**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN**

**GAME PUZZLE**

**Nhóm sinh viên thực hiện:**

**Trần Nguyên Tài 17110217**

**Giảng viên hướng dẫn:**

**TP. Hồ Chí Minh, tháng 8 năm 2019**

**Mục lục**

[**1.** **Mô tả về đề tài.** 1](#_Toc17049718)

[**1.1** ***Giới thiệu game Puzzle.*[1]** 1](#_Toc17049719)

[**1.2** ***Phân tích bài toán.*****[ 2]** 1](#_Toc17049720)

[**1.2.1** **Puzzle 3 x 3.** 1](#_Toc17049721)

[**1.2.2** **Puzzle 4 x 4.** 3](#_Toc17049722)

[**1.2.3** **Kết luận kết quả phân tích.** 3](#_Toc17049723)

[**2** **Mô tả quá trình thực hiện.** 4](#_Toc17049724)

[**2.1** ***Thiết kế giao diện*** 4](#_Toc17049725)

[**2.2** ***Các chức năng trong game.*** 4](#_Toc17049726)

[**2.3** ***Các class sử dụng trọng chương trình.*** 5](#_Toc17049727)

[**2.4** ***Các phương thức Form1.cs.*** 5](#_Toc17049728)

[**2.5** ***Cài đặt chương trình 8-Puzzle.*** 7](#_Toc17049729)

[**3** **Kết luận.** 15](#_Toc17049730)

[**3.1** ***Mức độ hoàn thành.*** 15](#_Toc17049731)

[**3.2** ***Ưu điểm và khuyết điểm.*** 15](#_Toc17049732)

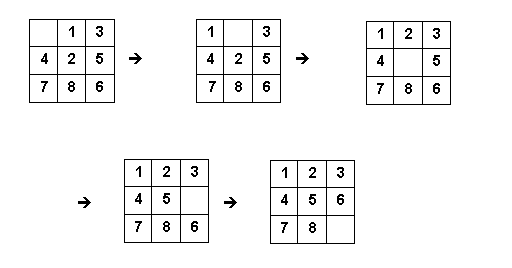
[**3.3** ***Hướng phát triển.*** 15](#_Toc17049733)

[**Tài liệu tham khảo** 16](#_Toc17049734)

# **Mô tả về đề tài.**

## ***Giới thiệu game Puzzle.*[1]**

*Puzzle* là một câu đố phát minh và phổ biến bởi Noyes Palmer Chapman trong những năm 1870. Bài toán gồm một bảng n × n với các ô số được đánh từ 1🡪n2 -1và một ô trống. Ở trạng thái bắt đầu, các ô được sắp đặt ngẫu nhiên, và nhiệm vụ của người giải là tìm cách đưa chúng về đúng thứ tự (lớn dần từ trái qua phải và từ trên xuống, ô cuối dùng là ô trống) bằng phép trượt các khối theo chiều ngang hoặc chiều dọc vào hình vuông trống như minh họa dưới với n=3:



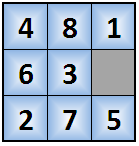
**Hình 1:Các bước chuyển từ trạng thái ban đầu về trạng thái mục tiêu của puzzle 3x3**

## ***Phân tích bài toán.*****[ 2]**

### **Puzzle 3 x 3.**

 Một trạng thái thì chỉ có tối đa 4 cách đi để chuyển sang trạng thái khác (trái, phải, lên, xuống). Người ta cũng nhận ra được rằng để có thể chuyển từ 1 trạng thái bất kì về trạng thái đích như trên thì trạng thái đầu đó phải theo một quy luật mà tôi trình bày sau đây.

Cho trạng thái đầu tiên như hình dưới, duyệt qua từng ô theo thứ tự từ trái qua và từ trên xuống, ở mỗi ô số duyệt đến, bạn hãy đếm xem có bao nhiêu ô số có giá trị bé hơn nó.



**Hình 2:Trạng thái bắt đầu của puzzle 3x3.**

Đầu tiên là ô số 4, ta thấy có 3 ô số {1;3;2} nằm phía sau và bé hơn nó nên n1 = 3.

– Tiếp đến là ô số 8 có 6 ô nhỏ hơn là {1;6;3;2;7;5}, vậy n2 = 6.

– Ô số 1 là bé nhất nên n3=0.

– Tương tự ô số 6, n4=3.

– Ô số 3, n5=1.

– Ô số 2, n6=0.

– Ô số 7, n7=1.

– Ô cuối luôn bằng 0 nên có thể bỏ qua, n8 = 0.

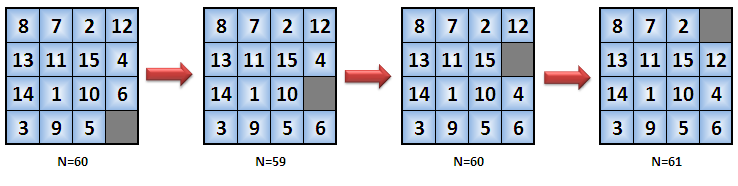
Tính tổng các số từ n1🡪 n8 ta có:

**N = 3 + 6 + 0 + 3 + 1 + 0 + 1 + 0 = 14.**

Với số N này ta chỉ cần biết 1 thông tin là nó có chia hết cho 2 hay không (lẻ hay chẵn). Nếu nó là số chẵn thì chắc chắn có thể chuyển về trạng thái đích từ trạng thái hiện tại này. Bởi vì khi di chuyển ô trống với hướng đi bất kì thì giá trị N mod 2 cũng không thay đổi. Tức là từ trạng thái đích bạn có thể xáo trộn bằng cách di chuyển ô trống nhiều lần thì giá trị N vẫn là số chẵn.

### **Puzzle 4 x 4.**

Gặp phải vấn đề khi kiểm tra theo cách tương tự như với bài toán puzzle 3x3. Khi di chuyển ô trống trong bảng giữa các dòng và tính toán giá trị N thì bạn nhận ra rằng N sẽ thay đổi giá trị chẵn, lẽ.



**Hình 3: Các trạng thái của puzzle 4x4 và N tương ứng.**

Ví dụ như các bảng phía trên ta tính được N tương ứng khi di chuyển ô trống giữa các dòng lần lượt là 60, 59, 60, 61. Qua vài lần thử ta có thể nhận ra quy luật là N là lẻ khi ô trống nằm ở các dòng lẻ tính từ trên xuống (dòng đầu tiên là 1), và N là chẵn khi ô trống nằm ở các dòng chẵn tính từ trên xuống.

Sự khác biệt này so với puzzle 3x3 là do độ dài cạnh (n) khác nhau. Tức là với n lẻ thì giá trị N mod 2 sẽ ko thay đổi, với n chẵn thì nó sẽ thay đổi tương ứng với vị trí dòng của ô trống trên bảng.

### **Kết luận kết quả phân tích.**

 Ta suy ra phương pháp tổng quát để áp dụng cho mọi n để **xác định được một puzzle** n × n **có thể chuyển về trạng thái đích với số lần di chuyển bất kì hay không**

**Với n lẻ:**

Chỉ cần N mod 2 = 0.

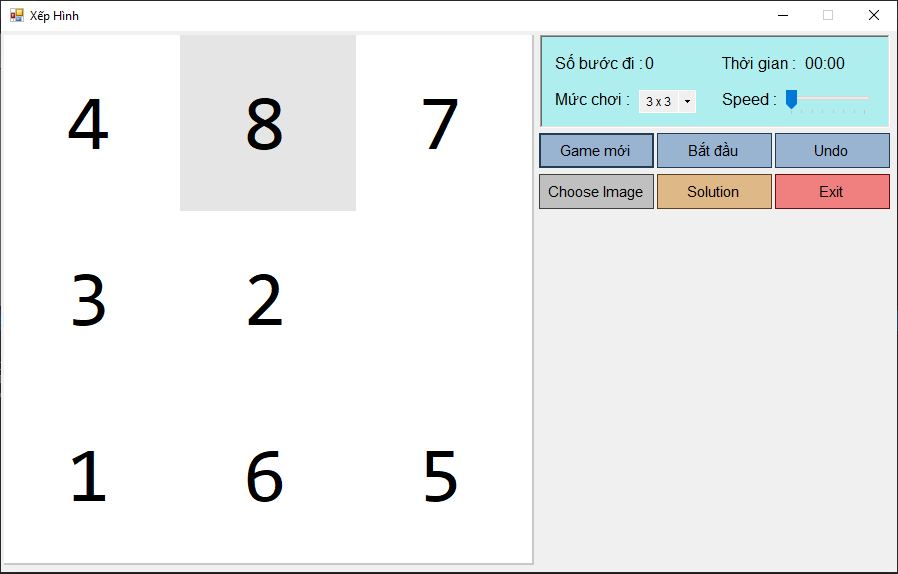
**Với n chẵn:**

N mod 2 = 0 và ô trống nằm trên dòng **chẵn** tính từ trên xuống (dòng đầu tiên là 1)

N mod 2 = 1 và ô trống nằm trên dòng **lẻ** tính từ trên xuống (dòng đầu tiên là 1).

# **Mô tả quá trình thực hiện.**

## ***Thiết kế giao diện***



**Hình 4: Giao diện chương trình**

## ***Các chức năng trong game.***

• Game mới để tạo mới lại game.

• Bắt đầu để chơi game, tạm dừng, tiếp tục.

• Undo để quay lại bước đi trước đó.

• Choose Image để chọn ảnh.

• Solution để máy chơi tự động.

• Exit để thoát chương trình.

## ***Các class sử dụng trọng chương trình.***

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên Class** | **Mục đích** |
| **1** | Form1.cs [Design] | Design giao diện game. |
| **2** | Form1.cs | Các hàm xử lý trong game. |
| **3** | MyRandom.cs | Tạo số ngẫu nhiên dùng trong khởi tạo trạng thái bắt đầu. |
| **4** | EnumMove.cs | Enum các trạng thái di chuyển Up, Down, Left, Right. |
| **5** | Node.cs | Lưu trạng thái của trò chơi. |
| **6** | Best\_First\_Search | Class giải bài toán tự động chơi. |

**Các lớp được sử dụng trong chương trình.**

## ***Các phương thức Form1.cs.***

|  |  |  |
| --- | --- | --- |
| **TT** | **Phương thức** | **Mục đích** |
| **1** | void initButton(int n) | Tạo ma trận Button[,] chứa các ô số. |
| **2** | public void swap(Button a, Button b) | Hoán đổi 2 Button. |
| **3** | public void go\_left(int n) | Hoán đổi ô trống và ô bên phải nó. |
| **4** | public void go\_right(int n) | Hoán đổi ô trống và ô bên trái nó. |
| **5** | public void go\_up(int n) | Hoán đổi ô trống và ô bên dưới nó. |
| **6** | public void go\_down(int n) | Hoán đổi ô trống và ô bên trên nó. |
| **7** | public void init(int n) | Khởi tạo trạng thái bắt đầu của game. |
| **8** | public bool ktWin() | Kiểm tra thắng. |
| **9** | private void bt\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào ma trận Button[,]. |
| **10** | private void btnPause\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào Button Bắt đầu(tạm dựng, chơi tiếp). |
| **11** | private void btnnewGame\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào Button New game. |
| **12** | private void btnExit\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào Button Exit. |
| **13** | private void Undo(int x) | Undo hành động. |
| **14** | private void btnUndo\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào Button Undo. |
| **15** | private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e) | Tạo event khi thay đổi combobox1. |
| **16** | private void timer\_Tick(object sender, EventArgs e) | Tính thời gian chơi game. |
| **17** | private void trackBar1\_Scroll(object sender, EventArgs e) | Tinh chỉnh timer1.Interval |
| **18** | private void Btnslt\_Click(object sender, EventArgs e) | Tạo event khi nhấn vào Button Tự động chơi. |
| **19** | private void timer1\_Tick(object sender, EventArgs e) | Tự động chơi game. |
| **20** | private void Choose\_Click(object sender, EventArgs e) | Event khi chọn ảnh. |
| **21** | private Image resize(Image img, Size newsize) | Resize ảnh. |
| **22** | private void CropImage(int n) | Cắt ảnh thành các mảnh nhỏ. |
| **22** | private void GetBoard(Button[,] btn) | Lấy ma trận trạng thái từ text của ma trận button. |
| **23** | private void MoveSlt(move a) | Di chuyển trong tự động chơi. |

**Các phương thức trong Form.cs**

## ***Cài đặt chương trình 8-Puzzle.***

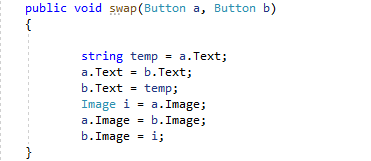
Trạng thái các ô số của game Puzzle n x n sẽ được lưu trong mảng 2 chiều các Button[,].

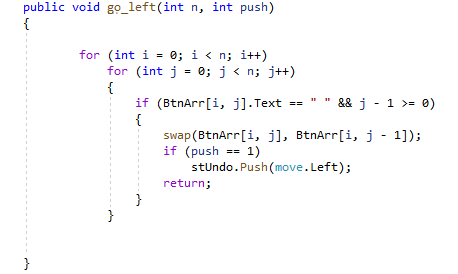


Khởi tạo trạng thái ban đầu ngẫu nhiên, các số từ 1🡪 n2 -1 sẽ được xếp ngẫu nhiên.

Ý tưởng khởi tạo:

Như trong phần phân tích, có ½ trạng thái ban đầu không thể đưa về trạng thái đích vì vậy để chắc chắn trạng thái ban đầu luôn hợp lệ thì ta sẽ khởi tạo bằng cách hoán vị vị trí ô trống với 1 ô cạnh nó (hoán vị text của Button có text = “0” và 1 Button cạnh nó trong mảng 2 chiều các Button[,] ) thông qua các thao tác lên, xuống, trái, phải. Bằng cách này ta có thể chắc chắc trạng thái ban đầu của game luôn là hợp lệ.





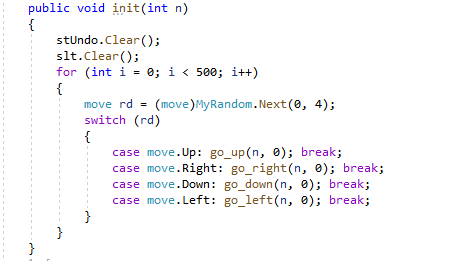
Tương tự với các hàm

public void go\_right(int n)

public void go\_top(int n)

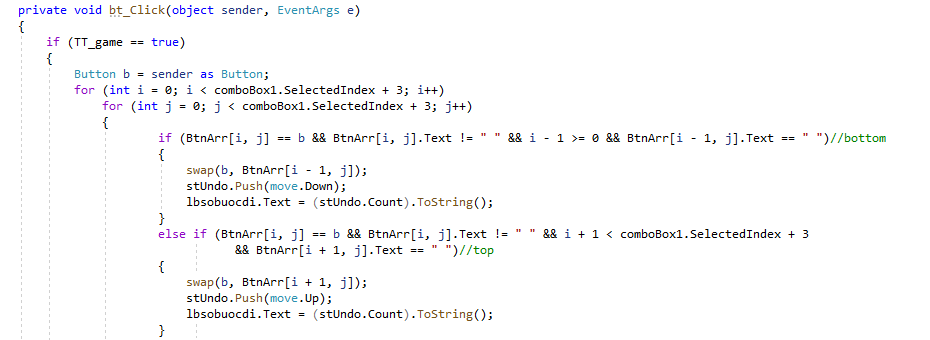
public void go\_bottom(int n)

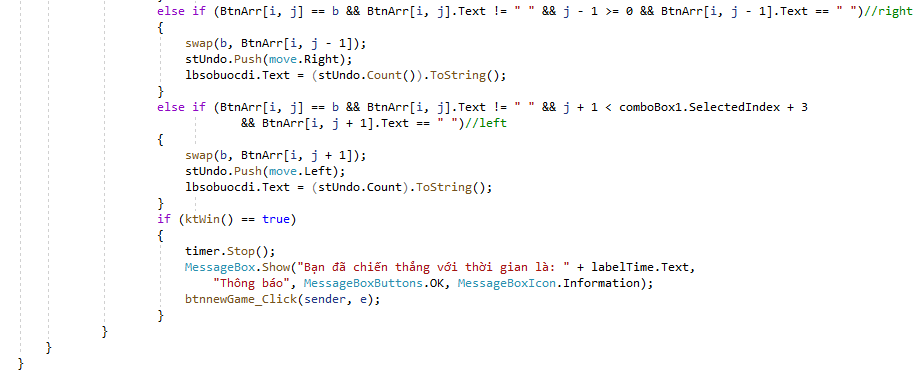
Khởi tạo trạng thái bắt đầu bằng cách cho Button[,] thực hiện 500 lần hành động hoán vị vị trí ô trống.



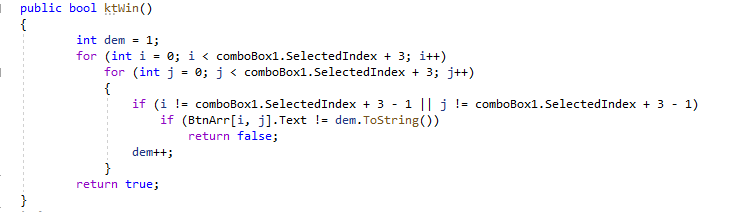
Tại một trạng thái thì chỉ có tối đa 4 cách đi để chuyển sang trạng thái khác (trái, phải, lên, xuống). Để di chuyển các ô trống ta thao tác trên chuột bằng cách click vào các button muốn di chuyển, ma trận Button [,] sẽ duyệt để tìm vị trí Button vừa click và tìm ô trống phía trên, dưới, trái, phải để thực hiện hành động di chuyển hợp lý, đồng thời trong thao tác di chuyển này ta cũng lưu các hành động vào stack để dùng cho tính năng undo. Cuối các hành động di chuyển ta kiểm tra xem game đã thắng hay chưa.

Dưới đây là code để di chuyển khi click vào Button.



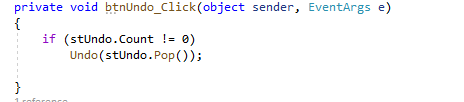


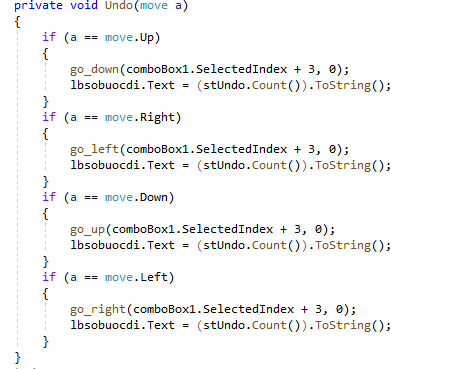
Hàm ktWin() trả về true nó đang ở trạng thái win.



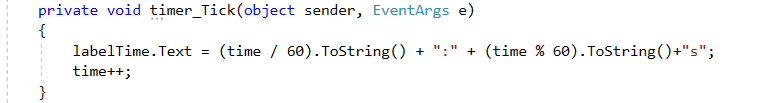
.

Hàm undo di chuyển của người chơi.

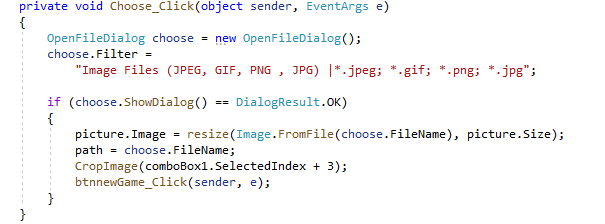




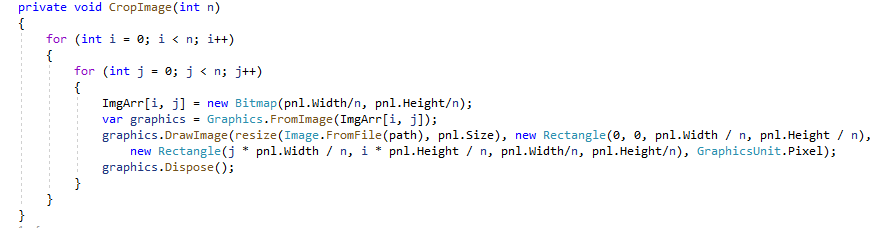
Hàm tính thời gian với Timer có interval = 1000.

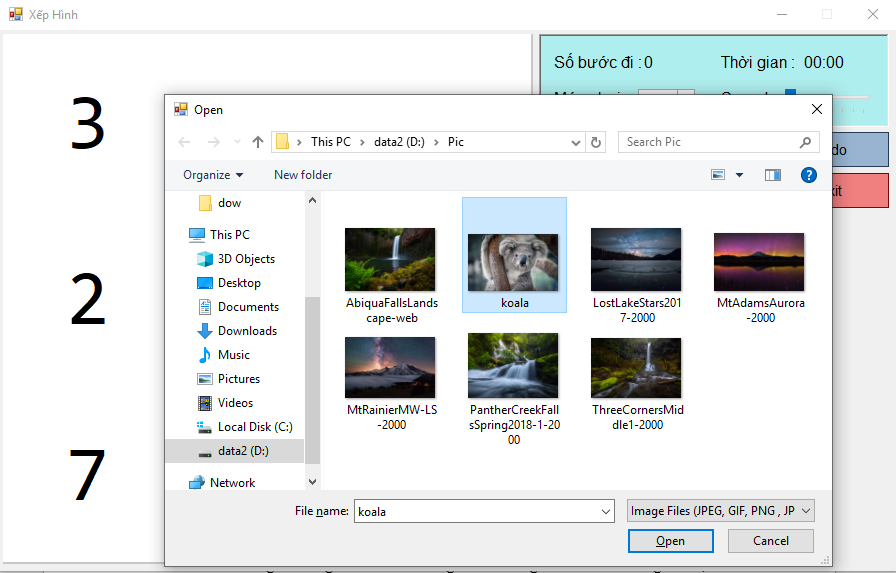


**Hàm Choose Image**

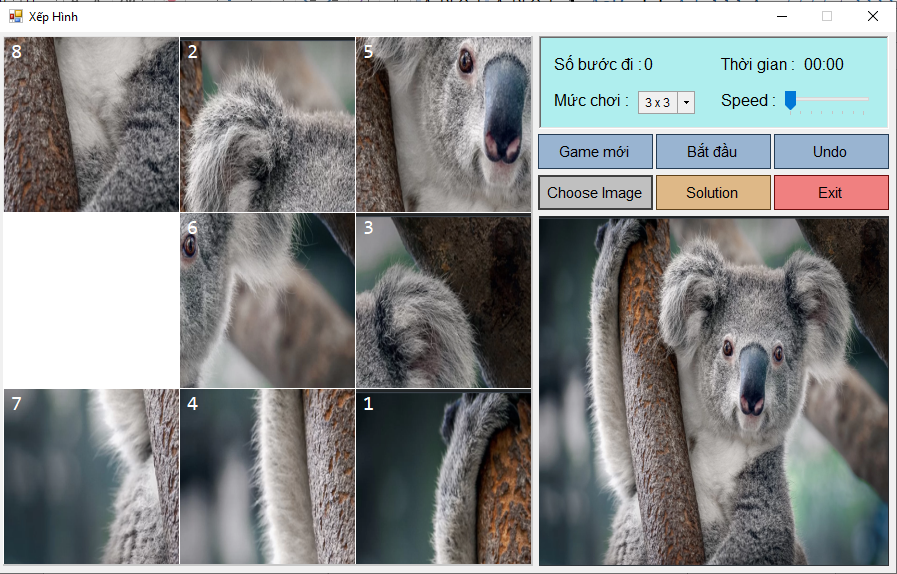


Hàm CropImage để cắt hình thành các hình nhỏ.





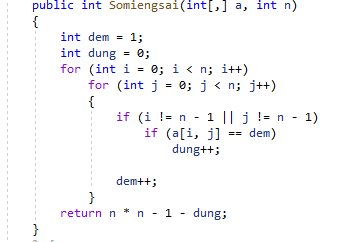
***Giao diện trò chơi khi chọn ảnh.***



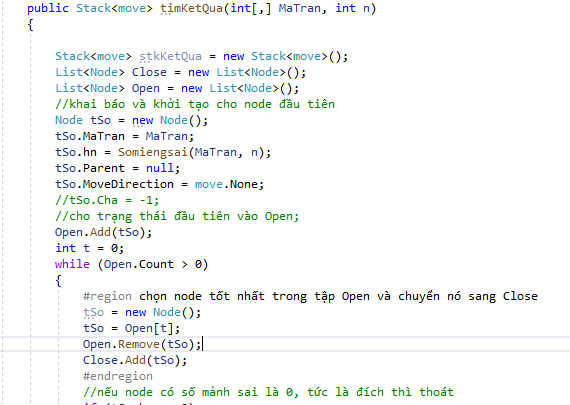
***Giao diện trò chơi sau khi chọn ảnh.***

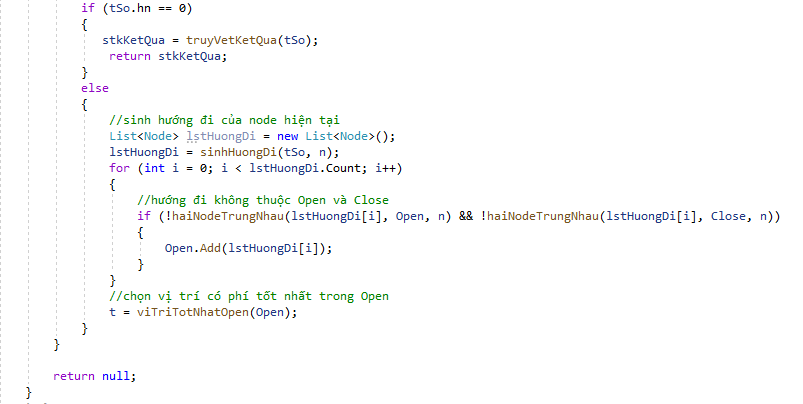
Chức năng tự động chơi của chương trình sử dụng thuật toán Greedy\_Best\_First\_Search. Để hạn chế không gian cây các lời giải tiềm năng, chúng ta đưa ra một hàm định hướng việc mở rộng cây tìm kiếm. Theo cách này, chúng ta sẽ mở rộng cây theo các nút lá có nhiều tiềm năng chứa trạng thái đích hơn các nút lá khác.

Trong chương trình hàm đánh giá được dùng là số mảnh sai vị trí.



Hàm tìm lời giải.





# **Kết luận.**

## ***Mức độ hoàn thành.***

Đồ án được hoàn thành 90% so với mục tiêu đặt ra.

## ***Ưu điểm và khuyết điểm.***

Ưu điểm: Giao diện đơn giản,dể sử dụng.

Chơi được các game puzzle 3x3, 4x4,..

Chức năng tự động chơi.

Nhược điểm: Chưa thể thao tác bằng bàn phím.

Chưa lưu được dữ liệu người chơi.

Chưa tìm ra lời giải tối ưu nhất.

## ***Hướng phát triển.***

* Thêm các lựa chọn game với n>5.
* Thêm tính năng giới hạn thời gian chơi.
* Thêm điều khiển bằng bàn phím.

# **Tài liệu tham khảo**

[1] <http://coursera.cs.princeton.edu/algs4/assignments/8puzzle.html>

[2] <https://yinyangit.wordpress.com/2010/12/11/algorithm-tim-hi%E1%BB%83u-v%E1%BB%81-bai-toan-n-puzzle-updated/>