人工智能程序设计


python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for 1 in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)

人工智能程序设计

9.3 经典游戏实现

北京石油化工学院 人工智能研究院

刘 强

本节概述

通过制作经典游戏，综合运用前面学到的所有知识：

- **贪吃蛇游戏**：移动、碰撞检测、分数统计
- **打砖块游戏**：物理运动、碰撞检测、关卡设计

体验完整的游戏开发流程。

9.3.1 贪吃蛇游戏

贪吃蛇是一个经典的游戏，包含核心游戏要素：

- 蛇由多个方块组成，头部控制移动方向
- 吃到食物后蛇身变长，分数增加
- 撞到墙壁或自己身体时游戏结束

示例 9.3.1：贪吃蛇游戏

代码分为6个步骤：

1. **步骤1**：导入库和类初始化
2. **步骤2**：食物生成方法
3. **步骤3**：事件处理方法
4. **步骤4**：游戏状态更新方法
5. **步骤5**：绘制方法
6. **步骤6**：主游戏循环

步骤1：导入库和类初始化

导入必要的库并建立游戏的基础框架。

```
import pygame
import random
import sys

class SnakeGame:
    def __init__(self):
        pygame.init()
        self.width, self.height = 640, 480
        self.screen = pygame.display.set_mode((self.width, self.height))
        pygame.display.set_caption("贪吃蛇游戏")
        self.clock = pygame.time.Clock()
```

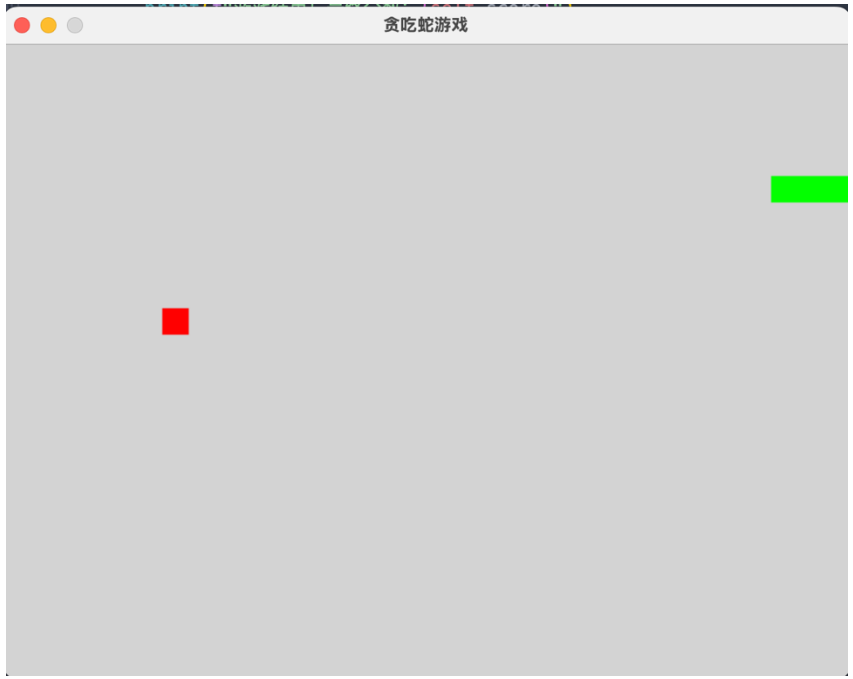
步骤1：颜色和游戏参数

- **self.snake**：列表存储蛇身各节的坐标，第一个元素是蛇头

```
# 颜色定义
self.BLACK = (0, 0, 0)
self.LIGHT_GRAY = (211, 211, 211)
self.GREEN = (0, 255, 0)
self.RED = (255, 0, 0)

# 游戏参数
self.block_size = 20
self.snake = [(100, 100), (80, 100), (60, 100)]
self.direction = (self.block_size, 0)
self.food = self.generate_food()
self.score = 0
```

贪吃蛇游戏效果



步骤2：食物生成方法

随机生成食物位置，确保不与蛇身重叠。

- `random.randint()`：生成指定范围内的随机整数
- `//`：整除运算符，确保坐标是 `block_size` 的倍数

```
def generate_food(self):
    """生成食物位置"""
    while True:
        x = random.randint(0, (self.width - self.block_size) // self.block_size) * self.block_size
        y = random.randint(0, (self.height - self.block_size) // self.block_size) * self.block_size
        if (x, y) not in self.snake:
            return (x, y)
```

步骤3：事件处理方法

处理用户的键盘输入来控制蛇的移动方向。

```
def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and self.direction != (0, self.block_size):
                self.direction = (0, -self.block_size)
            elif event.key == pygame.K_DOWN and self.direction != (0, -self.block_size):
                self.direction = (0, self.block_size)
            elif event.key == pygame.K_LEFT and self.direction != (self.block_size, 0):
                self.direction = (-self.block_size, 0)
            elif event.key == pygame.K_RIGHT and self.direction != (-self.block_size, 0):
                self.direction = (self.block_size, 0)
    return True
```

步骤3：事件处理说明

- `pygame.KEYDOWN`：按键按下事件类型
- `pygame.K_UP/DOWN/LEFT/RIGHT`：方向键常量
- `self.direction != (0, self.block_size)`：防止蛇反向移动

步骤4：游戏状态更新方法

实现游戏的核心逻辑：蛇的移动、碰撞检测和食物处理。

```
def update(self):
    # 移动蛇头
    head = self.snake[0]
    new_head = (head[0] + self.direction[0], head[1] + self.direction[1])

    # 检查边界碰撞
    if (new_head[0] < 0 or new_head[0] >= self.width or
        new_head[1] < 0 or new_head[1] >= self.height):
        return False

    # 检查自身碰撞
    if new_head in self.snake:
        return False
```

步骤4：食物处理

- `self.snake.insert(0, new_head)`：在列表开头插入新蛇头
- `self.snake.pop()`：移除蛇尾（如果没吃到食物）

```
        self.snake.insert(0, new_head)

        # 检查是否吃到食物
        if new_head == self.food:
            self.score += 10
            self.food = self.generate_food()
        else:
            self.snake.pop()

        return True
```

步骤5：绘制方法

将游戏的视觉元素绘制到屏幕上。

```
def draw(self):
    self.screen.fill(self.LIGHT_GRAY)

    # 绘制蛇
    for segment in self.snake:
        pygame.draw.rect(self.screen, self.GREEN,
                         (segment[0], segment[1], self.block_size, self.block_size))

    # 绘制食物
    pygame.draw.rect(self.screen, self.RED,
                     (self.food[0], self.food[1], self.block_size, self.block_size))

    pygame.display.flip()
```

步骤6：主游戏循环

整合所有功能模块，实现游戏的主要运行逻辑。

```
def run(self):
    running = True
    while running:
        running = self.handle_events()
        if running:
            running = self.update()
        self.draw()
        self.clock.tick(10)  # 控制游戏速度

    print(f"游戏结束！最终分数：{self.score}")

    # 等待用户关闭窗口
    waiting = True
    while waiting:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                waiting = False
        self.clock.tick(30)

    pygame.quit()
    sys.exit()
```

9.3.2 打砖块游戏

打砖块游戏包含球的物理运动、碰撞检测、关卡设计等要素：

- 球在屏幕中弹跳，遇到边界或物体会反弹
- 挡板跟随鼠标移动，防止球掉落
- 球击中砖块后砖块消失，所有砖块消失即胜利

打砖块：步骤1 - 类初始化

```
import pygame
import sys

class BreakoutGame:
    def __init__(self):
        pygame.init()
        self.width, self.height = 800, 600
        self.screen = pygame.display.set_mode((self.width, self.height))
        pygame.display.set_caption("打砖块游戏")
        self.clock = pygame.time.Clock()

        # 颜色定义
        self.WHITE = (255, 255, 255)
        self.BLUE = (0, 0, 255)
        self.RED = (255, 0, 0)
        self.BLACK = (0, 0, 0)
        self.LIGHT_GRAY = (211, 211, 211)
```

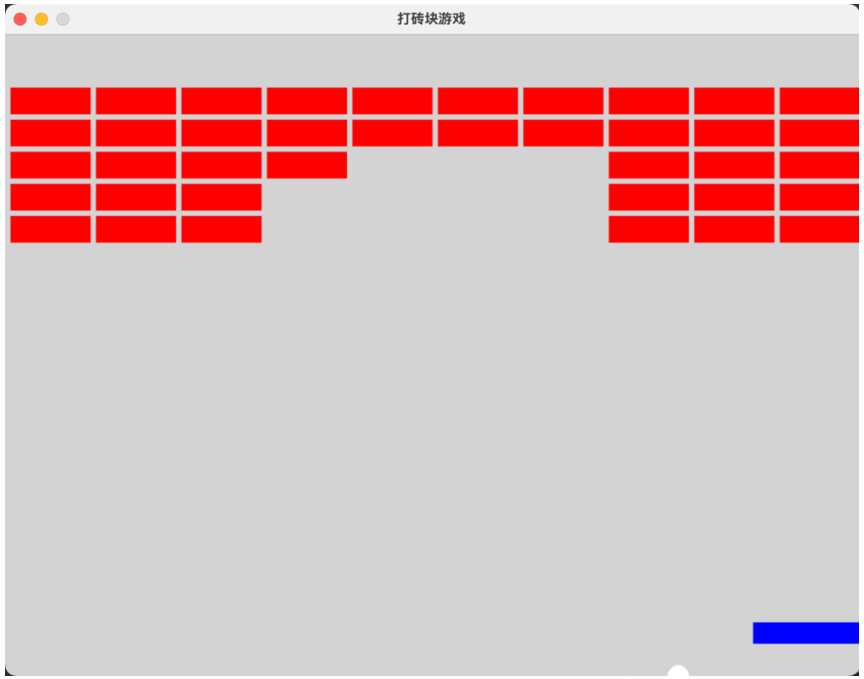
打砖块：游戏对象初始化

- **self.ball_pos**：列表存储球的 **x** 和 **y** 坐标位置
- **self.ball_speed**：列表存储球在 **x** 和 **y** 方向的移动速度

```
# 游戏对象
self.ball_pos = [400, 300]
self.ball_speed = [5, -5]
self.ball_radius = 10

self.paddle_width, self.paddle_height = 100, 20
self.paddle_x = 350
self.paddle_y = 550
```

打砖块游戏效果



打砖块：步骤2 - 创建砖块布局

使用嵌套循环创建砖块阵列布局。

- **`pygame.Rect()`**：创建矩形对象，参数为 **`(x, y, 宽度, 高度)`**
- **`col * 80 + 5`**：计算砖块的 **`x`** 坐标，每列间隔 **`80`** 像素
- **`row * 30 + 50`**：计算砖块的 **`y`** 坐标，每行间隔 **`30`** 像素

```
# 创建砖块
self.bricks = []
for row in range(5):
    for col in range(10):
        brick = pygame.Rect(col * 80 + 5, row * 30 + 50, 75, 25)
        self.bricks.append(brick)
```

打砖块：步骤3 - 事件处理

处理用户输入并实现挡板的鼠标控制。

- `pygame.mouse.get_pos()`：获取鼠标当前位置
- `max(0, min(...))`：限制挡板位置在屏幕范围内

```
def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return False

    # 挡板跟随鼠标
    mouse_x = pygame.mouse.get_pos()[0]
    self.paddle_x = max(0, min(mouse_x - self.paddle_width // 2,
                               self.width - self.paddle_width))
    return True
```

打砖块：步骤4 - 物理运动

实现球的物理运动和边界碰撞检测。

```
def update(self):
    # 更新球的位置
    self.ball_pos[0] += self.ball_speed[0]
    self.ball_pos[1] += self.ball_speed[1]

    # 边界碰撞检测
    if self.ball_pos[0] <= self.ball_radius or self.ball_pos[0] >= self.width - self.ball_radius:
        self.ball_speed[0] = -self.ball_speed[0]
    if self.ball_pos[1] <= self.ball_radius:
        self.ball_speed[1] = -self.ball_speed[1]

    # 球掉落检测
    if self.ball_pos[1] >= self.height:
        return False
```

打砖块：碰撞检测

```
# 挡板碰撞检测
paddle_rect = pygame.Rect(self.paddle_x, self.paddle_y,
                          self.paddle_width, self.paddle_height)
ball_rect = pygame.Rect(self.ball_pos[0] - self.ball_radius,
                        self.ball_pos[1] - self.ball_radius,
                        self.ball_radius * 2, self.ball_radius * 2)

if ball_rect.colliderect(paddle_rect) and self.ball_speed[1] > 0:
    self.ball_speed[1] = -self.ball_speed[1]

# 砖块碰撞检测
for brick in self.bricks[:]:
    if ball_rect.colliderect(brick):
        self.bricks.remove(brick)
        self.ball_speed[1] = -self.ball_speed[1]
        break
```

打砖块：碰撞检测说明

- `ball_rect.colliderect(paddle_rect)`：检测两个矩形是否相交
- `self.bricks[:]`：创建砖块列表的副本，安全地在循环中删除元素
- `self.ball_speed[0] = -self.ball_speed[0]`：速度反转实现反弹

打砖块：步骤5 - 绘制画面

绘制所有游戏元素到屏幕上。

```
def draw(self):
    self.screen.fill(self.LIGHT_GRAY)

    # 绘制球
    pygame.draw.circle(self.screen, self.WHITE,
                       (int(self.ball_pos[0]), int(self.ball_pos[1])),
                       self.ball_radius)

    # 绘制挡板
    pygame.draw.rect(self.screen, self.BLUE,
                     (self.paddle_x, self.paddle_y,
                      self.paddle_width, self.paddle_height))

    # 绘制砖块
    for brick in self.bricks:
        pygame.draw.rect(self.screen, self.RED, brick)

    pygame.display.flip()
```

打砖块：步骤6 - 主游戏循环

- `self.clock.tick(60)`：设置游戏运行速度为每秒 60 帧

```
def run(self):
    running = True
    while running:
        running = self.handle_events()
        if running:
            running = self.update()
        self.draw()
        self.clock.tick(60)

    # 等待用户关闭窗口
    waiting = True
    while waiting:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                waiting = False
        self.clock.tick(30)

    pygame.quit()
    sys.exit()
```

9.3.3 Ask AI：探索更多游戏类型

完成了基础游戏后，可以向 AI 助手询问更多游戏开发方向：

- "如何制作平台跳跃游戏？"
- "怎样实现简单的 RPG 游戏系统？"
- "如何添加游戏存档功能？"
- "怎样制作多人对战游戏？"

实践练习

练习 9.3.1：游戏功能扩展

1. 为贪吃蛇游戏添加分数显示和最高分记录
2. 为打砖块游戏添加多球模式和特殊砖块
3. 创建游戏开始菜单和游戏结束画面

实践练习

练习 9.3.2：原创游戏制作

1. 制作一个简单的飞机大战游戏
2. 设计一个迷宫寻路游戏
3. 创建属于自己的原创小游戏

本节小结

- **贪吃蛇游戏**：列表操作、方向控制、碰撞检测
- **打砖块游戏**：物理运动、`Rect` 碰撞、鼠标控制
- **核心技术**：
  - `pygame.Rect()` 矩形对象
  - `colliderect()` 碰撞检测
  - 游戏循环架构