BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

A blue and red logo

Description automatically generated**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: CÀI ĐẶT TRÒ CHƠI XẾP HÌNH TETRIS**

**Giảng viên hướng dẫn: THS. PHẠM THỊ KIM NGOAN**

**Sinh viên thực hiện: KIỀU THÁI TUẤN**

**Lớp: 63.CNTT-2**

**MSSV: 63135975**

Khánh Hòa, Tháng 12 năm 2023

**PHẦN NHẬN XÉT VÀ CHO ĐIỂM CỦA GIẢNG VIÊN HƯỚNG DẪN**

**Nhận xét**:

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

**Điểm (tính theo thang điểm 10):** ……………………………

*Khánh Hòa, ngày…….tháng…….năm………* **Giảng viên hướng dẫn** *(Ký và ghi rõ họ tên)*

**PHẦN NHẬN XÉT VÀ CHO ĐIỂM CỦA GIẢNG VIÊN PHẢN BIỆN**

**Nhận xét:**

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

…………………………………………………………………………………………….……

……………………………………………………………………………………….…………

………………………………………………………………………………………………….

………………………………………………………………………………………………….

**Điểm (tính theo thang điểm 10):** ……………………………

*Khánh Hòa, ngày……… tháng……năm………..*

**Giảng viên chấm phản biện**

*(Ký và ghi rõ họ tên)*

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 5](#_Toc155692965)

[PHẦN MỞ ĐẦU 6](#_Toc155692966)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 7](#_Toc155692967)

[1.1. LÝ DO CHỌN ĐỀ TÀI 7](#_Toc155692968)

[1.2. GIỚI THIỆU ĐỀ TÀI 7](#_Toc155692969)

[1.3. MỤC TIÊU CỦA ĐỀ TÀI 7](#_Toc155692970)

[1.4. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 8](#_Toc155692971)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 9](#_Toc155692972)

[2.1. TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG 9](#_Toc155692973)

[2.1.1. Các khái niệm cơ bản trong lập trình hướng đối tượng 9](#_Toc155692974)

[2.1.2. Những đặc điểm của lập trình hướng đối tượng 10](#_Toc155692975)

[2.1.3. Ưu và nhược điểm của lập trình hướng đối tượng 10](#_Toc155692976)

[2.1.4. Những ngôn ngữ lập trình hướng đối tượng phổ biến 11](#_Toc155692977)

[2.2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C# 11](#_Toc155692978)

[2.2.1. Các đặc trưng cơ bản của C# 11](#_Toc155692979)

[2.2.2. Các cấu trúc chương trình C# 12](#_Toc155692980)

[2.2.3. Các cú pháp lệnh cơ bản trong C# 13](#_Toc155692981)

[2.2.4. Các tính chất hướng đối tượng C# 14](#_Toc155692982)

[2.3. TỔNG QUAN VỀ THƯ VIỆN LỚP ĐỒ HỌA WPF 15](#_Toc155692983)

[2.3.1. WPF là gì? 15](#_Toc155692984)

[2.3.2. Công nghệ chung cho giao diện trên Windows và Web 16](#_Toc155692985)

[2.3.3. Các thành phần của WPF 16](#_Toc155692986)

[2.3.4. Khả năng làm việc giữa người thiết kế và lập trình viên 18](#_Toc155692987)

[2.3.5. Công cụ phát triển WPF 18](#_Toc155692988)

[CHƯƠNG 3: CÀI ĐẶT TRÒ CHƠI XẾP HÌNH TETRIS 19](#_Toc155692989)

[3.1. MÔ TẢ CHỈ TIẾT ỨNG DỤNG 19](#_Toc155692990)

[3.2. THIẾT KẾ CÁC LỚP TRONG ỨNG DỤNG 19](#_Toc155692991)

[3.2.1. GameGrid (Lưới trò chơi) 20](#_Toc155692992)

[3.2.2. Position (Vị trí) 22](#_Toc155692993)

[3.2.3. Block (Khối) 23](#_Toc155692994)

[3.2.4. BlockQueue (Khối ngẫu nhiên) 29](#_Toc155692995)

[3.2.5. GameState (Trạng thái trò chơi) 31](#_Toc155692996)

[3.2.6. MainWindow.xaml.cs (Code-Behind) 36](#_Toc155692997)

[3.2.7. MainWindow.xaml 42](#_Toc155692998)

[3.3. MỘT SỐ GIAO DIỆN CỦA ỨNG DỤNG 45](#_Toc155692999)

[3.4. MỘT SỐ ĐOẠN MÃ LỆNH XỬ LÝ CHÍNH TRONG ỨNG DỤNG 48](#_Toc155693000)

[CHƯƠNG 4: KẾT LUẬN 51](#_Toc155693001)

[4.1. ĐÁNH GIÁ VỀ HỆ THỐNG ĐỀ TÀI 51](#_Toc155693002)

[4.2. ƯU VÀ NHƯỢC ĐIỂM CỦA ĐỀ TÀI 51](#_Toc155693003)

[4.3. HƯỚNG PHÁT TRIỂN 52](#_Toc155693004)

[4.4. RÚT RA KINH NGHIỆM ĐỂ PHÁT TRIỂN BẢN THÂN 52](#_Toc155693005)

[TÀI LIỆU THAM KHẢO 53](#_Toc155693006)

# **DANH MỤC HÌNH ẢNH**

[Hình 2.1. Lập trình hướng đối tượng 8](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653400)

[Hình 2.2. Sơ đồ WPF 15](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653401)

[Hình 3.1. Giao diện trò chơi xếp hình Tetris 44](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653402)

[Hình 3.3. Giao diện chơi của chế độ HardMode 45](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653403)

[Hình 3.2. Giao diện chơi của chế độ EasyMode 45](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653404)

[Hình 3.4. Giao diện trò chơi kết thúc của chế độ EasyMode 46](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653405)

[Hình 3.5. Giao diện trò chơi kết thúc của chế độ HardMode 46](file:///C:\Lap%20trinh%20Dai%20Hoc\FileDuLieu\Word\BaoCaoThucTap_KieuThaiTuan_63135975.docx#_Toc155653406)

# **PHẦN MỞ ĐẦU**

Tận dụng sự tiến bộ nhanh chóng của công nghệ thông tin, ta thấy sự bùng nổ của nền tảng lập trình và phát triển ứng dụng. C#, với tính linh hoạt và mạnh mẽ của mình, cùng với Windows Presentation Foundation (WPF), đã mở ra cánh cửa cho việc tạo ra nhiều loại ứng dụng từ di động, web đến desktop.

Trong thế giới game, việc sử dụng C# và WPF để xây dựng một trò chơi Tetris không chỉ là việc áp dụng kiến thức lập trình mà còn là cơ hội để khám phá sức mạnh sáng tạo của hai công nghệ này.

Bắt đầu từ khâu thiết kế giao diện người dùng, sức mạnh của WPF làm nền tảng cho việc tạo ra những đồ họa đẹp mắt và giao diện thân thiện với người dùng. Từ việc vẽ các khối hình đơn giản đến việc xử lý animation phức tạp khi các khối rơi và các hàng hoàn chỉnh biến mất, mỗi chi tiết đều đòi hỏi sự cân nhắc và kỹ năng tạo ra trải nghiệm tuyệt vời.

Ở phía logic game, điều không chỉ dừng lại ở việc di chuyển và xếp các khối hình. Đây là nơi mà sự sáng tạo trong việc áp dụng thuật toán để kiểm soát các khối, xử lý va chạm và tạo ra một cảm giác chơi mượt mà. Điều này đòi hỏi khả năng suy nghĩ logic và tinh thần tối ưu hóa.

Nhìn xa hơn, việc tạo ra một trò chơi Tetris không chỉ là về việc xây dựng mã nguồn mà còn về quá trình học hỏi, sáng tạo và khám phá. Đây là cơ hội để vận dụng kiến thức, thử nghiệm, và tạo ra cái mới trong một môi trường không ngừng biến đổi. Và qua việc này, không chỉ tạo ra một trò chơi, mà còn là hành trình để hiểu sâu hơn về sức mạnh của công nghệ và khả năng tạo ra những trải nghiệm tuyệt vời cho người chơi.

Mặc dù đã cố gắng hoàn thành đề tài này nhưng chắc chắn không tránh khỏi những thiếu xót. Vì vậy, em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của thầy cô. Em xin chân thành cảm ơn cô Phạm Thị Kim Ngoan đã hướng dẫn em hoàn thành đề tài này.

# **CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI**

## **1.1. LÝ DO CHỌN ĐỀ TÀI**

Việc xây dựng một phiên bản game xếp hình Tetris không chỉ đơn thuần là việc tái hiện một trò chơi kinh điển mà còn là một cơ hội để khám phá và phát triển khả năng lập trình. Tính đơn giản và giáo dục của trò chơi này thúc đẩy sự phát triển của tư duy logic và khả năng quản lý thời gian một cách hiệu quả. Bên cạnh đó, việc xây dựng trò chơi xếp hình Tetris cũng mang lại những thử thách kỹ thuật, giúp bản thân củng cố được các kiến thức về lập trình và đồ họa. Điều này không chỉ là một cơ hội để áp dụng kiến thức lập trình mà còn mở ra khả năng sáng tạo trong việc thiết kế giao diện và tối ưu trải nghiệm người dùng.

Trong thời đại mà công nghệ và giáo dục kỹ thuật đang trở nên quan trọng hơn bao giờ hết, việc phát triển một ứng dụng giáo dục và giải trí như game Tetris mang lại nhiều giá trị học thuật và thú vị.

## **1.2. GIỚI THIỆU ĐỀ TÀI**

Đề tài này tập trung vào việc tạo ra trò chơi Tetris sử dụng ngôn ngữ lập trình C# và thư viện giao diện WPF (Windows Presentation Foundation). Bằng cách áp dụng phương pháp lập trình hướng đối tượng và tận dụng sức mạnh của WPF, có thể dễ dàng xây dựng các khối hình đặc trưng như "I", "J", "L", "O", "S", "T", và "Z", và điều khiển chúng thông qua các phím mũi tên hoặc các nút đặc biệt, kèm theo các tính năng logic của trò chơi Tetris.

Trong trò chơi này, người chơi sẽ phải sắp xếp các khối gạch để tạo thành các hàng hoàn chỉnh, từ đó mở ra không gian cho các khối mới và đạt điểm số cao. Sự nhanh nhẹn và kỹ năng định hình là yếu tố then chốt để duy trì không gian trống trên màn hình. Trò chơi chỉ kết thúc khi không còn chỗ trống nào để đặt các khối mới.

Việc kết hợp giữa C# và WPF không chỉ tạo ra hiệu ứng đồ họa cuốn hút, mà còn thiết kế giao diện người dùng trực quan và hấp dẫn người chơi. Đề tài không chỉ giúp hiểu rõ về cách xây dựng trò chơi Tetris mà còn cung cấp cơ hội áp dụng kiến thức lập trình và giao diện người dùng trong một môi trường phát triển hiện đại như C# và WPF.

## **1.3. MỤC TIÊU CỦA ĐỀ TÀI**

- Có thể tạo ra và phát triển một phiên bản game Tetris đầy đủ chức năng, bao gồm các tính năng cơ bản như di chuyển, xoay các khối hình, ghi điểm và xử lý logic chơi game.

- Áp dụng phương pháp lập trình hướng đối tượng bằng việc sử dụng các nguyên tắc lập trình để cấu trúc code một cách rõ ràng và linh hoạt.

- Tối ưu hóa hiệu suất và trải nghiệm người chơi đảm bảo game chạy mượt mà không gặp phải hiện tượng giật lag và có trải nghiệm không tốt cho người dùng.

- Hiểu sâu hơn về lập trình game và giao diện người dùng giúp cho bản thân nắm vững được các khái niệm lập trình game cơ bản và phát triển kỹ năng thiết kế giao diện người dùng trên nền tảng Windows.

## **1.4. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU**

* **Đối tượng**

- Block (Khối hình): Đại diện cho các khối hình trong trò chơi Tetris. Mỗi khối hình có thể được tạo ra từ một đối tượng riêng, với các thuộc tính như hình dạng, vị trí, và hành vi di chuyển.

- GameBoard (Bảng chơi): Là đối tượng biểu diễn bảng chơi trong trò chơi Tetris. Nó chứa thông tin về các khối hình đang tồn tại trên bảng, kiểm soát di chuyển và xử lý va chạm giữa các khối.

- Scoreboard (Bảng điểm): Đối tượng này có thể quản lý và hiển thị thông tin điểm số của người chơi trong trò chơi.

- GameController (Điều khiển trò chơi): Là đối tượng điều khiển luồng của trò chơi, xử lý các sự kiện như di chuyển khối hình, tính điểm, và quản lý trạng thái trò chơi.

* **Phạm vi nghiên cứu**

- Lập trình game: Nghiên cứu và triển khai cách thức xây dựng một game Tetris hoàn chỉnh sử dụng ngôn ngữ lập trình C# và thư viện giao diện WPF.

- Áp dụng kiến thức lập trình và giao diện người dùng: Đây cũng là cơ hội để áp dụng và thực hành kiến thức về lập trình hướng đối tượng, xử lý sự kiện, đồ họa, và quản lý dữ liệu trong môi trường C# và WPF.

- Giao diện người dùng (UI/UX): Tối ưu hóa trải nghiệm người chơi thông qua việc thiết kế giao diện người dùng hấp dẫn và dễ sử dụng. Nghiên cứu về cách thức tạo hiệu ứng, animation, và cách sắp xếp phần giao diện để tạo ra trải nghiệm tốt nhất cho người chơi.

- Tính năng và tối ưu hóa: Nghiên cứu và thực hiện các tính năng bổ sung như lưu điểm, điều chỉnh độ khó của trò chơi, hoặc thay đổi cách khối hình xuất hiện để tăng thêm sự thú vị cho trò chơi. Nghiên cứu về cách tối ưu hóa hiệu suất, đảm bảo trò chơi chạy mượt mà và hiệu quả trên các nền tảng Windows.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1. TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG****~~A group of colorful squares with blue text Description automatically generated~~**

Hình 2.1. Lập trình hướng đối tượng

Lập trình hướng đối tượng (Object Oriented Programming hay OOP) là kỹ thuật lập trình cho phép các lập trình viên tạo ra các đối tượng trong code. Các đối tượng được trừu tượng hóa từ đối tượng thực tế trong đời sống. [1][2]

Hầu hết các ngôn ngữ lập trình hiện nay như Java, PHP, .NET, Ruby, Python… đều hỗ trợ OOP.

### **2.1.1. Các khái niệm cơ bản trong lập trình hướng đối tượng**

* **Đối tượng (object)**

Đối tượng là những sự vật, sự việc có tính chất, đặc tính và hành động giống nhau, đó có thể là con người, điện thoại, máy tính,…

Đặc điểm chung gồm 2 thành phần chính:

- Thuộc tính (Attribute): Là các thông tin, đặc điểm của đối tượng. Ví dụ thuộc tính của máy tính là màu sắc, kích thước, bộ nhớ,…; Hay con người có các đặc điểm như mắt, mũi, tóc, tai, tuổi, sở thích,…

- Phương thức (Method): Là các hành động mà đối tượng có thể thực hiện. Ví dụ: Phương thức của máy tính hoạt động như tắt máy, bật máy, quét virus,…; Phương thức của con người như hành động ăn, nói, đi lại,… [1][2]

* **Lớp (class)**

Các đối tượng có những đặc tính tương tự nhau sẽ được gom lại thành một lớp đối tượng (class). Lớp đối tượng được hiểu là một kiểu dữ liệu, cũng bao gồm 2 thành phần thuộc tính và phương thức. [2]

* **Sự khác biệt của Đối tượng và Lớp**

Lớp là một khuôn mẫu, còn đối tượng là thể hiện cụ thể trạng thái và hành vi dựa trên khuôn mẫu đó. Sự xuất hiện 2 khái niệm đối tượng và lớp chính là đặc trưng của phương pháp lập trình hướng đối tượng. Hình thức lập trình này giải quyết các khuyết điểm của phương pháp lập trình trước (lập trình hướng cấu trúc) để lại. Hai khái niệm này cũng giúp biểu diễn tốt hơn về thế giới thực trên máy tính. [2]

### **2.1.2. Những đặc điểm của lập trình hướng đối tượng**

Để lập trình và thiết kế chương trình theo phương pháp lập trình hướng đối tượng cần hiểu rõ về 4 tính chất: tính đóng gói, tính trừu tượng, tính kế thừa và tính đa hình.

- Tính trừu tượng (Abstraction): lược giản đi những thông tin trong đối tượng, cho phép ta giao tiếp với các thành phần của đối tượng mà không cần biết cách mà các thành phần đó được xây dựng lên. Việc thiết kế đối tượng sẽ được rút tỉa ra những đặc trưng chung. Các đặc trưng đó sẽ được trừu tượng thành các interface và thiết kế xem chúng sẽ tương tác với nhau như thế nào. [2]

- Tính đóng gói (Encapsulation): Các dữ liệu và phương thức có liên quan đến nhau sẽ được đóng gói thành các lớp để tiện sử dụng và quản lý. Mỗi lớp sẽ được xây dựng nhằm thực hiện nhóm chức năng đặc trưng của riêng lớp đó. Bên cạnh đó, tính đóng gói cho phép dấu đi thông tin của đối tượng bằng việc kết hợp thông tin và các phương pháp liên quan đến thông tin cho đối tượng. [2]

- Tính kế thừa (Inheritance): Khi lập trình, ta sẽ thấy có nhiều trường hợp nhiều đối tượng có chung một số thuộc tính và phương thức nhất định, gây vi phạm nguyên tắc lập trình cơ bản. Nhờ có tính kế thừa thì vấn đề này sẽ được giải quyết. Kế thừa trong lập trình hướng đối tượng sẽ thừa hưởng lại những thuộc tính và phương thức của một lớp. [2]

- Tính đa hình (Polymorphism): Đa hình được hiểu là từng hoàn cảnh, từng trường hợp các đối tượng sẽ đóng vai trò khác nhau. Trong lập trình, một đối tượng hay phương thức sẽ có nhiều hơn một hình thái hay là có đa hình. Tính đa hình sẽ hiển thị ở 3 hình thức: Nạp chồng phương thức, Ghi đè phương thức và Thông qua đối tượng đa hình. [2]

### **2.1.3. Ưu và nhược điểm của lập trình hướng đối tượng**

* **Ưu điểm**

- Tính kế thừa giúp quá trình mô tả loại bỏ những chương trình bị lặp, bị dư. Nó giúp mở rộng khả năng sử dụng các lớp mà không cần thực hiện lại, tối ưu và tái sử dụng code hiệu quả. [2]

- Rút ngắn thời gian xây dựng hệ thống, tăng năng suất thực hiện.

- Đối tượng và lớp xuất hiện giúp giải quyết các khuyết điểm của phương pháp lập trình hướng cấu trúc, đồng thời giúp biểu diễn rốt hơn thế giới thực trên máy tính.

* **Nhược điểm**

- OOP có thể khiến cho dữ liệu được xử lý tách rời, đồng nghĩa với việc khi cấu trúc dữ liệu thay đổi sẽ khiến thuật toán thay đổi theo. [1]

- Không tự động khởi tạo, giải phóng dữ liệu động.

- Không mô tả được hệ thống trong thực tế một cách đầy đủ và chân thật.

### **2.1.4. Những ngôn ngữ lập trình hướng đối tượng phổ biến**

Phần lớn các ngôn ngữ lập trình hiện nay đều hỗ trợ lập trình hướng đối tượng. Trong đó, bạn có thể bắt gặp nhiều ngôn ngữ lập trình quen thuộc như:

- Java: Ngôn ngữ lập trình bậc cao được ưa chuộng. Đây là ngôn ngữ lập trình hướng đối tượng quen thuộc trong phát triển ứng dụng web và phần mềm cho các đơn vị.

- Python: Ngôn ngữ này lập trình hướng đối tượng nhờ sự kết hợp giữa khả năng đọc và sự linh hoạt trong kiến tạo các hoạt động khoa học dữ liệu phức tạp.

- C++: Là ngôn ngữ lập trình bậc cao hỗ trợ viết code hướng đối tượng. Con trỏ C++ cho phép thực hiện các tác vụ linh hoạt và dễ dàng. Khái niệm OOP trong C++ cung cấp thêm các tính năng trong thế giới thực cũng như ứng dụng bạn xây dựng.

- Ruby: Toàn bộ giá trị trong Ruby đều được coi là các đối tượng. Code của Ruby thân thiện với lập trình viên. Nhờ có các framework như Ruby mà lập trình viên có thể phát triển ra các ứng dụng web thông qua code có sẵn hiệu quả.

## **2.2. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C#**

C# hay còn được gọi là C sharp, là một ngôn ngữ lập trình hiện đại và dễ tiếp cận, được phát triển bởi Microsoft từ năm 2000, dưới sự chỉ đạo của Anders Hejlsberg và Scott Wiltamuth. Nó được xây dựng trên cơ sở của C++ và Java, hai ngôn ngữ mạnh mẽ. C# hướng đến việc lập trình hướng đối tượng và được thiết kế để hoạt động trên Common Language Infrastructure (CLI). Điều này cho phép sử dụng nó trên nhiều nền tảng và kiến trúc máy tính khác nhau thông qua Executable Code và Runtime Environment, giúp tạo ra các ứng dụng linh hoạt và dễ di động. [4]

### **2.2.1. Các đặc trưng cơ bản của C#**

Các đặc điểm để làm cho C# là ngôn ngữ lập trình chuyên nghiệp được sử dụng rộng rãi:

- Là ngôn ngữ đơn giản dựa trên nền tảng C++ và Java, do đó, nếu quen thuộc với C/C++ hoặc Java, bạn sẽ nhận thấy nó khá quen thuộc về cú pháp, diện mạo và các chức năng. Tuy giữ nguyên một số đặc điểm từ C và C++, nhưng C# đã được cải tiến để trở nên đơn giản hơn. Điều này bao gồm việc loại bỏ các dư thừa và thêm vào một số cú pháp tiện lợi. [4]

- C# Là ngôn ngữ hiện đại bao gồm nhiều khái niệm mới và quan trọng như xử lý ngoại lệ, kiểu dữ liệu mở rộng và bảo mật mã nguồn - những đặc điểm quan trọng của một ngôn ngữ hiện đại.

- Là một ngôn ngữ lập trình thuần hướng đối tượng và phương pháp này có 4 tính chất. Đó là tính trừu tượng (abstraction), tính đóng gói (encapsulation), tính đa hình (polymorphism) và tính kế thừa (inheritance).

- C# là một ngôn ngữ ít từ khóa, sử dụng giới hạn những từ khóa (gồm khoảng 80 từ khóa và mười mấy kiểu dữ liệu xây dựng sẵn).

### **2.2.2. Các cấu trúc chương trình C#**

* **Using**

Dùng để chỉ khai báo thư viện nào sẽ sử dụng trong chương trình (thư viện là tập hợp các phương thức, kiểu dữ liệu có sẵn nào đó được tạo ra nhằm hỗ trợ cho việc lập trình nhanh chóng và hiệu quả hơn). [4]

Cú pháp để khai báo thư viện như sau: using <tên thư viện>;.

* **Namespace**

Dùng để báo cho trình biên dịch biết rằng các thành phần nằm bên trong khối { } ngay dưới namespace thuộc về namespace đó và điều này sẽ giúp ta tránh được những xung đột về code. [4]

Cú pháp của một namespace như sau:

namespace <tên namespace>

{ // các thành phần bên trong name space bao gồm các class, enum.

// delegate hoặc các namespace con }

* **Class**

Cũng tương tự như namespace, class báo cho trình biên dịch biết rằng những thành phần trong khối { } ngay sau tên class thì thuộc vào chính class đó. [4]

Cú pháp một class: class <tên class> { }

* **Hàm (phương thức) main**

Đây là hàm chính của toàn chương trình. Mỗi khi trình biên dịch dịch chương trình sẽ đi vào hàm Main đầu tiên để bắt đầu chương trình. Từ đây ta sẽ viết code bên trong khối { } của hàm Main. [4]

Hàm sẽ được tạo sẵn khi tạo project với cấu trúc như sau:

static void Main(String[] args ) { }

* **Biến trong C#**

Các biến là các thuộc tính hoặc là thành viên dữ liệu của một lớp, được sử dụng để lưu giữ dữ liệu. [3]

* **Dấu chấm phẩy (;)**

Mỗi khi kết thúc một dòng lệnh chúng ta phải viết thêm một dấu “;” ngay sau dòng code để báo cho trình biên dịch biết rằng đã kết thúc dòng lệnh hiện tại. Có thể viết các dòng lệnh trên một hàng, nhưng không nên viết như vậy vì code sẽ rất khó rõ ràng cộng với đó là việc fix bug rất khó khăn. Vậy nên khi viết code phải: Mỗi dòng code là một hàng và các đoạn code con thì để trong khối lệnh { }. [4]

Ngoài ra cũng cần nên lưu ý một số điểm như: C# có phân biệt kiểu chữ, sự thực thi chương trình bắt đầu tại phương thức Main và tên file trong C# có thể khác tên lớp nếu bạn muốn. [4]

### **2.2.3. Các cú pháp lệnh cơ bản trong C#**

* **Lệnh if...else trong C#**

Mệnh đề if...else trong C# được sử dụng để kiểm tra giá trị dạng boolean của điều kiện. Mệnh đề này trả về giá trị True hoặc False. [3]

Cú pháp: if (condition)

{ // khối lệnh được thực thi nếu condition là True }

else

{ // khối lệnh được thực thi nếu condition là False }

C# có các câu lệnh điều kiện sau:

- Sử dụng if để chỉ định một khối lệnh sẽ được thực thi, nếu một điều kiện được chỉ định là True.

- Sử dụng else để chỉ định một khối lệnh sẽ được thực thi, nếu điều kiện tương ứng là False.

- Sử dụng else if để chỉ định kiểm tra một điều kiện mới, nếu điều kiện đầu tiên là False.

- Sử dụng switch để chỉ định nhiều khối lệnh thay thế sẽ được thực thi.

* **Mảng (Array) trong C#**

Thông thường, mảng là một tập hợp các phần tử có cùng kiểu được lưu trữ gần nhau trong bộ nhớ. Mảng (Array) trong C# là một đối tượng chứa các phần tử có kiểu dữ liệu giống nhau. Mảng là một cấu trúc dữ liệu nơi lưu trữ các phần tử giống nhau. Với mảng trong C# chúng ta chỉ có thể lưu trữ một tập các phần tử có số lượng phần tử cố định. Mảng trong C# lưu các phần tử theo chỉ số, chỉ số của phần tử đầu tiên là 0. [3]

Để khai báo một mảng, khai báo loại biến với dấu ngoặc vuông []: string[] cars;

* **Indexer trong C#**

Một indexer trong C# cho phép một đối tượng để được lập chỉ mục, ví dụ như một mảng. Khi định nghĩa một indexer cho một lớp, thì lớp này vận hành tương tự như một mảng ảo (virtual array). Sau đó, có thể truy cập instance (sự thể hiện) của lớp này bởi sử dụng toán tử truy cập mảng trong C# là “[ ]”. [3]

Một Indexer một chiều có cú pháp như sau trong C#:

kiểu\_phần\_tử this[int index]

{ get

{ // trả về giá trị được xác định bởi index }

set

{ // thiết lập giá trị được xác định bởi index }

}

* **Vòng lặp for trong C#**

Vòng lặp for trong C# được sử dụng để lặp một phần của chương trình nhiều lần. Nếu số lần lặp là cố định thì vòng lặp for được khuyến khích sử dụng, còn nếu số lần lặp không cố định thì nên sử dụng vòng lặp while hoặc do while. [3]

Cú pháp: for (khoi\_tao\_bien; check\_dieu\_kien; tang/giam\_bien)

{ // khối lệnh (code\_block) được thực thi }

Cú pháp: foreach (type variableName in arrayName)

{ // khối lệnh (code\_block) được thực thi }

### **2.2.4. Các tính chất hướng đối tượng C#**

* **Tính đóng gói (Encapsulation) trong C#**

Được định nghĩa là "tiến trình đóng gói một hoặc nhiều mục bên trong một gói logic hoặc vật lý". Tính đóng gói, trong phương pháp lập trình hướng đối tượng, ngăn cản việc truy cập tới chi tiết của trình trình triển khai. Tính trừu tượng và tính đóng gói là hai đặc điểm có liên quan với nhau trong lập trình hướng đối tượng. Tính trừu tượng cho phép tạo các thông tin liên quan có thể nhìn thấy và tính đóng gói cho lập trình viên khả năng triển khai độ trừu tượng đã được kế thừa. [3]

Tính đóng gói được triển khai bởi sử dụng bộ xác định quyền truy cập (Access Specifier). Một Access Specifier định nghĩa phạm vi và tính nhìn thấy của một thành viên lớp. C# hỗ trợ các Access Specifier sau: Public, Private, Protected, Internal, Protected internal. [3]

* **Tính kế thừa (Inheritance) trong C#**

Là một trong những khái niệm quan trọng nhất trong lập trình hướng đối tượng C#. Tính kế thừa cho phép định nghĩa một lớp trong điều kiện một lớp khác, mà làm cho nó dễ dàng hơn để tạo và duy trì một ứng dụng. Điều này cũng cung cấp một cơ hội để tái sử dụng tính năng code và thời gian thực thi nhanh hơn.[3]

Khi tạo một lớp, thay vì viết toàn bộ các thành viên dữ liệu và các hàm thành viên mới, lập trình viên có thể nên kế thừa các thành viên của một lớp đang tồn tại. Lớp đang tồn tại này được gọi là lớp cơ sở (Base Class), và lớp mới được xem như là lớp thừa kế (Derived Class). [3]

Cú pháp để tạo lớp kế thừa trong C# là:

<acess-specifier> class <base\_class> { ... }

class <derived\_class> : <base\_class> { ... }

* **Tính đa hình (Polymorphism) trong C#**

Trong lập trình hướng đối tượng, tính đa hình thường được diễn đạt như là "một Interface, nhiều hàm". Tính đa hình trong C# có thể là static hoặc dynamic. Trong đó, kiểu đa hình static có thể được gọi là đa hình tĩnh và kiểu đa hình dynamic có thể được gọi là đa hình động. [3]

- Đa hình static trong C#: Trong đa hình tĩnh, phần phản hồi tới một hàm được xác định tại compile time. Trong khi đó với đa hình động, nó được quyết định tại runtime. Kỹ thuật liên kết một hàm với một đối tượng trong thời gian biên dịch được gọi là Early Binding. Nó cũng được gọi là Static Binding. C# cung cấp hai kỹ thuật để triển khai đa hình tĩnh. Chúng là: Nạp chồng hàm (Function overloading), Nạp chồng toán tử (Operator overloading). [3]

- Đa hình dynamic trong C#: cho phép bạn tạo các lớp abstract (trừu tượng) mà được sử dụng để cung cấp trình triển khai cục bộ lớp của một Interface. Trình triển khai (Implementation) được hoàn thành khi một lớp kế thừa kế thừa từ nó. Các lớp Abstract chứa các phương thức abstract, mà được triển khai bởi lớp kế thừa. Lớp kế thừa này có tính năng chuyên dụng hơn. [3]

Dưới đây là một số qui tắc về các lớp abstract trong C#: Bạn không thể tạo một Instance (sự thể hiện) của một lớp abstract và không thể khai báo một phương thức abstract ở bên ngoài một lớp abstract. Khi một lớp được khai báo là sealed, nó không thể được kế thừa, các lớp abstract không thể được khai báo là sealed. [3]

## **2.3. TỔNG QUAN VỀ THƯ VIỆN LỚP ĐỒ HỌA WPF**

### **2.3.1. WPF là gì?**

WPF, viết tắt của Windows Presentation Foundation, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows. Được xem như thế hệ kế tiếp của WinForms, WPF tăng cường khả năng lập trình giao diện của lập trình viên bằng cách cung cấp các API cho phép tận dụng những lợi thế về đa phương tiện hiện đại. Là một bộ phận của .NET Framework 3.0, WPF sẵn có trong Windows Vista và Windows Server 2008. Đồng thời, WPF cũng có thể hoạt động trên nền Windows XP Service Pack 2 hoặc mới hơn, và cả Windows Server 2003. [5]

WPF được xây dựng nhằm vào ba mục tiêu cơ bản:

- Cung cấp một nền tảng thống nhất để xây dựng giao diện người dùng.

- Cho phép người lập trình và người thiết kế giao diện làm việc cùng nhau một cách dễ dàng.

- Cung cấp một công nghệ chung để xây dựng giao diện người dùng trên cả Windows và trình duyệt Web.

### **2.3.2. Công nghệ chung cho giao diện trên Windows và Web**

Trong thời đại bùng nổ của Internet, các ứng dụng Web ngày một phát triển. Việc trang bị giao diện người dùng với đầy đủ tính năng như một ứng dụng desktop sẽ thu hút nhiều người sử dụng, và do đó góp phần làm tăng giá trị doanh nghiệp. Tuy nhiên, như đã nêu trong phần đầu, với những công nghệ truyền thống, để phát triển một giao diện đồ họa vừa hoạt động trên desktop vừa trên trình duyệt Web, đòi hỏi phải sử dụng những công nghệ hoàn toàn khác nhau, giống như việc xây dựng hai giao diện hoàn toàn độc lập. Điều này tạo ra chi phí không cần thiết để phát triển giao diện. [5]

WPF là một giải pháp cho vấn đề này. Lập trình viên có thể tạo ra một ứng dụng trình duyệt XAML (XBAP) sử dụng WPF chạy trên Internet Explore. Trên thực tế, cùng đoạn code này có thể được dùng để sinh ứng dụng WPF chạy độc lập trên Windows và lợi điểm của việc dùng chung công nghệ cho cả giao diện Windows và giao diện Web là người xây dựng ứng dụng không nhất thiết phải quyết định trước loại giao diện nào được sử dụng. Miễn là máy client đáp ứng được những yêu cầu hệ thống để chạy XBAP, một ứng dụng có thể cung cấp cả giao diện Windows và giao diện Web, mà chỉ sử dụng phần lớn những đoạn mã giống nhau. [5]

Mỗi ứng dụng XBAP được download khi cần từ một Web server, nên nó phải tuân theo những yêu cầu về an ninh khắt khe hơn đối với một ứng dụng Windows độc lập. Theo đó, XBAP chạy trong phạm vi sandbox an ninh do hệ thống an ninh truy nhập mã của .NET Framework cung cấp. XBAP chỉ chạy với các hệ thống Windows có cài đặt WPF và chỉ với Internet Explore phiên bản 6 và 7 trở lên. [5]

### **A screenshot of a computer program Description automatically generated2.3.3. Các thành phần của WPF**

Hình 2.2. Sơ đồ WPF

#### **2.3.3.1. Layout và control**

Ứng dụng WPF sử dụng các panel để sắp xếp thành phần trên giao diện. Mỗi panel có thể chứa nhiều loại control khác nhau như nút bấm, hộp thoại, hoặc các panel khác. Các loại panel khác nhau cho phép sắp xếp các thành phần con theo cách khác nhau. Ví dụ, DockPanel cho phép đặt các thành phần con theo cạnh của panel, trong khi Grid cho phép sắp xếp chúng trên lưới tọa độ. [5]

WPF cung cấp nhiều control như Button, Label, TextBox, ListBox, Menu, Slider, và còn nhiều control phức tạp khác như SpellCheck, PasswordBox… Các control này có thể được đặc tả bằng XAML. Sự kiện như di chuyển chuột hoặc ấn phím có thể được control bắt và xử lý. Tuy nhiên, các sự kiện này phải được xử lý bằng mã trình khi không thể đặc tả hoàn chỉnh bằng XAML. [5]

#### **2.3.3.2. Style và template**

Giống như sử dụng Cascading Style Sheets (CSS) đối với HTML, việc định ra thuộc tính đồ họa cho các đối tượng giao diện một lần, rồi sau đó áp dụng lại cho các đối tượng khác cùng loại thường rất tiện lợi. WPF cũng cung cấp tính năng tương tự bằng việc sử dụng thành phần Style của XAML. Ví dụ, kiểu ButtonStyle có thể được định nghĩa như sau: [5]

[sourcecode language=”html”]

<Style x:Key=”ButtonStyle”>

<Setter Property=”Control.Background” Value=”Red”/>

<Setter Property=”Control.FontSize” Value=”16″/> </Style>

[/sourcecode]

Bất kỳ nút bấm nào sử dụng kiểu này sẽ có nền màu đỏ và sử dụng font chữ kích thước 16. Ví dụ:

[sourcecode language=”html”]

<Button Style=”{StaticResource ButtonStyle}”> Click Here </Button>

[/sourcecode]

Một Style có thể được dẫn xuất từ một Style khác, thừa kế hoặc chồng lên những thuộc tính đã thiết lập. Mỗi style có thể định nghĩa các trigger cho phép tạo ra những hiệu ứng tương tác đặc biệt, chẳng hạn như khi lướt chuột qua nút bấm, nút bấm chuyển thành màu vàng. WPF cũng hỗ trợ sử dụng template. [5]

Mỗi template tương tự như một style, và ở hai dạng:

- Template cho dữ liệu: sử dụng thành phần DataTemplate của XAML để thiết lập một nhóm thuộc tính hiển thị của dữ liệu như màu sắc, phương thức căn lề…

- Template cho control: sử dụng thành phần ControlTemplate của XAML để định ra diện mạo của một control.

#### **2.3.3.3. Text**

Giao diện người dùng ít nhiều đều hiển thị chữ hay text. Đối với phần lớn mọi người, đọc text trên màn hình thường khó hơn đọc trên giấy in. Đó là do chất lượng hiển thị text trên màn hình kém hơn so với khi in ra giấy. WPF tập trung giải quyết vấn đề này, làm chất lượng text hiển thị trên màn hình tương đương trên giấy in. Cụ thể, WPF hỗ trợ các font chữ OpenType chuẩn, cho phép sử dụng các thư viện font đã có.[5]

WPF cũng hỗ trợ công nghệ font chữ mới ClearType, cho phép hiển thị các ký tự mịn hơn đối với mắt người, đặc biệt là trên màn hình tinh thể lỏng (LCD). Để nâng cao hơn nữa chất lượng hiển thị text, WPF cho phép một số công nghệ khác như chữ ghép, theo đó một nhóm ký tự được thay thế bằng một ảnh đơn nhất, tạo tâm lý thoải mái hơn khi đọc đối với người dùng. [5]

### **2.3.4. Khả năng làm việc giữa người thiết kế và lập trình viên**

Xây dựng giao diện người dùng phức tạp đòi hỏi kỹ năng kết hợp giữa thiết kế và lập trình, điều mà không phải tất cả lập trình viên đều có. Sự hợp tác giữa người thiết kế và lập trình viên là quan trọng. Thông thường, người thiết kế tạo ra hình ảnh về giao diện, nhưng việc chuyển đổi thành mã lập trình có thể gặp khó khăn. Hạn chế về công nghệ, áp lực thời gian, thiếu kỹ năng hay hiểu nhầm có thể gây rối. Điều cần làm là tạo cơ chế để hai nhóm làm việc cùng nhau mà vẫn giữ được chất lượng giao diện ban đầu. Để thực hiện được điều này, WPF (Windows Presentation Foundation) sử dụng ngôn ngữ XAML (eXtensible Application Markup Language) để mô tả giao diện người dùng. XAML định nghĩa các phần tử XML như Button, TextBox, Label... để biểu thị các thành phần đồ họa như nút, hộp thoại, nhãn... XAML cho phép mô tả chính xác cấu trúc của giao diện người dùng và các thuộc tính của từng thành phần, giúp thiết lập các tính chất khác nhau của chúng. [5]

XAML có ý nghĩa quan trọng vì nó giúp xây dựng và sử dụng các công cụ tạo và chỉnh sửa đặc tả bằng XML một cách dễ dàng hơn so với việc làm tương tự với đoạn mã. Điều này tạo ra một cách thức hiệu quả để lập trình viên và người thiết kế cùng làm việc với giao diện người dùng. [5]

### **2.3.5. Công cụ phát triển WPF**

WPF không chỉ cung cấp tính linh hoạt và đa dạng cho lập trình viên mà còn đi kèm với các công cụ mạnh mẽ để hỗ trợ cả việc phát triển và thiết kế giao diện.

- Microsoft Visual Studio là công cụ chính dành cho lập trình viên, được tích hợp các thành phần mở rộng cho WPF từ phiên bản 2005. Đến phiên bản 2008, Visual Studio được bổ sung thêm tính năng phát triển ứng dụng WPF, bao gồm Visual Designer - môi trường thiết kế giao diện cho WPF. Điều này cho phép lập trình viên tạo giao diện WPF một cách trực quan và tự động tạo ra các mã XAML tương ứng. [5]

- Microsoft Expression Design, công cụ khác của Microsoft, hướng đến người thiết kế giao diện. Với mục tiêu nâng cao vị thế của người thiết kế, Expression Design cung cấp môi trường làm việc thuận tiện hơn, giúp tạo ra giao diện người dùng theo hướng mỹ thuật, kết hợp với công nghệ cốt lõi XAML của WPF. Điều này cho phép người thiết kế tạo ra các giao diện hấp dẫn và tinh tế mà không cần phải biết nhiều về lập trình. [5]

# **CHƯƠNG 3: CÀI ĐẶT TRÒ CHƠI XẾP HÌNH TETRIS**

## **3.1. MÔ TẢ CHỈ TIẾT ỨNG DỤNG**

Trong trò chơi này, người chơi phải điều khiển các khối hình gạch có hình dạng khác nhau rơi từ phía trên màn hình xuống và xếp chúng thành các hàng hoàn chỉnh. Các khối hình gạch, hay còn gọi là tetrominoes, được tạo thành từ bốn ô vuông nhỏ. Có tổng cộng bảy hình dạng khác nhau của tetrominoes, mỗi hình có một tên gọi riêng như "I", "J", "L", "O", "S", "T", và "Z". Mỗi khối hình có khả năng xoay, di chuyển sang trái, sang phải hoặc rơi nhanh hơn để người chơi có thể sắp xếp chúng theo ý muốn. Điều này được thực hiện dễ dàng thông qua việc sử dụng các phím mũi tên và các phím đặc biệt như C, Z và nút cách trên bàn phím.

Mục tiêu của Tetris là xếp các khối gạch sao cho tạo thành hàng hoàn chỉnh ngang từ viền trên cùng của màn hình xuống dưới. Khi một hàng được hoàn thành, nó sẽ biến mất, tạo ra không gian trống cho các khối tiếp theo. Điểm số tăng lên dựa trên số lượng hàng mà người chơi xóa được cùng một lúc và để làm tăng độ khó, các khối sẽ di chuyển nhanh hơn theo thời gian chơi, tạo thách thức trong quá trình chơi.

Trò chơi sẽ có 2 chế độ chơi và mỗi chế độ tương ứng với một độ khó khác nhau từ dễ cho đến khó (EasyMode và HardMode):

- Chế độ Dễ (EasyMode): Chế độ này thích hợp cho người mới bắt đầu hoặc muốn tập trung vào việc rèn kỹ năng cơ bản. Trò chơi sẽ có tốc độ rơi tetromino chậm hơn và ít thách thức hơn, cho phép người chơi có thêm thời gian để xử lý và xếp hình.

- Chế độ Khó (HardMode): Đây là mức độ cao nhất, thách thức người chơi có trình độ cao và muốn đối mặt với những thử thách khó khăn hơn. Tốc độ rơi của tetromino sẽ nhanh hơn, đòi hỏi người chơi phải có kỹ năng và sự linh hoạt cao để duy trì trò chơi và giữ được màn chơi không bị tràn đầy.

Người chơi cần có sự nhanh nhẹn và kỹ năng định hình để duy trì trạng thái trống trên màn hình và tiếp tục chơi. Trò chơi không kết thúc cho đến khi không còn không gian trống để chứa thêm các khối mới, khi đó màn chơi sẽ bị đầy và kết thúc trò chơi. Người chơi có thể chọn chơi lại nếu muốn, để cố gắng hơn khi họ cảm thấy chưa đạt được mục tiêu mong muốn.

## **3.2. THIẾT KẾ CÁC LỚP TRONG ỨNG DỤNG**

Trong trò chơi Tetris, ta sẽ chơi trên một lưới có kích thước 22 hàng và 10 cột. Trong số này, có 2 hàng ở phía trên cùng được sử dụng để tạo ra các khối mới, nhưng chúng bị ẩn và không hiển thị trong trò chơi. Các hàng được đánh số từ trên xuống dưới, còn cột được đánh số từ trái sang phải, bắt đầu từ ô góc trên cùng bên trái (hàng 0, cột 0). Lưới trò chơi được biểu diễn bằng một mảng hai chiều, trong đó các ô trống có giá trị 0, và các ô có màu sẽ được biểu diễn từ màu xanh lam, xanh biển đậm, cam, vàng, xanh lá cây, tím, cuối cùng là đỏ và mỗi màu sẽ có giá trị tương ứng từ 1 đến 7. Khi khối di chuyển và lắp vào lưới, các ô sẽ nhận giá trị tương ứng với màu của khối, tạo thành cấu trúc và hàng ngang đầy đủ để biến mất và ghi điểm.

### **3.2.1. GameGrid (Lưới trò chơi)**

Lớp GameGrid này sẽ cung cấp các phương thức để kiểm tra và điều chỉnh lưới của trò chơi Tetris, bao gồm việc kiểm tra và xóa các hàng đã đầy đủ, di chuyển các hàng xuống để điền vào các hàng đã bị xóa.

* **Thuộc tính:**

- private readonly int[,] grid: Một mảng hai chiều chứa dữ liệu của lưới, lưu giữ giá trị các ô trên lưới theo hàng và cột.

- public int Rows { get; }: Thuộc tính chỉ đọc, lưu giữ số hàng trong lưới.

- public int Columns { get; }: Thuộc tính chỉ đọc, lưu giữ số cột trong lưới.

A screenshot of a computer code

Description automatically generated- public int this[int r, int c]: Indexer cho phép truy cập và thiết lập giá trị của một ô cụ thể trong lưới thông qua chỉ số hàng “r” và cột “c”.

* **Phương thức:**

**-** public GameGrid(int rows, int columns): Phương thức khởi tạo, tạo ra một lưới với số hàng và số cột được chỉ định, và khởi tạo mảng grid theo kích thước này.

**-** public bool IsInside(int r, int c): Kiểm tra xem một vị trí trong lưới có nằm trong phạm vi hợp lệ không bằng cách so sánh tọa độ “r” với số hàng và tọa độ “c” với số cột, trả về true nếu nằm trong phạm vi, ngược lại trả về false.

**-** public bool IsEmpty(int r, int c): Kiểm tra xem một ô trong lưới có trống không (có giá trị là 0) và nằm trong phạm vi của lưới không bằng cách gọi phương thức IsInside để đảm bảo tọa độ hợp lệ, sau đó kiểm tra giá trị của ô trong mảng grid có bằng 0 hay không.

- public bool IsRowFull(int r): Kiểm tra xem một hàng đã đầy đủ không, tức là tất cả các ô trên hàng đó không trống bằng cách duyệt qua các ô trong hàng “r”, trả về false nếu gặp một ô trống, ngược lại trả về true.

A screenshot of a computer program

Description automatically generated- public bool IsRowEmpty(int r): Kiểm tra xem một hàng có trống không, tức là tất cả các ô trên hàng đó đều trống bằng cách duyệt qua các ô trong hàng “r”, trả về false nếu gặp một ô không trống, ngược lại trả về true.

- private void ClearRow(int r): Xóa toàn bộ nội dung của một hàng bằng cách đặt giá trị của tất cả các ô trên hàng đó về 0 bằng cách duyệt qua từng ô trong hàng “r” và gán giá trị của mỗi ô bằng 0.

- private void MoveRowDow(int r, int numRows): Di chuyển nội dung của một hàng xuống dưới numRows hàng. Các ô trên hàng gốc sẽ được chuyển xuống numRows hàng, với hàng gốc sau khi di chuyển trở thành hàng trống bằng cách duyệt qua từng ô trong hàng “r”, gán giá trị của mỗi ô cho ô cách numRows hàng dưới. Đồng thời, gán giá trị của các ô trong hàng r thành 0.

- public int ClearFullRows(): Xóa các hàng đầy đủ và di chuyển các hàng trống xuống để điền vào khoảng trống bằng cách duyệt qua từng hàng từ dưới lên trên cùng của lưới. Nếu hàng đầy đủ, gọi ClearRow để xóa hàng và tăng biến đếm cleared. Nếu có hàng trống phía trên hàng đã xóa, gọi MoveRowDown để di chuyển các hàng này xuống. Trả về số lượng hàng đã được xóa.

A screenshot of a computer program

Description automatically generated

Sẽ có bảy loại khối màu khác nhau trong Tetris, mỗi khối bao gồm 4 ô vuông. Hầu hết các khối xoay quanh trung tâm của chúng, nhưng khối I quay xung quanh một điểm không phải là trung tâm và khối O không thể xoay.

Mỗi khối sẽ có một lưới giới hạn xung quanh, đủ lớn để chứa cả bốn trạng thái xoay khác nhau. Góc trên cùng bên trái của lưới giới hạn này, được xác định là hàng 0 cột 0 giống như lưới trò chơi, ghi nhận các ô đã được chiếm trong lưới giới hạn này cho mỗi trạng thái xoay: trạng thái 0 là xoay 0 độ, trạng thái 1 là xoay 90 độ, trạng thái 2 là xoay 180 độ và trạng thái 3 là xoay 270 độ. Điều này giúp xác định vị trí của khối trong mỗi trạng thái xoay để phù hợp với không gian trên màn hình. Ngoài việc xoay, người chơi cũng có thể di chuyển khối từ trái sang phải hoặc xuống dưới. Việc này được biểu diễn thông qua việc lưu trữ các Row Offset và Column Offset, cho phép thay đổi offset để di chuyển khối từng ô một, tạo ra cảm giác di chuyển linh hoạt và kiểm soát trong trò chơi.

### **3.2.2. Position (Vị trí)**

Là lớp đại diện cho một vị trí trên lưới của trò chơi Tetris, cung cấp một cách thuận tiện để lưu trữ thông tin về hàng và cột để xác định vị trí chính xác của các khối, các thành phần trong trò chơi.

* **Thuộc tính:**

- Row: Lưu trữ số hàng của vị trí.

- Column: Lưu trữ số cột của vị trí.

* **Phương thức:**

A computer screen shot of a program code

Description automatically generated- Position(int row, int column): Phương thức khởi tạo của lớp, cho phép tạo một đối tượng Position mới với số hàng và số cột được chỉ định.

### **3.2.3. Block (Khối)**

Lớp này định nghĩa các phương thức cơ bản để quản lý việc quay và di chuyển các khối trong trò chơi Tetris, với các phương thức trừu tượng để xác định cách các khối cụ thể thực hiện các hành động này.

* **Thuộc tính:**

- protected abstract Position[][] Tiles: Một mảng hai chiều chứa các vị trí của các viên gạch trong khối. Đây là một thuộc tính trừu tượng, được kế thừa và cung cấp bởi các lớp con.

- protected abstract Position StartOffset: Một vị trí khởi đầu của khối. Tương tự như Tiles, đây cũng là một thuộc tính trừu tượng để định nghĩa vị trí bắt đầu của khối.

- public abstract int Id: Một số nguyên đại diện cho ID của khối. Cũng là một thuộc tính trừu tượng, được cung cấp bởi các lớp con.

- private int rotationState: Biến lưu trữ trạng thái hiện tại của việc quay khối.

A screen shot of a computer code

Description automatically generated- private Position offset: Vị trí hiện tại của khối trong không gian trò chơi.

* **Phương thức:**

- public Block(): Constructor của lớp Block, thiết lập vị trí ban đầu của offset.

- public IEnumerable<Position> TilePositions(): Trả về tất cả các vị trí của các ô trong khối hình dựa trên trạng thái quay hiện tại và vị trí offset, sử dụng yield return để trả về từng vị trí một một cách trực tiếp khi được yêu cầu.

- public void RotateCW(): Quay khối theo chiều kim đồng hồ (CW) bằng cách tăng giá trị rotationState lên một đơn vị và thực hiện việc lặp lại nếu vượt quá độ dài của mảng Tiles.

- public void RotateCCW(): Quay khối ngược chiều kim đồng hồ (CCW) bằng cách giảm giá trị rotationState đi một đơn vị và xử lý trường hợp khi rotationState là 0.

- public void Move(int rows, int columns): Di chuyển khối theo số hàng và cột được chỉ định, thay đổi offset tương ứng.

A screenshot of a computer program

Description automatically generated- public void Reset(): Đặt lại trạng thái của khối về trạng thái ban đầu, bao gồm rotationState và offset.

#### **3.2.3.1. I-Block kế thừa từ lớp Block (ID: 1)**

Lớp này kế thừa từ lớp trừu tượng Block, đại diện cho khối hình chữ "I" (IBlock) trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều của các vị trí (Position) đại diện cho các ô của khối "I" ở các trạng thái quay khác nhau từ 0 đến 3.

* **Phương thức:**

- public override int Id => 1;: Thuộc tính abstract Id được triển khai để trả về giá trị cụ thể của ID cho khối "I".

- protected override Position StartOffset => new Position(-1, 3);: Thuộc tính abstract StartOffset được triển khai để trả về vị trí bắt đầu của khối "I" trong lưới, cụ thể là vị trí hàng -1 và cột 3.

A screenshot of a computer program

Description automatically generated- protected override Position[][] Tiles => tiles;: Thuộc tính abstract Tiles được triển khai để trả về mảng các vị trí đại diện cho các ô của khối "I" ở các trạng thái quay khác nhau.

#### **3.2.3.2. J-Block kế thừa từ lớp Block (ID: 2)**

Lớp JBlock kế thừa từ lớp trừu tượng Block và đại diện cho khối hình chữ "J" (JBlock) trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều của các vị trí (Position) đại diện cho các ô của khối "J" ở các trạng thái quay khác nhau.

* **Phương thức:**

- public override int Id => 2;: Thuộc tính abstract Id được triển khai để trả về giá trị cụ thể của ID cho khối "J".

- protected override Position StartOffset => new Position(0, 3);: Thuộc tính abstract StartOffset được triển khai để trả về vị trí bắt đầu của khối "J" trong lưới, cụ thể là vị trí hàng 0 và cột 3.

A screenshot of a computer

Description automatically generated- protected override Position[][] Tiles => tiles;: Thuộc tính abstract Tiles được triển khai để trả về mảng các vị trí đại diện cho các ô của khối "J" ở các trạng thái quay khác nhau.

#### **3.2.3.3. L-Block kế thừa từ lớp Block (ID: 3)**

Lớp LBlock kế thừa từ lớp trừu tượng Block, đại diện cho khối hình chữ "L" (LBlock) trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều của các vị trí (Position) đại diện cho các ô của khối "L" ở các trạng thái quay khác nhau.

* **Phương thức:**

- public override int Id => 3;: Thuộc tính abstract Id được triển khai để trả về giá trị cụ thể của ID cho khối "L".

- protected override Position StartOffset => new Position(0, 3);: Thuộc tính abstract StartOffset được triển khai để trả về vị trí bắt đầu của khối "L" trong lưới, cụ thể là vị trí hàng 0 và cột 3.

A screenshot of a computer program

Description automatically generated- protected override Position[][] Tiles => tiles;: Thuộc tính abstract Tiles được triển khai để trả về mảng các vị trí đại diện cho các ô của khối "L" ở các trạng thái quay khác nhau.

#### **3.2.3.4. O-Block kế thừa từ lớp Block (ID: 4)**

Lớp OBlock là một lớp đại diện cho khối hình vuông trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều chứa các vị trí (Position) biểu diễn các ô của khối hình vuông.

* **Phương thức:**

- public override int Id => 4;: Triển khai thuộc tính abstract Id, trả về giá trị cụ thể của ID cho khối hình vuông.

- protected override Position StartOffset => new Position(0, 4);: Triển khai thuộc tính abstract StartOffset, cung cấp vị trí bắt đầu của khối hình vuông trong lưới, ở hàng 0 và cột 4.

A screenshot of a computer program

Description automatically generated- protected override Position[][] Tiles => tiles;: Triển khai thuộc tính abstract Tiles, trả về mảng các vị trí biểu diễn các ô của khối hình vuông.

#### **3.2.3.5. S-Block kế thừa từ lớp Block (ID: 5)**

Lớp SBlock là một lớp đại diện cho khối hình "S" trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều chứa các vị trí (Position) biểu diễn các ô của khối "S" ở các trạng thái quay khác nhau.

* **Phương thức:**

- public override int Id => 5;: Triển khai thuộc tính abstract Id, trả về giá trị cụ thể của ID cho khối "S".

- protected override Position StartOffset => new Position(0, 3);: Triển khai thuộc tính abstract StartOffset, cung cấp vị trí bắt đầu của khối "S" trong lưới, ở hàng 0 và cột 3.

- protected override Position[][] Tiles => tiles;: Triển khai thuộc tính abstract Tiles, trả về mảng các vị trí biểu diễn các ô của khối "S" ở các trạng thái quay khác nhau.

#### **3.2.3.6. T-Block kế thừa từ lớp Block (ID: 6)**

Lớp TBlock, định nghĩa một khối hình "T" trong trò chơi Tetris.

* **A screenshot of a computer program

  Description automatically generatedThuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều chứa các vị trí (Position) biểu diễn các ô của khối "T" ở các trạng thái quay khác nhau.

* **Phương thức:**

- public override int Id => 6;: Triển khai thuộc tính abstract Id, trả về giá trị cụ thể của ID cho khối "T".

- protected override Position StartOffset => new Position(0, 3);: Triển khai thuộc tính abstract StartOffset, cung cấp vị trí bắt đầu của khối "T" trong lưới, ở hàng 0 và cột 3.

A screenshot of a computer program

Description automatically generated- protected override Position[][] Tiles => tiles;: Triển khai thuộc tính abstract Tiles, trả về mảng các vị trí biểu diễn các ô của khối "T" ở các trạng thái quay khác nhau.

#### **3.2.3.7. Z-Block kế thừa từ lớp Block (ID: 7)**

Lớp Zblock, định nghĩa một khối hình "Z" trong trò chơi Tetris.

* **Thuộc tính:**

- private readonly Position[][] tiles: Một mảng hai chiều chứa các vị trí (Position) biểu diễn các ô của khối "Z" ở các trạng thái quay khác nhau.

* **Phương thức:**

- public override int Id => 7;: Triển khai thuộc tính abstract Id, trả về giá trị cụ thể của ID cho khối "Z".

- protected override Position StartOffset => new Position(0, 3);: Triển khai thuộc tính abstract StartOffset, cung cấp vị trí bắt đầu của khối "Z" trong lưới, ở hàng 0 và cột 3.

A screenshot of a computer program

Description automatically generated- protected override Position[][] Tiles => tiles;: Triển khai thuộc tính abstract Tiles, trả về mảng các vị trí biểu diễn các ô của khối "Z" ở các trạng thái quay khác nhau.

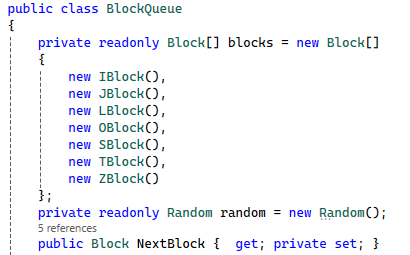
### **3.2.4. BlockQueue (Khối ngẫu nhiên)**

Lớp BlockQueue chứa các khối được sử dụng trong trò chơi Tetris và quản lý việc lấy và cập nhật khối tiếp theo trong hàng đợi.

* **Thuộc tính:**

- Blocks (private readonly Block[] blocks): Mảng chứa các đối tượng Block, bao gồm các loại khối khác nhau trong trò chơi Tetris. Đây là một mảng không thay đổi (readonly) và chỉ có thể được khởi tạo một lần trong constructor của lớp.

- Random (private readonly Random random): Đối tượng của lớp Random được sử dụng để tạo số ngẫu nhiên, cần để chọn ngẫu nhiên một khối từ mảng blocks.

- NextBlock (public Block NextBlock { get; private set; }): Đây là thuộc tính công khai (public) chỉ đọc (read-only) của lớp BlockQueue. Nó lưu trữ khối kế tiếp sẽ được sử dụng trong trò chơi Tetris.

* **Phương thức:**

- BlockQueue(): Constructor của lớp BlockQueue, khởi tạo một đối tượng BlockQueue. Trong constructor này, NextBlock được khởi tạo bằng cách gọi phương thức RandomBlock() để chọn một khối ngẫu nhiên từ mảng blocks.

- RandomBlock() (private Block RandomBlock()): Phương thức này chọn một khối ngẫu nhiên từ mảng blocks và trả về đối tượng Block tương ứng.

A screenshot of a computer program

Description automatically generated- GetAndUpdate() (public Block GetAndUpdate ()): Phương thức này được sử dụng để lấy khối tiếp theo trong hàng đợi và cập nhật NextBlock để chuẩn bị cho khối sau. Nó lấy khối hiện tại, sau đó lặp lại việc chọn ngẫu nhiên một khối mới cho đến khi khối mới được chọn khác với khối hiện tại (theo Id của chúng), sau đó trả về khối hiện tại đã được lấy ra.

### **3.2.5. GameState (Trạng thái trò chơi)**

Lớp GameState trong ứng dụng Tetris là một phần quan trọng của trò chơi, quản lý trạng thái hiện tại của trò chơi và các hành động diễn ra trong trò chơi.

* **Thuộc tính:**

- CurrentBlock (private Block currentBlock): Đây là khối hiện tại đang được sử dụng trong trò chơi.

- CurrentBlock (public Block CurrentBlock { get; private set; }): Đây là một thuộc tính công khai để truy cập và cập nhật khối hiện tại. Khi giá trị của CurrentBlock thay đổi, nó cũng kích hoạt việc thiết lập lại khối và di chuyển khối xuống một vị trí thích hợp trong lưới trò chơi.

- GameGrid (public GameGrid GameGrid { get; }): Đây là lưới trò chơi Tetris, đại diện cho các ô trong trò chơi.

- BlockQueue (public BlockQueue BlockQueue { get; }): Đối tượng hàng đợi khối, quản lý việc lấy khối tiếp theo.

- GameOver (public bool GameOver { get; private set; }): Biến boolean xác định trạng thái kết thúc của trò chơi.

- Score (public int Score { get; private set; }): Điểm số hiện tại trong trò chơi.

- HeldBlock (public Block HeldBlock { get; private set; }): Khối được giữ lại (nếu có).

A screenshot of a computer program

Description automatically generated- CanHold (public bool CanHold { get; private set; }): Biến boolean xác định khả năng giữ một khối mới trong trò chơi.

* **Phương thức:**

- Constructor (public GameState()): Phương thức khởi tạo của lớp GameState. Trong phương thức này, GameGrid và BlockQueue được khởi tạo. Khối hiện tại được thiết lập từ BlockQueue và khả năng giữ khối mới được kích hoạt.

- BlockFits(),:

**Chức năng:** Kiểm tra xem khối hiện tại có thể vừa với lưới trò chơi hay không, tức là kiểm tra xem các ô mà khối đang chiếm có trống không.

**Cách thực hiện:** Duyệt qua từng vị trí của khối hiện tại bằng cách sử dụng phương thức TilePositions() của đối tượng CurrentBlock. TilePositions() trả về danh sách các vị trí tương đối của các ô trong khối. Đối với mỗi vị trí của ô trong khối, kiểm tra xem ô đó có trùng với bất kỳ ô nào đang được sử dụng trên lưới trò chơi không thông qua phương thức IsEmpty() của đối tượng GameGrid. Nếu bất kỳ ô nào của khối trùng với ô nào đang được sử dụng trên lưới, trả về false để chỉ ra rằng khối không fit vào lưới. Nếu không có ô nào trùng khớp, trả về true để chỉ ra rằng khối có thể fit vào lưới mà không gây xung đột với các ô đã được sử dụng.

- HoldBlock(),:

**Chức năng:** Xử lý việc giữ khối.

**A screenshot of a computer program

Description automatically generatedCách thực hiện:** Kiểm tra xem có thể giữ khối hay không (CanHold). Nếu khối được giữ (HeldBlock chưa có), gán HeldBlock bằng CurrentBlock, sau đó cập nhật CurrentBlock từ BlockQueue. Nếu đã có khối được giữ (HeldBlock đã có), hoán đổi giữa CurrentBlock và HeldBlock, sau đó cập nhật CurrentBlock từ BlockQueue. Vô hiệu hóa việc giữ khối mới (CanHold).

- RotateBlockCW(),:

**Chức năng:** Quay khối hiện tại theo chiều kim đồng hồ.

**Cách thực hiện:** Quay CurrentBlock theo chiều kim đồng hồ bằng phương thức RotateCW(). Nếu sau khi quay xong, khối không phù hợp với lưới trò chơi, quay trở lại vị trí ban đầu bằng cách gọi RotateCCW() để quay ngược lại.

- RotateBlockCCW(),:

**Chức năng:** Quay khối hiện tại ngược chiều kim đồng hồ.

**Cách thực hiện:** Quay CurrentBlock ngược chiều kim đồng hồ bằng phương thức RotateCCW(). Nếu sau khi quay xong, khối không phù hợp với lưới trò chơi, quay trở lại vị trí ban đầu bằng cách gọi RotateCW() để quay ngược lại.

- MoveBlockLeft(),:

**Chức năng:** Di chuyển khối hiện tại sang trái.

**Cách thực hiện:** Di chuyển CurrentBlock sang trái bằng cách gọi Move(0, -1). Nếu sau khi di chuyển xong, khối không phù hợp với lưới trò chơi, di chuyển trở lại vị trí ban đầu.

- MoveBlockRight(),:

**Chức năng:** Di chuyển khối hiện tại sang phải.

**A screenshot of a computer program

Description automatically generatedCách thực hiện:** Di chuyển CurrentBlock sang phải bằng cách gọi Move(0, 1). Nếu sau khi di chuyển xong, khối không phù hợp với lưới trò chơi, di chuyển trở lại vị trí ban đầu.

- IsGameOver(),:

**Chức năng:** Kiểm tra xem trò chơi đã kết thúc chưa.

**Cách thực hiện:** Kiểm tra xem hàng đầu tiên và hàng thứ hai của GameGrid có rỗng không. Nếu không, trả về true (kết thúc trò chơi), ngược lại trả về false.

- PlaceBlock(),:

**Chức năng:** Đặt khối hiện tại vào GameGrid và xử lý khi khối chạm đáy hoặc khối khác.

**Cách thực hiện:** Với mỗi vị trí của CurrentBlock, đặt giá trị tương ứng vào GameGrid. Tăng điểm (Score) sau khi dọn sạch hàng. Nếu trò chơi kết thúc, gán GameOver = true, ngược lại lấy khối mới từ BlockQueue và kích hoạt việc giữ khối mới.

- MoveBlockDown(),:

**Chức năng:** Di chuyển khối hiện tại xuống dưới.

**A screenshot of a computer program

Description automatically generatedCách thực hiện:** Di chuyển CurrentBlock xuống dưới bằng cách gọi Move(1, 0). Nếu sau khi di chuyển xong, khối không phù hợp với lưới trò chơi, di chuyển trở lại vị trí ban đầu và đặt khối hiện tại vào vị trí cuối cùng có thể.

- TileDropDistance (),:

**Chức năng:** Đo khoảng cách từ một ô trong khối rơi xuống ô trống gần nhất phía dưới.

**Cách thực hiện:** TileDropDistance() nhận một Position p là vị trí của một ô trong khối. Biến drop được khởi tạo là 0 và tăng dần trong khi ô tiếp theo bên dưới ô này trống trên lưới (GameGrid.IsEmpty(p.Row + drop + 1, p.Column)). Khi không còn ô trống bên dưới, vòng lặp kết thúc và drop trả về giá trị là khoảng cách từ ô hiện tại đến ô trống gần nhất bên dưới nó trên lưới.

- BlockDropDistance(),:

**Chức năng:** Đo khoảng cách từ một khối đang rơi xuống ô trống gần nhất phía dưới.

**Cách thực hiện:** Tính toán khoảng cách từ mỗi ô của CurrentBlock đến ô trống gần nhất phía dưới. Trả về khoảng cách ngắn nhất từ tất cả các ô trong khối đến ô trống gần nhất phía dưới, để chỉ ra khối có thể rơi xuống tối đa bao xa mà không gây va chạm với khối khác.

- DropBlock().:

**Chức năng:** Di chuyển CurrentBlock xuống cuối cùng trong lưới trò chơi và đặt vào vị trí cuối cùng có thể.

**Cách thực hiện:** Sử dụng BlockDropDistance() để tính toán khoảng cách có thể di chuyển xuống cùng. Di chuyển CurrentBlock xuống vị trí cuối cùng có thể bằng cách gọi Move() với khoảng cách tính được từ BlockDropDistance(). Gọi PlaceBlock() để đặt khối vào vị trí cuối cùng trong lưới trò chơi sau khi không thể di chuyển xuống thêm nữa.

A screenshot of a computer program

Description automatically generated

### **3.2.6. MainWindow.xaml.cs (Code-Behind)**

Trong môi trường lập trình WPF của C#, file này là mã code điều khiển logic của cửa sổ chính (MainWindow) của ứng dụng chứa logic để thực hiện trò chơi Tetris và quản lý giao diện người dùng tương ứng.

* **Khai báo hình ảnh:**

- tileImages: Mảng chứa hình ảnh đại diện cho các ô trong trò chơi Tetris (ô trống, ô màu cyan, blue, orange, yellow, green, purple, red).

A screenshot of a computer

Description automatically generated- blockImages: Mảng chứa hình ảnh đại diện cho các khối trong trò chơi Tetris (khối rỗng, khối I, J, L, O, S, T, Z).

* **Biến và Thuộc tính:**

- imageControls: Mảng hai chiều chứa các Image để hiển thị trạng thái của trò chơi trên Canvas.

- easyMaxDelay, easyMinDelay, easyDelayDecrease: Hằng số quy định thời gian delay cho chế độ dễ.

- hardMaxDelay, hardMinDelay, hardDelayDecrease: Hằng số quy định thời gian delay cho chế độ khó.

- Các biến boolean (isEasyMode, isHardMode, gameStarted): Đánh dấu trạng thái của trò

- elapsedTimeInSeconds: Thời gian đã trôi qua trong trò chơi tính bằng giây.

- gameTimer: Sử dụng DispatcherTimer để đếm thời gian trôi qua trong trò chơi.

- gameStarted: Biến cờ chỉ ra xem trò chơi đã bắt đầu chưa.

A screen shot of a computer code

Description automatically generated- gameState: Đối tượng GameState lưu trữ thông tin về trạng thái hiện tại của trò chơi, bao gồm lưới chơi, khối gạch, điểm số và các thuộc tính khác.

* **Phương thức:**

**-** Khởi tạo và Thiết lập Ban Đầu (MainWindow() và SetupGameCanvas()):

**Chức năng:** Tạo cửa sổ chính của ứng dụng và thiết lập giao diện người dùng cho trò chơi Tetris.

**Cách thực hiện:**

MainWindow(): Khởi tạo các hình ảnh cho ô và khối từ tệp ảnh tương ứng. Khởi tạo các biến và thuộc tính cần thiết cho trạng thái trò chơi. Gắn sự kiện Window\_KeyDown để xử lý các sự kiện phím từ người chơi.

A screenshot of a computer program

Description automatically generatedSetupGameCanvas(): Tạo ra một lưới Canvas bằng cách thêm các Image tương ứng vào từng ô của trò chơi. Cài đặt kích thước và vị trí của từng ô trên Canvas.

- Vẽ Trạng Thái Trò Chơi (DrawGrid(), DrawBlock(), DrawNextBlock(), DrawHeldBlock(), DrawGhostBlock(), Draw()):

**Chức năng:** Hiển thị trạng thái của lưới và các khối trên giao diện người dùng. **Cách thực hiện:**

DrawGrid(): Vẽ lưới game dựa trên trạng thái của GameGrid.

DrawBlock(): Vẽ khối hiện tại dựa trên trạng thái của Block.

DrawNextBlock(): Hiển thị khối tiếp theo trên giao diện.

DrawHeldBlock(): Hiển thị khối được giữ trên giao diện.

DrawGhostBlock(): Vẽ khối bóng ma để dự đoán vị trí rơi của khối.

A screenshot of a computer program

Description automatically generatedDraw(): Gọi các phương thức vẽ để cập nhật trạng thái trò chơi trên giao diện.

A computer screen shot of a program

Description automatically generated- Vòng Lặp Chính Của Trò Chơi (GameLoop()):

**Chức năng:** Điều khiển vòng lặp chính của trò chơi Tetris, cập nhật trạng thái, xử lý đầu vào từ người chơi và vẽ lại giao diện.

**A screenshot of a computer program

Description automatically generatedCách thực hiện:** Vòng lặp trong phương thức này duy trì trạng thái của trò chơi, cập nhật thời gian trôi qua, tính toán thời gian chờ giữa các lần di chuyển của khối gạch, và vẽ lại giao diện dựa trên trạng thái hiện tại của trò chơi.

**-** Xử Lý Sự Kiện Phím (Window\_KeyDown(object sender, KeyEventArgs e)):

**Chức năng:** Phản hồi đối với các phím được nhấn từ bàn phím để điều khiển khối gạch trong trò chơi.

**Cách thực hiện:** Nhận diện các phím mũi tên (trái, phải, lên, xuống) để di chuyển khối gạch. Sử dụng các phím "Z" và "Up" để quay khối gạch theo chiều kim đồng hồ. Phím "C" dùng để giữ khối gạch. Phím "Space" để thả khối gạch xuống nhanh hơn.

**A screenshot of a computer program

Description automatically generated**

- Quản Lý Bộ Đếm Thời Gian (StartGameTimer(), StopGameTimer(), GameTimer\_Tick(object sender, EventArgs e), UpdateElapsedTime(), FormatTime(int seconds)):

**Chức năng:** Điều khiển và cập nhật bộ đếm thời gian của trò chơi.

**Cách thực hiện:**

StartGameTimer(): Khởi tạo và bắt đầu bộ đếm thời gian sử dụng DispatcherTimer.

StopGameTimer(): Dừng bộ đếm thời gian.

GameTimer\_Tick(object sender, EventArgs e): Xử lý sự kiện tick của bộ đếm thời gian, cập nhật thời gian đã trôi qua.

UpdateElapsedTime(): Cập nhật giao diện người dùng hiển thị thời gian đã trôi qua.

FormatTime(int seconds): Định dạng thời gian từ giây sang chuỗi thời gian có dạng mm:ss.

A screenshot of a computer program

Description automatically generated

- Xử Lý Sự Kiện GameCanvas\_Loaded(), Button Click (PlayButton\_Click(), ExitButton\_Click(), PlayAgain\_Click()):

**Chức năng:** Xử lý các sự kiện từ giao diện người dùng như sự kiện tải, nhấn nút để điều khiển trò chơi hoặc kết thúc ứng dụng.

**Cách thực hiện:**

GameCanvas\_Loaded(): Thực hiện các thiết lập ban đầu cho trò chơi khi GameCanvas được tải.

A screenshot of a computer program

Description automatically generatedPlayButton\_Click(), ExitButton\_Click(), PlayAgain\_Click(): Xử lý các sự kiện nhấn nút để điều khiển trò chơi hoặc kết thúc ứng dụng.

A computer screen shot of text

Description automatically generated

### **3.2.7. MainWindow.xaml**

MainWindow.xamllà một tệp XAML (eXtensible Application Markup Language) trong ứng dụng WPF (Windows Presentation Foundation) của Microsoft. Nó định nghĩa giao diện người dùng cho một ứng dụng chơi game Tetris và Mỗi thành phần trong giao diện này được định nghĩa trong các hàng và cột khác nhau của Grid để xác định vị trí và cấu trúc của chúng trên cửa sổ ứng dụng.

- Window:

x:Class="Tetris.MainWindow": Xác định lớp code-behind tương ứng với tệp XAML này.

Title="Tetris": Đặt tiêu đề cho cửa sổ là "Tetris"

Height="600" Width="800": Kích thước ban đầu của cửa sổ là 800x600 pixels.

MinWidth="600" MinHeight="600": Kích thước tối thiểu của cửa sổ là 600x600 pixels.

Foreground, FontFamily, FontSize: Các thuộc tính liên quan đến văn bản trong cửa sổ.

- Grid: Sử dụng Grid để chia giao diện thành hàng và cột.

- Grid.RowDefinitions và Grid.ColumnDefinitions: Định nghĩa số lượng và tỷ lệ chiều cao, chiều rộng của hàng và cột trong Grid.

- Grid.Background: Sử dụng “ImageBrus h” để đặt hình nền cho Grid từ một hình ảnh.

- Grid (nằm trong Row=1, Column=1): Là nơi chứa phần chính của trò chơi Tetris, bao gồm hình ảnh logo Tetris, RadioButton để chọn chế độ chơi (Easy/Hard), nút Play và Exit, và một Canvas để vẽ trò chơi.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

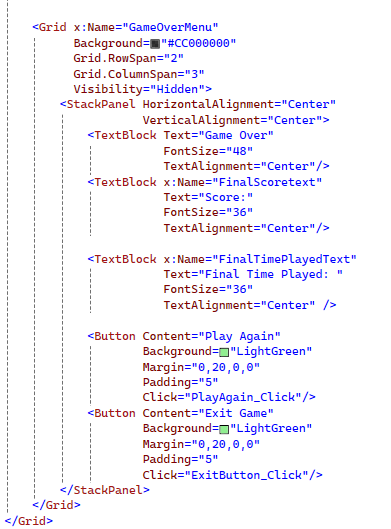
A screenshot of a computer program

Description automatically generated- TextBlocks: ScoreText, ElapsedTimeText, và ModeIndicator: Hiển thị thông tin điểm số, thời gian chơi và chỉ số chế độ chơi. Ban đầu, các TextBlock này được ẩn (Visibility="Collapsed") và sẽ hiển thị khi cần thiết.

A screenshot of a computer program

Description automatically generated- StackPanels: HoldPanel và NextPanel chứa hình ảnh về các khối tiếp theo và khối được giữ. Được căn giữa theo chiều ngang (HorizontalAlignment="Right"/"Left") và căn giữa theo chiều dọc (VerticalAlignment="Center").

**A screenshot of a computer

Description automatically generated**- Grid "GameOverMenu": Hiển thị menu khi trò chơi kết thúc. Bao gồm thông tin về kết quả cuối cùng và hai nút để chơi lại hoặc thoát game. Được ẩn ban đầu (Visibility="Hidden").

Hình 3.1. Giao diện trò chơi xếp hình Tetris

## **3.3. MỘT SỐ GIAO DIỆN CỦA ỨNG DỤNG**

**A screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generated**

Hình 3.3. Giao diện chơi của chế độ HardMode

Hình 3.2. Giao diện chơi của chế độ EasyMode

**A screenshot of a game

Description automatically generatedA screenshot of a game

Description automatically generated**

Hình 3.4. Giao diện trò chơi kết thúc của chế độ EasyMode

Hình 3.5. Giao diện trò chơi kết thúc của chế độ HardMode

## **3.4. MỘT SỐ ĐOẠN MÃ LỆNH XỬ LÝ CHÍNH TRONG ỨNG DỤNG**

* A computer code with text

  Description automatically generated with medium confidence**Xử lý việc kiểm tra và xóa các hàng đầy đủ trong lưới trò chơi Tetris.**
* **A screenshot of a computer program

  Description automatically generatedĐặt khối hiện tại vào lưới, xử lý việc xóa hàng đầy, cập nhật điểm số và kiểm tra kết thúc trò chơi**
* **A screenshot of a computer program

  Description automatically generatedBlock tiếp theo không trùng với block trước đó, giữ cho trò chơi mang tính ngẫu nhiên và đa dạng**
* **A screenshot of a computer program

  Description automatically generatedĐiều khiển luồng logic của trò chơi Tetris từ khi bắt đầu đến khi kết thúc.**
* **A screenshot of a computer program

  Description automatically generatedXử lý sự kiện phím trong trò chơi Tetris**

* **Biểu diễn vị trí của các ô trong lưới của trò chơi Tetris**

A screenshot of a computer code

Description automatically generated

* **Quản lý trạng thái của trò chơi Tetris**A computer code with text

  Description automatically generated with medium confidence

# **CHƯƠNG 4: KẾT LUẬN**

## **4.1. ĐÁNH GIÁ VỀ HỆ THỐNG ĐỀ TÀI**

Xây dựng được một trò chơi xếp hình Tetris với một số tính năng đơn giản như:

- Cơ bản của trò chơi: Tính năng cơ bản như di chuyển, quay và xếp các khối gạch để tạo hàng ngang đầy đủ, cùng với giao diện trực quan, thú vị.

- Điểm Số và Level: Tính điểm dựa trên việc xóa hàng hoặc xếp các khối, đồng thời cung cấp hai chế độ chơi khác nhau: dễ và khó, với cấp độ khó tăng dần theo thời gian và điểm số.

- Kết Thúc Trò Chơi: Bạn có thể đã thêm điều kiện kết thúc trò chơi khi không còn không gian trống cho các khối gạch mới.

- Xử Lý Sự Kiện Người Dùng: Thêm các sự kiện điều khiển từ bàn phím để người chơi di chuyển, quay và thả các khối gạch.

## **4.2. ƯU VÀ NHƯỢC ĐIỂM CỦA ĐỀ TÀI**

* **Ưu điểm**

- Tính giải trí và tập trung: Tetris yêu cầu sự tập trung cao đồng thời vẫn giữ được tính giải trí. Phiên bản đơn giản này có thể giúp người chơi tập trung hơn vào xử lý logic và nâng cao khả năng tập trung.

- Tính Tương Tác: Sự kết hợp giữa C# và WPF cung cấp cơ hội tạo hiệu ứng đồ họa và giao diện người dùng trực quan, làm tăng trải nghiệm tương tác và thị giác của người chơi.

- Áp dụng lập trình hướng đối tượng (OOP): Sử dụng phương pháp lập trình OOP giúp tạo ra mã nguồn dễ đọc, dễ bảo trì và mở rộng.

- Tăng kỹ năng lập trình: Xây dựng trò chơi Tetris cung cấp môi trường thực hành tốt để làm quen với kiến thức lập trình. Từ việc xử lý sự kiện đến quản lý trạng thái game, đây là cơ hội để áp dụng kiến thức và kỹ năng lập trình.

* **Nhược điểm**

- Thiếu tính năng và thử nghiệm: Không có các tính năng mở rộng như việc lưu trữ điểm số, leaderboard hoặc chế độ chơi đa người. Đồng thời, cũng không có đủ thời gian để thử nghiệm hoặc triển khai những tính năng khác một cách đầy đủ.

- Kiểm thử và sửa lỗi: Việc tìm ra và sửa lỗi trong một dự án lớn có thể là công việc tốn thời gian và công sức. Điều này càng trở nên phức tạp khi cần xử lý nhiều khả năng diễn ra trong game.

- Thời gian và phức tạp: Xây dựng một trò chơi có thể mất nhiều thời gian hơn dự kiến, đặc biệt là khi cố gắng thêm các tính năng phức tạp hoặc khi phải xử lý các vấn đề phức tạp hơn trong quá trình phát triển.

- Khả năng mở rộng và bảo trì: Việc thêm tính năng mới hoặc sửa đổi mã nguồn có thể gặp nhiều khó khăn nếu mã nguồn không được tổ chức cẩn thận từ đầu.

## **4.3. HƯỚNG PHÁT TRIỂN**

- Tính năng nâng cao: Bổ sung tính năng lưu trữ điểm số cao, leaderboard để thúc đẩy cạnh tranh giữa người chơi.

- Giao diện người dùng cải thiện: Tối ưu hóa giao diện người dùng để tăng tính tương tác và thẩm mỹ.

- Chế độ chơi mở rộng: Phát triển chế độ chơi đa người, hoặc thêm các loại khối gạch mới để làm phong phú hơn trò chơi.

- Tối ưu hóa hiệu suất: Tăng cường hiệu suất của trò chơi để đảm bảo trải nghiệm chơi mượt mà trên nhiều loại thiết bị.

## **4.4. RÚT RA KINH NGHIỆM ĐỂ PHÁT TRIỂN BẢN THÂN**

Sau khi hoàn thành khóa học thực tập cơ sở, em đã tích lũy được nhiều kiến thức mới về lập trình C#, đặc biệt là về lập trình hướng đối tượng và sử dụng Windows Presentation Foundation (WPF). Lập trình hướng đối tượng đã mở ra cánh cửa để hiểu sâu hơn về cách thiết kế các lớp, đối tượng và quan hệ giữa chúng khi xây dựng ứng dụng. Tận dụng tính kế thừa, đa hình và trừu tượng hóa, em đã phát triển một trò chơi thú vị bằng C#. Đồng thời việc làm quen với WPF đã mở ra trải nghiệm mới, đã giúp em tạo ra giao diện người dùng linh hoạt hơn. Quan trọng hơn, em đã học được cách xử lý lỗi và tối ưu mã nguồn, từ đó nâng cao khả năng làm việc độc lập và cách tiếp cận các dự án phức tạp hơn. Những kỹ năng này sẽ là nền tảng vững chắc cho sự nghiệp lập trình của em trong tương lai, cả trong việc làm việc độc lập và làm việc nhóm.

# **TÀI LIỆU THAM KHẢO**

[1][Tổng quan về Lập trình hướng đối tượng (OOP - Object Oriented Programming) - BumBii](https://bumbii.com/tong-quan-ve-lap-trinh-huong-doi-tuong-oop-object-oriented-programming/) (BumBii, 2023, Tổng quan về Lập trình hướng đối tượng (OOP – Object Oriented Programming)), truy cập ngày 2/1/2023.

[2][Tổng quan lập trình hướng đối tượng OOP: Những kiến thức cơ bản và ví dụ thực tế - CodeGym](https://codegym.vn/blog/tong-quan-lap-trinh-huong-doi-tuong-oop/) (CodeGym, 2023, Tổng quan lập trình hướng đối tượng OOP: Những kiến thức cơ bản và ví dụ thực tế), truy cập ngày 2/1/2023.

[3] [Cú pháp C# cơ bản - Học lập trình C# online - VietTuts](https://viettuts.vn/csharp/cu-phap-csharp-co-ban) (VietTuts, 2016, Cú pháp C# cơ bản), truy cập ngày 3/1/2023.

[4][C# là gì ? Tổng quan về C# - Freetuts](https://freetuts.net/c-sharp-la-gi-tong-quan-ve-c-sharp-1045.html) ((Freetuts, 2021, C# là gì ? Tổng quan về C#), truy cập ngày 3/1/2023.

[5][[WPF 1] Tổng quan về Windows Presentation Foundation - Kiến thức Lập trình (kienthuclaptrinh.vn)](https://kienthuclaptrinh.vn/2012/07/24/wpf-1-tong-quan-ve-windows-presentation-foundation-inprogress/)(kienthuclaptrinh, 2012, [WPF 1] Tổng quan về Windows Presentation Foundation), truy cập ngày 4/1/2023.

[6][(11) Programming a Tetris Game in C# - Full Guide - YouTube](https://www.youtube.com/watch?v=jcUctrLC-7M&t=1437s) (OttoBotCode, 2022, Programming a Tetris Game in C#), truy cập ngày 1/12/2023.