《Python程序设计基础》程序设计作品说明书

题目： 外星人入侵游戏👀

学院： 21计科

姓名： 金皓翔

学号： B20210302104

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

本次项目是对《外星人入侵》游戏的综合性优化和功能扩展。项目依据教材内容，主要完成了游戏基础功能的实现、交互体验的进阶和游戏元素的增强。具体包括创建游戏窗口、添加飞船图像、实现飞船驾驶及射击，以及教材13章的一群可移动外星人的创建和射击。此外，根据教材14章内容，添加了Play按钮、等级提升和计分功能，以及部分练习功能如在游戏背景中随机绘制星星和保存最高分。这些工作的完成不仅提升了游戏的互动性和用户体验，还增强了玩家的沉浸感和持续参与度。

关键词：游戏开发、用户交互、功能实现、Python编程。

第1章 需求分析

需求分析

1. 系统概述

《外星人入侵》是一款以太空为背景的射击游戏，旨在提供一个吸引人且富有挑战性的游戏体验。本系统的目标是实现一个用户友好、互动性强的游戏环境，同时确保游戏操作简单易懂，以吸引不同年龄层的玩家。

2. 主要功能需求

- **基础游戏设置**：包括创建游戏窗口、配置基本的游戏参数如屏幕大小、标题等。
- **角色控制**：实现玩家对飞船的完全控制，包括移动和射击。
- **敌人（外星人）逻辑**：创建并管理一群外星人，包括他们的出现、移动和与玩家互动。
- **游戏难度管理**：根据玩家的进展自动调整游戏难度，例如通过增加外星人的速度或数量。
- **用户界面**：包括游戏开始界面、得分板和等级显示。
- **游戏结束逻辑**：在特定条件下结束游戏，如玩家生命值耗尽或完成所有关卡。
- **高分记录**：记录并显示游戏的最高分数。

3. 问题解决

- **娱乐与休闲**：为玩家提供一种放松和娱乐的方式，帮助他们在忙碌的生活或工作中找到休息的时刻。
- **挑战与成就**：通过逐渐增加的游戏难度，给予玩家挑战自我和提升技能的机会。
- **互动体验**：通过丰富的游戏互动设计，如射击和躲避，增强玩家的游戏体验。
- **技能提升**：通过游戏中对策略和反应速度的要求，帮助玩家提升相关技能。

4. 用户群体

本游戏面向所有年龄层的玩家，特别是喜欢射击和太空主题游戏的用户。游戏设计考虑到了不同技能水平的玩家，以确保每个玩家都能在游戏中找到适合自己的挑战和乐趣。

第2章 分析与设计

分析与设计

1. 系统架构

《外星人入侵》游戏采用分层的客户端架构，主要包括以下几个层面：

- **呈现层**：使用 Pygame 库实现，负责渲染游戏界面和处理用户输入。
- **逻辑层**：处理游戏规则，如碰撞检测、得分计算和游戏状态管理。
- **数据层**：存储游戏状态和玩家记录，如当前得分和最高分。

2. 系统流程

游戏流程主要包括：

- **初始化**：设置游戏窗口、加载资源、初始化游戏元素。

```
pygame.init()
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Alien Invasion")
```

- **主循环**：处理事件、更新游戏状态、渲染画面。

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    # 更新游戏状态
    # 渲染画面
    pygame.display.flip()
```

- **结束处理**：游戏结束时保存得分、显示结束界面。

3. 系统模块

- **飞船模块（Ship）**：处理飞船的移动和射击。

```
class Ship:
    def __init__(self, ai_game):
        self.screen = ai_game.screen
        # ... 初始化代码 ...
```

```
    def update(self):
        # ... 更新飞船位置 ...
```

- **外星人模块（Alien）**：控制外星人的行为，如移动和出现。

```
class Alien:
    def __init__(self, ai_game):
        # ... 初始化代码 ...
    def update(self):
        # ... 更新外星人位置 ...
```

- **子弹模块（Bullet）**：管理子弹的发射和移动。

```
class Bullet:
    def __init__(self, ai_game):
        # ... 初始化代码 ...
    def update(self):
        # ... 更新子弹位置 ...
```

- **得分模块（Scoreboard）**：记录和显示得分。

```
class Scoreboard:
    def __init__(self, ai_game):
        # ... 初始化代码 ...
    def show_score(self):
        # ... 显示得分 ...
```

- **游戏控制模块**：处理游戏开始、结束和用户交互。

4. 数据库设计

不涉及复杂数据库设计，使用简单的文件系统来存储高分记录。

5. 关键实现

- **数据结构**：使用 Pygame 中的 Sprite 类来表示游戏中的各种元素，如飞船和外星人。
- **碰撞检测**：利用 Pygame 的 `sprite.groupcollide` 方法检测子弹和外星人之间的碰撞。
- **状态管理**：使用状态变量跟踪游戏的当前阶段，如活动、暂停或结束。

6. 代码实现

- **主游戏类（AlienInvasion）**：组织上述模块，管理游戏的主要流程。

```
class AlienInvasion:
    def __init__(self):
        # ... 初始化游戏和创建游戏资源 ...
    def run_game(self):
        # ... 主游戏循环 ...
```

7. 额外考虑

- **可拓展性**：代码结构允许轻松添加新元素和功能。
- **性能优化**：适当时进行代码重构和性能调优。

以上涵盖了《外星人入侵》游戏的系统设计和关键代码实现，确保了游戏的高可玩性和良好的用户体验。

第3章 软件测试

本章内容集中于《外星人入侵》游戏项目的单元测试，旨在验证关键类和函数的功能性和可靠性。以下是针对系统主要功能编写的单元测试用例及其执行结果，以及每个测试用例的具体代码实现。

单元测试用例

#	测试目标	输入	预期结果	测试结果
1	飞船移动	键盘事件（左/右键）	飞船左右移动	通过
2	发射子弹	空格键	屏幕上生成子弹并向上移动	通过
3	外星人移动	时间间隔	外星人左右下移	通过

单元测试用例及代码

1. 测试飞船移动

- **测试目标**：验证飞船是否能够根据用户输入向左和向右移动。
- **测试代码**：

```
import sys
import unittest

sys.path.insert(0, 'D:\\25448\\python课程\\python_practice\\项目\\daima')

from ship import Ship

from alien_invasion import AlienInvasion

class TestShipMovement(unittest.TestCase):

    def setUp(self):
        ai_game = AlienInvasion()
        self.ship = Ship(ai_game)
```

```
    def test_move_right(self):
        """测试飞船向右移动"""
        self.ship.moving_right = True
        self.ship.update()
        self.assertTrue(self.ship.rect.x > 0)

    def test_move_left(self):
        """测试飞船向左移动"""
        self.ship.moving_left = True
        self.ship.update()
        self.assertTrue(self.ship.rect.x < self.ship.screen_rect.right)

if __name__ == '__main__':
    unittest.main()
```

2. 测试发射子弹

- **测试目标**：验证子弹是否能够正确生成并向上移动。
- **测试代码**：

```
import sys
import unittest

sys.path.insert(0, 'D:\\25448\\python课程\\python_practice\\项目\\daima')

from bullet import Bullet
from alien_invasion import AlienInvasion

class TestBulletFiring(unittest.TestCase):

    def setUp(self):
        ai_game = AlienInvasion()
        self.bullet = Bullet(ai_game)

    def test_bullet_movement(self):
        """测试子弹向上移动"""
        initial_y = self.bullet.y
        self.bullet.update()
        self.assertTrue(self.bullet.y < initial_y)

if __name__ == '__main__':
    unittest.main()
```

3. 测试外星人移动

- **测试目标**：验证外星人是否能够按照预定路径移动。
- **测试代码**：

```
import sys
import unittest

sys.path.insert(0, 'D:\\25448\\python课程\\python_practice\\项目\\daima')

from alien import Alien
from alien_invasion import AlienInvasion

class TestAlienMovement(unittest.TestCase):

    def setUp(self):
        ai_game = AlienInvasion()
        self.alien = Alien(ai_game)

    def test_alien_movement(self):
        """测试外星人移动"""
        initial_x = self.alien.x
        self.alien.update()
        self.assertNotEqual(self.alien.x, initial_x)

if __name__ == '__main__':
    unittest.main()
```

测试方法和工具

- **测试方法**：使用 Python 的 `unittest` 框架进行自动化单元测试。
- **测试工具**：Python IDE或命令行工具。

测试执行和报告

- 所有测试用例均通过了预期的断言检查，未发现任何错误。
- 测试结果表明游戏的关键功能如飞船移动、子弹发射和外星人移动均按预期工作。
- 这些测试结果为游戏的稳定性和可靠性提供了保障，并确保了良好的玩家体验。

结论

项目总结

本章节对《外星人入侵》游戏项目进行总结，回顾项目实现的主要功能、达成的目标以及存在的不足之处，并提出可能的改进方向。

项目实现的主要功能

1. **基础游戏环境搭建**：成功创建游戏窗口，设置了基本的游戏参数如屏幕大小和标题。
2. **玩家控制实现**：实现了玩家对飞船的完全控制，包括移动和射击功能。
3. **敌方角色（外星人）逻辑**：创建了可以动态移动的外星人群，增加了游戏的挑战性。
4. **得分机制**：实现了玩家通过击落外星人获得分数的机制。
5. **游戏难度调节**：根据玩家的游戏表现逐步提升游戏难度。
6. **用户界面优化**：包括游戏开始界面、得分板和等级显示。

7. **特殊功能实现**：如背景中随机绘制星星和保存最高分数。

达成的目标

- **提高玩家互动性**：通过实现玩家对飞船的直接控制，游戏互动性得到显著提升。
- **增强游戏挑战性**：通过外星人的动态移动和逐渐增加的游戏难度，游戏更具挑战性。
- **优化用户体验**：改进的用户界面和得分系统使游戏体验更加丰富和有趣。

不足之处

- **游戏性能**：在高级难度或长时间运行下，游戏可能出现性能下降的情况。
- **功能多样性**：游戏在长时间玩耍后可能显得重复，缺少足够的多样性和深度。
- **用户定制化**：游戏目前缺乏用户定制选项，如不同的飞船选择或技能升级。

改进方向

- **性能优化**：对游戏的代码进行性能分析和优化，确保在各种条件下都能平稳运行。
- **增加游戏元素**：引入新的游戏元素和机制，例如不同类型的敌人、飞船技能升级或特殊关卡。
- **定制化功能**：提供更多的用户定制选项，允许玩家根据自己的喜好调整游戏体验。

总的来说，本项目在实现基本的游戏功能和提升用户体验方面取得了显著的成果，但在性能优化和功能多样性方面仍有提升空间。未来的工作将着重于这些方面，以进一步提升游戏的整体质量和玩家满意度。

参考文献

1. Matthes, E. (2016). *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press.

2. Sweigart, A. (2016). *Invent Your Own Computer Games with Python, 4th Edition*. No Starch Press.

3. Rouse, R. (2004). *Game Design: Theory and Practice, Second Edition*. Wordware Publishing, Inc.

4. Tychsen, A., & Hitchens, M. (2020). *Game Design: Software Development and Reality in the Global Game Industry*. Springer.

5. Chandler, H. M. (2013). *The Game Production Handbook, Third Edition*. Jones & Bartlett Learning.