中国地图我来拼(二)

1.探索新知

1.1 检测碎片是否正确放置

首先，所有碎片的正确位置，程序中已经预置好了，放在字典a中。

```
# 预置内容，请勿改动
a = {"01新疆碎片": (225, 231), "02西藏碎片": (244, 457), "03内蒙古碎片": (623, 203),
     "04青海碎片": (383, 396), "05四川碎片": (491, 521), "06黑龙江碎片": (861, 122),
     "07甘肃碎片": (472, 358), "08云南碎片": (457, 621), "09广西碎片": (580, 656),
     "10湖南碎片": (640, 584), "11陕西碎片": (586, 415), "12河北碎片": (718, 330),
     "13吉林碎片": (859, 228), "14湖北碎片": (654, 506), "15广东碎片": (678, 677),
     "16贵州碎片": (549, 593), "17河南碎片": (672, 446), "18江西碎片": (724, 584),
     "19山东碎片": (758, 392), "20山西碎片": (650, 369), "21辽宁碎片": (805, 284),
     "22安徽碎片": (740, 487), "23福建碎片": (766, 609), "24江苏碎片": (770, 462),
     "25浙江碎片": (793, 538), "26重庆碎片": (573, 525), "27宁夏碎片": (548, 375),
     "28台湾碎片": (824, 652), "29海南碎片": (610, 753), "30北京碎片": (715, 310),
     "31天津碎片": (730, 326), "32上海碎片": (818, 495)}    # 省份文件名称和对应的正确位置
```

鼠标释放的瞬间，先判断是否带着碎片，也就是判断是否有碎片被选中，即

```
def on_mouse_up(pos):
    if s_piece != None:
```

如果有选中的碎片，那么就用变量 c_pos 记录碎片应该放置的正确位置，通过被选中的碎片名称 s_name 就可以从字典 a 中找到对应的正确位置

```
        c_pos = a[s_name]
```

接下来我们要做的，就是对比碎片当前位置与正确位置的距离。
想要做到严丝合缝移到某个位置太困难了，所以我们设置到正确位置的 x方向距离 和 y方向距离 均在50以内时，就把碎片放置到正确位置。

1.1 检测碎片是否正确放置

abs() 函数
求数值的绝对值

x方向距离：`abs(pos[0] - c_pos[0])`
y方向距离：`abs(pos[1] - c_pos[1])`

碎片放到正确位置后，需要将 pieces 中存储的【**碎片放置状态**】，从 **False** 改为 **True** ，将该碎片标记为已正确放置

```
def on_mouse_up(pos):
    if s_piece != None:  # 如果有选中的碎片
        c_pos = a[s_name]  # 选中碎片的正确位置
        # 检查当前位置与正确位置的距离
        if abs(pos[0] - c_pos[0]) < 50 and abs(pos[1] - c_pos[1]) < 50:
            s_piece.pos = c_pos  # 放置到正确位置
            for piece in pieces:
                if piece[0] == s_piece and not piece[2]:
                    piece[2] = True  # 标记为已正确放置
```

设置全局变量 count 记录正确放置碎片的数量，初始值为 0 。每次正确放置碎片后，让计数器 count 加 1 ，要在函数内部修改这个变量的值，所以需要使用 global关键字来声明 count 为全局变量。

```
count = 0   # 记录游戏中正确放置的碎片数量
def on_mouse_up(pos):
    global count
    if s_piece != None:  # 如果有选中的碎片
        ...
        if abs(pos[0]-c_pos [0]) < 50 and abs(pos[1]-c_pos [1]) < 50:
            ...
            for piece in pieces:
                if piece[0] == s_piece and not piece[2]:
                    ...
                    count += 1  # 正确放置的计数器加一
                    break  # 找到选中的碎片，标记完成，不继续遍历
```

1.1 检测碎片是否正确放置

检查完选中碎片的位置后，我们将变量 **s_piece** 初始化为 None ，等待鼠标下次选中碎片。不要忘了声明 **s_piece** 为全局变量。

```
def on_mouse_up(pos):
    global count, s_piece
    if s_piece != None:  # 如果有选中的碎片
        c_pos = a[s_name]
        # 检查当前位置与正确位置的距离
        ...
        s_piece = None  # 清除当前选中的碎片
```

新增代码：

```
count = 0    # （新增）记录游戏中正确放置的碎片数量
# （新增）鼠标释放事件
def on_mouse_up(pos):
    global count, s_piece
    if s_piece != None:  # 如果有选中的碎片
        c_pos = a[s_name]
        # 检查当前位置与正确位置的距离
        if abs(pos[0]-c_pos[0]) < 50 and abs(pos[1]-c_pos[1]) < 50:
            s_piece.pos = c_pos
            for piece in pieces:
                if piece[0] == s_piece and not piece[2]:
                    piece[2] = True  # 标记为已正确放置
                    count += 1    # 正确放置的计数器加一
                    break    # 找到了选中的碎片，标记完成
    s_piece = None  # 清除当前选中的碎片
```

1.2 设置倒计时效果

拼图的基本效果已经实现，接下来我们对细节进行丰富，首先给游戏添加倒计时效果

首先，我们先设置一个 **time** 变量记录剩余时间，将游戏设定为120秒

```
time = 120  # 剩余时间
```

怎么在窗口绘制倒计时的文字呢？

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

- text文字内容、pos位置 这两个参数必须要有
- 锚点位置默认为文字的左上角
  以右图为例，锚点为红点所示位置

- fontsize：字体大小（以像素为单位）。默认值为 ：24
- color：文字的前景色。默认值为：white
- fontname：字体文件名，需放在同目录的fonts文件夹内。默认使用系统字体

在屏幕上显示倒计时，使用 **screen.draw.text()** 函数，位置、字号、颜色都一并设置好。


```
def draw():
    ...
    # 显示剩余时间
    screen.draw.text(f"Time: {time}", (40, 20), fontsize=40, color="white")
```

1.2 设置倒计时效果

想要实现每隔一秒， time 减少1，我们可以编写以下的程序，这是一个通用的倒计时模板

```
# 时间更新函数
def update_time():
    global time
    # 更新剩余时间
    if time > 0:
        time -= 1
# 每隔 1 秒调用更新时间的函数
clock.schedule_interval(update_time, 1)
```

这里千万不要加括号哦！

clock.schedule_interval(callback, delay)

- 参数 **callback** ，就是回调函数function，代表要调用的函数，这个函数不可以有参数
- **delay** 代表间隔的秒数

新增代码：

```
# 初始值设置
time = 120  # （新增）剩余时间
...
# 刷新屏幕
def draw():
    screen.blit('中国地图背景', (0,0))     # 绘制背景
    # 绘制每个碎片
    for piece in pieces:
        piece[0].draw()
    # （新增）显示剩余时间
    screen.draw.text(f"Time: {time}", (40, 20), fontsize=40, color="white")
...
```

1.2 设置倒计时效果

```
#  （新增）时间更新函数
def update_time():
    global time
    # 更新剩余时间
    if time > 0:
        time -= 1  # 剩余时间减 1

#  （新增）每隔 1 秒调用更新时间的函数
clock.schedule_interval(update_time, 1)
```

1.3 胜利失败效果

拼图的基本效果实现了，现在我们还需要给游戏添加胜利和失败的效果。

失败

失败条件：倒计时结束

`time == 0`

成功

胜利条件：拼完所有碎片

`count == len(a)`

1.3 胜利失败效果

首先，我们提前创建好游戏结束的画面角色，把它们放在窗口的中心。

```
# 加载碎片
...
# 加载游戏结束画面
win_img = Actor("成功", (WIDTH // 2, HEIGHT // 2))
lose_img = Actor("失败", (WIDTH // 2, HEIGHT // 2))
```

接着，我们在draw()函数中，添加绘制成功失败角色的代码

```
# 刷新屏幕
def draw():
    ...
    # 如果游戏结束，显示胜利或失败的图片
    if time == 0:
        lose_img.draw()
    if count == len(a):
        win_img.draw()
```

新增代码：

```
# 加载碎片
...
# （新增）加载游戏结束画面
win_img = Actor("成功", (WIDTH // 2, HEIGHT // 2))
lose_img = Actor("失败", (WIDTH // 2, HEIGHT // 2))
# 刷新屏幕
def draw():
    ...
    # （新增）如果游戏结束，显示胜利或失败的图片
    if time == 0:
        lose_img.draw()
    if count == len(a):
        win_img.draw()
```

1.4 效果优化

游戏失败后，碎片还可以被拖动，不符合游戏的正常逻辑。
那我们如何优化代码，使得游戏失败后，碎片不能再被拖动呢？

我们可以设置一个变量 game_over 记录游戏的状态。初始值设置为False，时间耗尽时将 game_over 设置为True。

```
game_over = False     # 标志游戏是否结束

def update_time():
    global time, game_over
    if time > 0:  # 倒计时
        time -= 1
    else:    # 倒计时为0，游戏结束
        game_over = True
```

鼠标按下事件中，判断游戏没有结束时，才会去检测是否点击到需要拼图的碎片

```
# 鼠标按下事件
def on_mouse_down(pos):
    global s_piece, s_name
    if not game_over:
        for piece, name, placed in pieces:
            # 是否点击没拼好的碎片
            if piece.collidepoint(pos) and not placed:
                s_piece = piece  # 记录选择的碎片
                s_name = name  # 记录碎片名称
                break
```

1.4 效果优化

拼完了所有的拼图，游戏胜利后，倒计时依然在减少。这是为什么呢？

分析代码可以发现，我们判断倒计时为0后，游戏结束，出现失败画面，而其他时候倒计时始终在减少

```
if time > 0:
    time -= 1
else:
    game_over = True
```

那我们是否可以在这里加一个判断条件，如果倒计时不为0，那么我再来判断 count < len(a) 是否成立，如果成立，也就是地图还没有拼完，那么倒计时再减少。这样的话，如果游戏胜利，倒计时将不再减少。

```
if time > 0:  # 倒计时未结束
    if count < len(a): # 仍有未拼完的碎片
        time -= 1
else:  # 倒计时结束
    game_over = True
```

至此，我们的拼图游戏就写好啦，快来比比谁能最快拼完吧！

1.4 效果优化

新增代码：

```
# 初始值设置
game_over = False      # （新增）标志游戏是否结束
...
# （修改）鼠标按下事件
def on_mouse_down(pos):
    global s_piece, s_name
    if not game_over:    # （新增）判断游戏状态，游戏中才能按下
        for piece, name, placed in pieces:
            # 检查是否点击到未正确放置的碎片
            if piece.collidepoint(pos) and not placed:
                s = piece  # 记录选择的碎片
                s_name = name  # 记录碎片名称
                break
...
# （修改）更新剩余时间
def update_time():
    global time, game_over
    if time > 0:
        if count < len(a):      # （新增）判断游戏是否胜利
            time -= 1
    else:     # （新增）修改游戏状态
        game_over = True  # 游戏结束
```

完整代码

```
# 预置内容，请勿改动
a = {"01新疆碎片": (225, 231), "02西藏碎片": (244, 457), "03内蒙古碎片": (623, 203),
     "04青海碎片": (383, 396), "05四川碎片": (491, 521), "06黑龙江碎片": (861, 122),
     "07甘肃碎片": (472, 358), "08云南碎片": (457, 621), "09广西碎片": (580, 656),
     "10湖南碎片": (640, 584), "11陕西碎片": (586, 415), "12河北碎片": (718, 330),
     "13吉林碎片": (859, 228), "14湖北碎片": (654, 506), "15广东碎片": (678, 677),
     "16贵州碎片": (549, 593), "17河南碎片": (672, 446), "18江西碎片": (724, 584),
     "19山东碎片": (758, 392), "20山西碎片": (650, 369), "21辽宁碎片": (805, 284),
     "22安徽碎片": (740, 487), "23福建碎片": (766, 609), "24江苏碎片": (770, 462),
     "25浙江碎片": (793, 538), "26重庆碎片": (573, 525), "27宁夏碎片": (548, 375),
     "28台湾碎片": (824, 652), "29海南碎片": (610, 753), "30北京碎片": (715, 310),
     "31天津碎片": (730, 326), "32上海碎片": (818, 495)}    # 省份文件名称和对应的正确位置

import pgzrun
import os
import random  # 导入操作系统和随机数库

# 设置窗口
WIDTH = 1000
HEIGHT = 800
TITLE = '中国地图我来拼'    # 设置窗口标题
# 初始值设置
s_piece = None     # 当前被选择的碎片
s_name = None     # 当前被选择的碎片名字
count = 0    # 记录游戏中正确放置的碎片数量
time = 120  # 游戏限时
game_over = False     # 标志游戏是否结束
# 加载碎片
pieces = []
# 遍历指定文件夹中的所有文件
for filename in os.listdir('images'):
    if filename[-6:] == '碎片.png':  # 只处理以 碎片.png 结尾的文件
        img = Actor(filename[:-4],(random.randint(0,WIDTH),random.randint(0,HEIGHT)))
        pieces.append([img, filename[:-4], False]) #添加碎片信息到列表


# 加载游戏结束画面
win_img = Actor("成功", (WIDTH // 2, HEIGHT // 2))
lose_img = Actor("失败", (WIDTH // 2, HEIGHT // 2))
```

完整代码

```
# 刷新屏幕
def draw():
    screen.blit('中国地图背景',(0,0))     # 绘制背景
    for piece in pieces:  # 绘制每个碎片
        piece[0].draw()
    # 显示剩余时间
    screen.draw.text(f"Time: {time}", (40, 20), fontsize=40, color="white")
    if time == 0: # 倒计时结束，游戏失败
        lose_img.draw()
    if count == len(a): # 拼好所有碎片，游戏胜利
        win_img.draw()
# 鼠标按下事件
def on_mouse_down(pos):
    global s_piece, s_name
    if not game_over:
        for piece, name, placed in pieces:
            # 检查是否点击到未正确放置的碎片
            if piece.collidepoint(pos) and not placed:
                s_piece = piece  # 记录选择的碎片
                s_name = name  # 记录碎片名称
                break
# 鼠标移动事件
def on_mouse_move(pos):
    # 如果有选中碎片且鼠标未移出窗口，就随着鼠标移动
    if s_piece != None and 0<pos[0]<1000 and 0<pos[1]<800:
        s_piece.pos = pos  # 更新选中碎片的位置
# 鼠标释放事件
def on_mouse_up(pos):
    global count, s_piece
    if s_piece != None:  # 如果有选中的碎片
        c_pos = a[s_name]  # 选中碎片的正确位置
        # 检查当前位置与正确位置的距离
        if abs(pos[0]-c_pos[0])<50 and abs(pos[1]-c_pos[1])<50:
            s_piece.pos = c_pos  # 放置到正确位置
            for piece in pieces:
                if piece[0] == s_piece and not piece[2]:
                    piece[2] = True  # 标记为已正确放置
                    count += 1  # 正确放置的计数器加一
                    break
        s_piece = None  # 清除当前选中的碎片
```

完整代码

```
# 更新剩余时间
def update_time():
    global time, game_over
    if time > 0:
        if count < len(a):
            time -= 1
    else:
        game_over = True  # 游戏结束

# 每隔 1 秒调用更新时间的函数
clock.schedule_interval(update_time, 1)

# 启动游戏
pgzrun.go()
```

2.强化练习

1. 以下哪个选项描述了abs()函数的返回值类型？（　　）

A. int

B. float

C. bool

D. 不确定，取决于输入参数类型

2. abs()是Python的内置函数，执行abs(-1.00)语句返回的结果是？（　　）

A. -1.00

B. 1

C. 1.0

D. 1.00

3. P仔做了一个计时器，下面哪个能够实现每2秒更新的效果？（　　）

A. clock.schedule_interval(update_time, 1)

B. clock.schedule_interval(update_time(), 1)

C. clock.schedule_interval(update_time, 2)

D. clock.schedule_interval(update_time(), 2)

2.强化练习

4．下面哪段代码可以实现倒计时效果？（　　）

A.

```
time = 0
def update_time():
    global time
    if time > 0:
        time += 1
clock.schedule_interval(update_time, 1)
```

B.

```
time = 120
def update_time():
    global time
    if time < 0:
        time -= 1
clock.schedule_interval(update_time, 1)
```

C.

```
time = 120
def update_time():
    global time
    if time > 0:
        time -= 1
clock.schedule_interval(update_time, 1)
```

D.

```
time = 120
def update_time():
    global time
    if time == 0:
        time -= 1
clock.schedule_interval(update_time, 1)
```

5．执行下面python代码后，输出的结果是？（　　）

A. 5#5#

B. 5#1#

C. 1#1#

D. 1#5#

```
s = 1
def sums(n):
    global s
    s = 0
    s = s + n
    print(s, end="#")
sums(5)
print(s, end="#")
```

3.术语箱

schedule	时间表，日程表	screen	屏幕
interval	间隔	fontsize	字体大小

4.课后挑战

小小地理学家

根据中国地图上的轮廓和简称，请同学们指出**23个省**、**5个自治区**、**4个直辖市**和**2个特别行政区**的名称，口述给老师或者爸爸妈妈听听吧！

小小地理学家

- **23个省：**

河北、山西、辽宁、吉林、黑龙江、江苏、浙江、安徽、福建、江西、山东、
河南、湖北、湖南、广东、海南、四川、贵州、云南、陕西、甘肃、青海、台湾

- **5个自治区：**

内蒙古自治区、广西壮族自治区、
西藏自治区、宁夏回族自治区、
新疆维吾尔自治区

- **4个直辖市：**

北京市、天津市、上海市、重庆市

- **2个特别行政区：**

香港特别行政区、澳门特别行政区