu và logic bên trong giúp chương trình hạn chế được các lỗi tiềm ẩn xảy ra.

-Dễ bảo trì và debug

Mã nguồn được tỏ chức theo hướng đối tượng nên việc tìm lỗi sẽ diễn ra nhanh chóng hơn và sửa lỗi cũng đơn giản hơn vì mỗi lớp sẽ đảm nhận các vai trò riêng nên sai ở đâu thì chỉ cần sửa ở đó là được, hạn chế được những lỗi dây chuyền phức tạp.

Nhờ các đặc tính cốt lõi như trừu tượng hóa, kế thừa, đa hình và đóng gói, lập trình hướng đối tượng đặc biệt phù hợp để phát triển trò chơi cờ vua. OOP giúp mã nguồn rõ ràng, dễ bảo trì, dễ mở rộng và đảm bảo tính ổn định của hệ thống khi bổ sung thêm nhiều tính năng nâng cao như trí tuệ nhân tạo hoặc quản lý lịch sử ván cờ.

## Kết quả

### Giao diện chính của chương trình

### Kết quả thực thi của chương trình

* + 1. **Mô tả kết quả thực hiện chương trình.**

### Nhận xét

Nhìn chung thì trò chơi đã vận hành ổn định, không xuất hiện lỗi nghiêm trọng khiến trải nghiệm của người dùng bị gián đoạn hay lỗi logic game, máy thực hiền nước đi khá tốt, thuật toán Minimax và AlphaBeta-prunning đã hoạt động tốt, tốc độ phản hồi ổn đinh không quá lâu. Nhưng vì thiết kế game theo hướng đối tượng nên khi sinh nước đi ở lượt người chơi và lượt của máy thì đều sẽ gọi liên kết động tới các hàm kiểm tra luật di chuyển của các quân cờ. điều này sẽ khiến chương trình xử lý nhiều hơn so với việc gọi tới các hàm tĩnh, điều này thể hiện rõ khi chơi với máy, nếu độ sâu của cây trò chơi càng tăng thì số hàm được gọi tới càng nhiều sẽ gây chậm trong quá trình quyết định của máy, đem lại cho người chơi trải nghiêmj không mấy thoải mái nên nhóm thường để độ sâu là 5 để cân bằng giữa thời gian phản hồi của máy và độ khó của trò chơi.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Cờ Vua là một đề tài rất thú vị và giúp nhóm có thể làm quen với xây dựng một hệ thống, ứng dụng Lập trình Hướng đối tượng kết hợp với Cấu trúc dữ liệu vào trong đề tài giúp nhóm có góc nhìn rõ hơn về những kiến thức này cũng như biết cách vận dụng kiến thức đã được học vào thực tiễn. Đề tài này giúp nhóm củng cố kiến thức về cách quản lý bộ nhớ của hệ thống và đặc biệt là làm quen với thư viện đồ hoạ đề làm giao diện đồ hoạ từ đó tiếp cận với người dùng. Trong quá trình xây dựng và phát triển đề tài thì nhóm cũng gặp các vấn đề khá phổ biến liên quan đến lỗi hệ thống, lỗi truy cập bộ nhớ, lỗi logic game,… những lỗi mà chỉ xuất hiện khi chương trình thực sự được vận hành, việc khắc phục những lối này đã giúp nâng cao việc tư duy giải quyết vấn đề một cách logic, có thêm kinh nghiệm về việc tối ưu mã nguồn xây dựng mã nguồn sạch hơn để trách các lỗi tiềm ẩn khó phát hiện ngay cả khi chương trình hoạt động

## Hướng phát triển

Nhóm sẽ tiếp tục phát triển game cờ vua, thêm các tính năng và tiện ích để nâng cao trải nghiệm người dùng, thêm tính năng tăng giảm độ khó của máy, thêm các chế độ chơi đặc biệt tạo sự đa dạng bên cạnh đó nhóm sẽ tiếp tục tối ưu mã nguồn để game tránh được những nguy cơ tiềm ẩn làm văng game, các hàm isValidMove() của các quân cờ có thể sẽ được xây dựng thành các hàm tĩnh bên ngoài và được gọi thông qua khối câu lệnh switch case để giảm thời gian tính toán của thuật toán tìm kiếm vì khi máy sinh nước đi sẽ phải gọi tới các hàm kiểm tra này rất nhiều lần nên việc gọi hàm thông qua liên kết động ở runtime sẽ rất tốn thời gian đặc biệt là khi độ sâu tìm kiếm từ 6 trờ đi

TÀI LIỆU THAM KHẢO

**[1]** Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản

**[2]** Tên chủ sở hữu, Tên bài viết, url, ngày truy cập