第 3 章

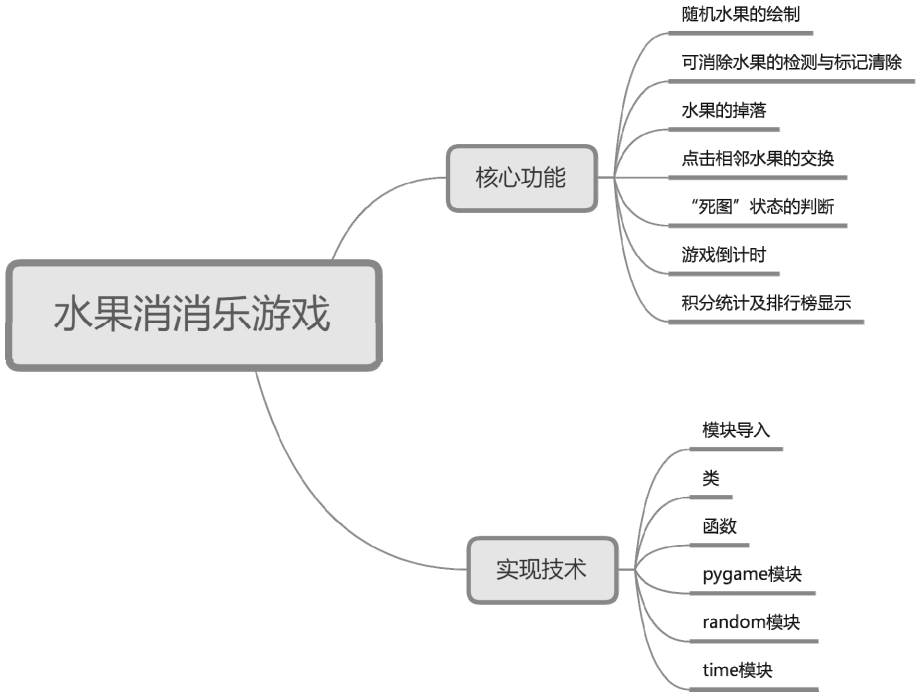
水果消消乐游戏

——模块导入 + 类 + 函数 + pygame + random + time


项目微视频

风靡全国的水果消消乐游戏，操作简单，生动有趣，是一款百玩不厌、广受大众喜爱的经典休闲小游戏。本章将使用 Python + pygame 模块实现该游戏。具体规则：在规定的时间内，玩家需要把至少 3 个相同的水果放置在一起才可以消除并得分。

本项目的核心功能及实现技术如下：



3.1 开 发 背 景

在当今数字化时代，休闲益智类游戏因其简单易懂、操作便捷、娱乐性强等特点，受到了广大玩家的喜爱。在这样的市场背景下，本章将使用 Python 中的 pygame 游戏模块开发一款休闲益智类游戏——水果消消乐。该游戏的灵感来源于经典的消除类游戏，采用了丰富多彩的水果元素作为消除对象，使得游戏画面更加生动有趣。

本游戏项目的实现目标如下：

☑ 游戏界面美观大方，操作简单。

☑ 遵循经典的消除类游戏规则。

☑ 能够实时显示积分和倒计时，提升挑战性。

☑ 通过游戏中的排行榜功能，玩家可以查看自己的成绩。

3.2 系 统 设 计

3.2.1 开发环境

本项目的开发及运行环境如下：

☑ 操作系统：推荐 Windows 10、Windows 11 及以上。
☑ 开发工具：PyCharm 2024（向下兼容）。
☑ 开发语言：Python 3.12。
☑ Python 内置模块：sys、os、time、random。
☑ 第三方模块：pygame（2.5.2）。

3.2.2 业务流程

在启动项目后，首先进入首屏界面。单击 Play 按钮，即可进入游戏主界面。在游戏主页面中单击相邻的水果方块（红框）切换位置，当至少 3 个一样的水果连在一起时，连在一起的水果（蓝框）即可消除。当水果消除获得一定的分数时，会显示鼓励性的提示。另外，当游戏结束、处于“死图”状态或者游戏规定时间到时，游戏进入排行榜界面。本项目的业务流程如图 3.1 所示。

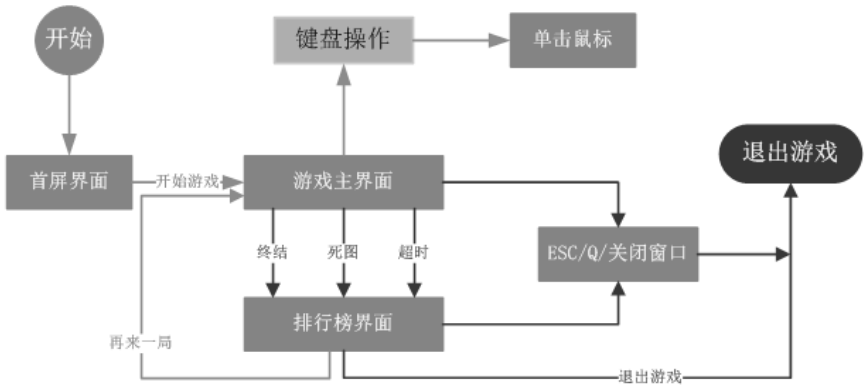


图 3.1 水果消消乐游戏业务流程

3.2.3 功能结构

本项目的功能结构已经在章首页中给出，该项目实现的具体功能如下：

☑ 游戏中水果的随机生成。
☑ 可消除水果的标记与清除。
☑ 水果的自动掉落。
☑ 单击相邻水果的切换。
☑ 游戏中无可消除水果时的判断。
☑ 倒计时的实现。
☑ 积分的统计和显示。
☑ 积分排行榜的实现。

3.3 技 术 准 备

3.3.1 技术概览

☑ 模块导入：在 Python 中，一个扩展名为.py 的文件就被称为一个模块。通常情况下，我们把能够实现某一特定功能的代码放置在一个文件中作为一个模块，从而方便被其他程序和脚本导入并使用。编写好一个模块后，如果要使用该模块，只需要在相应的代码文件中导入即可。在 Python 代码中导入模块主要有以下 5 种形式：

```
import pygame                                  # 导入整个模块
from pygame.locals import *                    # 导入模块中的所有类和对象
from functools import lru_cache                # 导入模块中的某个类
from .entity import Element, Font_Fact         # 导入自定义模块中的某个类（要导入的模块与当前文件在同一目录下）
from core.handler import Manager               # 导入自定义模块中的某个类（要导入的模块与当前文件不在同一目录下）
```

☑ 类：在 Python 中，类表示具有相同属性和方法的对象的集合。在使用类时，需要先定义类，然后创建类的实例，通过类的实例可以访问类中的属性和方法。类的定义使用 class 关键字来实现。本项目中多次用到了类，例如，将用到的图片和文字封装成类，代码如下：

```
class Element(pygame.sprite.Sprite):
    """
    绘制图片类
    """
class Font_Fact(pygame.sprite.Sprite):
    """
    绘制文字精灵组件类
    """
```

☑ 函数：在 Python 中，可以把实现某一功能的代码定义为一个函数，然后在需要使用时，随时调用即可。定义函数时需要使用 def 关键字，后面跟要定义的函数名，以及参数（如果有）。本项目中多次用到了函数，例如，定义一个名为 open_game_init()的函数，用来对游戏首屏页面进行初始化，代码如下：

```
def open_game_init(self):
    """ 游戏首屏页面初始化 """
    # 绘制首屏的背景图片
    Element(Element.bg_open_image, (0, 0)).draw(self.screen)
    # 开始按钮的绘制
    Element(Element.game_start_button_image, Element.game_start_button_posi).draw(self.screen)
    pygame.display.flip()
```

☑ pygame 模块：pygame 模块是一个完全免费、开源的 Python 游戏模块，它支持 Windows、Linux、macOS 等操作系统，具有良好的跨平台性。使用 pygame 模块进行游戏开发的流程如图 3.2 所示。

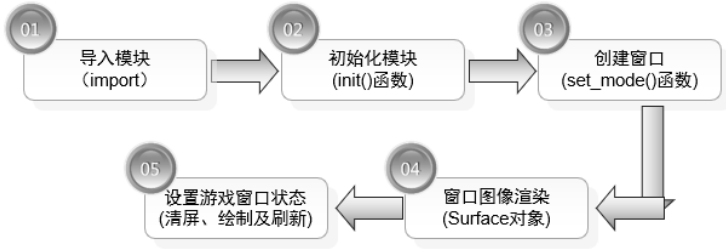


图 3.2 pygame 程序开发流程

例如，下面代码定义了一个基本的 pygame 游戏开发框架：

```
import sys

# 导入 pygame 及常量库
import pygame
from pygame.locals import *

SIZE = WIDTH, HEIGHT = 640, 396
FPS = 60

pygame.init()
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Pygame__明日")
clock = pygame.time.Clock()
# 创建字体对象
font = pygame.font.SysFont(None, 60, )

running = True
# 主体循环
while running:
    # 1. 清屏
    screen.fill((25, 102, 173))
    # 2. 绘制
    for event in pygame.event.get():                    # 事件索取
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    # 3.刷新
    pygame.display.update()
    clock.tick(FPS)
```

有关模块导入、类、函数、pygame 模块等知识在《Python 从入门到精通（第 3 版）》中有详细的讲解，对这些知识不太熟悉的读者可以参考该书对应的内容。除了以上知识，本项目还用到了 random 模块和 time 模块，这是 Python 内置的两个标准模块，下面对它们的使用进行必要的介绍，以确保读者可以顺利完成本项目。

3.3.2 random 模块的使用

random 模块用于实现各种分布的伪随机数生成器，可以根据不同的实数分布来随机生成值，如随机生成指定范围的整数、浮点数、序列等。random 模块的常用功能、方法及举例如表 3.1 所示。

表 3.1 random 模块的常用功能、方法及举例

功　能	方　法	举　例
生成普通随机数	random.random() random.choices(population) random.choice(seq) random.randint(a,b) random.randrange(stop) random.randrange(start, stop[,step]) random.uniform(a,b)	random.random()　# 返回 0.0 至 1.0 之间的随机浮点数 random.choices(['apple','orange','banana']))　# 随机输出列表中的元素 random.choice([1,2,3,4,5])　# 从列表中生成随机数 random.choice(('1','3','5'))　# 从元组中生成随机数 random.choice(["+","-"])　# 随机生成+或者-号 random.randint(0,10))　# 0 到 10 范围内的随机整数（包含 10） random.randrange(5))　# 0 至 5 之间的随机整数，不包含 5 random.randrange(1,10)　# 从[1, 2, 3, ... 8, 9]序列中返回一个随机数 random.randrange(20, 40, 2)　# 从[20, 22, 24, ... 36, 38]序列中返回一个随机偶数 random.uniform(1.0,5.0)　# 指定参数为浮点数 random.uniform(1,5)　# 指定参数为整数 random.uniform(5.0,1.0)　# 参数 a 大于参数 b

续表

功　能	方　法	举　例
生成不重复随机数	random.sample(population,k)	random.sample([1,2,3,4,5],3)　# 随机生成 3 个不重复的随机数 random.sample(['张','王','李','赵','周','吴','郑','徐'],3) random.sample([["java","oracle"],["C#","asp.net"],["PHP","mysql"],["C","C++"]],2)
随机排列元素	random.shuffle(x[,random])	random.shuffle([1,2,3,4,5,6])　# 将数字 1～6 的顺次打乱 random.shuffle(["北京","上海","广州","深圳"])　# 将 4 个城市的顺次打乱
特色规则的随机数	random.betavariate(alpha, beta) random.gauss(mu,sigma) random.getrandbits(k) random.normalvariate(mu,sigma) random.lognormvariate(mu, sigma) random.expovariate(lambd) random.gammavariate(alpha, beta) random.paretovariate(alpha) random.triangular(low, high, mode) random.vonmisesvariate(mu, kappa) random.weibullvariate(alpha, beta)	random.betavariate(1, 3)　# 生成 0～1 以 beta 概率分布的随机数 random.gauss(1, 3)　# 生成以高斯分布的随机数 random.getrandbits(5))　# 生成一个 K（指定值）随机位的整数 random.normalvariate(1, 3)　# 生成以正态分布的随机数 random.lognormvariate(1, 3)　# 生成以对数正态分布的随机数 random.expovariate(3.14)　# 生成以指数分布的随机数 random.gammavariate(1, 3)　# 生成以 gamma 分布的随机数 random.paretovariate(1)　# 生成以 Pareto 分布的随机数 random.triangular(0, 1, 0.5)　# 生成以三角形分布的随机数 random.vonmisesvariate(1, 3)　# 生成以 von Mises 分布的随机数 random.weibullvariate(1, 3)　# 生成以 Weibull 分布的随机数
初始化生成器	random.seed(a=None, version=2)	random.seed()　# 默认种子 random.seed(a=1)　# 整数种子 random.seed(a='1',version=1)　# 字符种子
生成器的状态	1．.random.getstate()获取当前生成器内部状态的对象 2．random.setstate(state)恢复生成器的内容状态	state = random.getstate()　# 获取当前生成器内部状态的对象 random.seed()　# 系统时间作为种子，初始化生成器 random.setstate(state)　# 恢复生成器的内部状态

例如，本项目中使用了 random 模块的 randint()方法为游戏页面中的小方块随机生成水果图片，代码如下：

```
for i in range(self._height):
    for j in range(self._width):
        self.animal[i][j] = random.randint(0, 5)
```

3.3.3　time 模块的使用

time 模块提供了 Python 中各种与时间处理相关的方法，该模块中对于时间表示的格式有如下 3 种。

☑　timestamp 时间戳：表示的是从 1970 年 1 月 1 日 00:00:00 开始按秒计算的偏移量。

☑　struct_time 时间元组：共有 9 个元素组。分别为年、月、日、时、分、秒、一周中的第几日、一年中的第几日、夏令时。

☑　format time 格式化时间字符串：格式化的结构使时间更具可读性，包括自定义格式和固定格式。

time 模块的常用方法及说明如表 3.2 所示。

表 3.2　time 模块的常用方法及说明

方　法	说　明	方　法	说　明
asctime()	接收时间元组并返回一个可读的长度为 24 个字符的字符串	mktime()	接收时间元组并返回时间戳
clock()	以浮点数返回当前的时间	sleep()	按指定的秒数使程序休眠若干时间
ctime()	接收时间戳并返回一个字符串	strftime()	将日期格式转换为字符串格式
gmtime()	接收时间戳并返回 UTC 时区的时间元组	strptime()	根据指定的格式把时间字符串转换为时间元组
localtime()	接收时间戳并返回本地时间的时间元组	time()	返回当前时间的时间戳

例如，本项目中使用 time 模块的 strftime()方法和 localtime()方法计算游戏的倒计时，代码如下：

```
time_str = time.strftime("%X", time.localtime(self.TIMEOUT // 1000 - self.running_time // 1000)).partition(":")[2]
```

另外，本项目中还用到了 pygame 模块中的 time 子模块，该子模块是 pygame 游戏中的一个专门的时间控制模块，其主要功能是管理时间和游戏帧数率（即 FPS）。本项目中主要用到了 pygame.time 模块中的 Clock()方法、get_ticks()方法和 delay()方法。其中，Clock()方法用来创建一个时钟对象，以确定游戏要以多大的帧数运行；get_ticks()方法用来获取时间（以毫秒为单位）；delay()方法用来使程序暂停一段时间。示例代码如下：

```
clock = pygame.time.Clock()                    # 创建时钟对象
self.end_time = pygame.time.get_ticks()        # 更新结束时间
pygame.time.delay(500)                         # 使程序暂停一段时间，以便能够看清楚绘制的文字
```

3.4 搭建游戏主框架

开发水果消消乐游戏之前，首先需要搭建游戏的主框架，步骤如下。

（1）在 PyCharm 中创建水果消消乐游戏项目，并在项目目录下依次创建 bin、core、conf 和 static 这 4 个 Python 包，然后在 static 包中创建 img 和 font 两个 Python 包，用于存放整个项目所用到的图片和字体文件。

（2）在 bin 包中创建一个 main.py 文件，作为整个项目的主文件，在该文件中创建主函数 main()，主要用来调用实现游戏各个功能逻辑的不同接口。在 main.py 主文件中实现 Pygame 程序循环，在该循环中绘制页面、监听事件，并基于事件改变游戏状态，从而执行不同的操作，示意图如图 3.3 所示。

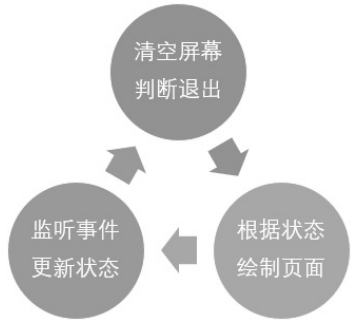


图 3.3 主程序循环逻辑示意图

main.py 主文件的初始代码如下：

```
import pygame
from pygame.locals import *
import sys

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                              # 设备的检测
    pygame.font.init()                         # 字体文件的初始化

    # 在这里创建每一个页面类的对象

    while 1:                                   # 游戏主循环

        # 在这里判断不同的游戏状态，从而执行不同的操作

        for event in pygame.event.get():       # 事件的监听与循环
            # 用户敲击键盘的键盘事件判断
            if event.type == KEYDOWN:
                # 判断用户按下 Q 键或者 ESC 键
                if event.key == K_q or event.key == K_ESCAPE:
                    sys.exit()
            # 用户关闭游戏窗口的鼠标事件判断
            if event.type == QUIT:
                sys.exit()
```

```
            # 在这里进行不同页面的事件监听，从而改变游戏状态
```

（3）在项目目录下创建一个 manage.py 文件，作为整个游戏的启动文件，该文件代码如下：

```
__auther__ = "明日科技"
__version__ = "master_v1"

from bin.main import main

if __name__ == '__main__':
    main()                                    # 游戏主函数
```

（4）在 core 包中创建一个 base.py 文件，并在其中创建 Base 类。该类中定义一个 status 类变量，默认值为 0，代表首屏状态；若为 1，则代表游戏页面状态；若为 2，则代表积分排行榜状态。然后通过此类的__init__()初始化方法执行窗口的初始化工作，如窗口的尺寸、窗口对象的实例化、窗口标题的设置等。base.py 文件代码如下：

```
import pygame

class Base:
    """
    一些公共变量的管理
    """

    clock = pygame.time.Clock()               # 创建一个对象，用来跟踪时间
    _screen_size = (900, 600)                 # 屏幕的尺寸
    # 0: 首屏状态， 1: 游戏页面状态  2: 排行榜状态
    status = 0                                # 游戏状态

    def __init__(self):
        # 窗口对象的获取
        self.screen = pygame.display.set_mode(self._screen_size)
        # 设置窗口的标题
        pygame.display.set_caption("水果消消乐游戏")
```

搭建完的游戏主框架如图 3.4 所示。

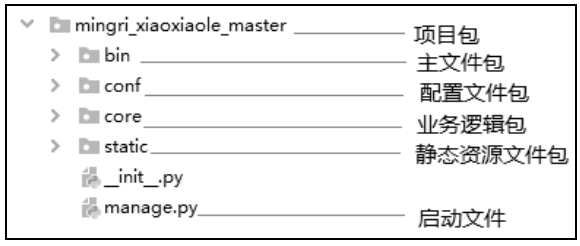


图 3.4　游戏主框架

3.5　功 能 设 计

3.5.1　精灵类设计

水果消消乐游戏中使用精灵类来表示游戏中用到的图片以及文本等，设计精灵类的步骤如下。

（1）在 core 包中创建一个 entity.py 文件，作为水果消消乐游戏中的精灵类。在 entity.py 文件中创建一个 Element 类，使其继承自 pygame.sprite.Sprite 类，该类主要用来绘制游戏中用到的图片，实现代码如下：

```
class Element(pygame.sprite.Sprite):
    """
    绘制图片类
    """
    bg_open_image = "static/img/tree.png"
    bg_choice_image = "static/img/bs.png"
    bg_start_image = "static/img/bg.png"
    game_start_button_image = "static/img/game_start_button.png"
    game_start_button_posi = (300, 250)                    # 开始游戏按钮的坐标
    start_button = (300, 120)                              # 开始游戏按钮的大小
    speed = [0, 0]
    stop = 'static/img/exit.png'                           # 暂停键
    stop_position = (20, 530)                              # 暂停键坐标
    frame_image = "static/img/frame.png"                   # 选中框

    board_score = "static/img/task.png"                    # 分数显示板
    score_posi = (736, 15)                                 # 分数显示板的坐标
    brick = 'static/img/brick.png'                         # 背景图片
    # 图标元组，包括 6 个水果
    animal = ('static/img/lemon.png', 'static/img/watermelon.png',
              'static/img/Grapefruit.png', 'static/img/Kiwifruit.png',
              'static/img/Nettedmelon.png', 'static/img/Avocado.png')
    # 消除动画图片
    bling = ("static/img/bling1.png", "static/img/bling2.png",
             "static/img/bling3.png", "static/img/bling4.png",
             "static/img/bling5.png", "static/img/bling6.png",
             "static/img/bling7.png", "static/img/bling8.png",
             "static/img/bling9.png")
    # 鼓励话语图片
    single_score = ('static/img/good.png', 'static/img/great.png',
                    'static/img/amazing.png',  'static/img/excellent.png',
                    'static/img/unbelievable.png')
    # 0～9 数字图片
    score = ('static/img/0.png', 'static/img/1.png', 'static/img/2.png',
             'static/img/3.png', 'static/img/4.png', 'static/img/5.png',
             'static/img/6.png', 'static/img/7.png', 'static/img/8.png',
             'static/img/9.png',)
    none_animal = 'static/img/noneanimal.png'              # 无可消除水果
    none_animal_posi = (230, 150)                          # 无可消除水果表示的坐标
    destory_animal_num = [0, 0, 0, 0, 0, 0]                # 消除各水果的个数
    mouse_replace_image = 'static/img/mouse.png'           # 鼠标替换图片

    time_is_over_image = "static/img/time_is_over.png"     # 游戏时间超时的图片
    time_is_over_posi = (233, 50)

    score_order_rect = (320, 125, 230, 400)                # 积分排行榜的 Rect 对象

    def __init__(self, image_file, posi):
        """
        初始化
        """
        # 注意必须用，否则精灵组不生效
        super(Element, self).__init__()
        self.image = pygame.image.load(image_file).convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.topleft = posi                           # 左上角坐标
        self.speed = [0, 0]
        self.init_position = posi                          # 记录原始位置

    def draw(self, screen):
        """
        在窗口位图图形上进行绘制
```

```
        """
        screen.blit(self.image, self.rect)

    def move(self, speed):
        """
        移动
        """
        # 加快移动速度
        if speed[1] == 1:                                    # 下
            speed[1] += 1
        elif speed[0] == 1:                                  # 右
            speed[0] += 1
        elif speed[0] == -1:                                 # 左
            speed[0] += -1
        self.speed = speed
        self.rect.move_ip(*self.speed)
        if self.speed[0] != 0:                               # 如果左右移动
            # 左右相邻，移动的距离正好为一个水果方块的宽度
            if abs(self.rect.left - self.init_position[0]) - 1 == self.rect[2]:
                self.init_position = self.rect.topleft
                self.speed = [0, 0]
        else:                                                # 上下移动
            # 上下相邻
            if abs(self.rect.top - self.init_position[1]) - 1 == self.rect[3]:
                self.init_position = self.rect.topleft
                self.speed = [0, 0]
```

（2）在 entity.py 文件中创建一个 Font_Fact 类，其继承自 pygame.sprite.Sprite 类，该类用来渲染游戏页面中用到的文本，实现代码如下：

```
class Font_Fact(pygame.sprite.Sprite):
    """ 绘制文字精灵组件类 """
    again_game_posi = (620, 160)                             # 再来一局文本坐标位置
    quit_game_posi = (620, 400)                              # 退出游戏文本坐标位置
    show_time_posi = (59, 46)                                # 倒计时时间文本坐标位置

    def __init__(self, text, posi, txt_size=13, txt_color=(255, 255, 255)):
        """
        文本初始化方法
        :param text:   要向页面中渲染的文本
        :param posi:   文本的位置（左上顶点）
        :param txt_size:   文本的字体大小
        :param txt_color:  文本的颜色
        """
        super(Font_Fact, self).__init__()
        self.posi = posi
        self.image = pygame.font.Font("static/font/zhengqingke.ttf", txt_size)
        self.image = self.image.render(text, False, txt_color)
        self.rect = self.image.get_rect()
        self.rect.topleft = posi

    def draw(self, screen):
        """ 绘制方法 """
        screen.blit(self.image, self.rect)
```

3.5.2 游戏首屏页面的实现

实现水果消消乐游戏的首屏页面时，首先需要设定窗口中的各项参数，创建跟踪时间对象，并初始化设备和字体模块。然后需要创建 pygame 游戏窗体，并加载背景图片。最后监听鼠标事件，判断是否关闭窗

口。具体实现步骤如下。

（1）在 core 包中创建一个 first_eye.py 文件，其中首先导入 pygame 库，以及 entity.py 文件中的精灵类，代码如下：

```
import pygame
from .base import Base
from .entity import Element, Font_Fact
from .handler import Manager
```

“from .base import Base”中的‘.’表示当前文件所在的目录。

（2）在 first_eye.py 文件中创建一个 Screen_Manager 类，并继承自 base.py 文件中的 Base 类，在该类的__init__()构造方法中，使用 super 关键字执行父类的同名构造方法，以进行窗体的初始化操作。代码如下：

```
def __init__(self):
    # 执行父类的初始化构造方法
    super(Screen_Manager, self).__init__()
```

（3）创建 open_game_init()方法，用来绘制首页的背景图片和“开始游戏”按钮图片，代码如下：

```
def open_game_init(self):
    """ 游戏首屏页面初始化 """
    # 绘制首屏的背景图片
    Element(Element.bg_open_image, (0, 0)).draw(self.screen)
    # 开始按钮的绘制
    Element(Element.game_start_button_image, Element.game_start_button_posi).draw(self.screen)
    pygame.display.flip()
```

（4）创建 mouse_select()方法，用来监听用户的鼠标单击事件，并根据“开始游戏”按钮的左上角坐标判断单击的是否为“开始游戏”按钮。如果是，则改变游戏的 status 状态值为 1，以便使程序进入游戏页面。mouse_select()方法的实现代码如下：

```
def mouse_select(self, event):
    """ 游戏首屏事件监听  """
    if self.status == 0:
        if event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x, mouse_y = event.pos
            # 开始游戏按钮监听
            if Element.game_start_button_posi[0] < mouse_x <
                Element.game_start_button_posi[0] + Element.start_button[0] and
                    Element.game_start_button_posi[1] < mouse_y <
                    Element.game_start_button_posi[1] + Element.start_button[1]:
                Base.status = 1                                    # 更改游戏的状态
```

（5）在游戏主函数 main()中实例化 first_eye.py 文件中的 Screen_Manager 类，并在主逻辑循环中调用首屏页面的绘制方法和事件监听方法。main.py 文件修改后的代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager

def main():
    """
    消消乐游戏主函数
    """
    pygame.init()                                                  # 设备的检测
    pygame.font.init()                                             # 字体文件的初始化
```

```
# 在这里创建每一个页面类的对象
mr = Screen_Manager()                                    # 实例化首屏页面管理对象
while 1:
    # 在这里判断不同的游戏状态，从而执行不同的操作
    if mr.status == 0:                                   # 判断游戏首屏状态
        mr.open_game_init()
    for event in pygame.event.get():                     # 事件的监听与循环
        # 用户敲击键盘的键盘事件判断
        if event.type == KEYDOWN:
            # 判断用户按下 Q 键或者 Esc 键
            if event.key == K_q or event.key == K_ESCAPE:
                sys.exit()
        # 用户关闭游戏窗口的鼠标事件判断
        if event.type == QUIT:
            sys.exit()

        # 在这里进行不同页面的事件监听，从而改变游戏状态
        mr.mouse_select(event)                           # 对游戏首屏的事件监听
```

游戏首屏页面效果如图 3.5 所示。

图 3.5　游戏首屏页面

3.5.3　游戏页面的实现

水果消消乐游戏的游戏页面中，主要是对背景图片、退出主页面按钮、显示积分面板，以及游戏运行时用到的水果等元素进行初始化。其中，游戏中的水果设计为 9×9 矩阵的形式，每一个水果占据一个规格为 50×50（单位：像素）的小方块。另外，一共有 6 种水果，用数字（0～5）表示，它们在矩阵中随机分布。实现游戏页面的步骤如下。

（1）在 core 包中创建一个 handler.py 文件，并在其中导入 pygame 模块、pygame 常量库、精灵类、time、random、base.py 文件中的 Base 类等。代码如下：

```
import pygame
from pygame.locals import *
import time
import random
from functools import lru_cache                          # 缓存相关
from .base import Base
from .entity import Element, Font_Fact
```

（2）在 handler.py 文件中创建一个 Manager 类，使其继承自 base.py 文件中的 Base 类，在该类的初始

化方法__init__()中定义水果矩阵，用于记录不同水果的二维列表变量。代码如下：

```
class Manager(Base):
    """
    游戏主页面的管理
    """
    stop_width = 63                                    # 正方形退出按钮的边长
    reset_layout = True                                # 重新布局元素的标志
    cur_sel = [-1, -1]                                 # 当前选中的小方块，值为矩阵索引
    score = 0                                          # 游戏得分

    def __init__(self):
        super(Manager, self).__init__()
        # 水果矩阵：存储每个小方块中所要绘制的水果的编号（0～5）
        # -1 代表不绘制，-2 代表将要消除
        self.animal = [[-1 for i in range(self._width)] for j in range(self._height)]
```

（3）在 base.py 文件中的 Base 类中定义 4 个类变量，用于初始化矩阵参数，代码如下：

```
_cell_size = 50                                        # 矩阵中每个小方块为边长为 50 的正方形
_width = 9                                             # 矩阵的行数
_height = 9                                            # 矩阵的列数
matrix_topleft = (250, 100)                            # 矩阵的左上顶点坐标
```

（4）在 Manager 类中定义 self.cell_xy(self, row, col)和 self.xy_cell(self, x, y)两个方法，分别用来对指定方块在页面中左上顶点的坐标和该方块在矩阵中的索引进行相互转换，代码如下：

```
@lru_cache(None)                                       # 必须要有一个参数，None 代表不限
def cell_xy(self, row, col):
    """ 矩阵索引转换为坐标 """
    return int(Base.matrix_topleft[0] + col * Base._cell_size),
           int(Base.matrix_topleft[1] + row * Base._cell_size)

@lru_cache(None)
def xy_cell(self, x, y):
    """ 坐标转换位矩阵索引 """
    return int((y - Base.matrix_topleft[1]) / Base._cell_size),
           int((x - Base.matrix_topleft[0]) / Base._cell_size)
```

（5）在 Manager 类中创建一个 start_game_init()方法，用来初始化游戏页面中的所有图片和文本。另外，该方法需要返回水果矩阵中所有水果的水果精灵组。start_game_init()方法代码如下：

```
def start_game_init(self):
    """ 游戏页面绘制  """
    # 绘制页面背景
    Element(Element.bg_start_image, (0, 0)).draw(self.screen)
    # 绘制暂停键
    Element(Element.stop, Element.stop_position).draw(self.screen)
    # 绘制分数显示板
    score_board = Element(Element.board_score, Element.score_posi)
    score_board.draw(self.screen)
    # 绘制游戏分数
    str_score = str(self.score)
    for k, sing in enumerate(str_score):
        Element(Element.score[int(sing)], (755 + k * 32, 40)).draw(self.screen)
    # 创建小方块背景图片精灵组
    BrickSpriteGroup = pygame.sprite.Group()
    # 创建水果精灵组
    AnimalSpriteGroup = pygame.sprite.Group()
    # 向精灵组中添加精灵
    for row in range(self._height):
        for col in range(self._width):
```

```
                x, y = self.cell_xy(row, col)
                BrickSpriteGroup.add(Element(Element.brick, (x, y)))
                if self.animal[row][col] != -2:
                    AnimalSpriteGroup.add(Element(Element.animal[self.animal[row][col]], (x, y)))
        # 绘制小方块的背景图
        BrickSpriteGroup.draw(self.screen)
        # 绘制小方块中的水果
        for ani in AnimalSpriteGroup:
            self.screen.blit(ani.image, ani.rect)
        # 绘制鼠标所单击的水果的突出显示边框
        if self.cur_sel != [-1, -1]:
            frame_sprite = Element(Element.frame_image, self.cell_xy(self.cur_sel[0], self.cur_sel[1]))
            self.screen.blit(frame_sprite.image, frame_sprite.rect)
        pygame.display.flip()                                    # 更新页面显示，必须添加
        return AnimalSpriteGroup
```

（6）在 Manager 类中创建一个 mouse_select()方法，用于对游戏页面中的事件进行监听，代码如下：

```
def mouse_select(self, event):
    """  游戏事件监听  """
    if event.type == MOUSEBUTTONDOWN:                            # 鼠标按下事件
        mouse_x, mouse_y = event.pos                             # 获取当前鼠标的坐标
        if self.status == 1:
            # 判断单击的是水果
            if self.matrix_topleft[0] < mouse_x < self.matrix_topleft[0] + self._cell_size * self._width
                    and self.matrix_topleft[1] < mouse_y < self.matrix_topleft[1] + self._cell_size * self._height:
                mouse_selected = self.xy_cell(mouse_x, mouse_y)
                # 记录当前鼠标单击的小方块
                self.cur_sel = mouse_selected
            # 判断单击的是退出按钮，需注意此退出按钮的坐标为左上角
            elif Element.stop_position[0] < mouse_x < Element.stop_position[0] + Manager.stop_width
                    and Element.stop_position[1] < mouse_y < Element.stop_position[1] + Manager.stop_width:
                Base.status = 2
                self.reset_layout = True                         # 布局下一盘游戏的元素
            else:
                self.cur_sel = [-1, -1]                          # 处理无效的单击
```

（7）在 Manager 类中创建一个 reset_animal()方法，用于对矩阵中的小方块随机分配水果，代码如下：

```
def reset_animal(self):
    """ 对矩阵中的小方块随机分配水果 """
    if self.reset_layout:
        for i in range(self._height):
            for j in range(self._width):
                self.animal[i][j] = random.randint(0, 5)
    self.reset_layout = False
```

（8）在游戏主函数 main()中实例化 handler.py 文件中的 Manager 类，并在主逻辑循环中调用游戏页面的绘制方法和事件监听方法。main.py 文件进一步修改后的代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                                                # 设备的检测
    pygame.font.init()                                           # 字体文件的初始化
```

```
# 在这里创建每一个页面类的对象
mr = Screen_Manager()                                    # 实例化首屏页面管理对象
mg = Manager()                                           # 实例化游戏页面管理对象
while 1:
    # 在这里判断不同的游戏状态，从而执行不同的操作
    if mr.status == 0:                                   # 判断游戏首屏状态
        mr.open_game_init()
    if mg.status == 1:                                   # 判断游戏页面状态
        mg.reset_animal()                                # 随机分配元素
        # 绘制游戏页面
        AnimalSpriteGroup = mg.start_game_init()
    for event in pygame.event.get():                     # 事件的监听与循环
        # 用户敲击键盘的键盘事件判断
        if event.type == KEYDOWN:
            # 判断用户按下 Q 键或者 Esc 键
            if event.key == K_q or event.key == K_ESCAPE:
                sys.exit()
        # 用户关闭游戏窗口的鼠标事件判断
        if event.type == QUIT:
            sys.exit()

        # 在这里进行不同页面的事件监听，从而改变游戏状态
        mg.mouse_select(event)                           # 对游戏页面的事件监听
        mr.mouse_select(event)                           # 对游戏首屏的事件监听
```

3.5.4 可消除水果的检测与标记清除

水果消消乐游戏的规则为，当至少 3 个以上相同的水果成一条直线（横竖都可以）或者“L”“T”形状时，消除这些水果，消除不同形状线路上的水果可以获得不同的分数。因此，在实现时，首先需要判断任意一个水果在下、左、右这 3 个方向中的任意一个方向是否有 n（$n \geq 2$）个相同的水果与其自身连成一条直线。

说明

由于水果的遍历是从（0,0）开始的，因此不需要判断上方。

实现可消除水果检测与标记清除的步骤如下。

（1）在 core/handler.py 文件的 Manager 类中添加一个 destory_animal_num 变量，用来记录每次游戏中每种水果的消除数量，便于计算分数，代码如下：

```
# 消除水果列表：记录消除各水果的个数
destory_animal_num = [0, 0, 0, 0, 0, 0]
```

（2）在 core/handler.py 文件的 Manager 类中创建一个 clear_ele()方法，用于封装水果矩阵列表中可消除水果的检测代码，其具体实现代码如下：

```
def clear_ele(self):
    """ 清除标记元素，且上方元素向下掉落  """
    single_score = self.score
    self.change_value_sign = False
    # 从（0,0）位置遍历水果矩阵
    for i in range(self._height):
        for j in range(self._width):
            # 水平向右五连消
            if self.exist_right(i, j, 5):
                self.change_value_sign = True
                # 第三个位置向下，并存在垂直三连消
                if self.exist_down(i, j + 2, 3):
```

```
                # 记录消除的水果数量
                self.destory_animal_num[self.animal[i][j]] += 7
                # 对矩阵中消除的水果位置标记为-2，代表为消除状态
                self.change_right(i, j, 5)
                self.change_down(i, j + 2, 3)
            else:
                self.destory_animal_num[self.animal[i][j]] += 5
                self.change_right(i, j, 5)
        # 水平四连消
        elif self.exist_right(i, j, 4):
            self.change_value_sign = True
            # 第二个位置向下，并存在垂直三连消
            if self.exist_down(i, j + 1, 3):
                self.destory_animal_num[self.animal[i][j]] += 6
                self.change_right(i, j, 4)
                self.change_down(i, j + 1, 3)
            # 第一个位置向下，并存在垂直三连消
            elif self.exist_down(i, j, 3):
                self.destory_animal_num[self.animal[i][j]] += 6
                self.change_right(i, j, 4)
                self.change_down(i, j, 3)
            else:
                self.destory_animal_num[self.animal[i][j]] += 4
                self.change_right(i, j, 4)
        # 水平三连消
        elif self.exist_right(i, j, 3):
            self.change_value_sign = True
            if self.exist_down(i, j, 3):
                self.destory_animal_num[self.animal[i][j]] += 5
                self.change_right(i, j, 3)
                self.change_down(i, j, 3)
            elif self.exist_down(i, j + 1, 3):
                self.destory_animal_num[self.animal[i][j]] += 5
                self.change_right(i, j, 3)
                self.change_down(i, j + 1, 3)
            elif self.exist_down(i, j + 2, 3):
                self.destory_animal_num[self.animal[i][j]] += 5
                self.change_right(i, j, 3)
                self.change_down(i, j + 2, 3)
            else:
                self.destory_animal_num[self.animal[i][j]] += 3
                self.change_right(i, j, 3)
        # 垂直五连消
        elif self.exist_down(i, j, 5):
            self.change_value_sign = True
            # 第三个位置向右，并存在三连消
            if self.exist_right(i + 2, j, 3):
                self.destory_animal_num[self.animal[i][j]] += 7
                self.change_down(i, j, 5)
                self.change_right(i + 2, j, 3)
            # 第三个位置向左，并存在三连消
            elif self.exist_left(i + 2, j, 3):
                self.destory_animal_num[self.animal[i][j]] += 7
                self.change_down(i, j, 5)
                self.change_left(i + 2, j, 3)
            else:
                self.destory_animal_num[self.animal[i][j]] += 5
                self.change_down(i, j, 5)
        # 垂直四连消
        elif self.exist_down(i, j, 4):
            self.change_value_sign = True
            if self.exist_right(i + 1, j, 3):
```

```
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 4)
                    self.change_right(i + 1, j, 3)
                elif self.exist_left(i + 1, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 4)
                    self.change_left(i + 1, j, 3)
                elif self.exist_right(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 4)
                    self.change_right(i + 2, j, 3)
                elif self.exist_left(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 4)
                    self.change_left(i + 2, j, 3)
                else:
                    self.destory_animal_num[self.animal[i][j]] += 4
                    self.change_down(i, j, 4)
            # 垂直三连消
            elif self.exist_down(i, j, 3):
                self.change_value_sign = True
                if self.exist_right(i + 1, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 5
                    self.change_down(i, j, 3)
                    self.change_right(i + 1, j, 3)
                elif self.exist_left(i + 1, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 5
                    self.change_down(i, j, 3)
                    self.change_left(i + 1, j, 3)
                elif self.exist_right(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 5
                    self.change_down(i, j, 3)
                    self.change_right(i + 2, j, 3)
                elif self.exist_left(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 5
                    self.change_down(i, j, 3)
                    self.change_left(i + 2, j, 3)
                elif self.exist_left(i + 2, j, 2) and \
                        self.exist_right(i + 2, j, 2):
                    self.destory_animal_num[self.animal[i][j]] += 5
                    self.change_down(i, j, 3)
                    self.change_left(i + 2, j, 2)
                    self.change_right(i + 2, j, 2)
                elif self.exist_left(i + 2, j, 2) and \
                        self.exist_right(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 3)
                    self.change_left(i + 2, j, 2)
                    self.change_right(i + 2, j, 3)
                elif self.exist_left(i + 2, j, 3) and \
                        self.exist_right(i + 2, j, 2):
                    self.destory_animal_num[self.animal[i][j]] += 6
                    self.change_down(i, j, 3)
                    self.change_left(i + 2, j, 3)
                    self.change_right(i + 2, j, 2)
                elif self.exist_left(i + 2, j, 3) and \
                        self.exist_right(i + 2, j, 3):
                    self.destory_animal_num[self.animal[i][j]] += 7
                    self.change_down(i, j, 3)
                    self.change_left(i + 2, j, 3)
                    self.change_right(i + 2, j, 3)
                else:
```

```
                    self.destory_animal_num[self.animal[i][j]] += 3
                    self.change_down(i, j, 3)
    return self.change_value_sign
```

（3）在 core/handler.py 文件的 Manager 类中定义 3 个方法，分别为 exist_right()、exist_down()和 exist_left()，它们分别用于判断某一个水果的右边、下边、左边是否存在与其自身同类型的 num-1 个水果。exist_right()、exist_down()和 exist_left()方法的实现代码如下：

```
def exist_right(self, row, col, num):
    """ 判断 self.animal[i][j]元素右边是否存在与其自身图像相同的 num - 1 个图像 """
    if col <= self._width - num:
        for item in range(num):
            if self.animal[row][col] != self.animal[row][col + item] or self.animal[row][col] == -2:
                break
        else:
            return True
        return False
    else:
        return False

def exist_down(self, row, col, num):
    """ 判断 self.animal[i][j]元素下方是否存在与其自身图像相同的 num - 1 个图像 """
    if row <= self._height - num:
        for item in range(num):
            if self.animal[row][col] != self.animal[row + item][col] or self.animal[row][col] == -2:
                break
        else:
            return True
        return False
    else:
        return False

def exist_left(self, row, col, num):
    """ 判断 self.animal[i][j]元素左边是否存在与其自身图像相同的 num - 1 个图像 """
    if col >= num - 1:
        for item in range(num):
            if self.animal[row][col] != self.animal[row][col - item] or self.animal[row][col] == -2:
                break
        else:
            return True
        return False
    else:
        return False
```

（4）在 core/handler.py 文件的 Manager 类中定义 3 个方法，分别为 change_right()、change_down()和 change_left()，它们分别用于改变某一个水果右边、下边、左边 num 个水果的状态为消除状态。change_right()、change_down()和 change_left()方法的实现代码如下：

```
def change_right(self, row, col, num):
    """ 改变当前水果及右边的 num 个水果为消除状态 """
    for item in range(num):
        self.animal[row][col + item] = -2

def change_down(self, row, col, num):
    for item in range(num):
        self.animal[row + item][col] = -2

def change_left(self, row, col, num):
    for item in range(num):
        self.animal[row][col - item] = -2
```

（5）在游戏主函数 main()中调用可消除水果的检测方法 clear_ele()，实现水果消除的效果，修改后的主

函数 main()代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                                   # 设备的检测
    pygame.font.init()                              # 字体文件的初始化

    # 在这里创建每一个页面类的对象
    mr = Screen_Manager()                           # 实例化首屏页面管理对象
    mg = Manager()                                  # 实例化游戏页面管理对象

    while 1:
        # 在这里判断不同的游戏状态，从而执行不同的操作
        if mr.status == 0:                          # 判断游戏首屏状态
            mr.open_game_init()
        if mg.status == 1:                          # 判断游戏页面状态
            mg.reset_animal()                       # 随机分配元素
            # 绘制游戏页面
            AnimalSpriteGroup = mg.start_game_init()
            mg.clear_ele()                          # 标记清除
        for event in pygame.event.get():            # 事件的监听与循环
            # 用户敲击键盘的键盘事件判断
            if event.type == KEYDOWN:
                # 判断用户按下 Q 键或者 Esc 键
                if event.key == K_q or event.key == K_ESCAPE:
                    sys.exit()
            # 用户关闭游戏窗口的鼠标事件判断
            if event.type == QUIT:
                sys.exit()

            # 在这里进行不同页面的事件监听，从而改变游戏状态
            mg.mouse_select(event)                  # 对游戏页面的事件监听
            mr.mouse_select(event)                  # 对游戏首屏的事件监听
```

运行程序，单击游戏首屏上的 Play 按钮进入游戏页面时，原始水果矩阵列表中默认可消除的水果会自动被消除，效果如图 3.6 所示。

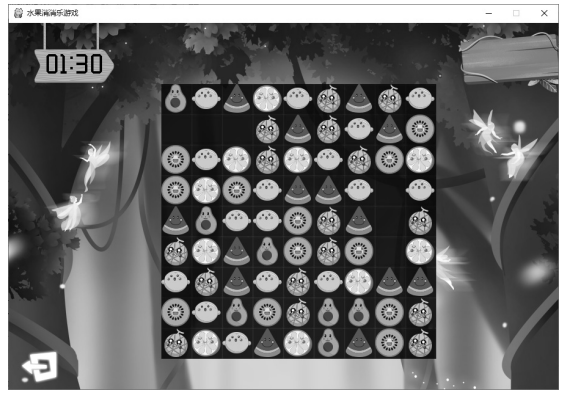


图 3.6　检测消除为空

3.5.5　水果的掉落

当水果矩阵列表中有水果被消除时，首先在要消除的水果方块中绘制消除动画，然后将每一个被消除水果所在列的上方所有水果依次向下移动，最终在该列的最上方会产生一个空缺，在此空缺处随机产生一个水果，这样即可继续进行游戏。

实现水果消消乐游戏中水果掉落功能的步骤如下。

（1）在 core/handler.py 文件的 Manager 类中创建一个 drop_animal()方法，用于实现水果掉落的功能。

该方法中，首先定义一个 position 列表，用于存储水果矩阵列表中每一个被消除水果所在的小方块的左上角坐标（x, y），然后判断如果 position 列表不为空，则在 position 列表所存储的每一个小方块处绘制消除动画，并按照从上到下的顺序依次移动每一行中被消除水果上方的所有水果。drop_animal()方法实现代码如下：

```
def drop_animal(self):
    """ 水果掉落函数 """
    clock = pygame.time.Clock()
    position = []                                          # 水果矩阵中要消除的水果列表
    for i in range(self._width):
        for j in range(self._height):
            if self.animal[i][j] == -2:
                x, y = self.cell_xy(i, j)
                position.append((x, y))
    # 绘制消除小方块的消除动画效果
    if position != []:
        for index in range(0, 9):
            # clock.tick(40)
            for pos in position:
                Element(Element.brick, pos).draw(self.screen)
                Element(Element.bling[index], (pos[0], pos[1])).draw(self.screen)
                pygame.display.flip()
    for i in range(self._width):
        # 此行之上所有要掉落的水果的背景图片列表
        brick_position = []
        # 此行之上所有要掉落的水果列表
        fall_animal_list = []
        speed = [0, 1]
        for j in range(self._height):
            if self.animal[i][j] == -2:
                x, y = self.cell_xy(i, j)
                brick_position.append((x, y))
                for m in range(i, -1, -1):
                    if m == 0:                             # 此列中最上方的水果（补缺）
                        self.animal[m][j] = random.randint(0, 5)
                    else:
                        x, y = self.cell_xy(m - 1, j)
                        brick_position.append((x, y))
                        animal = Element(Element.animal[self.animal[m - 1][j]], (x, y))
                        fall_animal_list.append(animal)
                        # 在水果矩阵列表中交换上下两个水果
                        self.animal[m][j] = self.animal[m - 1][j]
        # 使所消除的小方块的上方的小方块向下移动
        while speed != [0, 0] and fall_animal_list != []:
            # 绘制水果的背景图片
            for position in brick_position:
                Element(Element.brick, position).draw(self.screen)
            # 向下移动水果
            for animal_sprite in fall_animal_list:
                animal_sprite.move(speed)
                animal_sprite.draw(self.screen)
                speed = animal_sprite.speed
            pygame.display.flip()
```

（2）在可消除水果的检测方法 clear_ele()中调用 drop_animal()方法，代码如下：

```
self.drop_animal()                                         # 调用水果掉落方法
```

（3）在 Manager 类中定义一个 every_animal_score 列表，用来存储游戏得分规则，代码如下：

```
# 计分规则列表：消除每一类水果所得的分数
every_animal_score = [1, 2, 1, 1, 2, 1]
```

（4）在 Manager 类中创建一个 cal_score()方法，用于统计当前获得的分数，代码如下：

```
def cal_score(self, destory_animal_num):
    """ 统计当前分数 """
    self.score = 0
    for k, num in enumerate(destory_animal_num):
        self.score += self.every_animal_score[k] * num
```

（5）在 Manager 类的 clear_ele()方法中调用统计游戏分数的 cal_score()方法，并在水果矩阵每次消除完水果后，根据每次消除所得总分数在页面中绘制不同的鼓励性话语。Manager 类中 clear_ele()方法修改后的代码如下（注意：加粗的代码为新增代码）：

```
def clear_ele(self):
    """ 清除标记元素，且上方元素向下掉落 """
    single_score = self.score
    self.change_value_sign = False
    for i in range(self._height):
        for j in range(self._width):
            …                                          # 此处代码忽略

    self.drop_animal()                                 # 调用水果掉落方法

    self.cal_score(self.destory_animal_num)            # 计算分数

    # 根据此次鼠标单击交换获得的分数，绘制不同的鼓励性语句
    single_score = self.score - single_score

    if single_score < 5:
        pass
    elif single_score < 8:                             # 绘制 Good
        Element(Element.single_score[0], (350, 250)).draw(self.screen)
        pygame.display.flip()
        pygame.time.delay(500)
    elif single_score < 10:                            # 绘制 Great
        Element(Element.single_score[1], (350, 250)).draw(self.screen)
        pygame.display.flip()
        pygame.time.delay(500)
    elif single_score < 15:                            # 绘制 Amazing
        Element(Element.single_score[2], (350, 250)).draw(self.screen)
        pygame.display.flip()
        pygame.time.delay(500)
    elif single_score < 20:                            # 绘制 Excellent
        Element(Element.single_score[3], (350, 250)).draw(self.screen)
        pygame.display.flip()
        pygame.time.delay(500)
    elif single_score >= 20:                           # 绘制 Unbelievable
        PlaySound()
        Element(Element.single_score[4], (350, 250)).draw(self.screen)
        pygame.display.flip()
        pygame.time.delay(500)
    return self.change_value_sign
```

3.5.6 单击相邻水果时的交换

实现鼠标单击水果与相邻水果进行交换的步骤：首先定义一个交换开关，每单击一次水果，都记录当前单击的水果的矩阵索引；然后与上次单击的水果的矩阵索引进行比较以判断是否相邻，如果相邻，则开启交换开关，交换前后单击的两个水果；否则，更新代表前一次单击的水果的矩阵索引的值指向为当前单击的水果的矩阵索引。

（1）在 core/handler.py 文件的 Manager 类中定义一个 exchange_status 变量，用来作为交换开关；定义

一个 last_sel 列表，用来记录上一次鼠标单击的水果的矩阵索引。代码如下：

```
# 交换的标志，1: 代表交换，-1: 不交换
exchange_status = -1
# 记录上一次选中的水果，值为水果索引
last_sel = [-1, -1]
```

（2）在游戏主页面的事件监听方法 mouse_select()中记录当前单击的水果的矩阵索引，并将其与上一次单击的水果的矩阵索引相比较，判断两次单击的水果是否相邻，如果相邻，则开启交换开关。Manager 类的 mouse_select()方法修改后的代码如下（注意：加粗的代码为新增代码）：

```
def mouse_select(self, event):
    """  游戏事件监听  """
    if event.type == MOUSEBUTTONDOWN:                    # 鼠标按下事件
        mouse_x, mouse_y = event.pos                     # 获取当前鼠标的坐标
        if self.status == 1:
            # 判断单击的是水果
            if self.matrix_topleft[0] < mouse_x < self.matrix_topleft[0] + self._cell_size * self._width
                    and self.matrix_topleft[1] < mouse_y < self.matrix_topleft[1] + self._cell_size * self._height:
                mouse_selected = self.xy_cell(mouse_x, mouse_y)
                # 记录当前鼠标单击的小方块
                self.cur_sel = mouse_selected

                # 判断前后单击的两个水果是否相邻
                if (self.last_sel[0] == self.cur_sel[0] and abs(self.last_sel[1] - self.cur_sel[1]) == 1) or
                        (self.last_sel[1] == self.cur_sel[1] and abs(self.last_sel[0] - self.cur_sel[0]) == 1):
                    self.exchange_status = 1             # 确定相邻，交换值

            # 判断单击的是退出按钮，需注意此退出按钮的坐标为左上角
            elif Element.stop_position[0] < mouse_x < Element.stop_position[0] + Manager.stop_width
                    and Element.stop_position[1] < mouse_y < Element.stop_position[1] + Manager.stop_width:
                Base.status = 2
                self.reset_layout = True                 # 布局下一盘游戏的元素
            else:
                self.cur_sel = [-1, -1]                  # 处理无效的单击
```

（3）在 Manager 类中创建一个 exchange_ele()方法，用来实现水果的位置交换功能，该方法有一个参数，表示水果精灵组。具体实现时，当判断交换开关为开启时，首先获取两个水果所在小方块的左上顶点的坐标，根据获取的坐标值判断相邻的两个水果的相邻方向，遍历精灵组，找到这两个水果的 Surface 对象；然后设置这两个 Sruface 对象的 speed 属性，以便移动这两个对象，从而达到交换水果位置的功能。exchange_ele()方法实现代码如下：

```
def exchange_ele(self, AnimalSpriteGroup):
    """  交换鼠标前后单击的两个元素  """
    if self.exchange_status == -1:
        self.last_sel = self.cur_sel
    if self.exchange_status == 1:
        last_x, last_y = self.cell_xy(*self.last_sel)
        cur_x, cur_y = self.cell_xy(*self.cur_sel)
        # 左右相邻
        if self.last_sel[0] == self.cur_sel[0]:          # 比较矩阵索引
            for animal_sur in AnimalSpriteGroup:
                if animal_sur.rect.topleft == (last_x, last_y):
                    last_sprite = animal_sur
                    last_sprite.speed = [self.cur_sel[1] -  self.last_sel[1], 0]
                if animal_sur.rect.topleft == (cur_x, cur_y):
                    cur_sprite = animal_sur
                    cur_sprite.speed = [self.last_sel[1] -  self.cur_sel[1], 0]
        # 上下相邻
        elif self.last_sel[1] == self.cur_sel[1]:
```

```
                for animal_sur in AnimalSpriteGroup:
                    if animal_sur.rect.topleft == (last_x, last_y):
                        last_sprite = animal_sur
                        last_sprite.speed = [0, self.cur_sel[0] -  self.last_sel[0]]
                    if animal_sur.rect.topleft == (cur_x, cur_y):
                        cur_sprite = animal_sur
                        cur_sprite.speed = [0, self.last_sel[0] -  self.cur_sel[0]]
            # 移动水果
            while last_sprite.speed != [0, 0]:
                pygame.time.delay(5)
                Element(Element.brick, (last_x, last_y)).draw(self.screen)
                Element(Element.brick, (cur_x, cur_y)).draw(self.screen)
                last_sprite.move(last_sprite.speed)
                cur_sprite.move(cur_sprite.speed)
                last_sprite.draw(self.screen)
                cur_sprite.draw(self.screen)
                pygame.display.flip()

            self.change_value()                          # 交换水果值
            if not self.clear_ele():                     # 交换后，若不存在消除，则归位
                self.change_value()
            self.exchange_status = -1                    # 关闭交换
            self.cur_sel = [-1, -1]                      # 保证每次交换的两个元素不存在交叉
```

（4）上面代码中用到了一个 change_value()方法，该方法定义在 Manager 类中，用于交换相邻两个水果的矩阵值，代码如下：

```
def change_value(self):
    """ 交换水果的矩阵值 """
    temp = self.animal[self.last_sel[0]][self.last_sel[1]]
    self.animal[self.last_sel[0]][self.last_sel[1]] = self.animal[self.cur_sel[0]][self.cur_sel[1]]
    self.animal[self.cur_sel[0]][self.cur_sel[1]] = temp
```

（5）在游戏主函数 main()的游戏主逻辑循环中调用实现相邻两水果交换的 exchange_ele()方法。修改后的主函数 main()代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                                        # 设备的检测
    pygame.font.init()                                   # 字体文件的初始化

    # 在这里创建每一个页面类的对象
    mr = Screen_Manager()                                # 实例化首屏页面管理对象
    mg = Manager()                                       # 实例化游戏页面管理对象

    while 1:
        # 在这里判断不同的游戏状态，从而执行不同的操作
        if mr.status == 0:                               # 判断游戏首屏状态
            mr.open_game_init()
        if mg.status == 1:                               # 判断游戏页面状态
            mg.reset_animal()                            # 随机分配元素
            # 绘制游戏页面
            AnimalSpriteGroup = mg.start_game_init()
```

```
            mg.clear_ele()                          # 标记清除
            mg.exchange_ele(AnimalSpriteGroup)      # 元素交换

        for event in pygame.event.get():            # 事件的监听与循环
            # 用户敲击键盘的键盘事件判断
            if event.type == KEYDOWN:
                # 判断用户按下 Q 键或者 Esc 键
                if event.key == K_q or event.key == K_ESCAPE:
                    sys.exit()
            # 用户关闭游戏窗口的鼠标事件判断
            if event.type == QUIT:
                sys.exit()

            # 在这里进行不同页面的事件监听，从而改变游戏状态
            mg.mouse_select(event)                  # 对游戏页面的事件监听
            mr.mouse_select(event)                  # 对游戏首屏的事件监听
```

相邻水果交换的示意图如图 3.7 所示。

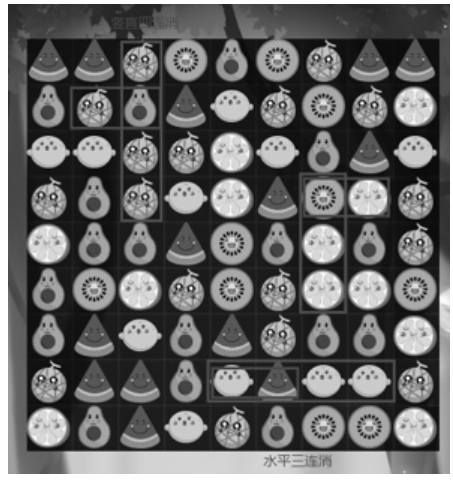


图 3.7　相邻水果交换示意图

3.5.7　“死图”状态的判断

在每次消除完可消除的水果，并且上方的水果降落后，如果没有水果能够再通过单击交换的方式进行消除，则称为“死图”状态。如果是“死图”状态，则当前游戏会自动退出，并进入排行榜页面。

实现水果消消乐游戏“死图”状态判断功能的步骤如下。

（1）在 core/handler.py 文件的 Manager 类中定义一个 death_sign 变量，用来标识是否为“死图”状态，初始值为 True，标识默认为“死图”。代码如下：

```
death_sign =  True                                  # 死图标识
```

（2）在 Manager 类中创建一个 is_death_map()方法，用于检测水果矩阵当前是否为“死图”状态。该方法中，只要检测到有可以通过交换相邻水果进行消除的情况，就关闭“死图”标识开关。is_death_map()方法实现代码如下：

```
def is_death_map(self):
    """ 判断当前水果矩阵是否为死图  """
    for i in range(self._width):
        for j in range(self._height):
            # 边界判断
            if i >= 1 and j >= 1 and i <= 7 and j <= 6:
                if self.animal[i][j] == self.animal[i][j + 1]:
                    """e      b
                          e e
                       b      e
                    """
                if (self.animal[i][j] in [self.animal[i - 1][j - 1], self.animal[i + 1][j - 1]] and self.animal[i][j - 1] != -1)
                    or (self.animal[i][j] in [self.animal[i - 1][j + 2], self.animal[i + 1][j + 2]] and self.animal[i][j + 2] != -1):
                    self.death_sign = False
                    break
            if i >= 1 and j >= 1 and i <= 6 and j <= 7:
                if self.animal[i][j] == self.animal[i + 1][j]:
                    if (self.animal[i][j] in [self.animal[i - 1][j - 1], self.animal[i - 1][j + 1]] and self.animal[i - 1][j] != -1)
                    or (self.animal[i][j] in [self.animal[i + 2][j - 1], self.animal[i + 2][j + 1]] and self.animal[i + 2][j] != -1):
                        """e    b
                              e
```

```
                                e
                            c    e"""
                        self.death_sign = False
                        break
            elif i >= 1 and j >= 1 and i <= 7 and j <= 7:
                if self.animal[i - 1][j - 1] == self.animal[i][j]:
                    if (self.animal[i][j] == self.animal[i - 1][j + 1] and self.animal[i - 1][j] != -1)
                            or (self.animal[i][j] == self.animal[i + 1][j - 1] and self.animal[i][j - 1] != -1):
                        """e    e        e    b
                            e            e
                            c            e     """
                        self.death_sign = False
                        break

                if self.animal[i][j] == self.animal[i + 1][j + 1]:
                    if (self.animal[i][j] == self.animal[i - 1][j + 1] and self.animal[i][j + 1] != -1)
                            or (self.animal[i][j] == self.animal[i + 1][j - 1] and self.animal[i + 1][j] != -1):
                        """     e            b
                            e            e
                            b    e        e    e"""
                        self.death_sign = False
                        break
```

（3）如果判定为“死图”状态，则结束本场游戏，进入排行榜页面。在 Manager 类中创建一个 stop_game() 方法，封装结束本场游戏的相关逻辑，代码如下：

```
def stop_game(self):
    """ 结束本场游戏，进入排行榜的页面  """
    if self.status == 1:
        if self.death_sign:
            pygame.time.delay(500)
            Element(Element.none_animal,Element.none_animal_posi).draw(self.screen)
            pygame.display.flip()
            pygame.time.delay(500)
            Base.status = 2                              # 结束本场游戏
            self.reset_layout = True                     # 布局下一盘游戏的元素
            self.time_is_over = False
        else:
            self.death_sign = True
```

（4）在主函数 main()中的游戏主逻辑循环中调用 is_death_map()和 stop_game()方法，实现游戏的“死图”状态判断功能。修改后的主函数 main()代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                                        # 设备的检测
    pygame.font.init()                                   # 字体文件的初始化

    # 在这里创建每一个页面类的对象
    mr = Screen_Manager()                                # 实例化首屏页面管理对象
    mg = Manager()                                       # 实例化游戏页面管理对象
```

```
while 1:
    # 在这里判断不同的游戏状态，从而执行不同的操作
    if mr.status == 0:                          # 判断游戏首屏状态
        mr.open_game_init()
    if mg.status == 1:                          # 判断游戏页面状态
        mg.reset_animal()                       # 随机分配元素
        # 绘制游戏页面
        AnimalSpriteGroup = mg.start_game_init()
        mg.clear_ele()                          # 标记清除
        mg.is_death_map()                       # 死图判断
        mg.exchange_ele(AnimalSpriteGroup)      # 元素交换
    mg.stop_game()                              # 结束游戏

    for event in pygame.event.get():            # 事件的监听与循环
        # 用户敲击键盘的键盘事件判断
        if event.type == KEYDOWN:
            # 判断用户按下 Q 键或者 Esc 键
            if event.key == K_q or event.key == K_ESCAPE:
                sys.exit()
        # 用户关闭游戏窗口的鼠标事件判断
        if event.type == QUIT:
            sys.exit()

        # 在这里进行不同页面的事件监听，从而改变游戏状态
        mg.mouse_select(event)                  # 对游戏页面的事件监听
        mr.mouse_select(event)                  # 对游戏首屏的事件监听
```

3.5.8 游戏倒计时的实现

消消乐游戏中的游戏倒计时功能是通过 pygame 中监控时间的 time 子模块来实现的。当游戏规定时间到时会自动出现如图 3.8 所示的效果。

图 3.8　游戏超时运行效果图

水果消消乐游戏的倒计时功能实现步骤如下。

（1）在 core/handler.py 文件的 Manager 类中分别定义 start_time、end_time、running_time 和 time_is_over 这 4 个变量和一个 TIMEOUT 常量，分别用于表示每场游戏的开始时间、结束时间、运行时间、游戏是否超时的开关和超时时间，代码如下：

```
start_time = 0                                  # 每场游戏的开始时间
end_time = 0                                    # 每场游戏的结束时间
running_time = 0                                # 每场游戏的运行时间（毫秒）
time_is_over = False                            # 每场游戏的时间状态控制
TIMEOUT = 1000 * 60 * 3                         # 每场游戏的规定时长
```

（2）在 Manager 类中定义一个 judge_time()方法，用于判断游戏是否超时，如果超时，则开启游戏超时开关。代码如下：

```
def judge_time(self):
    """ 判断游戏时间是否超时 """
    self.end_time = pygame.time.get_ticks()          # 更新结束时间
    # 避免在 self.status = 2 的情况下更新 self.end_time，而使 self.time_is_over == True
    if self.status == 1:
        self.running_time = self.end_time - self.start_time
        if self.running_time >= self.TIMEOUT:
            self.time_is_over = True
```

（3）在每一局游戏开始时（即游戏的状态变量 Manager.status 值为 1），更新每一局游戏的开始时间，该位置有两处：在游戏首屏页面单击 Play 按钮时、在排行榜页面中单击“再来一局”按钮时，即分别在 Screen_Manager 类中的 mouse_select()方法和 Score_Manager 类中的 mouse_select()方法中添加如下代码：

```
# 初始化游戏的开始时间
Manager.start_time = pygame.time.get_ticks()
```

（4）在主函数 main()中的游戏主逻辑循环中调用 judge_time()方法，修改后的主函数 main()代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager

def main():
    """
    消消乐游戏主函数
    :return:
    """
    pygame.init()                                # 设备的检测
    pygame.font.init()                           # 字体文件的初始化

    # 在这里创建每一个页面类的对象
    mr = Screen_Manager()                        # 实例化首屏页面管理对象
    mg = Manager()                               # 实例化游戏页面管理对象

    while 1:
        mg.judge_time()                          # 判断游戏超时
        # 在这里判断不同的游戏状态，从而执行不同的具体操作
        if mr.status == 0:                       # 判断游戏首屏状态
            mr.open_game_init()
        if mg.status == 1:                       # 判断游戏页面状态
            mg.reset_animal()                    # 随机分配元素
            # 绘制游戏页面
            AnimalSpriteGroup = mg.start_game_init()
            mg.clear_ele()                       # 标记清除
            mg.is_death_map()                    # 死图判断
            mg.exchange_ele(AnimalSpriteGroup)   # 元素交换
        mg.stop_game()                           # 结束游戏

        for event in pygame.event.get():         # 事件的监听与循环
```

```
            # 用户敲击键盘的键盘事件判断
            if event.type == KEYDOWN:
                # 判断用户按下 Q 键或者 Esc 键
                if event.key == K_q or event.key == K_ESCAPE:
                    sys.exit()
            # 用户关闭游戏窗口的鼠标事件判断
            if event.type == QUIT:
                sys.exit()

            # 在这里进行不同页面的事件监听，从而改变游戏状态
            mg.mouse_select(event)                    # 对游戏页面的事件监听
            mr.mouse_select(event)                    # 对游戏首屏的事件监听
```

（5）在游戏主页面的绘制方法（Manager 类的 start_game_init()方法）中添加绘制时间的代码，修改后的 start_game_init()方法代码如下（注意：加粗的代码为新增代码）：

```
def start_game_init(self):
    """ 游戏页面绘制  """
    # 绘制游戏页面的背景
    Element(Element.bg_start_image, (0, 0)).draw(self.screen)
    # 绘制暂停键
    Element(Element.stop, Element.stop_position).draw(self.screen)
    # 绘制分数显示板
    score_board = Element(Element.board_score, Element.score_posi)
    score_board.draw(self.screen)
    # 绘制游戏分数
    str_score = str(self.score)
    for k, sing in enumerate(str_score):
        Element(Element.score[int(sing)], (755 + k * 32, 40)).draw(self.screen)
    # 创建小方块背景图片精灵组
    BrickSpriteGroup = pygame.sprite.Group()
    # 创建水果精灵组
    AnimalSpriteGroup = pygame.sprite.Group()
    # 向精灵组中添加精灵
    for row in range(self._height):
        for col in range(self._width):
            x, y = self.cell_xy(row, col)
            BrickSpriteGroup.add(Element(Element.brick, (x, y)))
            if self.animal[row][col] != -2:
                AnimalSpriteGroup.add(Element(Element.animal[self.animal[row][col]], (x, y)))
    # 绘制小方块的背景图
    BrickSpriteGroup.draw(self.screen)
    # 绘制小方块中的水果
    for ani in AnimalSpriteGroup:
        self.screen.blit(ani.image, ani.rect)
    # 绘制鼠标所单击的水果的突出显示边框
    if self.cur_sel != [-1, -1]:
        frame_sprite = Element(Element.frame_image, self.cell_xy(self.cur_sel[0], self.cur_sel[1]))
        self.screen.blit(frame_sprite.image, frame_sprite.rect)
    # 绘制游戏倒计时时间
    if not self.time_is_over:
        if self.running_time <= self.TIMEOUT:
            try:
                time_str = time.strftime("%X",
                        time.localtime(self.TIMEOUT // 1000 - self.running_time // 1000)).partition(":")[2]
                Font_Fact(time_str, Font_Fact.show_time_posi, 43, (0, 0, 0)).draw(self.screen)
            except Exception as e:
                pass
    else:
        time_str = "00:00"
        Font_Fact(time_str, Font_Fact.show_time_posi, 43, (0, 0, 0)).draw(self.screen)

    pygame.display.flip()                             # 更新页面显示，必须添加
```

```
    return AnimalSpriteGroup
```

（6）在 core/handler.py 文件的 Manager 类的 stop_game()方法中，添加当判定游戏超时时，游戏退出的逻辑代码，包括绘制游戏超时图片、改变游戏状态、进入排行榜页面、重置水果矩阵、复位游戏超时开关等。修改后的 stop_game()方法代码如下（注意：加粗的代码为新增代码）：

```
def stop_game(self):
    """ 结束本场游戏，进入排行榜的页面  """
    if self.status == 1:
        # 游戏死图
        if self.death_sign:
            pygame.time.delay(500)
            Element(Element.none_animal, Element.none_animal_posi).draw(self.screen)
            pygame.display.flip()
            pygame.time.delay(500)
            Base.status = 2                              # 结束本场游戏
            self.reset_layout = True                     # 布局下一盘游戏的元素
            self.time_is_over = False
        else:
            self.death_sign = True
        # 游戏超时
        if self.time_is_over:
            pygame.time.delay(1000)                      # 暂停程序一段时间
            # 绘制游戏超时图片
            Element(Element.time_is_over_image, Element.time_is_over_posi).draw(self.screen)
            pygame.display.flip()
            pygame.time.delay(2000)                      # 暂停程序一段时间，避免 "game over" 图片一闪而过
            Base.status = 2                              # 结束本场游戏
            self.reset_layout = True                     # 布局下一盘游戏的元素
            self.time_is_over = False
```

3.5.9 游戏排行榜页面的实现

当游戏处于主页面时，如果玩家单击了页面左下角的“退出”按钮，则可以使游戏进入排行榜页面，该页面中记录本次游戏的分数，并会在排行榜页面中降序显示。

水果消消乐游戏排行榜页面的实现步骤如下。

（1）在游戏主页面的事件监听方法中判断玩家是否单击了“退出”按钮，如果已单击，则改变游戏的全局状态变量 Base.status 为 2，并重新布局水果矩阵。代码如下：

```
Base.status = 2                          # 需要将 Base.status 设置为 2，以便结束本场游戏，重新布局水果矩阵
```

（2）在 core/handler.py 文件的 Manager 类中创建一个 score_list 列表，用来记录每场游戏的得分，代码如下：

```
score_list = []                                  # 排行榜存储分数列表
```

（3）在 Manager 类中创建一个 record_score()方法，用于在每一局游戏退出时，向分数列表中添加本局游戏的得分。代码如下：

```
def record_score(self):
    """ 记录每场游戏得分 """
    if self.score != 0:
        self.score_list.append(self.score)
        self.destory_animal_num = [0, 0, 0, 0, 0, 0]
        self.score = 0                           # 分数复位归零
```

（4）在 core 包中创建一个 sort_score.py 文件，用于实现游戏排行榜页面的相关功能。该文件中，首先导入 pygame 和 pygame 常量库、sys、精灵组件类以及游戏主页面管理类 Manager，然后创建一个

Score_Manager 类，用于绘制游戏排行榜页面和监听其中的事件。sort_score.py 文件完整代码如下：

```
import sys
import pygame
from core.base import Base
from core.entity import Font_Fact, Element
from core.handler import Manager

class Score_Manager(Base):
    """
    游戏排行榜页面的管理
    """
    def __init__(self):
        super(Score_Manager, self).__init__()

    def choice_game_init(self):
        """ 游戏排行榜页面的初始化 """
        if self.status == 2:
            # 背景的绘制
            Element(Element.bg_choice_image, (0, 0)).draw(self.screen)
            li = sorted(Manager.score_list, reverse=True)
            for k, item in enumerate(li[:8]):
                Font_Fact(str(item) + "    Score", \
                          (Element.score_order_rect[0] + 50, Element.score_order_rect[1] +
                           k * 50), 35, (0, 0, 0)).draw(self.screen)
            pygame.display.flip()

    def mouse_select(self, event):
        """ 游戏排行榜页面的事件监听 """
        if event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x, mouse_y = event.pos
            if self.status == 2:
                # "再来一局"的鼠标监听
                if Font_Fact.again_game_posi[0] < mouse_x < Font_Fact.again_game_posi[0] + 200 and
                        Font_Fact.again_game_posi[1] < mouse_y < Font_Fact.again_game_posi[1] + 60:
                    # 游戏进入游戏页状态
                    Base.status = 1
                    # 分数置 0
                    Manager.destory_animal_num = [0, 0, 0, 0, 0, 0]
                    # 初始化游戏开始时间
                    Manager.start_time = pygame.time.get_ticks()
                # "退出游戏" 的鼠标监听
                elif Font_Fact.quit_game_posi[0] < mouse_x < Font_Fact.quit_game_posi[0] + 200 and
                        Font_Fact.quit_game_posi[1] < mouse_y < Font_Fact.quit_game_posi[1] + 60:
                    sys.exit()
            if event.type == pygame.MOUSEBUTTONUP:
                pass
```

（5）在 bin/main.py 文件中实例化排行榜页面管理类 Score_Manager，并在游戏主逻辑循环中判断当游戏状态为排行榜页面状态时，调用排行榜页面的初始化方法，并且监听排行榜页面的鼠标事件。修改后的 bin/main.py 文件代码如下（注意：加粗的代码为新增代码）：

```
import pygame
from pygame.locals import *
import sys
from core.first_eye import Screen_Manager
from core.handler import Manager
from core.sort_score import Score_Manager

def main():
    """
    消消乐游戏主函数
```

```
:return:
"""
pygame.init()                                    # 设备的检测
pygame.font.init()                               # 字体文件的初始化

# 在这里创建每一个页面类的对象
mr = Screen_Manager()                            # 实例化首屏页面管理对象
mg = Manager()                                   # 实例化游戏页面管理对象
ms = Score_Manager()                             # 实例化排行榜页面管理对象

while 1:
    mg.judge_time()                              # 判断游戏超时
    # 在这里判断不同的游戏状态，从而执行不同的具体操作
    if mr.status == 0:                           # 判断游戏首屏状态
        mr.open_game_init()
    if mg.status == 1:                           # 判断游戏页面状态
        mg.reset_animal()                        # 随机分配元素
        # 绘制游戏页面
        AnimalSpriteGroup = mg.start_game_init()
        mg.clear_ele()                           # 标记清除
        mg.is_death_map()                        # 死图判断
        mg.exchange_ele(AnimalSpriteGroup)       # 元素交换
    mg.stop_game()                               # 结束游戏

    if ms.status == 2:                           # 判断游戏排行榜状态
        mg.record_score()                        # 记录本场游戏得分，并分数清零
        ms.choice_game_init()

    for event in pygame.event.get():             # 事件的监听与循环
        # 用户敲击键盘的键盘事件判断
        if event.type == KEYDOWN:
            # 判断用户按下 Q 键或者 Esc 键
            if event.key == K_q or event.key == K_ESCAPE:
                sys.exit()
        # 用户关闭游戏窗口的鼠标事件判断
        if event.type == QUIT:
            sys.exit()

        # 在这里进行不同页面的事件监听，从而改变游戏状态
        mg.mouse_select(event)                   # 对游戏页面的事件监听
        ms.mouse_select(event)                   # 对游戏排行榜页面的事件监听
        mr.mouse_select(event)                   # 对游戏首屏的事件监听
```

游戏排行榜页面效果如图 3.9 所示。

图 3.9　游戏排行榜页面

3.6 项 目 运 行

通过前述步骤，设计并完成了“水果消消乐游戏”项目的开发。下面运行该游戏，检验一下我们的开发成果。如图 3.10 所示，在 PyCharm 的左侧项目结构中展开水果消消乐游戏的项目文件夹，在其中选中 manage.py 文件，单击鼠标右键，在弹出的快捷菜单中选择 Run 'manage'命令，即可成功运行该项目。

说明

运行项目之前，一定要确保本机已经安装了 pygame 模块，如果没有安装，请使用 pip install pygame 命令进行安装。

水果消消乐游戏页面如图 3.11 所示。

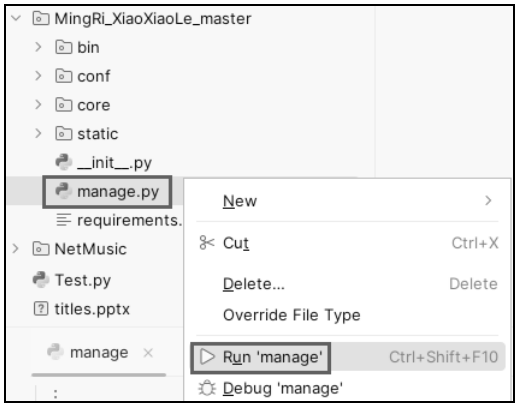


图 3.10 PyCharm 中的项目文件

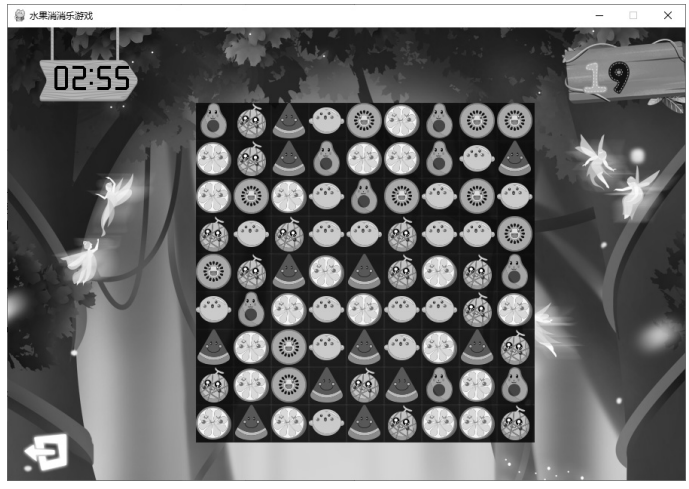


图 3.11 成功运行项目

本章使用 pygame 模块设计了一个功能完善的水果消消乐游戏，其中主要用到了 pygame 中的绘图、精灵、精灵组、事件监听等技术。学习本章时，重点要熟悉消消乐游戏的基本游戏规则以及实现流程，并掌握 pygame 中的绘图、精灵、精灵组及事件监听等技术在实际开发中的使用方法。

3.7 源 码 下 载

虽然本章详细地讲解了如何编码实现“水果消消乐游戏”的各个功能，但给出的代码都是代码片段，而非源码。为了方便读者学习，本书提供了完整的项目源码，扫描右侧二维码即可下载。

源码下载