MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Tên đầy đủ
1	USD	United States Dollar - đồng đô la Mỹ
2	IDE	Integrated Development Environment
3	ASCII	American Standard Code for Information Interchange
4	PYPL	PopularitY of Programming Language

LỜI MỞ ĐẦU

1. Lý do chọn đề tài

Xã hội ngày càng phát triển, nhu cầu giải trí của con người ngày càng tăng cao. Cùng với sự phát triển của công nghệ thông tin, điện tử, các thiết bị chơi game ngày càng phổ biến. Ngành công nghệ game cũng phát triển nhanh chóng trong những năm gần đây, với doanh thu khổng lồ. Game không chỉ là một phương tiện giải trí đơn thuần, mà còn là một lĩnh vực đáng quan tâm và theo đuổi. Thiết kế game là một công việc đòi hỏi rất nhiều thời gian và công sức, nhưng lại có thể biến những dòng code khô khan thành những trải nghiệm thú vị cho người chơi. Tuy nhiên, không thể nào quên được "Snake" hay "Game con rắn" - tựa game được xem là thủy tổ của đế chế giải trí - được Taneli Armanto khai sinh năm 1997 tại Phần Lan và trở thành hiện tượng của toàn cầu. Không những trò chơi này được cài đặt sẵn trên 350 triệu chiếc điện thoại được bán ra tính đến năm 2005 mà những giải đấu trò chơi này cũng được tổ chức từ cấp địa phương đến quốc gia thậm chí là quốc tế. Dần dần, "Game con rắn" đã trở thành một biểu tượng trong thế giới game và được tái tạo trên nhiều nền tảng khác nhau tiên phong cho ngành công nghiệp hàng chục tỷ USD.

Tuy ngày nay Smartphone ra đời và hiện diện hầu như trong mọi khía cạnh cuộc sống và Game Moblie cũng đã phát triển rất mạnh mẽ thu hút được hơn 3,2 tỷ người chơi trên toàn cầu (theo báo cáo của Newzoo năm 2022) nhưng hương vị cổ điển, sơ khai này nào của Game con rắn trên chiếc Nokia 6110 vẫn là một hương vị không thể quên.

Với cách chơi đơn giản nhưng vô cùng thử thách, trò chơi "Snake" đã trở thành một trong những trò chơi kinh điển trong lịch sử game. Việc lập trình một phiên bản của trò chơi này sẽ giúp cho chúng ta hiểu rõ hơn về cách thức hoạt động của game và có thể nâng cao kỹ năng lập trình của mình.

Với lý do nêu trên, em quyết định chọn đề tài “Chương trình Game con rắn trên C#” cho bài Đồ án cuối kỳ để tìm hiểu về khởi thủy của Game Mobile và cũng để củng cố và học hỏi thêm kiến thức về ngôn ngữ C#.

2. Mục tiêu nghiên cứu

Xây dựng trò chơi đơn giản giúp người chơi giải trí sau những giờ học mệt mỏi căng thẳng.

Ôn tập và tìm hiểu thêm những kiến thức ở môn Cơ sở lập trình

- Các khái niệm cơ bản về lập trình game
- Lập trình C# và .NET Framework, bao gồm cách sử dụng các thư viện và công cụ để xây dựng game.
- Các kỹ thuật lập trình game như đồ họa, âm thanh, và các hiệu ứng đặc biệt khác.
- Các trình biên dịch, trình gỡ lỗi, và các IDE phổ biến.

Vận dụng và tư duy để giải quyết những bài toán thực tế cụ thể.

3. Phạm vi nghiên cứu

Các khái niệm cơ bản của lập trình game

Các kiến thức về lập trình C# và .NET Framework, bao gồm cách sử dụng các thư viện và công cụ để xây dựng game.

Các kỹ thuật lập trình game như đồ họa, âm thanh, và các hiệu ứng đặc biệt khác.

Thiết kế và xây dựng game con rắn trên C# từ đầu đến cuối, bao gồm việc lập trình các tính năng chính của game, như chuyển động, ăn mồi, và xử lý va chạm.

Tài liệu hướng dẫn sử dụng các công cụ phát triển game trên C#, bao gồm các trình biên dịch, trình gỡ lỗi, và các IDE phổ biến.

4. Mô tả tài liệu

Nội dung đồ án Chương trình Game con rắn trên C# gồm có 3 chương chính:

Chương I: Giới thiệu tổng quan Game con rắn

Chương II: Cơ sở lý thuyết

Chương III: Phân tích thiết kế hệ thống

Chương IV: Ý tưởng thuật toán và cài đặt

PHẦN NỘI DUNG

Chương I- GIỚI THIỆU TỔNG QUAN

1. Yêu cầu chung

- Cho phép chọn mức chơi
- Khởi tạo, hiển thị con rắn và thức ăn
- Điều khiển con rắn theo các hướng được nhập từ bàn phím
- Xử lý tình huống rắn chạm tường, chướng ngại vật và tự cắn mình
- Xử lý chiều dài của rắn khi ăn thức ăn và tốc độ của rắn qua mỗi mức chơi
- Hiển thị điểm người chơi và level đang chơi

2. Mục tiêu

- Mô phỏng trò chơi Snake (Game con rắn) trên phần mềm Visual Studio 2022 bằng ngôn ngữ C#.

3. Đối tượng người dùng

- Những người yêu thích các trò chơi arcade đơn giản và dễ chơi.
- Những người muốn thử thách khả năng phản xạ của mình trong một trò chơi đơn giản nhưng hấp dẫn.
- Những người muốn giải trí và giảm stress sau những giờ làm việc căng thẳng.
- Các em nhỏ và thanh thiếu niên đam mê trò chơi điện tử và muốn tìm hiểu thêm về lập trình game.
- Những người muốn chơi game mà không cần phải đầu tư quá nhiều thời gian và tiền bạc.

Chương II- CƠ SỞ LÝ THUYẾT

1. Yêu cầu chung (General requirement)

1.1. Về C# và lý do nên chọn C#

C# là một loại ngôn ngữ lập trình hướng đối tượng đơn giản được đội ngũ kỹ sư tại Microsoft phát triển trên nền tảng của C++ và Java.

Ngôn ngữ này cho phép người dùng sử dụng nhiều ngôn ngữ bậc cao trên nhiều nền tảng và cấu trúc máy tính. Đây cũng là một trong số ít các ngôn ngữ lập trình có sự cân bằng giữa Java, C++, Visual Basic và cả Delphi.

C# làm việc trên framework .NET, có thể tạo ra các ứng dụng vừa mạnh mẽ vừa an toàn trên nền tảng Windows, các ứng dụng di động, web, v.v..

1.2. Lịch sử:

- Năm 2000, một kỹ sư Đan Mạch có tên Microsoft's Anders Hejlsberg đã sáng tạo ra C#. Ngoài ra, ông còn nổi tiếng về các phát minh nổi bật như tạo ra các công cụ và ngôn ngữ lập trình đáng tin cậy gồm Microsoft's TypeScript và Delphi, một sự thay thế phù hợp cho Turbo Pascal.
- Tháng 2 năm 2019, ngôn ngữ lập trình C# đã đứng hạng thứ 4 vượt PYPL và chỉ đứng sau Java và JavaScript. Trong đó, dữ liệu chính được sử dụng để biên dịch chỉ mục này dựa trên tần suất tìm kiếm hướng dẫn về các ngôn ngữ lập trình khác nhau trong Google.
- Hiện nay, C# ngày càng trở nên phổ biến trong danh sách mười ngôn ngữ lập trình hàng đầu của TIOBE Index được chứng minh từ các công cụ tìm kiếm phổ biến như Google, YouTube và Bing.

1.3. Đặc trưng:

- Đơn giản: Đặc trưng đầu tiên của C# là loại bỏ những vấn đề phức tạp đã có trong Java và C++ như macro, template, tính đa kế thừa, lớp cơ sở ảo (hay còn gọi virtual base class). Các cú pháp, toán tử, biểu thức và cả tính năng của C# khá tương đương Java và C++ song đã qua cải tiến nên đơn giản hơn nhiều.
- Hiện đại: C# sở hữu nhiều khả năng như xử lý ngoại lệ, tự động trong thu gom bộ nhớ, bảo mật mã nguồn, dữ liệu mở rộng,... Đây là tất cả những đặc điểm được mong chờ ở một ngôn ngữ lập trình hiện đại.
- Hướng đối tượng: C# là một trong những ngôn ngữ được đánh giá là thuần hướng đối tượng. Nó sở hữu cả 4 tính chất quan trọng, đặc trưng là tính kế thừa, tính đóng gói, tính trừu tượng và tính đa hình.

- Ít từ khóa: Một trong những đặc trưng cơ bản của C# là ít từ khóa. Từ khóa được dùng trong ngôn ngữ này chỉ nhằm mục đích mô tả thông tin. Tuy ít từ khóa song C# vẫn rất mạnh mẽ. Lập trình viên có thể sử dụng nó để thực hiện mọi nhiệm vụ.
- Mã nguồn mở: C# là một trong những ngôn ngữ lập trình mã nguồn mở, được phát triển, điều hành một cách độc lập với Microsoft. Đây là một trong những nét độc đáo khiến ngôn ngữ này được biết đến và ưa chuộng rộng rãi.
- Đa nền tảng: C# là ngôn ngữ được sử dụng trong lập trình trên nhiều nền tảng. Các ứng dụng hoặc website được xây dựng bằng ngôn ngữ này có thể hoạt động tốt trên nhiều nền tảng như Windows, Linux và Mac.
- Tiến hóa: C# vẫn đang được nâng cấp và cho ra mắt các phiên bản mới với nhiều tính năng vượt trội và khả năng làm việc mạnh mẽ hơn. Hiện C# có thể làm việc với console, điện toán đám mây, phần mềm học máy,...

1.4. Ưu điểm:

- C# gần gũi với Java và C++, nhờ vậy mà nó kế thừa được tất cả các 'tinh hoa' của hai ngôn ngữ này. Lập trình viên có kiến thức về hai ngôn ngữ trên có thể dùng C# dễ dàng.
- Cộng đồng những người sử dụng C# đang phát triển với tốc độ chóng mặt. Lập trình viên có thể tham khảo và tìm kiếm thông tin dễ dàng.
- C# có khả năng tạo ra mọi ứng dụng và phổ biến trong giới lập trình. Đặc biệt là lập trình game.

1.5. Nhược điểm: chỉ chạy trên nền Windows và có cài .NET Framework. Thao tác đối với phần cứng yếu hơn so với ngôn ngữ khác, hầu hết phải dựa vào windows.

2. Phương thức tĩnh (Static)

- Static là từ khóa static (sẽ tìm hiểu kỹ hơn ở bài sau). Có thể không sử dụng cũng được. Nhưng ở trường hợp hàm Main của console C# thì phải có.

- Void là kiểu trả về. Với hàm có kiểu trả về là void thì sẽ không cần từ khóa return trong hàm. Hoặc có nhưng chỉ đơn giản là ghi return;

3. Vòng lặp for

Cú pháp:

```
for ([Khởi tạo]; [Điều kiện lặp]; [Bước lặp lại])
{
        // Khối lệnh được lặp lại. Có thể bỏ trống
}
```

Tiến trình:

- Ban đầu trình biên dịch sẽ di vào phần khởi tạo chạy đoạn lệnh khởi tạo.
- Tiếp theo kiểm tra điều kiện lặp. Rồi thực hiện khối code bên trong vòng lặp for. Khi đến ký hiệu } thì sẽ quay lên bước lặp lại.
- Sau đó lại kiểm tra điều kiện lặp rồi tiếp tục thực hiện đoạn code trong khối lệnh. Đến khi điều kiện lặp không còn thõa mãn thì sẽ kết thúc vòng lặp for.
- Trường hợp khác:

```
for (; ;)
{
        // Khối lệnh được lặp lại. Có thể bỏ trống
}
```

* Khi đó: vòng lặp for này trở thành vòng lặp vô tận.

4. Vòng lặp While

Cú pháp:

```
    while (<Điều kiện lặp>)
        {
            // khối lệnh lặp lại
        }
```

Điều kiện lặp là một biểu thức logic bắt buộc phải có với kết quả trả về bắt buộc là true hoặc false.

Từ khóa while biểu thị đây là một vòng lặp while. Các câu lệnh trong khối lệnh sẽ được lặp lại đến khi không còn thỏa mãn điều kiện lặp sẽ kết thúc vòng lặp while.

Tiến trình:

- Đầu tiên trình biên dịch sẽ đi vào dòng while (<Điều kiện lặp>). Kiểm tra điều kiện lặp có thỏa mãn hay không. Nếu kết quả là true thì sẽ đi vào bên trong thực hiện khối code. Khi gặp ký tự } sẽ quay lên kiểm tra điều kiện lặp và tiếp tục thực hiện khối code. Quá trình chỉ kết thúc khi điều kiện lặp là false.
- Điều kiện lặp luôn bằng true thì vòng lặp while sẽ trở thành vòng lặp vô tận.
- Điều kiện lặp luôn bằng false thì vòng lặp sẽ không được thực thi.

5. Câu lệnh if

Dạng thiếu

Cú pháp: If ([Biểu thức điều kiện]) <Câu lệnh thực hiện>

- If là từ khóa bắt buộc.
- <Biểu thức điều kiện> là biểu trức dạng boolean (trả về true hoặc false).
- <Câu lệnh thực hiện> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là đúng.

Ý nghĩa:

- Nếu <Biểu thức điều kiện> trả về true thì thực hiện <Câu lệnh thực hiện> ngược lại thì không làm gì cả.

Dạng đủ

Cú pháp:

```
If <Biểu thức điều kiện>
        <Câu lệnh thực hiện 1>
else
        <Câu lệnh thực hiện 2>
```

- If, else là từ khóa bắt buộc.
- <Biểu thức điều kiện> là biểu trức dạng boolean (trả về true hoặc false).
- <Câu lệnh thực hiện 1> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là đúng.

- <Câu lệnh thực hiện 2> là câu lệnh muốn thực hiện nếu <Biểu thức điều kiện> là sai.

Ý nghĩa:

- Nếu <Biểu thức điều kiện> trả về true thì thực hiện <Câu lệnh thực hiện 1> ngược lại thì thực hiện <Câu lệnh thực hiện 2>.

6. Cấu trúc Switch case

Cú pháp:

```
switch (<biểu thức>)
{
case <giá trị thứ 1>: <câu lệnh thứ 1>;
                    break;
case <giá trị thứ 2>: <câu lệnh thứ 2>;
                    break;
. . .
case <giá trị thứ n>: <câu lệnh thứ n>;
                    break;
}
```

Trong đó:

- switch, case là từ khóa bắt buộc.
- break là một lệnh nhảy
- Ý nghĩa của nó là thoát ra khỏi cấu trúc, vòng lặp chứa nó (khái niệm về vòng lặp sẽ được trình bày ở bài CẤU TRÚC LẶP GOTO TRONG C#)
- Ngoài break ra vẫn còn lệnh nhảy khác như goto nhưng ít được sử dụng (chi tiết về lệnh goto sẽ được trình bày trong bài CẤU TRÚC LẶP GOTO TRONG C#).
- Vì trong cấu trúc switch. . . case chủ yếu chỉ sử dụng lệnh break nên mình cố tình để lệnh break vào trong cú pháp thay vì ghi chung chung là lệnh nhảy.
- <biểu thức> phải là biểu thức trả về kết quả kiểu:
- Số nguyên (int, long, byte, . . .)
- Ký tự hoặc chuỗi (char, string)
- Kiểu liệt kê (enum, sẽ được trình bày trong bài ENUM TRONG LẬP TRÌNH C#)

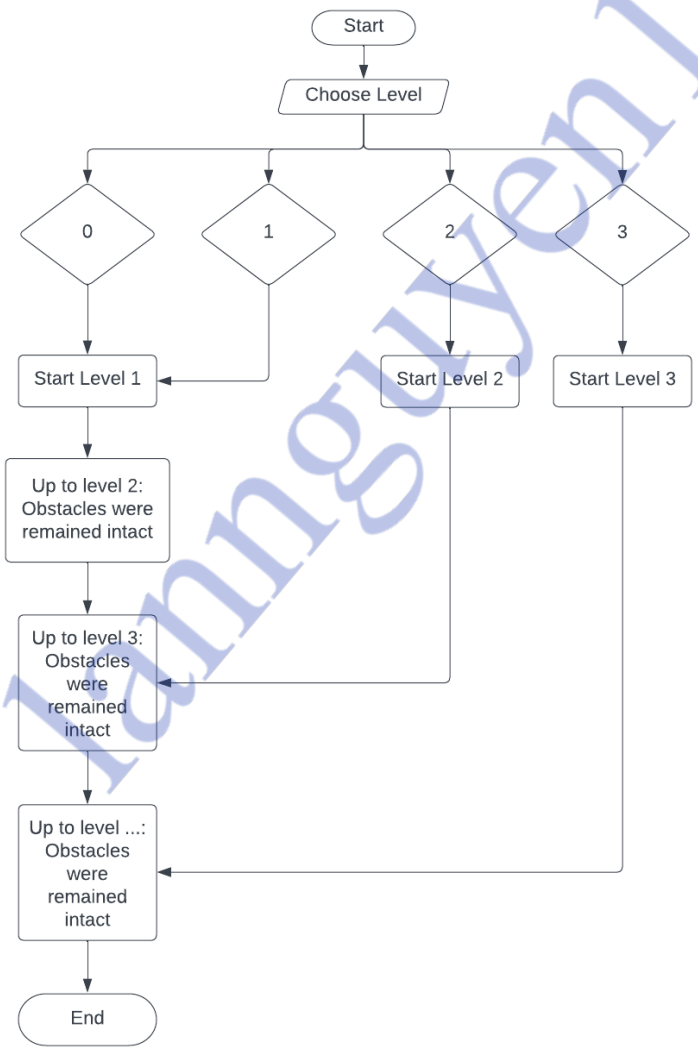
- <giá trị thứ i> với i = 1..n là giá trị muốn so sánh với giá trị của <biểu thức>.
- <câu lệnh thứ i> với i = 1..n là câu lệnh muốn thực hiện khi <giá trị thứ i> tương ứng bằng với giá trị của <biểu thức>.

Ý nghĩa:

- Duyệt lần lượt từ trên xuống dưới và kiểm tra xem giá trị của <biểu thức> có bằng với <giá trị thứ i> đang xét hay không. Nếu bằng thì thực hiện <câu lệnh thứ i> tương ứng.

Chương III- PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. Khởi động trò chơi



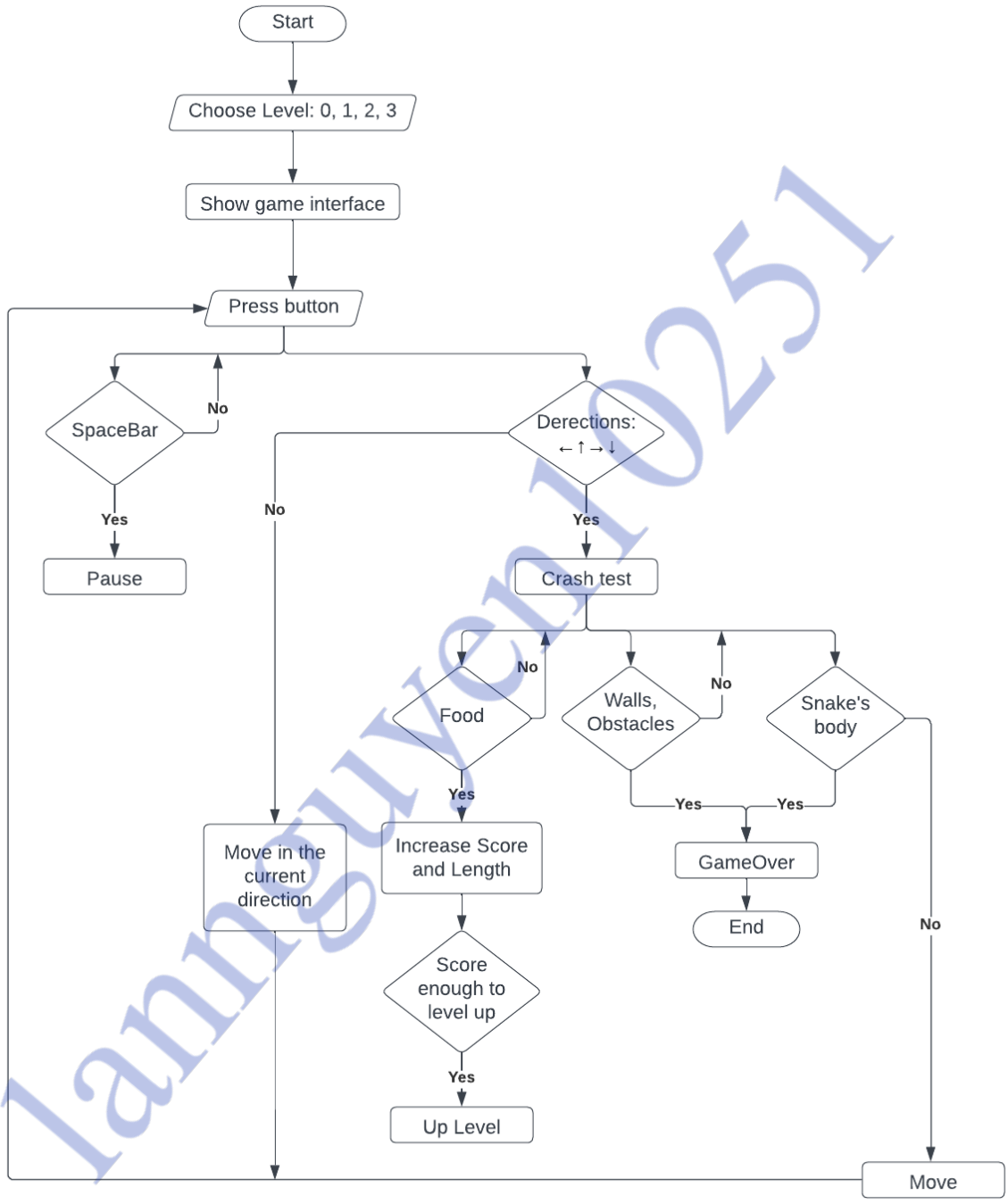
Hình 1 – Flowchart khởi tạo trò chơi.

Hướng thứ nhất: Người chơi thực hiện trò chơi, chương trình thực hiện cộng điểm, đến giới hạn điểm chương trình tự động nâng mức độ của trò chơi lên.

Hướng thứ hai: Người chơi thực hiện chọn mức độ trò chơi và thức hiện trò chơi đến khi khởi tạo lại trò chơi.

Hướng thứ ba: Người chơi thực hiện trò chơi, trong quá trình chơi người chơi muốn tạm dừng hoặc tiếp tục trò chơi.

2. Các chức năng chi tiết



Hình 2 – Flowchart các chức năng chính của trò chơi.

- Bắt đầu chơi:
    - Người chơi nhập phím “0” hoặc “1” hoặc “2” hoặc “3” để chọn mức độ chơi và nhấn phím “Enter” để bắt đầu.
- Tạm dừng trò chơi:

- Trong quá trình chơi, khi muốn tạm dừng trò chơi lại, người chơi nhấn phím “SpaceBar” để tạm dừng.
- Để tiếp tục trò chơi, người chơi nhấn phím “SpaceBar”.

- Hiển thị thức ăn và chướng ngại vật:
  - Thức ăn sẽ xuất hiện ngẫu nhiên, sau khi rắn “ăn” đi 1 thức ăn thì sẽ xuất hiện 1 thức ăn ở vị trí ngẫu nhiên khác.
  - Chướng ngại vật ngày càng nhiều, 1 thức ăn hiện ra là 1 vật cản hiện ra.
- Di chuyển:
  - Chức năng này cho phép người chơi điều khiển rắn di chuyển theo hướng mong muốn:
  - Di chuyển sang trái: ←
  - Di chuyển sang phải: →
  - Di chuyển đi lên: ↑
  - Di chuyển đi xuống: ↓
- Tính điểm: Khi người chơi điều khiển rắn ăn được một thức ăn thì biến “Score” sẽ tăng lên 10đ.
- Mức độ và hoàn thành
  - mức 1: đủ 50đ sang mức 2 (tốc độ nhanh hơn mức 1 và vật cản nhiều hơn)
  - mức 2: đủ 100đ sáng mức 3 (tốc nhanh hơn và có vật cản nhiều hơn mức 2)
  - mức 3: đủ 150đ sang mức 4 (tốc nhanh hơn và có vật cản nhiều hơn mức 3, độ dài rắn sẽ cản trở di chuyển làm trò chơi khó hơn)
  - Trò chơi không quy định kết thúc, trò chơi sẽ tiếp tục nếu người chơi vượt qua được các thử thách.
- Kết thúc game: khi đụng trúng tường, tự cắn vào mình, đụng vật cản thì GameOver

Chương IV- Ý TƯỞNG PHÁT TRIỂN THUẬT TOÁN VÀ CÀI ĐẶT

1. Các biến toàn cục (global variables)

Các biến toàn cục được khai báo bên ngoài phương thức Main() và có thể truy cập được từ bất kỳ phương thức nào trong lớp SnakeGame, bao gồm các biến kiểu int và bool để lưu trữ thông tin điểm số, cấp độ, tốc độ, tình trạng game, mảng hai chiều để lưu trữ thông tin bảng chơi, các đối tượng kiểu Point để lưu trữ thông tin về vị trí và chiều hướng của rắn. Ngoài ra, còn có đối tượng kiểu Random để tạo ra các giá trị ngẫu nhiên và đối tượng kiểu object để đồng bộ hóa các thread.

BOARD_WIDTH: biến kiểu int được khởi tạo giá trị là 40.

BOARD_HEIGHT: biến kiểu int được khởi tạo giá trị là 20.

FOOD_SCORE: biến kiểu int được khởi tạo giá trị là 10.

LEVEL_SCORE: biến kiểu int được khởi tạo giá trị là 50.

```
static int BOARD_WIDTH = 40;
static int BOARD_HEIGHT = 20;
static int FOOD_SCORE = 10;
static int LEVEL_SCORE = 50;
```

score: biến kiểu int được khởi tạo giá trị là 0.

level: biến kiểu int được khởi tạo giá trị là 1.

sleepTime: biến kiểu int được khởi tạo giá trị là 200.

GameOver: biến kiểu bool được khởi tạo giá trị là false.

Pause: biến kiểu bool được khởi tạo giá trị là false.

```
static int score;
static int level;
static int sleepTime;
static bool GameOver;
static bool Pause;
```

board: mảng hai chiều kiểu int với kích thước BOARD_WIDTH x BOARD_HEIGHT.

```
static int[,] board;
```

random: đối tượng kiểu Random.

```
static Random random = new Random();
```

direction: biến direction một enum (kiểu liệt kê) gồm 4 giá trị: Left, Right, Up và Down. Biến này sẽ lưu trữ hướng di chuyển hiện tại của con rắn.

```
static Direction direction;
```

lockObject: biến lockObject được khai báo với kiểu dữ liệu object, biến này được sử dụng để đồng bộ hóa các thread đang chạy trong trò chơi.

```
static object lockObject = new object();
```

head: một đối tượng kiểu Point đại diện cho đầu rắn.

body: một mảng đối tượng kiểu Point đại diện cho thân rắn.

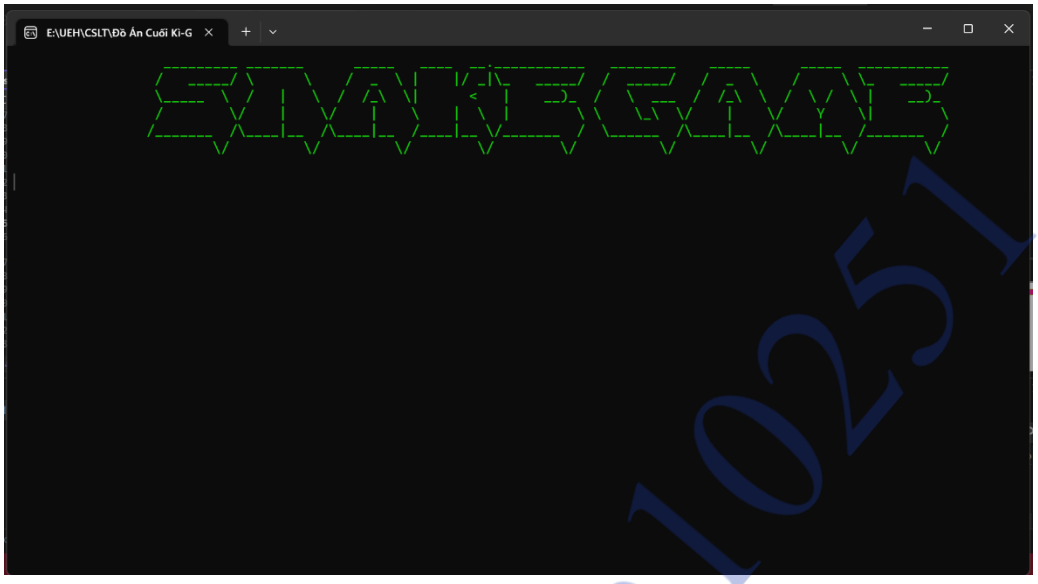
2. Các hàm và vai trò của nó

2.1. Main(): Phương thức chính của chương trình, là điểm bắt đầu của chương trình.

- In ra màn hình tên game (ASCII art).
- Thiết lập màu sắc của chữ in ra thành màu xanh lá cây.
- Chờ trong 2 giây rồi xóa màn hình.
- Chọn mức độ chơi bằng hàm ChooseLevel().
- Tạo một đối tượng game mới của lớp SnakeGame.
- Gọi hàm Start() của lớp SnakeGame để bắt đầu trò chơi.

```
static void Main()
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine(@"                    _________ _______
_____   ____ __.___________   ________    _____      _____
___________
                /   _____/ \      \    /  _  \ |    |/ _|\_
_____/ /   _____/   /  _  \    /     \ \_   _____/
                \_____  \  /   |   \  /  /_\  \|      <   |
__)_  /    \  ___  /  /_\  \  /  \ /  \ |    __)_
                /        \/    |    \/    |    \    |  \  |
\ \    \_\  \/    |    \/    Y    \|        \
                /_______  /\____|__  /\____|__  /____|__
\/_______  /  \_____  /\____|__  /\____|__  /_______  /
                     \/        \/        \/        \/
\/         \/        \/        \/        \/
");
        Console.ResetColor();
        Thread.Sleep(2000);
        Console.Clear();
        ChooseLevel();
```

```
        SnakeGame game = new SnakeGame();
        SnakeGame.Start();
    }
```



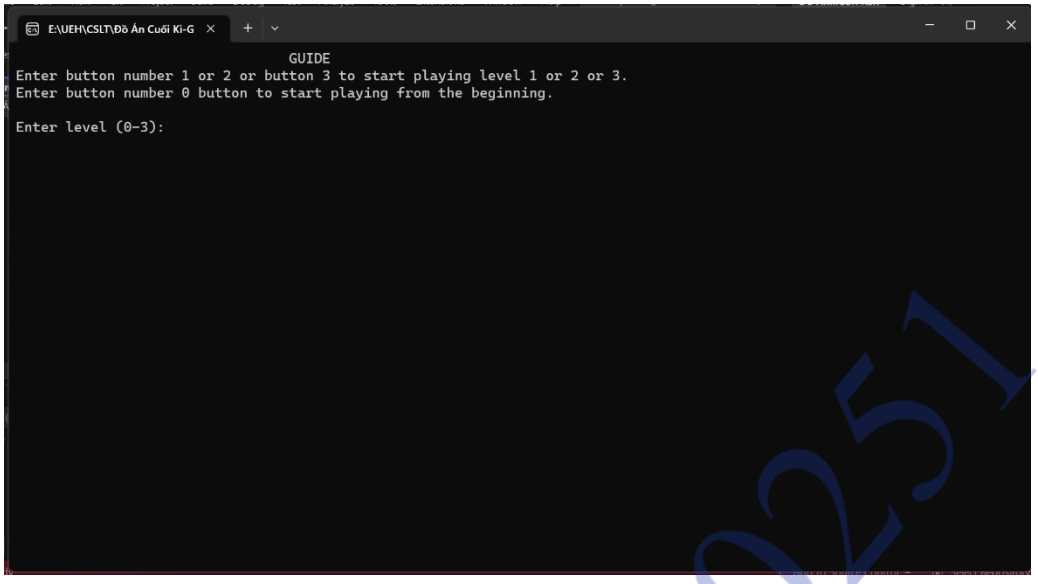
Hình – 3 Giao diện khi khởi động trò chơi

2.2. ChooseLevel(): Hàm này cho phép người dùng chọn cấp độ của trò chơi, từ đó ảnh hưởng đến độ khó của trò chơi.

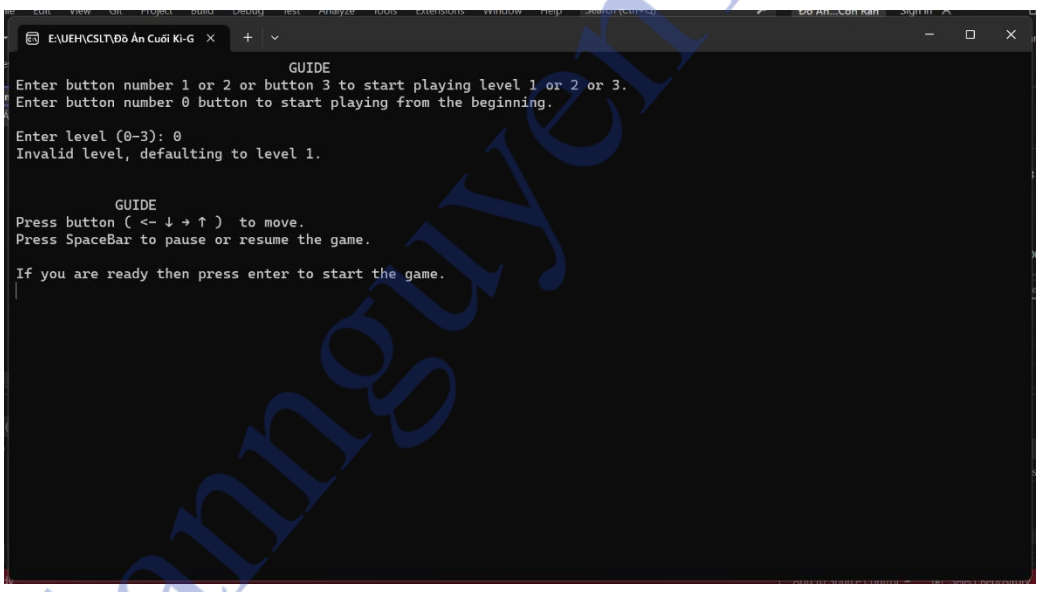
- In ra hướng dẫn chọn level và yêu cầu người chơi nhập số tương ứng với level mà họ muốn chơi.
- Đọc giá trị được nhập vào từ bàn phím và lưu vào biến selectedLevel.
- Kiểm tra xem giá trị selectedLevel có nằm trong khoảng từ 1 đến 3 hay không. Nếu có, thì gán giá trị của selectedLevel cho biến level.
- Nếu giá trị selectedLevel không nằm trong khoảng từ 1 đến 3, in ra thông báo lỗi và gán giá trị mặc định là 1 cho biến level.
- Dựa trên giá trị của biến level, gán giá trị cho biến sleepTime theo các case tương ứng.

```
static void ChooseLevel()
    {
        Console.WriteLine("\t \t \t \t GUIDE");
        Console.WriteLine("Enter button number 1 or 2 or button 3
to start playing level 1 or 2 or 3.");
        Console.WriteLine("Enter button number 0 button to start
playing from the beginning.");

        Console.Write("\nEnter level (0-3): ");
```

```
            int selectedLevel = Convert.ToInt32(Console.ReadLine());
            if (selectedLevel >= 1 && selectedLevel <= 3)
            {
                level = selectedLevel;
            }
            else
            {
                Console.WriteLine("Invalid level, defaulting to level
1.");
                level = 1;
            }
            Console.WriteLine();
            Console.Write(@"
               GUIDE
Press button ( <- ↓ → ↑ )  to move.
Press SpaceBar to pause or resume the game.

If you are ready then press enter to start the game.
");
            Console.ReadLine();
            Console.Clear();
             switch (level)
             {
                 case 1:
                     sleepTime = 200;
                     break;
                 case 2:
                     sleepTime = 150;
                     break;
                 case 3:
                     sleepTime = 100;
                     break;
                 case 0:
                     sleepTime = 200;
                     break;
             }
        }
```



Hình – 4 Giao diện chọn mức chơi



Hình 5 – Giao diện hướng dẫn chơi

2.3. PlaceFood(): Hàm này đặt thức ăn vào một vị trí ngẫu nhiên trên bảng, nếu đã có một vật cản hoặc thức ăn ở vị trí đó rồi thì hàm sẽ đặt thức ăn vào một vị trí khác.

- Khởi tạo giá trị x và y bằng cách sử dụng hàm random.Next(BOARD_WIDTH) và random.Next(BOARD_HEIGHT). Hai phương thức này sẽ trả về một giá trị ngẫu nhiên trong khoảng từ 0

đến BOARD_WIDTH - 1 và BOARD_HEIGHT - 1, tương ứng với vị trí của x và y trên bảng chơi.

- Kiểm tra xem vị trí (x,y) đã được sử dụng trước đó chưa bằng cách kiểm tra giá trị của board[x,y]. Nếu giá trị này khác 0, tức là ô đã được sử dụng, ta sẽ thực hiện bước tiếp theo.
- Lặp lại bước 1 để khởi tạo một vị trí mới cho thức ăn cho đến khi tìm được vị trí không được sử dụng trên bảng chơi.
- Đặt giá trị -1 vào ô tại vị trí (x,y) trên bảng chơi để đánh dấu đó là vị trí của thức ăn.

```
static void PlaceFood()
    {
        int x = random.Next(BOARD_WIDTH);
        int y = random.Next(BOARD_HEIGHT);

        while (board[x, y] != 0)
        {
            x = random.Next(BOARD_WIDTH);
            y = random.Next(BOARD_HEIGHT);
        }

        board[x, y] = -1;
    }
```

2.4. PlaceObstacle(): Hàm này đặt vật cản vào một vị trí ngẫu nhiên trên bảng, nếu đã có một vật cản hoặc thức ăn ở vị trí đó rồi thì hàm sẽ đặt vật cản vào một vị trí khác.

- Khởi tạo giá trị ngẫu nhiên cho x và y trong khoảng từ 0 đến BOARD_WIDTH và BOARD_HEIGHT tương ứng bằng phương thức random.Next().
- Sử dụng một vòng lặp while, kiểm tra xem ô tại vị trí [x, y] trên bảng có bị chiếm bởi thức ăn (-1) hoặc chướng ngại vật (2) hay không. Nếu có, thực hiện lại bước 1 để lấy một vị trí mới cho x và y. Vòng lặp sẽ tiếp tục cho đến khi tìm được một ô trống (có giá trị là 0).
- Gán giá trị 2 cho ô trống tìm được để đặt chướng ngại vật.

```
static void PlaceObstacle()
    {
```

```
        int x = random.Next(BOARD_WIDTH);
        int y = random.Next(BOARD_HEIGHT);

        while (board[x, y] != 0 && board[x, y] == -1)
        {
            x = random.Next(BOARD_WIDTH);
            y = random.Next(BOARD_HEIGHT);
        }

        board[x, y] = 2;
    }
```

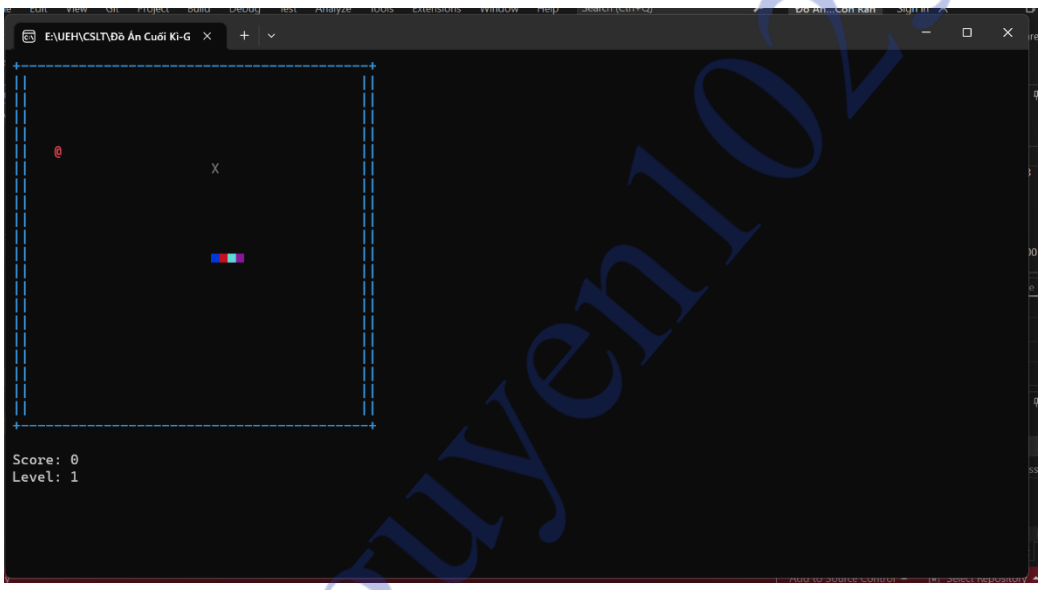
2.5. void DrawBoard(): Hàm này vẽ bảng trò chơi và hiển thị các vật thể trên bảng (rắn, thức ăn và vật cản).

- Đặt con trỏ ở góc trái của console và ẩn con trỏ.
- In ra một dòng gạch ngang trên cùng của bảng chơi.
- Vẽ nội dung của bảng chơi theo từng hàng và từng cột. Nếu ô đó chưa có gì, in ra khoảng trắng. Nếu ô đó chứa chướng ngại vật (-1), in ra ký tự @ màu đỏ. Nếu ô đó là ô mà con rắn đang đứng (2), in ra ký tự X màu xám đậm. Nếu ô đó có điểm số (khác 0 và -1), in ra ký tự hình chữ nhật kích thước 2x1 với màu sắc ngẫu nhiên.
- In ra một dòng gạch ngang dưới cùng của bảng chơi.
- In ra điểm số và mức độ của trò chơi ở phía dưới bảng chơi.
- Cuối cùng, vẽ 4 góc của bảng chơi bằng cách đặt con trỏ đến các vị trí tương ứng và in ra các ký tự cần thiết để tạo thành các góc.

```
static void DrawBoard()
    {

        Console.SetCursorPosition(0, 0);
         Console.CursorVisible = false;
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        // Draw top border
        for (int i = 0; i < BOARD_WIDTH +1; i++)
        {
            Console.Write("-");
        }
        Console.WriteLine();
        Console.ResetColor();

        // Draw board contents
        for (int y = 0; y < BOARD_HEIGHT; y++)
        {
```

```
            Console.ForegroundColor = ConsoleColor.DarkCyan;
            Console.Write("||");
            Console.ResetColor();
            for (int x = 0; x < BOARD_WIDTH; x++)
            {
                if (board[x, y] == 0)
                {
                    Console.Write(" ");
                }
                else if (board[x, y] == -1)
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.Write("@");
                    Console.ResetColor();
                }
                else if (board[x, y] == 2)
                {
                    Console.ForegroundColor =
ConsoleColor.DarkGray;
                    Console.Write("X");
                    Console.ResetColor();
                }

                else
                {
                    Random random = new Random();
                    Console.ForegroundColor =
(ConsoleColor)random.Next(1, 12);
                    Console.Write(@"■");
                    Console.ResetColor();
                }
            }
            Console.ForegroundColor = ConsoleColor.DarkCyan;
            Console.Write("||");
            Console.ResetColor();
            Console.WriteLine();
        }
        Console.ForegroundColor = ConsoleColor.DarkCyan;
        // Draw bottom border
        for (int i = 0; i < BOARD_WIDTH + 1; i++)
        {
            Console.Write("-");
        }

        Console.SetCursorPosition(0, 0); //trên bên trái
        Console.Write("+");

        Console.SetCursorPosition(BOARD_WIDTH + 1, 0); //trên
phải
        Console.Write("--+");
```

```
        Console.SetCursorPosition(0, BOARD_HEIGHT + 1); //đặt con
trỏ góc dưới bên trái
        Console.Write("+");

        Console.SetCursorPosition(BOARD_WIDTH + 1, BOARD_HEIGHT
+1); //dưới bên phải
        Console.Write("--+"); Console.ResetColor();
        Console.WriteLine();
        Console.WriteLine();

        Console.WriteLine("Score: {0}", score);
        Console.WriteLine("Level: {0}", level);
    }
```



Hình – 6 Giao diện màn hình trò chơi chính

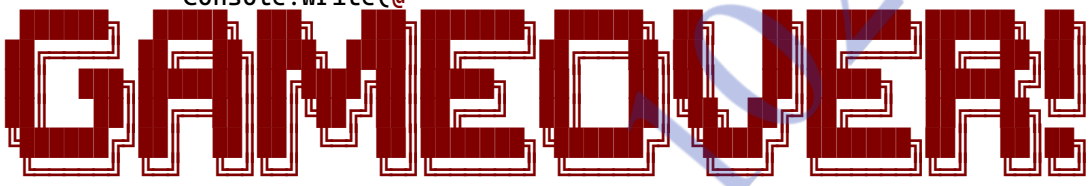
2.6. void Move(): Hàm này di chuyển rắn và xử lý khi rắn ăn thức ăn hoặc va chạm với vật cản hoặc tường.

- Khởi tạo biến dx và dy với giá trị ban đầu là 0.
- Dựa trên hướng di chuyển (direction), cập nhật giá trị của biến dx và dy.
- Lưu vị trí cuối cùng của thân rắn vào biến tail.
- Duyệt từ cuối mảng body, gán giá trị phần tử thứ i bằng phần tử thứ i-1. Tức là, tất cả các phần tử trong body di chuyển đến vị trí của phần tử liền trước nó.
- Gán vị trí đầu của rắn (head) bằng vị trí mới sau khi di chuyển, bằng cách cộng thêm giá trị dx và dy với tọa độ hiện tại của head.

- Nếu head di chuyển ra ngoài biên của board hoặc chạm vào tường, hoặc rắn ăn vào chính nó (board[head.X, head.Y] > 0), hoặc rắn chạm vào vật cản (board[head.X, head.Y] == 2), thì kết thúc trò chơi (GameOver = true) và thoát khỏi hàm.
- Nếu head di chuyển vào vị trí có thức ăn (-1), thì tăng điểm số (score) lên FOOD_SCORE, thêm một phần tử vào body, đặt vị trí của phần tử cuối cùng của body bằng tail, đặt lại vị trí của thức ăn (PlaceFood()), và đặt một vật cản mới (PlaceObstacle()).
- Nếu head di chuyển vào một vị trí khác, gán giá trị 0 cho phần tử cuối cùng của body (tail) trong board, và gán giá trị (body.Length + 1) cho vị trí của head trong board.
- Nếu điểm số (score) đạt hoặc vượt qua giới hạn LEVEL_SCORE, tăng cấp độ (level) lên 1, in ra màn hình một thông báo về việc tăng cấp độ và chờ trong 3 giây trước khi xóa màn hình và tiếp tục chơi với cấp độ mới và thời gian chờ (sleepTime) giảm đi 50 miligiây.
- Dừng chương trình trong một khoảng thời gian (sleepTime) trước khi tiếp tục di chuyển.

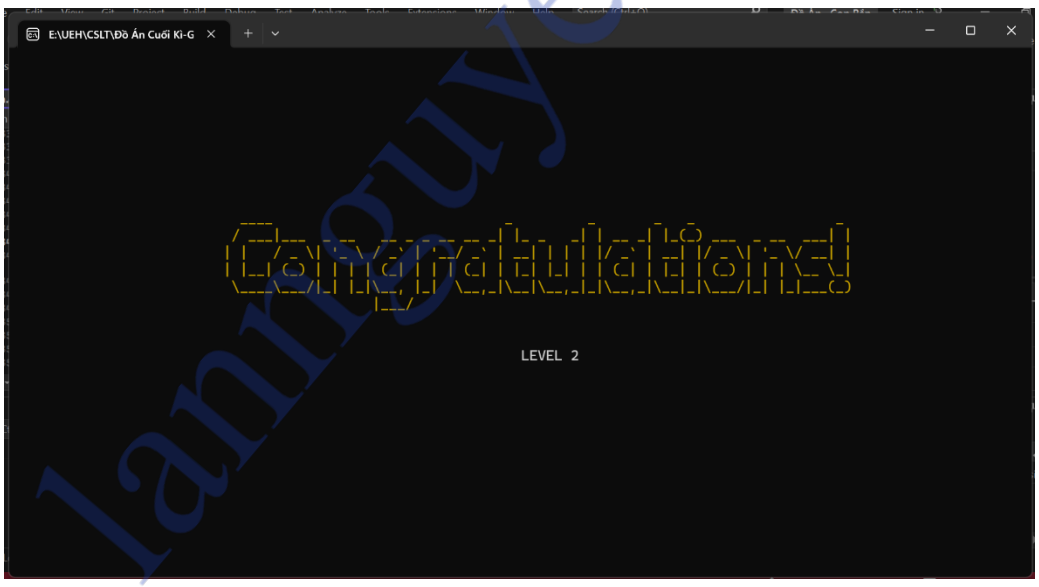
```
static void Move()
    {
        int dx = 0, dy = 0;

        switch (direction)
        {
            case Direction.Left:
                dx = -1;
                break;
            case Direction.Up:
                dy = -1;
                break;
            case Direction.Right:
                dx = 1;
                break;
            case Direction.Down:
                dy = 1;
                break;
        }

        Point tail = body[body.Length - 1];
```

```
        for (int i = body.Length - 1; i > 0; i--)
        {
            body[i] = body[i - 1];
        }

        body[0] = head;
        head = new Point(head.X + dx, head.Y + dy);

        if (head.X < 0 || head.X >= BOARD_WIDTH || head.Y < 0 ||
head.Y >= BOARD_HEIGHT || board[head.X, head.Y] > 0 ||
board[head.X, head.Y] == 2)
        {
            GameOver = true;
           Console.ForegroundColor = ConsoleColor.DarkMagenta;
          Console.Clear();
          Console.Write(@"
```

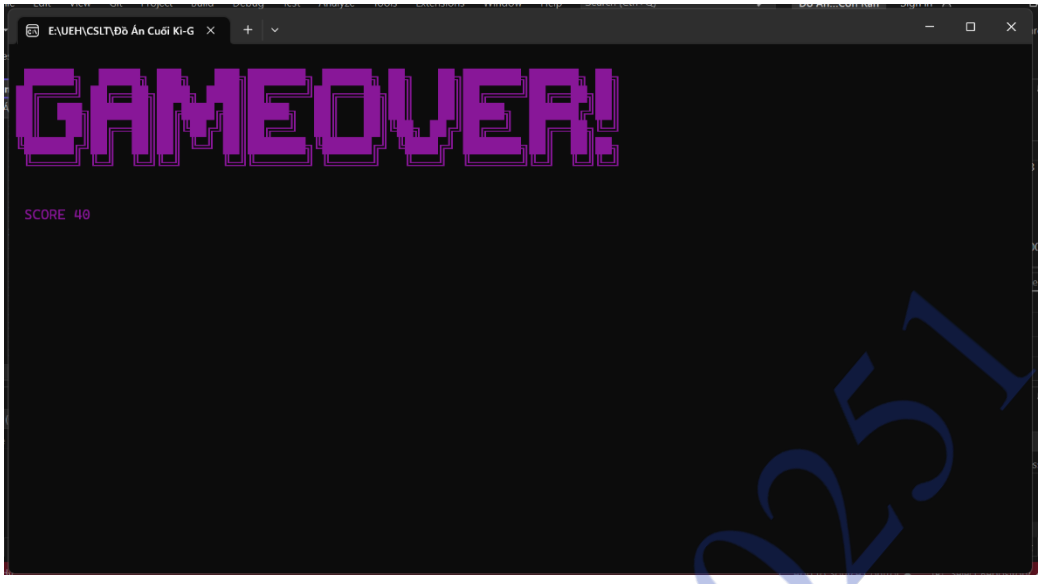



```
"+ "\n \n SCORE {0}", score);
          Thread.Sleep(3000);
          Console.Clear();
            return;
        }

        if (board[head.X, head.Y] == -1)
        {
            score += FOOD_SCORE;
            Array.Resize(ref body, body.Length + 1);
            body[body.Length - 1] = tail;
            PlaceFood();
            PlaceObstacle();
        }
        else
        {
            board[tail.X, tail.Y] = 0;

            board[head.X, head.Y] = body.Length + 1;

        }

        if (score >= level * LEVEL_SCORE)
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.DarkYellow;
            Console.SetCursorPosition(Console.WindowWidth/ 2,
Console.WindowHeight/2 - 7);

            Console.WriteLine(@"
                          ____                              _
_       _   _                 _
```

```
                          / ___|___  _ __   __ _ _ __ __ _| |_ _
_| | __ _| |_(_) ___  _ __  ___| |
                          | |   / _ \| '_ \ / _` | '__/ _` | __| |
| | |/ _` | __| |/ _ \| '_ \/ __| |
                          | |__| (_) | | | | (_| | | | (_| | |_|
|_| | | (_| | |_| | (_) | | | \__ \_|
                          \____\___/|_| |_|\__, |_|
\__,_|\__|\__,_|_|\__,_|\__|_|\___/|_| |_|___(_)
                                              |___/
");
            Console.ResetColor();
            Console.SetCursorPosition(Console.WindowWidth / 2 + 1
, Console.WindowHeight / 2 + 2);
            Console.WriteLine("LEVEL {0}", level + 1);

            Thread.Sleep(3000);
            Console.Clear();

            level++;
            sleepTime -= 50;
        }

        Thread.Sleep(sleepTime);
    }
```



Hình – 7 Giao diện khi người chơi qua mức mới



Hình – 8 Giao diện khi người chơi GameOver

2.7. void ReadInput(): Hàm này được sử dụng để đọc input từ bàn phím và hiển thị và di chuyển rắn trên màn hình.

- Trong vòng lặp while, chương trình đọc thông tin từ bàn phím bằng cách sử dụng hàm Console.ReadKey(true).
    - Nếu người dùng nhấn các phím di chuyển (trái, phải, lên, xuống), hướng di chuyển của rắn sẽ được cập nhật tương ứng.
    - Nếu phím Space được nhấn, biến Pause sẽ được chuyển đổi giữa true và false để tạm dừng hoặc tiếp tục trò chơi.

```
static void ReadInput()
    {
        while (!GameOver)
        {
            ConsoleKeyInfo keyInfo = Console.ReadKey(true);

            switch (keyInfo.Key)
            {
                case ConsoleKey.LeftArrow:
                    if (direction != Direction.Right)
                    {
                        direction = Direction.Left;
                    }
                    break;
                case ConsoleKey.UpArrow:
                    if (direction != Direction.Down)
                    {
                        direction = Direction.Up;
```

```
                }
                break;
            case ConsoleKey.RightArrow:
                if (direction != Direction.Left)
                {
                    direction = Direction.Right;
                }
                break;
            case ConsoleKey.DownArrow:
                if (direction != Direction.Up)
                {
                    direction = Direction.Down;
                }
                break;
            case ConsoleKey.Spacebar:
                Pause = !Pause;
                break;
        }
    }
}
```

2.8. void GameLoop(): Hàm này là vòng lặp chính của trò chơi. Nó kiểm tra trạng thái của trò chơi và cập nhật các phần tử trong trò chơi.

Trong khi trò chơi chưa kết thúc (GameOver = false), hàm sẽ lặp lại những bước sau:

- Sử dụng lockObject để khóa một phần của chương trình để tránh các tác vụ truy cập đồng thời vào tài nguyên chia sẻ.
- Kiểm tra xem trạng thái trò chơi có bị tạm dừng hay không (Pause = true).
- Nếu trò chơi không bị tạm dừng, hàm sẽ gọi hàm Move() để di chuyển rắn trên màn hình và hàm DrawBoard() để hiển thị trạng thái mới của màn hình.
- Cuối cùng, vòng lặp sẽ lặp lại và kiểm tra lại điều kiện trò chơi đã kết thúc hay chưa.

```
static void GameLoop()
{
    while (!GameOver)
    {
        lock (lockObject)
        {
            if (!Pause)
            {
                Move();
```

```
                    DrawBoard();
                }
            }
        }
    }
```

2.9. void Start(): Hàm này khởi tạo các giá trị ban đầu cho trò chơi, bắt đầu vòng lặp chính của trò chơi và tiếp tục cho đến khi trò chơi kết thúc.

- Tạo một luồng mới để chạy hàm ReadInput(), sau đó gọi hàm GameLoop() để bắt đầu vòng lặp chính của game.
- Trong khi luồng chính của game (vòng lặp GameLoop()) đang chạy, nó sẽ chờ đợi đến khi luồng inputThread kết thúc bằng cách gọi phương thức Join(). Điều này đảm bảo rằng chúng ta không bị mất dữ liệu khi chạy và đọc đầu vào từ bàn phím trong khi trò chơi đang chạy.
- Khi người dùng muốn thoát game, biến GameOver sẽ được đặt thành true trong luồng ReadInput(), sau đó vòng lặp chính sẽ kết thúc và game sẽ dừng lại.

```
static void Start()
{
    Thread inputThread = new Thread(ReadInput);
    inputThread.Start();

    GameLoop();

    inputThread.Join();
}
```

2.10. Âm thanh trò chơi

Âm thanh di chuyển

```
Console.Beep(800, 100);

        Thread.Sleep(50);
```

Âm thanh ăn thức ăn

```
Console.Beep(1000, 200); // phát ra âm thanh
            Thread.Sleep(500);
```

Âm thanh GameOver

- Trong vòng lặp, giá trị biến i sẽ được tăng dần từ 0 đến 4.

- Mỗi lần vòng lặp được thực thi, hàm Console.Beep() sẽ được gọi để phát ra một tín hiệu âm thanh. Tham số đầu tiên của hàm (300 + i * 100) đại diện cho tần số của âm thanh được phát ra, tăng dần lên 100 đơn vị mỗi lần vòng lặp được thực thi. Tham số thứ hai (200) đại diện cho thời gian phát ra của tín hiệu âm thanh (theo đơn vị mili giây).
- Sau mỗi lần phát ra tín hiệu âm thanh, vòng lặp sẽ sử dụng phương thức Thread.Sleep() để tạm dừng chương trình trong 200 mili giây trước khi tiếp tục thực thi lần lặp tiếp theo.

```
for (int i = 0; i < 5; i++)
{
      Console.Beep(300 + i * 100, 200);
      Thread.Sleep(200);
}
```

Âm thanh Chúc mừng qua màn chơi mới

- Khởi tạo mảng freqs chứa các tần số của các nốt nhạc.
- Khởi tạo biến time để lưu thời lượng của mỗi nốt nhạc.
- Sử dụng vòng lặp đầu tiên để phát ra chuỗi âm thanh bằng cách lặp lại các nốt nhạc trong mảng freqs sử dụng hàm Console.Beep().
- Sau khi phát xong chuỗi âm thanh đầu tiên, sử dụng hàm Thread.Sleep() để tạm dừng chương trình trong một khoảng thời gian ngắn.
- Sử dụng vòng lặp thứ hai để phát lại chuỗi âm thanh đó một lần nữa và có một khoảng thời gian nghỉ ngắn giữa các nốt nhạc.

```
int[] freqs = { 262, 392, 440, 494, 262, 330, 262, 330, 392, 440,
494, 523, 659, 659, 523, 494, 440, 392 };
int time = 100;
for (int i = 0; i < freqs.Length; i++)
{
     Console.Beep(freqs[i], time);
}
Thread.Sleep(500);
for (int i = 0; i < freqs.Length; i++)
{
     Console.Beep(freqs[i], time);
}
```

KẾT LUẬN

Trong quá trình học tập và áp dụng kiến thức từ trường học và tự học tập thêm trong quá trình làm Báo cáo, em đã viết được một trò chơi rắn săn mồi đơn giản với mục đích giải trí lành mạnh. Em hy vọng rằng các kỹ năng và kinh nghiệm thu được từ bài Báo cáo này sẽ giúp em phát triển được các trò chơi khác, chất lượng và hấp dẫn hơn để phục vụ cho nhu cầu giải trí trong cuộc sống hàng ngày của chúng ta.

Trong quá trình thực hiện dự án này, em đã nhận được sự hỗ trợ nhiệt tình của Thầy Lê Hữu Thanh Tùng để hoàn thành bài báo cáo này một cách tốt nhất có thể. Tuy nhiên, do thời gian thực hiện bài báo cáo có hạn và kinh nghiệm thực tiễn của em còn hạn chế, do đó không thể tránh khỏi những sai sót. Em rất mong nhận được sự góp ý từ Thầy để hoàn thiện hơn dự án này cũng như cải thiện kỹ năng và kinh nghiệm của chúng tôi trong tương lai.

PHẦN PHỤ LỤC

4.1. Mong muốn chưa đạt được

- Chưa có chức năng đổi giao diện khung trò chơi cho các mức chơi. Với sự đa dạng và phong phú của các mức chơi, việc có thể đổi giao diện để phù hợp với từng mức sẽ giúp cho trải nghiệm của người chơi được nâng cao.
- Chưa đáp ứng được mong muốn của người chơi về một âm thanh thú vị, cuốn hút. Việc cải thiện âm thanh sẽ giúp tạo nên một không khí chơi game thú vị hơn, đồng thời giúp người chơi cảm thấy thích thú hơn khi tham gia trò chơi.
- Chưa có chức năng đa người chơi cho phép hai người có thể thi đua với nhau. Việc bổ sung chức năng đa người chơi sẽ giúp tạo nên một sân chơi công bằng và thú vị hơn..
- Chưa có nhiều loại thức ăn với ảnh hướng khác nhau (đi nhanh, bị tạm dừng tự động 2 giây, …) để tăng thêm sự hấp dẫn và thử thách cho người chơi.

4.2. Tài liệu tham khảo

ii. **Tài liệu tham khảo chính:** Giáo trình môn học Cơ sở lập trình của Khoa Công nghệ thông tin kinh doanh Trường Đại học Kinh tế thành phố Hồ Chí Minh.

iii. **Các tài liệu tham khảo khác:** https://www.codecademy.com/