BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH (UEH)
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

Cấu Trúc Dữ Liệu & Giải Thuật

ĐỀ TÀI

STACK VÀ BÀI TOÁN THÁP HÀ NỘI

Chuyên Ngành: Kỹ thuật phần mềm

Khóa: K47 **Lớp: SE001**

Thành viên nhóm:

Phạm Nguyễn Chí Khoa - 31211020279
Phan Vũ Trường An - 31211027568
Lã Văn Hà - 31211024624
Nguyễn Tiến Dũng - 31211027572

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

Tp. Hồ Chí Minh, Ngày 15 tháng 12 năm 2021

MỤC LỤC

PHẦN 1: STACK VÀ BÀI TOÁN THÁP HÀ NỘI

1.1. Bài toán Tháp Hà Nội

Bài toán Tháp Hà Nội (còn được gọi là Vấn đề đền Benares hay Tháp Lucas) được giới thiệu đến phương Tây vào năm 1883 bởi nhà toán học người Pháp Edouard Lucas. Với dạng thường gặp nhất của trò chơi này là một bộ gồm các đĩa có kích thước khác nhau với một lỗ tròn ở giữa các đĩa cùng với ba cái cọc (vật liệu thường được làm từ gỗ hoặc sứ).

Về quy tắc của trò chơi: khi bắt đầu, các đĩa sẽ được sắp xếp nằm trên một cọc theo thứ tự đĩa lớn nhất nằm dưới cùng và đĩa nhỏ nhất nằm trên cùng, càng lên cao sẽ tạo ra một dạng hình nón. Trò chơi sẽ yêu cầu người chơi di chuyển toàn bộ số đĩa từ cọc ban đầu sang một cọc khác mà phải tuân theo các quy tắc sau:

+ Chỉ được phép di chuyển các đĩa trong 3 cọc đã cho

+ Mỗi lần chỉ được di chuyển một đĩa nằm trên cùng của các cọc

+ Mỗi đĩa chỉ có thể được đặt lên trên một đĩa khác lớn hơn nó (không nhất thiết hai đĩa phải có kích thước liền kề nhau)

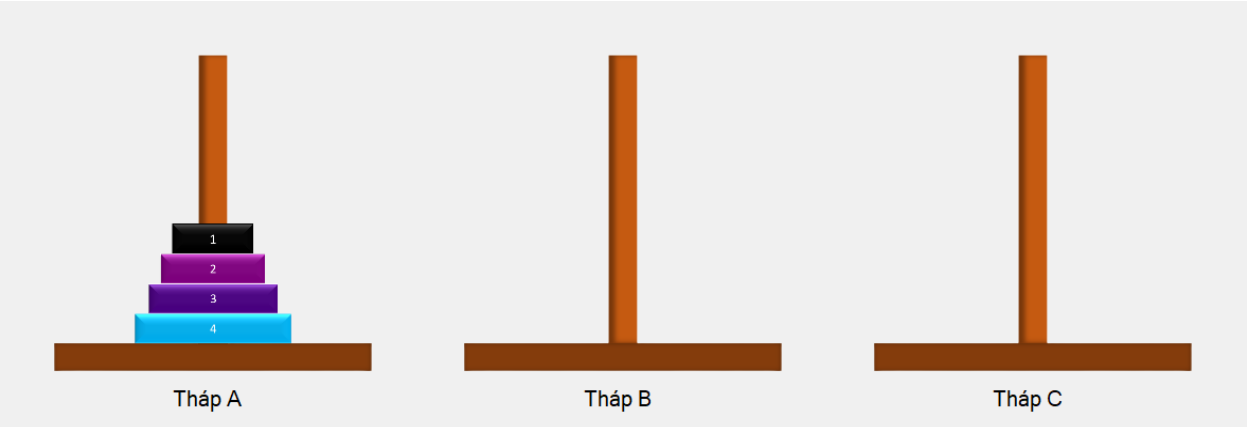
1.2. Ứng dụng Stack vào bài toán Tháp Hà Nội

Thông qua các quy tắc của trò chơi Tháp Hà Nội ta có thể nhận ra có một mối liên hệ với cấu trúc Stack theo nguyên lý LIFO (last in first out) khi mà mỗi lần chỉ có thể lấy được một đĩa từ phía trên cùng của cọc và đĩa đầu tiên được đặt vào cọc sẽ trở thành đĩa nằm dưới cùng và được lấy ra cuối cùng. Dựa vào đó, ta có thể tạo ra một trò chơi Tháp Hà Nội dựa trên cấu trúc Stack.

1.3. Thuật toán giải quyết bài toán Tháp Hà Nội

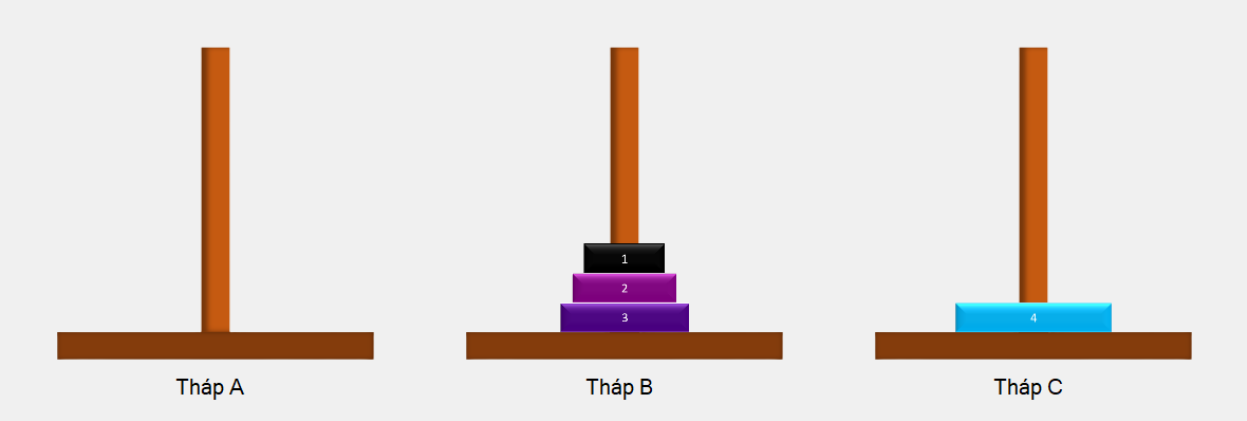
a) Phân tích bài toán

Ta đặt tên cho các cọc lần lượt là tháp A, tháp B và tháp C, với tháp khởi đầu là tháp A và yêu cầu chuyển toàn bộ đĩa sang tháp C. Ở trường hợp có hai đĩa, ta cần phải chuyển đĩa 1 sang tháp B thì mới có thể tiếp tục chuyển đĩa 2 sang tháp C được, rồi cuối cùng chuyển đĩa 1 từ B sang C và hoàn thành trò chơi.



Hình 1.1. Tháp Hà Nội với 4 đĩa (1)

Mở rộng hơn, ta lấy ví dụ với trường hợp bài toán có 4 đĩa, để chuyển cả 4 đĩa sang tháp C thì trước hết ta cần phải thao tác để chuyển cả 3 đĩa phía trên sang tháp B. Và nếu muốn chuyển cả 3 đĩa sang tháp B thì ta cần chuyển 2 đĩa phía trên sang tháp C, tiếp tục nếu muốn chuyển 2 đĩa sang tháp C thì ta cần phải bắt đầu với việc chuyển đĩa nhỏ nhất sang tháp B. Và nếu cứ tiếp tục giải bài toán, sau khi ta đã chuyển thành công đĩa thứ 4 sang tháp C, lúc này ở tháp B sẽ là một chồng gồm 3 đĩa và ta sẽ thực hiện các thao tác tương tự như trình bày ở trên để có thể đưa cả 3 đĩa từ tháp B sang tháp C và kết thúc bài toán.



Hình 1.2. Tháp Hà Nội với 4 đĩa (2)

Do quy tắc của trò chơi nên lúc này ở đáy các tháp trung gian sẽ luôn trống hoặc đĩa nằm trên cùng là các đĩa có kích thước lớn hơn đĩa cần chuyển nên việc chuyển các đĩa có thể thực hiện được mà không vi phạm quy tắc trò chơi.

b) Hướng giải quyết

Để giải quyết bài toán này, ta sẽ đặt các tên gọi "cột nguồn" sẽ là cọc có đĩa cần chuyển, "cột trung gian", và "cột đích" là cọc sẽ được chuyển đĩa từ cột nguồn đến. Và ta cần phải đặc biệt lưu ý rằng tên gọi của các "cột" sẽ không đại diện cố định cho bất kỳ cọc nào mà sẽ linh hoạt đại diện cho các cọc tùy vào từng bước của thuật toán.

Ta có thể thấy rằng ở trường hợp cột nguồn chỉ có 1 đĩa thì ta sẽ chuyển ngay đĩa ấy sang cột đích. Ở các trường hợp cọc có n đĩa (từ 2 đĩa trở lên) ta cần:

+ Bước 1: chuyển n – 1 đĩa phía trên từ cột nguồn sang cột trung gian

+ Bước 2: chuyển đĩa thứ n từ cột nguồn sang cột đích

+ Bước 3: chuyển n – 1 đĩa từ cột trung gian sang cột đích

Và với những phân tích trên phần phân tích bài toán, ta có thể dễ dàng thấy được sự xuất hiện của đệ quy ở bước thứ nhất và thứ ba ở trường hợp cọc có từ 2 đĩa trở lên. Bài toán với số đĩa lớn sẽ lần tìm về bài toán có số đĩa nhỏ hơn để giải quyết. Và như đã phân tích ở trên mỗi khi đi từ bài toán với số đĩa lớn hơn đi về bài toán có số đĩa nhỏ hơn một đĩa thì ta nhận thấy rằng cột trung gian và cột đích ở bước 1 sẽ bị hoán đổi, tương tự ở bước 3 thì cột nguồn và cột trung gian cũng sẽ bị hoán đổi với mỗi lần gọi đệ quy như vậy, đây là một điều cần lưu ý để viết thuật toán một cách chuẩn xác. Với thuật toán đệ quy trên thì số bước

tối thiểu để có thể hoàn thành trò chơi này là $(2^n - 1)$ bước. Lời giải tối ưu cho trò chơi có thể được tìm thấy chính xác thông qua thuật toán trên cho trường hợp 3 cọc, nhưng khi mở rộng cho 4 cọc hoặc nhiều hơn, lời giải chính xác cho đến nay vẫn chưa được khẳng định.

c) Mã nguồn

```
private void moveAction(int n, char first, char second)
{
      // Các thuật toán khi chuyển đĩa, ở đây đơn giản là thông báo ra
cách chuyển đĩa
      richTextBox1.Text += ($" Step {i}:  Chuyển đĩa số  ({n})  từ tháp
{first} --> {second}\n");
}

private void solveThapHN(int n, char cotNguon, char cotTrungGian, char
cotDich)
{
      if (n == 1) moveAction(1, cotNguon, cotDich);
      else
      {
            solveThapHN(n - 1, cotNguon, cotDich, cotTrungGian);
            moveAction(n, cotNguon, cotDich);
            solveThapHN(n - 1, cotTrungGian, cotNguon, cotDich);
      }
}
```

PHẦN 2: PHÂN TÍCH VÀ THIẾT KẾ CHƯƠNG TRÌNH

2.1. Phân tích về làm việc với Stack

Chúng ta muốn tạo ra trò chơi nơi mà người dùng có thể chơi được trò Tháp Hà Nội mà chỉ cần sử dụng chuột, với các thao tác đơn giản như click chuột thì ta cần phải phân tích về những vấn đề thực tế, về những hành động xảy ra khi ta chơi trò Tháp Hà Nội bằng tương tác vật lý.

Bài toán Tháp Hà Nội sẽ có 3 cọc chứa các đĩa, và như ta đã phân tích ở trên thì có sự liên hệ rất rõ ràng giữa các cọc và cấu trúc stack, nên ở đây ta sẽ cần có 3 Stack (ngăn xếp) để lưu trữ số "đĩa" hiện đang chứa của từng cọc.

Về các đĩa, khi bắt đầu chương trình, người dùng được phép chọn số đĩa để bắt đầu chơi, tương ứng với số đĩa được chọn thì các đĩa tương ứng sẽ được đẩy vào Stack của cọc khởi đầu theo thứ tự (bài toán có 4 đĩa thì các đĩa 4, 3, 2, 1 sẽ được đẩy lần lượt vào Stack). Do tính thứ tự ấy, các đĩa nên được đặt trong một Array (mảng) để có thể dễ dàng truy xuất đĩa cần tìm theo Index (chỉ mục) và có thể truy xuất lần lượt theo thứ tự.

Khi chơi, trò chơi bắt buộc cần phải lấy một đĩa từ một cọc (gọi là cọc thứ nhất) để chuyển sang cọc khác (gọi là cọc thứ hai) nên ta cần phải xác định được cọc nào được người dùng chọn và dùng hai Stack khác (2 Stack tạm) trỏ vào chính xác Stack trước đó (Stack lưu trữ các đĩa) của cọc được chọn.

Sau khi đã có được Stack các đĩa của hai cọc được chọn, ta cần so sánh đĩa đầu tiên của cọc thứ nhất với đĩa đầu tiên của cọc thứ hai. Nếu đĩa đầu tiên của cọc thứ nhất nhỏ

hơn đĩa đầu tiên của cọc thứ hai thì ta sẽ thực hiện lấy đĩa ra khỏi cọc thứ nhất và đẩy nó trở thành đĩa đầu tiên của cọc thứ hai, nếu không thỏa điều kiện thì ta sẽ không cho phép chuyển đĩa.

2.2. Sơ đồ về lớp và đường đi của chương trình

Program.cs

```
internal static class Program {
    static void Main() {
        Application.Run(new Form1());
    }
}
```

Form1.cs

```
public partial class Form1 : Form {
    public Form1() {
        InitializeComponent();
    }
}
// các phương thức, code xử lý sự kiện
```

Form1.Designer.cs

```
partial class Form1 {
    private void InitializeComponent() {
        // khởi tạo đồ họa
    }
}
```

Hình 2.1. Cấu trúc các lớp trong Windows Forms

Chương trình Windows Forms có hai lớp mặc định khi khởi tạo Project sẽ có đó là class Program và class Form1 (tên có thể thay đổi) được kế thừa từ class Form. Class Program sẽ nằm trong file *Program.cs*, còn class Form1 sẽ nằm trong cả hai file *Form1.cs* và *Form1.Designer.cs*, do đặc tính của nó là một *partial class*.

Như ta có thể thấy từ hình 2.1, hàm *Main* trong class Program chính là cổng vào của chương trình, nơi chương trình bắt đầu chạy. Trong hàm *Main* sẽ gọi đến hàm *Form1* của class Form1, tại đây hàm *Form1* lại gọi đến hàm *InitializeComponent* nằm trong *Form1.Designer.cs.* Với đặc tính của một *partial class,* class *Form1* được tách ra để dễ quản lý với việc *Form1.Designer.cs,* giống như cái tên của nó, là nơi để lập trình viên viết những dòng lệnh để thiết kế giao diện cho chương trình. Trong khi đó, *Form1.cs* sẽ đảm nhận là nơi viết các thuật toán, các phương thức và xử lý các sự kiện liên quan đến tương tác trong Form, việc khai báo các Stack, Array hay câu lệnh tương tác ứng dụng Stack để chơi trò chơi cũng sẽ được viết ở đây

2.3. Giới thiệu tổng quan trước cài đặt

Phần xử lý và làm việc với Stack trong việc chọn và chuyển đĩa có thể nói là phần cốt lõi của chương trình Tháp Hà Nội. Trước khi bước vào phần cài đặt của thuật toán thì ta sẽ nói sơ về các biến, một số đặc điểm của Windows Forms cũng như sơ đồ của thuật toán để có thể hiểu rõ về nó nhất.

Trong Windows Forms, các đồ họa thể hiện cho người dùng xem và tương tác được gọi là các Control, trong đó, các đĩa (trong chương trình được gọi là plate) và các cọc (được

gọi là Rod) trong chương trình cũng sẽ được hiển thị dưới dạng Control có kiểu là PictureBox. Ta có tể truy cập đến các Control này thông qua thuộc tính *Name* của nó. Các đĩa sẽ được lưu trong một Array có kiểu là PictureBox (`PictureBox[]`), các cọc sẽ có một Stack dùng để chứa các đĩa có kiểu PictureBox (`Stack<PictureBox>`).

Các Control trong Windows Forms cũng có một thuộc tính có tên gọi là Tag, thuộc tính này có thể dùng để lưu bất kỳ kiểu dữ liệu nào, cho dù đó có là một Stack, và khi cần sử dụng, chúng ta chỉ cần gọi đến phương thức này và ép sang kiểu mà chúng ta có thể sử dụng. Ở trong chương trình này, tương ứng với từng plate được đánh số từ 1 đến 8 thì Tag của nó cũng được gán bằng 1 đến 8 (`this.plate4.Tag = "4"`).

Các Control đều sẽ có thuộc tính Location để xác định vị trí mà nó sẽ xuất hiện trên giao diện của Windows Forms, hiểu đơn giản thì Location sẽ xác định tọa độ (X, Y) của Control, được xác định ở góc cùng trên bên trái của Control, so với góc trên cùng bên trái của Form có tọa độ (0, 0). Location được nhắc đến bởi vì nếu một đĩa đủ điều kiện để đặt sang cọc khác thì Location của đĩa đó cũng cần phải thay đổi để hiển thị chính xác, một cách đồng bộ và trực quan những gì đang xảy ra với các Stack lưu trữ của cọc.

Chúng ta sẽ đến với cấu trúc các biến được khai báo trong chương trình sẽ xuất hiện trong cài đặt bên dưới để có thể dễ dàng theo dõi thuật toán: plate là hình ảnh của các đĩa, Rod là hình ảnh của các tháp, trong khi đó tower sẽ là Stack chứa dữ liệu cho tháp. Hình ảnh biểu thị Rod và Stack tower sẽ được liên kết nhờ vào thuộc tính Tag của các PictureBox Control.

```
      plates = new PictureBox[] { plate1, plate2, plate3, plate4, plate5, plate6, plate7, plate8 }; // các đĩa được đưa vào array theo thứ tự

// stack đại diện cho các cọc, tag của control hình ảnh các cọc dùng để lưu trữ stack của cọc đó
      RodA.Tag = towerA = new Stack<PictureBox>();
      RodB.Tag = towerB = new Stack<PictureBox>();
      RodC.Tag = towerC = new Stack<PictureBox>();
```

Với dòng code trên, từ giờ ta sẽ biết Stack towerA sẽ đại diện cho cọc A và tương ứng với các cọc còn lại. *Và ta cần lưu ý*, `Rod` là hình ảnh của tháp và nhìn thấy được, còn `tower` là dữ liệu thì không thể được quan sát và tương tác, `tower` sẽ là đại diện cho vùng dữ liệu của từng tháp, nơi sẽ được các Stack trỏ vào.

2.4. Cài đặt thuật toán

Xem code hoàn chỉnh trên Github theo link và hướng dẫn ở Phụ Lục

a) Khởi tạo ban đầu

Để bắt đầu trò chơi, ta cần phải lấy số đĩa muốn chơi đặt vào cọc khởi đầu theo thứ tự quy tắc của trò chơi "đỉnh min đáy max". Với thao tác đó, ta có hai việc cần làm trong chương trình: đưa các đĩa vào Stack của cọc A và hiển thị các đĩa được đưa vào Stack để người dùng có thể quan sát. Người chơi có thể chọn số đĩa mà họ muốn bắt đầu và ta có thể lấy dữ liệu về số đĩa ấy từ thuộc tính Value của Control có tên là NumericUpDown (`numudNumPlate.Value`).

```
private void generatePlates()   // khởi tạo các đĩa
{
      int x = RodA.Location.X + plateRodDisX;
      int y = fPlatePosY;
      for (int i = (int)numudNumPlate.Value - 1; i >= 0; i--, y -=
plateHeight)
      {
            plates[i].Location = new Point(x, y);   → Vị trí các đĩa
            plates[i].Visible = true;               → Hiển thị các đĩa

            towerA.Push(plates[i]);
      }
}
```

Đối với phương thức ở trên, ta chỉ cần biết rằng x, y là tọa độ của các đĩa, do PictureBox của các đĩa có độ rộng ngang nhau nên trong vòng lặp chỉ có y là độ cao thay đổi (do các đĩa chồng lên nhau) mà x vẫn giữ nguyên, x ở đây sẽ được xác định dựa vào vị trí của cọc chứa nó cùng khoảng chênh lệch về độ rộng giữa cọc và đĩa. Do đĩa càng lớn hơn sẽ càng nằm sâu hơn trong Stack cũng như trên cọc nên trong vòng lặp ta sẽ đi từ đĩa thứ n (người chơi chọn n đĩa) về đĩa thứ nhất với tọa độ y mỗi lần lặp sẽ giảm bằng đúng chiều cao của đĩa (đảm bảo các đĩa sẽ nằm sát lên nhau), đồng thời ta sẽ đẩy lần lượt các đĩa vào trong Stack.

b) Xử lý chọn tháp

```
Stack<PictureBox> fClickedPlates, sClickedPlates; // lần lượt dùng để trỏ
đến stack của tháp được chọn thứ nhất và thứ hai | giải thích phía sau
```

Sau khi đã đưa được các đĩa vào cọc (từ giờ sẽ gọi là tháp) A thì ta đã có thể bắt đầu trò chơi. Trong lúc chơi, người chơi có thể sẽ có các thao tác sau:

+ Lấy đĩa từ tháp thứ nhất chuyển sang tháp thứ hai (nếu hợp lệ)

+ Lấy đĩa từ tháp thứ nhất nhưng không muốn chuyển nên trả về vị trí ban đầu

Đối với các trường hợp, ta nhận thấy chung quy lại sẽ có hai đối tượng được quan tâm là tháp cần chuyển đĩa (tháp được chọn đầu tiên) và tháp cần chuyển đĩa đến (tháp được chọn thứ hai), ở trường hợp người chơi hủy chuyển đĩa chỉ đơn giản là tháp được chọn thứ hai sẽ chính là tháp được chọn đầu tiên.

Vấn đề được đặt ra ở đây là, nếu như đối với trò chơi ở dạng vật lý, việc người chơi chọn tháp, lấy đĩa diễn ra một cách trực quan vì ta có thể xem như lúc ấy tháp chính là một Stack có thể nhìn thấy và quan sát được. Còn ở phiên bản chương trình 2D, việc chọn tháp thông qua click chuột và ta cần phải xác định xem tháp được người dùng nhấn chính xác là tháp nào và thông qua Tag của tháp đó để có thể liên kết được với dữ liệu trong Stack của nó.

```
private void Rod_Click(object sender, EventArgs e)
{
      // trỏ đến chính xác Rod được click
      PictureBox clickedRod = (PictureBox)sender;
```

```
      Stack<PictureBox> clickedRodPlates =
(Stack<PictureBox>)clickedRod.Tag; // stack lưu các đĩa của chính xác
rod được click

      // Còn tiếp phía dưới...
}
```

Hành động click chuột của người dùng sẽ được ghi nhận dưới dạng các sự kiện (event), và sự kiện có tên Rod_Click được thêm vào cho cả 3 tháp (khi người dùng click chuột vào PictureBox của 3 tháp thì các câu lệnh bên trong sự kiện sẽ được thi hành). Tham số *sender* chính là đối tượng mà hành động click hướng vào nên từ đấy ta có thể xác định được tháp nào trong 3 tháp là tháp được người dùng chọn. Khi này **`clickedRod`** chính là RodA, RodB hay RodC tùy thuộc vào người dùng click vào hình ảnh của tháp nào.

Ta cần lưu ý: việc gán một Stack (tạm gọi stackA) cho một Stack (tạm gọi stackB) giống như việc ta chia sẻ chìa khóa truy cập đến những dữ liệu trong stackB cho stackA. Lúc này những thao tác Push() hay Pop() trên một trong hai Stack cũng sẽ thay đổi dữ liệu của cả hai Stack. Khi ấy ta hiểu rằng stackA và stack B đang cùng liên kết đến một vùng lưu trữ dữ liệu. Mọi chuyện cũng tương tự nếu ta gán stackC bằng stackA, lúc này stackC cũng sẽ liên kết đến vùng dữ liệu ấy. Và khi ta gán một trong ba Stack trên bằng null, thì dữ liệu trong vùng dữ liệu không bị ảnh hưởng mà chỉ đơn giản là ta đã cắt đứt sự liên kết của Stack ấy đến vùng dữ liệu ban đầu.

Thế nên trong chương trình ta sẽ thấy:

```
RodA.Tag = towerA = new Stack<PictureBox>();
Stack<PictureBox> clickedRodPlates = (Stack<PictureBox>)clickedRod.Tag;
fClickedPlates = clickedRodPlates;
```

Tức là lúc này **`fClickedPlates`** (sẽ được gọi là Stack của tháp thứ nhất) sẽ trỏ đến Stack lưu dữ liệu của tháp được click, và những thao tác Push() hay Pop() trên đây cũng sẽ ảnh hưởng đến dữ liệu trong Stack mà tháp trỏ vào (các tower tương ứng). Nhờ có đặc tính này của Stack mà những công việc tiếp theo được thực hiện khá thuận lợi và dễ dàng.

Vì các lần click chuột là như nhau và không thể phân biệt nên để xem khi nào người chơi chọn tháp thứ nhất và khi nào người chơi chọn tháp thứ hai ta sẽ kiểm tra xem Stack của tháp thứ nhất có trỏ đến tower nào chưa, nếu chưa thì ta cho **`fClickedPlates`** trỏ đến tower tương ứng. Còn nếu nó không null thì tức lần click này người dùng muốn chọn tháp thứ hai. Ta sẽ cho **`sClickedPlates`** (Stack của tháp thứ hai) trỏ đến tower của Rod được click và thực hiện thao tác chuyển đĩa. Phương thức **`CheckMovePlate`**, với tham số là chính Rod của đĩa được click, sẽ kiểm tra các điều kiện để thực hiện chuyển đĩa.

```
private void Rod_Click(object sender, EventArgs e)
{
      // ...Nối tiếp phía trên
      // chưa chọn cọc cần di chuyển đĩa (chọn cọc thứ nhất)
      if (fClickedPlates == null)
      {
            if (clickedRodPlates.Count == 0) return;
```

```
            fClickedPlates = clickedRodPlates;
            clickedRod.BorderStyle = BorderStyle.FixedSingle;
      }
      else  // chọn cọc cần chuyển đĩa đến (chọn cọc thứ hai)
      {
            if (sClickedPlates == null)
            {
                  // hủy chọn tháp cần di chuyển
                  if (clickedRodPlates == fClickedPlates)
                  {
                        fClickedPlates = null;
                        clickedRod.BorderStyle = BorderStyle.None;
                        return;
                  }
                  sClickedPlates = clickedRodPlates;
                  ChecknMovePlate(clickedRod);
            }
      }
}
```

c) Xử lý chuyển đĩa

Ta dùng phương thức Peek() kiểm tra xem phần tử đầu tiên của tháp thứ nhất có nhỏ hơn tháp thứ hai hay không, và như ở trên có đề cập, trong thuộc tính Tag của từng đĩa có lưu độ dài của đĩa, bây giờ ta chỉ cần lấy chúng ra, chuyển kiểu và so sánh với nhau.

```
private void ChecknMovePlate(PictureBox clickedRod)
{
      if (sClickedPlates.Count == 0)
            MovePlate(new Point(clickedRod.Location.X + plateRodDisX,
      fPlatePosY));
      else
      {
            PictureBox fRod_TopPlate = fClickedPlates.Peek();
            PictureBox sRod_TopPlate = sClickedPlates.Peek();
            // Kiểm tra điều kiện chuyển đĩa
            if (Convert.ToInt32(fRod_TopPlate.Tag.ToString()) <
      Convert.ToInt32(sRod_TopPlate.Tag.ToString()))
            {
                  MovePlate(new Point(sRod_TopPlate.Location.X,
            sRod_TopPlate.Location.Y - plateHeight));
            }
            else  // bỏ đánh dấu lần chọn cọc thứ hai (cho chọn lại)
                  sClickedPlates = null;
      }
}
```

Vị trí (Location) của đĩa ở trường hợp tháp được chọn thứ hai chưa có đĩa nào sẽ được xác định như lúc ta khởi tạo đĩa khi bắt đầu trò chơi (xem lại 2.4.a Khởi tạo ban đầu). Còn ở trường hợp đã có đĩa thì toạ độ trục X sẽ bằng với tọa độ của đĩa đầu tiên trong tháp, còn tọa độ trục Y sẽ là chính là tọa độ Y của đĩa đầu tiên trong tháp trừ đi chiều cao của đĩa.

Phương thức **MovePlate** được gọi để thực hiện thao tác chuyển đĩa, phương thức này có tham số là một point thuộc class Point dùng để xác định tọa độ mới của đĩa khi đã được chuyển sang cọc mới.

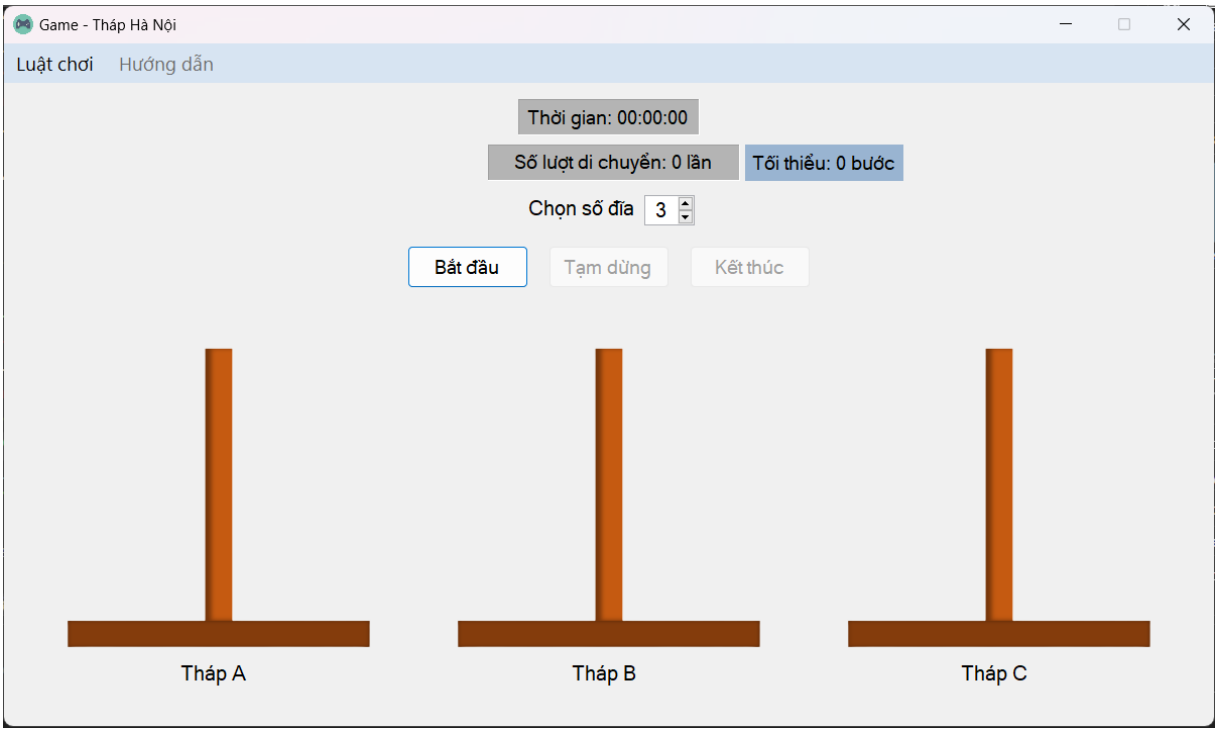
```
private void MovePlate(Point point)
{
      PictureBox fRod_TopPlate = fClickedPlates.Pop();
      sClickedPlates.Push(fRod_TopPlate);
      fRod_TopPlate.Location = point;

      fClickedPlates = sClickedPlates = null;
      countMove++;
      labelCountMove.Text = String.Format($"Số lần di chuyển: {countMove}
   lần");
      RodA.BorderStyle = RodB.BorderStyle = RodC.BorderStyle =
   BorderStyle.None;

      // Kiểm tra để kết thúc trò chơi (người chơi thắng)
      if (towerC.Count == numudNumPlate.Value)
      {
         btnGiveUp.PerformClick();
         MessageBox.Show($"Chúc mừng bạn đã hoàn thành trò chơi với số
      bước di chuyển là {countMove}, với số bước tối thiểu cho
      {numudNumPlate.Value} đĩa là {minMove} bước", "Congratulations");
      }
}
```

fRod_TopPlate là biến dùng để lưu phần tử đầu tiên được lấy ra khỏi tháp thứ nhất, ta đẩy nó vào Stack của tháp thứ hai, và cũng không quên thay đổi vị trí của đĩa đã được chuyển. Sau khi thực hiện xong các bước ta gán cả hai tháp giá trị null để chuẩn bị cho lần chuyển đĩa tiếp theo của người chơi. Song song với đó là cập nhật những thông tin của trò chơi để hiển thị các thông tin một cách chính xác. Trong phương thức này cũng là nơi ta kiểm tra xem người chơi đã hoàn thành trò chơi chưa bằng cách đếm số phần tử có trong towerC (trò chơi sẽ kết thúc nếu tất cả đĩa được chuyển sang tháp C) có bằng với số đĩa mà người chơi đã chọn hay không. Nếu bằng thì ta kết thúc trò chơi bằng cách gọi đến phương thức PerformClick của control btnGiveUp "Kết thúc". Phương thức này các tác dụng như việc chúng ta click vào Control ấy. Khi này trò chơi sẽ ở trạng thái kết thúc và một MessageBox thông báo hoàn thành trò chơi sẽ hiện ra.

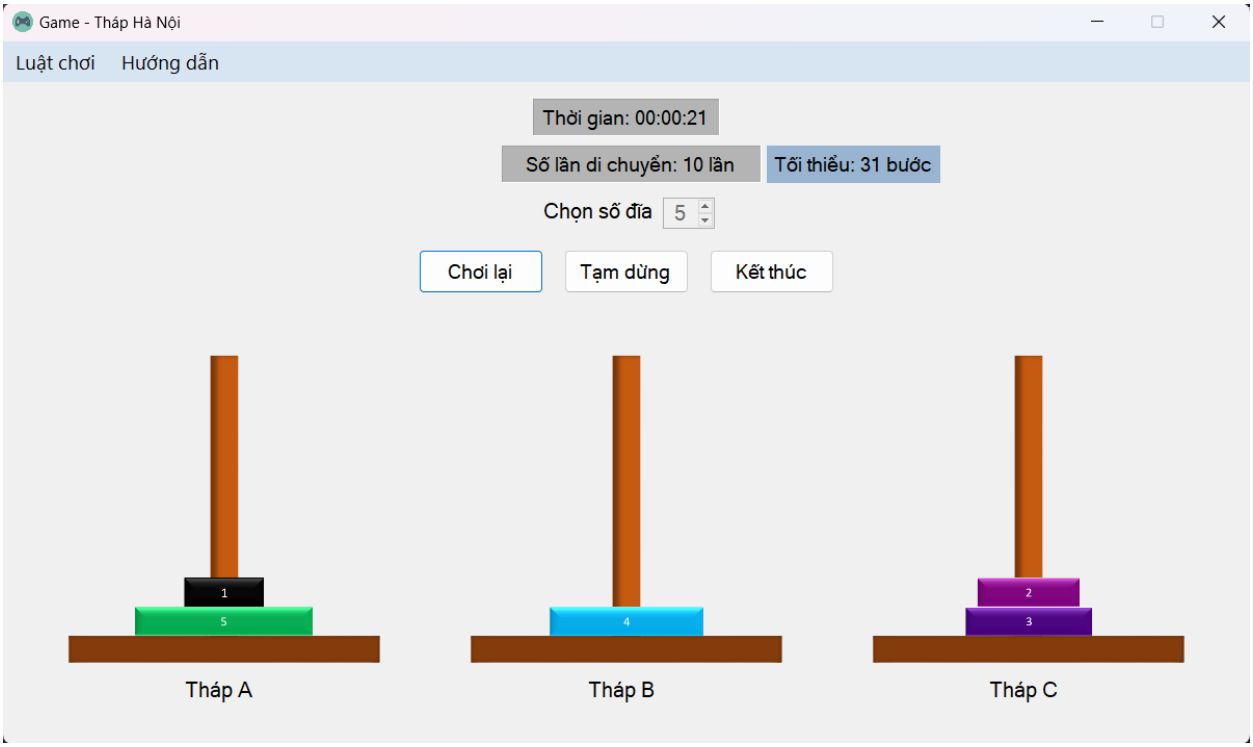
PHẦN 3: THIẾT KẾ GIAO DIỆN

3.1. Giao diện menu chính

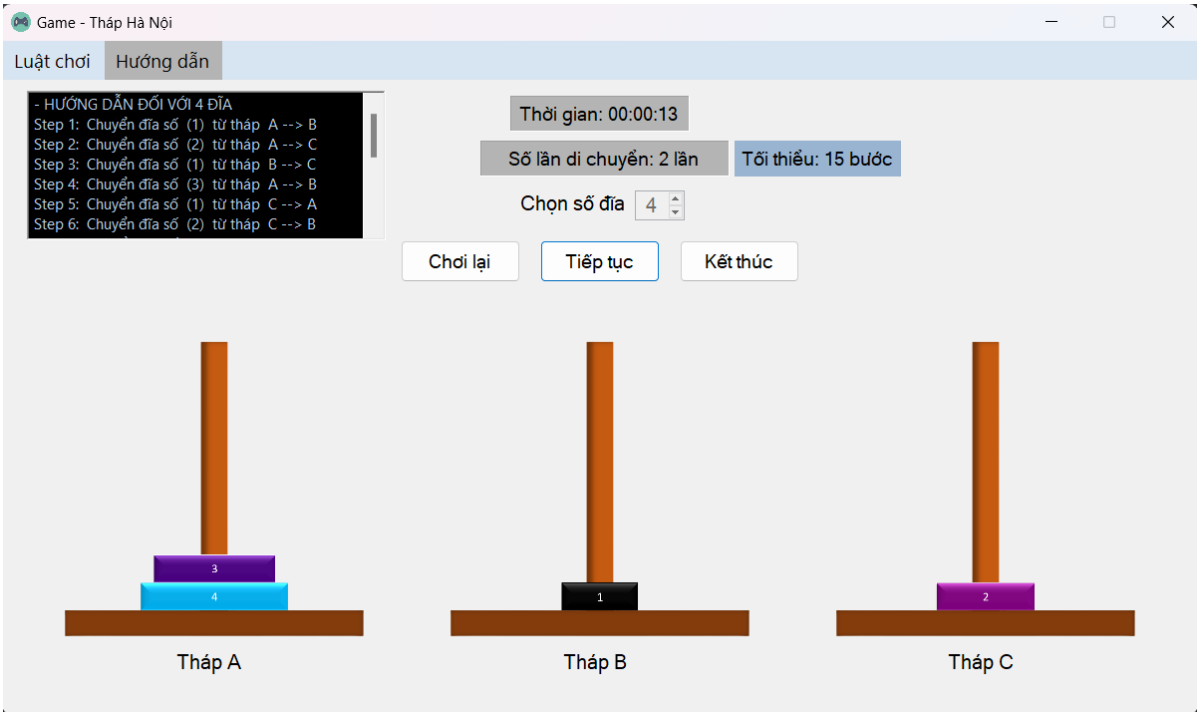
- Hình ảnh giao diện:



Hình 3.1. Giao diện khi vừa vào game



Hình 3.2.1. Giao diện khi đang chơi (1)



Hình 3.2.2. Giao diện khi đang chơi (2)

- Sơ lược về giao diện:

+ Cửa sổ giao diện không cho phép người dùng thay đổi kích thước của nó

+ Có một thanh menu chính gồm hai chức năng là cho biết luật chơi và hướng dẫn người chơi các bước hoàn thành trò chơi với số bước tối thiểu

+ Có một label thể hiện thời gian, một label hiện số lần người chơi di chuyển và một label thể hiện số bước tối thiểu để hoàn thành trò chơi

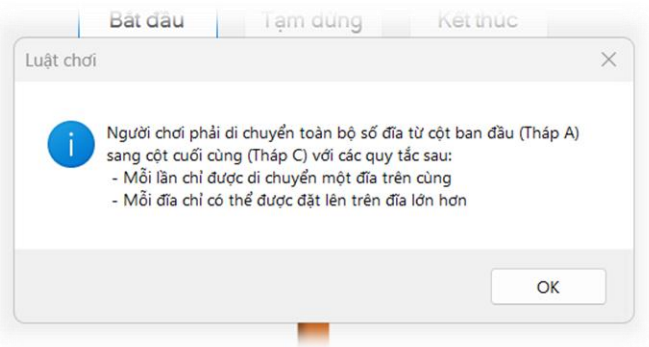
+ Ba nút chức năng cơ bản của một trò chơi, bắt đầu, tạm dừng và kết thúc

+ Cuối cùng là phần chiếm nhiều diện tích nhất của giao diện, đó chính là đồ họa của 3 tháp cũng như các đĩa

3.2. Chi tiết chức năng

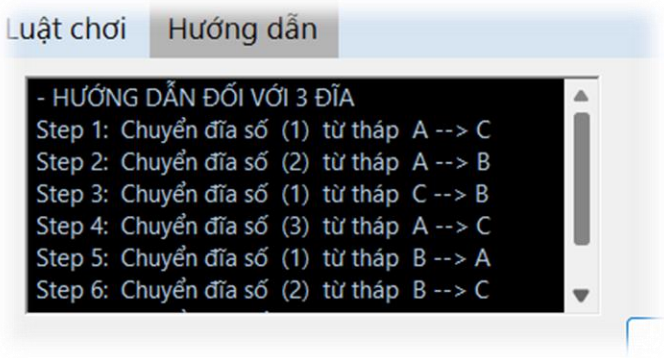
Code behind hiển thị bên dưới của các chức năng đã được giản lược

- *"Luật chơi"*: trên menu có chức năng hiển thị ra một MessageBox thông tin cho người chơi các quy tác của trò chơi Tháp Hà Nội.



Hình 3.3. MessageBox Luật chơi

- *"Hướng dẫn"*: khi được nhấn sẽ hiển thị ra một richTextBox. Ứng dụng thuật toán giải quyết bài toán Tháp Hà Nội đã nêu ở phần 1.3.Thuật toán giải quyết bài toán Tháp Hà Nội để in ra các hướng dẫn cho người chơi hoàn thành trò chơi với số bước tối thiểu.



Hình 3.4. richTextBox Hướng dẫn

Một số điều cần lưu ý:

+ Các biến theo dõi tình trạng: isPLaying: đang chơi, isPause: đang tạm dừng

+ "Hướng dẫn" chỉ nhấn được khi game trong tình trạng isPlaying hoặc isPause, kết thúc trò chơi sẽ đóng box hướng dẫn

+ Chỉ được phép chọn số đĩa khi game đã kết thúc

- *"Bắt đầu / Chơi lại"*: có chức năng bắt đầu game khi người dùng mới vào trò chơi hay khi người dùng đã kết thúc trò chơi. Ta có thể chia những hành động cần thực thi của button này thành hai phần, đưa game về trạng thái ban đầu và khởi tạo một game mới.

+ Reset game về trạng thái ban đầu:

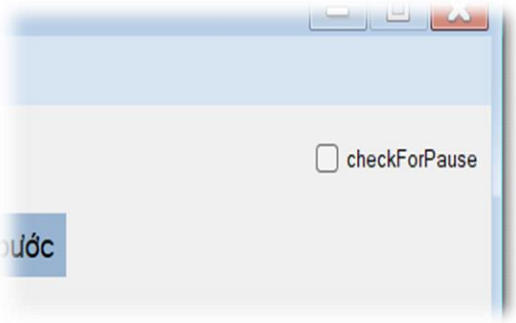
```
timerCountTime.Stop();                    → dừng bộ đếm thời gian
countMove = 0;                            → reset biến đếm về 0
timer = new TimeSpan(0);                  → trả bộ đếm thời gian về 0
minMove = (int)Math.Pow(2, (double)numudNumPlate.Value) - 1;
                                          → tìm minMove mới
foreach (PictureBox plate in plates)
     plate.Visible = false;               → ẩn tất cả các đĩa
towerA.Clear();
towerB.Clear();                           → clear stack các tháp
towerC.Clear();
fClickedPlates = sClickedPlates = null;→ không trỏ đến stack của tháp
RodA.BorderStyle = RodB.BorderStyle = RodC.BorderStyle =
BorderStyle.None;                         → reset border (bỏ focus tháp)
```

+ Khởi tạo game mới:

```
isPlaying = true;                         → Cập nhật trạng thái của game
isPause = false;

expToolStripMenuItem.Enabled = true;      → Cho phép xem hướng dẫn
numudNumPlate.Enabled = false;            → Không cho phép chọn số đĩa
cBoxPause.Checked = false;                → reset lại checkbox để chức
btnPause.Enabled = true;                  năng "Tạm dừng" không bị lỗi
btnGiveUp.Enabled = true;                 → Cho phép đầu hàng
```

```
generatePlates();                    → Thực thi phương thức khởi tạo
                                     đã nêu ở mục 2.4.a
timerCountTime.Start();              → Bắt đầu đếm thời gian
```

- *"Tạm dừng"*: Do chức năng tạm dừng là việc chuyển đổi giữa hai trạng thái là tạm dừng và không tạm dừng nên ta tạo ra một checkBox và kiểm tra value của checkBox đó để xác định tình trạng của game. Khi click vào tạm dừng thì ta sẽ đảo nghịch true - false của cBox, ta cần phải để ý reset lại cBox về false khi nhấn "Chơi lại" để game không bị lỗi. Control checkBox này sẽ được ẩn đối với người dùng. Chức năng của button này là ngừng bộ đếm thời gian khi người dùng muốn tạm dùng trò chơi.




```
isPlaying = !isPlaying;              → Đảo ngược true – false để
isPause = !isPause;                  cập nhật trạng thái của game

cBoxPause.Checked = !cBoxPause.Checked;→ Cập nhật trạng thái của cBox

if (cBoxPause.Checked)
{
      timerCountTime.Stop();
      btnPause.Text = "Tiếp tục";
}
else
{
      timerCountTime.Start();
      btnPause.Text = "Tạm dừng";
}
```

- *"Kết thúc"*: khi được nhấn sẽ dừng toàn bộ trò chơi, xem như người chơi đã đầu hàng và không thể tương tác được bất kỳ chức năng gì trên giao diện ngoài trừ hiển thị luật chơi, chọn số đĩa và bắt đầu một trò chơi mới. Nếu như lúc này box hướng dẫn đang hiển thị thì nó cũng sẽ được đóng lại và màu nền của "Hướng dẫn" sẽ được trả về như ban đầu.

```
isPlaying = false;
isPause = false;
timerCountTime.Stop();

expToolStripMenuItem.BackColor = SystemColors.GradientInactiveCaption;
expToolStripMenuItem.Enabled = false;
richTextBox1.Visible = false;

numudNumPlate.Enabled = true;
btnPlay.Text = "Bắt đầu";
btnPause.Enabled = false;
btnGiveUp.Enabled = false;
```

PHẦN 4: THẢO LUẬN & ĐÁNH GIÁ

4.1. Các kết quả nhận được

Sau khi hoàn thành đồ án, các thành viên trong nhóm đã:

+ Thực sự hiểu rõ hơn về Stack và cách thức hoạt động của nó, ứng dụng Stack vào giải quyết bài toán thực tế.

+ Tiếp thu và hiểu rõ được thuật toán giải quyết bài toán Tháp Hà Nội thông qua giải thuật đệ quy.

+ Học hỏi thêm được các thao tác làm việc với Windows Form Application

+ Nâng cao thêm khả năng xử lý và trình bày giao diện người dùng

+ Hiểu rõ hơn về cách sử dụng Git cũng như Github, làm việc nhóm hiệu quả

4.2. Một số tồn tại

a) Lỗi hiển thị ở màn hình DPI cao

Khi mới bắt đầu tạo dự án Windows Forms và thử chạy chương trình với một vài Button và PictureBox đơn giản thì nhóm đã nhận thấy rằng các Control hiển thị trong giao diện bị mờ và xuất hiện ở vị trí bị lệch đi so với Location đã được định sẵn trong thiết kế, trong khi đó các thành phần ở thanh tiêu đề vẫn rất rõ nét.

Hình 4.1. Giao diện hiển thị bị "vỡ"

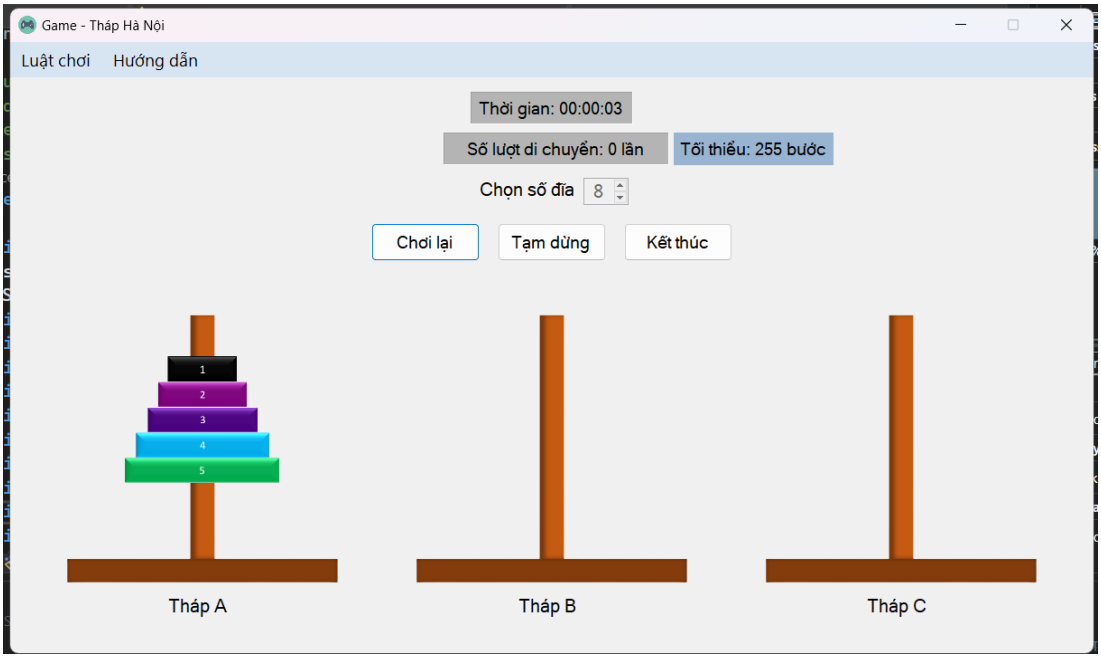
Sau khi tìm hiểu nhóm đã khắc phục được vấn đề này bằng việc thêm vào Project một Application Manifest File (C# Items) và ở trong file app.manifest, với các dòng lệnh đã được tạo sẵn, bỏ comment những dòng sau:

```
<application xmlns="urn:schemas-microsoft-com:asm.v3">
  <windowsSettings>
    <dpiAware
xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</d
piAware>
    <longPathAware
xmlns="http://schemas.microsoft.com/SMI/2016/WindowsSettings">true</l
ongPathAware>
  </windowsSettings>
</application>
```

Xem thêm chi tiết lỗi và những cách khắc phục tại Tài Liệu Tham Khảo

b) Lỗi hiển thị đồ họa với các đĩa

Sau khi trò chơi đã gần như hoàn thiện thì nhóm đã phát hiện ra lỗi đồ họa của các đĩa 6-7-8



Hình 4.2. Các đĩa 6-7-8 không được hiển thị

Lỗi này không ảnh hưởng đến cách thức hoạt động của trò chơi nhưng gây ra sự khó chịu và mất đi tính trực quan. Lỗi này đã được báo cáo từ trước, khi chương trình vẫn đang trong giai đoạn thử nghiệm, nhưng vì đây là một lỗi xuất hiện có điều kiện nên lúc ban đầu nhóm đã nhầm tưởng vấn đề là do máy tính cá nhân của thành viên.

- *Các thao tác dẫn đến lỗi:*

+ Ở lần chơi *đầu tiên*, ta chọn số đĩa bất kỳ trong khoảng từ 3 đến 5

+ Hoàn thành trò chơi

+ Chọn số đĩa từ 6 đến 8 và nhấn "Bắt đầu" → lỗi hiển thị xuất hiện từ đây

Khi này các đĩa từ 6 đến 8 sẽ không được hiển thị trong chương trình nữa

- *Một số điều đã được tìm hiểu:*

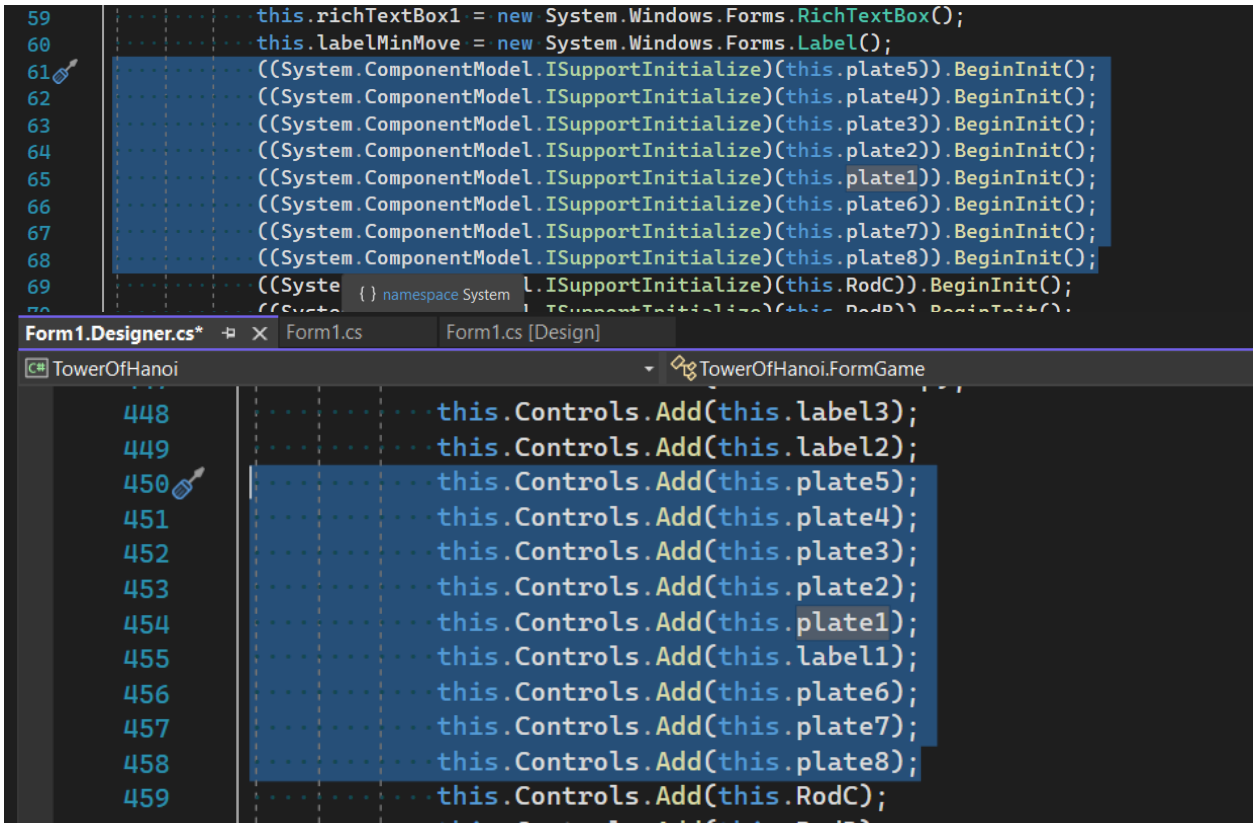
+ Nếu khi bắt đầu trò chơi, ta chọn số đĩa là hai và hoàn thành thì lỗi sẽ không xảy ra, tuy nhiên nếu vẫn ở mức đĩa là hai và ta thao tác chuyển đĩa với số bước di chuyển lớn rồi mới hoàn thành trò chơi thì lỗi vẫn có thể xảy ra

+ Ở lần chơi đầu tiên, nếu ta chọn mức đĩa từ 6 đến 8 rồi bắt đầu chơi thì đồ họa của những đĩa đó sẽ không bị lỗi. Lưu ý rằng nếu ta chỉ chọn 6 đĩa thì đĩa 7 và 8 vẫn sẽ bị lỗi

- *Cách sửa lỗi hiển thị:*

Do khoảng trống mà đĩa 6-7-8 để lại và cách tương tác với các "khoảng trống" ấy trong chương trình tương tự như một đĩa nên nhóm đã kết luận rằng PictureBox của đĩa vẫn nằm ở đấy và lỗi là do thuộc tính Image của đĩa đã không được load. Và các vấn đề về đồ họa sẽ do file *Form1.Designer.cs* quản lý nên nhóm đã thu hẹp phạm vi tìm kiếm lỗi.

Và một điều đáng chú ý nữa là tập đĩa bị lỗi luôn là từ 6 đến 8 nên chắc chắn sẽ có sự khác biệt giữa chúng với 5 đĩa còn lại. Và sau khi kiểm tra lại các dòng lệnh trong *Form1.Designer.cs*, nhóm đã phát hiện một điều rằng thứ tự model của các đĩa được thêm vào và Control của các đĩa được Add không nằm theo thứ tự mà là các đĩa từ 5 đến 1 được thêm vào theo thứ tự rồi mới đến các đĩa 6, 7, 8.



Hình 4.3. Nguyên nhân dẫn đến lỗi hiển thị

Dù vẫn chưa xác định rõ nguyên nhân để giải thích cho các trường hợp gây lỗi nhưng với những suy luận thì nhóm giả định rằng lỗi gây ra là do thứ tự thêm vào các model không hợp lý nên đã khắc phục lỗi bằng các thay đổi thứ tự của các dòng code trên ảnh: control và model của đĩa 8 sẽ được thêm đầu tiên rồi đến các đĩa 7, 6, ... , 1. Và lỗi đã được nhóm khắc phục thành công.

4.3. Hướng phát triển

- Trò chơi này chỉ mới cung cấp cho người chơi trải nghiệm trên phương diện nhìn mà rất nhàm chán về phương diện nghe. Việc thêm âm thanh nhạc nền cho trò chơi là một điều nên được lưu ý đến, nhưng cần thiết hơn đó chính là âm thanh thi người dùng tương tác, ví dụ âm thanh khi chọn tháp, âm thanh khi đặt đĩa vào tháp mới, âm thanh thể hiện rằng một tháp không thể được chọn,...

- Ngoài ra, trò chơi bây giờ chỉ được tương tác thông qua tương tác click chuột, ta có thể phát triển thành tương tác kéo thả. Có thể thêm các Animation vào cho các đĩa để thể hiện chuyển động (tuy nhiên các chuyển động của hình ảnh có thể xuất hiện hiện tự giật khung hình rất khó chịu và mất tính thẩm mỹ)

- "Hướng dẫn" thay vì hiện text hướng dẫn người chơi thì có thể tự động thể hiện các bước giải trực tiếp trên các đĩa và tháp với tốc độ hợp lý để người chơi có thể quan sát

- Ta có thể tạo một file để lưu thành tích mỗi lần chơi của người dùng để có thể tìm ra kỷ lục và thông báo cho người chơi để họ có thể phá kỷ lục ấy

PHỤ LỤC

1. Link mã nguồn (Github):

https://github.com/SunnyBiolie/DoAn_CTDL-GT.git

2. Hướng dẫn cách cài đặt:

• Nếu bạn sử dụng Visual Studio (từ 2017 trở lên):

- Nếu bạn đang ở màn hình khởi động của Visual Studio:
  + Ở mục *Get started*, chọn *Clone a repository*
  + Ở mục *Repository location*, copy link Github ở phía trên vào
  + Chọn nơi lưu ở mục *Path*, cuối đường dẫn chính là tên folder sẽ chứa Project
  + Nhấn *Clone*

- Nếu bạn đang ở trong Visual Studio:
  + Ở trên thanh menu phía trên bên trái bạn sẽ tìm thấy *Git*
  + Click vào *Git* và chọn *Clone Repository...*

  Một cửa sổ với tiêu đề "Clone a repository" sẽ xuất hiện
  + Ở mục *Repository location*, copy link Github ở phía trên vào
  + Chọn nơi lưu ở mục *Path*, cuối đường dẫn chính là tên folder sẽ chứa Project
  + Nhấn *Clone*

• Cách khác: sử dụng Git / yêu cầu máy phải cài đặt sẵn Git, nếu chưa có Downloads Git tại đây: https://git-scm.com/downloads

- Tại thư mục nơi bạn muốn clone Project về:

+ Nhấn chuột phải chọn *Open in Terminal*

+ Bạn nhập lệnh với cú pháp:

```
git clone https://github.com/SunnyBiolie/DoAn_CTDL-GT.git
link Project_CTDL-GT
```

Trong đó `Project_CTDL-GT` là tên folder nơi bạn lưu Project về.

- Hoặc: bạn có thể mở riêng *Windows PowerShell* hoặc *Command Prompt*, đưa đường dẫn đến nơi cần lưu và nhập lệnh với cú pháp như trên.

3. Phân công công việc:

Thành viên	Nhiệm vụ
Phạm Nguyễn Chí Khoa	Tổng hợp và trình bày báo cáo Thiết kế giao diện Tổng hợp và trình bày mã nguồn
Nguyễn Tiến Dũng	Xử lý các sự kiện trong chương trình Thiết kế hình ảnh các đĩa và cọc
Lã Văn Hà	Xử lý làm việc với Stack Hỗ trợ thuật toán, thiết kế giao diện
Phan Vũ Trường An	Hỗ trợ viết báo cáo, xử lý logic Tìm kiếm và báo cáo lỗi Tìm kiếm tài liệu, thông tin hữu ích

TÀI LIỆU THAM KHẢO

- BuaXua.vn: Trò chơi tháp Hà Nội
https://www.buaxua.vn/tro-choi/tro-choi-thap-ha-noi

- Hiếu Hoàng: Giao diện Windows Forms trên màn hình DPI cao
https://hieuhoang.dev/giao-dien-windows-forms-tren-man-hinh-dpi-cao

- HowKteam: Lập trình Winform cơ bản (series)
https://howkteam.vn/course/lap-trinh-winform-co-ban-27

- Wikipedia: Tháp Hà Nội
https://vi.wikipedia.org/wiki/Tháp_Hà_Nội

- Microsoft Documentation: Stack Class (series)
https://learn.microsoft.com/en-us/dotnet/api/system.collections.stack?view=net-7.0