**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

Icon

Description automatically generated with low confidence

**ĐỒ ÁN NGÔN NGỮ LẬP TRÌNH PYTHON**

**ĐỀ TÀI: Xây dựng trò chơi cờ vua**

Giảng viên hướng dẫn: **LÊ TẤN LONG**

Nhóm sinh viên:

**PHẠM ĐÌNH DUY THÁI – 3123410330**

**LÊ VĂN NHẤT – 3123410244**

**MAI THÀNH TRUNG – 3123410395**

**HỒ MINH TIẾN – 3123410370**

**TP.HCM, tháng 4 năm 2025**

**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN NGÔN NGỮ LẬP TRÌNH PYTHON**

**ĐỀ TÀI: Xây dựng trò chơi cờ vua**

**TP.HCM, tháng 4 năm 2025**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**........................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**.......................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**........................................................................................................................................................**

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến các thầy cô trong trường Đại học Sài Gòn, đặc biệt là các thầy cô thuộc Khoa Công nghệ Thông tin đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi để em có thể hoàn thành tốt đồ án môn học lần này.

Em xin bày tỏ lòng biết ơn sâu sắc đến thầy Lê Tấn Long – người đã trực tiếp hướng dẫn, định hướng và luôn nhiệt tình hỗ trợ em trong suốt quá trình thực hiện đồ án. Sự tận tâm, kiến thức sâu rộng và những chỉ dẫn của thầy đã giúp em hoàn thiện sản phẩm, nâng cao hiểu biết và rèn luyện kỹ năng lập trình.

Trong quá trình thực hiện đồ án, chúng em đã cố gắng hết sức nhưng chắc chắn không thể tránh khỏi những thiếu sót trong quá trình thực hiện đồ án. Em rất mong nhận được sự góp ý chân thành từ thầy cô để có thể tiếp thu, trau dồi thêm kiến thức và hoàn thiện hơn trong các đồ án, bài báo cáo sau này.

Em xin chân thành cảm ơn thầy cô!

TP.HCM, tháng 4 năm 2025

Nhóm sinh viên thực hiện

Lê Văn Nhất

Phạm Đình Duy Thái

Hồ Minh Tiến

Mai Thành Trung

**MỤC LỤC**

[**PHẦN 1: MỞ ĐẦU 1**](#_Toc196330558)

[**1. Giới thiệu đề tài 1**](#_Toc196330559)

[**2. Lý do chọn đề tài 1**](#_Toc196330560)

[**3. Mục đích và mục tiêu dồ án 2**](#_Toc196330561)

[**3.1. Mục đích 2**](#_Toc196330562)

[**3.2. Mục tiêu 2**](#_Toc196330563)

[**4. Yêu cầu đồ án 3**](#_Toc196330564)

[**4.1. Kế hoạch thực hiện và phân công công việc 3**](#_Toc196330565)

[**4.2. Yêu cầu chức năng 4**](#_Toc196330566)

[**PHẦN 2: GIỚI THIỆU 5**](#_Toc196330567)

[**1. Giới thiệu về ngôn ngữ lập trình Python 5**](#_Toc196330568)

[**2. Giới thiệu về thư viện Pygame 6**](#_Toc196330569)

[**2.1. Giới thiệu sơ bộ về thư viện Pygame 6**](#_Toc196330570)

[**2.2. Một số module trong thư viện Pygame 6**](#_Toc196330571)

[**3. Giới thiệu về thư viện Tkinter 8**](#_Toc196330572)

[**3.1. Giới thiệu sơ bộ về thư viện Tkinter 8**](#_Toc196330573)

[**3.2. Một số widget trong thư viện Tkinter 8**](#_Toc196330574)

[**3.3. Một số phương thức điều khiển bố cục giao diên trong thư viện Tkinter 9**](#_Toc196330575)

[**4. Giới thiệu về trò chơi cờ vua 9**](#_Toc196330576)

[**4.1. Lịch sử hình thành và phát triển trò chơi cờ vua 9**](#_Toc196330577)

[**4.2. Luật chơi cờ vua 10**](#_Toc196330578)

[**PHẦN 3: PHÁT TRIỂN TRÒ CHƠI CỜ VUA 12**](#_Toc196330579)

[**1. Phân tích trò chơi 12**](#_Toc196330580)

[**2. Thiết kế màn hình trò chơi 12**](#_Toc196330581)

[**3. Giao diện trò chơi 15**](#_Toc196330582)

[**4. Thực hiện viết mã Python 16**](#_Toc196330583)

[**4.1. Tổ chức cây thư mục 16**](#_Toc196330584)

[**4.2. Viết mã xử lí 17**](#_Toc196330585)

[**a. Quản lí điều hành trò chơi (main.py) 17**](#_Toc196330586)

[**b. Quản lí giao diện chọn chế độ chơi (menu.py) 24**](#_Toc196330587)

[**c. Quản lí ô cờ (square.py) 26**](#_Toc196330588)

[**d. Quản lí các lớp quân cờ (piece.py) 28**](#_Toc196330589)

[**e. Quản lí bàn cờ (board.py) 30**](#_Toc196330590)

[**f. Quản lí logic tổng thể của trò chơi (game.py) 46**](#_Toc196330591)

[**g. Quản lí nước đi từ vị trí ban đầu đến vị trí đích trên bàn cờ (move.py) 51**](#_Toc196330592)

[**h. Quản lí AI đánh cờ với người chơi (ai.py) 52**](#_Toc196330593)

[**i. Quản lí chọn quân cờ và vị trí hiện tại (selcetor.py) 65**](#_Toc196330594)

[**j. Quản lý giao diện, âm thanh và font chữ (config.py) 66**](#_Toc196330595)

[**k. Quản lí thông số cố định của game (const.py) 67**](#_Toc196330596)

[**PHẦN 4: TỔNG KẾT 69**](#_Toc196330597)

[**1. Ưu và nhược điểm 69**](#_Toc196330598)

[**1.1. Ưu điểm 69**](#_Toc196330599)

[**1.2. Nhược điểm 69**](#_Toc196330600)

[**2. Hướng cải tiến phát triển trong tương lai 69**](#_Toc196330601)

[**TÀI LIỆU THAM KHẢO 71**](#_Toc196330602)

# PHẦN 1: MỞ ĐẦU

## Giới thiệu đề tài

Đề tài: Xây dựng chương trình đánh cờ vua với nhiều với thư viện Pygame bằng ngôn ngữ lập trình Python

## Lý do chọn đề tài

Cờ vua là một trò chơi trí tuệ có sức hút mạnh mẽ trên toàn thế giới, không chỉ bởi tính giải trí mà còn bởi khả năng rèn luyện tư duy chiến lược và kỹ năng phân tích. Trò chơi đòi hỏi người chơi phải có sự tập trung cao độ, khả năng tính toán cẩn thận và xây dựng chiến thuật hợp lý để giành chiến thắng. Đây là sự kết hợp hoàn hảo giữa khả năng suy luận và tư duy sáng tạo, giúp người chơi phát triển trí tuệ lẫn kỹ năng giải quyết vấn đề.

Python hiện nay là một trong những ngôn ngữ lập trình phổ biến và có sức ảnh hưởng nhất trên thế giới. Kể từ khi được tạo ra bởi Guido van Rossum, Python đã bùng nổ và phát triển mạnh mẽ để trở thành ngôn ngữ ưa thích trong nhiều lĩnh vực như Phát triển web, Khoa học dữ liệu, Trí tuệ nhân tạo (AI) và Machine Learning, Phát triển game (Pygame) , Tự dộng hóa. Python được ưa chuộng vì cú pháp đơn giản, dễ đọc, dễ học. Bên cạnh đó, cộng đồng Python rất lớn mạnh tạo điều kiện cho việc tìm tài liệu, trao đổi kinh nghệm và nhận được sự hỗ trợ toàn cầu

Việc xây dựng game cờ vua bằng Python là cơ hội tuyệt vời để chúng em kết hợp giữa tư duy lập trình và chiến thuật trong trò chơi. Dự án này giúp người lập trình áp dụng các kiến thức về lập trình hướng đối tượng, quản lý sự kiện, thiết kể giao diện và xử lý dữ liệu một cách thành thạo. Đồng thời, cờ vua là một trò chơi dễ chơi, dễ học, có giao diện thân thiện và hoạt động mượt mà, những yếu tố trên sẽ mang lại trải nghiệm thú vị cho người chơi

Trong quá trình tìm hiểu lập trình, nhóm chúng em thấy khá hứng thú với việc phát triển các trò chơi thông qua thư viện Pygame của Python. Pygame là một bộ thư viện đa nền tảng được thiết kể để phát triển trò chơi điện tử. Nó bao gồm đồ họa máy tính và thư viện âm thanh đa dạng đã tạo nên các tựa game huyền thoại một thời trong quá khứ. Để tạo một môi trường lành mạnh cũng như thử sức với nhiều thử thách mới trong việc khám phá thư viện Pygame nói riêng và lập trình Python nói chung, nhóm chúng em đã chọn xây dựng game cờ vua có nhiều chế độ chơi Tựa game được ưa chuộng rộng rãi, dễ chơi, dễ học nhưng cần tư duy, khả năng suy luận- một game đơn giản nhưng nhiều thử thách và thú vị.

## Mục đích và mục tiêu dồ án

### Mục đích

Với đồ án xây dựng chương trình đánh cờ vua với nhiều với thư viện Pygame bằng ngôn ngữ lập trình Python được thực hiện nhằm hướng tới các mục đích sau:

* Rèn luyện và nâng cao kĩ năng lập trình, đông thời rèn luyện kĩ năng làm việc nhóm và phối hợp giữa các thành viên
* Phát triển tư duy, khả năng sáng tạo trong lập trình, cải thiện kỹ năng giải quyết vấn đề thực tế thông qua việc xử lý các tình huống phát sinh trong quá trình xây dựng game.
* Áp dụng và nâng cao các kiến thức đã được học.
* Tìm hiểu, áp dụng thư viện Pygame để, Tkinter vào trong đồ án để phát triển chương trình một cách tối ưu.
* Tạo ra một sản phẩm mang tính giải trí.

### Mục tiêu

Để thực hiện thực hóa các mục đích trên, đồ án đặt ra các mục tiêu như sau:

* Vận dụng được tính chất lập trình hướng đối tượng (OOP) trong Python
* Tạo một môi trường lành mạnh, sáng tạo nơi vừa giải trí vừa nâng cao kỹ năng lập trình
* Sử dụng thư viện Pygame trong việc xây dựng game cờ vua
* Sử dụng thư viện Tkinter để xây dựng giao diện thân thiện cho người dùng
* Xây dựng được hệ thống luật chơi cờ vua đầy đủ và chính xác bao gồm các nước đi hợp lệ, phong tốt, nhập thành, chiếu và chiếu hết.
* Thiết kế giao diện điều khiển linh hoạt giữa các chế độ chơi.
* Đảm bảo sản phẩm có thể chạy mượt mà, ổn định và có trải nghiệm người dùng tốt.

## Yêu cầu đồ án

### Kế hoạch thực hiện và phân công công việc

Bảng phân công công việc:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc** | **Ghi chú** |
| 1 | Phạm Đình Duy Thái | Thiết kế bố cục màn hình và bàn cờ  Vẽ hình ảnh các quân cờ  Thiết kế và xây dựng logic game  Xây dựng AI đánh cờ  Kiểm thử các chức năng và sửa lỗi  Viết báo cáo |  |
| 2 | Lê Văn Nhất | Thiết kế, xây dựng và sửa lỗi logic game  Xây dựng bố cục giao diện chọn chế độ chơi  Xây dựng AI đánh cờ  Kiểm thử các chức năng và sửa lỗi  Viết báo cáo |  |
| 3 | Mai Thành Trung | Thiết kế và xây dựng logic game  Thiết kế bố cục giao diện chọn chế độ chơi  Thiết kế và xây dựng AI đánh cờ  Kiểm thử các chức năng  Viết báo cáo |  |
| 4 | Hồ Minh Tiến | Thiết kế và xây dựng logic game  Xây dựng bố cục giao diện chọn chế độ chơi  Thiết kế, xây dựng và cải tiến AI đánh cờ  Kiểm thử các chức năng và sửa lỗi  Viết báo cáo |  |

Các mốc thời gian:

|  |  |  |
| --- | --- | --- |
| **Nhiệm vụ** | **Mốc thời gian** | **Tiến độ hoàn thành** |
| Khảo sát và lựa chọn đề tài | Tuần 1 | Hoàn thành |
| Tham khảo tài liệu liên quan | Tuần 2 | Hoàn thành |
| Thiết kế logic game | Tuần 3 | Hoàn thành |
| Thiết kế logic game | Tuần 4 | Hoàn thành |
| Thiết kế logic game | Tuần 5 | Hoàn thành |
| Thiết kế logic game | Tuần 6 | Hoàn thành |
| Thiết kế giao diện chọn chế độ chơi | Tuần 7 | Hoàn thành |
| Xây dựng AI đánh cờ | Tuần 8 | Hoàn thành |
| Xây dựng AI đánh cờ | Tuần 9 | Hoàn thành |
| Xây dựng AI đánh cờ | Tuần 10 | Hoàn thành |
| Tối ưu hóa hiệu suất AI đánh cờ | Tuần 11 | Hoàn thành |
| Kiểm thử, sửa lỗi và hoàn thành báo cáo | Tuần 12 | Hoàn thành |

### Yêu cầu chức năng

* Luật chơi hợp lệ, bám sát thực tế
* Giao diện người dùng
* Đánh cờ với máy bằng thuật toán Minimax với cắt tỉa Alpha-Beta
* Quản lí kết quả trận đấu

# PHẦN 2: GIỚI THIỆU

## Giới thiệu về ngôn ngữ lập trình Python

Ngôn ngữ lập trình Python là một ngôn ngữ lập trình bậc cao, được phát triển bởi Guido van Rossum và ra mắt lần đầu vào năm 1991. Python được thiết kế với ưu điểm dễ học, dễ nhớ, dễ đọc giúp lập trình viên hiệu suất và giảm thiểu lỗi trong quá trình. Ngoài ra, Python tạo điều kiện cho những người mới có cơ hội tiếp cận với lập trình đồng thời Python là một công cụ mạnh mẽ được sử dụng trong nhiều lĩnh vực khác nhau hiện nay.

Python là ngôn ngữ thông dịch – tức là mã nguồn Python sẽ được thực thi trực tiếp mà không cần phải biên dịch thành mã máy như các ngôn ngữ khác (C, C++, Java, …). Đồng thời, Python còn là ngôn ngữ đa năng và động (dynamically typed), có cơ chế cấp phát bộ nhớ tự động (garbage collection). Do đó, Python hỗ trợ nhiều phong cách lập trình như lập trình hướng đối tượng (OOP), lập trình hàm (functional), lập trình thủ tục (procedural). Chính vì vậy, Python là sự lựa chọn phù hợp để phát triển nhiều loại ứng dụng khác nhau.

Python được phát triển như một dự án mã nguồn mở và được quản lý bởi Python Foundation Software. Ban đầu, dự án mang tên Python được thiết kế để chạy trên Unix, nhưng đến ngày nay, Python đã tương thích với hầu hết các hệ điều hành từ Windows, MacOS đến Linux.

Một trong những ưu điểm nổi trội của Python so với những ngôn ngữ lập trình khác đó chính là hệ thống thư viện đa dạng phù hợp với nhiều mục đích lập trình khác nhau đủ để đáp ứng việc phát triển các loại ứng dụng thuộc nhiều lĩnh vực khác nhau.Ví dụ như :

* Phát triển web: Python cung cấp những framework mạnh mẽ như Flask, Django đủ sức giúp bạn xây dựng những trang web động hiệu quả và bảo mật
* Khoa học dữ liệu: Các thư viện như NumPy, Pandas, Matplotlib, Seaborn hỗ trợ xử lý và giúp bạn nhìn nhận dữ liệu một cách trực quan
* Trí tuệ nhân tạo và Machine Learning: TensorFlow, PyTorch sẽ giúp bạn dễ dàng làm việc với các mô hình học máy và mạng nơ-ron nhân tạo
* Phát triển Game: Pygame là thư viện phổ biến sẽ hỗ trợ bạn xây dựng các tựa game từ đơn giản và phức tạp.

Những ưu điểm trên đã giúp Python trở thành một trong những ngôn ngữ phổ biến nhất trên thế giới. Sự đơn giản, linh hoạt, khả năng mở rộng mạnh mẽ làm cho Python trở thành sự lựa chọn của người mới bắt đầu bước vào ngưỡng cửa lập trình và cả các chuyên gia trong các lĩnh vực công nghệ.

## Giới thiệu về thư viện Pygame

### Giới thiệu sơ bộ về thư viện Pygame

Pygame là một thư viện mã nguồn mở được viết bằng ngôn ngữ lập trình Python.Pygame được ra đời bởi Pete Shinners vào năm 2000. Pygame hỗ trợ phát triển các ứng dụng đa phương tiện và trò chơi điện tử một cách dễ dàng và chuyên nghiệp. Khi ra đời Pygame đã trở thành một công cụ mạnh mẽ cho việc phát triển trò chơi từ đơn giản đến phức tạp bằng Python.

Với việc chạy được trên hầu hết các hệ điều hành như Windows, MacOS, Linux,.., khả năng xử lý sự kiện tốt cùng việc hỗ trợ nhiều định dạng tệp bao gồm cả hình ảnh và âm thanh đã biến Pygame là sự lựa chọn hàng đầu cho tựa game hàng đầu.

Có nhiều tính năng như vậy nhưng Pygame lại hoàn toàn miễn phí khi được phát hành theo giấy phép LGPL cho phép chúng ta sử dụng Pygame để phát triển các trò chơi mã nguồn mở, phần mềm miễn phí, phần mềm thương mại mà không có bất kì sự giới hạn nào.

### Một số module trong thư viện Pygame

|  |  |
| --- | --- |
| **Module** | **Công dụng** |
| pygame.display | Điều khiển cửa sổ hiển thị và màn hình.  Các hàm phổ biến:   * pygame.display.set\_mode(): Tạo cửa sổ game với kích thước nhất định. * pygame.display.set\_caption(): Đặt tiêu đề cho cửa sổ. * pygame.display.update(): Cập nhật lại nội dung hiển thị. |
| pygame.draw | Hỗ trợ vẽ các hình cơ bản lên màn hình.  Một số hình phổ biến:   * pygame.draw.rect(): vẽ hình chữ nhật * pygame.draw.circle(): vẽ hình tròn |
| pygame.event | Tương tác với các sự kiện và các hàng đợi.  Hàm thường dùng:   * pygame.event.get(): Lấy danh sách các sự kiện mới nhất. |
| pygame.key và pygame.mouse | Xử lý bàn phím và chuột:  Hàm thường dùng:   * pygame.key.get\_pressed(): Kiểm tra phím nào đang được giữ. * pygame.mouse.get\_pos(): Lấy vị trí con trỏ chuột. * pygame.mouse.get\_pressed(): Kiểm tra nút chuột đang nhấn. |
| pygame.font | Tải phông chữ và hiển thị chữ.  Một số hàm phổ biến:   * pygame.font.init(): Khởi tạo module font. * pygame.font.SysFont(name, size): Tạo font từ hệ thống * pygame.font.Font(file, size): Tạo font từ file tùy chỉnh. * Font.render(text, antialias, color): Vẽ văn bản thành Surface (đối tượng hiển thị). * Font.size(text): Trả về kích thước chiều rộng và chiều cao |
| pygame.image | Dùng để tải và xử lý hình ảnh.  Hàm thường dùng:   * pygame.image.load(): Tải hình ảnh từ file. * Surface.blit(): Dán (render) hình ảnh lên màn hình game. |
| pygame.mixer | Dùng để thêm âm thanh vào game (nhạc nền, hiệu ứng…).  Các thành phần:   * pygame.mixer.Sound(): Tạo đối tượng âm thanh. * pygame.mixer.music.load(): Tải nhạc nền. * pygame.mixer.music.play(): Phát nhạc nền. |
| pygame.time | Điều khiển tốc độ khung hình và xử lý thời gian.  Ví dụ:   * + pygame.time.Clock(): Tạo đồng hồ để giới hạn FPS.   + Clock.tick(fps): Điều chỉnh tốc độ cập nhật game. |

## Giới thiệu về thư viện Tkinter

### Giới thiệu sơ bộ về thư viện Tkinter

Tkinter là một thư viện trong ngôn ngữ lập trình Python được sử dụng để tạo giao diện cho người dùng (GUI). Tkinter là viết tắt của từ “Toolkit interface”. Tkinter là một phần thư viện được tích hợp sẵn. Điều này giúp Tkinter trở thành một trong những lựa chọn khá phổ biến trong việc phát triển ứng dụng và giao diện đơn giản bằng Python. Tkinter cung cấp đầy đủ các sự kiện và phương thức để xử lý trạng thái người dùng.

### Một số widget trong thư viện Tkinter

|  |  |
| --- | --- |
| **Widget** | **Chức năng** |
| Button | Thêm nút để tương tác với ứng dụng Python |
| Canvas | Vẽ các đối tượng hoặc hình ảnh trên cửa sổ |
| Checkbutton | Hiển thị và cho phép người dùng chọn hoặc bỏ chọn |
| Entry | Hiển thị trường nhập dữ liệu một dòng cho người dùng |
| Frame | Vùng chứa để nhóm hoặc tổ chức widget khác |
| Label | Hiển thị văn bản hoặc thông điệp cho người dùng |
| Listbox | Hiển thị danh sách các tùy chọn hoặc mục |
| Menubutton | Cung cấp một nút mở rộng để hiển thị danh sách menu |
| Menu | Thêm các mục menu vào ứng dụng |
| Message | Hiển thị tin nhắn hoặc thông báo cho người dùng |
| Radiobutton | Cho phép người dùng chọn một trong nhiều tùy chọn |
| Scale | Cung cấp thanh trượt cho người dùng để chọn giá trị trong một dải |
| Scrollbar | Cung cấp thanh cuộn để người dùng có thể di chuyển trong nội dung |
| Text | Cung cấp một vùng nhập văn bản nhiều dòng cho việc chỉnh sửa và hiển thị nội dung |
| Toplevel | Tạo một cửa sổ phụ độc lập |
| Spinbox | Cho phép người dùng chọn từ danh sách các giá trị |
| PanedWindow | Chia một phần giao diện thành nhiều khu vực có thể thay đổi kích thước |
| LabelFrame | Vùng chứa giúp nhóm các widget lại với nhau và cung cấp tiêu đề cho nhóm đó |

### Một số phương thức điều khiển bố cục giao diên trong thư viện Tkinter

|  |  |
| --- | --- |
| **Bố cục** | **Cách thức hoạt động** |
| Pack Layout | Phương pháp này cho phép bạn sắp xếp các widget dọc theo một trục (thường là theo chiều dọc) và chiếm không gian tối đa trên cửa sổ. |
| Grid Layout | Phương pháp này giúp phép bạn sắp xếp các widget dựa trên một lưới ô. Tương ứng với mỗi widget được đặt tại một hàng và một cột cụ thể. |
| Place Layout | Phương pháp này cho phép bạn chỉ định vị trí cụ thể của các widget trên cửa sổ. Như vậy, bạn cần chỉ định tọa độ (x, y) của widget để đặt chúng trên cửa sổ. |

## Giới thiệu về trò chơi cờ vua

### Lịch sử hình thành và phát triển trò chơi cờ vua

Cờ vua là một trong những trò chơi trí tuệ lâu đời, có nguồn gốc từ Ấn Độ vào khoảng thế kỷ VI với tên gọi Chaturanga. Từ Ấn Độ, trò chơi lan sang Ba Tư và trở thành Shatranj, sau đó, cờ vua bắt đầu phổ biến ở Châu Âu vào thời Trung Cổ

Đến cuối thế kỷ XV, cờ vua dần hoàn thiện với các quy tắc như hiện tại như phong cấp tốt, nhập thành, và quân hậu có thể di chuyển mạnh hơn. Năm 1886, giải đấu tranh chức vô địch cờ vua thế giới đầu tiên được tổ chức, đánh dấu sự chuyên nghiệp hóa của bộ môn thể thao này.

Ngày nay, cờ vua được quản lý bởi Liên đoàn Cờ vua Quốc tế (FIDE), với nhiều giải đấu lớn như Giải vô địch cờ vua thế giới, Chess Olympiad, Giải Candidates.

### Luật chơi cờ vua

Cờ vua được chơi trên bàn cờ có 8 hàng được đánh số từ 1-8, và 8 cột được ký hiệu từ A-H, tổng cộng có 64 ô vuông xen kẽ trắng và đen.

Mỗi bên có 16 quân cờ gồm:

* **1 Vua (King):** Đi 1 ô theo mọi hướng, không được để bị chiếu
* **1 Hậu (Queen):** Đi theo hàng ngang, dọc, chéo không giới hạn ô
* **2 Xe (Rook):** Đi ngang hoặc dọc không giới hạn ô
* **2 Tượng (Bishop):** Đi chéo không giới hạn ô
* **2 Mã (Knight):** Đi theo hinh chứ “L” (2 ô theo một hướng và 1 ô vuông góc)
* **8 Tốt (Pawn):**
* Đi 1 ô về phía trước, nếu chưa di chuyển có thể đi 2 ô.
* Ăn chéo 1 ô
* Khi đến hàng cuối, được phong cấp thành một quân hậu

Một số luật đặc biệt

***Nhập thành (Castling):***

* Chỉ được thực hiện nếu Vua và Xe chưa di chuyển
* Không có quân nào chắn giữa Vua và Xe
* Vua không được đang bị chiếu hoặc đi qua ô bị kiểm soát

***Bắt tốt qua đường (En Passant):***

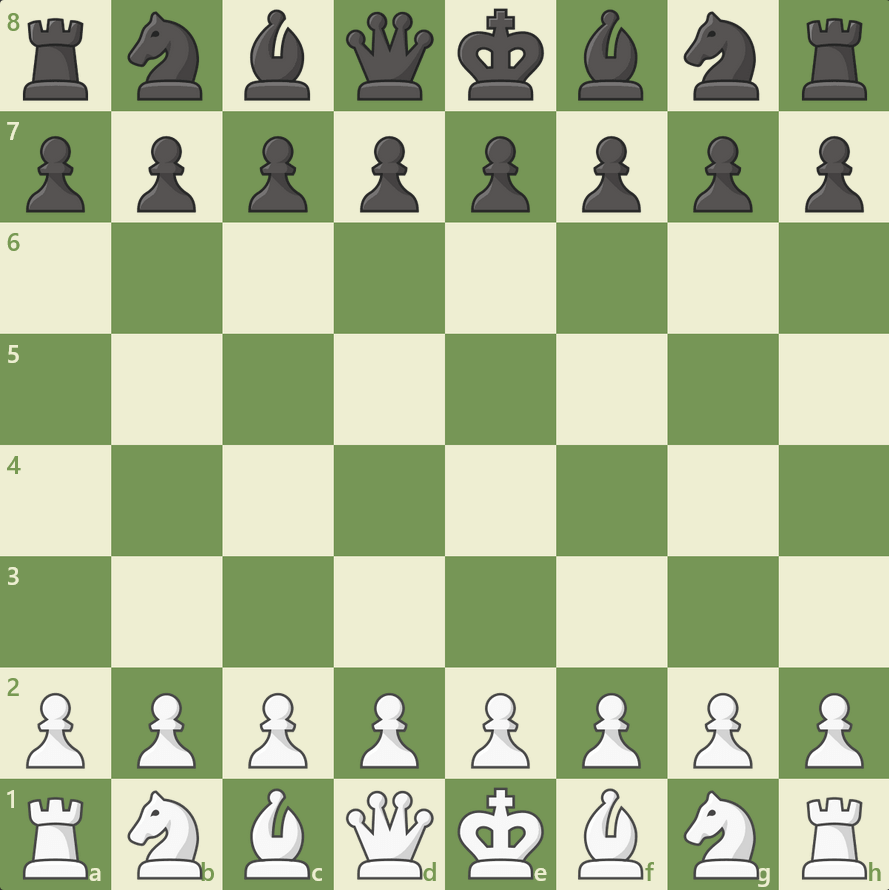
* Nếu một quân tốt di chuyển 2 ô từ vị trí ban đầu và đứng cạnh quân tốt đối phương, đối phương có thể bắt nó như khi nó chỉ đi 1 ô.

***Chiếu hết (Checkmate):***

* Khi Vua bị chiếu và không có nước đi nào để thoát, ván đấu kết thúc.

***Hòa (Draw):*** Có nhiều cách hòa như:

* Thế cờ bế tắc (Stalemate): Người chơi không còn nước đi hợp lệ nhưng Vua không bị chiếu
* Lặp lại 3 lần: Khi cùng một vị trí lặp lại 3 lần trong ván đấu.
* Luật 50 nước đi: Nếu trong 50 nước đi liên tiếp không bắt quân hoặc di chuyển tốt, ván đấu hòa.



# PHẦN 3: PHÁT TRIỂN TRÒ CHƠI CỜ VUA

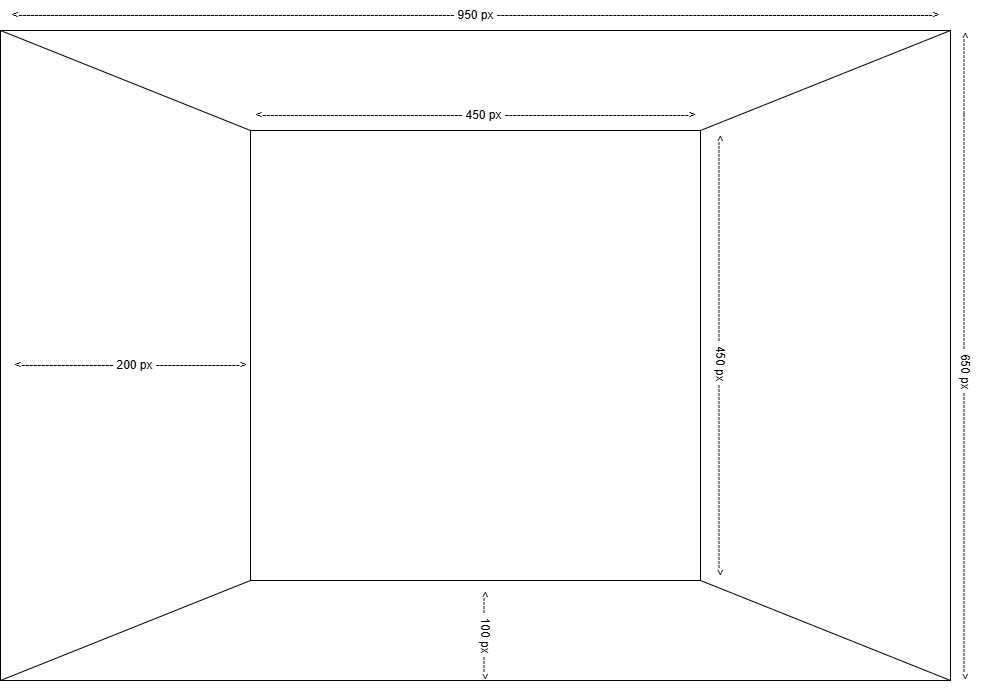
## Phân tích trò chơi

Trò chơi bao gồm một khung hình được giới hạn, bên trong là bàn cờ 8x8 với các quân cờ có các cách đi khác nhau được sắp xếp trên bàn cờ. Các điểm quan trọng cần lập tình là:

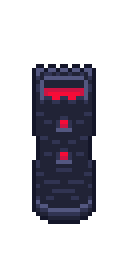
1. Bàn cờ: bao gồm 64 ô vuông trắng đen xen kẽ
2. Các quân cờ: 6 loại quân cờ khác nhau và các bước di chuyển khác biệt
3. Luật chơi: Kiểm tra nước đi hợp lệ, chiếu hết, phong cấp tốt, …
4. Trạng thái ván đấu: Xác định ván đấu khi nào kết thúc lúc nào chiếu hết, lúc nào hòa
5. Tương tác với người chơi: Chọn quân cờ, hiển thị nước đi hợp lệ, thực hiện nước đi
6. Chế độ chơi: Chơi với người hoặc chơi với máy

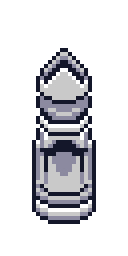
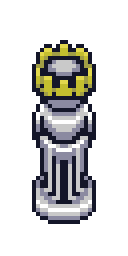
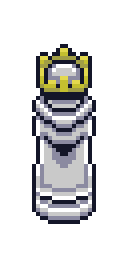
## Thiết kế màn hình trò chơi

* Thiết kế bố cục kích thước màn hình



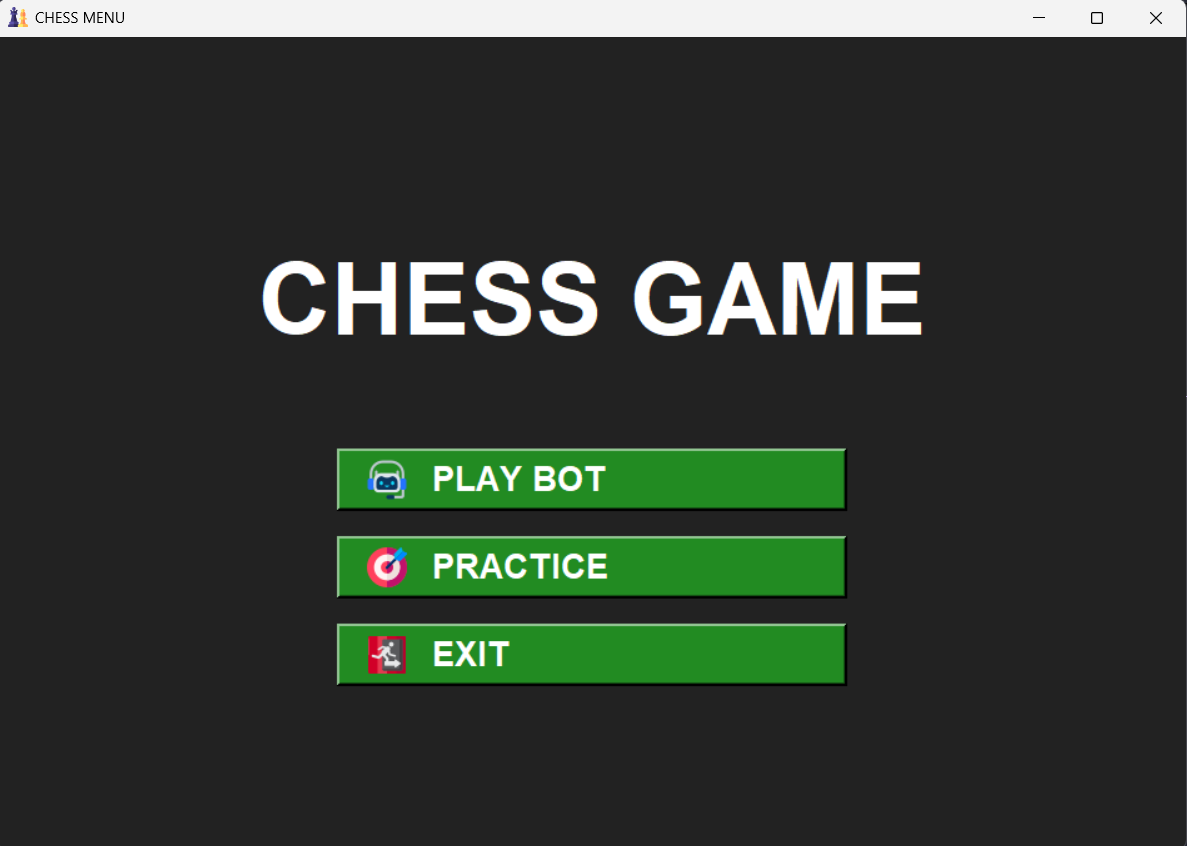
* Thiết kế hình ảnh các quân cờ
* Trong quá trình xây dựng giao diện đồ họa cho game cờ vua, nhóm đã tham khảo một số mẫu quân cờ có sẵn trên internet để lấy cảm hứng thiết kế. Sau đó, nhóm tự vẽ và có một số diều chỉnh. Mục tiêu là tạo nên những quân cờ có tính thẩm mỹ, dễ nhận diện và nhất quán với giao diện tổng thể, ý tưởng mà nhóm mong muốn.
* Nguồn tham khảo hình ảnh gốc: xem tại mục [5] trong phần Tài liệu tham khảo.
* Công cụ sử dụng để thiết kế: công cụ thiết kế trực tuyến **Pixilart**

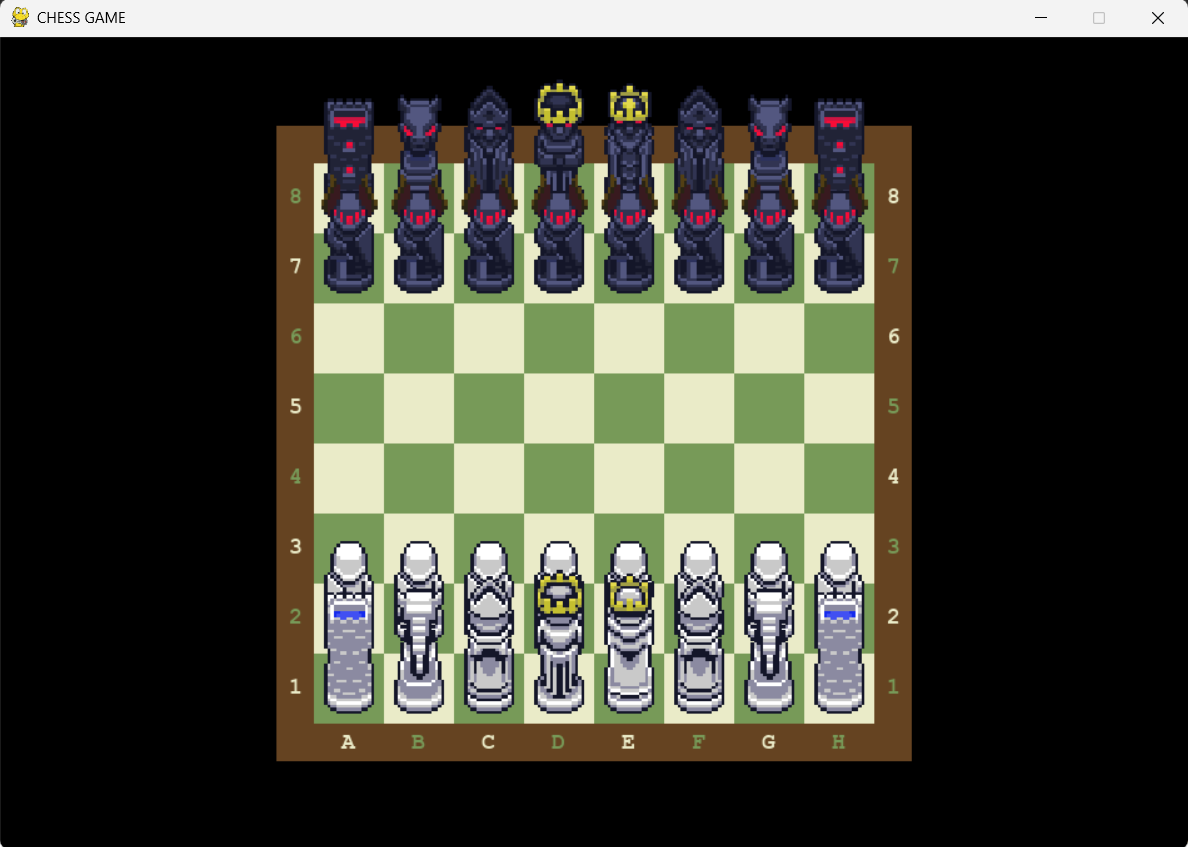
     

## Giao diện trò chơi

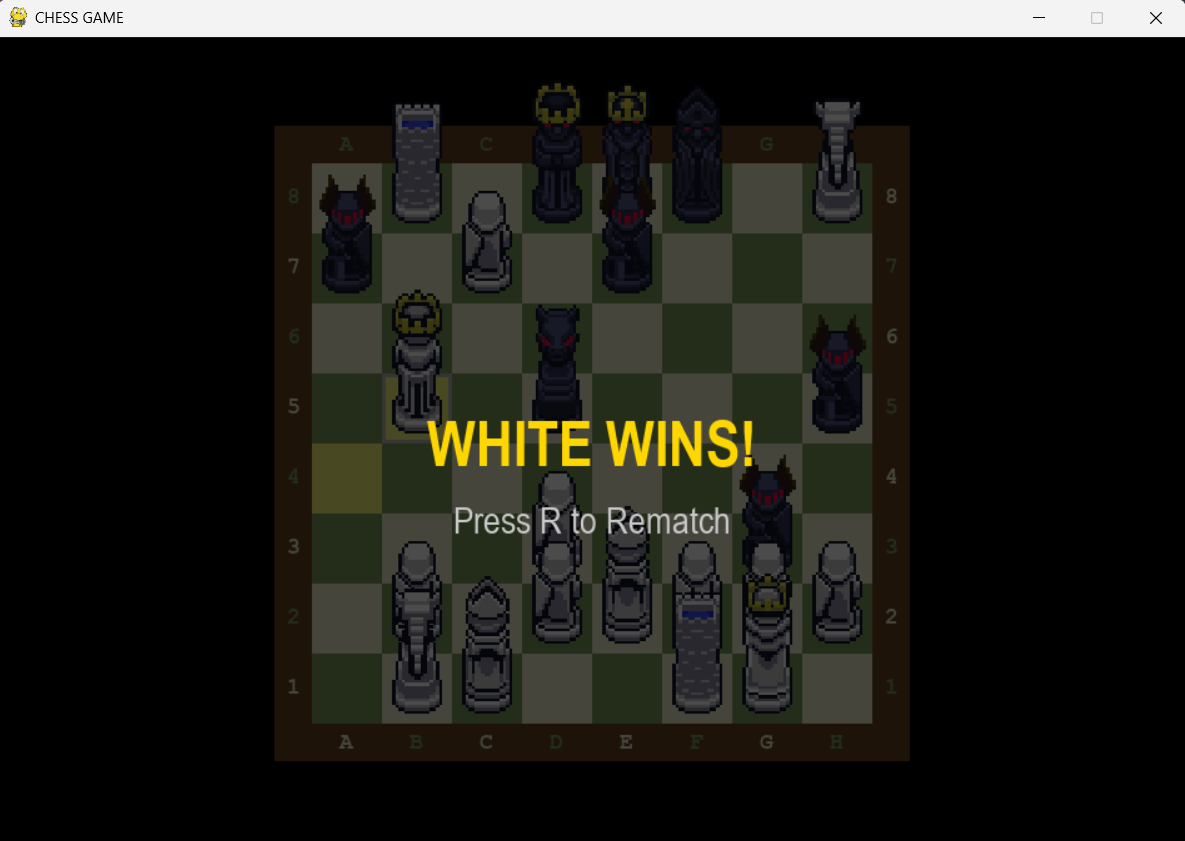
* Giao diện chọn chế độ chơi



* Giao diện trò chơi



* Giao diện kết thúc trận cờ



## Thực hiện viết mã Python

### Tổ chức cây thư mục

CHESSGAMEWITHAI/

├── assets/

│ ├── icons/

│ ├── images/

│ └── sounds/

├── src/

│ ├── \_\_pycache\_\_/

│ ├── ai.py

│ ├── board.py

│ ├── config.py

│ ├── const.py

│ ├── game.py

│ ├── main.py

│ ├── menu.py

│ ├── move.py

│ ├── pieces.py

│ ├── selector.py

│ └── square.py

├── README.md

└── requirements.txt

### Viết mã xử lí

#### Quản lí điều hành trò chơi (main.py)

import pygame

import sys

import copy

from const import \*

from game import Game

from square import Square

from move import Move

from ai import AI\_Minimax

from menu import get\_game\_mode

**class Main:**

**def \_\_init\_\_(self):**

        pygame.init()

        self.screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))

        pygame.display.set\_caption('CHESS GAME')

        self.game = Game()

        self.restart\_to\_menu = False

        self.ai\_enabled = True

        self.ai\_thinking = False

        self.ai\_move = None

        self.ai\_delay = 400  # Thời gian delay để quan sát nước đi AI (ms)

**def handle\_ai\_turn(self):**

        """Xử lý lượt đi của AI cho quân đen"""

        if (self.ai\_enabled

            and not self.ai\_thinking

            and self.game.next\_player == 'black'

            and not self.game.game\_over):

            self.ai\_thinking = True

            # Tạo bản sao bàn cờ để AI tính toán

            board\_copy = copy.deepcopy(self.game.board)

            ai = AI\_Minimax(board\_copy)

            self.ai\_move = ai.get\_best\_move('black')

            self.ai\_thinking = False

**def execute\_ai\_move(self):**

        """Thực hiện nước đi của AI"""

        if self.ai\_move and not self.game.game\_over:

            piece, move = self.ai\_move

            final\_square = self.game.board.squares[move.final.row][move.final.col]

            captured = final\_square.has\_piece()

            # Thực hiện nước đi

            self.game.board.move(piece, move)

            self.game.board.set\_true\_en\_passant(piece)

            self.game.play\_sound(captured)

            self.game.next\_turn()

            # Kiểm tra trạng thái game

            self.check\_game\_status()

            self.ai\_move = None

**def check\_game\_status(self):**

        """Kiểm tra trạng thái kết thúc game"""

        for color in ['white', 'black']:

            status = self.game.board.check\_game\_status(color)

            if status == 'checkmate':

                self.game.game\_over = True

                self.game.winner = 'black' if color == 'white' else 'white'

                self.game.result = 'checkmate'

                break

            elif status == 'stalemate':

                self.game.game\_over = True

                self.game.result = 'stalemate'

                break

**def handle\_events(self):**

        """Xử lý tất cả sự kiện đầu vào"""

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

                sys.exit()

            if not self.game.game\_over:

                if event.type == pygame.MOUSEBUTTONDOWN:

                    self.handle\_mouse\_click(event)

                elif event.type == pygame.MOUSEMOTION:

                    self.handle\_mouse\_motion(event)

            if event.type == pygame.KEYDOWN:

                self.handle\_key\_press(event)

**def handle\_mouse\_click(self, event):**

        """Xử lý click chuột của người chơi"""

        mouseX, mouseY = event.pos

        clicked\_col = (mouseX - BOARD\_X) // SQSIZE

        clicked\_row = (mouseY - BOARD\_Y) // SQSIZE

        if 0 <= clicked\_row < ROWS and 0 <= clicked\_col < COLS:

            clicked\_square = self.game.board.squares[clicked\_row][clicked\_col]

            selector = self.game.selector

            # Đã chọn quân cờ trước đó

            if selector.selecting:

                self.handle\_piece\_move(clicked\_row, clicked\_col, clicked\_square)

            # Chọn quân cờ mới

            elif clicked\_square.has\_piece() and clicked\_square.piece.color == self.game.next\_player:

                self.select\_piece(clicked\_row, clicked\_col, clicked\_square)

**def handle\_piece\_move(self, row, col, square):**

        selector = self.game.selector

        selected\_piece = selector.piece

        start\_row, start\_col = selector.rowcol

        move = Move(Square(start\_row, start\_col), Square(row, col))

        if self.game.board.valid\_move(selected\_piece, move):

            captured = square.has\_piece()

            self.game.board.move(selected\_piece, move)

            self.game.board.set\_true\_en\_passant(selected\_piece)

            self.game.play\_sound(captured)

            self.draw\_game()

            pygame.display.update()

            self.game.next\_turn()

            self.check\_game\_status()

        selector.unselect\_piece()

**def select\_piece(self, row, col, square):**

        """Chọn quân cờ để di chuyển"""

        piece = square.piece

        if piece.color == self.game.next\_player:

            self.game.board.calc\_moves(piece, row, col, True)

            self.game.selector.select\_piece(piece, (row, col))

**def handle\_mouse\_motion(self, event):**

        """Xử lý di chuyển chuột"""

        hover\_row = (event.pos[1] - BOARD\_Y) // SQSIZE

        hover\_col = (event.pos[0] - BOARD\_X) // SQSIZE

        self.game.set\_hover(hover\_row, hover\_col)

**def handle\_key\_press(self, event):**

        """Xử lý phím bấm"""

        if event.key == pygame.K\_t:

            self.game.change\_theme()

        elif event.key == pygame.K\_r:

            self.reset\_game()

        elif event.key == pygame.K\_a:

            self.ai\_enabled = not self.ai\_enabled  # Bật/tắt AI

        elif event.key == pygame.K\_ESCAPE:

            self.restart\_to\_menu = True  # Đặt cờ để thoát về menu

**def reset\_game(self):**

        """Reset game về trạng thái ban đầu"""

        self.game.reset()

        self.game.game\_over = False

        self.game.winner = None

        self.game.result = None

        self.ai\_thinking = False

        self.ai\_move = None

**def draw\_game(self):**

        """Vẽ tất cả thành phần game"""

        self.screen.fill((0, 0, 0))

        self.game.show\_bg(self.screen)

        self.game.show\_last\_move(self.screen)

        self.game.show\_moves(self.screen)

        self.game.show\_hover(self.screen)

        self.game.show\_pieces(self.screen)

        if self.game.game\_over:

            self.draw\_game\_over()

**def draw\_game\_over(self):**

        """Hiển thị thông báo kết thúc game"""

        overlay = pygame.Surface((SCREEN\_WIDTH, SCREEN\_HEIGHT), pygame.SRCALPHA)

        overlay.fill((0, 0, 0, 180))

        self.screen.blit(overlay, (0, 0))

        font = pygame.font.SysFont('Arial', 50, bold=True)

        if self.game.result == 'checkmate':

            winner = 'WHITE' if self.game.winner == 'white' else 'BLACK'

            text = f"{winner} WINS!"

            color = (255, 215, 0)

        else:

            text = "DRAW!"

            color = (255, 255, 255)

        text\_surface = font.render(text, True, color)

        text\_rect = text\_surface.get\_rect(center=(SCREEN\_WIDTH//2, SCREEN\_HEIGHT//2))

        self.screen.blit(text\_surface, text\_rect)

        font\_small = pygame.font.SysFont('Arial', 30)

        restart\_text = font\_small.render("Press R to Rematch", True, (200, 200, 200))

        restart\_rect = restart\_text.get\_rect(center=(SCREEN\_WIDTH//2, SCREEN\_HEIGHT//2 + 60))

        self.screen.blit(restart\_text, restart\_rect)

**def mainLoop(self):**

        clock = pygame.time.Clock()

        while True:

            # Xử lý sự kiện

            self.handle\_events()

            # Nếu người dùng bấm ESC, thoát vòng lặp để quay lại menu

            if self.restart\_to\_menu:

                pygame.display.quit()

                break

            # Xử lý AI

            if not self.game.game\_over and self.game.next\_player == 'black' and self.ai\_enabled:

                self.handle\_ai\_turn()

                if self.ai\_move:

                    pygame.time.delay(self.ai\_delay)  # Delay để quan sát

                    self.execute\_ai\_move()

                    self.draw\_game()  # Vẽ lại sau khi AI đi

                    pygame.display.update()

            # Vẽ game (bao gồm cả khi người chơi đi)

            self.draw\_game()

            pygame.display.update()

            clock.tick(60)

if \_\_name\_\_ == "\_\_main\_\_":

    while True:

        mode = get\_game\_mode()

        if mode is None:

            sys.exit()

        # Khởi chạy game

        main = Main()

        main.ai\_enabled = mode

        main.mainLoop()

        if not main.restart\_to\_menu:

            break

#### Quản lí giao diện chọn chế độ chơi (menu.py)

import tkinter as tk

from tkinter import PhotoImage

import const

import os

**class MenuUI:**

**def \_\_init\_\_(self, root):**

        self.root = root

        self.root.title("CHESS MENU")

        self.root.configure(bg="#222222")

        self.selected\_mode = None

        icon\_path = os.path.join(os.path.dirname(\_\_file\_\_), "../assets/icons")

        chess\_icon = PhotoImage(file=os.path.join(icon\_path, "chess.png"))

        self.root.tk.call('wm', 'iconphoto', self.root.\_w, chess\_icon)

        self.icon\_bot = PhotoImage(file=os.path.join(icon\_path, "bot.png"))

        self.icon\_practice = PhotoImage(file=os.path.join(icon\_path, "practice.png"))

        self.icon\_exit = PhotoImage(file=os.path.join(icon\_path, "exit.png"))

        root.update\_idletasks()

        w = const.SCREEN\_WIDTH

        h = const.SCREEN\_HEIGHT

        screen\_width = root.winfo\_screenwidth()

        screen\_height = root.winfo\_screenheight()

        x = (screen\_width - w) // 2

        y = (screen\_height - h) // 2

        root.geometry(f"{w}x{h}+{x}+{y}")

        frame = tk.Frame(root, bg="#222222")

        frame.pack(expand=True)

        title = tk.Label(frame, text="CHESS GAME", font=("Arial", 60, "bold"),

                         fg="white", bg="#222222")

        title.pack(pady=(40, 60))

        self.create\_button(frame, "PLAY BOT", lambda: self.select\_mode(True), self.icon\_bot)

        self.create\_button(frame, "PRACTICE", lambda: self.select\_mode(False), self.icon\_practice)

        self.create\_button(frame, "EXIT", root.quit, self.icon\_exit)

**def on\_enter (self, e):**

        e.widget.config(bg="#FF1493")

**def on\_leave (self, e):**

        e.widget.config(bg="#228B22")

**def create\_button (self, parent, text, command, icon=None):**

        btn = tk.Button(parent, text=text, command=command,

                        width=360, height=40,

                        font=("Arial", 20, "bold"),

                        bg="#228B22", fg="white",

                        image=icon, compound="left",

                        activebackground="#45a049", activeforeground="white",

                        relief="raised", bd=3,

                        anchor="w", padx=20)

        btn.bind("<Enter>", self.on\_enter)

        btn.bind("<Leave>", self.on\_leave)

        btn.pack(pady=10)

**def select\_mode (self, ai\_enabled):**

        self.selected\_mode = ai\_enabled

        self.root.destroy()

**def get\_game\_mode ():**

    root = tk.Tk()

    menu = MenuUI(root)

    root.mainloop()

    return menu.selected\_mode

#### Quản lí ô cờ (square.py)

**class Square:**

  ALPHACOLS = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E', 5:'F', 6:'G', 7:'H'}

  """ KHỞI TẠO MỘT Ô CỜ VỚI : HÀNG(row), CỘT(col), QUÂN CỜ(piece-nếu có) """

**def \_\_init\_\_(self, row, col, piece = None):**

    self.row = row

    self.col = col

    self.piece = piece

    self.alphacol = self.ALPHACOLS[col]

  """So sánh hai ô cờ dựa trên vị trí hàng và cột."""

**def \_\_eq\_\_(self, other):**

    if not hasattr(other, 'row') or not hasattr(other, 'col'):

        return False

    return self.row == other.row and self.col == other.col

  """Kiểm tra xem ô cờ có chứa quân cờ hay không."""

**def has\_piece (self):**

    return self.piece != None

  """Kiểm tra xem ô cờ có trống không."""

**def isempty (self):**

    return not self.has\_piece()

  """Kiểm tra xem ô có chứa quân cờ của cùng một đội (màu) hay không."""

**def has\_team\_piece (self, color):**

    return self.has\_piece() and self.piece.color == color

  """Kiểm tra xem ô có chứa quân cờ của đối thủ hay không."""

**def has\_enemy\_piece (self, color):**

    return self.has\_piece() and self.piece.color != color

  """Kiểm tra xem ô có trống hoặc chứa quân cờ của đối thủ hay không."""

**def isempty\_or\_enemy (self, color):**

    return self.isempty() or self.has\_enemy\_piece(color)

  """Kiểm tra xem các giá trị hàng, cột có nằm trong phạm vi hợp lệ (0-7) hay không."""

**@staticmethod**

**def in\_range(\*args):**

    for arg in args:

      if arg < 0 or arg > 7:

        return False

    return True

  """Trả về ký tự chữ cái tương ứng với số cột (A-H)."""

**@staticmethod**

**def get\_alphacol(col):**

    ALPHACOLS = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E', 5:'F', 6:'G', 7:'H'}

    return ALPHACOLS[col]

#### Quản lí các lớp quân cờ (piece.py)

import os

**class Piece:**

**def \_\_init\_\_(self, name, color, value, texture = None, texture\_rect = None):**

    self.name = name

    self.color = color

    value\_sign = 1 if color == "white" else -1

    self.value = value \* value\_sign

    self.moves = [] # Danh sách các nước đi hợp lệ

    self.moved = False # Kiểm tra quân cờ đã di chuyển chưa

    self.set\_texture() # Gán đường dẫn hình ảnh cho các quâ cờ

    self.texture\_rect  = texture\_rect # Lưu vị trí hình ảnh trên màn hình

**def set\_texture(self, size = 80):**

    self.texture = os.path.join(

      f'assets/images/imgs-{size}px/{self.color}\_{self.name}.png’ )

**def add\_move(self, move):**

    self.moves.append(move)

**def clear\_move(self):**

    self.moves = []

**class Pawn(Piece):**

**def \_\_init\_\_(self, color):**

      if color == "white":

          self.dir = -1

      else:

          self.dir = 1

      self.en\_passant = False

      super().\_\_init\_\_("pawn", color, 1.0)

**class Knight(Piece):**

  def \_\_init\_\_(self, color):

    super().\_\_init\_\_("knight", color, 3.0)

**class Bishop(Piece):**

  def \_\_init\_\_(self, color):

    super().\_\_init\_\_("bishop", color, 3.0)

**class Rook(Piece):**

  def \_\_init\_\_(self, color):

    super().\_\_init\_\_("rook", color, 5.0)

**class Queen(Piece):**

  def \_\_init\_\_(self, color):

    super().\_\_init\_\_("queen", color, 9.0)

**class King(Piece):**

  def \_\_init\_\_(self, color):

    self.left\_rook = None

    self.right\_rook = None

    super().\_\_init\_\_("king", color, 1000.0)

#### Quản lí bàn cờ (board.py)

from const import \*

from pieces import \*

from square import Square

from move import Move

from config import Sound

import copy

import os

class Board:

  """Khởi tạo bàn cờ với 8x8 ô và thêm các quân cờ vào vị trí ban đầu."""

**def \_\_init\_\_(self):**

    self.squares = [

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 0, 0, 0, 0, 0, 0]

    ]

    self.last\_move = None

    self.\_create()

    self.\_add\_pieces("white")

    self.\_add\_pieces("black")

  """============== TẠO CÁC Ô TRÊN BÀN CỜ DƯỚI DẠNG Square =============== """

**def \_create(self):**

    for row in range(ROWS):

      for col in range(COLS):

        self.squares[row][col] = Square(row, col)

  """================ THÊM QUÂN CỜ VÀO BÀN CỜ THEO MÀU SẮC =============== """

**def \_add\_pieces(self,color):**

    row\_pawn, row\_other = (6, 7) if color == "white" else (1, 0)

    # Thêm pawns(tốt)

    for col in range(COLS):

      self.squares[row\_pawn][col] = Square(row\_pawn, col, Pawn(color))

    # Thêm knight(mã)

    self.squares[row\_other][1] = Square(row\_other, 1, Knight(color))

    self.squares[row\_other][6] = Square(row\_other, 6, Knight(color))

    # Thêm bishop(tượng)

    self.squares[row\_other][2] = Square(row\_other, 2, Bishop(color))

    self.squares[row\_other][5] = Square(row\_other, 5, Bishop(color))

    # Thêm rook(xe)

    self.squares[row\_other][0] = Square(row\_other, 0, Rook(color))

    self.squares[row\_other][7] = Square(row\_other, 7, Rook(color))

    # Thêm king(vua)

    self.squares[row\_other][4] = Square(row\_other, 4, King(color))

    # Thêm queen(hậu)

    self.squares[row\_other][3] = Square(row\_other, 3, Queen(color))

  """ DI CHUYẺN QUÂN CỜ TỪ Ô HIỆN TẠI ĐẾN Ô ĐƯỢC CHỌN VỚI CÁC TRƯỜNG HỢP ĐẶC BIỆT """

**def move(self, piece, move, testing = False):**

    initial = move.initial

    final = move.final

    en\_passant\_empty = self.squares[final.row][final.col].isempty() # ô điểm đến trống

    if isinstance(piece, Pawn):

        piece.has\_moved = True

    # console board move update

    self.squares[initial.row][initial.col].piece = None

    self.squares[final.row][final.col].piece = piece

    # PHONG CẤP QUÂN TỐT

    if isinstance(piece, Pawn):

      differecne = final.col - initial.col

      if differecne != 0 and en\_passant\_empty:

        self.squares[initial.row][initial.col + differecne].piece = None # xóa tốt bị bắt

        self.squares[final.row][final.col].piece = piece

        if not testing:

          sound = Sound(os.path.join('assets/sounds/capture.wav'))

          sound.play()

      # pawn

      else:

        self.promote\_pawn(piece, final)

    # king castling

    if isinstance(piece, King):

      if self.check\_castling(initial, final) and not testing:

        differecne = final.col - initial.col

        rook = piece.left\_rook if (differecne < 0) else piece.right\_rook

        # if rook is not None and rook.moves:

        self.move(rook, rook.moves[-1])

    piece.moved = True

    piece.clear\_move()

    self.last\_move = move

  """===============================================KIỂM TRA TRẠNG THÁI================================================"""

**def check\_game\_status(self, color):**

    # Tìm Vua

    print(f"\n=== Kiem tra trang thai cho {color} ===")

    king\_pos = None

    for row in range(ROWS):

        for col in range(COLS):

            piece = self.squares[row][col].piece

            if isinstance(piece, King) and piece.color == color:

                king\_pos = (row, col)

                print(f"[DEBUG] Tim thay Vua tai ({row}, {col})")

                break

        if king\_pos:

            break

    if not king\_pos:

        print("[DEBUG] Không tìm thấy Vua!")

        return None

    # Kiểm tra chiếu

    in\_check = False

    for row in range(ROWS):

        for col in range(COLS):

            piece = self.squares[row][col].piece

            if piece and piece.color != color:

                self.calc\_moves(piece, row, col, bool=False)

                for move in piece.moves:

                    if (move.final.row, move.final.col) == king\_pos:

                        print(f"[DEBUG] {piece.\_\_class\_\_.\_\_name\_\_} tại ({row}, {col}) đang chiếu Vua!")

                        in\_check = True

                piece.clear\_move()

                if in\_check:

                    break

        if in\_check:

            break

    # Kiểm tra nước đi hợp lệ

    has\_legal\_move = False

    for row in range(ROWS):

        for col in range(COLS):

            piece = self.squares[row][col].piece

            if piece and piece.color == color:

                self.calc\_moves(piece, row, col, bool=True)

                if piece.moves:

                    print(f"[DEBUG] {piece.\_\_class\_\_.\_\_name\_\_} tai ({row}, {col}) co nuoc di hop le")

                    has\_legal\_move = True

                piece.clear\_move()

                if has\_legal\_move:

                    break

        if has\_legal\_move:

            break

    # Xác định trạng thái

    if in\_check:

        if not has\_legal\_move:

            print("[DEBUG] Kết quả: CHIẾU BÍ!")

            return 'checkmate'

        print("[DEBUG] Kết quả: Đang bị chiếu nhưng còn nước đi")

        return 'check'

    elif not has\_legal\_move:

        print("[DEBUG] Kết quả: HẾT NƯỚC ĐI (hòa)")

        return 'stalemate'

    print("[DEBUG] Ket qua: Trang thai binh thuong")

    return None

  """=============================================END KIỂM TRA TRẠNG THÁI============================================"""

  """===================== KIỂM TRA NƯỚC ĐI HỢP LỆ ======================="""

**def valid\_move(self, piece, move):**

    return move in piece.moves

  """===============THĂNG CẤP QUÂN TỐT KHI ĐẾN CUỐI BÀN CỜ ==============="""

**def promote\_pawn(self, piece, final):**

    if final.row == 0 or final.row == 7:

      self.squares[final.row][final.col].piece = Queen(piece.color)

  """======================== KIỂM TRA NHẬP THÀNH ========================"""

**def check\_castling(self, initial, final):**

    return abs(initial.col - final.col) == 2

  """============ ĐÁNH DẤU QUÂN TỐT CÓ THỂ BÁT TỐT QUA ĐƯỜNG ============="""

**def set\_true\_en\_passant(self, piece):**

    if not isinstance(piece, Pawn):

      return

    for row in range(ROWS):

      for col in range(COLS):

        if isinstance(self.squares[row][col].piece, Pawn):

          self.squares[row][col].piece.en\_passant = False

    piece.en\_passant = True

  """===== KIỂM TRA NƯỚC ĐI CÓ ĐẶT VUA VÀO TÌNH TRẠNG BỊ CHIẾU KHÔNG ===== """

**def in\_check(self, piece, move):**

    temp\_piece = copy.deepcopy(piece)

    temp\_board = copy.deepcopy(self)

    temp\_board.move(temp\_piece, move, testing=True)

    # Tìm vị trí vua sau nước đi

    king\_pos = None

    for r in range(ROWS):

        for c in range(COLS):

            p = temp\_board.squares[r][c].piece

            if isinstance(p, King) and p.color == piece.color:

                king\_pos = (r, c)

                break

        if king\_pos:

            break

    # Kiểm tra xem quân địch có thể tấn công vị trí vua không

    for row in range(ROWS):

        for col in range(COLS):

            if temp\_board.squares[row][col].has\_enemy\_piece(piece.color):

                enemy\_piece = temp\_board.squares[row][col].piece

                temp\_board.calc\_moves(enemy\_piece, row, col, bool=False)

                for m in enemy\_piece.moves:

                    if (m.final.row, m.final.col) == king\_pos:

                        return True

    return False

  """============ TÍNH TOÁN CÁC NƯỚC ĐI HỢP LỆ CẢU QUÂN CỜ =============== """

**def calc\_moves(self, piece, row, col, bool = True):**

**def pawn\_moves():**

      steps = 1 if piece.moved else 2

      possible\_move\_row = row + piece.dir

      if Square.in\_range(possible\_move\_row) and self.squares[possible\_move\_row][col].isempty():

        initial = Square(row, col)

        final = Square(possible\_move\_row, col)

        move = Move(initial, final)

        if bool:

            if not self.in\_check(piece, move):

                piece.add\_move(move)

        else:

            piece.add\_move(move)

    # Bước đi đầu tiên (2 ô) - kiểm tra bằng vị trí ban đầu

      if (piece.color == "white" and row == 6) or (piece.color == "black" and row == 1):

        possible\_move\_row = row + (piece.dir \* 2)

        intermediate\_row = row + piece.dir

        if (Square.in\_range(possible\_move\_row) and

            self.squares[possible\_move\_row][col].isempty() and

            self.squares[intermediate\_row][col].isempty()):

            initial = Square(row, col)

            final = Square(possible\_move\_row, col)

            move = Move(initial, final)

            if bool:

                if not self.in\_check(piece, move):

                    piece.add\_move(move)

            else: piece.add\_move(move)

      # diagonal moves

      possible\_move\_row = row + piece.dir

      possible\_move\_cols = [col-1, col+1]

      for possible\_move\_col in possible\_move\_cols:

        if Square.in\_range(possible\_move\_row, possible\_move\_col):

          if self.squares[possible\_move\_row][possible\_move\_col].has\_enemy\_piece(piece.color):

            # create initial and final move squares

            initial = Square(row, col)

            final\_piece = self.squares[possible\_move\_row][possible\_move\_col].piece

            final= Square(possible\_move\_row, possible\_move\_col, final\_piece)

            # create a new move

            move = Move(initial,final)

            # check potencial checks

            if bool:

              if not self.in\_check(piece, move):

                #append new move

                piece.add\_move(move)

            else:

              #append new move

              piece.add\_move(move)

      # en passant moves

      r = 3 if piece.color == "white" else 4

      fr = 2 if piece.color == "white" else 5

      if Square.in\_range(col - 1) and row == r:

        if self.squares[row][col-1].has\_enemy\_piece(piece.color):

          p = self.squares[row][col-1].piece

          if isinstance(p, Pawn):

            if p.en\_passant:

              # create initial and final move squares

              initial = Square(row, col)

              final= Square(fr, col - 1, p)

              # create a new move

              move = Move(initial,final)

              # check potencial checks

              if bool:

                if not self.in\_check(piece, move):

                  #append new move

                  piece.add\_move(move)

              else:

                #append new move

                piece.add\_move(move)

      if Square.in\_range(col+1) and row == r:

        if self.squares[row][col+1].has\_enemy\_piece(piece.color):

          p = self.squares[row][col+1].piece

          if isinstance(p, Pawn):

            if p.en\_passant:

              # create initial and final move squares

              initial = Square(row, col)

              final= Square(fr, col+1, p)

              # create a new move

              move = Move(initial,final)

              # check potencial checks

              if bool:

                if not self.in\_check(piece, move):

                  #append new move

                  piece.add\_move(move)

              else:

                #append new move

                piece.add\_move(move)

**def knight\_moves():**

      # 8 possive moves

      possible\_moves = [

        (row-2, col+1),

        (row-1, col+2),

        (row+1, col+2),

        (row+2, col+1),

        (row+2, col-1),

        (row+1, col-2),

        (row-1, col-2),

        (row-2, col-1),

      ]

      for possible\_move in possible\_moves:

        possible\_move\_row, possible\_move\_col = possible\_move

        if Square.in\_range(possible\_move\_row, possible\_move\_col):

          if self.squares[possible\_move\_row][possible\_move\_col].isempty\_or\_enemy(piece.color):

            #create squares of new move

            initial = Square(row, col)

            final\_piece = self.squares[possible\_move\_row][possible\_move\_col].piece

            final= Square(possible\_move\_row, possible\_move\_col, final\_piece)

            # create a new move

            move = Move(initial, final)

            # append new valid move

            if bool:

              if not self.in\_check(piece, move):

                #append new move

                piece.add\_move(move)

              else:

                break

            else:

              #append new move

              piece.add\_move(move)

**def straightlinr\_moves(incrs):**

      for incr in incrs:

        row\_incr, col\_incr = incr

        possible\_move\_row = row + row\_incr

        possible\_move\_col = col + col\_incr

        while True:

          if Square.in\_range(possible\_move\_row, possible\_move\_col):

            #create squares of new move

            initial = Square(row, col)

            final\_piece = self.squares[possible\_move\_row][possible\_move\_col].piece

            final= Square(possible\_move\_row, possible\_move\_col, final\_piece)

            # create a new move

            move = Move(initial, final)

            # empty = countinue looping

            if self.squares[possible\_move\_row][possible\_move\_col].isempty():

              if bool:

                if not self.in\_check(piece, move):

                  #append new move

                  piece.add\_move(move)

              else:

                #append new move

                piece.add\_move(move)

            # has enemy piece = add\_move + break

            elif self.squares[possible\_move\_row][possible\_move\_col].has\_enemy\_piece(piece.color):

              if bool:

                if not self.in\_check(piece, move):

                  #append new move

                  piece.add\_move(move)

              else:

                #append new move

                piece.add\_move(move)

              break

            # has team piece = break

            elif self.squares[possible\_move\_row][possible\_move\_col].has\_team\_piece(piece.color):

              break

          else: break

          possible\_move\_row = possible\_move\_row + row\_incr

          possible\_move\_col = possible\_move\_col + col\_incr

**def king\_moves():**

      # Các ô xung quanh vua

      adjs = [

          (row - 1, col), (row - 1, col + 1),

          (row,     col + 1), (row + 1, col + 1),

          (row + 1, col), (row + 1, col - 1),

          (row,     col - 1), (row - 1, col - 1)

      ]

      # Các nước đi thông thường

      for r, c in adjs:

          if Square.in\_range(r, c):

              if self.squares[r][c].isempty\_or\_enemy(piece.color):

                  initial = Square(row, col)

                  final = Square(r, c)

                  move = Move(initial, final)

                  if bool:

                      if not self.in\_check(piece, move):

                          piece.add\_move(move)

                  else:

                      piece.add\_move(move)

      # Kiểm tra castling nếu vua chưa di chuyển

      if not piece.moved:

          # Queen-side castling (bên trái)

          left\_rook = self.squares[row][0].piece

          if isinstance(left\_rook, Rook) and not left\_rook.moved:

              if all(not self.squares[row][c].has\_piece() for c in range(1, 4)):

                  moveK = Move(Square(row, col), Square(row, 2))

                  moveR = Move(Square(row, 0), Square(row, 3))

                  if bool:

                      if not self.in\_check(piece, moveK) and not self.in\_check(left\_rook, moveR):

                          piece.left\_rook = left\_rook

                          piece.add\_move(moveK)

                          left\_rook.add\_move(moveR)

                  else:

                      piece.left\_rook = left\_rook

                      piece.add\_move(moveK)

                      left\_rook.add\_move(moveR)

          # King-side castling (bên phải)

          right\_rook = self.squares[row][7].piece

          if isinstance(right\_rook, Rook) and not right\_rook.moved:

              if all(not self.squares[row][c].has\_piece() for c in range(5, 7)):

                  moveK = Move(Square(row, col), Square(row, 6))

                  moveR = Move(Square(row, 7), Square(row, 5))

                  if bool:

                      if not self.in\_check(piece, moveK) and not self.in\_check(right\_rook, moveR):

                          piece.right\_rook = right\_rook

                          piece.add\_move(moveK)

                          right\_rook.add\_move(moveR)

                  else:

                      piece.right\_rook = right\_rook

                      piece.add\_move(moveK)

                      right\_rook.add\_move(moveR)

    if isinstance(piece ,Pawn):

      pawn\_moves()

    elif isinstance(piece, Knight):

      knight\_moves()

    elif isinstance(piece, Bishop):

      straightlinr\_moves([

        (-1, 1), # up-right

        (-1, -1), # up-left

        (1, 1), # down-right

        (1, -1) # down-left

      ])

    elif isinstance(piece,Rook):

      straightlinr\_moves([

        (-1, 0), # up

        (0 ,1), # right

        (1, 0), # down

        (0, -1) # left

      ])

    elif isinstance(piece, Queen):

      straightlinr\_moves([

        (-1, 1),

        (-1, -1),

        (1, 1),

        (1, -1),

        (-1, 0),

        (0 ,1),

        (1, 0),

        (0, -1)

      ])

    elif isinstance(piece,King):

      king\_moves()

#### Quản lí logic tổng thể của trò chơi (game.py)

import pygame

from const import \*

from board import Board

from config import Config

from square import Square

from selector import Selector

**class Game:**

**def \_\_init\_\_(self):**

    """ Khởi tạo trò chơi với một bàn cờ. """

    self.next\_player = "white"

    self.hovered\_sqr = None

    self.board = Board()

    self.config = Config()

    self.selector = Selector()

    self.game\_over = False  # Thêm trạng thái kết thúc game //

    self.winner = None      # Lưu màu người thắng ('white'/'black')

    self.result = None      # Lưu kết quả ('checkmate'/'stalemate')

**def show\_bg(self, surface):**

    theme = self.config.theme

     # Vẽ viền nâu xung quanh bàn cờ (dày 20px)

    border\_color = (101, 67, 33)  # Màu nâu (Brown)

    border\_rect = pygame.Rect(

        BOARD\_X - 30,  #

        BOARD\_Y - 30,

        COLS \* SQSIZE + 60,

        ROWS \* SQSIZE + 60

    )

    pygame.draw.rect(surface, border\_color, border\_rect)

    """ Hiển thị nền bàn cờ với hai màu ô xen kẽ."""

    for row in range(ROWS):

      for col in range(COLS):

        # Xác định màu sắc ô dựa trên vị trí của nó (ô sáng hoặc tối)

        color = theme.bg.light if (row + col) % 2 == 0 else theme.bg.dark

        # Xác định hình chữ nhật tương ứng với ô cờ

        rect = (col \* SQSIZE + BOARD\_X, row \* SQSIZE + BOARD\_Y, SQSIZE, SQSIZE)

        # Vẽ ô cờ lên màn hình

        pygame.draw.rect(surface, color, rect)

        # Hiển thị số hàng (1-8) bên trái bàn cờ

        if col == 0:

          color = theme.bg.dark if row % 2 == 0 else theme.bg.light

          lbl = self.config.font.render(str(ROWS - row), 1, color)

          lbl\_x = BOARD\_X - 20

          lbl\_y = BOARD\_Y + row \* SQSIZE + (SQSIZE - lbl.get\_height()) // 2

          surface.blit(lbl, (lbl\_x, lbl\_y))

        # Hiển thị số hàng (1-8) bên phải bàn cờ

        if col == 7:

          color = theme.bg.dark if (row + col) % 2 == 0 else theme.bg.light

          lbl = self.config.font.render(str(ROWS - row), 1, color)

          lbl\_x = BOARD\_X + COLS \* SQSIZE + 10

          lbl\_y = BOARD\_Y + row \* SQSIZE + (SQSIZE - lbl.get\_height()) // 2

          surface.blit(lbl, (lbl\_x, lbl\_y))

        # Hiển thị chữ cái cột (A-H) ở hàng dưới cùng

        if row == 7:

          color = theme.bg.dark if (row + col) % 2 == 0 else theme.bg.light

          lbl = self.config.font.render(Square.get\_alphacol(col), 1, color)

          lbl\_x = BOARD\_X + col \* SQSIZE + (SQSIZE - lbl.get\_width()) // 2

          lbl\_y = BOARD\_Y + ROWS \* SQSIZE + 5

          surface.blit(lbl, (lbl\_x, lbl\_y))

        # Hiển thị chữ cái cột (A-H) ở hàng trên cùng

        if row == 0:

          color = theme.bg.dark if (row + col) % 2 == 0 else theme.bg.light

          lbl = self.config.font.render(Square.get\_alphacol(col), 1, color)

          lbl\_x = BOARD\_X + col \* SQSIZE + (SQSIZE - lbl.get\_width()) // 2

          lbl\_y = BOARD\_Y - 25

          surface.blit(lbl, (lbl\_x, lbl\_y))

  """ Hiển thị tất cả quân cờ lên bàn cờ (trừ quân đang được chọn)."""

**def show\_pieces(self, surface):**

    for row in range(ROWS):

        for col in range(COLS):

            square = self.board.squares[row][col]

            if square.has\_piece():

                piece = square.piece

                if self.selector.selecting and piece == self.selector.piece:

                    continue

                piece.set\_texture(size=80)

                img = pygame.image.load(piece.texture)

                img = pygame.transform.scale(img, (80, 160))

                img\_width, img\_height = img.get\_size()

                img\_x = col \* SQSIZE + BOARD\_X + (SQSIZE - img\_width) // 2

                img\_y = row \* SQSIZE + BOARD\_Y + (SQSIZE - img\_height - 128 + 48) // 2

                piece.texture\_rect = pygame.Rect(img\_x, img\_y, img\_width, img\_height)

                surface.blit(img, piece.texture\_rect)

    if self.selector.selecting and self.selector.piece:

      piece = self.selector.piece

      for row in range(ROWS):

          for col in range(COLS):

              if self.board.squares[row][col].piece == piece:

                  piece.set\_texture(size=80)

                  img = pygame.image.load(piece.texture)

                  img = pygame.transform.scale(img, (80, 160))

                  img\_width, img\_height = img.get\_size()

                  img\_x = col \* SQSIZE + BOARD\_X + (SQSIZE - img\_width) // 2

                  img\_y = row \* SQSIZE + BOARD\_Y + (SQSIZE - img\_height - 128 + 48) // 2

                  piece.texture\_rect = pygame.Rect(img\_x, img\_y, img\_width, img\_height)

                  surface.blit(img, piece.texture\_rect)

                  break

  """ Hiển thị nước đi hợp lệ của quân cờ đang được chọn. """

**def show\_moves(self, surface):**

    theme = self.config.theme

    if self.selector.selecting:

        piece = self.selector.piece

        for move in piece.moves:

            # Chọn màu nền cho nước đi (tuỳ theo màu ô bàn cờ)

            color = theme.moves.light if (move.final.row + move.final.col) % 2 == 0 else theme.moves.dark

            rect = ( move.final.col \* SQSIZE + BOARD\_X, move.final.row \* SQSIZE + BOARD\_Y, SQSIZE, SQSIZE)

            pygame.draw.rect(surface, color, rect)

  """ Hiển thị nước đi cuối cùng (tức là quân cờ vừa đi: vị trí bắt đầu và vị trí hiện tạitại) """

**def show\_last\_move(self, surface):**

    theme = self.config.theme

    if self.board.last\_move:

      initial = self.board.last\_move.initial

      final = self.board.last\_move.final

      for pos in [initial, final]:

        color = theme.trace.light if (pos.row + pos.col) % 2 == 0 else theme.trace.dark

        rect = (pos.col \* SQSIZE + BOARD\_X, pos.row \* SQSIZE + BOARD\_Y, SQSIZE, SQSIZE)

        pygame.draw.rect(surface, color, rect)

  """ Hiển thị viền ô cờ khi di chuột qua. """

**def show\_hover(self, surface):**

    if self.hovered\_sqr:

      color = (180, 180, 180)

      rect = (self.hovered\_sqr.col \* SQSIZE + BOARD\_X, self.hovered\_sqr.row \* SQSIZE + BOARD\_Y, SQSIZE, SQSIZE)

      pygame.draw.rect(surface, color, rect, width = 3)

**def next\_turn(self):**

    self.next\_player = "white" if self.next\_player == "black" else "black"

**def set\_hover(self, row, col):**

    if 0 <= row < ROWS and 0 <= col < COLS:

      self.hovered\_sqr = self.board.squares[row][col]

    else:

      self.hovered\_sqr = None

**def change\_theme(self):**

    self.config.change\_theme()

**def play\_sound(self, capture = False):**

    if capture:

      self.config.capture\_sound.play()

    else:

      self.config.move\_sound.play()

**def reset(self):**

    self.\_\_init\_\_()

#### Quản lí nước đi từ vị trí ban đầu đến vị trí đích trên bàn cờ (move.py)

**class Move:**

**def \_\_init\_\_(self, initial, final):**

    self.initial = initial

    self.final = final

**def \_\_str\_\_(self):**

    s = ''

    s += f'({self.initial.col},{self.initial.row})'

    s += f'  -> ({self.initial.col},{self.initial.row})'

    return

**def \_\_eq\_\_(self, other):**

    return self.initial == other.initial and self.final == other.final

#### Quản lí AI đánh cờ với người chơi (ai.py)

import random

import copy

import time

from collections import defaultdict

from move import Move

**class AI:**

**def \_\_init\_\_(self, board):**

        self.board = board

**def evaluate\_board(self):**

        # Bảng giá trị vị trí cho từng loại quân

        piece\_square\_tables = {

            'Pawn': [

                [0,  0,  0,  0,  0,  0,  0,  0],

                [50, 50, 50, 50, 50, 50, 50, 50],

                [10, 10, 20, 30, 30, 20, 10, 10],

                [5,  5, 10, 25, 25, 10,  5,  5],

                [0,  0,  0, 20, 20,  0,  0,  0],

                [5, -5,-10,  0,  0,-10, -5,  5],

                [5, 10, 10,-20,-20, 10, 10,  5],

                [0,  0,  0,  0,  0,  0,  0,  0]

            ],

            'Knight': [

                [-50,-40,-30,-30,-30,-30,-40,-50],

                [-40,-20,  0,  0,  0,  0,-20,-40],

                [-30,  0, 10, 15, 15, 10,  0,-30],

                [-30,  5, 15, 20, 20, 15,  5,-30],

                [-30,  0, 15, 20, 20, 15,  0,-30],

                [-30,  5, 10, 15, 15, 10,  5,-30],

                [-40,-20,  0,  5,  5,  0,-20,-40],

                [-50,-40,-30,-30,-30,-30,-40,-50]

            ],

            'Bishop': [

                [-20,-10,-10,-10,-10,-10,-10,-20],

                [-10,  0,  0,  0,  0,  0,  0,-10],

                [-10,  0,  5, 10, 10,  5,  0,-10],

                [-10,  5,  5, 10, 10,  5,  5,-10],

                [-10,  0, 10, 10, 10, 10,  0,-10],

                [-10, 10, 10, 10, 10, 10, 10,-10],

                [-10,  5,  0,  0,  0,  0,  5,-10],

                [-20,-10,-10,-10,-10,-10,-10,-20]

            ],

            'Rook': [

                [0,  0,  0,  0,  0,  0,  0,  0],

                [5, 10, 10, 10, 10, 10, 10,  5],

                [-5,  0,  0,  0,  0,  0,  0, -5],

                [-5,  0,  0,  0,  0,  0,  0, -5],

                [-5,  0,  0,  0,  0,  0,  0, -5],

                [-5,  0,  0,  0,  0,  0,  0, -5],

                [-5,  0,  0,  0,  0,  0,  0, -5],

                [0,  0,  0,  5,  5,  0,  0,  0]

            ],

            'Queen': [

                [-20,-10,-10, -5, -5,-10,-10,-20],

                [-10,  0,  0,  0,  0,  0,  0,-10],

                [-10,  0,  5,  5,  5,  5,  0,-10],

                [-5,  0,  5,  5,  5,  5,  0, -5],

                [0,  0,  5,  5,  5,  5,  0, -5],

                [-10,  5,  5,  5,  5,  5,  0,-10],

                [-10,  0,  5,  0,  0,  0,  0,-10],

                [-20,-10,-10, -5, -5,-10,-10,-20]

            ],

            'King': [

                [-30,-40,-40,-50,-50,-40,-40,-30],

                [-30,-40,-40,-50,-50,-40,-40,-30],

                [-30,-40,-40,-50,-50,-40,-40,-30],

                [-30,-40,-40,-50,-50,-40,-40,-30],

                [-20,-30,-30,-40,-40,-30,-30,-20],

                [-10,-20,-20,-20,-20,-20,-20,-10],

                [20, 20,  0,  0,  0,  0, 20, 20],

                [20, 30, 10,  0,  0, 10, 30, 20]

            ]

        }

        value\_map = {

            'Pawn': 100,

            'Knight': 320,

            'Bishop': 330,

            'Rook': 500,

            'Queen': 900,

            'King': 20000

        }

        score = 0

        mobility = 0

        for row in range(8):

            for col in range(8):

                piece = self.board.squares[row][col].piece

                if piece:

                    # Giá trị cơ bản

                    value = value\_map.get(piece.\_\_class\_\_.\_\_name\_\_, 0)

                    # Giá trị vị trí

                    if piece.\_\_class\_\_.\_\_name\_\_ in piece\_square\_tables:

                        table = piece\_square\_tables[piece.\_\_class\_\_.\_\_name\_\_]

                        value += table[row][col] if piece.color == 'black' else table[7-row][col]

                    score += value if piece.color == 'black' else -value

                    # Tính toán khả năng di chuyển (mobility)

                    self.board.calc\_moves(piece, row, col, bool=True)

                    mobility += len(piece.moves) \* (1 if piece.color == 'black' else -1)

        # Thêm các yếu tố đánh giá khác

        score += mobility \* 0.1  # Trọng số mobility

        return score

**def get\_all\_moves(self, color):**

        moves = []

        for row in range(8):

            for col in range(8):

                square = self.board.squares[row][col]

                piece = square.piece

                if piece and piece.color == color:

                    self.board.calc\_moves(piece, row, col, bool=True)

                    for move in piece.moves:

                        moves.append((piece, move))

        return moves

**class AI\_Minimax(AI):**

**def \_\_init\_\_(self, board):**

        super().\_\_init\_\_(board)

        # Cấu hình thời gian

        self.time\_limit = 3  # Giới hạn 3 giây cho mỗi nước đi

        self.max\_nodes = 50000  # Giới hạn số node tối đa

        # Cấu hình giai đoạn game

        self.endgame\_threshold = 6  # Số quân tối đa để xem là endgame

        self.midgame\_depth = 3  # Độ sâu midgame

        self.endgame\_depth = 5   # Độ sâu endgame

        # Hệ thống tối ưu

        self.transposition\_table = {}

        self.killer\_moves = defaultdict(list)

        self.nodes\_searched = 0

        self.start\_time = 0

**def is\_endgame(self, board):**

        """Xác định có phải endgame dựa trên số quân còn lại (không tính vua)"""

        piece\_count = 0

        for row in range(8):

            for col in range(8):

                if board.squares[row][col].has\_piece():

                    piece = board.squares[row][col].piece

                    if piece.\_\_class\_\_.\_\_name\_\_ not in ['King']:

                        piece\_count += 1

                        if piece\_count > self.endgame\_threshold:

                            return False

        return True

**def get\_search\_depth(self, is\_endgame):**

        """Điều chỉnh độ sâu theo giai đoạn game"""

        return self.endgame\_depth if is\_endgame else self.midgame\_depth

**def get\_board\_hash(self, board):**

        """Tạo hash cho bàn cờ bao gồm cả giai đoạn game"""

        hash\_str = []

        for row in board.squares:

            for square in row:

                piece = square.piece

                if piece:

                    hash\_str.append(f"{piece.color[0]}{piece.\_\_class\_\_.\_\_name\_\_[0]}")

                else:

                    hash\_str.append("--")

        return "|".join(hash\_str) + f"|{'E' if self.is\_endgame(board) else 'M'}"

**def evaluate\_board(self):**

        """Mở rộng phương thức evaluate\_board của lớp cha"""

        score = super().evaluate\_board()  # Sử dụng đánh giá cơ bản từ lớp cha

        # Bổ sung đánh giá endgame

        if self.is\_endgame(self.board):

            for row in range(8):

                for col in range(8):

                    piece = self.board.squares[row][col].piece

                    if piece and piece.\_\_class\_\_.\_\_name\_\_ == 'King':

                        # Ưu tiên đưa vua vào trung tâm trong endgame

                        center\_dist = max(abs(row-3.5), abs(col-3.5))

                        score += (4 - center\_dist) \* (10 if piece.color == 'black' else -10)

        return score

**def get\_all\_moves(self, color, depth=0):**

        """Mở rộng phương thức get\_all\_moves với move ordering"""

        moves = super().get\_all\_moves(color)  # Lấy tất cả nước đi cơ bản

        # Thêm điểm số cho move ordering

        scored\_moves = []

        for piece, move in moves:

            score = 0

            # Ưu tiên nước bắt quân

            target = self.board.squares[move.final.row][move.final.col].piece

            if target:

                score = 1000 + self.get\_piece\_value(target) - self.get\_piece\_value(piece)

            # Ưu tiên chiếu tướng

            elif getattr(move, 'is\_check', False):

                score = 500

            # Killer moves

            initial\_pos = self.find\_piece\_position(piece, self.board)

            if initial\_pos and (initial\_pos[0], initial\_pos[1], move.final.row, move.final.col) in self.killer\_moves[depth]:

                score += 400

            scored\_moves.append((piece, move, score))

        # Sắp xếp theo điểm số

        scored\_moves.sort(key=lambda x: x[2], reverse=True)

        return [(p, m) for p, m, \_ in scored\_moves]

**def get\_piece\_value(self, piece):**

        """Lấy giá trị quân cờ từ value\_map"""

        value\_map = {

            'Pawn': 100,

            'Knight': 320,

            'Bishop': 330,

            'Rook': 500,

            'Queen': 900,

            'King': 20000

        }

        return value\_map.get(piece.\_\_class\_\_.\_\_name\_\_, 0)

**def find\_piece\_position(self, piece, board):**

        """Tìm vị trí quân cờ trên bàn cờ"""

        for row in range(8):

            for col in range(8):

                if board.squares[row][col].piece == piece:

                    return (row, col)

        return None

**def quiescence(self, board, color, alpha, beta, depth=0):**

        """Tìm kiếm ổn định - chỉ xét nước bắt quân/chiếu"""

        self.nodes\_searched += 1

        # Giới hạn số node và độ sâu

        if self.nodes\_searched > self.max\_nodes or depth > 6:

            return self.evaluate\_board()

        stand\_pat = self.evaluate\_board()

        if color == 'black':

            if stand\_pat >= beta:

                return beta

            alpha = max(alpha, stand\_pat)

        else:

            if stand\_pat <= alpha:

                return alpha

            beta = min(beta, stand\_pat)

        # Chỉ xét các nước đi bắt quân hoặc chiếu

        captures = []

        for row in range(8):

            for col in range(8):

                piece = board.squares[row][col].piece

                if piece and piece.color == color:

                    board.calc\_moves(piece, row, col, bool=True)

                    for move in piece.moves:

                        target = board.squares[move.final.row][move.final.col].piece

                        if target or getattr(move, 'is\_check', False):

                            score = self.get\_piece\_value(target) if target else 0

                            score += 50 if getattr(move, 'is\_check', False) else 0

                            captures.append((piece, move, score))

        # Sắp xếp và giới hạn số nước đi xét

        captures.sort(key=lambda x: x[2], reverse=True)

        captures = captures[:12]  # Chỉ xét 12 nước đi tốt nhất

        for piece, move, \_ in captures:

            copied\_board = copy.deepcopy(board)

            copied\_board.move(piece, move, testing=True)

            score = self.quiescence(copied\_board, 'white' if color == 'black' else 'black', alpha, beta, depth+1)

            if color == 'black':

                if score >= beta:

                    return beta

                alpha = max(alpha, score)

            else:

                if score <= alpha:

                    return alpha

                beta = min(beta, score)

        return alpha if color == 'black' else beta

**def minimax(self, board, color, depth, alpha, beta, current\_depth=0):**

        """Thuật toán minimax với alpha-beta pruning và transposition table"""

        self.nodes\_searched += 1

        # Kiểm tra giới hạn

        if (time.time() - self.start\_time > self.time\_limit or

            self.nodes\_searched > self.max\_nodes):

            raise TimeoutError()

        board\_hash = self.get\_board\_hash(board)

        if board\_hash in self.transposition\_table:

            entry = self.transposition\_table[board\_hash]

            if entry['depth'] >= depth:

                return entry['value']

        is\_endgame = self.is\_endgame(board)

        # Đến độ sâu tối đa thì chuyển sang quiescence

        if depth == 0:

            q\_score = self.quiescence(board, color, alpha, beta)

            return [q\_score, None, None]

        # Lấy các nước đi đã sắp xếp

        moves = self.get\_all\_moves(color, current\_depth)

        best\_move = (None, None)

        best\_value = -float('inf') if color == 'black' else float('inf')

        for piece, move in moves:

            pos = self.find\_piece\_position(piece, board)

            if not pos:

                continue

            copied\_board = copy.deepcopy(board)

            copied\_board.move(piece, move, testing=True)

            eval\_result = self.minimax(

                copied\_board,

                'white' if color == 'black' else 'black',

                depth - 1,

                alpha,

                beta,

                current\_depth + 1

            )

            eval\_score = eval\_result[0]

            if color == 'black':

                if eval\_score > best\_value:

                    best\_value = eval\_score

                    best\_move = (pos, move)

                    # Cập nhật killer move

                    if not board.squares[move.final.row][move.final.col].piece:

                        self.killer\_moves[current\_depth].append((

                            pos[0], pos[1],

                            move.final.row, move.final.col

                        ))

                        if len(self.killer\_moves[current\_depth]) > 2:

                            self.killer\_moves[current\_depth].pop(0)

                alpha = max(alpha, best\_value)

            else:

                if eval\_score < best\_value:

                    best\_value = eval\_score

                    best\_move = (pos, move)

                beta = min(beta, best\_value)

            if beta <= alpha:

                break

        # Lưu vào transposition table

        self.transposition\_table[board\_hash] = {

            'value': [best\_value, best\_move[0], best\_move[1]],

            'depth': depth

        }

        return [best\_value, best\_move[0], best\_move[1]]

**def get\_best\_move(self, color):**

        """Tìm nước đi tốt nhất với iterative deepening"""

        self.start\_time = time.time()

        self.nodes\_searched = 0

        self.transposition\_table.clear()

        best\_move = None

        is\_endgame = self.is\_endgame(self.board)

        max\_depth = self.get\_search\_depth(is\_endgame)

        try:

            for depth in range(1, max\_depth + 1):

                \_, pos, move = self.minimax(

                    self.board,

                    color,

                    depth,

                    -float('inf'),

                    float('inf'),

                    0

                )

                if pos and move:

                    row, col = pos

                    best\_move = (self.board.squares[row][col].piece, move)

                # Thoát sớm nếu hết thời gian hoặc tìm thấy chiếu hết

                if (time.time() - self.start\_time > self.time\_limit \* 0.9 or

                    (best\_move and getattr(best\_move[1], 'is\_checkmate', False))):

                    break

        except TimeoutError:

            pass

        # Log thông tin debug

        print(f"Depth: {depth}, Nodes: {self.nodes\_searched}, "

              f"Time: {time.time()-self.start\_time:.2f}s, "

              f"Stage: {'Endgame' if is\_endgame else 'Midgame'}")

        return best\_move or self.get\_fallback\_move(color)

**def get\_fallback\_move(self, color):**

        """Nước đi dự phòng nếu hết thời gian"""

        moves = super().get\_all\_moves(color)

        return random.choice(moves) if moves else None

#### Quản lí chọn quân cờ và vị trí hiện tại (selcetor.py)

from const import \*

**class Selector:**

**def \_\_init\_\_(self):**

        self.piece = None             # Quân cờ đang được chọn

        self.selecting = False        # Đang trong trạng thái đã chọn quân cờ

        self.initial\_row = None       # Vị trí dòng ban đầu (nơi quân cờ được chọn)

        self.initial\_col = None # Vị trí cột ban đầu (nơi quân cờ được chọn)

**def save\_initial (self, row, col):**

        self.initial\_row = row

        self.initial\_col = col

**def select\_piece (self, piece, rowcol):**

      self.selecting = True

      self.piece = piece

      self.rowcol = rowcol

**def unselect\_piece(self):**

        self.piece = None

        self.selecting = False

        self.initial\_row = None

        self.initial\_col = None

#### Quản lý giao diện, âm thanh và font chữ (config.py)

import pygame

import os

**class Color:**

**def \_\_init\_\_ (self, light, dark):**

    self.light = light

    self.dark = dark

**class Theme:**

**def \_\_init\_\_ (self, light\_bg, dark\_bg, light\_trace, dark\_trace, light\_moves, dark\_moves):**

    self.bg = Color (light\_bg, dark\_bg)

    self.trace = Color (light\_trace, dark\_trace)

    self.moves = Color (light\_moves, dark\_moves)

**class Sound:**

**def \_\_init\_\_(self, path):**

    self.path = path

    self.sound = pygame.mixer.Sound(path)

**def play(self ):**

    pygame.mixer.Sound.play(self.sound)

**class Config:**

**def \_\_init\_\_(self):**

    self.themes = []

    self.\_add\_theme()

    self.idx = 0

    self.theme = self.themes[self.idx]

    self.font = pygame.font.SysFont('monospace', 18, bold = True)

    self.move\_sound = Sound(os.path.join('assets/sounds/move.wav'))

    self.capture\_sound = Sound(os.path.join('assets/sounds/capture.wav'))

**def change\_theme(self):**

    self.idx += 1

    self.idx %= len(self.themes)

    self.theme = self.themes[self.idx]

**def \_add\_theme(self):**

    gray = Theme ((120, 119, 118), (86, 85, 84), (99, 126, 143), (82, 102, 128), '#C86464', '#C84646')

    green = Theme ((234, 235, 200), (119, 154, 88), (244, 247, 116), (172, 195, 51), '#C86464', '#C84646')

    brown = Theme ((235, 209, 166), (165, 117, 88), (245, 234, 100), (209, 185, 59), '#C86464', '#C84646')

    blue = Theme ((229, 228, 200), (60, 95, 135), (123, 187, 227), (43, 119, 191), '#C86464', '#C84646')

    self.themes = [green, brown, blue, gray]

#### Quản lí thông số cố định của game (const.py)

WIDTH = 450

HEIGHT = 450

# SCREEN DIMENSION

SCREEN\_WIDTH = WIDTH + 500

SCREEN\_HEIGHT = HEIGHT + 200

# BROAD DIMENSION

ROWS = 8

COLS = 8

SQSIZE = WIDTH // COLS

BOARD\_WIDTH = SQSIZE \* COLS

BOARD\_HEIGHT = SQSIZE \* ROWS

# BOARD ALIGNMENT (Căn giữa bàn cờ trong cửa sổ)

BOARD\_X = (SCREEN\_WIDTH - BOARD\_WIDTH) // 2

BOARD\_Y = (SCREEN\_HEIGHT - BOARD\_HEIGHT) // 2

# PHẦN 4: TỔNG KẾT

## Ưu và nhược điểm

### Ưu điểm

* Giao diện người dùng thân thiện, dễ sử dụng, được xây dựng bằng thư viện Tkinter và Pygame, phù hợp với nhiều đối tượng người chơi.
* Ứng dụng tốt các kiến thức về lập trình hướng đối tượng (OOP), xử lý sự kiện và quản lý trạng thái trong quá trình xây dựng game.
* Chương trình hoạt động ổn định, có thể chạy trên nhiều nền tảng hệ điều hành khác nhau (Windows, macOS, Linux).
* Là môi trường lý tưởng để rèn luyện khả năng tư duy chiến thuật thông qua trò chơi trí tuệ cờ vua.
* Có chế độ luyện tập hỗ trợ người mới chơi làm quen với luật cờ, các chiến thuật cơ bản thông qua phần hướng dẫn trực quan và bài tập cụ thể.

### Nhược điểm

* Trí tuệ nhân tạo (AI) trong chế độ người với máy còn khá đơn giản, chưa thực sự mạnh. AI hiện tại chỉ áp dụng một số thuật toán cơ bản như đánh giá vị trí và chọn nước đi theo mức độ ưu tiên đơn giản, chưa thể đưa ra những chiến lược sâu sắc hoặc phòng thủ/phản công hiệu quả trong các thế trận phức tạp.
* Chưa tích hợp hệ thống đăng ký/đăng nhập và không có kết nối với máy chủ (server), do đó không thể lưu trữ thông tin người dùng, lịch sử các ván cờ hoặc phân tích thành tích cá nhân. Điều này cũng giới hạn khả năng mở rộng trò chơi theo hướng trực tuyến, thi đấu nhiều người chơi qua mạng.
* Giao diện đồ họa chưa thực sự bắt mắt: Mặc dù đảm bảo tính thân thiện và dễ sử dụng, nhưng giao diện còn khá đơn giản, thiếu các hiệu ứng hình ảnh, âm thanh và animation sinh động để tăng trải nghiệm người chơi.
* Chế độ chơi chưa được đa dạng và phong phú, chương trình chỉ có 2 chế độ chơi là chơi với máy và luyện tập, khó gây ấn tượng với người dùng.

## Hướng cải tiến phát triển trong tương lai

* Nâng cấp trí tuệ nhân tạo (AI**)**: Trong các phiên bản tiếp theo, nhóm sẽ tập trung cải tiến AI bằng cách áp dụng các cải thiện thuật hơn như Minimax có cắt tỉa Alpha-Beta, hoặc thậm chí kết hợp các mô hình học máy để AI có khả năng đánh giá sâu hơn các tình huống và đưa ra chiến lược hiệu quả hơn, mang đến trải nghiệm thách thức hơn cho người chơi.
* Tích hợp hệ thống đăng ký/đăng nhập và lưu trữ dữ liệu trên server: Xây dựng hệ thống xác thực tài khoản người dùng và kết nối với cơ sở dữ liệu từ xa để lưu trữ thông tin người chơi, kết quả ván đấu, lịch sử đối đầu… Đây là bước tiến quan trọng để mở rộng trò chơi theo hướng thi đấu online hoặc tổ chức giải đấu giữa các người chơi.
* Cải thiện giao diện đồ họa và trải nghiệm người dùng (UX/UI): Tăng cường hiệu ứng đồ họa, âm thanh, animation mượt mà khi di chuyển quân cờ, khi chiến thắng hoặc thua ván cờ. Đồng thời cập nhật giao diện theo xu hướng hiện đại để thu hút người dùng và tạo cảm giác chuyên nghiệp hơn khi chơi.
* Phát triển thêm nhiều chế độ chơi: Bổ sung các chế độ như chơi hai người trên cùng máy, chơi online, chế độ giải đố (puzzle), chế độ theo cấp độ từ dễ đến khó, hoặc thử thách theo thời gian. Những chế độ này giúp tăng tính giải trí, giữ chân người chơi lâu hơn.
* Thêm tính năng lưu ván cờ, phát lại và hoàn tác nước đi: Cho phép người chơi lưu ván đấu để xem lại, học hỏi chiến thuật hoặc phân tích điểm sai lầm. Ngoài ra, tính năng "Undo/Redo" cũng sẽ được xem xét bổ sung để người chơi luyện tập tốt hơn.
* Tối ưu hóa hiệu suất và khả năng tương thích: Nâng cấp kiến trúc chương trình để hoạt động hiệu quả hơn, hạn chế lỗi phát sinh, đảm bảo khả năng chạy mượt trên cả máy tính cấu hình thấp hoặc các hệ điều hành khác nhau.

# TÀI LIỆU THAM KHẢO

[1] <https://www.geeksforgeeks.org/pygame-tutorial>

[2] <https://www.geeksforgeeks.org/python-tkinter-tutorial>

[3] <https://www.programiz.com/python-programming>

[4] <https://en.wikipedia.org/wiki/Rules_of_chess>

[5] fromchris, *Pixel Chess*, Newgrounds Art Portal, [Online]. Available: <https://www.newgrounds.com/art/view/fromchris/pixel-chess>

[6] <https://www.geeksforgeeks.org/python-database-tutorial>

[7] <https://www.geeksforgeeks.org/python-oops-concepts/>