火眼金睛(二)

1.探索新知

1.1 配置方格颜色

上节课已经准备好了随机基础色和轻微异色，接下来我们给方格配置颜色。

分下面三步完成：

1. 存储生成的基础色和轻微异色
2. 随机选择一个方格设置异色
3. 给方格矩阵配置颜色

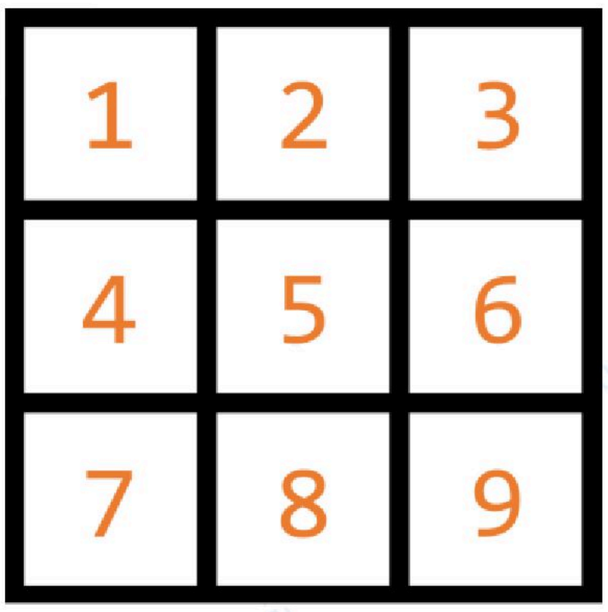
【存储生成的基础色和轻微异色】

```
grids = []     # 创建列表grids，后面存储每个方格信息

# 获取基础色和异色
bc = base_color()
dc = dif_color(bc)

for row in range(n):    # 遍历行
    for col in range(n):     # 遍历列
        ...
```

【随机选择一个方格设置异色】

方格没有名字，怎么指定某个方格设定为异色呢？

1.1 配置方格颜色



有了编号，想要随机选择一个方格，就容易多了

异色方格编号 = random.randint(1,9)

【随机选择一个方格设置异色】

用 i 给方格编号，从 1 到 n*n

```
# 获取基础色和异色
bc = base_color()
dc = dif_color(bc)
# 记录方格编号
i = 1
dif_index = random.randint(1, n * n)    # 随机选一个方格设置轻微异色
for row in range(n):  # 遍历行
    for col in range(n):  # 遍历列
        if i == dif_index:
            color = dc
        else:
            color = bc
        info = {
            'pos': (space + col * (side + space), space + row * (side + space)),
            'color': color
            }
        grids.append(info)
        i += 1
```

1.1 配置方格颜色

【给方格矩阵配置颜色】

```
def draw():
    screen.blit('火眼金睛.png',(0,0))
    for grid in grids:
        screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])
```

基础色和异色已成功配置，运行看看效果吧！

拓展--三元表达式

```
# 记录方格编号
i = 1
# 随机选一个方格设置轻微异色
dif_index = random.randint(1, n * n)
for row in range(n):    # 遍历行
    for col in range(n):    # 遍历列
        if i == dif_index:
            color = dc
        else:
            color = bc
        info = {
            'pos': (space + col * (side + space), space + row * (side + space)),
            'color': color
            }
        grids.append(info)
        i += 1
```

再教大家用一行代码来代替这部分内容，

新知识 —— 三元表达式

- **定义**

Python的三元表达式，也叫条件表达式，是一种简洁高效的编写条件逻辑的方式。可以在一行代码中写出一个if-else条件语句。

拓展--三元表达式

- **格式**

```
值1 if 条件表达式 else 值2
```

等同于

```
if 条件表达式:
    值1
else:
    值2
```

通过下面的例子，对比感受一下：

```
# 常规的 if-else 语句
x = 10
y = 20
if x > y:
    result = x
else:
    result = y
print(result)
```

```
# 三元表达式
x = 10
y = 20
print(x if x > y else y)
```

自己试着用三元表达式简化这段代码吧！

1.1 配置方格颜色

新增代码：

```
def draw():
    screen.blit('火眼金睛.png',(0,0))
    for grid in grids:
        screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])

...


# 创建列表grids，后面存储每个方格信息
grids = []
# 获取基础色和异色
bc = base_color()
dc = dif_color(bc)
# 记录方格编号
i = 1
# 随机选择一个位置的方格设置略微不同的颜色
dif_index = random.randint(1, n * n)
for row in range(n):    # 遍历行
    for col in range(n):    # 遍历列
        info = {
            'pos': (space + col * (side + space), space + row * (side + space)),
            'color': dc if i == dif_index else bc
            }
        grids.append(info)
        i += 1
```

1.2 鼠标点击选择

按下鼠标后，要确认是否点到了异色的方格，也就是判断下面两项：

1. 方格（矩形对象）碰到了鼠标
2. 方格为异色

还记得怎么检测角色与指定点的碰撞吗？
换成矩形对象又该如何检测呢？

```
actor.collidepoint((x, y))
```

→

```
矩形对象.collidepoint((x, y))
```

grid = {
'pos': 方格坐标,
'color': 方格颜色
}

```
def on_mouse_down(pos):
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        # 点击了异色的方格
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            # <进入下一关>
```

前面我们已经写过加载 3*3 矩阵方格的代码。只需要把加载方格矩阵的代码封装到函数game()中，每次切换到新关卡时调用即可。

```
# 初始化游戏参数
n = 3    # 初始为3×3的矩阵

# 根据格子数量计算格子边长和间距
side = 700// n * 0.9
space = 700// n * 0.1
...
for row in range(n):
    for col in range(n):
        ...
        i += 1
```

game()

```
n = 3    # 初始为3×3的矩阵
game()
```

思考：

哪些变量需要声明为全局变量？

1.2 鼠标点击选择

```
def game():
    global grids, side, dc
    ...

def on_mouse_down(pos):
    global n
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            n += 1  # 进入下一关
            game()

# 初始化游戏参数
n = 3   # 初始为3×3的矩阵
game()
```

新增代码：

```
# (修改)封装为game()函数
def game():
    global grids, side, dc   # (新增)声明为全局变量
    # 根据格子数量计算格子边长和间距
    side = 700// n * 0.9
    space = 700// n * 0.1
    # 创建列表grids，后面存储每个方格信息
    grids = []
    # 获取基础色和异色
    bc = base_color()
    dc = dif_color(bc)
    # 记录方格编号
    i = 1
    # 随机选择一个位置的方格设置略微不同的颜色
    dif_index = random.randint(1, n * n)
```

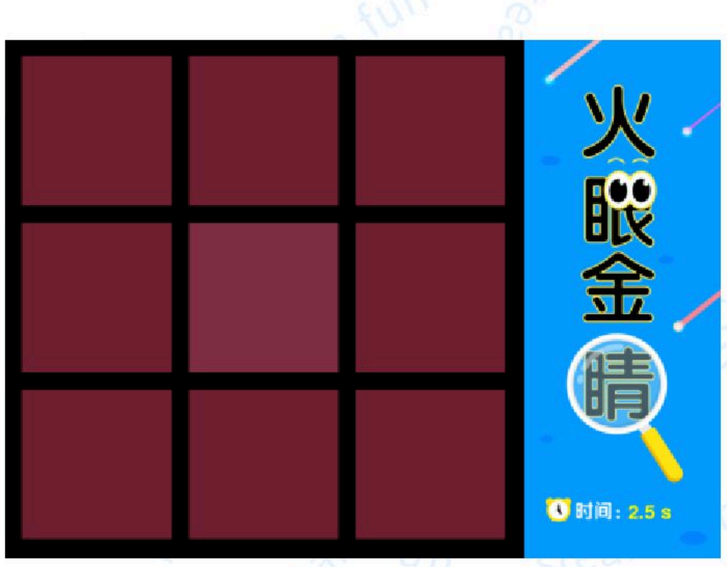
1.2 鼠标点击选择

```
    for row in range(n):
        for col in range(n):
            info = {
                'pos': (space+col*(side+space), space+row*(side+space)),
                'color': dc if i == dif_index else bc
                }
            grids.append(info)
            i += 1

# （新增）鼠标点击方格
def on_mouse_down(pos):
    global n
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            n += 1  # 进入下一关
            game()

n = 3    # 初始为3×3的矩阵
game()
```

截至目前，游戏的基本效果已经实现了，
先自己玩玩吧！体验一下无限关卡的乐趣

1.3 游戏胜利效果

如果不设置停止的条件，游戏将一直进行下去。为了游戏的完整性，我们可以给游戏设置有限的关卡，示例以 10*10 的方形矩阵为最后一关



n <= 10

win = False

n > 10

win = True

我们可以设置一个变量 win 来记录游戏的状态，设置初始值为 False

```
def on_mouse_down(pos):
    global n, win
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            if n > 10:
                win = True
            else:
                n += 1  # 进入下一关
                game()
```

接着我们根据游戏的状态 win ，来绘制胜利的背景

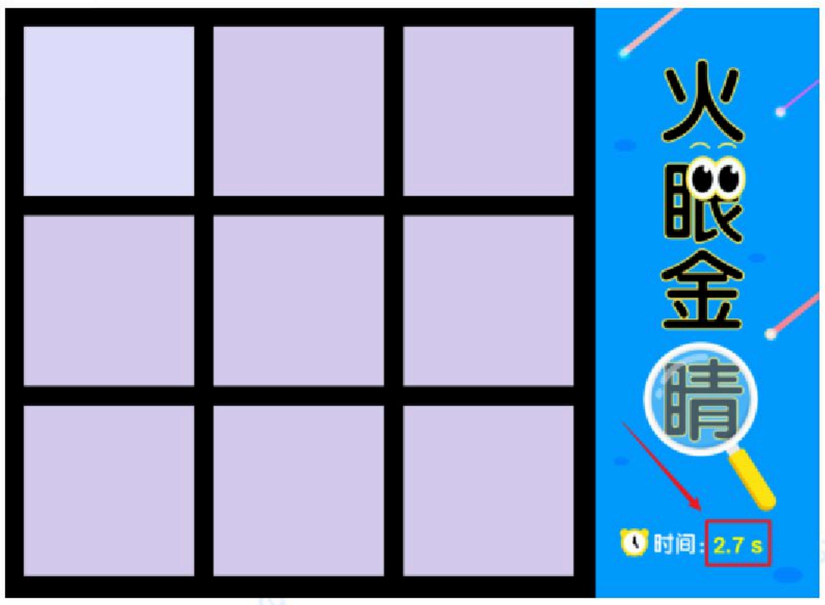
```
def draw():
    screen.blit('火眼金睛.png',(0,0))
    if win:
        screen.blit('游戏成功.png', (0,0))
    else:
        for grid in grids:
            screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])
```

1.3 游戏胜利效果

新增代码：

```
...
def draw():
    screen.blit('火眼金睛.png',(0,0))
    if win:
        screen.blit('游戏成功.png', (0,0))
    else:
        for grid in grids:
            screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])
...

def on_mouse_down(pos):
    global n, win
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            if n > 10:
                win = True
            else:
                n += 1  # 进入下一关
                game()
# 初始化游戏参数
n = 3   # 初始为3×3的矩阵
win = False
game()
pgzrun.go()
```

1.4 计时效果



游戏进行中

游戏胜利后

还记得如何绘制文本吗？

`screen.draw.text(text, [pos, ]**kwargs)`

- text文字内容、pos位置 这两个参数必须要有
- 可以设置其它属性，如：颜色、字体、字号、阴影等

center = 位置坐标 表示文本中心的坐标

```
screen.draw.text( 文字内容 , center= 位置坐标 , ...)
```

我们需要准备下面俩参数的内容：

1. 文本内容 text
2. 位置坐标 pos

【文本内容】

游戏过程页面，呈现的是**游戏耗时**，设为变量t

导入time库，获取游戏开始时的时间为 start_t ，

t = 现在时间 - 开始时间，t值随着游戏的进行不停变化

time.time() start_t

1.4 计时效果

【文本内容】

我们可以将 t 值的计算过程放在update()函数中，每秒钟调用 60 次。
文字内容是 `f"{t:.1f} s"`

```
import time
...
def update():
    """更新函数，这里主要更新总耗时"""
    global t
    t = time.time() - start_t
...
n = 3    # 初始为3×3的矩阵
win = False  # 记录游戏成功状态
start_t = time.time()     #记录开始时间
game()
```

【位置坐标】

如何获取指定点的坐标？

```
def on_mouse_down(pos):
    print(pos)
    ...
```

经测试后，建议坐标如下：
游戏过程页面，`(900, 662)`

```
def draw():
    ...
    if win:
        ...
    else:
        screen.draw.text(f"{t:.1f} s", center=(900, 662), fontsize=40, color="yellow")
```

1.4 计时效果

【文本内容】

游戏结束页面，呈现的是**游戏总耗时**，记录的是游戏win的**那一刻**的 t 值，我们用变量 total_t 来记录。文字内容是 f"{total_t:.1f}"

```
def on_mouse_down(pos):
    global n, win, total_t
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            if n > 10:
                win = True
                total_t = t
            else:
                n += 1  # 进入下一关
                game()
```

【位置坐标】

如何获取指定点的坐标？

```
def on_mouse_down(pos):
    print(pos)
    ...
```

经测试后，建议坐标如下：

游戏结束页面，(520, 570)

最后记得删掉测试用的print(pos)语句

新增代码：

```
def draw():
    screen.blit('火眼金睛.png',(0,0))
    if win:
        screen.blit('游戏成功.png', (0,0))
        screen.draw.text(f"{total_t:.1f}", center=(520, 570), fontsize=90, color="yellow")
    else:
        screen.draw.text(f"{t:.1f} s", center=(900, 662), fontsize=40, color="yellow")
        for grid in grids:
            screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])
```

完整代码

```
import pgzrun
import random
import time

# 设置窗口
WIDTH = 1000  # 窗口的宽度
HEIGHT = 725  # 窗口的高度

def draw():
    screen.blit('火眼金睛.png',(0,0))
    if win:
        screen.blit('游戏成功.png', (0,0))
        screen.draw.text(f"{total_t:.1f}", center=(520, 570), fontsize=90, color="yellow")
    else:
        screen.draw.text(f"{t:.1f} s", center=(900, 662), fontsize=40, color="yellow")
        for grid in grids:
            screen.draw.filled_rect(Rect(grid['pos'], (side, side)), grid['color'])

def update():
    """更新函数，用于实时更新游戏状态相关信息，这里主要更新总耗时"""
    global t
    t = time.time() - start_t

def base_color():
    """生成随机基础颜色"""
    return (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

def dif_color(base):
    """
    生成轻微异色
    参数:
    base: 表示基础颜色的元组，格式为（r,g,b）
    """
    r = min(base[0] + random.randint(20, 30), 255)
    g = min(base[1] + random.randint(20, 30), 255)
    b = min(base[2] + random.randint(20, 30), 255)
    return (r, g, b)

def game():
    global grids, side, dc
    # 根据格子数量计算格子边长和间距
    side = 700// n * 0.9
    space = 700// n * 0.1
```

完整代码

```
    # 创建列表grids，后面存储每个方格信息
    grids = []
    # 获取基础色和异色
    bc = base_color()
    dc = dif_color(bc)
    # 记录方格编号
    i = 1
    # 随机选择一个位置的方格设置略微不同的颜色
    dif_index = random.randint(1, n * n)
    for row in range(n):
        for col in range(n):
            info = {
                'pos': (space + col * (side + space),
                        space + row * (side + space)),
                'color': dc if i == dif_index else bc
                }
            grids.append(info)
            i += 1

def on_mouse_down(pos):
    global n, win, total_t
    for grid in grids:
        grid_rect = Rect(grid['pos'], (side, side))
        if grid_rect.collidepoint(pos) and grid['color'] == dc:
            if n > 10:
                win = True
                total_t = t
            else:
                n += 1  # 进入下一关
                game()

# 初始化游戏参数
n = 3    # 初始为3×3的矩阵
win = False  # 记录游戏成功状态
start_t = time.time()     #记录开始时间
game()

# 启动游戏
pgzrun.go()
```

2.强化练习

1．运行下面代码段，输出的结果是（　　）？

A. 5 11

B. 5 9

C. 11 11

D. 11 9

```
def reverse(b, c):
    global a
    a = c
    c = b
    b = a
a, b, c = 5, 9, 11
reverse(b, c)
print(a, c)
```

2．运行下列程序，输出的结果是？（　　）

A. 3

B. 6

C. 9

D. 0

```
s = 0
def f(n):
    global s
    for i in range(n):
        s = s + i
    return s
print(f(f(3)))
```

3．执行下列语句，得到的结果是？（　　）

A. 0.3

B. 0.33

C. 0.333

D. 00.333

```
print('{0:.3f}'.format(0.333333333333))
```

2.强化练习

4. 请将给出的三元表达式改写为常规的 if-else 语句

三元表达式：

```
# 定义两个变量
num1 = 10
num2 = 20
# 找出两个数中较大的数
result = num1 if num1 > num2 else num2
print(result)
```

常规的 if-else 语句：

5. 请补全红线处代码，绘制出右图所示方格矩阵

```
...
    i = 0
    for row in range(n):
        for col in range(n):
            info = {
                'pos': (space + col * (side + space),
                        space + row * (side + space)),
                'color': dc if i ==  ____  else bc
                }
            grids.append(info)
            i += 1
...
```

3.术语箱

index	索引
center	中心，中心点
total	全部的，总共的

4.课后挑战

作品优化 —— 增大游戏难度

提示：

1．增加关卡数量

2．减小基础色和异色的区别