TRƯỜNG ĐẠI HỌC PHENIKAA

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN "*CÔNG NGHỆ .NET*"

<u>Đề tài:</u>

Lập trình xây dựng trò chơi ghép tranh

sử dụng công nghệ Dot Net

GVHD: **ThS. Nguyễn Thành Trung**

Sinh viên thực hiện: **19010044 – Đỗ Minh Vượng**

19010016 – Nguyễn Trung Kiên

19010029 – Nguyễn Thị Thanh

Hà Nội, 24/07/2021

Lời cam kết

Họ và tên nhóm sinh viên:

- Đỗ Minh Vượng
- Nguyễn Trung Kiên
- Nguyễn Thị Thanh

Điện thoại liên lạc: 0379672066

Email: vuong.dm19010044@st.Phenikaa-uni.edu.vn

Lớp: K13 – CNTT Hệ đào tạo: chính quy

Chúng tôi xam kết Bài tập lớn (BTL) là công trình nghiên cứu của nhóm tôi. Các kết quả nêu trong BTL là trung thực, là thành quả của chúng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong BTL – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục Tài liệu tham khảo. Chúng tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế nhà trường.

Hà Nội, ngày 24 tháng 07 năm 2021

Nhóm tác giả BTL

Vượng

Đỗ Minh Vượng

MỤC LỤC:

Tài liệu tham khảo

- "**ListBox Drag & Drop Using Attached Properties**" - Mr. Rudi Grobler
- **Programming WPF**: **Building Windows UI** with **Windows Presentation Foundation** - Kindle edition by Sells, Chris, Griffiths, Ian
- Các hình ảnh từ nhiều nguồn khác nhau trên internet

I. Lời nói đầu

Trong thế giới công nghệ hiện nay, nền công nghệ máy tính phát triển rất nhanh kéo theo sự phát triển của các ngôn ngữ lập trình hiện đại giúp cho các chương trình, phần mềm máy tính của chúng ta ngày càng trở lên mạnh mẽ hơn. Song song với đó là sự tồn tại của các nền tảng lập trình hiện đại, tối ưu nhằm giúp người lập trình viên tiết kiệm thời gian xây dựng, triển khai và chạy các ứng dụng, chương trình của mình.

Việc nắm được nền tảng lập trình cho bản thân rất quan trọng, nó giúp chúng ta định hướng chính xác cách thức tổ chức xây dựng chương trình và sử dụng các ngôn ngữ lập trình thích hợp. Chính vì vậy nên việc lựa chọn công nghệ, nền tảng của người lập trình viên sẽ quyết định mực độ thành công trong công việc của bản thân sau này. Vậy nên trong học phần "Công nghệ .NET" lần này, để nâng cao hiểu biết, kinh nghiệm bản thân cũng như để rèn luyện kỹ năng lập trình và làm việc nhóm, chúng em xin chọn chủ đề "Xây dựng chương trình trò chơi Ghép tranh" để thực hiện.

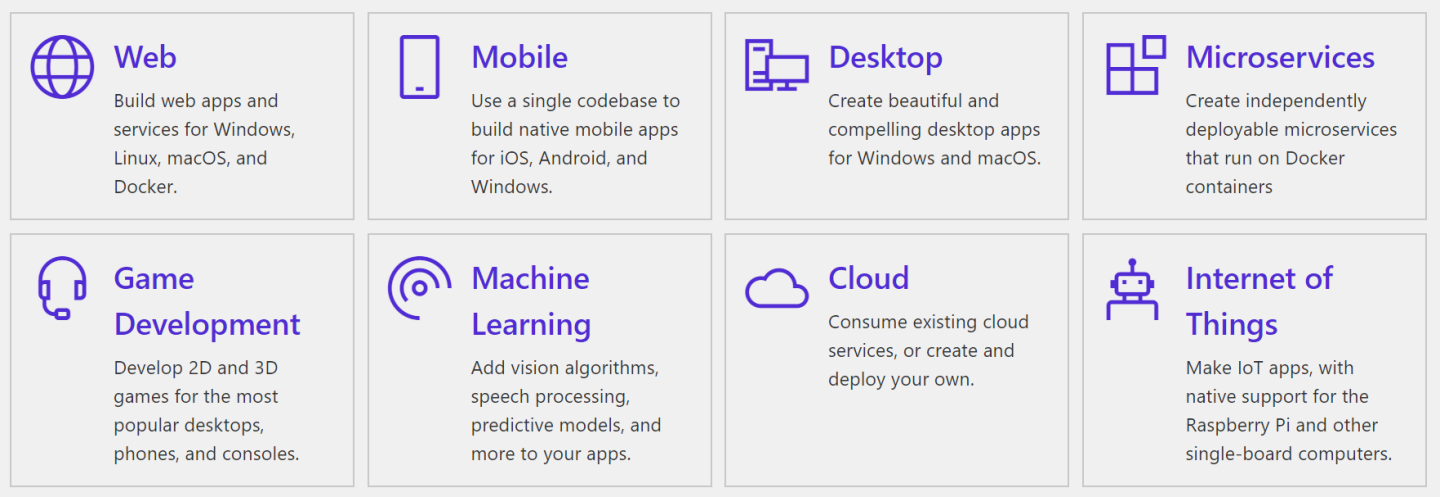
Để hoàn thành được bài tập lớn lần này, chúng em xin chân thành cảm ơn thày giáo đã giảng dạy học phần này là thày Nguyễn Thành Trung, Giảng viên khoa Công Nghệ Thông Tin trường Đại học PHENIKAA – thày đã rất nhiệt tình giảng dạy và hướng dẫn, chỉ bảo chi tiết mặc cho điều kiện giảng dạy còn nhiều khó khăn và bất tiện, chúng em xin cảm ơn thày.

II. Tổng quan về công nghệ .NET và framework WPF

1. Tổng quan về .NET

a. .NET là gì?

Công nghệ .NET (dot NET) là tên gọi chung của một loạt các công nghệ phát triển ứng dụng của Microsoft hiện đang được sử dụng vô cùng rộng rãi trên thế giới. Nó được dùng để xây dựng các ứng dụng cho desktop, web, cloud, các ứng dụng điện thoại, video game, IoT và các chương trình AI.



.NET có thể được dùng để xây dựng ứng dụng cho hầu hết mọi thứ

b. Ưu điểm của công nghệ .NET

Nền tảng này có nhiều ưu điểm nổi bật, sau đây là một số các ưu điểm có thể kể đến:

- **Thư viện lập trình lớn:** .NET sở hữu những thư viện lập trình rất lớn, có khả năng hỗ trợ tối đa cho việc tạo lập, xây dựng các ứng dụng web, truy cập, kết nối các cơ sở dữ liệu, cấu trúc dữ liệu, lập trình giao diện,...
- **Năng suất làm việc cao:** Lập trình, thiết kế ứng dụng với .NET tiết kiệm rất nhiều thời gian bởi nó cung cấp sẵn khá nhiều thành phần dùng trong thiết kế.
- **Biến đổi linh hoạt nhờ Loose Coupling:** .NET được thiết kế, xây dựng với khả năng biến đổi linh hoạt nhờ cấu trúc loose coupling. Điều này đem lại nhiều lợi thế về năng suất.
- **Đa ngôn ngữ:** .NET là nền tảng hỗ trợ cho đa ngôn ngữ. NGười lập trình viên có thể sử dụng nền tảng này để xây dựng các ứng dụng web bằng nhiều ngôn ngữ lập trình khác nhau mà vẫn đảm bảo khả năng tích hợp của nó.
- **Bảo mật cao:** .NET có phần kiến trúc bảo mật được thiết kế theo dạng từ dưới lên. Điều này giúp bảo vệ dữ liệu cũng như các ứng dụng khỏi các mối đe dọa thông qua mô hình bảo mật tân tiến evidence-based.
- **Tận dụng các dịch vụ có sẵn trong hệ điều hành:** Windows sở hữu rất nhiều dịch vụ có khả năng hoạt động trên mọi nền tảng như truy cập dữ liệu, mô hình dạng đối tượng thành phần, giao diện người dùng tương tác, bảo mật tích hợp và cả giám sát giao dịch. .NET tận dụng những dịch vụ này để đơn giản hóa cách sử dụng, giúp lập trình trên nền tảng này dễ dàng hơn.

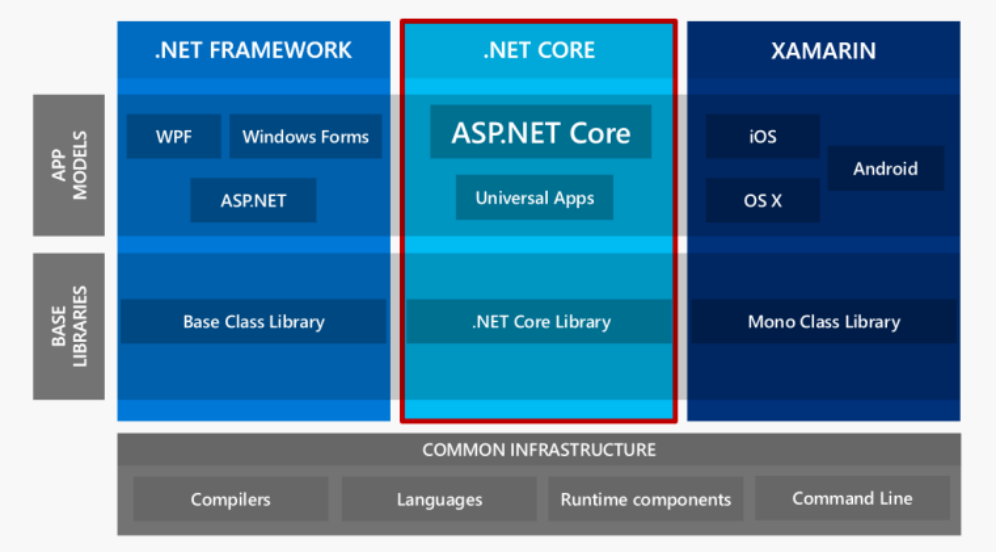
c. Các ngôn ngữ lập trình trên .NET

.NET hỗ trợ nhiều ngôn ngữ lập trình khác nhau, có thể kể đến các ngôn ngữ cần chú ý sau:

- **C#**: là một ngôn ngữ lập trình đơn giản, hiện đại, hướng đối tượng và thuộc loại an toàn.
- **F#**: là ngôn ngữ lập trình đa nền tảng, mã nguồn mở và cung cấp các chức năng lập trình cho .NET. Ngôn ngữ này cũng bao gồm lập trình hướng đối tượng và lập trình mệnh lệnh.
- **Visual Basic**: là một ngôn ngữ dễ tiếp cận với cú pháp đơn giản. Thường được dùng để xây dựng các ứng dụng loại an toàn và hướng đối tượng.

.NET sẽ giúp bạn xử lý phần công việc nặng nhọc thông qua các triển khai đi kèm:

- **.NET Core** là triển khai đa nền tảng của .NET, nó dùng cho các trang web, máy chủ và console app trên Windows, Linux và MacOS.
- **.NET Framework** hỗ trợ các trang web, dịch vụ và ứng dụng dành cho desktop và trên Windows.
- **Xamarin/Mono** là triển khai của .NET được dùng để chạy các ứng dụng trên tất cả các hệ điều hành di động chính.

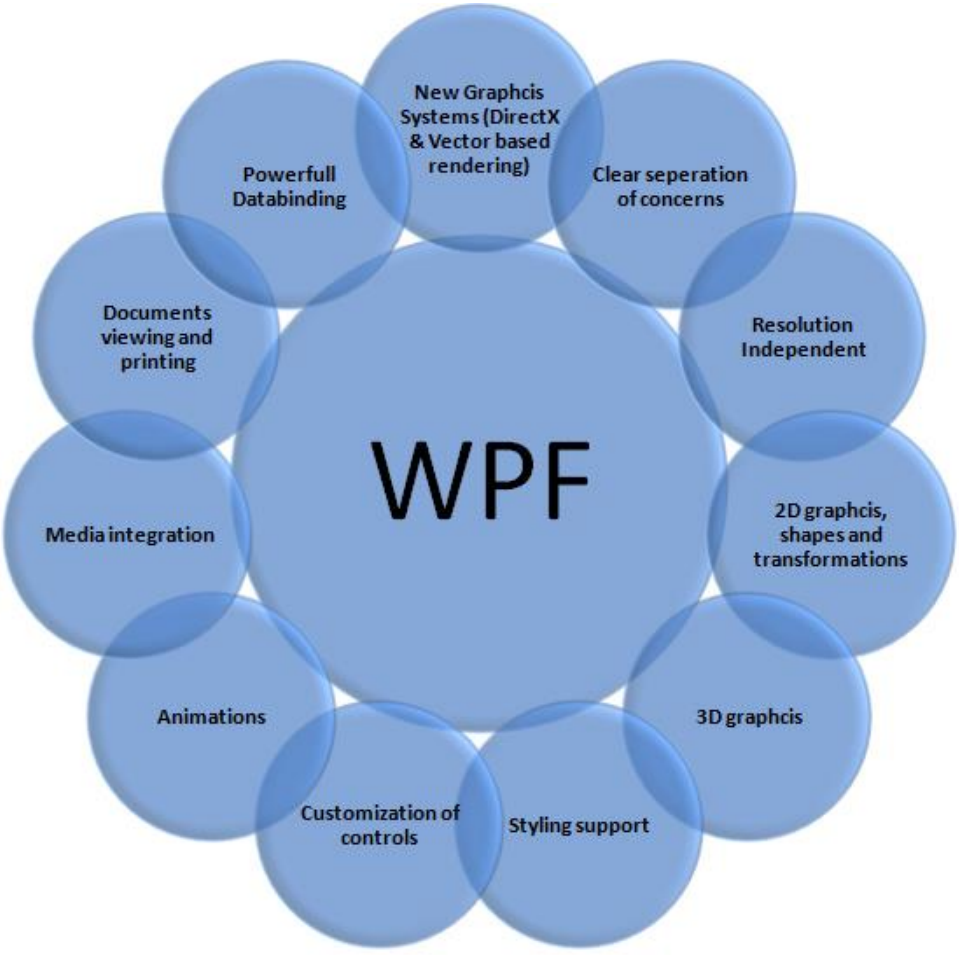


2. Tổng quan về WPF

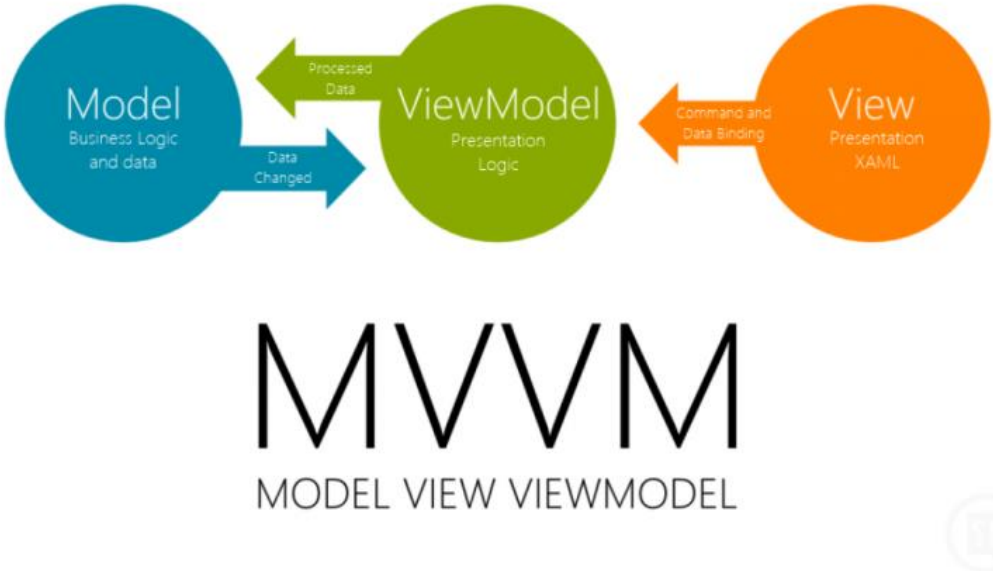
a.WPF là gì?

Windows Presentation Foundation (WPF) ra đời cùng với .NET 3.5 là một framework mới và rất hiện đại dành cho phát triển ứng dụng desktop cho Windows trên .NET Framework. Nó được ưa chuộng và sử dụng rất rộng rãi ngày nay. WPF hoàn toàn đơn giản hóa việc lập trình GUI (giao diện đồ họa), hỗ trợ thiết kế giao diện trực quan, đồng thời nhận được sự hỗ trợ rất tốt từ các hãng thứ ba bên ngoài (Devexpress, Syncfusion, ...) và cộng đồng người sử dụng.

Framework này sử dụng DirectX để tạo ra giao diện với khả năng xử lý đồ họa rất mạnh cho phép tạo ra những giao diện hiện đại, đẹp, mượt mà và những hiệu ứng tân tiến, hiện đại. Ngoài ra, WPF còn sử dụng ngôn ngữ riêng, XAML, để biểu diễn giao diện, cho phép các UI/UX designer hoạt động độc lập với các coder.



Ngoài các mẫu kiến trúc đã quen thuộc từ trước, WPF còn có một mẫu kiến trúc riêng là Model – View – Viewmodel (MVVM).



Mẫu kiến trúc mới MVVM

Cơ chế Data binding của WPF mạnh hơn nhiều so với Winforms, giúp giải quyết vấn đề hiển thị dữ liệu rất hiệu quả, ổn định và an toàn.

b.<u>Một số tính năng nổi trội của WPF</u>

WPF mang đến nhiều ưu việt, giúp cho việc lập trình trở lên đơn giản hơn từ thiết kế giao diện, xử lý dữ liệu giúp cho việc lập trình dễ dàng và mạnh mẽ hơn rất nhiều. Cùng với nền tảng .NET 4.5 hỗ trợ Web API (trước đó, .NET 3.5 dùng WCF, hay Web Service cho .NET trước đó nữa), giúp chúng ta dễ dàng xây dựng các ứng dụng điện toán đám mây.

WPF hỗ trợ hiệu ứng dựa trên thời gian thực thi của chương trình. Các hiệu ứng đơn giản có thể xử lý bằng việc quản lý thời gian chạy, còn các xử lý phức tạp hơn cần đến sự hỗ trợ của lớp Animation.

- Tất cả các thuộc tính của đối tượng trong WPF đều có thể được xử lý để trở nên sinh động hơn.
- Các lớp quản lý hiệu ứng tùy theo loại của thuộc tính được xử lý.

Các tính năng nổi trội của WPF có thể thể hiện thông qua bảng so sánh đơn giản sau:

	Windows Forms	PDF	Windows Forms/ GDI+	Windows Media Player	Direct3D	WPF
Giao diện đồ họa (form và các control)	x					x
On-screen văn bản	x					x
Fixed-format văn bản		x				x
Hình ảnh			x			x
Video và âm thanh				x		x
Đồ họa 2 chiều			x			x
Đồ họa 3 chiều					x	x

Tính năng nổi trội của WPF

Tuy nhiên WPF vẫn tồn tại một số hạn chế nhất định, nhược điểm lớn nhất của WPF là phức tạp, khó học, khó thành thạo (nếu so với Windows Forms). Để phát huy hiệu quả thực sự của WPF đòi hỏi người người lập trình phải sử dụng mẫu thiết kế MVVM xây dựng riêng cho WPF, thành thạo XAML, hiểu cơ chế Data Binding, có khả năng thiết kế, và rất nhiều kỹ thuật nâng cao khác.

b.Thành phần của WPF

WPF tổ chức các chức năng theo một nhóm namespace cùng trực thuộc namespace System.Windows.

Cấu trúc cơ bản của mọi ứng dụng WPF đều gần như nhau. Là ứng dụng Windows độc lập hay là một XBAP, một ứng dụng WPF điển hình bao giờ cũng gồm một tập các trang XAML và phần code tương ứng được viết bằng C# hoặc Visual Basic, còn gọi là các file code-behind.

Mặc dù WPF cung cấp một nền tảng thống nhất để tạo giao diện người dùng, những công nghệ mà WPF chứa đựng vẫn có thể phân chia thành những thành phần độc lập.

III. Trò chơi Ghép tranh

1. Mục tiêu

Tìm hiểu cách xây dựng và chạy một chương trình trò chơi đơn giản sử dụng .NET Framework và áp dụng phương pháp lập trình hướng đối tượng OOP. Qua đó nâng cao hiểu biết và kỹ năng lập trình cho bản thân.

Ta sẽ sử dụng ngôn ngữ C#, chạy và lập trình trên Visual Studio. Lưu ý, ta cũng có thể sử dụng các ngôn ngữ lập trình trong .NET và hỗ trợ OOP khác.

2. Nền tảng

Hệ điều hành: Windows / MAC / Linux

.NET Framework 4.7 trở lên

Microsoft Visual Studio 2017 hoặc 2019

SQLite hoặc các hệ quản trị cơ sở dữ liệu khác.

Ngôn ngữ C#

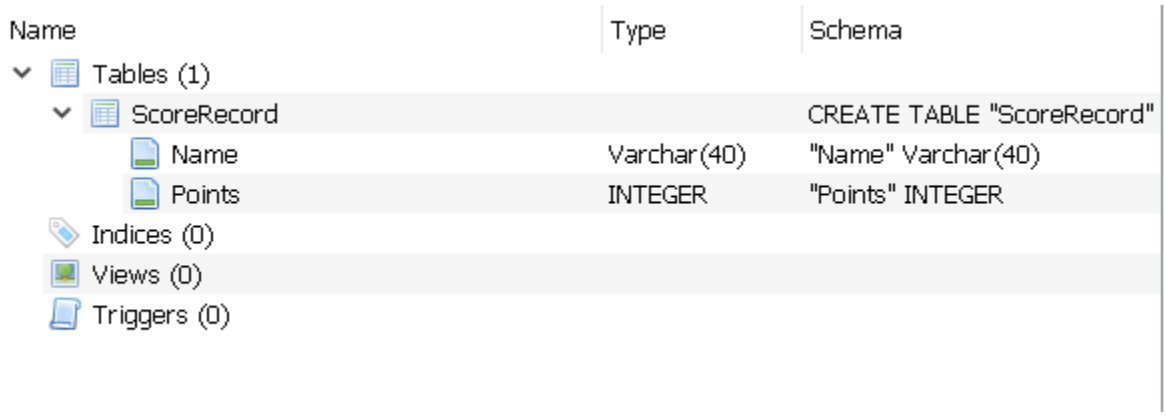
Tải Visual Studio tại đây

Tải SQLite tại đây

Tải .NET Framework tại đây

3. Các bước thực hiện dự án

- Tạo một project mới sử dụng .NET Framework
- Tùy chỉnh SQL Server (nếu sử dụng, tuy nhiên trong bài này sẽ sử dụng SQLite):
  - Chuyển chế độ mặc định sang SQL Server Authentication
  - Chuyển permission người dùng của bạn sang Grant, server role để những mục public và serveradmin
  - Tùy chỉnh kết nối server trong SQL Server Configuration Manager
- Tạo database cho chương trình
  Tên database: Playerdata
    - Bảng ScoreRecord: dùng để lưu tên người chơi và số điểm đạt được
      -Name (varchar)
      -Points (Integer)

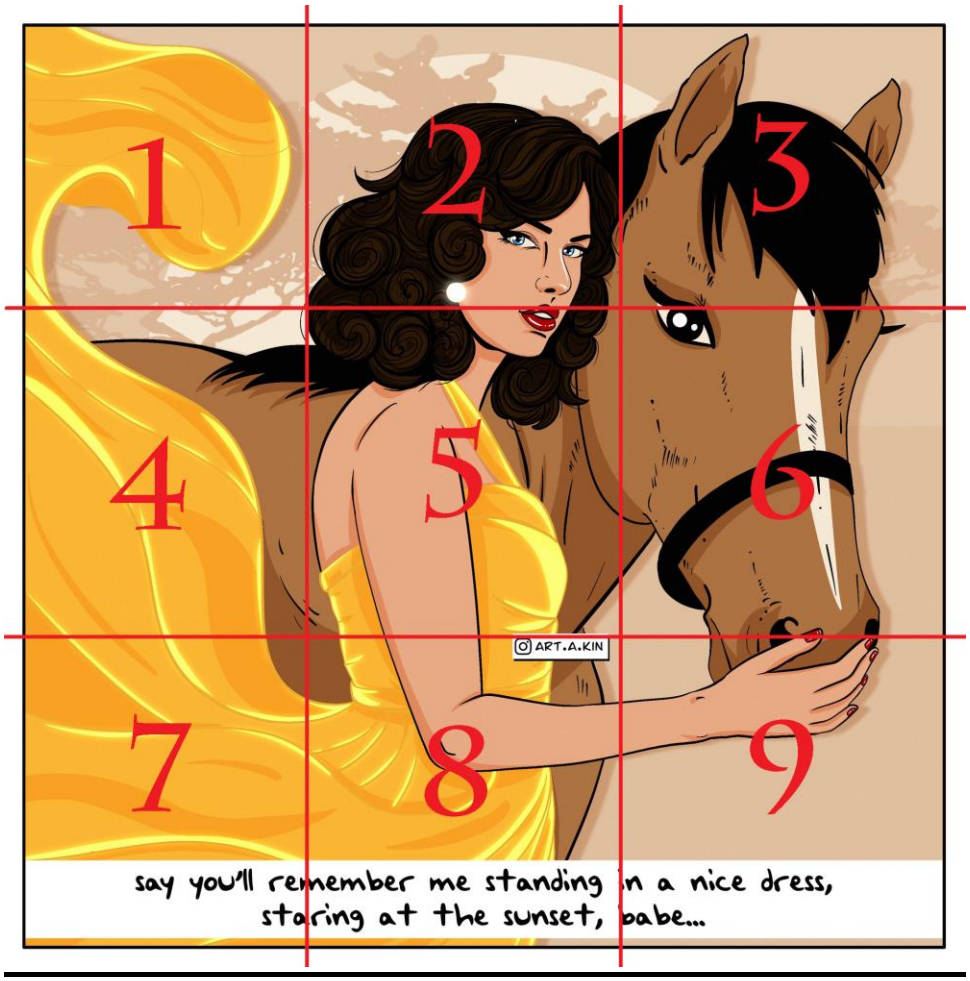


- Xuất databse vừa tạo ra một file .db để đưa vào project
- Chuẩn bị 3 hình ảnh để sử dụng cho 3 màn chơi

4. Xây dựng chương trình

Tổng quan:

- ✓ Giao diện gọn gàng, khoa học, dễ nhìn.
- ✓ Cho phép người chơi chọn độ khó (beginner, advance, expert).
- ✓ Chia hình ảnh thành các phần bằng nhau: chế độ dễ 3x3, trung bình 4x4 và đối với khó là 5x5. Các phần sau khi tách sẽ được đặt tên theo thứ tự.



Minh họa mức độ dễ 3x3

- ✓ Người chơi có thể chọn một ảnh từ máy tính, hệ thống sẽ tự chia ảnh và đặt tên, tạo một màn chơi mới.
- ✓ Màn chơi sẽ gồm 2 phần chính: 1 listbox để chứa các mảnh hình ảnh đã được xáo chộn, các canvas để chứa các mảnh người chơi kéo vào.

✓ Sau khi hoàn thành sẽ tính điểm dựa vào số lần di chuyển của người chơi sau đó lưu tên và điểm người chơi vào file database, các bản ghi này có thể xuất hiện tại mục điểm cao.

Các class chính:

- **PicturePiece: mảnh ảnh (tranh):**
  Thuộc tính và đối tượng:
  - **index**: index của 1 mảnh ảnh (đã cắt)
  - **PuzzleImageSource**: chứa mảnh ảnh
  - **Uristring**: đường dẫn
  - **DragFrom**: để xác định người chơi di chuyển ảnh từ đâu (listbox hoặc canvas)

```
class PicturePiece
{
    //index (stt)
    public int index = -1;

    22 references
    public ImageSource PuzzleImageSource { get; set; }
    //duong dan den hinh anh
    14 references
    public string UriString { get; set; }

    28 references
    public Type DragFrom { get; set; }
}
```

- **Puzzle: class màn chơi:**
  Thuộc tính và đối tượng:
  - **PicPieces**: một collection các mảnh ảnh (đã cắt)
  - **Name**: tên màn chơi

  Phương thức:

  - **Puzzle**: constructor
  - **OnEdit**: dùng để xác định mỗi khi người chơi di chuyển
  - **Initialize**: khởi tạo và xáo chộn các mảnh ảnh tùy theo độ khó
  - **Validate**: kiểm tra xem người chơi đã hoàn thành màn chơi chưa

- **Edited**: event mỗi khi người chơi di chuyển

```
class Puzzle
{

    public ObservableCollection<PicturePiece> PicPieces = new ObservableCollection<PicturePiece>();

    public string name;

    public event EventHandler Edited;

    3 references
    public Puzzle()...

    3 references
    public void OnEdit(EventArgs e)
    {
        if (Edited != null)
            Edited(this, e);
    }

    3 references
    public void Initialize(int chosen)...

    //check
    3 references
    public bool Validate(ObservableCollection<PicturePiece> itemPlacement)...
}
```

- **<u>CustomPuzzle:</u> class tạo màn chơi:**
  Tương tự như class Puzzle nhưng phương thức Initialize sẽ khởi tạo thêm **lvlname** (string) và **newimgForLevel** (BitmapImage) để tạo level và cắt ảnh.

```
class CustomPuzzle
{
    public ObservableCollection<PicturePiece> PicPieces = new ObservableCollection<PicturePiece>();

    public EventHandler Edited;

    //id che do choi
    public static int id ;

    1 reference
    public CustomPuzzle()...

    1 reference
    public void OnEdit(EventArgs e)...

    1 reference
    public void Initialize(int chosen, string lvlname, BitmapImage newimgForLevel)...
    //check
    1 reference
    public bool Validate(ObservableCollection<PicturePiece> itemPlacement)...
}
```

- **<u>PlayingWindow:</u> chế độ chơi dễ:**
  Các chế độ trung bình và khó thì tương tự, chỉ khác index và số lượng canvas.
  Thuộc tính và đối tượng:
    - **puzzle**: đại diện cho màn chơi (class Puzzle), chứa các mảnh tranh (PicPieces) thuộc kiểu ObservableCollection, tên màn chơi (name) và sự kiện mỗi khi người chơi di chuyển mảnh tranh (Edited).
    - **itemPlacement**: một ObservableCollection chứa mảnh tranh và vị trí đúng của mảnh (index của canvas), dùng để xác định người chơi đặt có đúng không.
    - **emptyItem**: một mảnh tranh còn trống, để xác định xem canvas đã có mảnh tranh nào chưa
    - **lbDragSource**: một đối tượng Listbox để chỉ tới Listbox khi người chơi kéo mảnh tranh từ Listbox đó
    - **cvDragSource:** tương tự như **lbDragSource** nhưng để chỉ Canvas
    - **bonk:** hiệu ứng âm thanh khi bấm một nút
    - **playTheme:** nhạc nền trò chơi

  Phương thức:

    - **PlayingWindow**: constructor
    - **itemList_PreviewMouseLeftButtonDown**: xử lý thao tác kéo vào Listbox của người chơi
    - **PzItmCvs_MouseLeftButtonDown**: xử lý thao tác kéo vào Canvas
    - **PuzzleItemDrop:** xử lý thao tác thả mảnh ảnh (vào Canvas) của người chơi
    - **Puzzle_Edited:** xử lý sự kiện Edited
    - **GetDataFromCanvas:** xử lý và lấy dữ liệu một mảnh ảnh khi bị người chơi thay đổi vị trí (kéo từ Canvas)
    - **GetObjectDataFromPoint:** xử lý và lấy dữ liệu một mảnh ảnh khi bị người chơi kéo từ Listbox
    - **Button_Click:** nút hướng dẫn chơi

- **MainWindow_Closing:** dùng để tạm dừng nhạc nền khi đóng cửa sổ
- **Media_ended:** để lặp lại nhạc nền

```
public partial class PlayingWindow : Window
{
    #region objects
    //bonk effect
    public static MediaPlayer bonk = new MediaPlayer();

    //bonk effect
    public static MediaPlayer playTheme = new MediaPlayer();

    //Score
    public static int score = 50000;

    Puzzle puzzle = new Puzzle();
    ObservableCollection<PicturePiece> itemPlacement = new ObservableCollection<PicturePiece>();
    PicturePiece emptyItem = new PicturePiece();

    ListBox lbxDragSource;
    Canvas cvDragSource;
    //ObservableCollection<> stulist = new ObservableCollection<string>();
    #endregion

    //Initialized khoi tao anh va vi tri
    1 reference
    public PlayingWindow()...

    music methods

    thao tac chon & keo

    thao tac tha bang chuot

    Check if wins

    associated item with index (i-> manh picture[i])
    //---------
    get data from draging item

    Tutorial
}
```

Kịch bản ứng dụng:

1. Các mảnh ảnh (tranh) được trộn và đưa vào trong Listbox.
2. Người chơi sẽ kéo mảnh ảnh trong Listbox bên phải vào các Canvas bên trái hoặc kéo từ Canvas này sang Canvas khác, nếu Canvas đích còn trống (emptyItem) thì di chuyển mảnh ảnh và dữ liệu sang, còn nếu không trống thì 2 mảnh tráo đổi vị trí cho nhau.
3. Mỗi khi người chơi thực hiện một thao tác thì một sự kiện sẽ được gọi lên để kiểm tra, nếu các mảnh đã đầy đủ và đúng vị trí thì người chơi hoàn thành màn chơi.
4. Người chơi có thể lưu lại tên và điểm số của mình và xem lại tại mục điểm cao ngoài màn hình chính.

Trong trường hợp người chơi muốn tạo một màn chơi (cần chuẩn bị trước file ảnh):

1. Người chơi chọn một file ảnh tùy ý qua 1 của sổ OpenFileDialog.
2. Hệ thống chuyển ảnh sang BitmapImage sau đó chuyển ảnh thành kích thước vuông và chia đều thành 9 phần bằng nhau.
3. Người chơi đặt tên cho màn chơi, hệ thống sẽ tạo một thư mục trong folder chứa các màn chơi rồi lưu các mảnh ảnh đã cắt vào đó đồng thời đặt tên từ 1-9 (các độ khó khác tương tự).
4. Người chơi bắt đầu chơi như bình thường.

```
//Get the picture
//Split into 9 pieces -> put in itemplacement;
int indx = 1;

int indxInCollection = 0;

for (int i = 0; i < 3; i++) //x
{
    for (int j = 0; j < 3; j++) //y
    {
        CroppedBitmap cp1 = new CroppedBitmap(myBitmap, new Int32Rect(j * 100, i * 100, 100, 100));
        BitmapEncoder newCroped = new PngBitmapEncoder();
        newCroped.Frames.Add(BitmapFrame.Create(cp1));

        //index for one piece
        using (var fileStream = new System.IO.FileStream(NewFolderPath + "\\" + indx + ".png", System.IO.FileMode.Create))
        {
            newCroped.Save(fileStream);
        }
        //Put in collection <>PicturePiece
        this.PicPieces.Add(new PicturePiece());

        this.PicPieces[indxInCollection].index = indxInCollection;

        //this.PicPieces[i].UriString = "Puzzle/" + leveltitle + "/" + (i + 1).ToString() + ".png";

        this.PicPieces[indxInCollection].PuzzleImageSource = cp1;

        indxInCollection++;
        indx++;
    }
}
```

Xử lý cắt ảnh

Khởi tạo:

- Đầu tiên, người chơi sẽ chọn màn chơi tùy theo độ khó thông qua bấm các nút bấm, mỗi nút bấm sẽ đặt một giá trị khác nhau cho một đối số, hệ thống dựa vào đối số này để đưa ra màn chơi mà người chơi yêu cầu:
    - Đối với các màn chơi có sẵn:
        - 1 - màn chơi dễ (beginner)
        - 2 - màn chơi trung bình (advanced)
        - 3 - màn chơi khó (expert)
        - 4 – màn chơi đã tạo

```
public void Initialize(int chosen)
{
    string directorySource = "";
    int numberOfpiece = 0;

    //Beginer
    if (chosen == 1)...

    //Advance
    if (chosen == 2)...

    //Expert
    if (chosen == 3)...

    //Custom
    if (chosen == 4)...

    //tron random
    Random rand = new Random();

    for (int i = 0; i < numberOfpiece; i++)...
}
```

- Đối với phần tạo màn chơi:
  - 1 - Tạo và chơi một màn mới
  - 2 – Chơi lại các màn cũ

```
1 reference
public void Initialize(int chosen, string lvlname, BitmapImage newimgForLevel)
{
    int numberOfpiece = 9;
    if (chosen == 1)
    Cut image method and put it in itemplacement

    #region play old level (co san trong folder Puzzle)
    else if (chosen == 2)...
    #endregion

    random pieces
}
//check
1 reference
public bool Validate(ObservableCollection<PicturePiece> itemPlacement)...
}
```

*Collection đối tượng itemPlacemant (class PlayingWindow) dùng để liên kết Canvas với mảnh ảnh được kéo vào, mặc định của nó là emptyItem.

Thao tác kéo:

Có 2 đối tượng trong XAML có thể kéo là Listbox (nguồn) và Canvas (Đích hoặc có thể là nguồn khi kéo từ Canvas này sang Canvas khác), cả hai sử dụng phương thức **PreviewMouseLeftButtonDown.**

Đối với Listbox, dữ liệu (**PicturePiece**) sẽ được chuyển qua các thao tác kéo và thả từ phương thức **GetObjectDataFromPoint.** Phương thức này sử dụng nguồn kéo và tọa độ chuột trên nguồn kéo làm đối số.

```
1 reference
private object GetObjectDataFromPoint(ListBox dragSource, Point point)
{
    UIElement element = dragSource.InputHitTest(point) as UIElement;

    if (element != null)
    {
        object data = DependencyProperty.UnsetValue;
        while (data == DependencyProperty.UnsetValue)
        {
            data = dragSource.ItemContainerGenerator.ItemFromContainer(element);

            if(data == DependencyProperty.UnsetValue)
            {
                element = VisualTreeHelper.GetParent(element) as UIElement;
            }

            if (element == dragSource)
            {
                return null;
            }
        }
        if (data != DependencyProperty.UnsetValue)
        {
            return data;
        }
    }
    return null;
}
```

Thao tác thả:

Sẽ có 4 trường hợp của thao tác thả:

- Nếu Canvas trống và nguồn kéo là Listbox: mảnh tranh được thả vào.
- Nếu Canvas trống và nguồn kéo là Canvas khác: mảnh tranh được thả vào.
- Nếu Canvas không trống và nguồn kéo là Listbox: mảnh không được thả vảo.
- Nếu Canvas không trống và nguồn kéo là Canvas khác: đổi chỗ 2 mảnh.

Sau khi thả vào Canvas hợp lệ (trường hợp 1, 2 và 4): cập nhật vị trí của mảnh ảnh, xóa khỏi nguồn kéo (trừ trường hợp 4). Đối với trường hợp 4, lấy dữ liệu 2 mảnh, index của nguồn và đích sau đó đổi vị trí cho nhau. Đối với trường hợp 3 thì chỉ cần return về.

```
//if empty canvas
if(destination.Children.Count == 0)
{
    //put item in canvas
    destination.Children.Add(imageControl);

    //Update PuzzleItemPlacement
    int indexToUpdate = int.Parse(destination.Tag.ToString());

    if (itemTransferred.DragFrom == typeof(ListBox))
    {
        //updat
        this.itemPlacement[indexToUpdate] = itemTransferred;

        //delete item from listbox
        ((IList)lbxDragSource.ItemsSource).Remove(itemTransferred);

    }
    else if (itemTransferred.DragFrom == typeof(Canvas))
    {
        //keo item tuw canvas khac(dk)
        //udate vi tri manh duoc keo
        int previousIndex = itemPlacement.IndexOf(itemTransferred);

        itemPlacement[indexToUpdate] = itemTransferred;


        itemPlacement[previousIndex] = emptyItem;

        //delete picture from listbox
        Canvas associated = GetAssociatedCanvasByIndex(previousIndex);
        associated.Children.Clear();
        associated = null;

    }
    else
    {
        MessageBox.Show("error");
    }
}
```

Nếu đích trống

```
//if canvas isnt empty
else if(destination.Children.Count > 0)
{

    //canvas isnt empty and drag from listbox - stop draging do nothing
    if (itemTransferred.DragFrom == typeof(ListBox))
    {
        return;
    }

    //canvas isnt empty and drag from another canvas -> switch them
    else if (itemTransferred.DragFrom == typeof(Canvas))
    {
        //get the draging item and destination index
        //index anh goc
        int sourceIndex = itemPlacement.IndexOf(itemTransferred);
        //index manh dich
        int destinationIndex = int.Parse(destination.Tag.ToString());

        object buffer = null;

        //doi index
        Image image0 = new Image() { Width = destination.Width, Height = destination.Height, Stretch = Stretch.Fill };
        image0.Source = itemPlacement[sourceIndex].PuzzleImageSource;

        Image image1 = new Image() { Width = destination.Width, Height = destination.Height, Stretch = Stretch.Fill };
        image1.Source = itemPlacement[destinationIndex].PuzzleImageSource;
        //trao doi vi tri 2 picture
        GetAssociatedCanvasByIndex(sourceIndex).Children.Clear();
        GetAssociatedCanvasByIndex(destinationIndex).Children.Clear();

        GetAssociatedCanvasByIndex(sourceIndex).Children.Add(image1);
        GetAssociatedCanvasByIndex(destinationIndex).Children.Add(image0);

        image0 = null;
        image1 = null;

        //switch placement
        buffer = itemPlacement[sourceIndex];
        itemPlacement[sourceIndex] = itemPlacement[destinationIndex];
        itemPlacement[destinationIndex] = buffer as PicturePiece;

        buffer = null;

    }
}
```

Nếu đích không trống

Kiểm tra hoàn thành và lưu kết quả

Hệ thống sẽ kiểm tra vị trí của mảnh ảnh với index ban đầu của mảnh, nếu trùng nhau (index trùng với số chỉ thì tính là đúng) và đầy đủ các mảnh thì người chơi hoàn thành màn chơi của mình.

```
public bool Validate(ObservableCollection<PicturePiece> itemPlacement)
{
    ObservableCollection<PicturePiece> placement = itemPlacement;

    foreach (PicturePiece item in placement)
    {
        if ((placement.IndexOf(item) != item.index) || placement.IndexOf(item) < 0)

            return false;
    }

    return true;
}
```

Người chơi sẽ được quyền lưu lại điểm hay không, trong trường hợp lưu:

- Hệ thống khởi tạo kết nối tới file database (**Playerdata**).
- Thực hiện một lệnh ghi thêm một bản ghi mới gồm tên người chơi và số điểm đạt được.
- Người chơi có thể xem lại bảng xếp hạng từ màn hình chính.

```
1 reference
private void Button_Click(object sender, RoutedEventArgs e)
{
    bonk.Open(new Uri(@"D:\BuhBuhLmao\BuBuLmao\Asset\Audio\bonk.mp3", UriKind.Relative));
    bonk.Volume = 1;
    bonk.Play();


    //Thiet lap ket noi database dung sqlite
    SQLiteConnection conn = new SQLiteConnection(dbconn);
    conn.Open();
    SQLiteDataAdapter ad = new SQLiteDataAdapter();
    SQLiteCommand cmd = new SQLiteCommand();

    //Query
    String query = "INSERT INTO ScoreRecord VALUES ('" + Playername.Text + "', '" + CongratulationsWindow.score + "')";
    cmd.CommandText = query;
    ad.SelectCommand = cmd;
    cmd.Connection = conn;
    SQLiteDataReader reader = cmd.ExecuteReader();


    MessageBox.Show("Done!");
    this.Close();
}
```

Lưu thành tích

```
1 reference
public RankWindow()
{
    InitializeComponent();

    clapping.Open(new Uri(@"D:\BuhBuhLmao\BuBuLmao\Asset\Audio\Clapping.mp3", UriKind.Relative));
    clapping.Play();


    //Thiet lap ket noi database dung sqlite
    SQLiteConnection conn = new SQLiteConnection(dbconn);
    conn.Open();
    SQLiteDataAdapter ad = new SQLiteDataAdapter();
    SQLiteCommand cmd = new SQLiteCommand();

    //Query
    String query = "SELECT * FROM ScoreRecord ORDER BY Points DESC";
    cmd.CommandText = query;
    ad.SelectCommand = cmd;
    cmd.Connection = conn;
    SQLiteDataReader reader = cmd.ExecuteReader();

    //Add data to 2 listview
    string record = "";
    while (reader.Read())
    {
        record += String.Format("{0} {1} \n", reader[0], reader[1]);
        string name = reader.GetString(0);
        int points = reader.GetInt32(1);
        PlayernameLV.Items.Add(name);
        PlayerscoreLV.Items.Add(points);
    }
}
}
```

Hiển thị các người chơi có điểm số cao

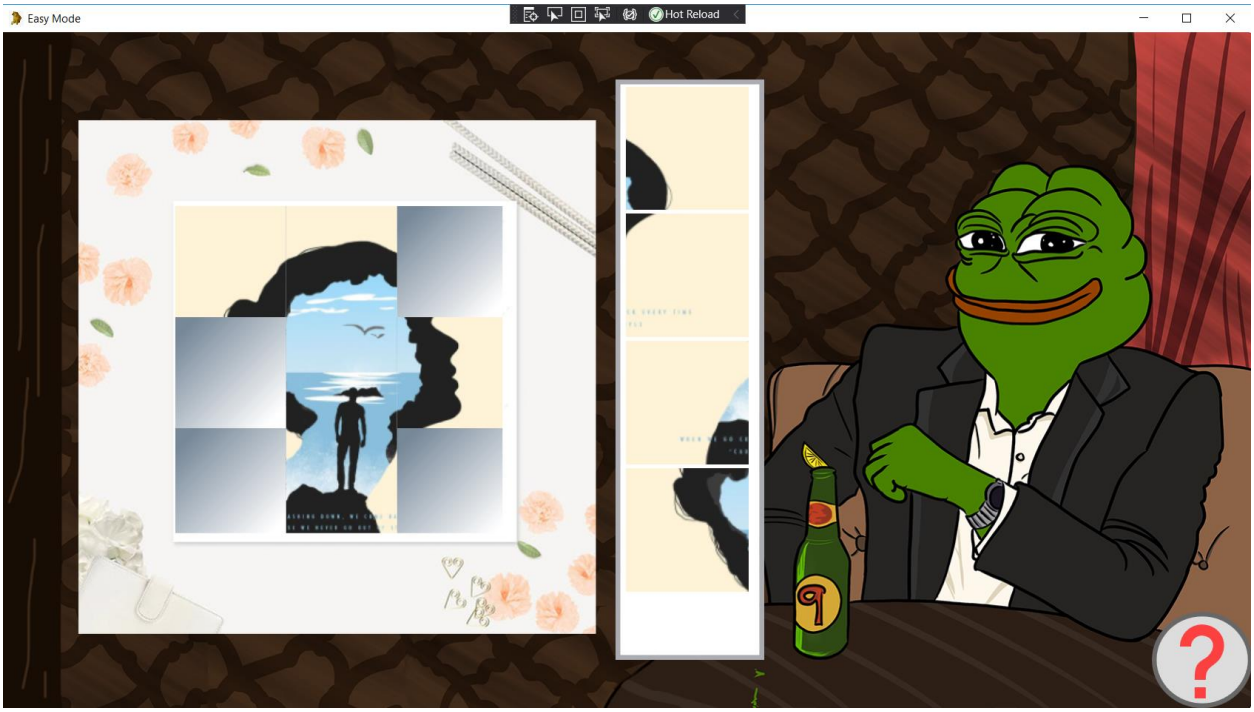
Một số cửa sổ cùng chức năng khác:

- **Setting Window:** màn hình cài đặt (ngôn ngữ và âm thanh).
- **CongratulationWindow:** màn hình chúc mừng người chơi hoàn thành.
- **DifficultiesWindow:** chọn màn chơi.
- **SaveScore:** lưu thông tin người chơi (tên, điểm số).
- **RankWindow:** bảng xếp hạng người chơi.

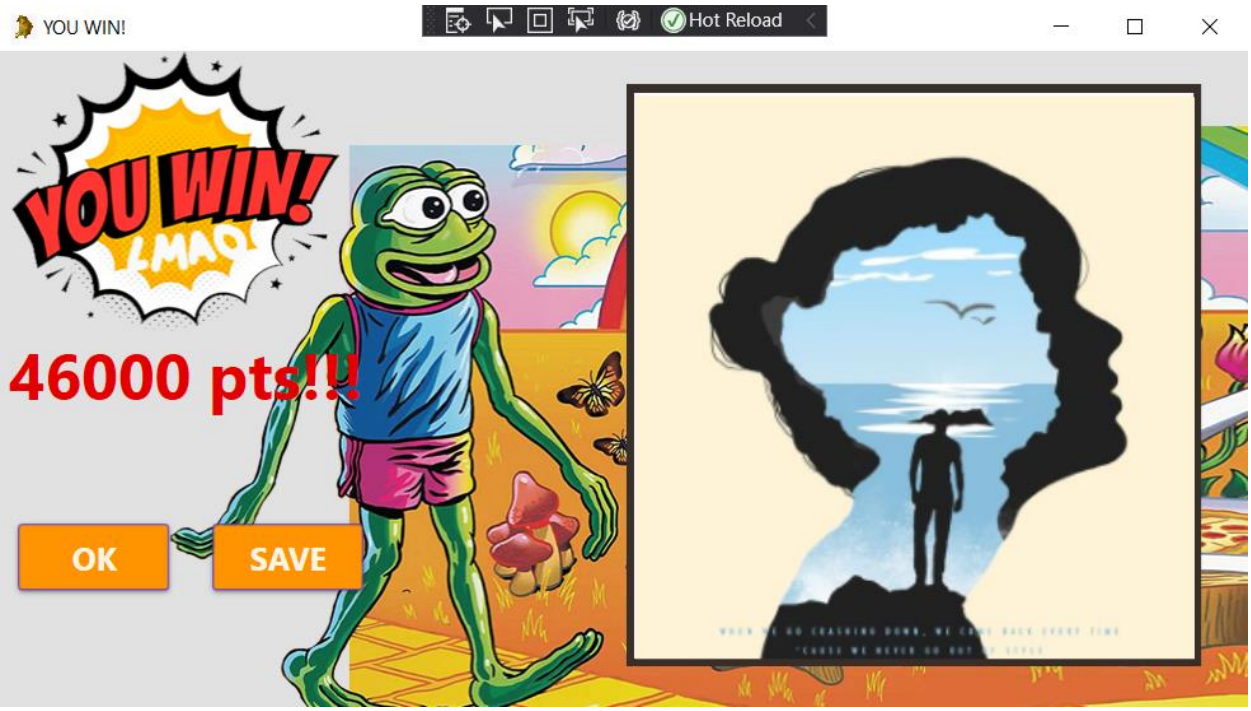
4. Giao diện chương trình

Giao diện của trò chơi yêu cầu cần thiết kết một cách khoa học, dễ nhìn, dễ thao tác cho người chơi. Sau đây là một số hình ảnh từ trò chơi:

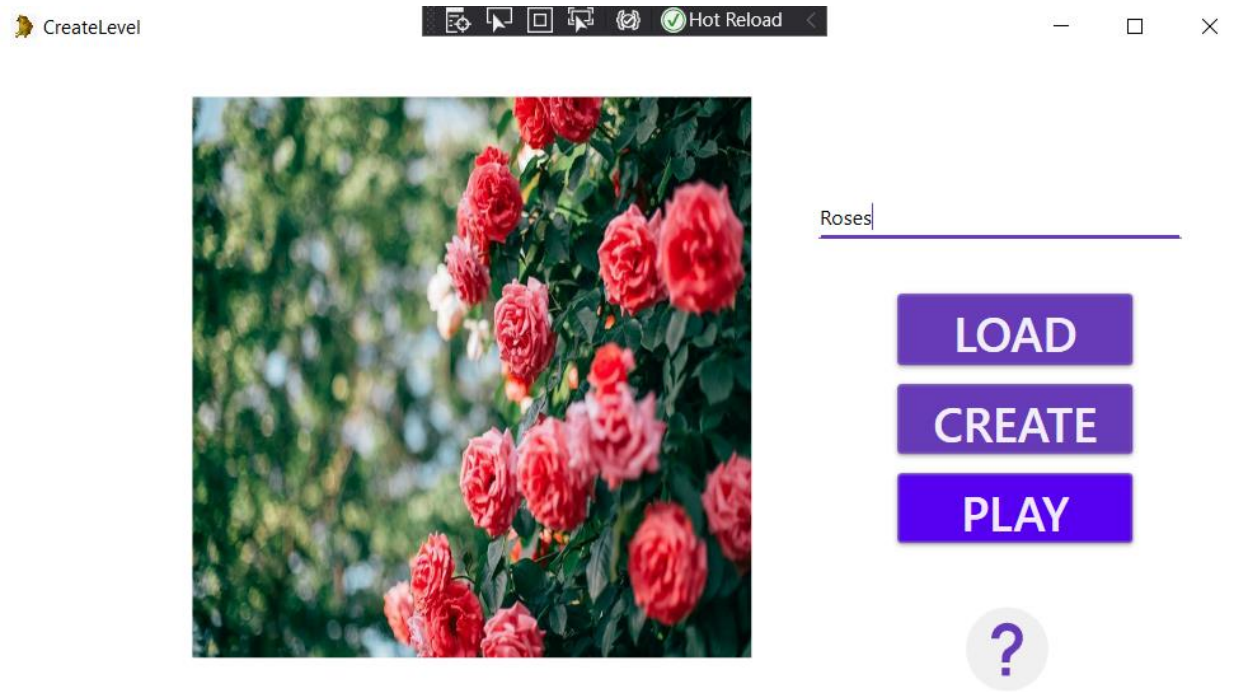
Màn hình chính



Màn chơi mức độ dễ



Màn hình chúc mừng



Màn hình tạo màn chơi

Màn hình bảng xếp hạng

IV. Kết luận

Nhóm chúng em đã hoàn thành được bài tập lớn "Xây dựng chương trình trò chơi Ghép tranh". Mặc dù chương trình còn đơn giản nhưng về cơ bản chúng em đã hoàn thành được phần thiết yếu của yêu cầu.

Trong quá trình thực hiện, do mới làm quen và chưa có tư duy cũng như kinh nghiệm lập trình tốt nên bài của nhóm vẫn còn nhiều hạn chế và thiếu xót. Rất mong nhận được những lời nhận xét và góp ý của thày và các bạn để chúng em lỗ lực hơn.

Sau cùng, chúng em xin chân thành cảm ơn thày Nguyễn Thành Trung, Giảng viên khoa Công Nghệ Thông Tin đã nhiệt tình hướng dẫn, giảng dạy giúp chúng em hoàn thành được bài bào cáo này.