ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐỒ ÁN CỜ VUA

Lớp: 21-2

Người thực hiện:

21120499 – Nguyễn Duy Long

21120561 – Bùi Đức Thịnh

21120541 – Hoàng Văn Quốc

Môn học: Phương pháp lập trình hướng đối tượng

2023 - Thành phố Hồ Chí Minh

Phụ lục :

ĐỒ ÁN CỜ VUA

I. Vai trò và phần trăm điểm số

STT	Người thực hiện	Công việc		Phần trăm điểm số
		Cơ bản	Nâng cao	
1	21120499 Nguyễn Duy Long	- Trong chế độ 2 người chơi (2): +Thiết kế quân cờ King, Pawn +Các class khác: Moves, Point, Board, PieceFactory. - Lưu và chơi lại ván cờ trước đó (1) (Chỉ làm trên console không làm phần đồ họa)	- Chế độ chơi với máy(mức trung bình có thể tránh 1 vài nước đi nguy hiểm nhưng chưa dự đoán được khi ăn quân cờ phía đối phương (tự đánh giá))(0.75) - Undo/Redo(0.5)	4.25
2	21120561 Bùi Đức Thịnh	- Thiết kế sơ đồ lớp (1) - Thiết lập nước đi có thể của quân cờ Knight và Queen (1)	- Tạo đồ hoạ cho game với thư viện SFML (0.5) -Tạo hình ảnh bàn cờ và hình ảnh các quân cờ (0.5) -Thao tác thông qua click chuột điều khiển nước đi của các quân cờ (0.5)	3.5
3	21120541 Hoàng Văn Quốc	- Hỗ trợ thiết kế sơ đồ lớp. (1đ) - Thiết kế class quân cờ Piece, Knight, và Bishop. (1đ)	- Dùng thư viện SFML thiết kế thành phần game: + Giao diện người dùng: menu chính, setting, menu trong game. (1đ) + Âm thanh trò chơi: nhạc nền, hiệu ứng âm thanh chọn, di chuyển quân cờ, ăn quân cờ đối phương, chiếu tướng, kết thúc ván game. (0.5đ)	3.5

II. Hướng dẫn chương trình:

File thực thi (Game) và Source (c++) nằm trong thư mục ChessProject

Vào thư mục tên ChessProject chứa tập tin tên là: **Makefile**

Đảm bảo có sẵn trình biên dịch G++ 7.3 windowx64 trở lên, mở terminal tại thư mục hiện

Chạy lệnh "make" tương ứng của trình biên dịch: ví dụ như dung mingw thì chạy: mingw32-make.exe

III. Cách thiết kế:

1. *CƠ BẢN:*

- Quân cờ Piece và cách thức tạo PieceFactory:

Mỗi 1 quân cờ có 2 thuộc tính là

- Int color: -1 khi là quân đen, và 1 là quân trắng, tương ứng với lượt đi

- Enum TypePiece type được tạo trong class Piece, TypePiece sẽ được set mặc định ở mỗi loại tướng có 1 TypePiece khác, nên khi Init chỉ cần chuyền tham số Màu quân cờ, VD: tạo quân Tốt màu đen: Piece* a = new Pawn(-1);

PieaceFactory được ra để khắc phục vấn đề khi tạo quân cờ, nếu bạn muốn tạo cả bàn cờ mà làm như ví dụ trên bạn sẽ phải làm lần lượt 16*2 cho một bàn cờ. PieaceFactory được tạo ra giải quyết điều đó khi chỉ cần truyền 1 tham số duy nhất chính là thứ có thể xác định cả Type và Color của 1 Piece, bằng cách lấy Color*Type. Hàm createPiece(int) trong PieaceFactory sẽ phân tích tham số này và trả về 1 quân cờ tương ứng.

- Cài đặt nước đi cho các quân cờ Piece:

Ở mỗi class Piece sẽ có hàm canMove(Piece* board[8][8], Point start) sẽ trả về danh sách các nước đi có thể của quân cờ tại vị trí bắt đầu trong bàn cờ, kết hợp với hàm canMove(Point start) trong class Board sẽ loại bỏ tiếp các nước đi dẫn đến chiếu tướng và thua dễ dàng. Phải di chuyển quân cờ từ vị trí ban đầu đến vị trí kết thúc có trong danh sách trên thì mới di chuyển được.

- Thiết lập hàm di chuyển: Move() trong Board:

Chỉ cho phép chọn những quân cờ hợp lệ (thuộc canMove()), nếu chọn điểm start và end đúng, sẽ di chuyển quân cờ này trên Board, nếu trường hợp con Pawn đi đến cuối bàn cờ sẽ thực hiện thêm bước biến đổi. thực hiện đảo lượt đi (turn) và thoát ra.

- Thiết lập Bàn cờ Board:

Bàn cờ là thành phần quan trọng nhất của trò chơi, nó là nơi dàn trận, nơi cung cấp logic và hình dung trận đấu, và là môi trường để các Piece tương tác, đặc biệt là Piece* board[8][8]. Ngoài ra Board có cũng có 2 hàm quan trọng nhất trò chơi đó chính là: checkmate(int _turn) và endgame():

+checkmate(int _turn): Kiểm tra người chơi có lượt bằng turn có bị chiếu tướng: khi King của lượt bên _turn thuộc những vị trí nguy hiểm được trả về trong hàm vector<Point> canLosePiece(int color) là những vị trí phía quân địch có thể tấn công được vào dựa trên hàm. Hàm này còn trực tiếp ứng dụng đến canMove(Point start) để chọn ra nước đi hợp lệ.

+endgame(): 1 trường hợp mặc định lúc khởi tạo game(bên nào mất vua bên đó thua). Ngoài ra sẽ phân thành 2 trường hợp:

- 1 Thắng, 1 Thua: khi bị chiếu, và không có bất kỳ nước đi nào để đi.
- 2 bên Hòa: khi một bên bị không bị chiếu, và cũng không thể đi bất kỳ nước đi nào.

- Thiết lập lưu trò chơi:

+ Trò chơi sẽ lưu tất cả thông tin của bàn cờ xuống file, như loại bàn cờ: với người hay máy. Với trạng thái bàn cờ hiện tại(lượt đi, trạng thái win, ma trận tham số quân cờ như ở PieceFactory), đồng thời lưu các lượt Di chuyển(cấu trúc Moves lưu các bước đi, để phục vụ undo, redo) trong trận đấu xuống File "savegame.txt".

2. NÂNG CAO:

*Các chức năng cho game logic trong game:

- **Undo, Redo:** trong Board, có thêm vector<Moves> lưu các nước đi:

+ Moves chứa thông tin của 1 nước đi bao gồm điểm bắt đầu, kết thúc, và quân cờ tại ô kết thúc.

+ AfterMoves cho Redo, và BeforeMoves cho undo. Khi thực hiện hàm **Move():** của board, nếu di chuyển thành công trong sẽ tạo ra bước di chuyển mới và đẩy nó vào cuối beforeMoves. Nếu lúc này bên phía afterMove có số lượng lơn hơn 0 (tức đã nhấn Undo rồi), thì sẽ clear() đi vector này. Trường hợp di chuyển tốt xuống cuối sẽ được lưu thành 1 nước đi, với (điểm start = end, và giá trị tại end là quân con Tốt biến đổi thành.

+Undo: được thiết kế ngược lại hàm Move() và, cấp phát lại quân tốt nếu nó di chuyển xuống cuối, sẽ chuyển đổi con Tướng được phong đó ngược về thành Tốt.

+Redo được thiết kế giống với Move() nhưng ko clear đi afterMove() mà nó sẽ lấy phần tử cuối trong afterMoves đẩy ngược lại beforeMoves, giống như cơ chế stack vậy. và trong redo sẽ không cho phép chọn tướng phong cho tốt, mà sẽ sử dụng lựa chọn trước đó của bạn.

- **MoveMachine()** random ra nước đi trong trường hợp chơi với máy: Thiết kế giống Move() nhưng cách chọn điểm: start, và end sẽ được tách riêng ra sử dụng hàm: Void **Find(Point &start, Point &end)**.

Hàm **Find(Point &start, Point &end)** trong Board: được chia làm 3 phần chính là: **Khảo sát bàn cờ**, tìm quân có thể ưu tiên lớn hơn và **thực hiện trao đổi cờ** kết thúc hàm, Nếu ở lượt 2 không thành công sẽ vào Phần 3 là **Tìm điểm bắt đầu**, phần 4 là **tìm điểm kết thúc**.

+ Khảo sát: tìm những vị trí nguy hiểm và những vị trí, bên lượt đi có thể đi tới, và cách thức đi đến đó đồng thời tạo ma trận ưu tiên dựa trên float a[8][8] một mặt nào đó đo lường được độ nguy hiểm tại các vị trí(càng nhỏ càng nguy hiểm, càng lớn sẽ càng an toàn).

+**Có thể trao đổi cờ:** dựa trên khảo sát phần 1 ta biết được những vị trí có thể đi tới, thông qua vector canGet, dựa vào đó ta ưu tiên lấy được con Cờ có hệ số TypePiece lớn nhất (end), và tìm cách để đi đến vị trí này, nếu điểu quân cờ trao đổi(start) có hệ số thấp hơn, hoặc nước đi này thỏa mãn điều kiện để trao đổi sẽ truyền tham sô đó vào start, end và kết thúc.

+**Tìm điểm bắt đầu:** Khi phần 2 không thành công chương trình sẽ rời tiếp xuống mục 3: dựa trên khảo sát lượt 1 ta tìm những quân cờ bắt đầu khả năng cao bị uy hiếp **shouldRun** dựa trên a[i][j] <0

- Khi số lương **shouldRun** > 0 ta chọn quân được ưu tiên cao trong số này và tìm ra list nước đi của nó: **canM**, ta bắt đầu sử lý canM bằng cách xóa đi những vị trí Nguy hiểm trong canLose: Nếu bị uy hiếp và không thể chạy được nữa, ta có lẽ chấp nhận bỏ con đó và ưu tiên tìm lại nước đi phù hợp khác(trường hợp mặc định tìm vị trí min), Nếu bị uy hiếp, nhưng canM vẫn còn nước đi an toàn ta sẽ rơi xuống phần 4 là **Tìm điểm kết thúc**
- Khi số lương **shouldRun** <= 0, tức ta không có quân cờ nào trong tầm chiến đấu với Địch, ta sẽ ưu tiên chọn ra quân nhỏ nhất ví dụ như tốt để tiến công.

+**Tìm vị trí kết thúc**: ta sẽ chia làm 2 trường hợp

- Nếu canM của start có vị trí ưu tiên cao nhất không nằm trong khả bị ăn hoặc vị trí chấp nhân trao đổi, trả về vị trí đó.
- Nếu không trả về vị trí trả về vị trí ưu tiên cao nhất(tức chấp nhận trao đổi hoặc mất quân)

=> Đánh giá mức độ hoàn thành chơi với máy: có thể "Random chọn" 1 nước đi, chưa thiết kế gì đến khả năng quay lui hay dự đoán trước 1 nước đi nào. Thuật toán có thể tránh né vài lần quân cờ gặp nguy hiểm, hoặc ăn những quân cờ có hệ số lớn hơn, nhưng thiếu trường hợp khi điểm kết thúc là tướng phe địch thì chưa khảo sát được ở đó có nguy hiểm.

Với việc sử dụng thư viện SFML, ở mỗi class Piece sẽ có hàm initTexturePiece(int color) và hàm initSpritePiece() để tải hình ảnh quân cờ tương ứng theo màu, sau đó sẽ đặt vị trí của chúng tương ứng trên bàn cờ .Mỗi quân cờ (Piece) khi được tạo ra sẽ có thêm phần hình ảnh tương ứng. Việc tải hình ảnh bàn cờ, hình nền, các nút bấm khác tương tự như vậy, Sau đó ở hàm draw(GameManager gameMan) ở class GameState sẽ vẽ chúng lên màn hình.

1. Thao tác chơi game thông qua click chuột

Toạ độ của chuột khi ta di chuyển sẽ được cập nhật liên tục bằng hàm updateMousePosition() ở class GameState.

Việc xử lý các sự kiện chơi game với chuột sẽ được xử lý thông qua hàm pollEvents(GameManager* gameMan) ở class GameState. Trong hàm này xử lý nếu người chơi click chuột vào tọa độ có quân cờ theo đúng lượt của mình thì sẽ lưu vị trí bắt đầu tại đây, đồng thời dựa vào danh sách các nước đi có thể của quân cờ vừa chọn, sẽ thực hiện hàm hiển thị trên màn hình các nước đi đó, khi người chơi kéo hoặc click vào vị trí muốn đến và thả chuột, lúc này hàm Move(sf::Vector2f mousePosView) trong class Board sẽ được gọi và thực hiện kiểm tra xem điểm bắt đầu và kết thúc của quân cờ đó có đi được không để có thể di chuyển được hay không. Nếu đi được sẽ tiến hành chuyển lượt sang bên đối thủ

Tương tự việc xử lý chuột khi click vào toạ độ các nút khác như undo, redo, save, phong cấp cho quân Tốt… sẽ thực hiện các chức năng tương ứng

Ở chế độ chơi với máy, khi đến lượt của bên người chơi, người chơi sẽ thực hiện thao tác với chuột để di chuyển, sau khi hoàn thành lượt của mình, bên máy sẽ tự động thực hiện các nước đi đã được cài đặt trước thông qua hàm MoveMachine() trong class Board

Khi kết thúc game, lúc này người chơi không thể thực hiện các thao tác đối với quân cờ được nữa, nếu người chơi click chuột vào nút Undo, chương trình sẽ vào trạng thái replay (tự động đưa bàn cờ về trạng thái lúc ban đầu chơi), người chơi click chuột vào các nút Undo và Redo để xem lại sự di chuyển của các quân cờ.

2. Giao diện người dùng:

❖ *Ý tưởng:* Sử dụng design pattern State để cài đặt thiết kế giao diện cho game: GameManger là class chứa một cửa sổ render chính và điều phối, quản lý các trạng thái của game: trạng thái Main Menu, trạng thái Setting và ở trong game (GameState) được concreate từ interface WindowState. Các trạng thái có các hành động giống nhau tác động lên GameManager nhưng với hành vi khác nhau bao gồm: init(GameManager*), pollEvents(GameManager*), update(GameManager*), draw(GameManager*).

a. Các thức hoạt động

Khi bắt đầu game, một object GamaManager được khởi tạo với các thông số mặc định, cửa sổ render, cùng với đó là hai State của GamaManager được khởi tạo là MainMenu và Setting. GameManger sẽ đặt trạng thái hiện tại là Main Menu để cho người dùng có thể tùy chọn các lựa chọn trong menu chính như: New game, load game, setting, exit.

Với trạng thái setting, có các lựa chọn tùy chỉnh âm thanh nhạc nền và âm thanh hiệu, nút quay về menu chính ứng thông qua click chuột.

Trạng thái trong game là nơi thể hiện bàn cờ và thao tác trên đó, đồng thời cũng có thanh menu để thực hiện các chức năng: lưu game hiện tại, trở về menu, undo, redo, thể hiện lượt chơi.

b. Cách thức cài đặt chức năng

GameManager dùng trạng thái hiện tại thực hiện vòng lặp các phương thức chính bao gồm pollEvents, update, draw và display. Bắt đầu mỗi vòng duyệt, GameManager xóa nội dung tại màn hình render, sau đó thực hiện tiếp nhận dữ liệu đầu vào từ người dùng bằng phương thức pollEvents, sau khi đã có dữ liệu thì update và tiến hành vẽ (draw()) lên cửa sổ render và hiển thị.

- Chức năng tại Menu chính: Hiển thị lựa chọn cho người chơi bao gồm bắt đầu game mới, cài đặt, load game, thoát, resume (nếu đang chơi mà thoát ra menu chính). Các lựa chọn đó là các nút bấm được tạo từ class Button, chúng sẽ lắng nghe và mỗi khi được click thì sẽ thể hiện các hành vi tương ứng với tên gọi.
  - New game: nếu người dùng ấn nút này thì trạng trái dùng hàm pollEvents lấy ra được lựa chọn này và ra tín hiệu cho Game Manager mở popup lựa chọn chế độ chơi và bắt đầu game mới bằng cách tạo một instance kiểu GameState, sau đó tự nó init và Game Manager sẽ chuyển trạng thái hiện tại sang trạng thái trong game và bắt đầu trò chơi.
  - Load game: Nếu không có dữ liệu đã lưu trước đó thì không có phản ứng gì, còn không sẽ tạo một instance kiểu GameState và tải dữ liệu game đã lưu trước đó rồi chuyển trạng thái.
  - Setting: lập tức chuyển sang trạng thái setting.
  - Exit lập tức thoát chương trình.
- Chức năng trong setting: cho phép người dùng thay đổi hai loại âm thanh tách biệt là nhạc nền và nhạc hiệu ứng bằng việc tinh chỉnh độ dài của hai thanh âm lượng.

Thực chất thanh âm lượng chính là object kiểu Button không có title và được cài màu nền với độ đài có thể tinh chỉnh. Do button có hàm isMoveOver(): bool cho biết mỗi lần click chuột thì có phải chuột nằm trong nó hay không, điều này thuận tiện cho việc lấy được dữ liệu từ người dùng từ đó thay đổi độ dài của thanh âm lượng về mặt trực quan, và thay đổi giá trị âm lượng do GameManager nắm giữ.

- Chức năng của GameState: phần chính của game nằm ở đây. Khi GameManager chuyển sang trạng thái này thì vẫn như trước đó, nó lặp liên tục các hàm chung của 3 trạng thái. Phần tiêu điểm nhất chính là pollEvents và update, pollEvents nhận dữ liệu từ người dùng (click, drag mouse) và thực hiện các chức năng chính của game (di chuyển quân cờ, về menu chính, undo, redo, lưu game).

- Đối với chức năng thực hiện trên bàn cờ thì đã được mô tả rõ ràng ở mục 1 của phần nâng cao.
- Chức năng của các nút bên thanh công cụ được cài đặt bằng class Button nên các thức hoạt động giống với các nút ở State khác, chỉ khác chỗ có Sprite là icon của nút ấn thay vì là chữ hoặc background màu.

IV. Tài liệu tham khảo :

Thư viện đồ hoạ: SFML

https://www.sfml-dev.org/tutorials/2.5/