重庆工商大学派斯学院

课程设计说明书

软件工程学院计算机科学与技术专业(本科)

2022 级 3 班

课题名称：找你坤小游戏的设计与实现

学生姓名：周德恩　学号：2022105083

指导教师：孙亚南

课程设计起止时间：

2024 年 12 月 16 日～ 12 月 29 日(共 2 周)

目录

1.选题背景

记忆游戏是一种经典的益智类游戏，它通过考验玩家的记忆力和观察力来提供娱乐和挑战。随着计算机技术的发展，利用编程语言开发具有良好用户体验的记忆游戏成为一个有趣且具有实践意义的项目。Python 作为一种简洁、高效且功能强大的编程语言，结合丰富的库（如 Pygame 用于图形界面开发），非常适合用于开发此类游戏。

本游戏以卡片配对为主要玩法。游戏开始时，屏幕上会显示若干张卡片，玩家的目标是找出两两相同的卡片进行配对消除，为了增加游戏的乐趣，添加了若干音效，配对消除成功时，随机播放音效；配对失败时，也会播放提示音效。

2.开发技术概述

本游戏是基于 Python 3.11.2 开发实现的。引用的库包括：pygame、os、random、numpy、PIL。开发者应具有 Python 程序设计语言、面向对象思想的基本知识和技能，以及掌握 pip 包管理器和 PyCharm 2021.1.1 集成开发环境的基本使用。

3.功能需求分析及功能设计

（1）游戏初始化模块。

负责游戏启动时的初始化工作，包括加载游戏所需的各种资源（如图像、音效等）、创建游戏窗口、设置游戏参数（如游戏名、屏幕尺寸等）、初始化游戏棋盘和卡片状态等。

（2）显示游戏界面模块。

负责游戏界面的绘制和显示，包括绘制游戏背景、卡片、按钮等界面元素，以及处理界面的更新和重绘操作，确保游戏界面在不同状态下都能正确显示相关信息。

（3）处理游戏逻辑模块。

实现游戏的核心逻辑，包括卡片的生成与分布、卡片是否匹配的判断、游戏状态管理等功能。该模块是游戏的关键部分，负责处理玩家的操作并根据游戏规则更新游戏状态。

（4）音效模块。

负责加载和播放游戏中的各种音效，根据游戏事件（如点击卡片、卡片配对成功、游戏开始、游戏结束等）触发相应的音效播放，增强游戏的音效体验。

（5）事件处理模块。

监听和处理玩家的各种输入事件（如鼠标点击、键盘按键等），将事件传递给相应的模块进行处理。例如，当玩家点击鼠标时，事件处理模块判断点击位置并将点击事件发送给游戏逻辑模块或游戏界面模块进行相应处理。

4. 系统流程分析

本游戏的目标是用户点击两张相同的卡片进行配对消除。运行程序，显示的是一个“Start Game”按钮，用户点击按钮后开始游戏。游戏过程中，用户点击了两张相同的卡片后会随机播放配对成功的音效，如果用户点击了两张不同的卡片或者同一张卡片点击了两下会播放一段配对失败的音效。当所有卡片都被消除是，会显示一个“Play Again”按钮，用户点击此按钮后会重置游戏，需要继续点击“Start Game”按钮，再玩一次。系统流程图如图 4-1 所示。

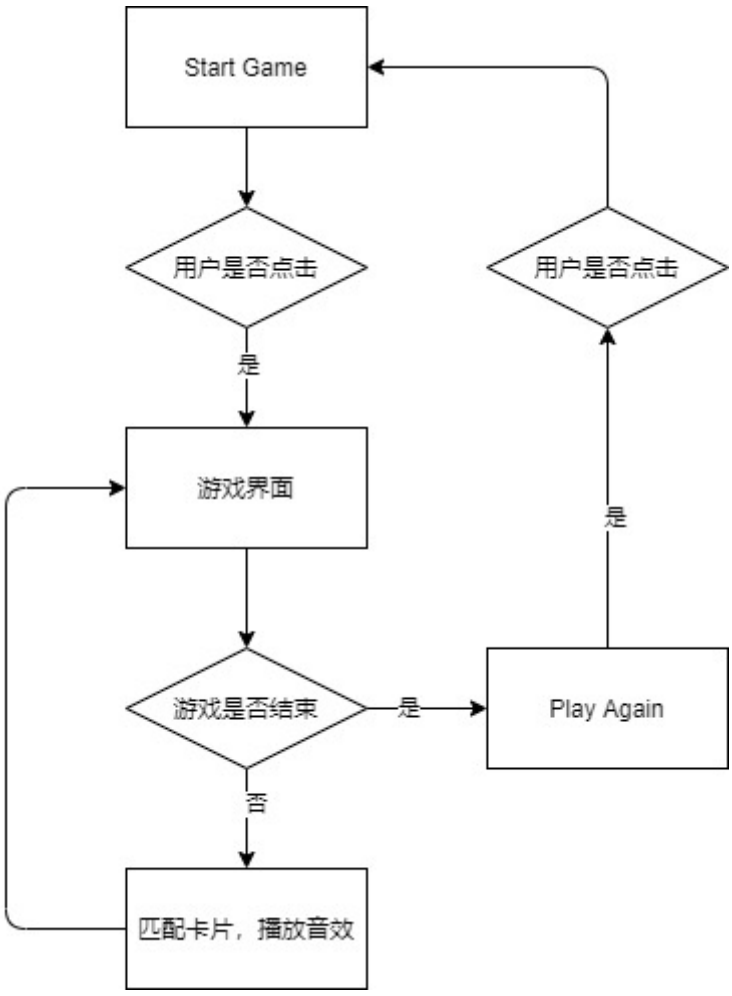

图 4-1 系统流程图

5. 系统实现

（1）游戏初始化模块。

创建游戏窗口：

1. 使用 pygame.display.set_mode(Constants.SCREEN_SIZE)创建一个指定大小的游戏窗口，其中 Constants.SCREEN_SIZE 是一个包含屏幕宽度和高度的元组。

2. 通过 pygame.display.set_caption("找你坤")设置游戏窗口的标题为“找你坤”。

代码：

```
    def __init__(self):
        pygame.init()
        pygame.mixer.init()
        pygame.font.init()
        pygame.display.set_caption("找你坤")
        self.screen = pygame.display.set_mode(Constants.SCREEN_SIZE)
        self.load_assets()
        self.init_game()
        self.game_started = False
```

加载资源:

1. 在 load_assets 函数中，首先获取资源文件的路径。对于图片资源，使用 os.path.join 函数结合项目目录和相对路径来构建完整的图片文件路径，如 self.image_paths = [os.path.join(base_path,'resource/images', f'image{i}.jpg') for i in range(1, 17)]，其中 base_path 是通过 os.path.dirname(os.path.abspath(__file__))获取的当前脚本文件所在目录。然后，使用 PIL 库（通过 Image.open 函数）加载图片，调整大小（img.thumbnail(Constants.TILE_SIZE)）并旋转（img.rotate(90, expand=True)），最后将图片转换为 numpy 数组（img.convert('RGB')），再通过 numpy_to_surface 函数将其转换为 Pygame Surface 对象并存储在 self.images_surfaces 列表中。

2.对于背景图片，同样使用 os.path.join 构建路径，加载后根据屏幕大小进行缩放（pygame.transform.scale）并存储在 self.background_image 中。

3.音效文件的加载类似，在 load_sounds 函数中，根据指定的目录（如 resource/sounds/wav）构建完整路径，使用 pygame.mixer.Sound 加载音效文件并存储在列表中返回。

代码：

```
    def load_assets(self):
        # 获取当前文件所在的目录
        base_path = os.path.dirname(os.path.abspath(__file__))

        # 加载并调整所有图片大小
        self.image_paths = [os.path.join(base_path, 'resource/images',
f'image{i}.jpg') for i in range(1, 17)]
        self.images_surfaces = []
        for path in self.image_paths:
            if not os.path.exists(path):
                raise FileNotFoundError(f"Image file not found: {path}")
            img_array = self.load_resize_and_rotate_image(path,
Constants.TILE_SIZE)
            img_surface = self.numpy_to_surface(img_array)
            self.images_surfaces.append(img_surface)

        # 加载并调整背景图片大小
```

```
            background_path = os.path.join(base_path,
'resource/images/background.png')
            if not os.path.exists(background_path):
                raise FileNotFoundError(f"Background image not found:
{background_path}")
            original_background_image =
pygame.image.load(background_path).convert()
            screen_width, screen_height = Constants.SCREEN_SIZE
            img_width, img_height = original_background_image.get_size()
            ratio = min(screen_width / img_width, screen_height / img_height)
            new_width = int(img_width * ratio)
            new_height = int(img_height * ratio)
            self.background_image =
pygame.transform.scale(original_background_image, (new_width, new_height))

    @staticmethod
        def load_resize_and_rotate_image(image_path, target_size):
            """加载图像，调整大小并旋转 90 度"""
            with Image.open(image_path) as img:
                img.thumbnail(target_size)
                img = img.rotate(90, expand=True)
                img = img.convert('RGB')
                return np.array(img)

        @staticmethod
        def numpy_to_surface(numpy_array):
            """将 Numpy 数组转换为 Pygame Surface 对象"""
            return pygame.surfarray.make_surface(numpy_array)
```

初始化游戏参数和棋盘:

1.在 init_game 函数中，创建一个二维列表 self.game_board 来表示游戏棋盘，初始化为 None。

2.为卡片生成唯一 ID,通过创建一个字典 id_mapping 来记录每个图片对应的 ID。先将所有图片进行水平翻转（self.flip_image_horizontally(img)）并重复一次（*2），然后随机打乱顺序（random.shuffle(all_images)）。遍历图片列表，为每个未在字典中出现的图片分配一个新的 ID，并将其与 ID 的映射关系存储在字典中。最后，将卡片依次放置在游戏棋盘上，创建 Card 对象并设置其图片、ID 和初始状态（未匹配），将其放置在 self.game_board 的相应位置。

3.初始化 self.selected_positions 列表，用于记录玩家选择的卡片位置。

代码：

```
        def init_game(self):
            """
            初始化游戏板，并为相同图案的卡片分配相同的唯一 ID。
            """
```

```
            self.game_board = [[None for _ in range(Constants.BOARD_SIZE)]
for _ in range(Constants.BOARD_SIZE)]
            unique_id = 0
            id_mapping = {}
            all_images = [self.flip_image_horizontally(img) for img in
self.images_surfaces] * 2
            random.shuffle(all_images)

            for img in all_images:
                if img not in id_mapping:
                    id_mapping[img] = unique_id
                    unique_id += 1

            for i in range(Constants.BOARD_SIZE):
                for j in range(Constants.BOARD_SIZE):
                    if all_images:
                        img = all_images.pop()
                        card = Card(img, id_mapping[img])
                        self.game_board[i][j] = card
            self.selected_positions = []
```

（2）显示游戏界面模块。

绘制背景:

1. 在 draw_background 函数中，使用 self.screen.blit 将背景图片 self.background_image 绘制在屏幕的中心位置，计算偏移量以确保背景图片居中显示（((Constants.SCREEN_SIZE[0] - self.background_image.get_width()) // 2, (Constants.SCREEN_SIZE[1] - self.background_image.get_height()) // 2)）。

代码：

```
# 绘制背景
    def draw_background(self):
        self.screen.blit(self.background_image,
((Constants.SCREEN_SIZE[0] - self.background_image.get_width()) // 2,

(Constants.SCREEN_SIZE[1] - self.background_image.get_height()) // 2))
```

绘制卡片:

1.在 draw_images 函数中，遍历游戏棋盘的每一行和每一列。如果棋盘位置上存在卡片（card = self.game_board[row][col]且 card 不为 None），则计算卡片在屏幕上的绘制位置（x = col * (Constants.TILE_SIZE[0] + Constants.MARGIN), y = row * (Constants.TILE_SIZE[1] + Constants.MARGIN)），然后使用 card.draw(self.screen, x, y)将卡片绘制在屏幕上。最后使用 pygame.display.flip 更新屏幕显示。

代码：

```
    def draw_images(self):
        for row in range(Constants.BOARD_SIZE):
```

```
            for col in range(Constants.BOARD_SIZE):
                card = self.game_board[row][col]
                if card:
                    x = col * (Constants.TILE_SIZE[0] + Constants.MARGIN)
                    y = row * (Constants.TILE_SIZE[1] + Constants.MARGIN)
                    card.draw(self.screen, x, y)
        pygame.display.flip()
```

绘制按钮:

1.在 draw_button 函数中，首先使用 pygame.font.Font 创建一个字体对象，然后使用 render 函数将按钮文本（如 “Start Game” 或 “Play Again”）渲染为 Surface 对象。计算文本的矩形区域（text_rect = text_surface.get_rect(center=(x + width // 2, y + height // 2))）以使其居中显示在按钮区域内。使用 pygame.draw.rect 绘制按钮矩形（可根据需要取消注释），并使用 self.screen.blit 将文本绘制在按钮上。通过获取鼠标位置（pygame.mouse.get_pos）和鼠标点击状态（pygame.mouse.get_pressed）来检测按钮是否被点击，如果点击则执行相应的 action 函数（如果有）。

代码：

```
    def draw_button(self, button_text, button_color, text_color, x, y, width, height, action=None):
        font = pygame.font.Font(None, 48)
        text_surface = font.render(button_text, True, text_color)
        text_rect = text_surface.get_rect(center=(x + width // 2, y + height // 2))
        # pygame.draw.rect(self.screen, button_color, [x, y, width, height])
        self.screen.blit(text_surface, text_rect)
        mouse = pygame.mouse.get_pos()
        click = pygame.mouse.get_pressed()
        if x < mouse[0] < x + width and y < mouse[1] < y + height:
            if click[0] == 1 and action:
                action()
```

游戏界面：

1.开始游戏界面如图 5-1 所示。

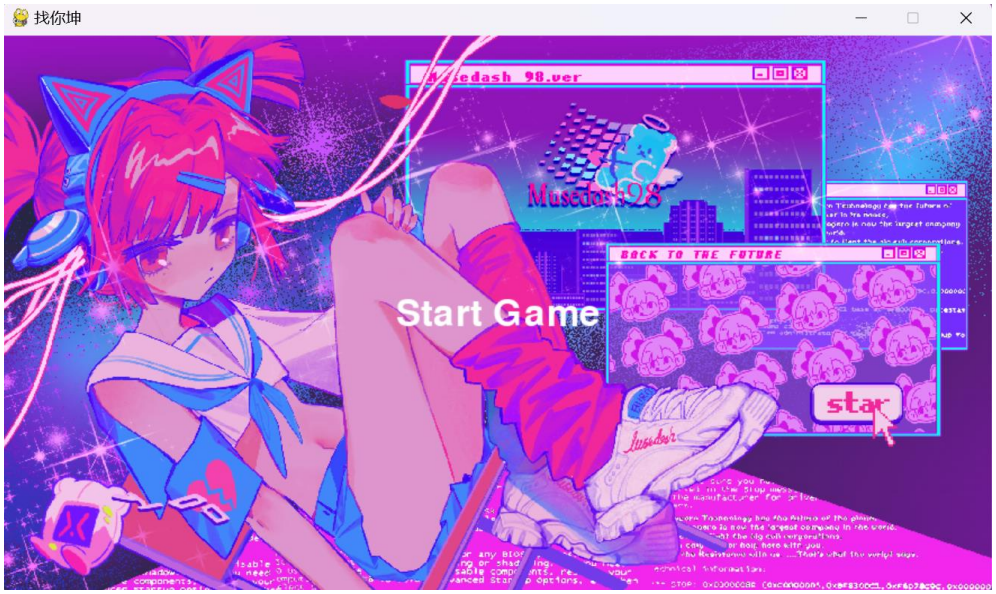

图 5-1 开始游戏界面

2. 游戏运行界面如图 5-2 所示。

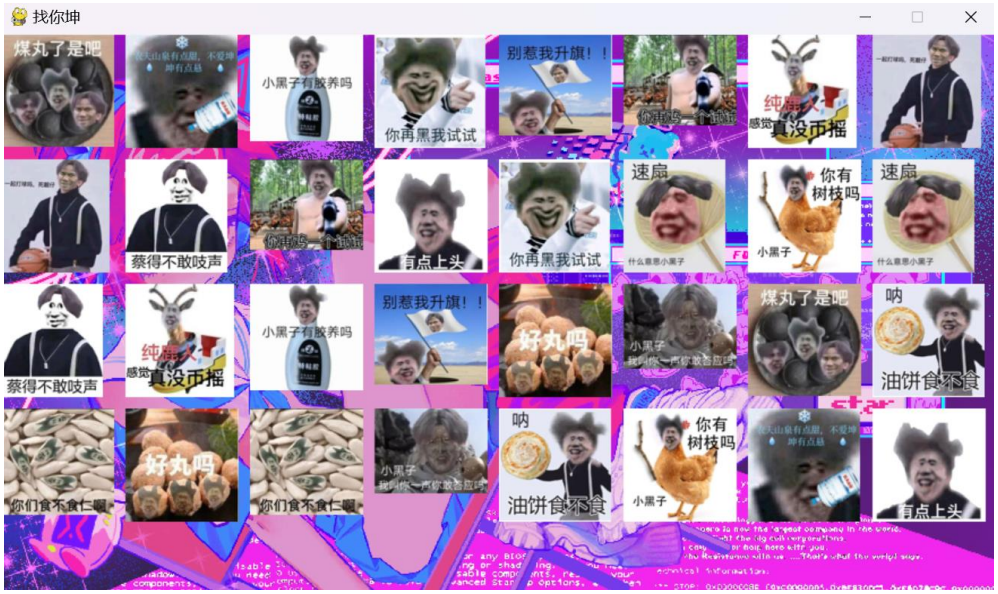

图 5-2 游戏运行界面

3. 游戏介绍界面如图 5-3 所示。

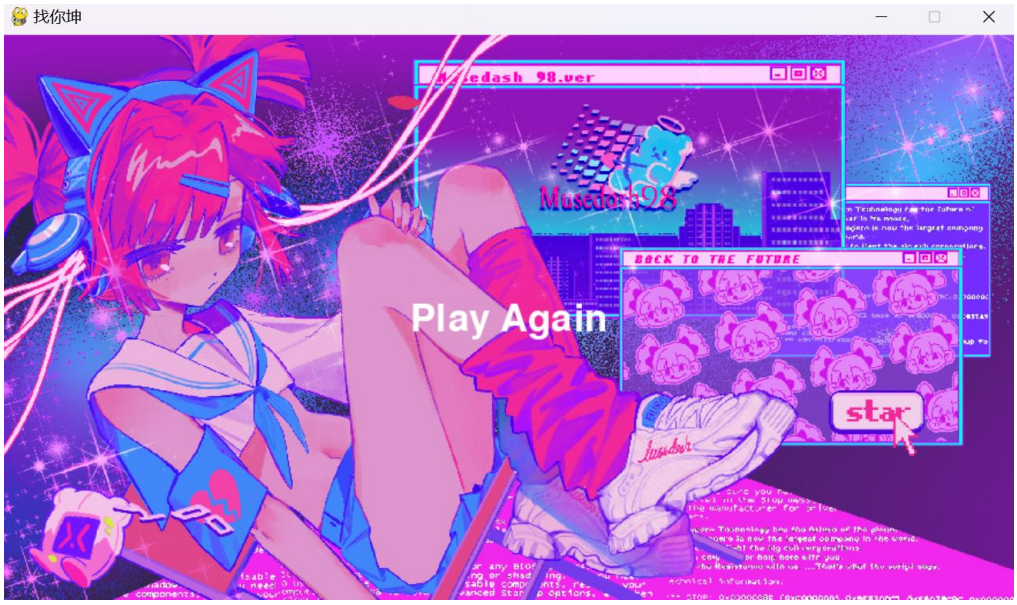

图 5-3 游戏结束界面

（3）处理游戏逻辑模块。

卡片匹配判断：

1.在 main_loop 函数中，当检测到鼠标点击事件（pygame.MOUSEBUTTONDOWN）且游戏已开始（self.game_started 为 True）且选择的卡片数量小于 2 时，计算点击位置对应的卡片行和列(col = mouse_x // (Constants.TILE_SIZE[0] + Constants.MARGIN), row = mouse_y // (Constants.TILE_SIZE[1] + Constants.MARGIN))。检查点击位置是否在游戏棋盘范围内且该位置有卡片（self.is_within_bounds(row, col) and self.game_board[row][col]）。如果选择了两张卡片（len(self.selected_positions) == 2），使用 check_match 函数判断两张卡片是否匹配（通过比较卡片的 ID）。如果匹配成功，播放随机音效（self.play_random_sound(sounds)），将匹配的卡片标记为已匹配（self.game_board[pos[0]][pos[1]].is_matched = True）并从棋盘上移除（self.game_board[pos[0]][pos[1]] = None）；如果匹配失败，播放提示音效（self.play_sound(sound_ji)）。无论匹配成功与否，都清空选择的卡片位置列表（self.selected_positions.clear()）。

代码：

```
                elif event.type == pygame.MOUSEBUTTONDOWN:
                    if not self.game_started:
                        # 检查是否点击了开始按钮
                        button_width, button_height = 200, 100
                        button_x = (Constants.SCREEN_SIZE[0] -
button_width) // 2
                        button_y = (Constants.SCREEN_SIZE[1] -
button_height) // 2
                        self.draw_button("Start Game", (0, 255, 0), (255,
255, 255), button_x, button_y, button_width,
                                         button_height,
                                         action=lambda: setattr(self,
```

```
'game_started', True))
                        elif self.is_game_over():
                            button_width, button_height = 200, 100
                            button_x    =    (Constants.SCREEN_SIZE[0]    -
button_width) // 2
                            button_y    =    (Constants.SCREEN_SIZE[1]    -
button_height) // 2
                            self.draw_button("Play Again", (0, 255, 0), (255,
255, 255), button_x, button_y, button_width,
                                             button_height,
                                             action=self.reset_game)
                        elif len(self.selected_positions) < 2:
                            mouse_x, mouse_y = event.pos
                            col  =  mouse_x  //  (Constants.TILE_SIZE[0]  +
Constants.MARGIN)
                            row  =  mouse_y  //  (Constants.TILE_SIZE[1]  +
Constants.MARGIN)
                            if    self.is_within_bounds(row,    col)    and
self.game_board[row][col]:
                                if  len(self.selected_positions)  ==  1  and
(row, col) == self.selected_positions[0]:
                                    print("不能选择同一张图片两次！")
                                    self.play_sound(sound_ji)
                                    continue
                                self.selected_positions.append((row, col))
                                if len(self.selected_positions) == 2:
                                    if
self.check_match(*self.selected_positions):
                                        print("Match found!")
                                        self.play_random_sound(sounds)
                                        for pos in self.selected_positions:

self.game_board[pos[0]][pos[1]].is_matched = True
                                            # 将匹配成功的卡片从游戏板上移
除
                                            self.game_board[pos[0]][pos[1]]
= None
                                    else:
                                        print("No match.")
                                        self.play_sound(sound_ji)
                                    self.selected_positions.clear()
```

游戏状态管理：

1. 在 main_loop 函数中，不断检查游戏是否结束（self.is_game_over 函数），该

函数遍历游戏棋盘的所有卡片，只要存在未匹配的卡片就返回 False，否则返回 True。当游戏结束时，显示 “Play Again” 按钮，玩家点击后可重置游戏（self.reset_game 函数）。在游戏开始前，显示 “Start Game” 按钮，玩家点击后开始游戏（通过设置 self.game_started = True）。

代码：

```
    def reset_game(self):
        self.load_assets()
        self.init_game()
        self.selected_positions.clear()
        self.game_started = False

    def is_game_over(self):
        for row in self.game_board:
            for card in row:
                if card and not card.is_matched:
                    return False
        return True
```

（4）音效模块。

音效加载：

1.在 load_sound 函数中,根据传入的音效文件路径(如 resource/sounds/鸡.wav)，使用 os.path.join 构建完整路径，然后尝试使用 pygame.mixer.Sound 加载音效文件。如果加载失败，打印错误信息并返回 None。

2.在 load_sounds 函数中，根据指定的目录（如 resource/sounds/wav），使用 os.listdir 获取目录中的所有文件名，筛选出符合指定扩展名（如.wav 或.ogg）的文件，构建完整路径并加载为音效对象，存储在列表中返回。

代码：

```
    # 加载多个音效文件到列表中
    @staticmethod
    def load_sounds(directory, extensions=('wav', 'ogg')):
        sounds = []
        sound_directory = os.path.join(os.path.dirname(os.path.abspath(__file__)), directory)
        for filename in os.listdir(sound_directory):
            if any(filename.lower().endswith(ext) for ext in extensions):
                sound_path = os.path.join(sound_directory, filename)
                sounds.append(pygame.mixer.Sound(sound_path))
        return sounds

    # 加载一个音效
    def load_sound(self, sound_file_path):
        try:
            full_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), sound_file_path)
```

```
            return pygame.mixer.Sound(full_path)
        except pygame.error as e:
            print(f"无法加载音效文件{sound_file_path}:{e}")
            return None
```

音效播放：

1.在play_sound函数中,检查传入的音效对象是否不为None,如果是则调用其play方法播放音效。

2.在play_random_sound函数中，首先检查音效列表是否为空，如果为空则打印提示信息。否则，从列表中随机选择一个音效（random.choice(sounds)）并播放。

代码：

```
    # 播放一个指定音效
    def play_sound(self, sound):
        if sound:
            sound.play()

    # 定义一个函数来随机播放音效
    def play_random_sound(self, sounds):
        if not sounds:
            print("没有可用的音效文件")
            return
        # 随机选择一个音效
        selected_sound = random.choice(sounds)
        # 播放音效
        selected_sound.play()
```

（5）事件处理模块。

在main_loop函数中，使用pygame.event.get获取所有的事件。遍历事件列表，当检测到 pygame.QUIT 事件时，设置 running = False 以退出游戏主循环。对于pygame.MOUSEBUTTONDOWN事件，根据游戏状态（未开始、进行中、已结束）进行相应的处理,如点击按钮、处理卡片点击等操作,通过调用其他模块的函数来实现具体功能（如self.draw_button、self.check_match等）。

代码：

```
        while running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False
```

6.总结和建议

本次课程设计通过开发“找你坤”小游戏，成功运用了 Python 编程语言以及Pygame、os、random、numpy、PIL 等相关库，实现了一个具有基本功能的游戏项目。

在开发过程中,深入理解了游戏开发的流程,从选题背景的确定,到功能需求分析、

系统设计、详细设计以及最终的编码实现，每个环节都紧密相连，缺一不可。通过游戏初始化模块，能够顺利地创建游戏窗口、加载各种资源并初始化游戏棋盘和卡片状态，为游戏的运行奠定了基础。显示游戏界面模块让游戏具备了良好的视觉呈现，背景、卡片和按钮的绘制使游戏界面更加生动和易于操作。处理游戏逻辑模块作为游戏的核心，准确地实现了卡片的生成与分布、匹配判断以及游戏状态的管理，确保了游戏的正常玩法逻辑。音效模块为游戏增添了趣味性和互动性，不同事件触发的音效让玩家有了更丰富的游戏体验。事件处理模块则有效地监听和处理了玩家的输入事件，实现了游戏与玩家之间的交互。

该游戏仍存在许多的问题。游戏界面的美观度还有提升空间，卡片的样式和布局可以更加精致和多样化，以吸引玩家的注意力。目前的界面设计相对较为简单，缺乏一些动画效果或视觉特效来增强游戏的趣味性。对于玩家操作的反馈可以更加丰富，除了音效和简单的提示信息外，可以增加一些视觉上的反馈，例如卡片点击时的短暂变色或放大效果，让玩家更直观地感受到操作的响应。部分代码的结构可能不够清晰，函数和类的职责划分在一些复杂逻辑处理中略显模糊。例如，在游戏逻辑模块中，卡片匹配和游戏状态管理的代码交织在一起，可能会给后续的代码维护和功能扩展带来一定的困难。

Python 程序设计

课程设计成绩评定

成 绩 评 定 ：_____

指导教师签字：_____

2024 年 12 月 26 日