

# Python程序设计报告

设计题目：贪吃蛇小程序

学生姓名：敬成超

专 业：物联网

班 级：一班

学 号：2023212388

指导教师：李俊照

完成日期：2025.1.4

# （一）需求和规格说明

**问题描述：**这是一个小游戏，游戏的目标是通过控制蛇头的移动来吃掉屏幕上出现的食物，同时避免撞到墙壁或自己的身体。游戏开始时，用户会看到一个蓝色的欢迎界面，提示他们按回车键开始游戏。游戏界面包括一个800x600像素的窗口，窗口中有一个网格，蛇的身体用黑色方块表示，食物用绿色和红色表示，分别代表普通食物和生命食物。

**编程任务：**

1. 初始化游戏环境，包括设置游戏窗口、定义颜色、游戏维度、蛇的初始设置、字体设置等。
2. 创建游戏窗口，设置标题，并定义游戏界面的初始状态。
3. 实现游戏循环，包括处理游戏事件、更新游戏状态、绘制游戏元素、检查游戏结束条件等。
4. 在游戏循环中，实现蛇的移动逻辑，包括蛇头的移动、蛇身的增长、食物的生成和生命食物的生成。
5. 实现碰撞检测，包括蛇头与自身碰撞、蛇头与游戏边界碰撞、蛇头与食物碰撞以及蛇头与生命食物碰撞。
6. 实现计分系统，记录蛇的长度变化，并在屏幕上显示当前得分。
7. 实现生命系统，记录玩家的生命数，并在屏幕上显示剩余生命数。
8. 实现游戏开始菜单，允许玩家通过按键开始游戏。
9. 当玩家游戏结束时，显示游戏结束信息，并提供重新开始或退出游戏的选项。

# （二）设计

## 1．设计思想

#### （1）. ****游戏核心逻辑****

**蛇的移动**：

蛇的移动通过方向键控制（上、下、左、右）。

每次移动时，蛇头向指定方向移动一格，蛇尾缩短一格。

如果蛇头碰到边界或自身身体，则扣除生命值或游戏结束。

**食物生成与碰撞检测**：

食物分为两种：普通食物（增加分数）和生命食物（增加生命值）。

食物随机生成在游戏窗口的网格中。

当蛇头与食物坐标重合时，触发碰撞检测，增加分数或生命值，并生成新的食物。

**生命值管理**：

初始生命值为 3。

当蛇碰到边界或自身身体时，扣除生命值。

当生命值为 0 时，游戏结束。

**时间限制**：

游戏设有时间限制（60 秒）。

时间结束后，扣除生命值并重置计时器。

如果生命值为 0，则游戏结束。

#### （2）. ****数据结构与存储****

#### 蛇身存储：

蛇的身体由一个列表 snake\_list 存储，每个元素表示蛇身体的一节，包含该节的坐标点。

每个坐标点表示为一个列表，例如 snake\_list = [[x1, y1], [x2, y2], ...]，每一对 (x, y) 表示蛇的一节在游戏窗口中的位置。

**增加节点**：当蛇吃到食物时，在 snake\_list 的末尾添加一个新的坐标点，表示蛇的身体变长。

**删除节点**：蛇移动时，从 snake\_list 的开头删除一个节点（即蛇尾），同时在末尾添加新的蛇头的位置，以此保持蛇的整体长度不变。

### 食物存储：

食物的位置通过随机函数生成，存储为坐标点，例如 foodx 和 foody 分别表示普通食物的 x 和 y 坐标。

### 生命值与分数：

生命值和分数通过变量存储，例如使用整数变量来表示当前的生命值和当前得分。

在游戏过程中，这些值会实时更新并在界面上显示。

#### （3）****游戏界面与交互****

**网格绘制**：

游戏窗口被划分为 10x10 的网格，蛇和食物的位置均对齐到网格。

网格线通过 pygame.draw.line 绘制。

**分数与生命值显示**：

分数显示在窗口左上角。

生命值显示在窗口右上角。

时间显示在窗口顶部中央。

**菜单与游戏状态**：

游戏开始时显示欢迎菜单，用户按 ENTER 键开始游戏。

游戏结束时显示“Game Over”界面，用户可以选择退出或重新开始。

## 2. 设计表示

初始化模块

设置窗口

定义颜色

设置初始参数

开始菜单模块

显示欢迎界面

等待用户输入

游戏主循环模块

处理用户输入

更新蛇的位置

检测碰撞

更新分数、生命值

| - 绘制界面元素

游戏结束模块

显示“Game Over”

等待用户选择

## 3. 核心算法

**事件处理 (**for event in pygame.event.get()**)**

监听用户输入事件，包括键盘按键和窗口关闭事件。

根据用户按下的方向键（上下左右），更新蛇的移动方向 (x1\_change, y1\_change)。

防止蛇反向移动（例如，如果蛇正在向右移动，则不能直接向左移动）。

**蛇的移动**

每次循环中，根据 (x1\_change, y1\_change) 更新蛇头的位置 (x1, y1)。

将新的蛇头位置 [x1, y1] 添加到 snake\_list 中。

如果蛇的长度超过 length\_of\_snake，则删除蛇尾（即 snake\_list 的第一个元素），以保持蛇的长度不变。

**边界碰撞检测**

检查蛇头是否超出游戏窗口的边界（x1 >= WIDTH 或 x1 < 0 或 y1 >= HEIGHT 或 y1 < 0）。

如果超出边界，则减少生命值 lives，并重置蛇的位置和长度。

如果生命值为0，则游戏结束。

**蛇身碰撞检测**

遍历 snake\_list 中除蛇头以外的所有部分，检查是否与蛇头位置重合。

如果重合，则表示蛇撞到了自己，游戏结束。

**食物碰撞检测**

检查蛇头是否与普通食物 (foodx, foody) 重合。

如果重合，则增加蛇的长度 length\_of\_snake，并重新生成食物的位置。

**生命食物碰撞检测**

检查蛇头是否与生命食物 (life\_foodx, life\_foody) 重合。

如果重合，则增加生命值 lives 和蛇的长度 length\_of\_snake，并重新生成生命食物的位置。

。

# （三）用户手册

**1. 游戏简介**

贪吃蛇游戏是一款经典的街机游戏，玩家通过控制一条蛇在游戏区域内移动，吃掉食物以增加蛇的长度，同时避免蛇头与自身身体或游戏边界碰撞。游戏中还有生命食物，玩家可以通过吃掉这些食物来增加生命值。

**2. 游戏目标**

吃掉绿色的普通食物以增加蛇的长度和分数。

吃掉红色的生命食物以增加生命值和体长。

避免蛇头与蛇身和游戏边界碰撞，保持自己存活。

**3. 游戏开始**

启动游戏后，您会看到欢迎界面。

按 ENTER 键开始游戏。

**4. 控制说明**

使用以下箭头键控制蛇的移动方向：

**左箭头**：向左移动

**右箭头**：向右移动

**上箭头**：向上移动

**下箭头**：向下移动

**注意**：蛇不能立即反向移动。例如，如果蛇正在向右移动，您不能直接按左箭头。

**5. 游戏界面**

**分数**：游戏顶部左侧显示您的当前分数（蛇的长度减去1）。

**生命值**：游戏顶部右侧显示剩余生命值。

**计时器**：游戏中间显示剩余时间（60秒），时间耗尽会减少生命值。

**网格**：游戏区域中有白色网格，帮助您更好地判断蛇的移动。

**6. 食物**

**普通食物**：绿色矩形，蛇吃到后会增加长度和分数。

**生命食物**：红色矩形，蛇吃到后会增加生命值和长度。

**7. 游戏规则**

蛇会在画面内不断移动，您需要控制它的方向。

如果蛇头与自身身体或游戏边界相撞，您将失去一条生命。

游戏开始时，您有3条生命，生命值归零后游戏结束。

游戏时间为60秒，如果时间耗尽，您将失去一条生命。

**8. 游戏结束**

当您失去所有生命值时，游戏会结束。

屏幕会显示“Game Over”信息，您可以选择按 Q 键退出游戏或按 C 键重启游戏。

**9. 常见问题**

**游戏无法启动**：请确保您已经安装了Pygame库，您可以使用 pip install pygame 来安装。

**如何提高分数**：尽量多吃食物，特别是生命食物，增加生命值后可以更好地继续游戏。

**蛇的移动不流畅**：请确保游戏以合理的帧速率运行，适当调整 snake\_speed 参数。

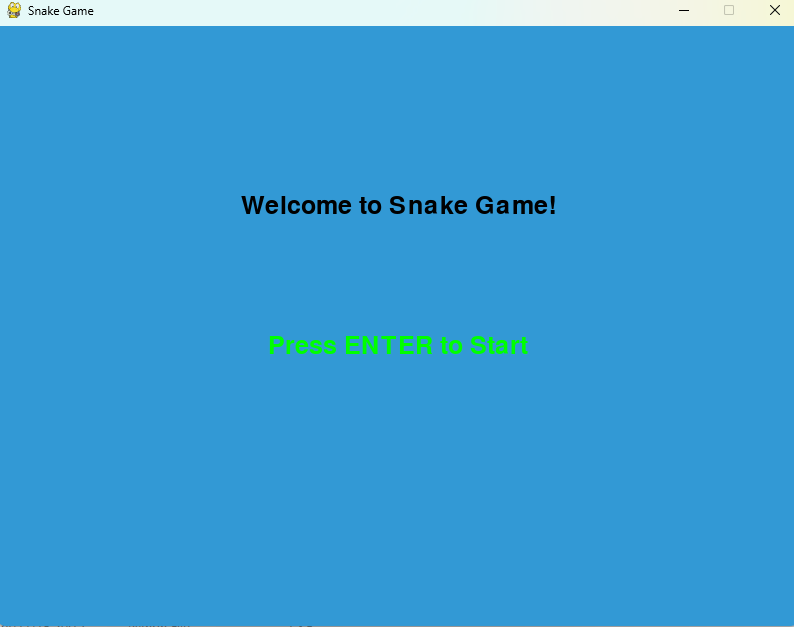
# （四）调试及测试

## 1. 测试数据：

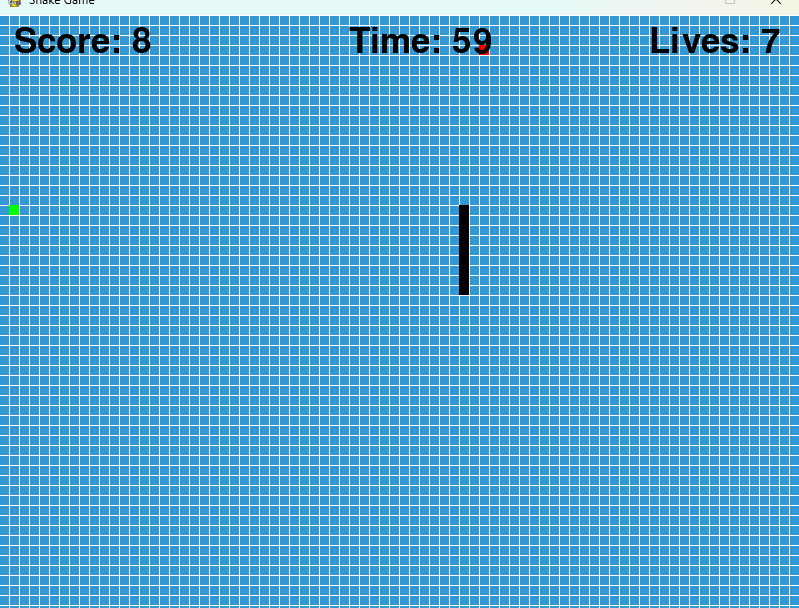
无

## 2. 测试结果：

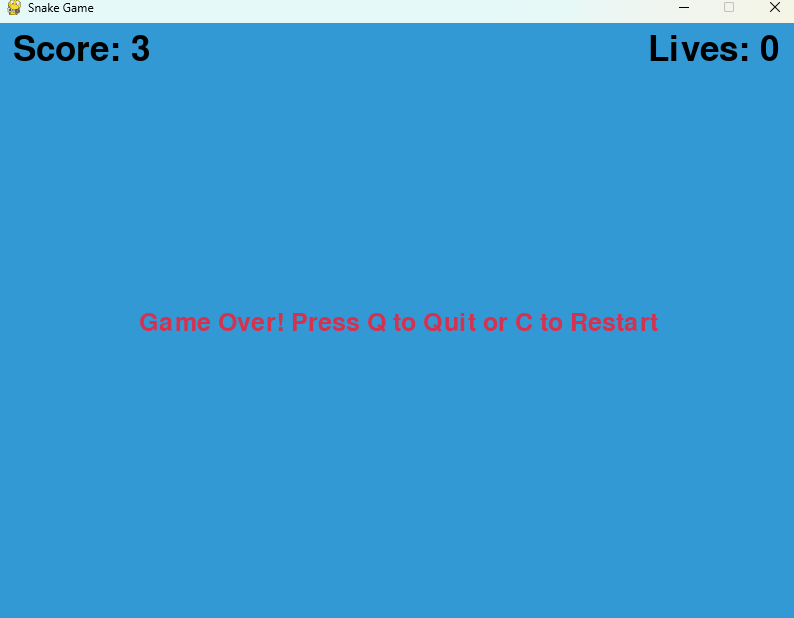
**主菜单**



**开始游戏**



**游戏结算**



## 3. 进一步改进：

一开始没有添加网格，对于位置不太看得清，后来加了网格，玩起来更容易了。另外，还调低了速度，让难度降低了很多，也有了游戏体验。

# (五) 感想

# 在开发这个贪吃蛇游戏的过程中，我遇到了一些困难，同时也获得了一些收获。确保蛇头与食物、蛇身、边界之间的碰撞检测的逻辑较为复杂，尤其是在需要同时检测多个条件时。初始版本中，处理碰撞时可能出现漏判的问题，需要仔细检查逻辑以确保所有情况都能如预期处理。管理游戏的不同状态（如开始、进行中、游戏结束）对于简单的游戏来说可看似简单，但需要合理设计变量的使用。需要确保在游戏结束后能够正常重启或退出，有时状态之间的切换会引起意想不到的错误。在实现时间限制时，需要准确计算已用时间，并在时间耗尽时及时减少生命值，确保游戏逻辑的连贯性。对时间的管理要与游戏的整体逻辑紧密结合，确保玩家能够感知到时间的紧迫感。游戏的视觉效果和用户交互是一个重要的方面，最初可能没有考虑到用户体验，比如食物与蛇的颜色对比、信息的展示位置等。设计出一个既简单又美观的用户界面是个挑战，尤其是对于初学者来说。在实现过程中，巩固和提升了对Python和Pygame库的使用技能，学习了如何利用其功能来构建游戏逻辑。在调试过程中，提高了代码阅读和理解能力，对基本的编程结构和逻辑逐渐熟悉。遇到困难时，通过查找文档、参考资料和调试代码，锻炼了自己的问题解决能力。学会了如何系统地分析问题，寻找解决方案，而不是仅依赖于直觉。理解了游戏设计的基本原则，包括如何设计游戏循环、处理用户输入、管理游戏状态等。对游戏的可玩性、平衡性以及玩家体验有了更深刻的理解。过程激发了创造力，在面临设计挑战时，我尝试了不同的解决方案，并思考如何增加游戏的趣味性。学会了将逻辑思维应用于游戏设计中，使得游戏的运行更为流畅和合理。最终能够完成这个游戏项目，亲眼看到自己的代码变成一个完整的、可玩的游戏，带来了很大的成就感和满足感。当然还有**改进的空间**：****

可以考虑添加音效和背景音乐来增强游戏的体验，使得游戏更加生动有趣。

随着时间推移而增加蛇的速度，或者在游戏中引入不同的难度级别，可以提高游戏的挑战性。

可以增加保存游戏状态或实现排行榜功能，让玩家能够查看自己的分数和排名，提高游戏的竞争性

# 附录：

import pygame

import time

import random

# Initialize pygame

pygame.init()

# Define colors

WHITE = (255, 255, 255)

YELLOW = (255, 255, 102)

BLACK = (0, 0, 0)

RED = (213, 50, 80)

GREEN = (0, 255, 0)

BLUE = (50, 153, 213)

LIFE\_COLOR = (255, 0, 0) # Color for life food

# Game window dimensions

WIDTH = 800

HEIGHT = 600

# Create game window

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Snake Game')

# Snake initial settings

snake\_block = 10

snake\_speed = 10 # Slightly slower speed

lives = 3 # Number of lives

time\_limit = 60 # Time limit in seconds

# Font settings

font\_style = pygame.font.Font(pygame.font.get\_default\_font(), 25)

score\_font = pygame.font.Font(pygame.font.get\_default\_font(), 35)

def draw\_grid():

for x in range(0, WIDTH, snake\_block):

pygame.draw.line(screen, WHITE, (x, 0), (x, HEIGHT))

for y in range(0, HEIGHT, snake\_block):

pygame.draw.line(screen, WHITE, (0, y), (WIDTH, y))

def score\_display(score):

value = score\_font.render(f"Score: {score}", True, BLACK)

screen.blit(value, [15, 10]) # Set display position

def lives\_display(lives):

value = score\_font.render(f"Lives: {lives}", True, BLACK)

screen.blit(value, [WIDTH - 150, 10]) # 修改显示位置

def timer\_display(seconds):

value = score\_font.render(f"Time: {seconds}", True, BLACK)

screen.blit(value, [(WIDTH // 2) - 50, 10]) # Centered display position

def draw\_snake(snake\_block, snake\_list):

for x in snake\_list:

pygame.draw.rect(screen, BLACK, [x[0], x[1], snake\_block, snake\_block])

def message(msg, color, center\_position):

mesg = font\_style.render(msg, True, color)

text\_rect = mesg.get\_rect(center=center\_position)

screen.blit(mesg, text\_rect)

def show\_start\_menu():

screen.fill(BLUE)

message("Welcome to Snake Game!", BLACK, (WIDTH / 2, HEIGHT / 3 - 20)) # Message at upper center

message("Press ENTER to Start", GREEN, (WIDTH / 2, HEIGHT / 2 + 20)) # Below the welcome message

pygame.display.update()

waiting = True

while waiting:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_RETURN:

waiting = False

def game\_loop():

global lives

game\_over = False

game\_close = False

x1 = WIDTH / 2

y1 = HEIGHT / 2

x1\_change = 0

y1\_change = 0

snake\_list = []

length\_of\_snake = 1

foodx = round(random.randrange(0, WIDTH - snake\_block) / 10.0) \* 10.0

foody = round(random.randrange(0, HEIGHT - snake\_block) / 10.0) \* 10.0

life\_foodx = round(random.randrange(0, WIDTH - snake\_block) / 10.0) \* 10.0

life\_foody = round(random.randrange(0, HEIGHT - snake\_block) / 10.0) \* 10.0

start\_time = time.time() # Start timer

while not game\_over:

while game\_close:

screen.fill(BLUE)

message("Game Over! Press Q to Quit or C to Restart", RED, (WIDTH / 2, HEIGHT / 2)) # Game over message

score\_display(length\_of\_snake - 1)

lives\_display(lives) # Display lives

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_q:

game\_over = True

game\_close = False

if event.key == pygame.K\_c:

game\_loop()

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_over = True

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT and x1\_change != snake\_block: # Prevent reverse

x1\_change = -snake\_block

y1\_change = 0

elif event.key == pygame.K\_RIGHT and x1\_change != -snake\_block: # Prevent reverse

x1\_change = snake\_block

y1\_change = 0

elif event.key == pygame.K\_UP and y1\_change != snake\_block: # Prevent reverse

y1\_change = -snake\_block

x1\_change = 0

elif event.key == pygame.K\_DOWN and y1\_change != -snake\_block: # Prevent reverse

y1\_change = snake\_block

x1\_change = 0

if x1 >= WIDTH or x1 < 0 or y1 >= HEIGHT or y1 < 0:

lives -= 1 # Lose a life

if lives > 0:

x1 = WIDTH / 2

y1 = HEIGHT / 2

snake\_list.clear()

length\_of\_snake = max(1, length\_of\_snake - 1) # Reduce length

else:

game\_close = True

x1 += x1\_change

y1 += y1\_change

screen.fill(BLUE)

draw\_grid() # Draw grid

# Draw food

pygame.draw.rect(screen, GREEN, [foodx, foody, snake\_block, snake\_block]) # Normal food

pygame.draw.rect(screen, LIFE\_COLOR, [life\_foodx, life\_foody, snake\_block, snake\_block]) # Life food

snake\_head = []

snake\_head.append(x1)

snake\_head.append(y1)

snake\_list.append(snake\_head)

if len(snake\_list) > length\_of\_snake:

del snake\_list[0]

for x in snake\_list[:-1]:

if x == snake\_head:

game\_close = True

draw\_snake(snake\_block, snake\_list)

score\_display(length\_of\_snake - 1)

lives\_display(lives) # Update lives display

# Calculate elapsed time and update the timer display

elapsed\_time = int(time.time() - start\_time)

remaining\_time = time\_limit - elapsed\_time

if remaining\_time <= 0:

lives -= 1

start\_time = time.time() # Reset timer after losing a life

timer\_display(max(remaining\_time, 0)) # Display timer

pygame.display.update()

# Check for food collision

if x1 == foodx and y1 == foody:

foodx = round(random.randrange(0, WIDTH - snake\_block) / 10.0) \* 10.0

foody = round(random.randrange(0, HEIGHT - snake\_block) / 10.0) \* 10.0

length\_of\_snake += 1

start\_time = time.time() # Reset timer upon eating food

# Check for life food collision

if x1 == life\_foodx and y1 == life\_foody:

life\_foodx = round(random.randrange(0, WIDTH - snake\_block) / 10.0) \* 10.0

life\_foody = round(random.randrange(0, HEIGHT - snake\_block) / 10.0) \* 10.0

lives += 1 # Increase lives by 1

length\_of\_snake += 1 # Increase snake length

pygame.time.Clock().tick(snake\_speed)

pygame.quit()

quit()

if \_\_name\_\_ == "\_\_main\_\_":

show\_start\_menu() # Show start menu before entering the game loop

game\_loop()