24 点小游戏说明文档

1. 项目简介与分工

1.1 项目分工

组长：吴义豪　　学号：41824334　　任务：逻辑设计与文档说明

组员：刘志欣　　学号：41824329　　任务：界面设计与美化

组员：王　彦　　学号：41824240　　任务：游戏功能测试

1.2 项目简介

项目本是棋牌类益智游戏，要求四个数字运算结果等于二十四，这个游戏用扑克牌更容易来开展。拿一副牌，抽去大小王后(初练也可以把 J/Q/K/大小王也拿去)，剩下 1～10 这 40 张牌(以下用 1 代替 A)。任意抽取 4 张牌(称为牌组)，用加、减、乘、除(可加括号，高级玩家也可用乘方开方与阶乘运算)把牌面上的数算成 24。每张牌必须用且只能用一次。如抽出的牌是 3、8、8、9，那么算式为(9-8)×8×3=24。现在利用 pygame 做一个小项目实现。

2.代码实现

2.1　24 点生成器

2.1.1 随机生成 4 个有解的数字，且范围在 1~10 之间，代码实现如下：

```
def generate(self):
        self._reset()
    while True:
        self.numbers_ori = [random.randint(1, 10) for i in range(4)]
        self.numbers_now = copy.deepcopy(self.numbers_ori)
        self.answers = self._verify()
        if self.answers:
                break
```

2.1.2 验证 4 个数字是否有解并求出所有解部分

采用的方法是递归枚举，然后一一验证是否有解，代码如下：

```
def _verify(self):
    answers = []
    for item in self._iter(self.numbers_ori, len(self.numbers_ori)):
        item_dict = []
        list(map(lambda i: item_dict.append({str(i): i}), item))
        solution1 = self._func(self._func(self._func(item_dict[0], item_dict[1]), item_dict[2]), item_dict[3])
        solution2 = self._func(self._func(item_dict[0], item_dict[1]), self._func(item_dict[2], item_dict[3]))
        solution = dict()
        solution.update(solution1)
        solution.update(solution2)
        for key, value in solution.items():
            if float(value) == self.target:
                answers.append(key)
    # 避免有数字重复时表达式重复
    answers = list(set(answers))
    return answers
'''递归枚举'''
def _iter(self, items, n):
    for idx, item in enumerate(items):
        if n == 1:
            yield [item]
        else:
            for each in self._iter(items[:idx]+items[idx+1:], n-1):
                yield [item] + each
'''计算函数'''
def _func(self, a, b):
    res = dict()
    for key1, value1 in a.items():
        for key2, value2 in b.items():
            res.update({'('+key1+'+'+key2+')': value1+value2})
            res.update({'('+key1+'-'+key2+')': value1-value2})
            res.update({'('+key2+'-'+key1+')': value2-value1})
            res.update({'('+key1+'×'+key2+')': value1*value2})
            value2 > 0 and res.update({'('+key1+'÷'+key2+')': value1/value2})
            value1 > 0 and res.update({'('+key2+'÷'+key1+')': value2/value1})
    return res
```

2.2 定义卡片类

属性值(内容、颜色、字体等)可根据用户喜好进行修改，卡片类定义后进行显示即可，具体代码实现如下：

```
class Card(pygame.sprite.Sprite):
    def __init__(self, x, y, width, height, text, font, font_colors, bg_colors, attribute, **kwargs):
        pygame.sprite.Sprite.__init__(self)
        self.rect = pygame.Rect(x, y, width, height)
        self.text = text
        self.attribute = attribute
        self.font_info = font
        self.font = pygame.font.Font(font[0], font[1])
        self.font_colors = font_colors
        self.is_selected = False
        self.select_order = None
        self.bg_colors = bg_colors
    '''画到屏幕上'''
    def draw(self, screen, mouse_pos):
        pygame.draw.rect(screen, self.bg_colors[1], self.rect, 0)
        if self.rect.collidepoint(mouse_pos):
            pygame.draw.rect(screen, self.bg_colors[0], self.rect, 0)
        font_color = self.font_colors[self.is_selected]
        text_render = self.font.render(self.text, True, font_color)
        font_size = self.font.size(self.text)
        screen.blit(text_render, (self.rect.x+(self.rect.width-font_size[0])/2,
                                  self.rect.y+(self.rect.height-font_size[1])/2))
```

2.3 定义按钮类

按钮类的定义类似于卡片，不同在于用户点击按钮时需要根据该按钮的功能来响应用户的本次点击操作(即实现一次该功能)。因此只需要继承卡片类，然后再定义一个响应用户点击按钮事件的回调函数即可，代码实现如下：

```
class Button(Card):
    def __init__(self, x, y, width, height, text, font, font_colors, bg_colors, attribute, **kwargs):
        Card.__init__(self, x, y, width, height, text, font, font_colors, bg_colors, attribute)
    '''根据 button function 执行响应操作'''
    def do(self, game24_gen, func, sprites_group, objs):
        if self.attribute == 'NEXT':
            for obj in objs:
                obj.font = pygame.font.Font(obj.font_info[0], obj.font_info[1])
                obj.text = obj.attribute
```

```
            self.font = pygame.font.Font(self.font_info[0], self.font_info[1])
            self.text = self.attribute
            game24_gen.generate()
            sprites_group = func(game24_gen.numbers_now)
        elif self.attribute == 'RESET':
            for obj in objs:
                obj.font = pygame.font.Font(obj.font_info[0], obj.font_info[1])
                obj.text = obj.attribute
            game24_gen.numbers_now = game24_gen.numbers_ori
            game24_gen.answers_idx = 0
            sprites_group = func(game24_gen.numbers_now)
        elif self.attribute == 'ANSWERS':
            self.font = pygame.font.Font(self.font_info[0], 20)
            self.text = '[%d/%d]: ' % (game24_gen.answers_idx+1, len(game24_gen.answers)) + game24_gen.answers[game24_gen.answers_idx]
            game24_gen.answers_idx = (game24_gen.answers_idx+1) % len(game24_gen.answers)
        else:
            raise ValueError('Button.attribute unsupport <%s>, expect <%s>, <%s> or <%s>...' % (self.attribute, 'NEXT', 'RESET', 'ANSWERS'))
        return sprites_group
```

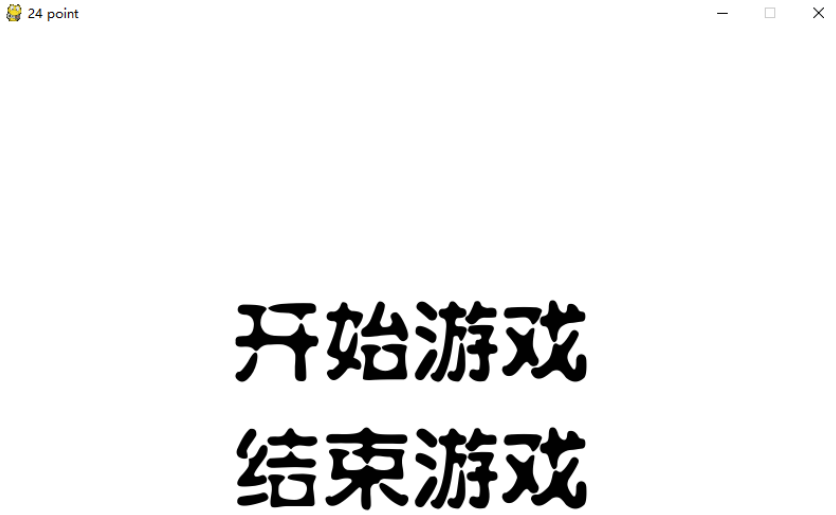
2.4 游戏主循环

2.4.1 开始界面

```
#游戏开始界面
start_ck = pygame.Surface(screen.get_size()) # 充当开始界面的画布
start_ck = start_ck.convert()
start_ck.fill((255, 255, 255)) # 白色画布 1（开始界面用的）
# 加载各个素材图片 并且赋予变量名
i1 = pygame.image.load("./images/s1.png")
i1.convert()
i11 = pygame.image.load("./images/s2.png")
i11.convert()
i2 = pygame.image.load("./images/n2.png")
i2.convert()
i21 = pygame.image.load("./images/n1.png")
i21.convert()

# 以下为选择开始界面鼠标检测结构。
n1 = True
while n1:
    clock.tick(30)
```

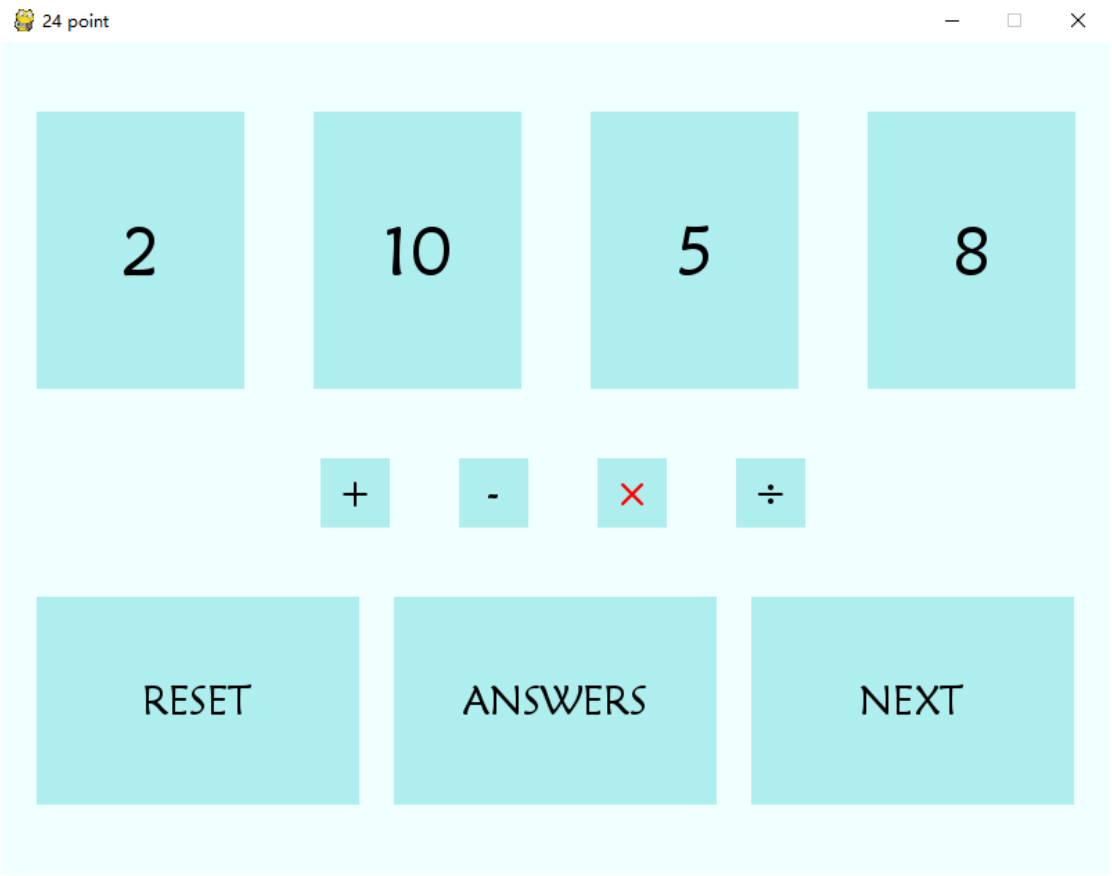
```
19.     buttons = pygame.mouse.get_pressed()
20.     x1, y1 = pygame.mouse.get_pos()
21.     if x1 >= 227 and x1 <= 555 and y1 >= 261 and y1 <= 327:
22.         start_ck.blit(i11, (200, 240))
23.         if buttons[0]:
24.             n1 = False
25.     elif x1 >= 227 and x1 <= 555 and y1 >= 381 and y1 <= 447:
26.         start_ck.blit(i21, (200, 360))
27.         if buttons[0]:
28.             pygame.quit()
29.             exit()
30.     else:
31.         start_ck.blit(i1, (200, 240))
32.         start_ck.blit(i2, (200, 360))
```



开始界面

鼠标触发后开始游戏

2.4.2 游戏主界面



效果图

初始化设置，代码如下：

```
# 初始化, 导入必要的游戏素材
pygame.init()
pygame.mixer.init()
screen = pygame.display.set_mode(SCREENSIZE)
pygame.display.set_caption('24 point')
win_sound = pygame.mixer.Sound(AUDIOWINPATH)
lose_sound = pygame.mixer.Sound(AUDIOLOSEPATH)
warn_sound = pygame.mixer.Sound(AUDIOWARNPATH)
pygame.mixer.music.load(BGMPATH)
pygame.mixer.music.play(-1, 0.0)
# 24 点游戏生成器
game24_gen = game24Generator()
game24_gen.generate()
# 精灵组
# --数字
number_sprites_group = getNumberSpritesGroup(game24_gen.numbers_now)
# --运算符
operator_sprites_group = getOperatorSpritesGroup(OPREATORS)
# --按钮
button_sprites_group = getButtonSpritesGroup(BUTTONS)
clock = pygame.time.Clock()
```

2.4.3 按键检测部分

```
for event in pygame.event.get():
if event.type == pygame.QUIT:
    pygame.quit()
    sys.exit(-1)
elif event.type == pygame.MOUSEBUTTONUP:
    mouse_pos = pygame.mouse.get_pos()
    selected_numbers = checkClicked(number_sprites_group, mouse_pos, 'NUMBER')
    selected_operators = checkClicked(operator_sprites_group, mouse_pos, 'OPREATOR')
    selected_buttons = checkClicked(button_sprites_group, mouse_pos, 'BUTTON')
```

2.4.4 实时更新卡片状态和变量

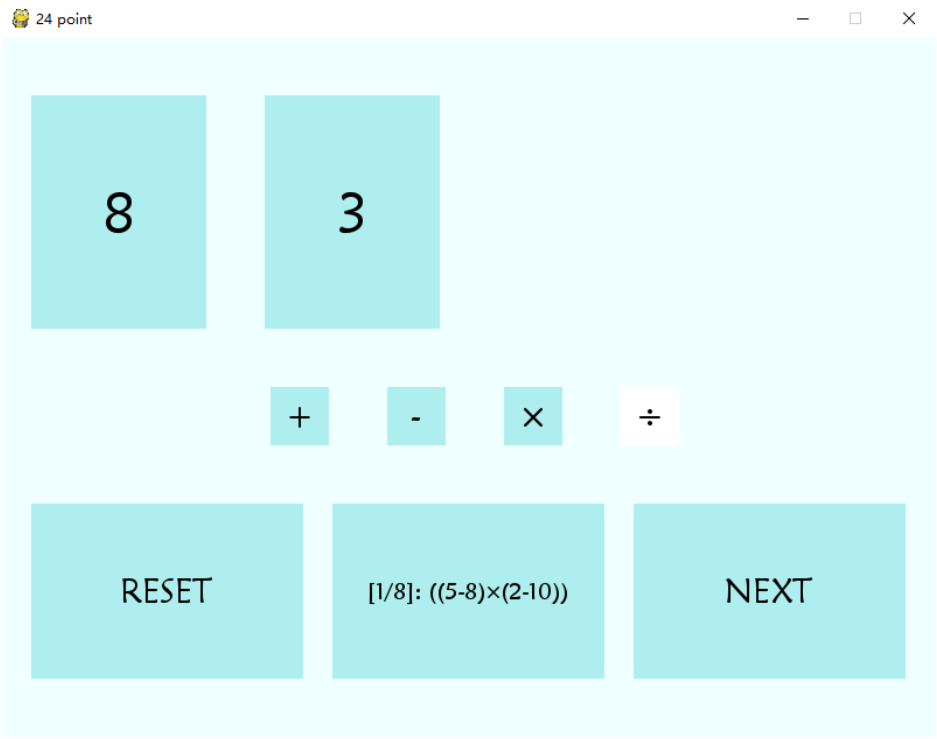
```
'''检查控件是否被点击'''
ef checkClicked(group, mouse_pos, group_type='NUMBER'):
selected = []
# 数字卡片/运算符卡片
```

```
if group_type == GROUPTYPES[0] or group_type == GROUPTYPES[1]:
    max_selected = 2 if group_type == GROUPTYPES[0] else 1
    num_selected = 0
    for each in group:
        num_selected += int(each.is_selected)
    for each in group:
        if each.rect.collidepoint(mouse_pos):
            if each.is_selected:
                each.is_selected = not each.is_selected
                num_selected -= 1
                each.select_order = None
            else:
                if num_selected < max_selected:
                    each.is_selected = not each.is_selected
                    num_selected += 1
                    each.select_order = str(num_selected)
        if each.is_selected:
            selected.append(each.attribute)
# 按钮卡片
elif group_type == GROUPTYPES[2]:
    for each in group:
        if each.rect.collidepoint(mouse_pos):
            each.is_selected = True
            selected.append(each.attribute)
# 抛出异常
else:
    raise ValueError('checkClicked.group_type unsupport <%s>, expect <%s>, <%s> or <%s>...' % (group_type, *GROUPTYPES))
return selected
```

2.4.5 运算操作及状态更新

```
if len(selected_numbers) == 2 and len(selected_operators) == 1:
noselected_numbers = []
for each in number_sprites_group:
    if each.is_selected:
        if each.select_order == '1':
            selected_number1 = each.attribute
        elif each.select_order == '2':
            selected_number2 = each.attribute
        else:
            raise ValueError('Unknow select_order <%s>, expect <1> or <2>...' % each.select_order)
    else:
```

```
            noselected_numbers.append(each.attribute)
        each.is_selected = False
    for each in operator_sprites_group:
        each.is_selected = False
    result = calculate(selected_number1, selected_number2, *selected_operators)
    if result is not None:
        game24_gen.numbers_now = noselected_numbers + [result]
        is_win = game24_gen.check()
        if is_win:
            win_sound.play()
        if not is_win and len(game24_gen.numbers_now) == 1:
            lose_sound.play()
    else:
        warn_sound.play()
    selected_numbers = []
    selected_operators = []
    number_sprites_group = getNumberSpritesGroup(game24_gen.numbers_now)
```

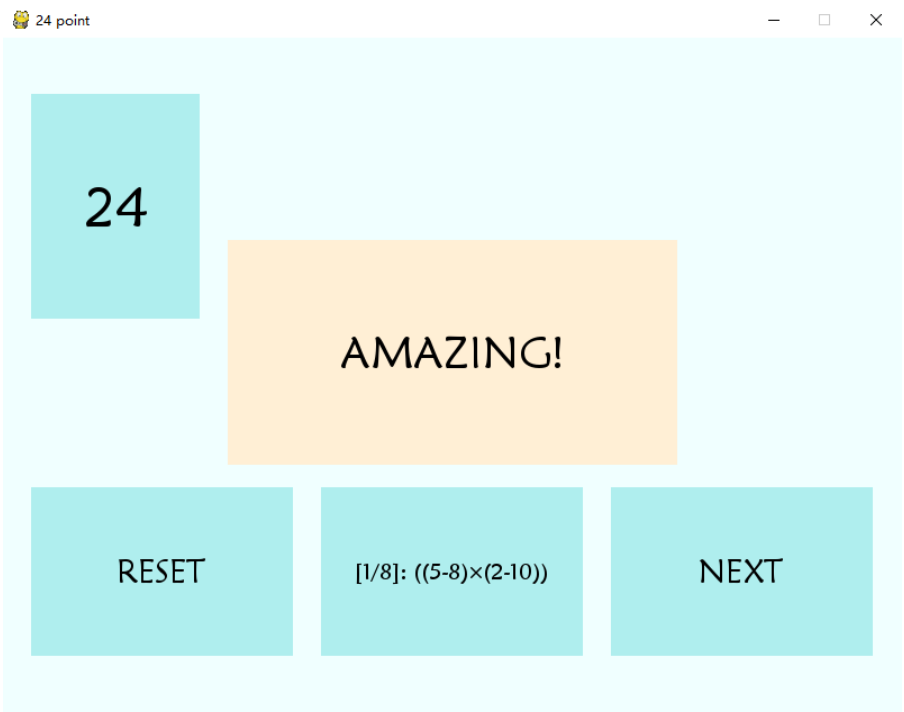


效果图

2.4.6 游戏胜利/游戏失败的结果显示

```
for each in number_sprites_group:
    each.draw(screen, pygame.mouse.get_pos())
for each in operator_sprites_group:
```

```
    each.draw(screen, pygame.mouse.get_pos())
for each in button_sprites_group:
    if selected_buttons and selected_buttons[0] in ['RESET', 'NEXT']:
        is_win = False
    if selected_buttons and each.attribute == selected_buttons[0]:
        each.is_selected = False
        number_sprites_group = each.do(game24_gen, getNumberSpritesGroup, number_sprites_group
, button_sprites_group)
        selected_buttons = []
    each.draw(screen, pygame.mouse.get_pos())
# 游戏胜利
if is_win:
    showInfo('Congratulations', screen)
# 游戏失败
if not is_win and len(game24_gen.numbers_now) == 1:
    showInfo('Game Over', screen)
pygame.display.flip()
clock.tick(30)
```



示例

2.4.7 游戏运行过程

文件夹中附加游戏运行视频，含各个按钮演示及界面操作。

3. 结论与未来方向

3.1 结论

通过制作此次游戏制作，利用 Pygame 完成了页面基本布局，基本上实现了预期功能。

在项目实践过程中，自己的项目开发能力也得到了提升，也掌握了团队开发一个基本软件的流程步骤。项目中也出现了很多 bug 和闪退现象，但最终经过耐心调试，逐渐解决，最终完成了整个项目。

3.2 未来方向

3.2.1 项目缺点

（1）由于游戏流程简单，故没有增加 Save/Load 功能，未来可设计关卡模式。

（2）界面美观程度有待提升。

3.2.2 未来改进方向

（1）增加关卡模式，提高游戏感。

（2）区分难易关卡，循序渐进。

（3）改进界面设计，增加美观程度。

4. 致谢

感谢皇甫老师上半学期的教学指导，受益匪浅。