

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM



ĐỒ ÁN MÔN HỌC
LẬP TRÌNH TRỰC QUAN

TRÒ CHƠI XẾP HÌNH

Giảng viên hướng dẫn : Nguyễn Thị Xuân Hương

Sinh viên thực hiện 1 : Trần Thị Thu Hoài

Mã sinh viên 1 : 23520509

Sinh viên thực hiện 2 : Bùi Quốc Bảo

Mã sinh viên 2 : 23520093

Sinh viên thực hiện 3 : Nguyễn Đình Hoài Nam

Mã sinh viên 3 : 23520975

Lớp : IT008.P13

Bộ môn : Phát triển phần mềm

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2024



NHIỆM VỤ ĐỒ ÁN MÔN HỌC

Họ và tên SV 1: **Trần Thị Thu Hoài**

MSSV: **23520509**

Họ và tên SV 2: **Bùi Quốc Bảo**

MSSV: **23520093**

Họ và tên SV 2: **Nguyễn Đình Hoài Nam**

MSSV: **23520975**

Lớp: **IT008.P13**

Tên đề tài: **Trò chơi xếp hình**

Giảng viên giảng dạy: **Nguyễn Thị Xuân Hương**

Thời gian thực hiện: **từ 3/10/2024 đến 4/1/2025**

Nhiệm vụ đồ án môn học: (phụ thuộc vào từng chủ đề)

1. Xây dựng CSDL trong SQL Server.
2. Thiết kế giao diện phần mềm.
3. Lập trình xử lý phần mềm với các chức năng sau:
 - Đăng nhập / đăng xuất
 - Quản lý thông tin người chơi
 - Quản lý cấp độ chơi
 - Phân mảnh hình ảnh theo lưới
 - Quản lý âm thanh
 - Quản lý trạng thái trò chơi
 - Quản lý bảng xếp hạng
 - Điều khiển trò chơi
4. Nộp file nén (*.rar) lưu sản phẩm đề tài bao gồm:
 - File báo cáo word (*.docx)
 - File thuyết trình (*.pptx)
 - Thư mục chứa dự án (project), các class thư viện, CSDL, hình ảnh, ...)

Tp.HCM, ngày 21 tháng 12 năm 2024

GIẢNG VIÊN GIẢNG DẠY

(Ký và ghi rõ họ tên)

.....

BẢNG PHÂN CÔNG THỰC HIỆN ĐỒ ÁN MÔN HỌC (Nếu đồ án chỉ có 1 SV thực hiện thì không làm trang này)		
Họ tên SV1: Trần Thị Thu Hoài MSSV: 23520509	Họ tên SV2: Bùi Quốc Bảo MSSV: 23520093	Họ tên SV3: Nguyễn Đình hoài Nam MSSV: 23520975
Thiết kế ,xử lí logic giao diện Xử lí thuật toán, logic trò chơi Báo cáo word	Thiết kế,xử lí logic giao diện Thiết kế cơ sở dữ liệu Powerpoint báo cáo	Thiết kế, xử lí logic giao diện Video demo Powerpoint báo cáo
SV thực hiện 1 (Ký tên)	SV thực hiện 2 (Ký tên)	SV thực hiện 3 (Ký tên)

Phân công chi tiết công việc: <https://neon-power-7ec.notion.site/Task-Project-1127b6dc0527800a831bf43a46da41d8?pvs=4>

LỜI CẢM ƠN

Nhóm chúng em xin chân thành gửi lời cảm ơn tới cô Nguyễn Thị Xuân Hương truyền đạt những kiến thức bổ ích và tận tình hướng dẫn nhóm chúng em trong suốt quá trình học tập, giúp chúng em hiểu sâu hơn về chuyên môn và rèn luyện được những kỹ năng cần thiết để hoàn thành bài kết quả môn học này.

Bài làm này là kết quả của quá trình học tập và nỗ lực của cả nhóm, nhưng chắc chắn vẫn còn những thiếu sót và hạn chế về nhiều mặt. Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ cô và các bạn để có thể cải thiện và hoàn thiện hơn trong những lần sau.

Một lần nữa, em xin trân trọng cảm ơn sự giúp đỡ và đồng hành cả nhóm và của tất cả mọi người trong suốt thời gian qua.

Nhóm sinh viên thực hiện

Trần Thị Thu Hoài - Bùi Quốc Bảo - Nguyễn Đình Hoài Nam

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp.HCM, ngày ... tháng ... năm ...

GVHD

MỤC LỤC

Chương 1: GIỚI THIỆU ĐỀ TÀI	10
1.1. Tên đề tài:.....	10
1.2. Mô tả đề tài:	10
1.3. Lí do chọn đề tài:	11
1.4. Các chức năng chính của đề tài:	11
1.5. Công nghệ sử dụng:	12
1.6. Môi trường lập trình:	12
1.7. Công cụ hỗ trợ (nếu có):	12
Chương 2 : GIỚI THIỆU CÔNG NGHỆ	13
2.1. WPF (Windows Presentation Foundation):	13
2.3.1. Lịch sử:	13
2.3.2. Nhiệm vụ, chức năng:	13
2.3.3. Áp dụng :	13
2.2. SQL Server:.....	13
2.3.1. Lịch sử:	13
2.3.2. Nhiệm vụ, chức năng:	13
2.3.3. Áp dụng:	13
2.3. Ngôn ngữ lập trình C#:.....	13
2.3.1. Lịch sử:	13
2.3.2. Nhiệm vụ, chức năng:	14
2.3.3. Áp dụng:	14
Chương 3: THIẾT KẾ HỆ THỐNG.....	14

3.1. Mô tả bài toán :	14
3.2. Khảo sát hiện trạng & Xác định chức năng hệ thống	14
3.4.1. Đăng kí /đăng nhập:	14
3.4.2. Thiết lập lượt chơi	14
3.4.3. Khởi tạo lượt chơi	15
3.4.4. Xử Lí đầu vào điều khiển:	15
3.4.5. Kết thúc lượt chơi:	15
3.4.6. Tra cứu thứ tự xếp hạng:	15
3.3. Xây dựng cơ sở dữ liệu SLIDING_PUZZLE_DB:	15
3.4.1. Mô tả tân từ:	15
3.4.2. Database diagram trong SQL:	16
3.4.3. Cấu trúc các bảng dữ liệu trong SQL:	16
3.4.4. Dữ liệu mẫu cho các bảng :	16
3.4. Thuật toán:	18
3.4.1. Chia nhỏ ảnh thành các mảnh & tạo ma trận trò chơi:	18
3.4.2. Khởi tạo trạng thái bắt đầu:	18
3.4.3. Thuật toán kiểm tra trạng thái trò chơi và bộ đếm giờ:	19
3.5. Sơ đồ phân tích trò chơi:	20
Chương 4: XÂY DỰNG ỨNG DỤNG	20
4.1. Thiết kế Giao diện :	20
4.1.1. Trang menu chính:	20
4.1.2. Trang đăng nhập/đăng xuất:	21
4.1.3. Trang thiết lập game:	22
4.1.4. Trang thư viện ảnh cá nhân:	23

4.1.5.	Cửa sổ thêm ảnh:.....	23
4.1.6.	Trang chơi game chính:	24
4.1.7.	Cửa sổ cài đặt (tích hợp với cửa sổ chính) :.....	25
4.1.8.	Cửa sổ kết thúc lượt chơi:	26
4.1.9.	Dialog thông báo:	27
4.1.10.	Trang bảng xếp hạng:	27
4.2.	Thiết kế và tạo các class:	28
4.2.1.	Class CusPiece:.....	28
4.2.2.	Class Connection:	29
4.2.3.	Class Player & Picture:	29
4.2.4.	Class GameModel:	30
4.3.	Thiết kế các hàm & thuật toán cốt lõi:	31
4.2.1.	Hàm:	31
4.2.2.	Thuật toán :.....	34
4.4.	Hệ thống âm thanh với MusicSystemService:	36
4.5.	Các Service thao tác với Database:.....	37
4.6.5.	LoadPictureListService:	37
4.6.6.	LeaderBoardService:	37
4.6.	Các Class hỗ trợ:	38
4.6.1.	FocusHelper:	38
Chương 5: KẾT LUẬN.....		38
5.1.	Các kết quả đạt được của đề án:.....	39
5.2.	Ưu điểm của đề án:	39
5.3.	Hạn chế của đề án:.....	39

5.4.	Hướng phát triển của đồ án:.....	39
-------------	---	-----------

Chương 1: GIỚI THIỆU ĐỀ TÀI

1.1. Tên đề tài:

TRÒ CHƠI XẾP HÌNH (SlideFun)

1.2. Mô tả đề tài:

- Thu thập và phân tích yêu cầu đề tài:
 - Xác định mục tiêu game và tiến hành thu thập ý kiến về yêu cầu trải nghiệm với các đối tượng người chơi (giải trí , giáo dục,...) ở nhiều lứa tuổi.
 - Phân tích các chức năng cơ bản, đồng thời xem xét tới các yêu cầu phi chức năng (âm thanh, thông báo,...)
 - Xác định quy trình làm việc, tổ phản hồi ,đánh giá và điều chỉnh
- Thiết kế giao diện người dùng:
 - Giao diện thân thiện, dễ hiểu, dễ sử dụng, phù hợp với đối tượng người dùng mục tiêu.
 - Màu sắc hài hòa, hình ảnh sinh động và thu hút.
 - Hiển thị thông tin rõ ràng, sắp xếp các nút chức năng logic dễ dàng trong việc chọn ảnh, phân mảnh ảnh,...
- Phát triển phần mềm:
 - Lựa chọn công cụ phát triển hợp lí.
 - Lập trình và phát triển phần mềm dựa trên thiết kế và yêu cầu đã xác định.
 - Đảm bảo hoàn thành các chức năng game chọn ảnh, thêm ảnh, xóa ảnh, xử lí đầu vào,....
- Kiểm tra chất lượng:
 - Tiến hành test phần mềm , đảm bảo chạy mượt mà, phát hiện và sửa lỗi kịp thời.

- đảm bảo chức năng của phần mềm hoạt động đúng theo yêu cầu, hiệu quả và đáp ứng các yêu cầu đã đặt ra.

1.3. Lí do chọn đề tài:

- Việc phát triển một game xếp hình mang đến cơ hội tạo ra một trò chơi không chỉ dễ tiếp cận mà còn đầy thử thách, kích thích tư duy và khả năng giải quyết vấn đề của người chơi. Đây là một thể loại game đơn giản nhưng lại mang lại sự quyến rũ mạnh mẽ, thu hút người chơi ở mọi độ tuổi và trình độ. Với sự kết hợp hoàn hảo giữa trí tuệ và giải trí, game xếp hình trở thành một công cụ tuyệt vời để rèn luyện khả năng tư duy logic, khả năng phân tích và giải quyết vấn đề. Mỗi thử thách trong game như một câu đố nhỏ, khơi dậy sự tò mò và niềm vui khi người chơi hoàn thành, mang lại cảm giác thỏa mãn và chiến thắng sau mỗi lần vượt qua.
- Game xếp hình tương đối dễ phát triển và triển khai mà không đòi hỏi quá nhiều tài nguyên đồ họa hay âm thanh phức tạp, mang lại cơ hội phát triển không giới hạn, với khả năng bổ sung nhiều chế độ chơi đa dạng, bảng xếp hạng hấp dẫn, các phần thưởng đầy thử thách, tạo ra một trải nghiệm không bao giờ nhàm chán.
- Với tất cả những yếu tố này, game xếp hình không chỉ là một trò chơi giải trí, mà còn là một hành trình khám phá vô tận, một sân chơi trí tuệ, và là cơ hội để phát triển một sản phẩm có giá trị cao về mặt giáo dục, kinh tế và trải nghiệm người dùng. Việc chọn đề tài này làm đề án môn học đầu tay của chúng em là một nước đi khôn ngoan để tạo ra một sản phẩm vừa đơn giản mà lại hấp dẫn, vừa thách thức mà lại dễ tiếp cận, mang đến cho người chơi những giờ phút thú vị và bổ ích.

1.4. Các chức năng chính của đề tài:

- Quản lý cấp độ chơi :
 - Hiện thị một số kích thước lưới phân mảnh hình ảnh theo mức tăng dần.
 - Thiết lập thời gian chơi tối đa trong một lượt chơi.
 - Lựa chọn hình ảnh .
- Quản lý âm thanh :
 - Thiết lập nhạc nền và âm thanh hiệu ứng .
 - Điều chỉnh âm lượng.
 - Tắt mở âm thanh.
- Quản lý thư viện ảnh :
 - Thêm hình ảnh.
 - Xóa hình ảnh (trừ các hình ảnh mặc định).
 - Hiện thị kho ảnh với từng người chơi.
- Quản lý người chơi

- Hiển thị thông tin người chơi.
- Sao chép ID người chơi vào Clipboard.
- Điều khiển trò chơi:
 - Điều khiển xếp hình sử dụng chuột.
 - Điều khiển xếp hình sử dụng bàn phím.
- Quản lí bảng xếp hạng:
 - Tra cứu thứ tự người chơi.
 - Hiện thị thứ tự xếp hạng của các người chơi
- Hiện thị thông tin về nhóm phát triển , hướng dẫn chơi.

1.5. Công nghệ sử dụng:



Language



Framework



DB management system

1.6. Môi trường lập trình:



1.7. Công cụ hỗ trợ (nếu có):



Chương 2 : GIỚI THIỆU CÔNG NGHỆ

2.1. WPF (Windows Presentation Foundation):

2.3.1. Lịch sử:

WPF (Windows Presentation Foundation) ra đời năm 2006 cùng với .NET Framework 3.0, ban đầu có tên mã là Avalon. Đây là một framework của Microsoft nhằm thay thế Windows Forms, sử dụng XAML để định nghĩa giao diện và DirectX để xử lý đồ họa, mang lại khả năng phát triển ứng dụng với giao diện phong phú và tùy chỉnh cao. Qua các phiên bản như .NET 3.5, 4.0, và .NET Core 3.0, WPF được cải thiện hiệu năng, hỗ trợ mô hình MVVM và đồ họa 3D. Năm 2019, WPF trở thành mã nguồn mở, tiếp tục phát triển trong các phiên bản .NET hiện đại như .NET 6, giữ vững vị trí trong việc phát triển ứng dụng desktop trên Windows.

2.3.2. Nhiệm vụ, chức năng:

- Thiết kế giao diện Sử dụng XAML để tạo UI đẹp mắt, hỗ trợ hiệu ứng đồ họa, bố cục linh hoạt (Grid, StackPanel, ...).
- Quản lý tương tác các sự kiện (click, textchanged,...). Bên cạnh đó còn có cơ chế binding mạnh mẽ, với MVVM là thể mạnh lớn trong việc tách biệt logic xử lý với giao diện.
- Cung cấp các Control dễ dàng trong việc thiết kế giao diện ứng dụng.

2.3.3. Áp dụng :

- Thiết kế, xây dựng giao diện các page, window của trò chơi.
- Tận dụng thế mạnh của WPF là Data Binding, Commands, và Dependency Properties, là các tính năng cốt lõi để MVVM hoạt động hiệu quả.

2.2. SQL Server:

2.3.1. Lịch sử:

SQL Server, phát triển bởi Microsoft từ 1989, ban đầu dựa trên SQL của Sybase. Sau nhiều phiên bản, như SQL Server 7.0 (1998) và SQL Server 2005 (2005), nó tiếp tục phát triển mạnh mẽ, thêm tính năng bảo mật, hiệu suất và tích hợp đám mây, trở thành một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất.

2.3.2. Nhiệm vụ, chức năng:

SQL Server là hệ quản trị cơ sở dữ liệu giúp quản lý, lưu trữ, và truy vấn dữ liệu. Nó hỗ trợ bảo mật, sao lưu, phục hồi, tối ưu hiệu suất, và phân tích dữ liệu, đồng thời tích hợp với môi trường đám mây.

2.3.3. Áp dụng:

quản lý dữ liệu người dùng, lưu trữ các hình ảnh trong kho ảnh cá nhân và lịch sử chơi thông qua các bảng :PLAYER, PICTURE và...

2.3. Ngôn ngữ lập trình C#:

2.3.1. Lịch sử:

C# được Microsoft phát triển vào đầu những năm 2000 dưới sự lãnh đạo của Anders Hejlsberg, ra mắt lần đầu vào năm 2000 như một

phần của .NET Framework. C# nhắm đến việc cung cấp một ngôn ngữ đơn giản, hiện đại và mạnh mẽ cho phát triển ứng dụng. Các phiên bản sau này tiếp tục cải tiến với các tính năng như LINQ, lập trình bất đồng bộ, và hỗ trợ đa nền tảng qua .NET Core. Hiện nay, C# được sử dụng rộng rãi trong phát triển ứng dụng desktop, web, di động, và game.

2.3.2. Nhiệm vụ, chức năng:

Hỗ trợ phát triển ứng dụng trên windows cung cấp khả năng lập trình hướng đối tượng như kế thừa, đa hình và đóng gói.

2.3.3. Áp dụng:

Hỗ trợ xây dựng giao diện người dùng với WPF xây dựng các class cần thiết cũng như các hàm để xử lý logic trò chơi.

Chương 3: THIẾT KẾ HỆ THỐNG

3.1. Mô tả bài toán :

- Puzzle Game là một trò chơi bắt nguồn từ năm 1760 bởi **John Spilsbury** và trở nên phổ biến hơn từ những năm 1980 đến 1990, người chơi phải sắp xếp các mảnh ghép bị xáo trộn để tái tạo một bức tranh hoàn chỉnh. Người chơi có thể lựa chọn từ nhiều hình ảnh đa dạng, điều chỉnh số lượng mảnh ghép để thay đổi độ khó. Khi hoàn thành bức tranh, trò chơi hiển thị kết quả với thời gian hoàn thành, tạo động lực cho người chơi chinh phục những thử thách tiếp theo.
- Trong đề án này, trò chơi xếp tranh có thể được coi là một bài toán N-puzzle bằng cách xem mỗi mảnh ghép là một ô trong bảng lưới $N \times N$. Mỗi ô đều được đánh số, và có một ô trống để thực hiện các nước đi. Mục tiêu là sắp xếp lại các ô để chúng tạo thành bức tranh hoàn chỉnh, tương tự như sắp xếp các số theo thứ tự tăng dần với ô trống ở cuối.

3.2. Khảo sát hiện trạng & Xác định chức năng hệ thống

3.4.1. Đăng kí /đăng nhập:

Người chơi sử dụng tài khoản để tham gia trò chơi.

3.4.2. Thiết lập lượt chơi

Người chơi lựa chọn kích thước lưới phân mảnh hình ảnh phù hợp. Người chơi có thể lựa chọn hình ảnh yêu thích trong kho ảnh mặc định hoặc có thể thêm/xóa(trừ cá hình ảnh mặc định) hình ảnh trong kho ảnh cá nhân, các hình ảnh được thêm mới vào sẽ được lưu trữ và không bị mất đi sau khi người chơi logout .Ngoài ra người chơi có thể đặt thời gian chơi giới hạn nếu muốn.

3.4.3. Khởi tạo lượt chơi

Cắt hình ảnh thành dạng lưới theo kích thước đã được người chơi thiết lập trong giai đoạn [thiết lập lượt chơi](#) và khởi tạo các đối tượng cần thiết như hình ảnh gốc hỗ trợ, bộ đếm thời gian,...

3.4.4. Xử Lí đầu vào điều khiển:

Xử lí các thao tác click chuột hay dùng các phím điều khiển (←↑→↓) di chuyển các khối hình về đúng vị trí như hình gốc ban đầu.

3.4.5. Kết thúc lượt chơi:

Trò chơi kết thúc khi thời gian của bộ đếm ngược trở về = 0 hoặc các khối hình được đưa về đúng vị trí so với hình gốc, lưu lại thời gian chơi ngắn nhất ứng với mỗi kích thước lưới phục vụ cho bảng xếp hạng.

3.4.6. Tra cứu thứ tự xếp hạng:

Người chơi có thể tra cứu thứ tự xếp hạng của mình so với các người chơi khác ứng với từ kích thước lưới cho trước trong bảng xếp hạng.

3.3. Xây dựng cơ sở dữ liệu SLIDING_PUZZLE_DB:

3.4.1. Mô tả tân từ:

PLAYER (PLAYERID, PLAYERNAME, PLAYERPASSWORD)

Tân từ: : **PLAYER** lưu trữ thông tin người chơi bao gồm: mã người chơi(P_{PLAYERID}), Tên người chơi(P_{PLAYERNAME}), mật khẩu đăng nhập (P_{PLAYERPASSWORD}).

PICTURE (PICNAME, P_{PLAYERID}, PICPATH, isDEFAULT)

Tân từ: : **PICTURE** lưu trữ thông tin người chơi bao gồm: Tên hình ảnh (P_{PICNAME}), mã người chơi(P_{PLAYERID}) với mục đích cá nhân hóa kho ảnh , đường dẫn hình ảnh trong kho ảnh của ứng dụng(P_{PICPATH}) và một bit (isDEFAULT) để phân biệt hình ảnh mặc định cho trước của ứng dụng với các hình ảnh khác.

GAMREROUND (GAMEID, PLAYERNAME, PLAYERID, PIECES, PLAYTIME, PLAYDATE)

Tên từ: : GAMEROUND lưu trữ thông tin về lịch sử chơi: mã lượt chơi (GAMEID), tên người chơi (PLAYERNAME), mã người chơi (PLAYERID) , kích thước lưới ảnh (PIECES) , thời gian chơi (PLAYTIME) và thời điểm chơi(PLAYDATE).

3.4.2. Database diagram trong SQL:

3.4.3. Cấu trúc các bảng dữ liệu trong SQL:

3.3.3.1. Bảng PLAYER:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
PLAYERID	VARCHAR(6)	PRIMARY KEY	Mã người chơi
PLAYERNAME	VARCHAR(40)	NOT NULL, UNIQUE	Tên người chơi
PLAYERPASSWORD	VARCHAR(MAX)	NOT NULL	Mật khẩu đăng nhập

3.3.3.2. Bảng PICTURE:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
PICNAME	VARCHAR(40)	PRIMARY KEY	Tên hình ảnh
PLAYERID	VARCHAR(6)	PRIMARY KEY	Mã người chơi thêm ảnh vào kho
PICPATH	VARCHAR(MAX)	NOT NULL	Đường dẫn hình ảnh
isDEFAULT	BIT	NOT NULL	Biến kiểm tra ảnh mặc định

3.3.3.3. Bảng GAMEROUND:

GAMEID	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
GAMEID	VARCHAR(6)	PRIMARY KEY	Mã lượt chơi
PLAYERNAME	VARCHAR(40)	NOT NULL	Tên người chơi
PLAYERID	VARCHAR(6)	NOT NULL	Mã người chơi
PIECES	VARCHAR(5)	NOT NULL	Kích thước lưới ảnh
PLAYTIME	VARCHAR(8)	NOT NULL	Thời gian chơi
PLAYDATE	SMALLDATETIME	NOT NULL	Thời điểm chơi

3.4.4.Dữ liệu mẫu cho các bảng:

3.3.4.1. Bảng PLAYER

	PLAYERID	PLAYERNAME	PLAYERPASSWORD
1	000000	SlideFun	slidefun
2	000001	BAO	bao123@qer
3	000002	HOAI	hoai123
4	000003	Kass	kasinmidair1207
5	000004	Namnguyen9743	NguyenDinhHoaiNam@137

3.3.4.2. Bảng PICTURE

	PICNAME	PICPATH	PLAYERID	isDEFAULT
1	babystar	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000001babystar.png	000001	0
2	cutegirl	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000003cutegirl.png	000003	0
3	gloss	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000003gloss.png	000003	0
4	InTheCloud	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000004InTheCloud.png	000004	0
5	moon	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000001moon.png	000001	0
6	musicGhost	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000003musicGhost.png	000003	0
7	pic1	pack://application:../Assets/picture/pic1.png	000000	1
8	pic2	pack://application:../Assets/picture/pic2.png	000000	1
9	pic3	pack://application:../Assets/picture/pic3.png	000000	1
10	pic4	pack://application:../Assets/picture/pic4.png	000000	1
11	pic5	pack://application:../Assets/picture/pic5.png	000000	1
12	pic6	pack://application:../Assets/picture/pic6.png	000000	1
13	pic7	pack://application:../Assets/picture/pic7.png	000000	1
14	pic8	pack://application:../Assets/picture/pic8.png	000000	1
15	pic9	pack://application:../Assets/picture/pic9.png	000000	1
16	picA	pack://application:../Assets/picture/picA.png	000000	1
17	picB	pack://application:../Assets/picture/picB.png	000000	1
18	picC	pack://application:../Assets/picture/picC.png	000000	1
19	puzzle	D:\Intuitive_programming_Final_Project\src\Final_Project\PuzzleGame\Assets\picture\000004puzzle.png	000004	0

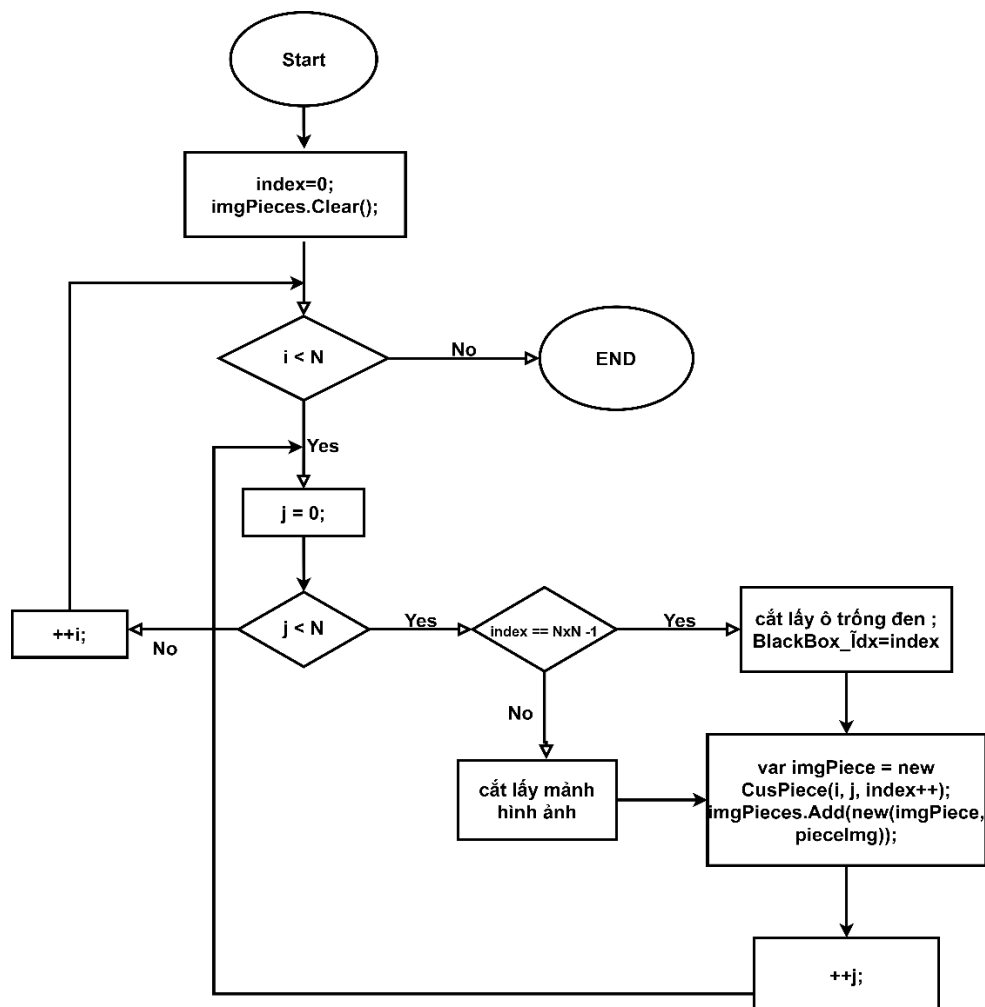
3.3.4.3. Bảng GAMEROUND

	GAMEID	PLAYERNAME	PLAYERID	PIECES	PLAYTIME	PLAYDATE
1	000001	Kass	000003	3 x 3	00:03:45	2024-12-25 14:30:00
2	000002	BAO	000001	4 x 4	00:05:30	2024-12-26 09:15:00
3	000003	HOAI	000002	3 x 3	00:07:50	2024-12-15 16:45:00
4	000004	Kass	000003	5 x 5	00:30:15	2024-12-13 11:00:00
5	000005	BAO	000001	6 x 6	01:03:12	2024-12-21 19:30:00
6	000006	BAO	000001	5 x 5	00:35:03	2024-12-13 19:30:00
7	000007	Kass	000003	4 x 4	00:03:20	2024-12-05 13:20:00
8	000008	BAO	000001	7 x 7	02:00:00	2024-12-29 19:30:00
9	000009	BAO	000001	9 x 9	02:50:05	2024-12-16 22:24:00
10	000010	Namnguyen9743	000004	5 x 5	00:27:35	2024-12-25 14:30:00
11	000011	Namnguyen9743	000004	9 x 9	01:01:49	2024-12-25 14:55:00
12	000012	Namnguyen9743	000004	3 x 3	00:00:45	2024-12-25 14:31:00
13	000013	HOAI	000002	8 x 8	00:57:50	2024-12-13 22:35:00
14	000014	HOAI	000002	5 x 5	00:04:50	2024-12-11 16:56:00
15	000015	Kass	000003	7 x 7	00:17:50	2024-12-22 23:17:00
16	000016	HOAI	000002	6 x 6	00:03:50	2024-12-29 01:25:00

3.4. Thuật toán:

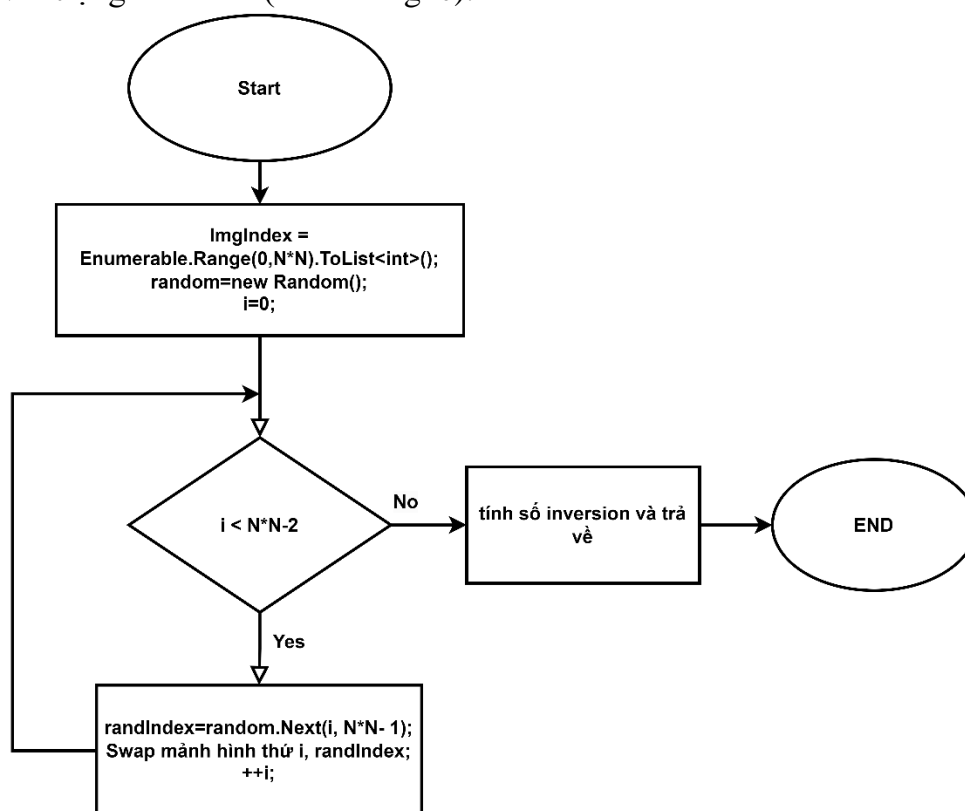
3.4.1. Chia nhỏ ảnh thành các mảnh & tạo ma trận trò chơi:

- Chia hình thành các mảnh nhỏ theo kích thước lưới được chọn trước đó, mỗi mảnh được đánh dấu số thứ tự trong ảnh gốc và vị trí của chúng trong lưới. Trong lưới sẽ có một ô trống để có thể di chuyển các mảnh (mảnh cuối cùng).
- Cấu trúc trò chơi có thể được mô phỏng bởi một ma trận vuông chứa các mảnh hình 2D, khi chuyển về dạng mảng 1 chiều ta sẽ có một mảng chứa các mảnh được đánh số từ $0 \rightarrow (n*n-1)$.



3.4.2. Khởi tạo trạng thái bắt đầu:

- Trạng thái ban đầu sẽ được khởi tạo bằng thuật toán Fisher-Yates để đảm bảo tính ngẫu nhiên, tránh trường hợp trạng thái bắt đầu quá gần với trạng thái đích (hình ảnh gốc).

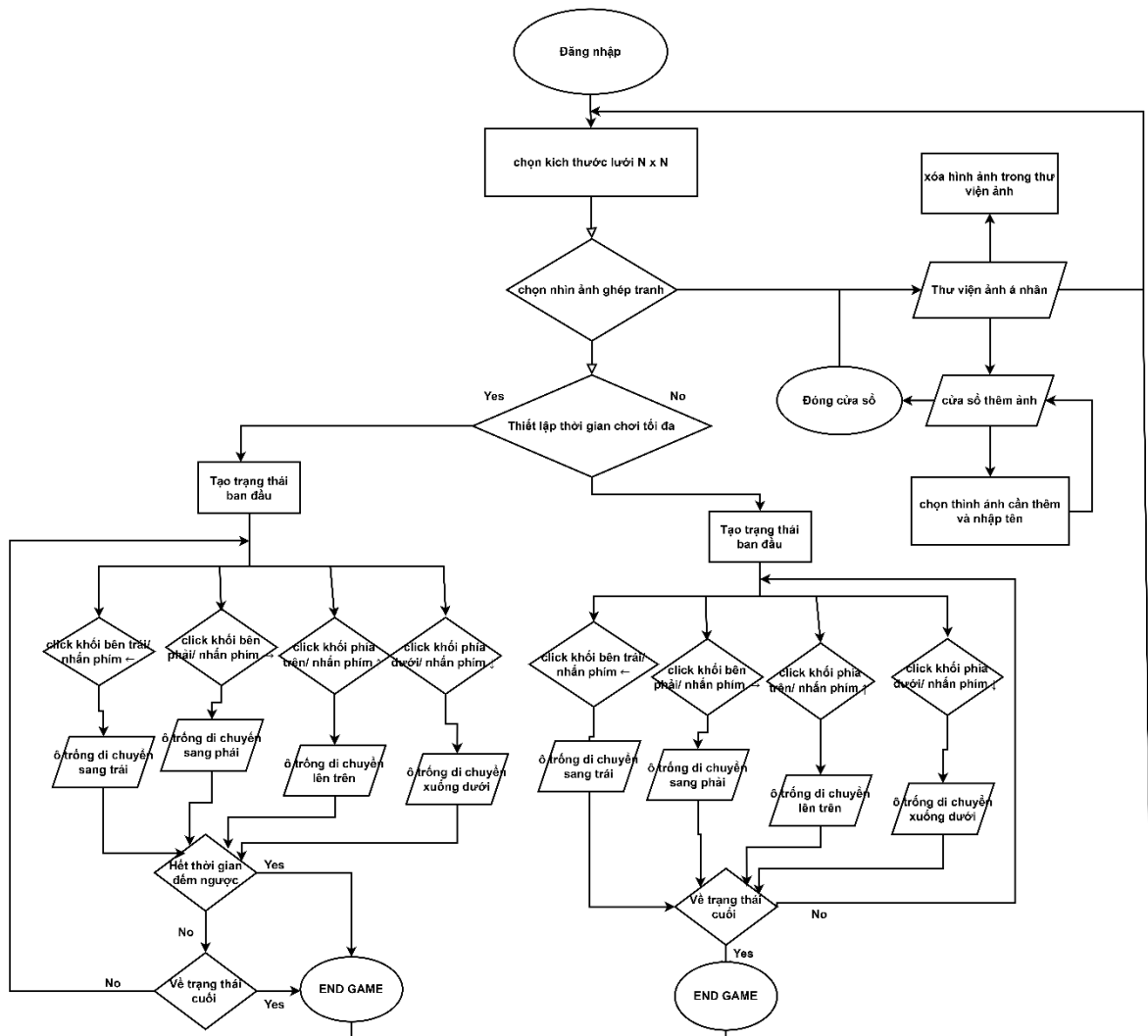


- Sau đó kiểm tra tính giải được của trạng thái khởi đầu, có thể dùng thuật toán MergeSort để tính số nghịch đảo (inversions) trong ma trận từ đó có thể kiểm tra các điều kiện ràng buộc và chuyển đổi về trạng thái có thể giải được

3.4.3. Thuật toán kiểm tra trạng thái trò chơi và bộ đếm giờ:

Sau mỗi giây kể từ khi bắt đầu game, bộ đếm giờ sẽ bắt đầu đếm ngược hoặc đếm tiến (nếu không đặt thời gian chơi giới hạn). Với mỗi lượt di chuyển các mảnh, hàm chứa thuật toán dùng để kiểm tra trạng thái game sẽ được gọi thực thi đảm bảo kiểm tra trạng thái kết thúc game chính xác.

3.5. Sơ đồ phân tích trò chơi:



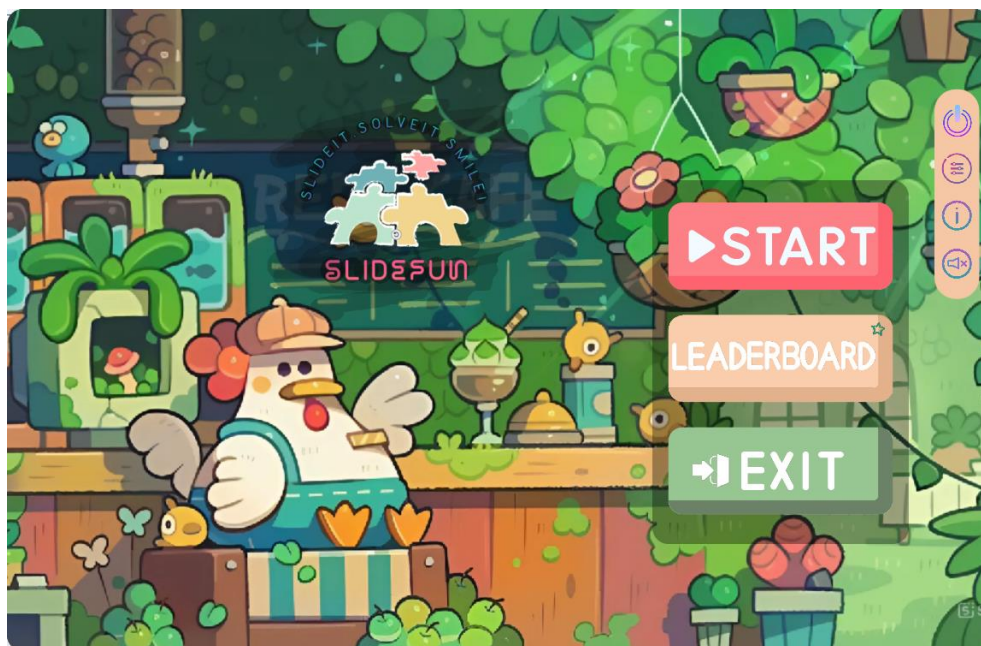
Chương 4: XÂY DỰNG ỨNG DỤNG

Ứng dụng trò chơi sau khi cân nhắc đã sử dụng đến pattern MVVM, trong phần này sẽ chỉ đề cập tới các phần quan trọng của đề án.

4.1. Thiết kế Giao diện :

4.1.1. Trang menu chính:

a) Giao diện thiết lập game:



b) Mô tả chức năng:

Giao diện chính của trò chơi bao gồm các button để bắt đầu game (Start button) và các button khác với các chức năng khác nhau.

4.1.2. Trang đăng nhập/đăng xuất:

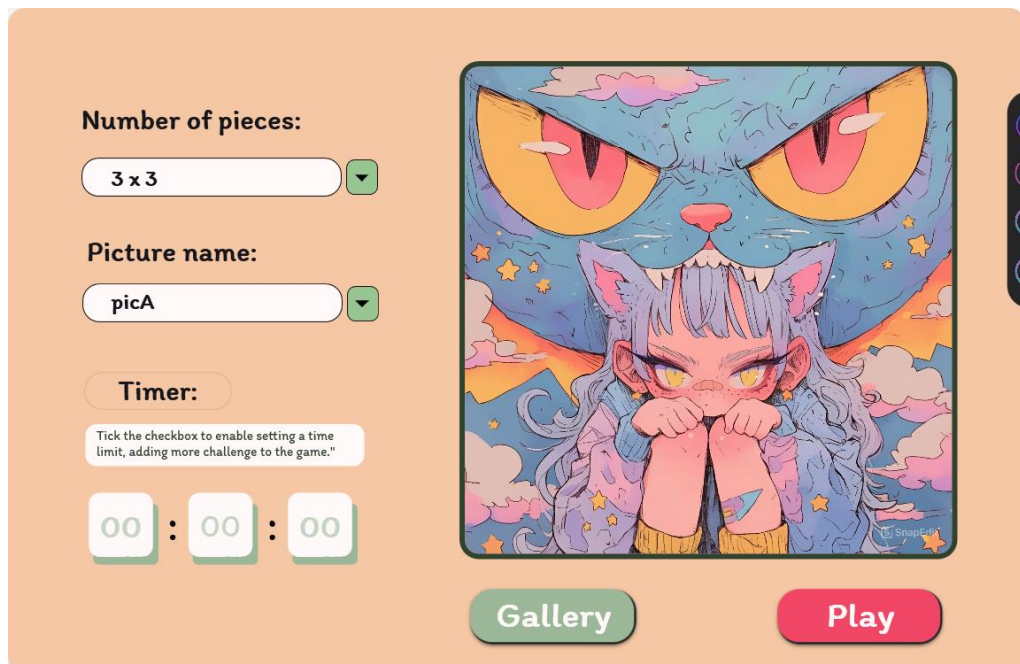
a) Giao diện thiết lập game:

b) Mô tả chức năng:

- Người chơi nhập tên và mật khẩu để đăng nhập tài khoản chơi.
- Nếu chưa có tài khoản thì nhập tên và mật khẩu , sau đó click button Add Player.
- Click button Enter để đăng nhập.

4.1.3. Trang thiết lập game:

a) Giao diện thiết lập game:



Number of pieces:

3 x 3

Picture name:

picA

Timer:

Tick the checkbox to enable setting a time limit, adding more challenge to the game."

00 : 00 : 00

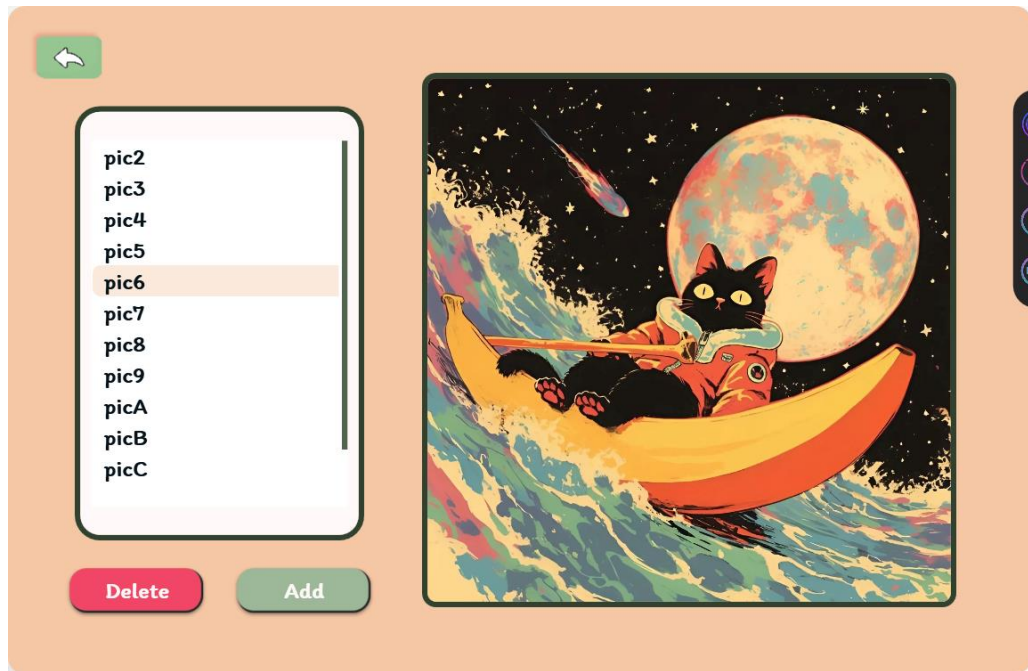
Gallery Play

b) Mô tả chức năng:

- Khi người dùng click vào các combobox, sẽ có một box dropdown cho phép người chơi thiết lập 2 cài đặt cơ bản cho game là hình ảnh và kích thước lưới phân mảnh hình ảnh.
- Người dùng có thể click vào checkbox Timer và điền thời gian đó và các ô textbox bên dưới để giới hạn thời gian chơi tăng độ khó cho game.
- Các button gallery và play sẽ điều hướng cửa sổ trò chơi tới các trang tương ứng là thư viện ảnh cá nhân và giao diện bắt đầu lượt chơi .

4.1.4. Trang thư viện ảnh cá nhân:

a) Giao diện thư viện ảnh:

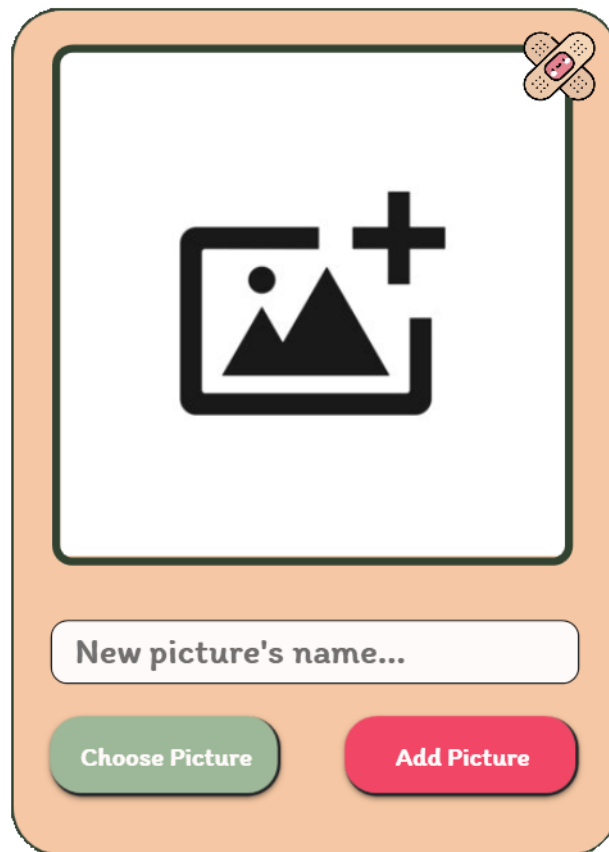


b) Mô tả chức năng:

- Hiển thị kho ảnh cá nhân với từng người chơi bao gồm các hình ảnh mặc định cho trước của ứng dụng và các hình ảnh đã được người chơi thêm vào .
- Hiển thị cửa sổ preview cho phép xem hình ảnh tương ứng .
- Button Delete cho phép xóa hình ảnh có trong thư viện ảnh (trừ hình ảnh mặc định).
- Button Add cho phép mở cửa sổ thêm hình ảnh vào thư viện ảnh cá nhân. Hình ảnh thêm vào trong thư viện ảnh sẽ không bị mất khi ở động lại ứng dụng.

4.1.5. Cửa sổ thêm ảnh:

a) Giao diện cửa sổ thêm ảnh:

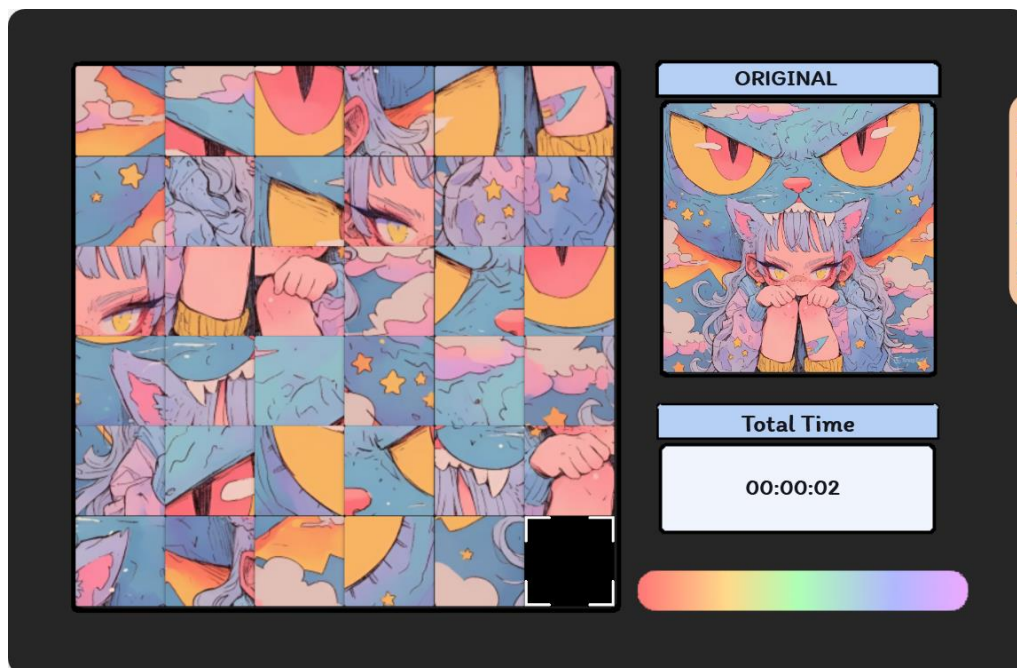


b) Mô tả chức năng:

- Khi người chơi click vào khung hình ảnh, button Choose Picture sẽ đều mở FileDialog cho phép chọn hình ảnh thêm vào thư viện ảnh.
- Người chơi nhập tên hình ảnh để lưu vào thư viện ảnh.
- Button Add Picture để thêm hình ảnh vào thư viện ảnh.

4.1.6. Trang chơi game chính:

a) Giao diện chơi game chính:

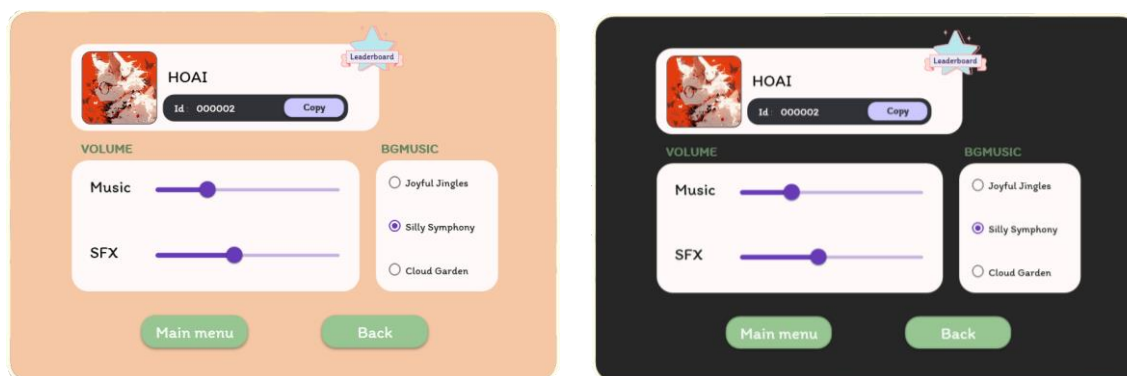


b) Mô tả chức năng:

- Người chơi có thể thao tác trò chơi bằng bàn phím hoặc chuột để di chuyển các mảnh cắt hình ảnh về đúng vị trí.
- Hình ảnh gốc sẽ được hiển thị ở góc trên bên phải của cửa sổ hỗ trợ người chơi.
- Bộ đếm thời gian sẽ đếm tiến nếu người chơi không đặt thời gian giới hạn và ngược lại sẽ đếm ngược.

4.1.7. Cửa sổ cài đặt (tích hợp với cửa sổ chính) :

a) Giao diện cửa sổ cài đặt:



b) Mô tả chức năng:

- Hiển thị thông tin người chơi như tên, id .
- Button Copy cho phép người chơi sao chép Id người chơi Clipboard .
- Bảng BGMusic cho phép lựa chọn nhạc nền cho trò chơi, bên cạnh đó bảng Volume cũng cho phép điều chỉnh âm lượng nhạc nền và hiệu ứng âm thanh.
- Khi người chơi click vào button MainMenu sẽ xuất hiện cửa sổ xác nhận đăng xuất, nếu người chơi click yes thì sẽ điều hướng tới cửa sổ .
- Button back sẽ đóng cửa sổ settings.

4.1.8. Cửa sổ kết thúc lượt chơi:

a) Giao diện cửa sổ kết thúc lượt chơi:



Win

Lose

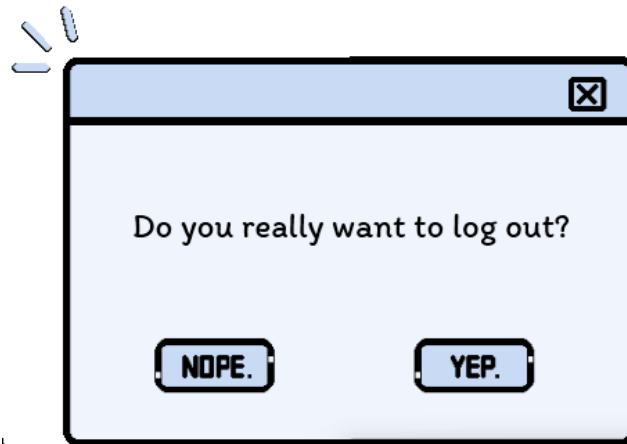
b) Mô tả chức năng:

- Hiển thị trạng thái kết thúc game (Win, Lose) thông qua đoạn text ở phía trên.

- Các button LeaderBoard, Play Again sẽ điều hướng cửa sổ tới các trang tương ứng là bảng xếp hạng và trang thiết lập game để bắt đầu lượt chơi khác.

4.1.9. Dialog thông báo:

a) Giao diện Dialog:



b) Mô tả chức năng:

- Hiện thị đoạn text với mục đích thông báo hoặc xác nhận .Có 2 loại button được sử dụng trong Dialog này :YesNo, OK.

4.1.10. Trang bảng xếp hạng:

a) Giao diện bảng xếp hạng:



b) Mô tả chức năng:

- Người chơi có thể nhập kích thước cắt lưới vào combobox ở trên để tra cứu thứ hạng.
- Người chơi có thể sử dụng Id của mình, click button Search để tra cứu chỉ thứ hạng ở toàn bộ các loại kích thước lưới.
- Button back để quay về trang trước đó.

4.2. Thiết kế và tạo các class:

4.2.1. Class CusPiece:

```
namespace PuzzleGame.MVVM.Models
{
    public class CusPiece
    {
        int yIndex;    // position follow y axis in GameImage map
        int imgIdx;    // nth of img in whole SrcImage
        int xIndex;    // position follow x axis in GameImage map
        public int ImgIdx{...}
        public int XIndex{...}
        public int YIndex{...}

        public CusPiece(CusPiece _custom){...}
        public CusPiece(int xIndex, int yIndex, int imgIdx){...}

        /// <summary>
        /// swap between 2 CusPiece
        /// </summary>
        /// <param name="p2"></param>
        /// <exception cref="InvalidOperationException"></exception>
        public void SwapPieces(CusPiece p2){...}

        /// <summary>
        /// Check if this CusPiece is in the correct position
        /// </summary>
        /// <returns></returns>
        public bool Match(){...}

        public int CurrentImgIndex(){...}
    }
}
```

- Class **CusPiece** được định nghĩa gồm 2 constructor với mục đích lưu trữ dữ liệu về vị trí **ImgIdx** của mỗi mảnh trong hình ảnh gốc và tọa độ (**xIndex**, **yIndex**) trong lưới hình ảnh.
- Bên cạnh đó, class cũng cung cấp các method : **CurrentImgIndex()** trả về vị trí của mảnh trong danh sách hình ảnh mảng một chiều ,**Match()** để kiểm tra vị trí hiện tại trên lưới hình ảnh có trùng với vị trí gốc không .

4.2.2. Class Connection:

```
namespace PuzzleGame.MVVM.Models
{
    //connection
    public partial class Connection
    {
        public string connStr = "data source=MasinMidair-Dss\\SQLEXPRESS_BEGIN;initial catalog=SLIDING_PUZZLE_DB;trusted_connection=true";
        static string _connStr = "data source=MasinMidair-Dss\\SQLEXPRESS_BEGIN;initial catalog=SLIDING_PUZZLE_DB;trusted_connection=true";
        public SqlConnection Conn = new SqlConnection(_connStr);
        public SqlDataAdapter dataAdapter;
        public DataSet ds = new DataSet();
        public DataTable dt = new DataTable();
    }
}
```

- Class **Connection** cung cấp một phương thức để lấy đối tượng **SqlConnection**, thường được sử dụng trong ứng dụng .NET để thiết lập kết nối đến cơ sở dữ liệu Microsoft SQL Server.

4.2.3. Class Player & Picture:

```
namespace PuzzleGame.MVVM.Models
{
    public class Player
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public string Password { get; set; }
    }
}
```

```
namespace SlideFun.MVVM.Models
{
    public class GameRound
    {
        public string Ranking { get; set; }
        public string PlayerName { get; set; }
        public string PlayerID { get; set; }
        public string Pieces { get; set; }
        public string Time { get; set; }
        public string Date { get; set; }
    }
}
```

```

namespace PuzzleGame.MVVM.Models
{
    public class Picture
    {
        public string Name { get; set; }
        public string Url { get; set; }
        public string PlayerID { get; set; }
        public bool isDefault { get; set; }
    }
}

```

- Các class trên được dùng để lưu thông tin về người chơi, thông tin về các hình ảnh game làm kho ảnh cá nhân của người chơi và thông tin về các lượt chơi trong database.

4.2.4. Class GameModel:

```

private static object key = new object();
private static volatile GameModel _instance;
public static GameModel Instance
{
    get
    {
        if (_instance == null )
        {
            lock (key)
            {
                _instance = new GameModel();
            }
        }

        return _instance;
    }
}

```

- Class **GameModel** được thiết kế theo mô hình Singleton cho phép chỉ sử dụng một thể hiện tính duy nhất được tạo xuyên suốt ứng dụng giúp dễ dàng chia sẻ dữ liệu toàn cục, dễ dàng kiểm soát và gỡ lỗi.

```

public double UnitX { get; set; }
public double UnitY { get; private set; }
public int row { get; set; }
public int col { get; set; }
public Player Player { get; set; }
public bool isSetCountDown { get; set; }

public BitmapSource SrcImg, blkBoxImg;
public int BlackBox_Idx;
public int gamePlayBoxX, gamePlayBoxY;
string srcPath;

private GameStatus status;
public GameStatus Status{...}
long playTime;
public long PlayTime{...}

```

- Class này định nghĩa một số biến thành viên công khai, được sử dụng để lưu trữ các cài đặt liên quan đến ứng dụng , một số thuộc tính quan trọng cần chú ý :
 - o **UnitX, unitY**: lưu trữ kích thước mỗi mảnh cần cắt ra từ hình ảnh gốc.

- **srcPath**: đường dẫn tới hình ảnh gốc được chọn để bắt đầu lượt chơi.
- **row, col**: kích thước lưới hình ảnh .
- **isSetCountDown**: kiểm tra xem người chơi có đặt thời gian chơi giới hạn hay không.

```
private GameModel()...
//setting data for a game round.
public void SetData(int r,int c,string imgSrcPath ,long _playTime)....
//resize the original picture and setting needed properties
public void PicDeviding()...
```

- Ngoài constructor , class còn định nghĩa thêm 2 method :
 - **SetData()** : thiết lập các cài đặt dựa người chơi chọn trong phần thiết lập trò chơi, chuẩn bị cho lượt chơi sắp bắt đầu.
 - **PicDeviding()** : thay đổi kích thước ảnh gốc ban đầu sao cho vừa vặn với khung trò chơi trong phần giao diện. Việc sử dụng hàm này cũng Đảm bảo hình ảnh với các định dạng khác nhau, kích thước khác nhau và có DPI khác nhau đều được hiển thị trên UI mà không gặp lỗi .

4.3. Thiết kế các hàm & thuật toán cốt lõi:

4.2.1. Hàm:

a) Điều chỉnh kích thước hình ảnh:

```
public void PicResize()
{
    BitmapImage originalImage = new BitmapImage(new Uri(srcPath, UriKind.RelativeOrAbsolute));

    // caculate to crop Image to filltoUniform
    int originPixelDimension= Math.Min(originalImage.PixelWidth,originalImage.PixelHeight);
    int offsetX = (originalImage.PixelWidth - originPixelDimension) / 2;
    int offsetY = (originalImage.PixelHeight - originPixelDimension) / 2;
    CroppedBitmap croppedImage = new CroppedBitmap(originalImage,
        new Int32Rect(offsetX, offsetY, originPixelDimension, originPixelDimension));

    //resize image to fit diff DPI
    double uniformScale = (double)gamePlayBoxX * originalImage.DpiX / (croppedImage.PixelWidth * 96);

    SrcImg = new TransformedBitmap(croppedImage, new ScaleTransform(uniformScale, uniformScale));

    //set dimension of Piece
    UnitX = SrcImg.PixelWidth / col;
    UnitY = SrcImg.PixelHeight / row;
}
```

- Hàm **PicResize** hỗ trợ việc hiển thị với các định dạng hình ảnh khác nhau, dẫn đến DPI khác nhau cũng như các hình ảnh có kích thước khác nhau.

b) Hàm bắt đầu game:

```

/// <summary>
/// Add Piece into Controls of pnlGamePlaySpace and Playing
/// </summary>
///
1 reference
public void StartGame()
{
    GameModel.Instance.Status = GameStatus.StartGame;
    _imageProcessingService.SplitIntoPieces(imgPieces);
    int inversion;
    do
    {
        inversion = _imageProcessingService.ShufflePieces(imgPieces);

        if (!IsSolvable(inversion))           //check whether it is solvable or not
        {                                     //If not ,make it sovable.
            MakeItSovable(ref inversion);
        }
    }
    while (inversion == 0);

    if (imgPieces == null)
        throw new Exception("Load Image Error");

    MovingFocus();
    _clock.Start();                       //start timer
}

```

- Trạng thái trò chơi được đặt là StartGame khi phương thức StartGame được gọi, đánh dấu sự bắt đầu của trò chơi.
- Gọi thực hiện các hàm SplitIntoPieces chuẩn bị các mảnh ảnh , xáo trộn chúng và kiểm tra tính khả thi của trò chơi thông qua giá trị trả về từ hàm ShufflePieces .
- Kiểm tra tính giải được của trạng thái ban đầu thông qua Isolvable .Bên cạnh đó sử dụng hàm MakeItsovable đảm bảo trạng thái ban đầu của lưới hình ảnh là có thể giải được.
- Bắt đầu lượt chơi bằng cách khởi động bộ đếm giờ.

c) Hàm kiểm tra trạng thái giải ban đầu có thể giải được & hàm thay đổi trạng thái.

```

private bool IsSolvable(int inversion)
{
    if (GameModel.Instance.col * GameModel.Instance.row % 2 == 0)
    {
        return (GameModel.Instance.row - 1 +inversion ) % 2!= 0;
    }
    return inversion % 2 == 0;
}

void MakeItSovable( ref int inversion)
{
    for(int i = 0; i < imgPieces.Count-1; ++i)
    {
        if (imgPieces[i].CusPiece.ImgIdx > imgPieces[i + 1].CusPiece.ImgIdx)
        {
            imgPieces[i].SwapCusVM(imgPieces[i+1]);
            if (IsSolvable(--inversion)) return;           //check whether it is solvable or not
        }
    }
}

```

- Như đã nói ở trên, với N là kích thước lưới hình ảnh , ta có thể quy về bài toán N-puzzle và một trạng thái của N-puzzle có thể đưa về trạng thái đích khi:

- N lẻ : Một nước đi hợp lệ sẽ làm thay đổi **inversions** trên lưới bởi một số chẵn. Tức là **inversions** bảo toàn tính chẵn lẻ khi hoán đổi vị trí blank box với một ô bất kì kề với nó. Trạng thái đích có **inversions** là một số chẵn ($=0$) \Rightarrow có thể giải được nếu **inversion** là một số chẵn
- N chẵn: ngược lại ,trong trường hợp này tính chẵn lẻ của số lượng **inversions** không được bảo toàn. Tuy nhiên, tính chẵn lẻ của tổng số lượng **inversions** vị trí hàng chứa blank box (0,1,2,...) được bảo toàn. Mỗi nước đi hợp lệ sẽ làm thay đổi tổng này bởi một số chẵn. Trạng thái đích có tổng này là một số lẻ ($=N-1$) \Rightarrow có thể giải được nếu tổng trên là một số lẻ.
- Nếu **IsSolvable** trả về **false**, hàm **MakeItSolvable** được gọi thực thi hoán đổi vị trí 2 ô liền kề nhau đầu tiên \Rightarrow thay đổi tính chẵn lẻ của **inversions**, tạo trạng thái ban đầu có thể giải được.

d) Hàm điều khiển trò chơi:

```
private void Game_Control(string sKey)
{
    if (GameModel.Instance.Status != GameStatus.StartGame)
        return;
    int moveIndex = -1;
    CusPiece tmp = imgPieces[GameModel.Instance.BlackBox_Indx].CusPiece;
    switch (sKey)
    {
        case "Down":
            MovingPiece(tmp.XIndex - 1, tmp.YIndex, ref moveIndex);
            break;
        case "Up":
            MovingPiece(tmp.XIndex + 1, tmp.YIndex, ref moveIndex);
            break;
        case "Left":
            MovingPiece(tmp.XIndex, tmp.YIndex + 1, ref moveIndex);
            break;
        case "Right":
            MovingPiece(tmp.XIndex, tmp.YIndex - 1, ref moveIndex);
            break;
    }
    if (moveIndex == -1)
        return;

    MovingFocus();

    (imgPieces[GameModel.Instance.BlackBox_Indx]).SwapCusVH(imgPieces[moveIndex]);
    GameModel.Instance.BlackBox_Indx = moveIndex;

    if (UpdateStatus(imgPieces))
        IsWin();
    return;
}
```

- Ta có thể coi việc thực hiện một nước đi hợp lệ là việc di chuyển blank box sang các vị trí kề với nó (4 hướng) .
- Hàm thực hiện tính toán vị trí của các ô ở kề với blank box sau đó hoán đổi nó với ô tương ứng, nếu đó là một vị trí không thể hoán đổi thì kết thúc hàm .
- Sau mỗi nước đi hàm kiểm tra trạng thái là trạng thái đích hay không đều được gọi, đảm bảo phát hiện sự kiện Win game , kết thúc lượt chơi.

e) Các hàm kết thúc trò chơi:

```

public void IsLose()
{
    ReleaseClock();
    if (GameModel.Instance.Status == GameStatus.PreStart) return;
    MusicSystemService.Instance.EndGame_Sound(0);
    EndGameImageSource = "pack://application:,,,/Assets/Imgs/Lose.png";
    EndGameText = "\\Amazing try, Don't sad! Winning takes practice, and you're on the right path!\\\\";
    GameModel.Instance.Status = GameStatus.EndGame;
    IsEndGameVisible = false;
}

public void IsWin()
{
    ReleaseClock();
    if (GameModel.Instance.Status == GameStatus.PreStart) return;
    MusicSystemService.Instance.EndGame_Sound(1);
    if (GameModel.Instance.isSetCountDown)
    {
        time = "Time: " + TimeSpan.FromSeconds(timeSet - GameModel.Instance.PlayTime).ToString(@"hh\:mm:ss");
    }
    else
        time = "Time: " + LastGameTimeStr;

    SaveGameRoundData();
    EndGameImageSource = "pack://application:,,,/Assets/Imgs/Win.png";
    EndGameText = $"\\You completed the round in just {time}!You're a true champion! Keep up the amazing work!\\\\";
    GameModel.Instance.Status = GameStatus.EndGame;
    IsEndGameVisible = false;
}

```

- Các hàm isLose thực hiện chức năng chuyển đổi trạng thái của trò chơi và kết thúc một lượt chơi, đồng thời kích hoạt biến được coi như trigger để hiện pannel kết thúc lượt chơi
- Hàm IsWin có chức năng tương tự như isLose nhưng sẽ lưu dữ liệu lượt chơi của người chơi để phục vụ cho bảng xếp hạng

4.2.2. Thuật toán :

- Các thuật toán sau đây được cung cấp bởi các class service giúp dễ dàng tái sử dụng mã, gỡ lỗi.

a) Thuật toán cắt hình ảnh (ImageProcessingService):

```

public void SplitIntoPieces(ObservableCollection<CusPieceViewModel> imgPieces)
{
    int index = 0; //position of piece in original picture
    int i, j;
    imgPieces.Clear();

    for (i = 0; i < GameModel.Instance.row; ++i)
    {
        for (j = 0; j < GameModel.Instance.col; ++j)
        {
            CroppedBitmap pieceImg;
            // assign Image source of the last PICTURE's piece is a blank box
            if (index == GameModel.Instance.row * GameModel.Instance.col - 1)
            {
                pieceImg = new CroppedBitmap(
                    GameModel.Instance.blackBoxImg,
                    new Int32Rect(0, 0,
                        (GameModel.Instance.gamePlayBoxX / GameModel.Instance.col),
                        (GameModel.Instance.gamePlayBoxY / GameModel.Instance.row)
                    ));
                //save the blank box position
                GameModel.Instance.BlackBox_Indx = index;
            }
            else
            {
                Int32Rect rct = new Int32Rect((int)(GameModel.Instance.UnitX * j),
                    (int)(GameModel.Instance.UnitY * i), (int)GameModel.Instance.UnitX, (int)GameModel.Instance.UnitY);
                pieceImg = new CroppedBitmap(GameModel.Instance.SrcImg, rct);
            }
            //Create a piece & add it to Game play Grid.
            var imgPiece = new CusPiece(i, j, index++);
            imgPieces.Add(new (imgPiece, pieceImg));
        }
    }
}

```

- Hàm splitIntoPiece được thiết kế để chia một bức ảnh thành nhiều mảnh nhỏ, theo dạng lưới với số hàng (row) và cột (col) đã được định nghĩa trong GameModel. Mỗi mảnh nhỏ là một phần của

bức ảnh ban đầu, mảnh cuối được gán ImageSource là hình ảnh có nền đen đại diện cho **blankBox**.

- Từ hình ảnh được cắt ra sẽ tạo thành một thể hiện của lớp **CusPieceViewModel**. Ban đầu các mảnh được giữ nguyên vị trí, lấy (i, j) là tọa độ của mảnh trong lưới hình ảnh.

b) Thuật toán xáo trộn hình ảnh (ImageProcessingService):

```
1 reference
public int ShufflePieces(ObservableCollection<CusPieceViewModel> imgPieces)
{
    // Create a list of indices from 0 to row * col - 1 to track the positions of the pieces.
    // This list will be used later to count inversions after the shuffle.
    List<int> ImgIndex = Enumerable.Range(0, GameModel.Instance.row * GameModel.Instance.col).ToList<int>();

    Random random = new Random();
    for (int i = 0; i < imgPieces.Count - 2; ++i)
    {
        //random a position between i and number of pieces -1
        int randIndex = random.Next(i, imgPieces.Count - 1);

        //swap 2 Piece VM
        imgPieces[i].SwapCusVM(imgPieces[randIndex]);
        //Swap 2 index in ImgIndex
        Swap(ImgIndex, i, randIndex);
    };

    // remove the index of blank box .The blank box position is the last.
    ImgIndex.Remove(GameModel.Instance.BlackBox_Index);

    return InversionCountingService.CountInversions(ImgIndex); //counting inversions and return it.
}
```

- Hàm **ShufflePieces** thực hiện việc xáo trộn các mảnh trong lưới tạo trạng thái ban đầu một cách ngẫu nhiên thông qua **Fisher-Yates algorithm** (Knuth shuffle algorithm).
- **Fisher-Yates algorithm** là một thuật toán tương đối phổ biến để xáo trộn vị trí các phần tử của mảng. Thuật toán duyệt qua mảng và tại vòng lặp thứ i sẽ sinh ngẫu nhiên ra chỉ số **randIndex** từ $i \rightarrow n-2$, sau đó hoán đổi phần tử thứ i và **randIndex**. Từ đó đảm bảo mỗi phần tử có xác suất bằng nhau để xuất hiện ở mỗi vị trí, tránh trường hợp trạng thái ban đầu được khởi tạo quá gần với trạng thái đích (hình ảnh gốc).
- Sau cùng sẽ trả về số **inversions** (số cặp hoán vị) thông qua hàm **CountInversions** và kết thúc hàm.

c) Thuật toán đếm số Inversions (InversionCountingService):

```
1 reference
public class InversionCountingService
{
    1 reference
    public static int CountInversions(List<int> ls)
    {
        List<int> temp = new List<int>(ls.Count);
        return MergeSortAndCount(ls, 0, ls.Count - 1);
    }

    3 references
    private static int MergeSortAndCount(List<int> ls, int left, int right)
    {
        int mid, inversionCount = 0;
        if (left < right)
        {
            mid = (left + right) / 2;

            inversionCount += MergeSortAndCount(ls, left, mid);
            inversionCount += MergeSortAndCount(ls, mid + 1, right);

            inversionCount += MergeAndCount(ls, left, mid, right);
        }
        return inversionCount;
    }

    private static int MergeAndCount(List<int> ls, int left, int mid, int right)
    {
        int i = left;
        int j = mid + 1;
        int k = left;
        int inversionCount = 0; //start counting
        List<int> temp = new List<int>(ls.Count);

        while (i <= mid && j <= right)
        {
            if (ls[i] <= ls[j]) temp.Add(ls[i++]);
            else
            {
                inversionCount += (mid - i + 1); //counting number > ls[j]
                temp.Add(ls[j++]);
            }
        }
        while (i <= mid) { temp.Add(ls[i++]); }
        while (j <= right) { temp.Add(ls[j++]); }

        for (i = left; i <= right; i++)
        {
            ls[i] = temp[i - left];
        }

        return inversionCount;
    }
}
```

- Ứng dụng thuật toán **MergeSort** để khai đếm đảo ngược(inversion) thông qua danh sách vị trí các mảng được truyền vào.
- Khi gộp hai mảng con, nếu phần tử ở mảng bên trái lớn hơn phần tử ở mảng bên phải ($arr[i] > arr[j]$), thì tất cả các phần tử còn lại ở mảng bên trái đều lớn hơn phần tử hiện tại của mảng bên phải. Do đó, số lượng đảo ngược là số phần tử còn lại ở mảng bên trái.
- Tổng số đảo ngược của danh sách vị trí các mảng được trả về.

4.4. Hệ thống âm thanh với **MusicSystemService**:

- Hệ thống âm thanh được cung cấp thông qua **MusicSystemService**. Service này cho phép phát các âm thanh liên quan tới nhạc nền, hiệu ứng âm thanh(click button, endgame sound).

```
private static volatile MusicSystemService _instance;
public static MusicSystemService Instance
{
    get
    {
        if (_instance == null)
        {
            _instance = new MusicSystemService();
        }

        return _instance;
    }
}
```

- Service này cũng được cung cấp với Singleton pattern để dễ dàng quản lý và gọi thực hiện các hàm của service.

```
List<string> bgAudioSources;
List<string> endGameAudioSources;
string sfxAudioSource;
readonly MediaPlayer _sfx;
MediaClock _sfxClock;
int curBgAudio;
readonly MediaPlayer _backgroundMusic;
MediaClock _backgroundClock;
readonly MediaPlayer _endGameMusic;
MediaClock _endGameClock;
```

- Trong WPF, **MediaPlayer** được sử dụng để phát âm thanh hoặc video, trong khi **MediaClock** giúp đồng bộ và kiểm soát thời gian phát. Sự kết hợp này cho phép lập lịch và quản lý playback chính xác, tính ổn định và linh hoạt cao.
- Hệ thống âm thanh của ứng dụng được chia ra làm 2 loại chính:
 - o Nhạc nền: ứng dụng có 3 file .mp3 cho phép thiết kế tính năng music selector cho nhạc nền.
 - o Hiệu ứng âm thanh: button click, winGamestatus, LoseGame status,...

```

public void PlayBTN_ClickSound()...
public void EndGame_Sound(int endGameStatus)...
public void ChangeBackgroundMusic(int bgIndex)...
public void Dispose()...

/// <summary>
/// Extract the embedded resource to temporary file
/// </summary>
/// <param name="resourcePath"></param>
/// <returns></returns>
private string ExtractEmbeddedResource(string resourcePath)...

public void SetVolume(AudioType audioType, double bgVolume, double sfxVolume)...
```

- Bên cạnh đó **MusicSystemService** cũng cung cấp các dịch vụ để quản lí âm lượng, thay đổi nhạc nền, phát hiệu ứng âm thanh,...

4.5. Các Service thao tác với Database:

4.6.5. LoadPictureListService:

```

public class LoadPictureListService
{
    private static volatile LoadPictureListService _instance;
    public static LoadPictureListService Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new LoadPictureListService();
            }

            return _instance;
        }
    }

    public ObservableCollection<Picture> Piclist = new ObservableCollection<Picture>();
    Connection connection = new Connection();

    // Reload the Piclist and get data from database
    public void LoadPictureList(string id)...)

    // delete a picture from database
    public bool DeletePicture(Picture pic, string id)...)

    // add a picture from database
    public void AddPicture(Picture newPicture, string playerId)...)
}
```

- **LoadPictureListService** cung cấp các phương thức phục vụ việc thao tác với bảng Picture chứa dữ liệu về các hình ảnh được người chơi thêm vào kho ảnh cá nhân :
 - **LoadPictureList**: Load lại kho hình ảnh cá nhân mỗi khi có sự thao tác với database, cập nhập giao diện,...
 - **DeletePicture/AddPicture**: thêm/xóa hình ảnh.

4.6.6. LeaderBoardService:

```

public class LeaderBoardService
{
    private static volatile LeaderBoardService _instance;
    public static LeaderBoardService Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new LeaderBoardService();
            }
            return _instance;
        }
    }

    public ObservableCollection<GameRound> GameRoundsList = new ObservableCollection<GameRound>();
    Connection connection = new Connection();
    // Find all records in GAMEROUNDTABLE matching inputPieces and playerId or just inputPieces when playerId = null
    public void LoadGameRounds(ObservableCollection<GameRound> gameRounds, string inputPieces, string playerId = null)
    // them gameRound vào table
    public void AddGameRound(GameRound gameRound)
    // đếm số dòng trong table GAMEROUND
    int GetRowCount()
}

```

- Tương tự như **LoadPictureListService**, service này cũng được sử dụng để thao tác với database bảng GAMEROUND lưu trữ lịch sử chơi

4.6. Các Class hỗ trợ:

4.6.1. FocusHelper:

```

public static class FocusHelper
{
    public static readonly DependencyProperty IsFocusedProperty = //register attached property
        DependencyProperty.RegisterAttached("IsFocused", typeof(bool),
            typeof(FocusHelper), new PropertyMetadata(false, OnIsFocusedChanged));

    public static void SetIsFocused(UIElement element, bool value)
    {
        element.SetValue(IsFocusedProperty, value);
    }

    public static bool GetIsFocused(UIElement element)
    {
        return (bool)element.GetValue(IsFocusedProperty);
    }

    private static void OnIsFocusedChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
    {
        if (d is UIElement element && (bool)e.NewValue)
        {
            element.Focus(); //set focus on element
        }
    }
}

```

- sử dụng Attached Dependency Property trong WPF hỗ trợ ràng buộc dữ liệu (data binding), kế thừa giá trị từ cây logic quản lý trạng thái focus của phần tử giao diện người dùng (UIElement). Cụ thể được dùng để lấy focus bàn phím cho trang chơi game chính.
- Attached Property Là thuộc tính được đăng ký trên một lớp nhưng có thể được áp dụng cho bất kỳ phần tử nào.
- Property “IsFocused” được định nghĩa có kiểu dữ liệu bool, giá trị mặc định là false và hàm callback được gọi thực thi khi giá trị của nó thay đổi. Ngoài ra còn có các hàm get, set để lấy giá trị và thiết lập giá trị cho thuộc tính.

Chương 5: KẾT LUẬN

5.1. Các kết quả đạt được của đồ án:

- Xây dựng thành công một game xếp tranh bằng WPF theo mô hình MVVM, đảm bảo tính ngăn nắp và dễ bảo trì trong mã nguồn.
- Đồ án tích hợp được tương đối các tính năng cơ bản của một trò chơi xếp hình.
- Giao diện trực quan, thân thiện.
- Nắm bắt được cách lưu trữ và truy xuất dữ liệu một cách hiệu quả thông qua việc tích hợp sử dụng database.
- Kỹ năng làm việc nhóm được cải thiện,

5.2. Ưu điểm của đồ án:

- Việc áp dụng mô hình MVVM giúp mã nguồn rõ ràng, dễ quản lý và dễ mở rộng.
- Thiết kế WPF cho phép giao diện mượt mà, đẹp mắt với khả năng thay đổi hình ảnh dễ dàng.
- Game cho phép người dùng chọn ảnh bất kỳ từ thư viện cá nhân để làm hình xếp, tăng tính cá nhân hóa, cho phép lựa chọn độ khó, thời gian,...
- Có thể dễ dàng thêm các cấp độ khó hoặc tính năng mới như gợi ý nước đi, 1vs1,...

5.3. Hạn chế của đồ án:

- Hiệu năng xử lý: Với các hình ảnh kích thước lớn, hay hình ảnh được phân thành quá nhiều mảnh, ứng dụng có thể gặp tình trạng chậm trễ trong việc chia nhỏ và xử lý.
- Tính năng hạn chế: Hiện tại, game chỉ hỗ trợ cơ chế xếp hình cơ bản mà chưa có tính năng nâng cao như gợi ý, 1vs1,...
- Giao diện chưa linh hoạt: phần UI của đồ án được thiết kế với Fixed Layout, không thể phóng to hay thu nhỏ cửa sổ ứng dụng một cách linh hoạt.

5.4. Hướng phát triển của đồ án:

- Để tăng cường trải nghiệm người dùng (UX), trò chơi có thể được bổ sung các hiệu ứng chuyển động mượt mà khi người chơi di chuyển các mảnh ghép,.
- Có thể kết nối với cơ sở dữ liệu trực tuyến để người chơi cạnh tranh với bạn bè.
- Phát triển đa nền tảng, điều chỉnh trò chơi để chạy trên các hệ điều hành khác ngoài Windows hoặc mở rộng sang web và di động.
- Chuyển đổi sang giao diện linh hoạt (Responsive UI), đảm bảo thích nghi với các độ phân giải khác nhau, tránh tình trạng giao diện bị vỡ khi thay đổi kích thước cửa sổ.
- Tích hợp thêm AI hoặc thuật toán hỗ trợ gợi ý có thể giúp người chơi giải quyết các tình huống khó khăn, làm tăng sự hài lòng khi trải nghiệm.
- Tối ưu hóa hiệu suất, đặc biệt khi xử lý các bảng ghép lớn hoặc chạy trên thiết bị có cấu hình thấp, giúp trò chơi mượt mà và hấp dẫn hơn.

TÀI LIỆU THAM KHẢO

Algorithm and solution:

https://drive.google.com/file/d/1vwwdycqaiGz1LOm3O__qwMa9STFRgiMi/view

UI design:

<http://materialdesigninxaml.net/home>

https://www.tutorialspoint.com/wpf/wpf_templates.htm

MVVM Model:

<https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>

<https://www.youtube.com/playlist?list=PLA8ZIAM2I03hS41Fy4vFpRw8AdYNBXmNm>