



# SNAKE GAME



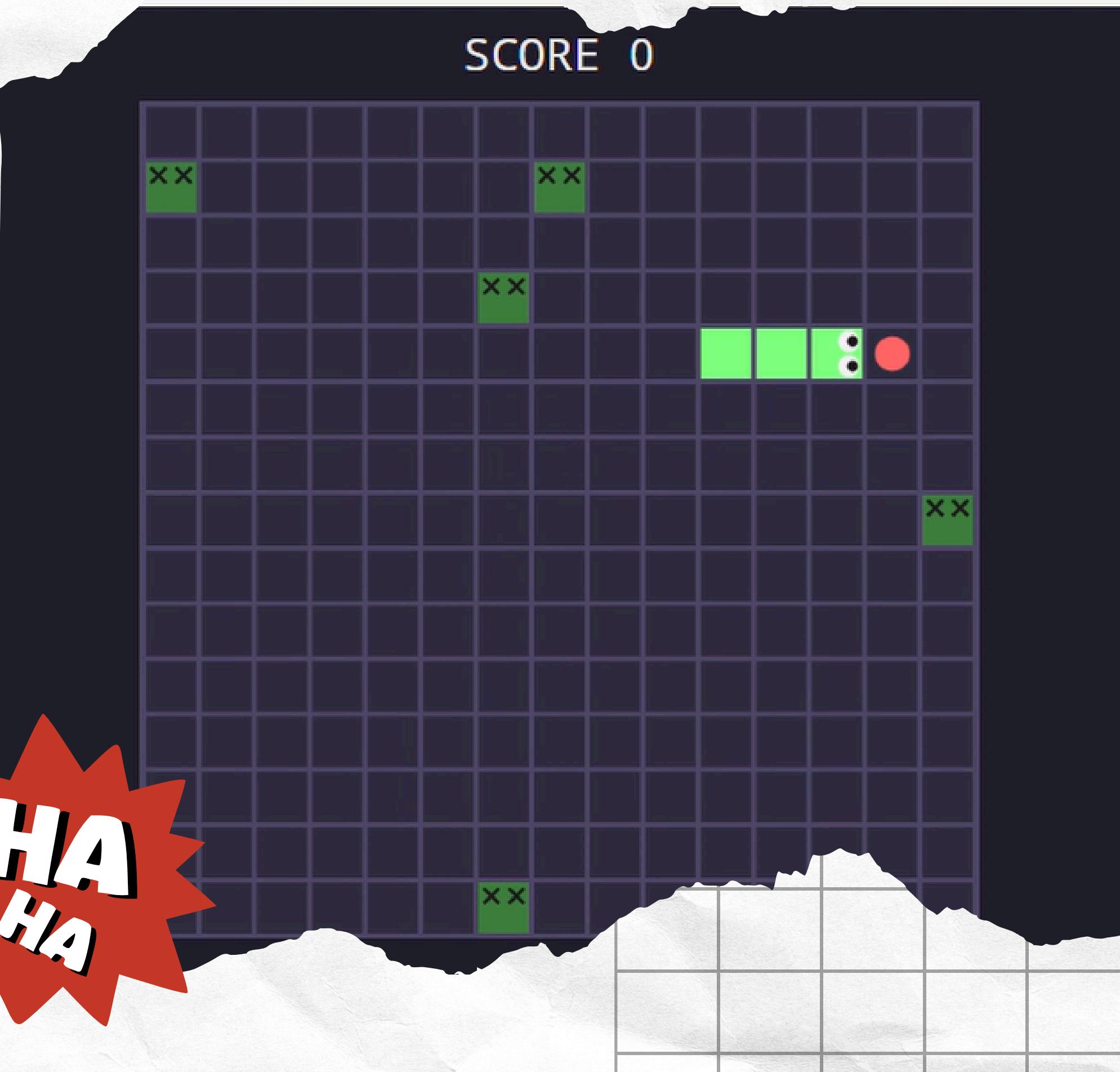
66030192 สุภัตรา พันธุ์กุหลาบ  
66030270 ภูมิ อัครธรรม



# SNAKE GAME

Snake Game เป็นเกมแนวอาร์เคดที่มีรูปแบบการเล่นง่าย ๆ คือ ผู้เล่นจะควบคุมตัวงูให้เคลื่อนที่ไปบนกระดานเพื่อกินอาหาร ซึ่งเมื่อกินอาหาร ตัวงูจะยาวขึ้นและคะแนนจะเพิ่มขึ้น ความท้าทายคือ ผู้เล่นต้องหลีกเลี่ยงการชนตัวเองและขอบสนาม ซึ่งถ้าชนเมื่อไหร่ เกมจะจบทันที

ในเวอร์ชันนี้ เราได้เพิ่มฟังก์ชันพิเศษที่เรียกว่า 'Dynamic Obstacles' หรือ สิ่งกีดขวางที่สามารถเปลี่ยนตำแหน่งได้เอง ซึ่งจะทำให้ผู้เล่นต้องพยายามหลบสิ่งกีดขวางที่อาจเปลี่ยนที่ได้ทุก ๆ 10 วินาที



Popular repositories

- weatherApp** Public  
CSS
- Snake-game-in-C-sharp** Public  
A snake game running on desktop created in C# using .NET 8.0, WPF and XAML  
C# 2

6 contributions in the last year

Learn how we count contributions Less More

Contribution activity

March 2025

traversiolo has no activity yet for this period.

Show more activity

Watch 1 Fork 2 Star 0

**Snake-game-in-C-sharp** Public

master 1 Branch 0 Tags

Go to file Add file Code

traversiolo Aggiungere i file di progetto. 43a39ea · 8 months ago 2 Commits

Snake	Aggiungere i file di progetto.	8 months ago
.gitattributes	Aggiungi .gitattributes e .gitignore.	8 months ago
.gitignore	Aggiungi .gitattributes e .gitignore.	8 months ago
Snake.sln	Aggiungere i file di progetto.	8 months ago

About

A snake game running on desktop created in C# using .NET 8.0, WPF and XAML

Activity

0 stars 1 watching 2 forks Report repository

Releases

No releases published

Packages

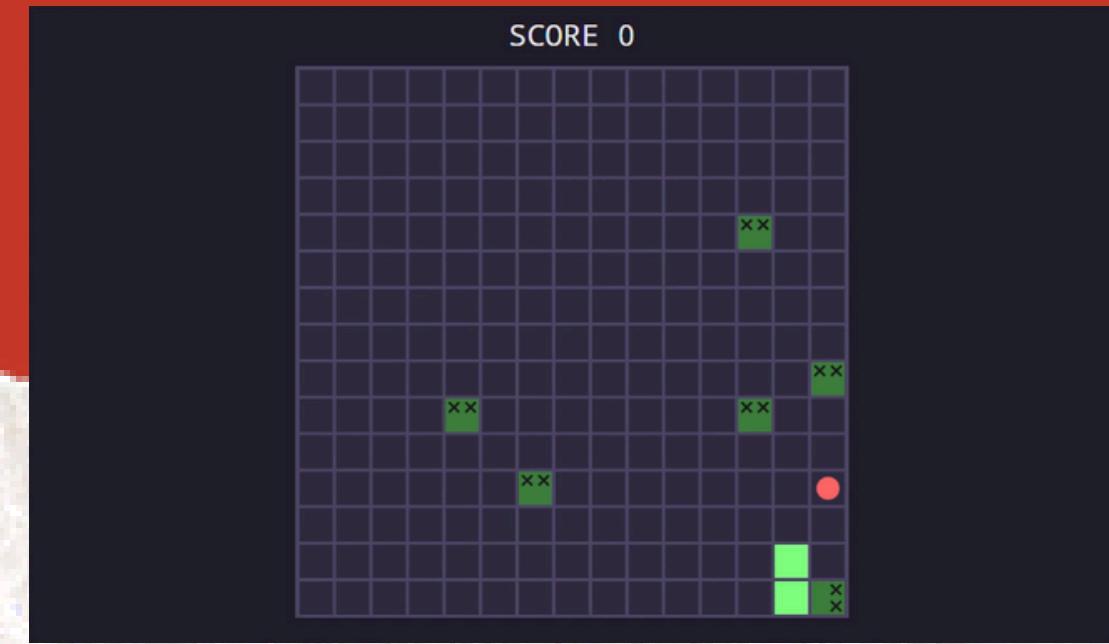
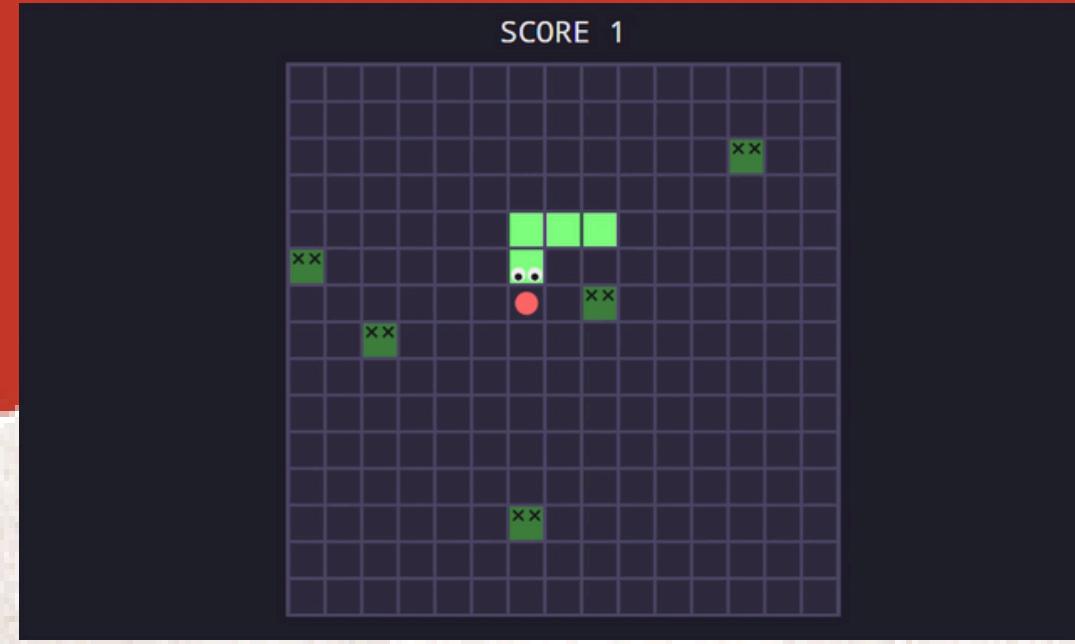
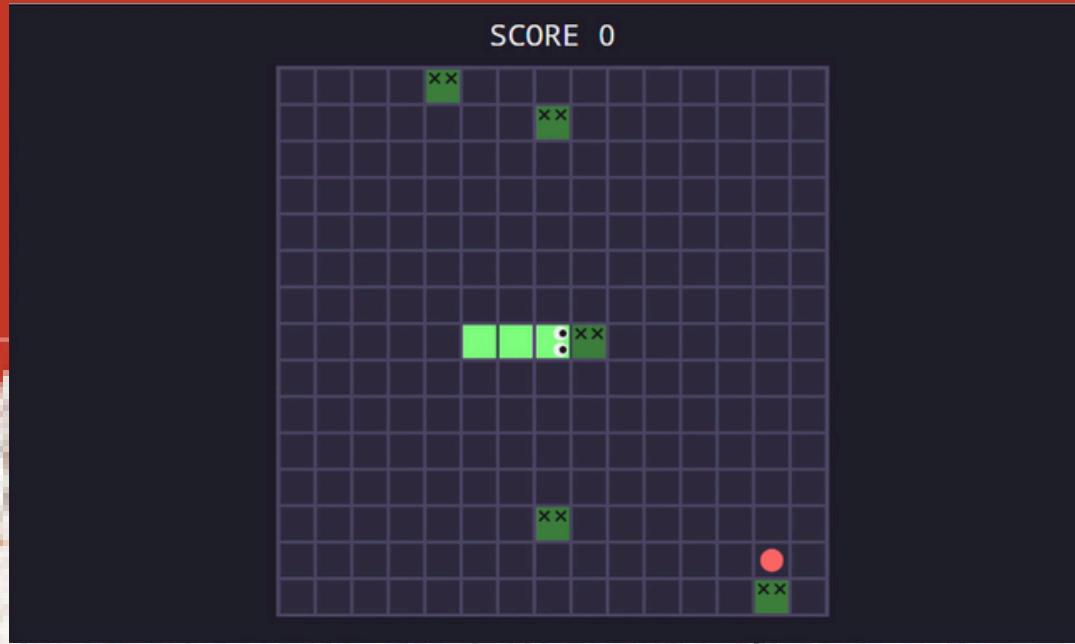
No packages published

Languages

C# 100.0%



# หลักการเล่น



## \* ควบคุม

- ผู้เล่นสามารถบังคับงูให้เคลื่อนที่ไปใน 4 ทิศทาง ได้แก่ ขึ้น (W หรือ  $\uparrow$ ), ลง (S หรือ  $\downarrow$ ), ซ้าย (A หรือ  $\leftarrow$ ) และ ขวา (D หรือ  $\rightarrow$ )
- งูจะเคลื่อนที่ไปเรื่อย ๆ ตามทิศทางที่กำหนด และ ไม่สามารถเดินถอยหลังได้ เช่น ถ้ากำลังไปทางขวา จะไม่สามารถเลี้ยวกลับซ้ายได้ทันที

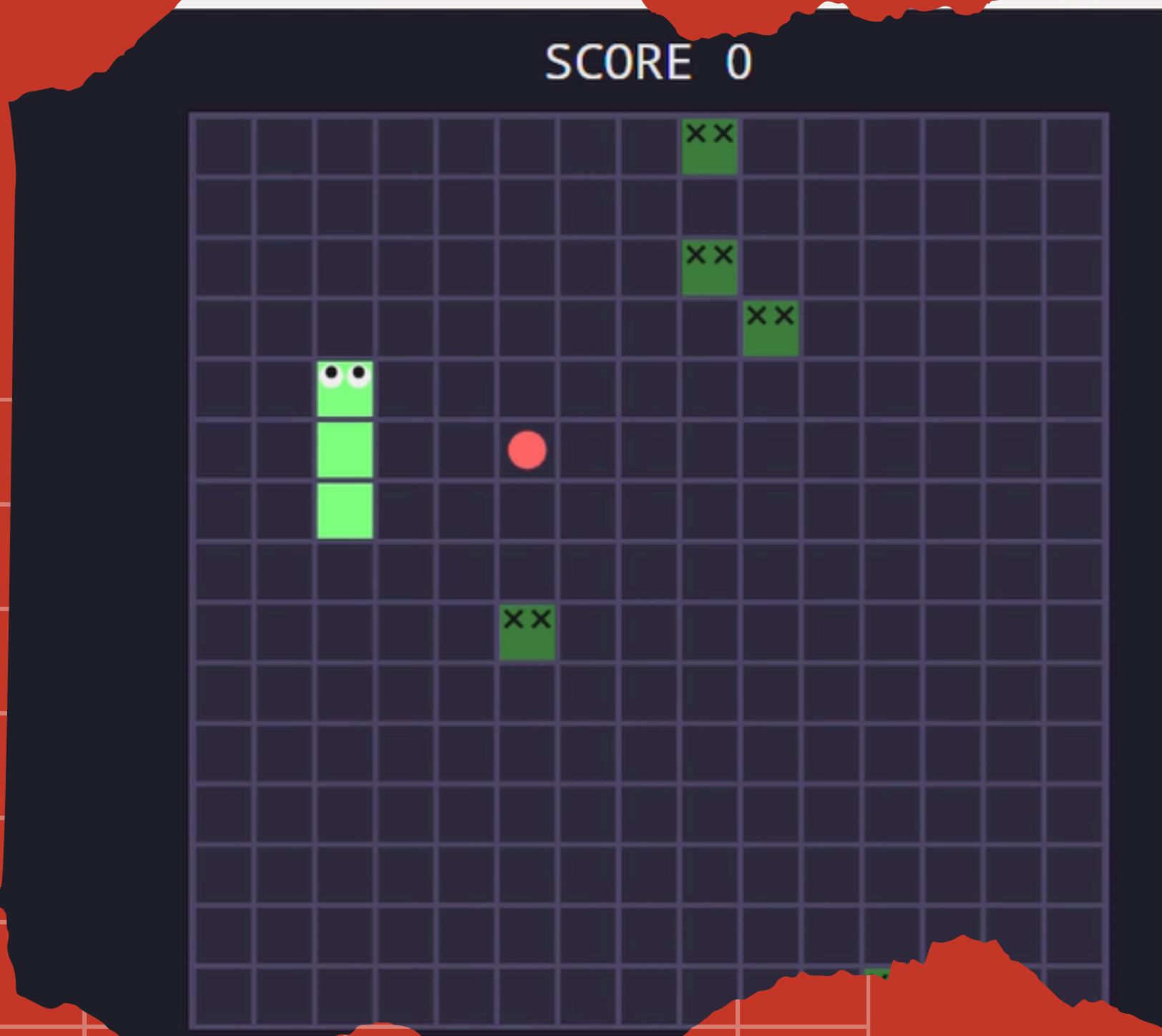
## \* เป้าหมายของเกม

- ผู้เล่นต้อง บังคับงูให้กินอาหาร ที่ปรากฏขึ้นบนกระดาน โดยเมื่อกินอาหารได้ งูจะตัวยาวขึ้น และ คะแนนจะเพิ่มขึ้น
- เป้าหมายหลักคือ ทำคะแนนให้ได้มากที่สุด และ อยู่รอดให้นานที่สุด

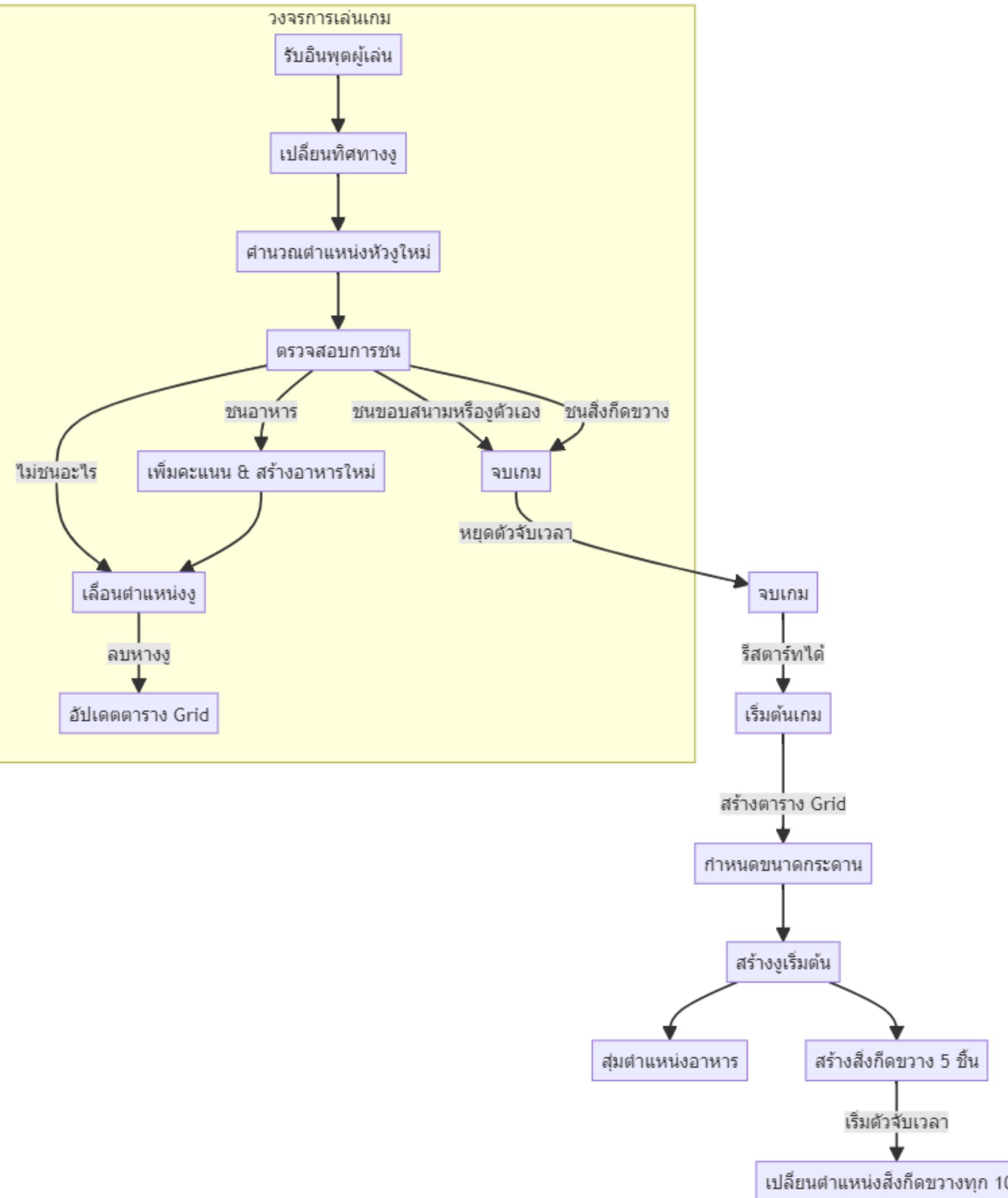
## \* สิ่งที่ต้องระวัง

- อย่าชนกำแพง : ถ้างูเคลื่อนที่ออกนอกขอบสนาม เกมจะจบทันที
- อย่าชนตัวเอง : หากงูเคลื่อนที่ไปชนตัวเอง เกมก็จะจบทันทีเช่นกัน
- ระวังสิ่งกีดขวาง : เกมของเรามีพังก์ชั่น 'Dynamic Obstacles' หรือ สิ่งกีดขวางที่เปลี่ยนตำแหน่งได้ทุก ๆ 10 วินาที ผู้เล่นต้องปรับตัวให้ทัน

# การเพิ่มฟังก์ชัน



ในตอนแรกเกมเริ่มต้นด้วยการมีเพียงงู (Snake) และอาหาร (Food) เท่านั้น โดยที่งูจะเคลื่อนที่และเติบโตเมื่อกินอาหาร โดยไม่มีอุปสรรคใดๆ ที่จะขัดขวางการเคลื่อนไหวของงู ซึ่งเราได้เพิ่มฟังก์ชัน "สิ่งกีดขวาง" (Obstacles) ลงในเกม ซึ่งทำหน้าที่เป็นอุปสรรคที่งูไม่สามารถเคลื่อนที่ผ่านไปได้เหมือนกับพื้นที่ว่างหรืออาหาร สิ่งกีดขวางเหล่านี้จะถูกสร้างขึ้นในตำแหน่งที่ต้องการ และตำแหน่งของมันจะเปลี่ยนไปทุกๆ 10 วินาที โดยใช้ Timer ในการควบคุมการอัปเดต การตรวจสอบสิ่งกีดขวางจะเกิดขึ้นทุกครั้งที่งูเคลื่อนที่ ถ้างูชนกับสิ่งกีดขวาง เกมจะจบลงทันทีและจะไม่สามารถเคลื่อนที่ผ่านสิ่งกีดขวางได้ ซึ่งเป็นความท้าทายที่เพิ่มเข้ามาในเกมที่ทำให้มันยากขึ้นเมื่อเทียบกับเวอร์ชันก่อนหน้า



# ผังการทำงาน

## \* การเริ่มต้นเกม

- การสร้างตาราง Grid : กำหนดขนาดของกระดาน (rows x columns) เพื่อใช้ในการวางแผนตำแหน่งของงู, อาหาร, และสิ่งกีดขวาง
- สร้างงูเริ่มต้น : งูเริ่มต้นมีความยาว 3 ช่อง และอยู่ในตำแหน่งกลางของกระดาน
- สุมตำแหน่งอาหาร : สุมตำแหน่งที่ว่างบนกระดานเพื่อวางอาหาร
- สร้างสิ่งกีดขวาง 5 ชิ้น : สิ่งกีดขวางถูกสุมตำแหน่งในกริดและจะมีการเปลี่ยนแปลงตำแหน่งทุกๆ 10 วินาที

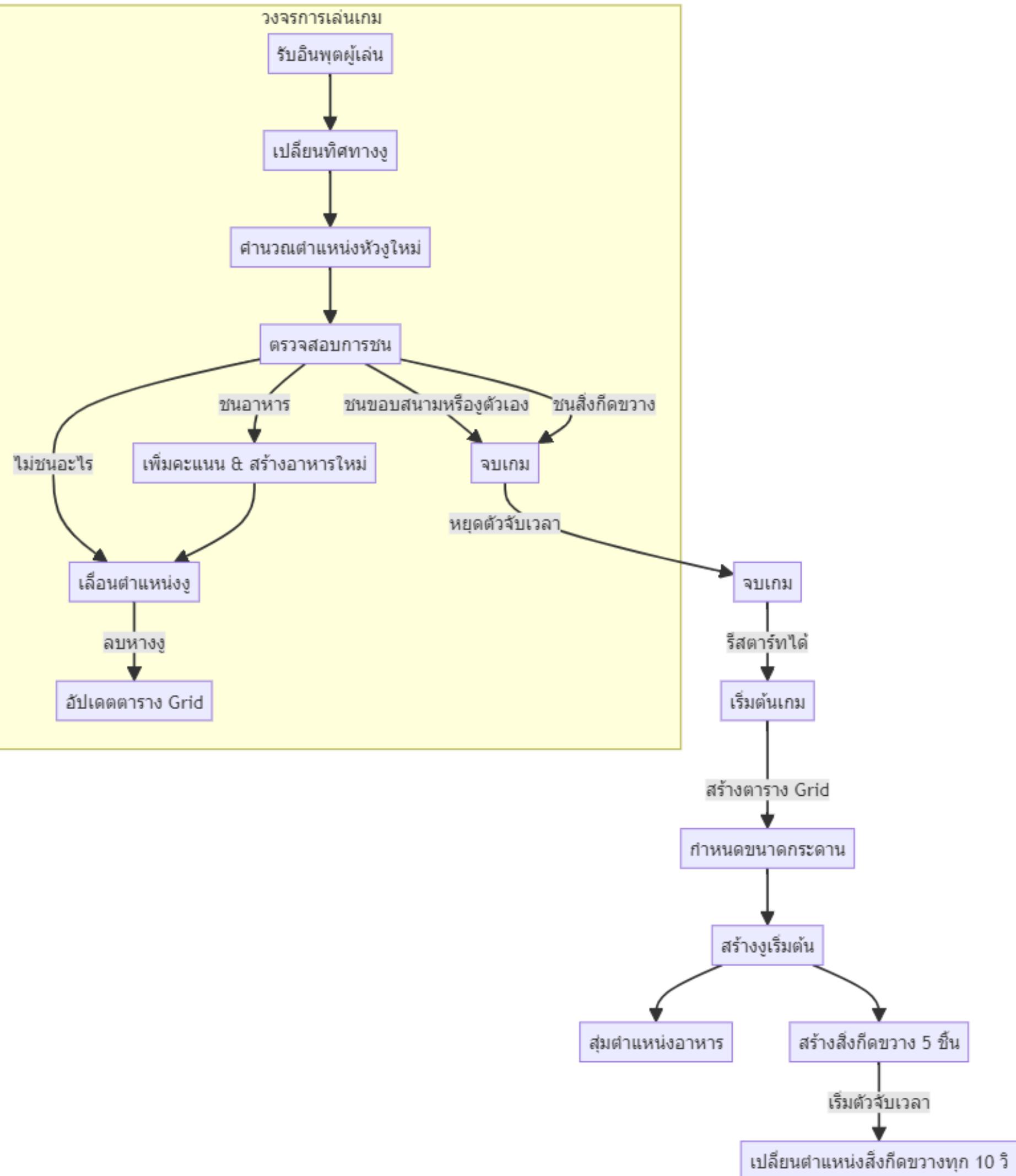
## \* การเปลี่ยนแปลงตำแหน่งสิ่งกีดขวาง (ใช้ Timer)

- ตั้งค่าตัวจับเวลา : ใช้ System.Threading.Timer เพื่อให้สิ่งกีดขวางเปลี่ยนตำแหน่งทุกๆ 10 วินาที

# ผังการทำงาน

## \* วงจรการเล่นเกม

- **รับอินพุตจากผู้เล่น** : รับคำสั่งจากผู้เล่นเพื่อเปลี่ยนทิศทางของงู
- **คำนวณตำแหน่งหัวงูใหม่** : เลื่อนตำแหน่งหัวงูไปตามทิศทางที่ผู้เล่นกำหนด
- **ตรวจสอบการชน** : งูชนขอบสนาม, งูชนตัวเอง, งูชนสิ่งกีดขวาง, งูชนอาหาร
  - หากชนอาหาร : เพิ่มคะแนนและสร้างอาหารใหม่
  - หากไม่ชนอะไร : เลื่อนตำแหน่งงูและลบทาง



## \* จบเกม

- **ถ้าเกิดการชนขอบสนาม, งูตัวเอง หรือสิ่งกีดขวาง** : เกมจะจบลงและหยุดการทำงานของตัวจับเวลา
- **รีสตาร์ทเกม** : ผู้เล่นสามารถเริ่มใหม่ได้จากจุดเริ่มต้น

เปลี่ยนตำแหน่งสิ่งกีดขวางทุก 10 วิ

# Code

## ส่วนที่หนึ่ง : การกำหนดค่าเริ่มต้น

```
public GameState(int rows, int cols)
{
    Rows = rows;
    Cols = cols;
    Grid = new GridValue[rows, cols];
    Dir = Direction.Right;

    AddSnake();
    AddFood();
    GenerateObstacles(5);
    obstacleTimer = new Timer(_ =>
        RegenerateObstacles(), null, 10000, 10000);
}
```

## ส่วนที่สอง : การสร้างงู

```
private void AddSnake()
{
    int r = Rows / 2;
    for (int c = 0; c < 3; c++)
    {
        Grid[r, c] = GridValue.Snake;
        snakePositions.AddFirst(new Position(r, c));
    }
}
```

## ส่วนที่สาม: การย้ายงู

```
private void Move()
{
    if (dirChanges.Count > 0)
    {
        Dir = dirChanges.First.Value;
        dirChanges.RemoveFirst();
    }
}
```

```
Position newHeadPos =
    HeadPosition().Translate(Dir);
    GridValue hit = WillHit(newHeadPos);
}
```

# Code

## ส่วนที่สี่ : การจัดการกับอาหาร

```
private void AddFood()
{
    List<Position> empty = new List<Position>
(EmptyPosition());
    if (empty.Count == 0)
    {
        return;
    }

    Position pos =
empty[random.Next(empty.Count)];
    Grid[pos.Row, pos.Col] = GridValue.Food;
}
```

## ส่วนที่ห้า : การสร้างสิ่งกีดขวาง

```
private void GenerateObstacles(int count)
{
    Obstacles.Clear();
    for (int i = 0; i < count; i++)
    {
        Position pos;
        do
        {
            pos = new Position(random.Next(Rows),
random.Next(Cols));
        } while (Grid[pos.Row, pos.Col] != GridValue.Empty);

        Obstacle newObstacle = new Obstacle(pos);
        Obstacles.Add(newObstacle);
        Grid[pos.Row, pos.Col] = GridValue.Obstacle;
    }
}
```

# Code

## ส่วนที่หก : การตรวจสอบการชน

```
private GridView WillHit(Position  
newHeadPos)  
{  
    if (OutsideGrid(newHeadPos))  
    {  
        return GridView.Outside;  
    }  
  
    if (newHeadPos == TailPosition())  
    {  
        return GridView.Empty;  
    }  
  
    return Grid[newHeadPos.Row,  
    newHeadPos.Col];  
}
```

## ส่วนที่เจ็ด : การจบเกม

```
f (hit == GridView.Outside || hit == GridView.Snake ||  
hit == GridView.Obstacle)  
{  
    GameOver = true;  
    obstacleTimer.Dispose();  
}
```

# หลักการ OOP

## \* Encapsulation (การห่อหุ้มข้อมูล)

- **GameState** : ข้อมูลของเกม เช่น ขนาดกริด (Rows, Cols), ทิศทางของงู (Dir), คะแนน (Score), และสถานะของเกม (GameOver) ถูกห่อหุ้มในคลาส GameState เพื่อให้สามารถจัดการได้อย่างเป็นระเบียบ โดยไม่ให้ข้อมูลเหล่านี้ถูกเปลี่ยนแปลงโดยตรงจากภายนอก
- **Obstacle** : ข้อมูลตำแหน่งของอุปสรรคถูกเก็บไว้ในคลาส Obstacle โดยที่ Position จะถูกเก็บไว้เป็น private set ซึ่งป้องกันไม่ให้ตำแหน่งของอุปสรรคถูกแก้ไขจากภายนอกโดยตรง
- **GridValue** : ใช้ enumeration (enum) เพื่อแสดงสถานะต่างๆ ในเกม (Empty, Food, Snake, Outside, Obstacle) โดยการเก็บสถานะนี้จะช่วยให้การจัดการกริดและการตรวจสอบสถานะต่างๆ ของเกมมีความชัดเจนและปลอดภัย

## \* Abstraction (การซ่อนรายละเอียด)

- พังก์ชันต่างๆ ใน GameState เช่น Move(), ChangeDirection(), WillHit() ถูกซ่อนรายละเอียดการทำงานภายในไว้ ทำให้ผู้ใช้ไม่ต้องรู้ถึงขั้นตอนการคำนวณต่างๆ แต่สามารถใช้พังก์ชันเพื่อควบคุมการเล่นเกมได้อย่างง่ายดาย
- **Obstacle** : คลาส Obstacle ซ่อนรายละเอียดการจัดการกับตำแหน่งของอุปสรรค โดยไม่ต้องให้ผู้ใช้งานทราบว่า การจัดการตำแหน่งและลักษณะของอุปสรรค

## \* Inheritance (การสืบทอด)

- ในโปรเจกต์นี้ไม่ได้ใช้ Inheritance โดยตรงในคลาสที่เกี่ยวข้อง แต่การใช้คลาส GridValue และ Direction ที่เป็น enumeration ช่วยให้โค้ดอ่านง่ายและสามารถขยายได้ง่ายในอนาคต

# หลักการ OOP

## \* Polymorphism (การแปรปรวน)

- **CellValue** ใช้เพื่อแสดงผลลัพธ์ต่างๆ ที่อาจเกิดขึ้นระหว่างการเคลื่อนที่ของงู เช่น ถ้างูชนขอบสนาม หรือชนอุปสรรค จะได้ค่ากลับเป็น Outside หรือ Obstacle
- ฟังก์ชัน **Move()** ตรวจสอบการชนด้วย **WillHit()** และทำการแปรปรวนพฤติกรรมตามผลลัพธ์ที่ได้รับจากการชน

## \* Composition (การประกอบส่วนประกอบ)

- **GameState** ประกอบไปด้วยส่วนประกอบต่างๆ เช่น Grid, Snake, Food, Obstacle, และ Timer ซึ่งทำงานร่วมกันเพื่อสร้างการเล่นเกมที่สมบูรณ์
- **Images** : คลาส **Images** ประกอบด้วยการเก็บภาพต่างๆ ขององค์ประกอบในเกม (งู, อาหาร, สิ่งกีดขวาง) โดยแยกการจัดการรูปภาพออกเป็นส่วนๆ และใช้ฟังก์ชัน **LoadImage()** เพื่อโหลดภาพจากไฟล์



Thak  
you

