**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and yellow logo

AI-generated content may be incorrect.

**BÁO CÁO CÔNG NGHỆ JAVA**

**Đề Tài Game Tetris**

**NĂM HỌC** **2024-2025**

**Giáo viên bộ môn:** Nguyễn Trọng Phúc

**Nhóm sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| TRẦN HOÀNG VIỆT | 231230961 | CNTT |
| ĐỖ ĐỨC MINH QUANG | 231230876 | CNTT |

**Hà Nội, tháng 5 năm 2025**

A

A

A

A

A

A

A

**Mục lục**

[**Mục lục** 2](#_Toc197629370)

[**1.Giới thiệu đề tài** 4](#_Toc197629371)

[1.1.Mô tả ngắn gọn về Tetris: 4](#_Toc197629372)

[1.2.Lý do chọn Tetris để triển khai 4](#_Toc197629373)

[1.3. Mục tiêu của đề tài 4](#_Toc197629374)

[**2. Phân tích hệ thống** 5](#_Toc197629375)

[2.1 Yêu cầu chức năng: 5](#_Toc197629376)

[2.2 Yêu cầu phi chức năng: 5](#_Toc197629377)

[**3. Thiết kế hệ thống** 5](#_Toc197629378)

[3.1 Cấu trúc thư mục và chia module 5](#_Toc197629379)

[3.2 Phân tích các lớp và chức năng chính 6](#_Toc197629380)

[3.3. Luồng hoạt động của chương trình 8](#_Toc197629381)

[3.4.Thiết kế thuật toán chính 10](#_Toc197629382)

[3.5. Công nghệ sử dụng 10](#_Toc197629383)

[**4. Kiểm thử và đánh giá:** 11](#_Toc197629384)

[**5. Hướng phát triển** 11](#_Toc197629385)

[**6. Kết luận** 12](#_Toc197629386)

[**7.Tài liệu tham khảo** 12](#_Toc197629387)

**MỞ ĐẦU**

Trong kỷ nguyên công nghệ phát triển mạnh mẽ như hiện nay, ngành công nghiệp phần mềm không ngừng đổi mới và mở rộng trên nhiều lĩnh vực, trong đó lập trình game ngày càng đóng vai trò quan trọng và thu hút sự quan tâm đặc biệt từ các bạn trẻ yêu thích công nghệ. Game không chỉ đơn thuần là công cụ giải trí, mà còn là một môi trường lý tưởng để áp dụng các kiến thức lập trình, tư duy logic, thiết kế giao diện và quản lý hệ thống.

Trò chơi **Tetris**, dù đã ra đời từ những năm 1980, vẫn là một trong những tựa game kinh điển bền bỉ theo thời gian. Với luật chơi đơn giản nhưng đầy thử thách, Tetris trở thành nguồn cảm hứng lý tưởng để nghiên cứu và thực hành lập trình. Thông qua việc xây dựng game Tetris, người học có thể tiếp cận nhiều kiến thức cốt lõi như: xử lý sự kiện, điều khiển luồng game, thao tác đồ họa 2D, và đặc biệt là áp dụng nguyên lý lập trình hướng đối tượng trong Java.

Trong khuôn khổ môn học **Công nghệ Java**, em đã chọn thực hiện đề tài **“Xây dựng game Tetris bằng ngôn ngữ Java”** với mong muốn nâng cao kỹ năng lập trình của bản thân, đồng thời vận dụng hiệu quả những kiến thức đã học vào một sản phẩm thực tế. Đây là cơ hội để em làm quen với quy trình phát triển phần mềm từ khâu phân tích yêu cầu, thiết kế hệ thống, triển khai mã nguồn đến kiểm thử và cải tiến.

Quá trình thực hiện đề tài không tránh khỏi những khó khăn, thử thách, đặc biệt trong việc xử lý logic xoay khối theo chuẩn Super Rotation System (SRS), thiết kế giao diện trực quan bằng Java Swing, và tối ưu hóa hiệu năng khi chơi. Tuy nhiên, nhờ sự hỗ trợ tận tình của thầy **Nguyễn Trọng Phúc** và tài liệu học tập, em đã từng bước hoàn thiện sản phẩm và rút ra được nhiều bài học quý báu.

Em xin chân thành cảm ơn **thầy Nguyễn Trọng Phúc** đã nhiệt tình chỉ dẫn, góp ý và tạo điều kiện thuận lợi để em có thể hoàn thành đề tài này. Đồng thời, em cũng xin gửi lời cảm ơn đến các thầy cô bộ môn và bạn bè đã hỗ trợ em trong suốt quá trình thực hiện báo cáo.

Hy vọng rằng nội dung báo cáo này sẽ phản ánh đầy đủ quá trình nghiên cứu và triển khai của em, đồng thời là minh chứng cho sự nỗ lực và đam mê trong lĩnh vực lập trình game.

**NỘI DUNG BÁO CÁO**

# **1.Giới thiệu đề tài**

## **1.1.Mô tả ngắn gọn về Tetris:**

Tetris là một trò chơi xếp hình cổ điển do Alexey Pajitnov phát triển vào năm 1984. Trò chơi xoay quanh việc điều khiển các khối hình (gọi là Tetromino) rơi từ trên xuống trong một vùng chơi có kích thước giới hạn. Mục tiêu của người chơi là xoay và di chuyển các khối sao cho chúng tạo thành hàng ngang hoàn chỉnh, từ đó làm biến mất hàng đó và ghi điểm. Trò chơi kết thúc khi các khối xếp chồng lên nhau và vượt quá phần trên của vùng chơi.  
Tetris nổi bật bởi luật chơi đơn giản nhưng đòi hỏi người chơi phải có tư duy logic và phản xạ nhanh nhạy, khiến nó trở thành một trong những trò chơi phổ biến và có ảnh hưởng sâu rộng nhất trong lịch sử ngành công nghiệp game.

## **1.2.Lý do chọn Tetris để triển khai**

Tetris là lựa chọn lý tưởng để áp dụng các kiến thức lập trình Java trong môn học Công nghệ Java bởi các lý do sau:

* **Tính kinh điển và quen thuộc**: Dễ tiếp cận với người dùng và phù hợp cho việc thực hành.
* **Cấu trúc rõ ràng**: Dễ triển khai theo hướng đối tượng với các lớp như Board, Tetromino, GameManager.
* **Thử thách hợp lý**: Đòi hỏi xử lý va chạm, xoay khối, cập nhật trạng thái và điểm số liên tục.
* **Dễ mở rộng**: Có thể bổ sung các tính năng nâng cao như Hold, Next Queue, tăng độ khó theo thời gian, lưu điểm cao.  
  Việc lựa chọn Tetris giúp sinh viên vừa rèn luyện kỹ năng lập trình, vừa rèn tư duy thiết kế phần mềm rõ ràng, có cấu trúc.

## **1.3.** **Mục tiêu của đề tài**

Mục tiêu của đề tài là xây dựng một trò chơi Tetris đơn giản, có khả năng chạy độc lập dưới dạng ứng dụng Java. Cụ thể:

* **Xây dựng vùng chơi (Board)** gồm lưới 10x20 ô.
* **Cài đặt các khối Tetromino** và điều khiển xoay theo chuẩn **Super Rotation System (SRS)**.
* **Xử lý logic va chạm**, rơi tự động theo thời gian, và xóa hàng hoàn chỉnh.
* **Quản lý điểm số, cấp độ và số hàng đã xóa**, tăng tốc độ rơi theo cấp độ.
* **Thiết kế giao diện trực quan bằng Java Swing**.
* Từ đó, sinh viên có thể làm quen với mô hình phát triển game đơn giản, đồng thời nắm chắc kiến thức về xử lý sự kiện, vòng lặp trò chơi và lập trình giao diện người dùng trong Java.

# **2. Phân tích hệ thống**

## **2.1 Yêu cầu chức năng:**

* **Tạo bảng chơi (grid):** Hệ thống cần phải có một bảng chơi 2D có thể hiển thị các khối và kiểm tra trạng thái của các khối này.
* **Sinh các khối Tetromino ngẫu nhiên:** Hệ thống sẽ tạo ra các khối Tetromino (các khối vuông, chữ T, L, Z, v.v.) theo cách ngẫu nhiên.
* **Di chuyển, xoay khối, xử lý va chạm:** Khối có thể di chuyển xuống, sang trái, sang phải và xoay. Hệ thống cần kiểm tra và xử lý va chạm với các khối khác hoặc biên của bảng chơi.
* **Xóa dòng, tính điểm**: Khi một dòng trên bảng chơi đầy, dòng đó sẽ bị xóa đi và người chơi sẽ được cộng điểm.
* **Bắt đầu mới, kết thúc game**: Người chơi có bắt đầu game mới và kết thúc game khi cần thiết.

## **2.2 Yêu cầu phi chức năng:**

* Dễ sử dụng
* Giao diện đơn giản
* Chạy mượt trên máy cấu hình thấp

# **3. Thiết kế hệ thống**

## **3.1 Cấu trúc thư mục và chia module**

Dự án được tổ chức thành ba thư mục chính nhằm phân tách rõ ràng các thành phần logic, giao diện và dữ liệu:

**Board/**: Chứa các lớp liên quan đến quản lý lưới chơi và khối Tetromino.

* **Board.java**: Quản lý ma trận lưới, xử lý va chạm, xóa dòng.
* **Tetromino.java**: Đại diện cho các khối Tetris, định nghĩa hình dạng.
* **WallKick.java:** Xử lý offset khi xoay khối theo chuẩn Super Rotation System (SRS).

**GUI/**: Chứa các lớp giao diện thành phần trong trò chơi.

* **HoldPanel.java**: Hiển thị khối được HOLD.
* **NextPanel.java**: Hiển thị các khối sắp tới.
* **ScorePanel.java**: Hiển thị điểm, cấp độ, số dòng đã xóa.

**main/:** Chứa lớp khởi tạo, điều khiển chính và giao diện tổng thể.

* **Main.java**: Điểm bắt đầu của chương trình.
* **App.java**: Thiết lập cửa sổ giao diện chính.
* **TetrisGame.java**: Logic xử lý vòng lặp trò chơi, điều khiển trạng thái và sự kiện.
* **TetrisMenu.java:** Giao diện menu khi bắt đầu game.

**3.2 Phân tích các lớp và chức năng chính**

**📁 Board**

* **Board.java**Chịu trách nhiệm quản lý lưới trò chơi (ma trận), xử lý va chạm, xóa hàng, và cập nhật trạng thái trò chơi sau mỗi khung hình.

**Chức năng chính:**

* + update() – Cập nhật trạng thái của trò chơi theo từng frame.
  + canMove(Tetromino) – Kiểm tra xem khối hiện tại có thể di chuyển đến vị trí mới hay không.
  + lockPiece() – Cố định khối hiện tại xuống lưới khi không thể rơi tiếp.
  + clearLines() – Xóa các hàng đã đầy và cập nhật điểm số/trạng thái tương ứng.
* **Tetromino.java**Đại diện cho các khối (I, O, T, J, L, S, Z). Quản lý vị trí, hình dạng, màu sắc và xoay.  
  Phương thức chính:
  + rotateCW() / rotateCCW() – Xoay khối theo chiều kim/lùi kim đồng hồ.
  + getShape() – Trả về ma trận hình dạng hiện tại.
  + move(x, y) – Di chuyển khối.
* **WallKick.java**Cài đặt quy tắc xoay chuẩn SRS (Super Rotation System) cho các khối Tetris. Khi một khối xoay và gặp va chạm, hệ thống này sẽ thử dịch chuyển nó theo một loạt các offset (dịch chuyển nhỏ) để cho phép xoay hợp lệ mà không bị kẹt.

**Thành phần chính:**

* + JLSTZ\_KICKS:  
    Mảng offset dùng cho các khối J, L, S, T, Z. Mỗi chuyển đổi xoay (ví dụ 0 → R, R → 2, …) có một danh sách offset sẽ thử lần lượt.
  + I\_KICKS:  
    Mảng offset đặc biệt cho khối I, có quy tắc xoay khác biệt hoàn toàn so với các khối còn lại.
  + transitionIndex(from, to):  
    Tính chỉ số chuyển đổi trạng thái xoay từ from sang to một cách an toàn, tránh lỗi vượt chỉ số.
    - getKicks(type, from, to):  
      Trả về danh sách offset phù hợp với loại khối và hướng xoay.  
      Nếu là khối O, không cần dịch chuyển nên trả về {0, 0}.  
      Nếu là khối I, dùng I\_KICKS. Các khối còn lại dùng JLSTZ\_KICKS.

📁 **GUI**  
**Giao diện người dùng** – quản lý việc hiển thị trò chơi như bảng chính, khối kế tiếp (Next), khối đang giữ (Hold), điểm số, cấp độ, và dòng đã xóa.

**🔹 Các thành phần chính:**

* **GamePanel.java :**Khu vực **chính** nơi trò chơi diễn ra – vẽ ma trận, khối hiện tại và khối đang rơi.

**Phương thức chính:**

* + paintComponent(Graphics g) – vẽ toàn bộ trạng thái bảng chơi.
  + drawGrid(g) – vẽ lưới ô vuông.
  + drawPiece(g) – vẽ các khối trên màn hình.
* **NextPanel.java :**Hiển thị các khối **sẽ xuất hiện tiếp theo** (thường 3-5 khối).

**Phương thức chính:**

* + setNextQueue(List<Tetromino>) – cập nhật hàng chờ.
  + paintComponent(Graphics g) – vẽ các khối trong hàng đợi.
* **HoldPanel.java :** Hiển thị khối **đang giữ** (nếu có).

**Phương thức chính:**

* + setHold(Tetromino) – cập nhật khối giữ.
  + paintComponent(Graphics g) – hiển thị khối đó.
* **ScorePanel.java**
* Hiển thị các thông tin phụ:
  + **Điểm (Score)**
  + **Cấp độ (Level)**
  + **Số dòng đã xóa (Lines)**

**📁 main**

* **Main.java**  
  Điểm khởi chạy ứng dụng (main() method).Chỉ có hàm main để khởi tạo code.
* **App.java**: Khởi tạo cửa sổ chính, gắn các panel,dùng Singleton để quản lý cho GameMenu,GamePlay,EndGame.

**Phương thức chính:**

* + startGame(): Tạo và khởi động trò chơi chính.
  + showMenu(): Hiển thị menu.
  + endGame(): Hiển thị màn hình kết thúc game
* **TetrisGame.java**  
  Khởi tạo cửa sổ chính ,quản lý vòng lặp trò chơi,cập nhật trạng thái game và tạo các Score,Next,Hold.  
  Phương thức chính:
  + SetupWindow():: Tạo cửa sổ chính.
  + initComponents():Khởi tạo và gắn các Panel vào layout.
* **TetrisMenu.java**  
  Giao diện menu khởi động, cho phép chọn người chơi bắt đầu game và xem hướng dẫn chơi.

## **3.3. Luồng hoạt động của chương trình**

Quy trình hoạt động của trò chơi Tetris được tổ chức rõ ràng theo từng giai đoạn, kết hợp giữa logic xử lý và hiển thị giao diện người dùng.

**3.3.1. Khởi chạy trò chơi**

* Tập tin Main.java được thực thi đầu tiên và gọi đến phương thức khởi tạo của lớp App.
* App.java tạo cửa sổ chính của ứng dụng (JFrame), thiết lập kích thước, tiêu đề và hiển thị menu (TetrisMenu).
* Trong TetrisMenu, người dùng có thể chọn:
  + Bắt đầu trò chơi mới.
  + Xem hướng dẫn hoặc thoát.

**3.3.2. Bắt đầu trò chơi**

* Khi người chơi chọn “Bắt đầu”, App ẩn TetrisMenu và tạo một phiên chơi mới thông qua lớp TetrisGame.
* TetrisGame chịu trách nhiệm:
  + Tạo bảng chơi (GamePanel) để hiển thị các khối và lưới.
  + Khởi tạo các thành phần giao diện phụ như:
    - NextPanel: hiển thị các khối sắp tới.
    - HoldPanel: hiển thị khối đang giữ.
    - ScorePanel: hiển thị điểm, cấp độ và số dòng đã xóa.
  + Khởi chạy vòng lặp trò chơi bằng Timer hoặc luồng riêng biệt.

**3.3.3. Vòng lặp trò chơi**

Trong mỗi frame, các hành động sau được thực hiện:

* 1. **Xử lý input từ người chơi** (di chuyển, xoay, giữ khối).
  2. **Cập nhật trạng thái trò chơi:**
     + Khối đang rơi (Tetromino) được di chuyển xuống theo thời gian.
     + Nếu gặp va chạm, gọi lockPiece() để cố định khối.
     + Gọi clearLines() nếu có hàng đầy.
     + Cập nhật điểm và cấp độ tương ứng.
  3. **Sinh khối mới:**
     + Lấy từ hàng đợi trong NextPanel.
     + Nếu không thể sinh khối mới (vì va chạm), kết thúc trò chơi.
  4. **Hiển thị lại:**
     + Gọi repaint() trên các panel: GamePanel, NextPanel, HoldPanel, ScorePanel.

**3.3.4. Kết thúc trò chơi**

* Khi trò chơi kết thúc (không còn chỗ sinh khối mới), App gọi giao diện EndGame.

Tại đây, người dùng có thể:

* Quay lại menu chính hoặc chơi lại.
* Thoát khỏi trò chơi.

## **3.4.Thiết kế thuật toán chính**

**🔹 a) Thuật toán kiểm tra va chạm (canMove)**

* Duyệt qua từng ô của khối Tetromino hiện tại.
* Kiểm tra xem ô đó sau khi di chuyển/ xoay có nằm ngoài biên hoặc trùng với ô đã khóa trên Board hay không.
* Nếu có va chạm, trả về false → không di chuyển được.

**🔹 b) Thuật toán cố định khối (lockPiece)**

* Khi khối không thể rơi xuống thêm nữa, các ô của khối sẽ được sao chép vào ma trận chính của Board.
* Sau đó khối mới sẽ được sinh ra.

**🔹 c) Thuật toán xóa hàng (clearLines)**

* Duyệt từng hàng từ dưới lên.
* Nếu một hàng đầy (không còn ô trống), tiến hành xóa hàng.
* Các hàng bên trên được dịch xuống.
* Cập nhật điểm số và số dòng đã xóa.

**🔹 d) Thuật toán xoay chuẩn SRS (WallKick)**

* Khi xoay, nếu có va chạm, thử di chuyển khối theo thứ tự offset định nghĩa trong WallKick.
* Nếu tìm được vị trí hợp lệ, áp dụng xoay và dịch chuyển đó.
* Nếu không tìm được, hủy thao tác xoay.

## **3.5. Công nghệ sử dụng**

* Trong quá trình phát triển trò chơi Tetris, chúng tôi đã sử dụng một số công nghệ và công cụ để đảm bảo hệ thống hoạt động ổn định và hiệu quả. Dưới đây là các công nghệ chính được sử dụng trong dự án:
* **3.5.1 Java Swing**
* Java Swing là thư viện GUI mạnh mẽ của Java, giúp tạo giao diện người dùng một cách dễ dàng và linh hoạt. Trong trò chơi này, Swing được sử dụng để xây dựng giao diện chính của trò chơi, bao gồm bảng chơi, các nút điều khiển, và các thành phần hiển thị như điểm số, cấp độ, và số dòng đã xóa.
* **3.5.2 Java 2D API**
* Để xử lý các yếu tố đồ họa, như vẽ các khối tetrominoes và các hình dạng khác trên bảng, chúng tôi đã sử dụng Java 2D API. Các lớp như Graphics và Graphics2D cho phép vẽ các hình dạng cơ bản và thao tác với đồ họa.
* **3.5.3 Thuật toán SRS (Super Rotation System)**
* Hệ thống xoay SRS (Super Rotation System) được áp dụng để xử lý cơ chế xoay các khối tetromino. SRS đảm bảo rằng việc xoay các khối diễn ra mượt mà và chính xác, tuân thủ các nguyên tắc của trò chơi Tetris.
* **3.5.4 Xử lý sự kiện người dùng**
* Để xử lý các sự kiện người dùng, chúng tôi sử dụng **KeyListener** và **ActionListener**. Các sự kiện này giúp người chơi điều khiển trò chơi bằng cách di chuyển, xoay các khối hoặc tạm dừng, bắt đầu lại trò chơi.

# **4. Kiểm thử và đánh giá:**

Trong quá trình phát triển trò chơi, việc kiểm thử là rất quan trọng để đảm bảo rằng tất cả các chức năng hoạt động như mong đợi. Các phương pháp kiểm thử sẽ bao gồm:

* **Kiểm thử đơn vị (Unit Test)**: Kiểm tra các chức năng riêng biệt như di chuyển khối, xoay khối, xóa dòng đầy và tính toán điểm số.
* **Kiểm thử tích hợp (Integration Test)**: Kiểm tra sự tương tác giữa các thành phần của hệ thống, như việc di chuyển khối và cập nhật điểm số.
* **Kiểm thử giao diện người dùng (UI Test)**: Đảm bảo rằng giao diện người dùng hiển thị chính xác thông tin và cho phép người chơi tương tác một cách dễ dàng.
* **Kiểm thử hiệu suất**: Đánh giá tốc độ của trò chơi khi chạy trên các nền tảng khác nhau và đảm bảo trò chơi không bị giật lag.

# **5. Hướng phát triển**

Để nâng cao chất lượng và tính hấp dẫn của trò chơi, tôi sẽ triển khai các tính năng mới như:

* **Chế độ chơi mới:** Các chế độ như Time Attack hoặc Endless.
* **Cải thiện giao diện người dùng:** Thiết kế lại giao diện và thêm hiệu ứng âm thanh và hình ảnh.
* **Tối ưu hóa hiệu suất:** Giảm độ trễ và cải thiện tốc độ xử lý.
* **Chế độ chơi theo cấp độ:** Thêm các cấp độ khó dần với các yếu tố mới như thay đổi tốc độ rơi của các khối gạch.
* **Cải thiện AI đối thủ:** Thêm đối thủ AI để chơi đối kháng trong các chế độ nhiều người.

# **6. Kết luận**

Dự án Tetris này không chỉ giúp tôi nâng cao kỹ năng lập trình Java, mà còn hiểu rõ hơn về cách sử dụng các cấu trúc dữ liệu và thuật toán trong việc xây dựng một trò chơi. Với những tính năng hiện tại và những hướng phát triển trong tương lai, trò chơi sẽ ngày càng trở nên thú vị và hấp dẫn hơn đối với người chơi.

# **7.Tài liệu tham khảo**

**1."Tetris: From the Soviet Union to Global Phenomenon" by Box Brown (2016).**

**2.**[**https://tetris.fandom.com/wiki/Tetris\_Guideline**](https://tetris.fandom.com/wiki/Tetris_Guideline)**(Trang web Tetris Fandom).**

**3.** [**https://www.geeksforgeeks.org/java-swing/**](https://www.geeksforgeeks.org/java-swing/) **(GeeksforGeeks).**

**4.** [**https://vi.wikipedia.org/wiki/Tetris**](https://vi.wikipedia.org/wiki/Tetris)

**5.** [**https://harddrop.com/wiki/SRS**](https://harddrop.com/wiki/SRS)