《Python程序设计基础》程序设计作品说明书

题目： 外星人入侵

学院： 计算机科学与工程学院

姓名： 刘卓涵

学号： B20210302404

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

介绍本次设计完成的项目的概述，本文的主要内容，总结你主要完成的工作以及关键词。
外星人入侵项目是做一个二维的游戏，故事背景以一架飞船为主角，面对外星人的入侵进行反抗战斗的一个游戏。
在这个项目中，我进行了代码编写，使得外星人和主角的生成以及战斗计分。玩家可以控制飞船的移动以及发射子弹。
使用了Pygame包来开发的2D游戏。它在玩家每消灭一群向下移动的外星人后，将玩家提高一个等级。每射杀一个外星人获得的相应的分数，等级越高，游戏的节奏越快，难度越大，击杀每个外星人获得的分数就越高。
在此之外，我还加入了其他的因素，比如不同的子弹类型， 有普通的子弹， 穿甲弹， 多弹等等特殊效果。

关键词：

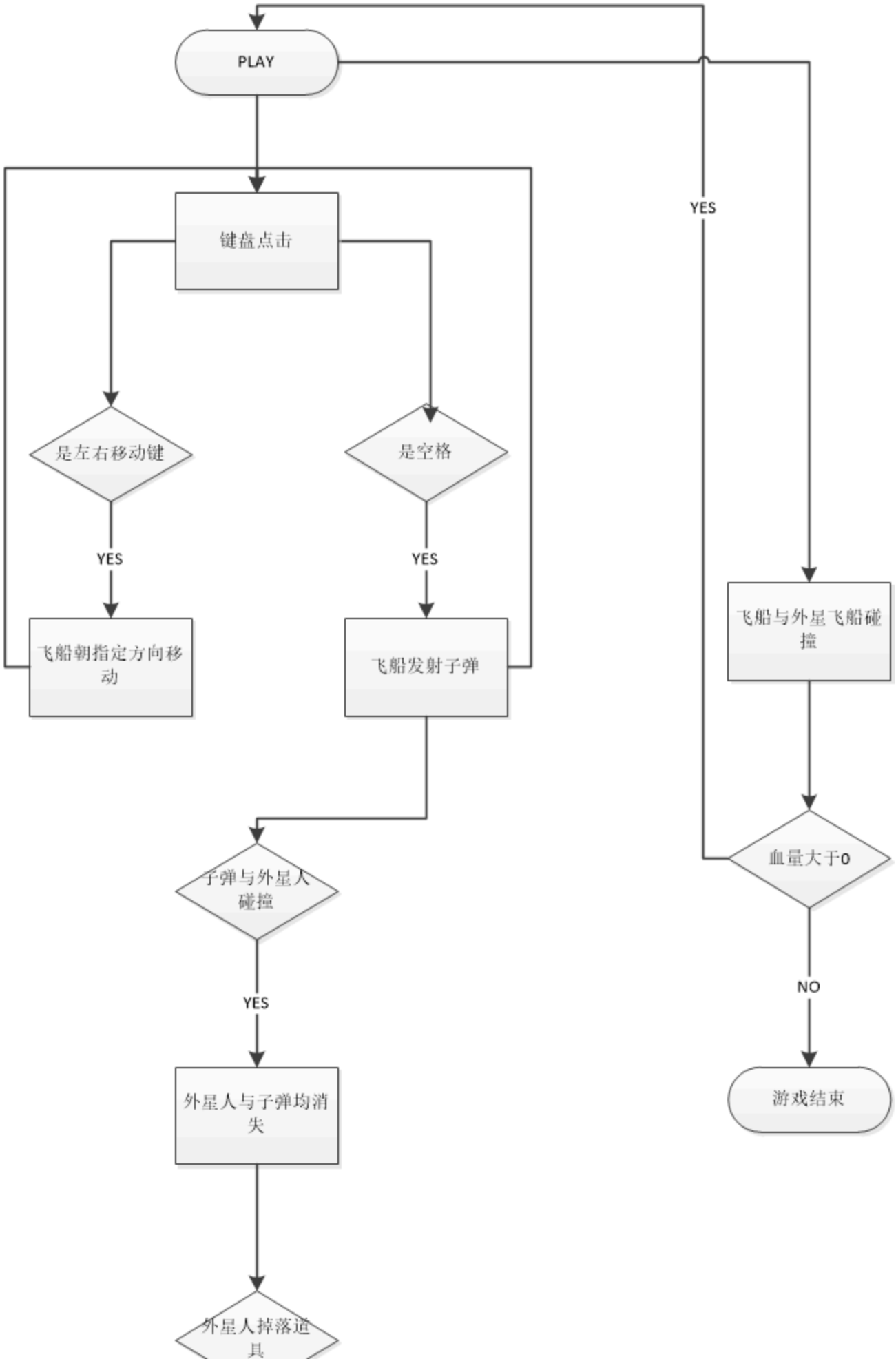
第1章 需求分析

本章的内容主要包括系统的需求分析，系统主要需要实现的功能有哪些，可以帮助用户解决哪些问题等等。
1.设计一个可以显示外星人飞船群以及玩家自身的屏幕。

2.实现点击屏幕上的PLAY键后进入游戏。
3.玩家自身可以点击空格使飞船射击以及通过移动键进行左右移动。
4.能够计算分数以及游戏等级并且实时的在屏幕上显示。
5.打死外星飞船会概率掉落道具，拾取道具可以回复玩家血量或者变化子弹类型。

第2章 分析与设计

本章的内容主要包括系统的设计，例如：系统架构、系统流程、系统模块、数据库的设计，以及关键的实现，例如：使用的数据结果、算法。
游戏基本逻辑：


PLAY
键盘点击
是左右移动键
是空格
YES
YES
飞船朝指定方向移动
飞船发射子弹
子弹与外星人碰撞
YES
外星人与子弹均消失
外星人掉落道具
YES
飞船与外星飞船碰撞
血量大于0
NO
游戏结束

1.运用pygame、sys库进行屏幕初始化

```
def __init__(self):
        pygame.init()
        self.clock = pygame.time.Clock()
        self.settings = Settings(-1)

        #指定Size
        self.screen = pygame.display.set_mode(
            (self.settings.screen_width, self.settings.screen_height))

        #全屏
        # self.screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
        # self.settings.screen_width = self.screen.get_rect().width
        # self.settings.screen_height = self.screen.get_rect().height

        #存储一个用于存储游戏统计信息的实例
        self.stats = GameStats(self)

        pygame.display.set_caption("Alien Invasion")

        #创建存储游戏统计信息的实例，  并创建记分牌
        self.stats = GameStats(self)
        self.sb = Scoreboard(self)

        self.ship = Ship(self, 1)
        self.bullets = pygame.sprite.Group()
        self.aliens = pygame.sprite.Group()
        self.props = pygame.sprite.Group()

        self._create_fleet()

        #设置背景颜色
        self.bg_color = (230, 230, 230)
        self.game_active = False

        #创建Play按钮
        self.play_button = Button(self, "Play")
```

2.游戏的主体运行逻辑

使用一个死循环来维持游戏的运行， 实时更新屏幕来达到动画效果。

设置好游戏的帧率， 保证游戏画面的稳定性。

```
def run_game(self):
        while True:
            self._check_events()

            if self.game_active:
                self.ship.update()
                self._update_bullets()
                self._update_aliens()
                self._update_props()
            self._update_screen()
            self.clock.tick(60)
```

3.玩家飞船的图像显示以及其初始化

利用pygame.sprite来进行设置

```
def __init__(self, ai_game, no):
        """初始化飞船并设置其初始位置"""
        super().__init__()
        self.screen = ai_game.screen
        self.settings = ai_game.settings
        self.screen_rect = ai_game.screen.get_rect()

        #加载飞船图像并获取其外接矩形


        #no
        #0：生命标识
        #1：基本状态
        if no == 0:
            self.image = pygame.image.load('images/小飞船(1).jpg')
        elif no == 1:
                self.image = pygame.image.load('images/小飞船.jpg')

        self.rect = self.image.get_rect()

        #每艘飞船都放在屏幕底部的中央
        self.rect.midbottom = self.screen_rect.midbottom

        #在飞船的属性中存储一个浮点数
        self.x = float(self.rect.x)

        #移动标志 （飞船一开始不移动）
        self.moving_right = False
        self.moving_left = False
```

4.外星飞船群的设置

先是对外星飞船单个的属性进行设置，然后再通过for循环在屏幕上有规律的显示。利用sprite.group来做到统计场上的外星飞船余下数量。

外星飞船的初始设置

```
def __init__(self, ai_game):
        """初始化外星人并设置其起始位置"""
        super().__init__()
        self.screen = ai_game.screen
        self.settings = ai_game.settings

        # 加载外星人图像并设置其rect属性
        # x = 10
        # if x == 10:
        self.image = pygame.image.load('images\\外星飞船(1).jpg')
        self.rect = self.image.get_rect()

        #每个外星人最初都在屏幕的左上角附近
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height + 30

        #存储外星人的精确水平位置
        self.x = float(self.rect.x)
```

5. 子弹设计

   1.初始子弹，为普通的子弹，与敌方单位触碰后便会同时消失。

   2.穿甲子弹：与敌人碰撞后不会直接消失，而是会穿越敌人继续它的使命。

   3.巨型子弹：一次性可以与多名地方单位触碰并且与这些敌方单位一起消失。

   其中穿甲弹与巨弹需要拾取敌方死亡后的道具方可获得。

   子弹初始化

```
def __init__(self, ai_game, no):
        """在飞船的当前位置创建一个子弹对象"""
        super().__init__()
        self.screen = ai_game.screen
        self.settings = B_settings(no)

        self.image = self.settings.image

        self.rect = self.image.get_rect()

        self.rect.midtop = ai_game.ship.rect.midtop
        self.y = float(self.rect.y)
```

子弹的类型属性, 单独一个bullet_settings类来存储他们的属性

```
def __init__(self, no):
        #子弹设置
        self.bullets_allowed = 5 #子弹数量上限
        #子弹编号不同，效果不同
        #0 普通 1 快速 2 穿甲
        if no == 0:
            self.bullet_speed = 15
            self.image = pygame.image.load('images/子弹(1).jpg')

        elif no == 1:
            self.bullet_speed = 10
            self.image = pygame.image.load('images/散弹(2).jpg')
        elif no == 3:
            self.bullet_speed = 15
            self.image = pygame.image.load('images/穿甲弹1(1).jpg')
```

6. 做出一个记分系统来实时记录并且显示此次游戏的分数以及所处的等级， 并且显示最高分。 以及剩余的生命值显示在屏幕最上方。

```
class Scoreboard:
    """显示得分信息的类"""
    def __init__(self, ai_game):
        """初始化显示得分涉及的属性"""
        self.ai_game = ai_game
        self.screen = ai_game.screen
        self.screen_rect = self.screen.get_rect()
        self.settings = ai_game.settings
        self.stats = ai_game.stats

        #显示得分信息时使用的字体设置
        self.text_color = (255, 255, 255)
        self.font = pygame.font.SysFont(None, 48)

        #准备初始得分图像和最高分
        self.prep_score()
        self.prep_high_score()
        self.prep_level()
        self.prep_ships()
```

7。各种属性的设置， 比如飞船的移动速度， 初始血量， 消灭单个敌方获得的分数等等。均在settingns类中实现，开局时均会进行初始化。

第3章 软件测试

本章的内容主要包括以类和函数作为单元进行单元测试，编写的对系统的主要功能的测试用例，以及测试用例执行的测试报告。

单元测试用例

测试目标	输入	
飞船能否移动以及发射子弹	按压指定键位	飞船根据指定方向
外星人飞船群能否正确移动到最下方并且与玩家相碰撞	开始游戏	外星人可以移动到
分数记录、等级记录等各项统计是否有效且正确	开始游戏并射杀外星人	随着外星人的减少

测试目标	输入	
子弹与外星人碰撞是否会消失	发射子弹	子弹与外星人接触后
外星人死亡后是否有概率掉落道具	消灭100个外星人	会有道具掉落
拾取道具是否会有效果	控制飞船拾取道具	玩家飞船会获得特殊

结论

本章的内容主要是对项目的总结，项目主要实现了哪些功能，达到了哪些目标，哪些不足之处，可以如何改进。
此项目实现了游戏的正常行为包括移动设计以及死亡， 并且能够记录统计各项数据。
不足之处：

1. 飞船无法升级，攻击形式单一，可玩性不高。
2. 敌机的不能攻击
3. 道具功能还不完善

改进方式：
1.为飞船增加几种形态，使其攻击手段增加
2.为敌机增加攻击功能，使游戏可玩性更高
3.增加道具的功能性

参考文献

Python编程从入门到实践[第三版]- [美]埃里克·马瑟斯 著

代码附录：
见附带文件夹alien_invasion