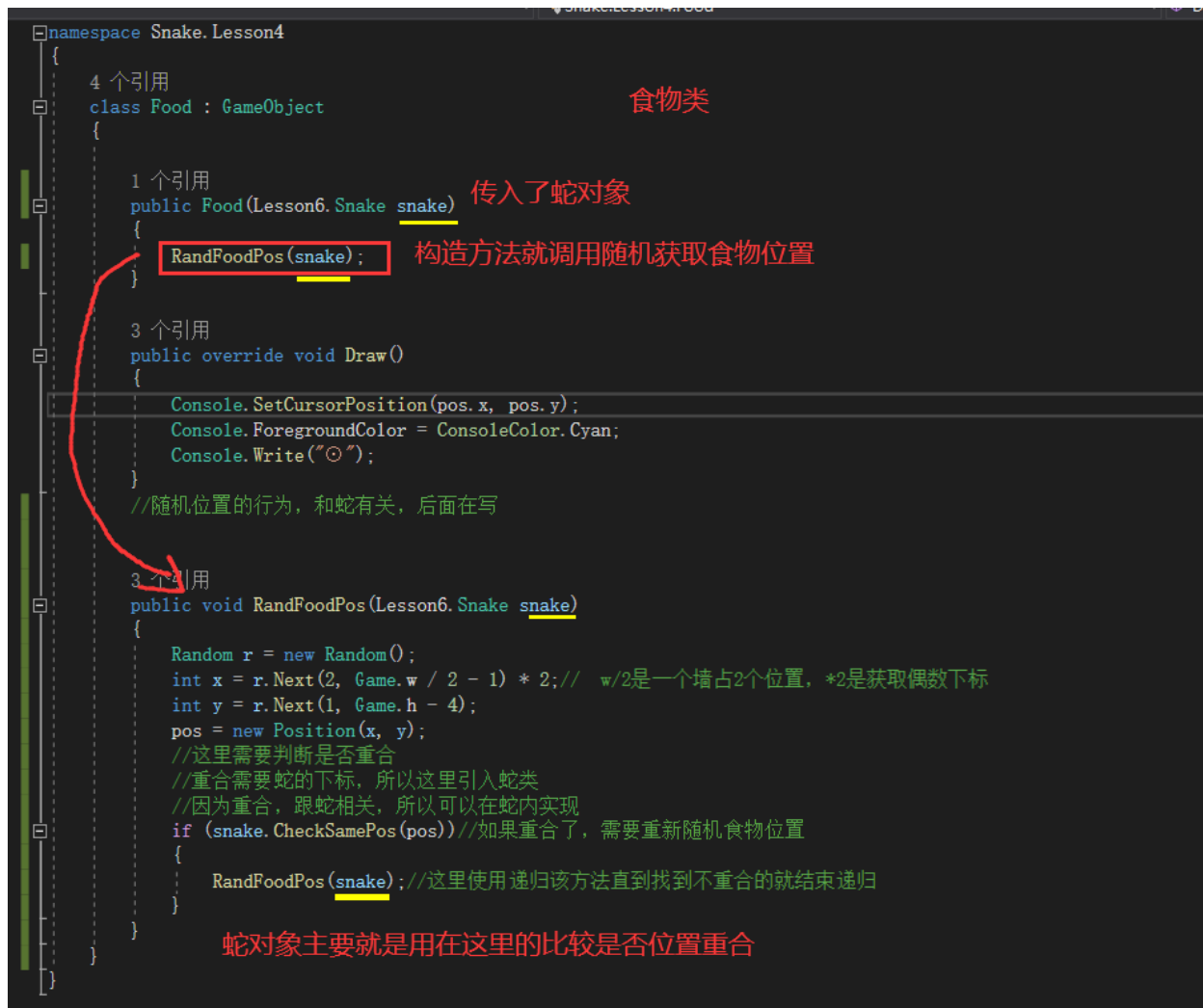


任务65：吃食物 知识点

吃食物，首先要考虑到食物的位置，食物是随机的，但是不能和蛇身体和墙体重合，食物被吃了，会再次随机出现一个位置。

然后就是考虑方法的实现位置，首先吃食物，那么是蛇的行为，就可以写在蛇类中，检查是否吃到了食物，也是可以写在蛇类方法中，食物方法需要提供一个随机生成食物的地方：



```
namespace Snake.Lesson4
{
    4 个引用
    class Food : GameObject
    {
        1 个引用
        public Food(Lesson6.Snake snake) 传入了蛇对象
        {
            RandFoodPos(snake); 构造方法就调用随机获取食物位置
        }

        3 个引用
        public override void Draw()
        {
            Console.SetCursorPosition(pos.x, pos.y);
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.Write("○");
        }

        //随机位置的行为，和蛇有关，后面在写

        3 个引用
        public void RandFoodPos(Lesson6.Snake snake)
        {
            Random r = new Random();
            int x = r.Next(2, Game.w / 2 - 1) * 2; // w/2是一个墙占2个位置，*2是获取偶数下标
            int y = r.Next(1, Game.h - 4);
            pos = new Position(x, y);
            //这里需要判断是否重合
            //重合需要蛇的下标，所以这里引入蛇类
            //因为重合，跟蛇相关，所以可以在蛇内实现
            if (snake.CheckSamePos(pos)) //如果重合了，需要重新随机食物位置
            {
                RandFoodPos(snake); //这里使用递归该方法直到找到不重合的就结束递归
            }
        }
    }
}
```

食物类

蛇对象主要就是用在这里的比较是否位置重合

食物是游戏场景的对象，所以需要在游戏场景初始化：

2 个引用

```
class GameScene : ISceneUpdate
{
    private Map map;
    Lesson6.Snake snake;
    private Food food;
    public int updateTime = 0;
```

1 个引用

```
public GameScene()
```

```
{
    map = new Map();
    snake = new Lesson6.Snake(40, 10);
```

初始化游戏场景，就跟着把地图、食物、和蛇进行初始化

```
    food = new Food(snake);
```

传入了才初始化的蛇对象

2 个引用

```
public void upData()
```

```
{
    if (updateTime == 4000)
```

```
    {
        map.Draw();
```

```
        food.Draw();
```

画食物

```
        snake.Move();
```

```
        snake.Draw();
```

```
        if (snake.CheckEnd(map))//这里刚好有地图，就直接传入
```

```
        {
            Game.changeScene(E_SceneType.End); //切换到结束场景
        }
```

```
        snake.CheckEatFood(food);
```

检查是否是否被吃

```
        updateTime = 0;
```

```
    }
    updateTime++;
```

```
    //new 判断有没有键盘输入，如果有才会进入if
```

```
    if (Console.KeyAvailable)//这里会导致循环变慢，所以上面的计数次数要减少
```

```
    {
        //检测输入输出，不能在间隔帧里面去处理，应该每次都检测，这样才准确
```

```
        switch (Console.ReadKey(true).Key)
```

```
        {
            case ConsoleKey.W:
                snake.changeDir(E_MoveDir.Up);
                break;
```

```
            case ConsoleKey.S:
                snake.changeDir(E_MoveDir.Down);
                break;
```

```
            case ConsoleKey.A:
                snake.changeDir(E_MoveDir.Left);
                break;
```

```
            case ConsoleKey.D:
                snake.changeDir(E_MoveDir.Right);
                break;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
Snake.cs  Food.cs  Wall.cs  Game.cs  EndScene.cs  GameScene.cs  ISceneUpdate.cs  GameC
Snake
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159

}
#endregion

//是否和食物重合
1 个引用
public bool CheckSamePos(Position pos)
{
    for (int i = 0; i < nowNum; i++)
    {
        if(bodys[i].pos == pos)
        {
            return true;
        }
    }
    return false;
}

//是否吃到了食物
1 个引用
public void CheckEatFood(Food food)
{
    for (int i = 0; i < nowNum; i++)
    {
        if(bodys[0].pos == food.pos)
        {
            food.RandFoodPos(this); //吃到了食物，就随机下一个食物出现的位置
        }
    }
}
```

如果重合就是下标和蛇重合就返回true，就会递归随机位置，直到不重合

如果蛇头和食物位置重合，就吃到了食物，就随机下一次食物出现的位置

主要就是

- 1、随机食物的位置，不能和墙、蛇重合
- 2、判断蛇是否吃到了食物
- 3、对食物的初始化位置进行设置以及绘制