

# 第五课 吃豆子

这节课让我们回到pygame游戏中，并且我们同步要讲一个编程中重要的概念，类(Class)。

## 类的基础知识

让我们向GPT提问

1 什么是Python编程中的类？请给我讲解相关的基础知识并给出一些例子

GPT的回答为:

在Python编程中，类（Class）是一种面向对象编程（OOP）的基本概念。类可以看作是创建对象的蓝图或模板。通过类，我们可以定义对象的属性（变量）和方法（函数）。以下是类的基础知识及一些例子：

### 基础知识

- 定义类：**使用关键字 `class` 来定义一个类。
- 构造方法：**在类中，`__init__` 方法是一个特殊的方法，它会在创建类的实例时自动调用。  
`self` 参数代表类的实例本身。
- 属性：**类的属性是类中定义的变量，用于存储数据。
- 方法：**类中的函数称为方法，用于定义对象的行为。
- 实例化：**创建类的实例（对象）的过程称为实例化。

### 例子

下面是一个简单的类定义和实例化的例子：

```
1 # 定义一个名为Person的类
2 class Person:
3     def __init__(self, name, age):
4         self.name = name # 属性
5         self.age = age    # 属性
6     def say_hello(self): # 方法
7         print(f"Hello, my name is {self.name} and I am {self.age} years old.")
```

```
8 # 实例化Person类
9 person1 = Person("Alice", 30)
10 person2 = Person("Bob", 25)
11 # 调用方法
12 person1.say_hello()
13 person2.say_hello()
```

输出：

```
1 Hello, my name is Alice and I am 30 years old.
2 Hello, my name is Bob and I am 25 years old.
```

在这个例子中，`Person` 是一个类，它有两个属性：`name` 和 `age`，以及一个方法 `say_hello`。

...

同学们可以阅读这里的例子，并且理解类的定义和使用。

## 阅读基础的pygame代码

我们在之前学习过一个最基础的用键盘控制pygame游戏。在这个游戏中，会有一个红色的正方形，并且对应的坐标`square_x`和`square_y`，会随着键盘的响应而变化。

```
1 import pygame, sys
2 pygame.init() # 初始化Pygame
3 width, height = 640, 480 # 设置窗口大小
4 screen = pygame.display.set_mode((width, height)) # 创建窗口
5 RED = (255, 0, 0) # 设置颜色常量
6 square_size = 20 # 设置正方形大小
7 square_x = width // 2 - square_size // 2 # 设置正方形初始x坐标
8 square_y = height // 2 - square_size // 2 # 设置正方形初始y坐标
9 clock = pygame.time.Clock() # 创建时钟对象
10 fps = 60 # 设置帧率
11 running = True # 游戏主循环标志
12 while running: # 游戏主循环
13     for event in pygame.event.get(): # 遍历事件
14         if event.type == pygame.QUIT: # 检查退出事件
15             running = False # 修改循环标志以退出
16     keys = pygame.key.get_pressed() # 获取按键状态
17     square_y -= 10 if keys[pygame.K_w] else 0 # 向上移动
18     square_y += 10 if keys[pygame.K_s] else 0 # 向下移动
```

```

19     square_x -= 10 if keys[pygame.K_a] else 0 # 向左移动
20     square_x += 10 if keys[pygame.K_d] else 0 # 向右移动
21     screen.fill((0, 0, 0)) # 填充背景色
22     # 绘制正方形
23     pygame.draw.rect(screen, RED, (square_x, square_y, square_size,
square_size))
24     pygame.display.flip() # 更新屏幕显示
25     clock.tick(fps) # 控制帧率
26 pygame.quit() # 退出Pygame
27 sys.exit() # 退出程序

```

这里我们高亮标出了和红色正方形直接有关的代码。这里同学可以思考一个问题，如果我们游戏中有更多的正方形，那么对应的黄色部分的代码就会成倍地增加，这是我们不希望看到的。同时，我们也可以看到，正方形的代码主要涉及三个板块，初始化、位置更新和渲染。这也为我们后续进行面向对象的重构提出了思路。

## 让GPT给出类的设计

我们可以让GPT来帮助我们给出类设计的建议

- 1 我想为pygame中的物体设计一个类，这些类一般具有哪些方法？

GPT的回复为

在pygame中设计一个物体类时，通常会包含以下几种方法，这些方法可以帮助你管理物体的状态和行为：

1. **初始化方法** (`**__init__**`) \*\*: 用于初始化物体，设置其初始属性，如位置、速度、颜色、大小等。

```

1 def __init__(self, x, y, width, height, color):
2     self.x = x
3     self.y = y
4     self.width = width
5     self.height = height
6     self.color = color
7     # 其他属性

```

2. **绘制方法** (`**draw**`) \*\*: 用于在屏幕上绘制物体。

```
1 def draw(self, screen):
2     pygame.draw.rect(screen, self.color, (self.x, self.y, self.width,
3     self.height))
```

3. 更新方法 ( `**update**` ) \*\*：用于更新物体的状态，比如位置、速度等。

```
1 def update(self):
2     self.x += self.vx
3     self.y += self.vy
4     # 更新其他属性
```

...

这里可以看到，GPT给出的类的建议。在大多数时候，pygame的物体都会有这三个成员函数。让我们进一步来看怎么实现一个pygame中物体的类。

## 让GPT写一个小汽车的类

因为很多时候我们是希望写出来的游戏是有动画的，动画可以理解成每个物体都有自己的目标点。结合之前的分析，我们可以尝试写一个小汽车的类。

```
1 在images/car.jpg存储了一张小汽车的照片，
2
3 我希望实现一个pygame中的Car类
4
5 这个类可以用( x, y, speed = 5, size = 100)来进行初始化
6
7 初始化后current_x, current_y分别为x,y为汽车中心的位置
8
9 并且可以用set_target( x, y ) 来设置目标的位置
10
11 并且Car类有update()方法，每次会以speed的速度向目标移动，如果移动到了目标位置(speed以
    内)，则在目标位置停止移动
12
13 另外还有draw(screen)方法可以在pygame中进行渲染
14
15 请帮我实现
```

这里GPT很快帮我们写好了对应的程序

```

1 import pygame
2
3 class Car:
4     def __init__(self, x, y, speed=5, size=100): # 初始化Car类
5         self.image =
pygame.transform.scale(pygame.image.load('images/car.jpg'), (size, size)) # 加
载并缩放汽车图片
6         self.current_x = x # 初始化汽车中心x坐标
7         self.current_y = y # 初始化汽车中心y坐标
8         self.speed = speed # 初始化汽车速度
9         self.target_x = x # 初始化目标x坐标
10        self.target_y = y # 初始化目标y坐标
11        self.size = size # 初始化汽车尺寸
12
13    def set_target(self, x, y): # 设置汽车移动的目标位置
14        self.target_x = x # 更新目标x坐标
15        self.target_y = y # 更新目标y坐标
16
17    def update(self): # 更新汽车位置
18        dx = self.target_x - self.current_x # 计算x轴方向距离
19        dy = self.target_y - self.current_y # 计算y轴方向距离
20        distance = (dx**2 + dy**2)**0.5 # 计算与目标位置的距离
21
22        if distance > self.speed: # 如果距离大于速度, 则移动
23            dx /= distance # 计算x轴单位向量
24            dy /= distance # 计算y轴单位向量
25            self.current_x += dx * self.speed # 沿x轴移动
26            self.current_y += dy * self.speed # 沿y轴移动
27        else: # 到达目标位置
28            self.current_x = self.target_x # 设置汽车位置为目标的x坐标
29            self.current_y = self.target_y # 设置汽车位置为目标的y坐标
30
31    def draw(self, screen): # 在屏幕上绘制汽车
32        rect = self.image.get_rect(center=(self.current_x, self.current_y)) #
创建以汽车中心为基准的矩形
33        screen.blit(self.image, rect) # 将汽车图片绘制到屏幕上
34
35 if __name__ == '__main__': # 使用示例
36     pygame.init() # 初始化pygame
37
38     screen = pygame.display.set_mode((800, 600)) # 设置屏幕大小
39
40     car = Car(50, 50) # 创建Car实例
41
42     car.set_target(400, 300) # 设置目标位置
43
44     running = True

```

```

45     while running: # 游戏主循环
46         for event in pygame.event.get():
47             if event.type == pygame.QUIT:
48                 running = False
49
50         car.update() # 更新汽车位置
51         screen.fill((255, 255, 255)) # 填充背景色
52         car.draw(screen) # 绘制汽车
53         pygame.display.flip() # 更新屏幕显示
54
55         pygame.time.Clock().tick(60) # 控制游戏帧率
56
57     pygame.quit() # 退出游戏

```

同学们可以把这段代码完整的保存在Car.py中，运行程序，可以发现汽车会一开始生成在画面的左上角，并逐渐移动到屏幕的中间。这个Car类就被成功建立出来了。

## 使用Car类和鼠标交互

这里我们需要把Car.py放在项目目录中，然后我们额外开一个程序，比如叫做main.py。现在我们在main.py中引用这个Car.py，要怎么引用呢？在main.py中尝试运行如下代码

```

1 from Car import Car
2 my_car = Car(0,0)

```

如果程序可以顺利运行，就说明代码和Car.py被放在了同一个目录下，并且能够顺利运行。这个时候我们就可以告诉GPT我们的类已经顺利写完了，可以进一步去生成更复杂的代码了

```

1 我已经实现了一个Car类，可以通过from Car import Car载入
2 这个类可以用( x, y, speed = 5, size = 100)来进行初始化
3 可以用set_target( x, y ) 来设置目标的位置
4 有update()方法，每次会以speed的速度向目标移动
5 有draw(screen)方法可以在pygame中进行渲染
6
7 请基于Car类，帮我写一个鼠标交互的游戏，游戏的画面是800 * 600
8
9 鼠标点到哪里，就把Car的target设置为对应的坐标

```

前面五句相当于对Car类的功能进行了说明，当然，第二到四句的部分，也可以用Car类对应的代码来替代。但是我们要注意，随着我们的代码功能越来越复杂，一般来说代码的长度，是会远超文字版本

的功能描述的。所以这种向GPT描述我们自己实现的类如何载入，并且对应的功能和用法是什么的开发模式，会变得更加容易维护。对于大型的项目来说，也更容易理清各个模块之间的关系。

```
1 import pygame
2 from Car import Car # 导入自定义的Car类
3 running = True
4 pygame.init() # 初始化pygame
5 screen = pygame.display.set_mode((800, 600)) # 设置屏幕大小
6 car = Car(50, 50) # 创建Car实例
7 while running: # 游戏主循环
8     for event in pygame.event.get():
9         if event.type == pygame.QUIT: # 检测到退出事件
10             running = False
11         elif event.type == pygame.MOUSEBUTTONDOWN: # 检测鼠标点击事件
12             mouse_x, mouse_y = event.pos # 获取鼠标点击的位置
13             car.set_target(mouse_x, mouse_y) # 设置Car的目标位置
14         car.update() # 更新汽车位置
15         screen.fill((255, 255, 255)) # 填充背景色
16         car.draw(screen) # 绘制汽车
17         pygame.display.flip() # 更新屏幕显示
18         pygame.time.Clock().tick(60) # 控制游戏帧率
19 pygame.quit() # 退出游戏
```

这里我们高亮出了car类涉及到的对应的代码。这个代码我们存储在了lesson5/Car\_with\_mouse.py中。

## 使用Car类和键盘交互

我们再来尝试看看做键盘的交互

- 1 我已经实现了一个Car类，可以通过from Car import Car载入
- 2 这个类可以用( x, y, speed = 5, size = 100)来进行初始化
- 3 可以用set\_target( x, y ) 来设置目标的位置
- 4 有update()方法，每次会以speed的速度向目标移动
- 5 有draw(screen)方法可以在pygame中进行渲染
- 6
- 7 请基于Car类，帮我写一个键盘交互的游戏，游戏的画面是800 \* 600
- 8
- 9 按上下左右的时候，汽车会向着对应的方向移动50像素

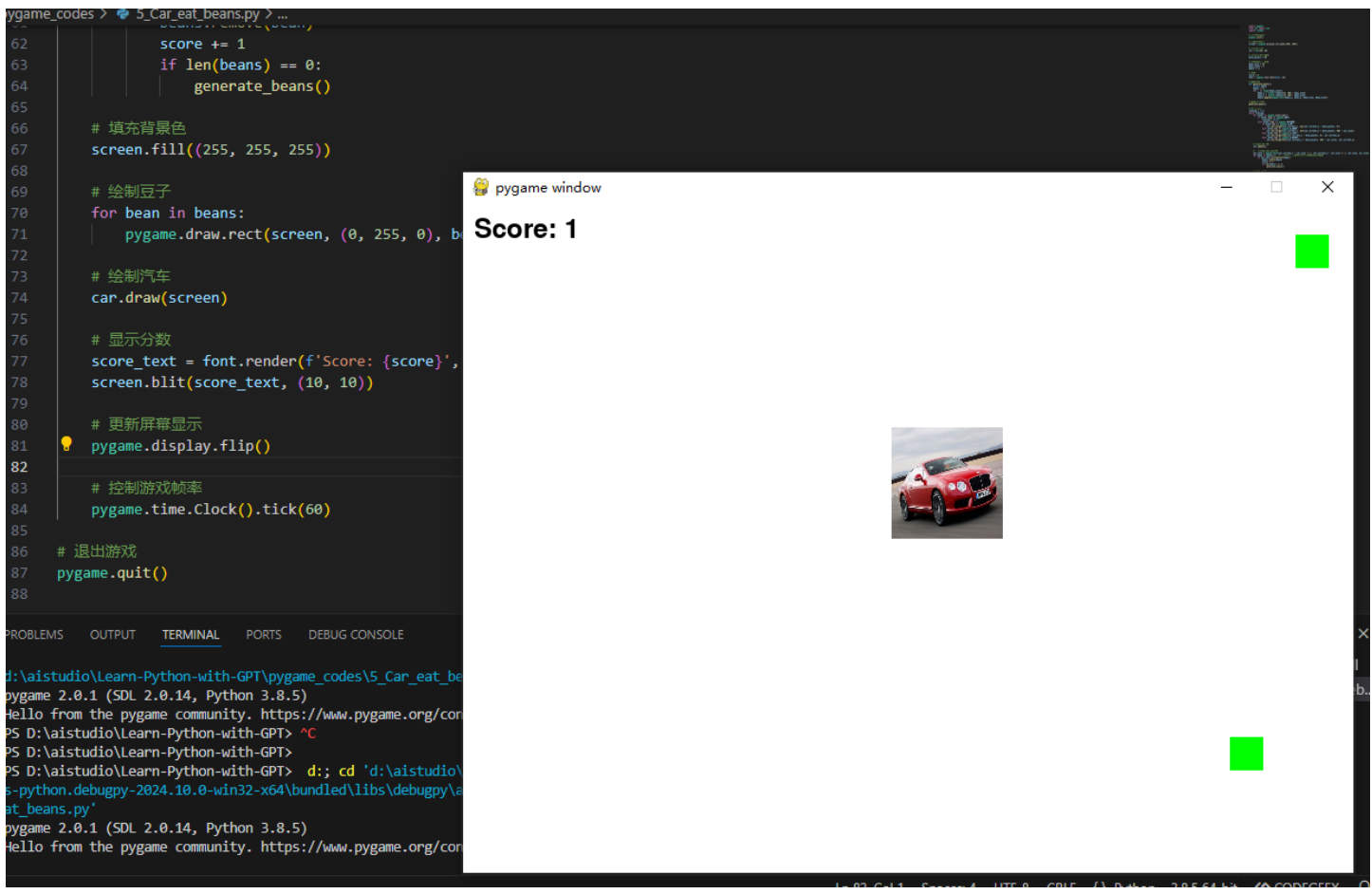
这里对应这段代码的功能解读和鼠标交互的代码十分类似，我们把对应的代码存储在lesson5/Car\_with\_keyboard.py。读者可以对自己生成的代码或者老师已经生成好的代码进行运行和

解读。

## 让Car类吃“豆子”

当然我们迄今为止只是让小汽车动了起来，我们可能需要给这个游戏增加一定的目标。这里我们举一个吃豆子的游戏的例子。

- 1 {确认刚才键盘交互的例子，或者复制5\_Car\_with\_keyboard.py中的代码，确保代码已经顺利运行}
- 2
- 3 这段代码可以顺利运行。我希望在画面中生成3个30\*30的“豆子”
- 4
- 5 当小车撞到豆子的时候，左上角的分数会加一
- 6
- 7 如果画面中所有的豆子都被撞到，则重新生成3个随机位置的豆子



- ```
1 import pygame
2 from Car import Car
3 import random
4
5 pygame.init() # 初始化pygame
```



```
6 screen = pygame.display.set_mode((800, 600)) # 设置屏幕大小
7 car = Car(50, 50) # 创建Car实例
8 move_pixels = 50 # 定义移动的像素数
9 bean_size = 30 # 豆子的大小
10 bean_count = 3 # 豆子的数量
11 beans = [] # 初始化豆子列表
12 score = 0 # 初始化分数
13 font = pygame.font.Font(None, 36) # 设置分数显示的字体
14
15 def generate_beans(): # 定义生成豆子的函数
16     global beans
17     beans = []
18     for _ in range(bean_count):
19         bean_x = random.randint(0, 800 - bean_size)
20         bean_y = random.randint(0, 600 - bean_size)
21         beans.append(pygame.Rect(bean_x, bean_y, bean_size, bean_size))
22
23 generate_beans() # 生成初始豆子
24 running = True # 游戏主循环标志
25 while running:
26     for event in pygame.event.get():
27         if event.type == pygame.QUIT: # 检测退出事件
28             running = False
29         elif event.type == pygame.KEYDOWN: # 检测键盘按下事件
30             if event.key == pygame.K_UP:
31                 car.set_target(car.current_x, max(car.current_y - move_pixels,
32 0))
33             elif event.key == pygame.K_DOWN:
34                 car.set_target(car.current_x, min(car.current_y + move_pixels,
35 600 - car.size))
36             elif event.key == pygame.K_LEFT:
37                 car.set_target(max(car.current_x - move_pixels, 0),
38 car.current_y)
39             elif event.key == pygame.K_RIGHT:
40                 car.set_target(min(car.current_x + move_pixels, 800 -
41 car.size), car.current_y)
42
43 car.update() # 更新汽车位置
44
45 car_rect = pygame.Rect(car.current_x - car.size // 2, car.current_y -
46 car.size // 2, car.size, car.size) # 获取汽车矩形区域
47 for bean in beans[:]: # 遍历豆子列表
48     if car_rect.colliderect(bean): # 检测汽车是否撞到豆子
49         beans.remove(bean) # 移除被撞到的豆子
50         score += 1 # 增加分数
51     if len(beans) == 0: # 如果所有豆子都被撞到
52         generate_beans() # 重新生成豆子
```

```
48
49     screen.fill((255, 255, 255)) # 填充背景色
50     for bean in beans: # 绘制豆子
51         pygame.draw.rect(screen, (0, 255, 0), bean)
52     car.draw(screen) # 绘制汽车
53     score_text = font.render(f'Score: {score}', True, (0, 0, 0)) # 创建分数文本
54     screen.blit(score_text, (10, 10)) # 显示分数
55     pygame.display.flip() # 更新屏幕显示
56     pygame.time.Clock().tick(60) # 控制游戏帧率
57 pygame.quit() # 退出游戏
```

可以看到，目前不计算空行的话，代码行数已经超过了50行。如果算上Car类的实现，整体的代码行数要接近100行了。所以在这一章，我们使用了一种“分模块开发”的方法，或者说是一种“自底向上”的开发方法，我们先去实现了Car类，然后再让GPT去调用这个Car类，形成了整个游戏。如果同学们想要开发更大型的程序，这种分模组实现的方法是非常实用的。

## 总结

在这一章我们实践了Python中面向对象类的实现，并且对于pygame中的小汽车这个物体进行抽象。并且使用了先让GPT生成小汽车类的代码，再基于这个类进行代码开发的开发流程。我们将更多可以深入改进的方向放在课后练习中，同学们可以在这些练习中，进一步体会面向对象在编程中的意义。

## 课后练习

### 增加小车的轨迹

尝试在鼠标小车交互的游戏中，增加小车的历史轨迹显示。

### 向对方行走的小车

在正方形的四个角上放置四个点，让这四个点都向着顺时针方向的下一辆车行走，直到相遇。小车经过的轨迹是什么样的呢？能否用我们之前实现的Car类来实现这一点？

### 吃豆子的持续改进

我们可以持续改进之前的吃豆子类

- 交互上的改进，按下一次方向键小车只会动一下，能否让小车更连续地运动
- 有的时候随机生成的豆子会在小车的旁边，甚至直接被小车吃掉，这一点可以改进吗？
- 吃到豆子的时候，没有动画和声音，这一点可以改进吗？
- \*可以给场地增加一些遮挡的障碍物，使得小车必须绕过这些障碍物才能吃到豆子。

# 坦克大战

和GPT讨论一个pygame版本的坦克大战中，玩家坦克、AI坦克和子弹的类分别要如何实现。

写一个基础的（没有地图的）坦克大战。有一些地方坦克，玩家可以操控坦克进行射击摧毁敌方坦克。

[目 第一课-钢琴键盘](#)

[目 第二课 循环语句](#)

[目 第三课 点菜程序与条件语句](#)

[目 第四课 在程序中调用大模型](#)

[目 第五课 吃豆子](#)

[目 第六课 文字搜索和RAG](#)

[目 第七课 背单词软件](#)