Pygame 从零到实战完整教程（适合初学者）

Pygame 是一个基于 Python 的游戏开发库，适合用来学习游戏开发、图形编程以及事件驱动思想。它封装了 SDL，提供了窗口、图像、音频、键盘鼠标、碰撞检测等功能，非常适合入门。

本教程将 **从零开始**，一步步讲解 Pygame 的使用方式，最终完成一个可玩的 2D 游戏示例。即使你从未接触过游戏开发，也可以顺利跟着做完。

一、准备工作

1.1 环境要求

- Python 3.8 或更高版本（推荐 3.10+）
- Windows / Linux / macOS 均可
- 基础 Python 语法（变量、函数、if、while）

1.2 安装 Pygame

打开终端或命令提示符：

```
pip install pygame
```

验证是否安装成功：

```
import pygame
print(pygame.ver)
```

如果没有报错，说明安装成功。

二、Pygame 基本结构

2.1 一个最小的 Pygame 程序

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("My First Game")

clock = pygame.time.Clock()

while True:
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    screen.fill((0, 0, 0))
    pygame.display.flip()
    clock.tick(60)
```

2.2 程序结构解析

- `pygame.init()`：初始化所有模块
- `display.set_mode()`：创建窗口
- `event.get()`：处理事件（键盘、鼠标、关闭）
- `display.flip()`：刷新画面
- `clock.tick(FPS)`：控制帧率

核心思想：游戏 = 无限循环 + 不断刷新画面

三、颜色、坐标与绘图

3.1 坐标系统

- 左上角是 `(0, 0)`
- 向右是 X 正方向
- 向下是 Y 正方向

3.2 常用颜色

```
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED   = (255, 0, 0)
```

3.3 绘制图形

```
pygame.draw.rect(screen, RED, (100, 100, 200, 100))
pygame.draw.circle(screen, WHITE, (400, 300), 50)
pygame.draw.line(screen, WHITE, (0, 0), (800, 600), 3)
```

四、键盘与鼠标控制

4.1 键盘检测（持续按键）

```
keys = pygame.key.get_pressed()
if keys[pygame.K_w]:
    print("W pressed")
```

4.2 事件方式检测

```
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_SPACE:
        print("Space pressed")
```

4.3 鼠标操作

```
mouse_pos = pygame.mouse.get_pos()
mouse_pressed = pygame.mouse.get_pressed()
```

五、图片与精灵（Sprite）

5.1 加载图片

```
player_img = pygame.image.load("player.png").convert_alpha()
screen.blit(player_img, (100, 100))
```

5.2 使用 Sprite 类（推荐）

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((50, 50))
        self.image.fill((0, 255, 0))
        self.rect = self.image.get_rect(center=(400, 300))

    def update(self):
        self.rect.x += 3
```

```
all_sprites = pygame.sprite.Group()
player = Player()
all_sprites.add(player)
```

六、碰撞检测

6.1 Rect 碰撞

```
if player.rect.colliderect(enemy.rect):
    print("碰撞了")
```

6.2 Sprite 碰撞

```
pygame.sprite.spritecollide(player, enemies, True)
```

七、文字与字体

```
font = pygame.font.SysFont("SimHei", 36)
text = font.render("Score: 10", True, (255, 255, 255))
screen.blit(text, (10, 10))
```

八、音效与背景音乐

8.1 播放音效

```
shoot_sound = pygame.mixer.Sound("shoot.wav")
shoot_sound.play()
```

8.2 背景音乐

```
pygame.mixer.music.load("bgm.mp3")
pygame.mixer.music.play(-1)
```

九、实战项目：简单躲避游戏

9.1 游戏目标

- 玩家控制方块移动
- 敌人从上方掉落
- 碰到敌人游戏结束

9.2 核心逻辑

- 玩家移动

- 敌人生成
- 碰撞检测
- 分数系统

（此部分可拆分为多个模块文件进行扩展）

十、代码结构建议

```
game/
├── main.py
├── settings.py
├── player.py
├── enemy.py
├── assets/
│   ├── images/
│   └── sounds/
```

良好的结构有助于后期维护和扩展。

十一、性能与优化建议

- 使用 `convert()` / `convert_alpha()`
- 合理控制 FPS
- 使用 Sprite Group 批量更新
- 避免在循环中频繁加载资源

十二、学习路线建议

1. 熟悉基础 API
2. 多做小 Demo（贪吃蛇、打砖块）
3. 学习面向对象设计
4. 了解物理、动画、状态机
5. 尝试完整独立游戏

结语

Pygame 并不是最强大的游戏引擎，但它 **足够简单、足够直观、足够适合学习**。通过 Pygame，你可以真正理解：

- 游戏循环是如何工作的
- 键盘鼠标事件如何驱动逻辑
- 图形、碰撞、状态管理的本质

当你完成第一个 Pygame 游戏时，你已经迈出了成为游戏开发者的重要一步。

祝你编码愉快 🎮