《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01

姓名： 金库

学号： B20210302101

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

概述： 外星人大战是一款基于 Pygame 开发的2D射击游戏，玩家控制飞船在屏幕上移动并射击外星人，目标是躲避外星人的攻击并尽可能击败更多外星人，获取高分。游戏具有基本的飞船控制、碰撞检测、得分统计、等级提升等核心功能，同时通过添加星星背景、背景音乐、最高分统计等元素提升了游戏的娱乐性和视觉效果。

关键词： Pygame、飞船控制、背景音乐、碰撞检测、最高分统计、星星背景

第1章 需求分析

系统需求分析：

外星人大战游戏的系统需求分析旨在明确系统的功能和性能要求，以满足用户期望并提供良好的游戏体验。

主要功能：

1. **星星背景：** 添加星星背景，增强游戏的视觉吸引力。
2. **射击功能：** 提供射击外星人的能力，通过发射子弹来击败外星人。
3. **外星人生成：** 实现外星人在屏幕上的生成和移动，随游戏难度逐渐增加。
4. **碰撞检测：** 检测子弹与外星人的碰撞，确保游戏的交互性和挑战性。
5. **得分统计：** 记录玩家的得分，根据击败外星人数量提供奖励分数。
6. **等级提升：** 随着游戏进行，提高游戏等级，增加游戏难度。
7. **生命机会：** 记录玩家的生命数量，当生命用尽时游戏结束。
8. **按钮交互：** 提供Play按钮，使玩家能够开始游戏或重新开始游戏。
9. **飞船控制：** 允许玩家控制飞船在屏幕上自由移动，上下左右方向的灵活掌控。

问题解决：

- **娱乐：** 提供一种轻松娱乐的方式，让用户在游戏中享受乐趣。
- **挑战性：** 通过逐渐增加游戏难度，为用户提供挑战，增强游戏的吸引力。
- **放松：** 提供一个放松的平台，让用户在游戏中放松身心，减轻压力。

用户期望：

- **简单上手：** 游戏界面简单直观，易于上手，不需要复杂的学习过程。

- **流畅体验：** 游戏运行流畅，没有明显的卡顿或延迟，提供良好的用户体验。
- **趣味性：** 游戏具有足够的趣味性，使用户产生持续的兴趣。

系统性能要求：

通过满足以上需求，外星人大战游戏将能够提供一种愉悦的游戏体验，满足用户的娱乐需求。

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。 **系统设计：**

外星人大战游戏的设计旨在确保系统具有清晰的架构、合理的流程和模块。

1. **系统架构：**

- **呈现层**：游戏界面的生成，背景声音图片的使用，由Pygame库实现。
- **逻辑层**：飞船移动，碰撞检测，得分统计。
- **数据层**：最高分的记录。

1. **系统流程：**

    - 游戏开始时，初始化游戏设置和统计信息。
    - 用户通过客户端与游戏交互，包括飞船移动、射击、开始游戏等。
    - 服务器处理用户操作，更新游戏状态，检测碰撞等。
    - 游戏根据用户表现更新得分、等级，并动态调整游戏难度。
    - 游戏结束时，展示得分和游戏结果。

2. **系统模块：**

    - **游戏模块：** 负责游戏的初始化、运行和结束。
    - **飞船模块：** 处理飞船的移动、射击和碰撞检测。

```
class Ship(Sprite):

    def __init__(self, ai_settings, screen):
        """初始化飞船并设置其初始位置"""
        super(Ship, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings
```

- **外星人模块：** 处理外星人的生成、移动和碰撞检测。

```
class Alien(Sprite):
    """表示单个外星人的类"""

    def __init__(self, ai_settings, screen):
        """初始化外星人并设置其起始位置"""
```

```
        super(Alien, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings
```

- **子弹模块：** 管理子弹的生成、移动和碰撞检测。

```
class Bullet(Sprite):
    """一个对飞船发射的子弹进行管理的类"""

    def __init__(self, ai_settings, screen, ship):
        """在飞船所处的位置创建一个子弹对象"""
        super(Bullet, self).__init__()
        self.screen = screen
```

- **得分模块：** 记录玩家的得分和等级。

```
class Scoreboard():
    """显示得分信息的类"""

    def __init__(self, ai_settings, screen, stats):
        """初始化显示得分涉及的属性"""
        self.screen = screen
        self.screen_rect = screen.get_rect()
        self.ai_settings = ai_settings
        self.stats = stats
```

- **星星模块：**

```
class Star(Sprite):
    """表示单个星星的类。"""
    def __init__(self, ai_settings, screen):
        """初始化星星并设置其随机位置。"""
        super(Star, self).__init__()
        self.screen = screen
        self.ai_settings = ai_settings
        self.image = pygame.image.load('images/stars.bmp')
        self.rect = self.image.get_rect()
```

1. **数据库设计：**
    - 游戏数据主要集中在服务器端的数据库中。
    - 数据库包括用户信息、得分记录、等级信息等。
    - 数据库设计合理，支持高效的数据检索和更新操作。

2. **关键实现：**

   - **碰撞检测算法：** 采用高效的碰撞检测算法，确保实时性和准确性。

```
def check_bullet_alien_collisions(ai_settings, screen, stats, sb, ship,aliens,
bullets):
    """响应子弹和外星人的碰撞"""
    # 删除发生碰撞的子弹和外星人
    collisions = pygame.sprite.groupcollide(bullets, aliens, True, True)

    if collisions:
        for aliens in collisions.values():
            stats.score += ai_settings.alien_points * len(aliens)
            sb.prep_score()
        check_high_score(stats, sb)

    if len(aliens) == 0:
        # 如果整群外星人都被消灭，就提高一个等级
        bullets.empty()
        ai_settings.increase_speed()

        # 提高等级
        stats.level += 1
        sb.prep_level()

        create_fleet(ai_settings, screen, ship, aliens)



def update_aliens(ai_settings, stats, screen, sb, ship, aliens, bullets):
    """
    检查是否有外星人位于屏幕边缘，并更新整群外星人的位置
    """
    check_fleet_edges(ai_settings, aliens)
    aliens.update()

    # 检测外星人和飞船之间的碰撞
    if pygame.sprite.spritecollideany(ship, aliens):
        ship_hit(ai_settings, stats, screen, sb, ship, aliens, bullets)

    # 检查是否有外星人到达屏幕底端
    check_aliens_bottom(ai_settings, stats, screen, sb, ship, aliens, bullets)
```

- **等级提升算法：** 根据用户得分动态调整游戏等级，增加游戏难度。

```
    if len(aliens) == 0:
        # 如果整群外星人都被消灭，就提高一个等级
        bullets.empty()
```

```
            ai_settings.increase_speed()

            # 提高等级
            stats.level += 1
            sb.prep_level()
```

通过以上设计，外星人大战游戏系统具有清晰的结构和高效的实现，满足用户的娱乐需求。

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：**

为了确保系统的主要功能正常运作，我们进行了单元测试。单元测试是通过对系统的各个模块、类和函数进行独立测试来验证其功能的一种测试方法。下面是我们进行的主要功能测试用例及测试报告：

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	键盘上下左右键	飞船移动正常	通过
2	发射子弹	空格键	子弹正常运动	通过

1. **飞船模块测试：**
    - **测试目标：** 验证飞船的移动、射击和碰撞检测功能。
    - **测试用例：** 模拟飞船移动、射击、与外星人碰撞的情况。
    - **测试报告：** 飞船模块通过测试，各项功能运作正常。

```
import unittest
import pygame
from ship import Ship
from settings import Settings

class TestShipMovement(unittest.TestCase):

    def setUp(self):
        """初始化测试"""
        pygame.init()
        self.ai_settings = Settings()
        self.screen = pygame.display.set_mode((self.ai_settings.screen_width,
self.ai_settings.screen_height))
        pygame.display.set_caption("Test Ship Movement")
        self.ship = Ship(self.ai_settings, self.screen)

    def test_move_right(self):
        """测试向右移动"""
        self.ship.moving_right = True
        original_center = self.ship.rect.centerx
```

```
        self.ship.update()
        self.assertGreater(self.ship.rect.centerx, original_center)

    def test_move_left(self):
        """测试向左移动"""
        self.ship.moving_left = True
        original_center = self.ship.rect.centerx
        self.ship.update()
        self.assertLess(self.ship.rect.centerx, original_center)

    def test_move_up(self):
        """测试向上移动"""
        self.ship.moving_up = True
        original_center = self.ship.rect.centery
        self.ship.update()
        self.assertLess(self.ship.rect.centery, original_center)


    def tearDown(self):
        """清理测试"""
        pygame.quit()
        self.screen = None
        self.ai_settings = None

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

2. **子弹模块测试：**
    - **测试目标：** 验证子弹的生成、移动和碰撞检测功能。
    - **测试用例：** 模拟子弹生成、移动、与外星人碰撞的情况。
    - **测试报告：** 子弹模块通过测试，各项功能正常。

```
import unittest
import pygame
from ship import Ship
from settings import Settings

class TestShipMovement(unittest.TestCase):

    def setUp(self):
        """初始化测试"""
        pygame.init()
        self.ai_settings = Settings()
        self.screen = pygame.display.set_mode((self.ai_settings.screen_width,
self.ai_settings.screen_height))
        pygame.display.set_caption("Test Ship Movement")
        self.ship = Ship(self.ai_settings, self.screen)

    def test_move_right(self):
        """测试向右移动"""
        self.ship.moving_right = True
```

```
        original_center = self.ship.rect.centerx
        self.ship.update()
        self.assertGreater(self.ship.rect.centerx, original_center)

    def test_move_left(self):
        """测试向左移动"""
        self.ship.moving_left = True
        original_center = self.ship.rect.centerx
        self.ship.update()
        self.assertLess(self.ship.rect.centerx, original_center)

    def test_move_up(self):
        """测试向上移动"""
        self.ship.moving_up = True
        original_center = self.ship.rect.centery
        self.ship.update()
        self.assertLess(self.ship.rect.centery, original_center)


    def tearDown(self):
        """清理测试"""
        pygame.quit()
        self.screen = None
        self.ai_settings = None

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

通过以上单元测试，我们确保了系统的主要功能在各个模块中正常运作，提高了系统的稳定性和可靠性。

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。 外星人大战项目实现了以下主要功能和目标：

实现了玩家飞船的控制，包括上下左右移动，以及发射子弹。实现了外星人群的生成，初始时在屏幕上生成一群外星人，并支持外星人的左右移动。实现了子弹与外星人的碰撞检测，当子弹击中外星人时，得分增加，并移除相应的子弹和外星人。记录了玩家的得分、等级、剩余生命数等游戏统计信息。随着游戏的进行，外星人的移动速度和生成速度逐渐增加，提高了游戏难度。添加了Play按钮，玩家可以通过点击按钮开始游戏，同时游戏结束时按钮可重新开始游戏。 在屏幕上生成了随机星星背景，增强了游戏的视觉效果。

不足之处和改进方向，目前的星星背景每次更新都重新生成，可能导致画面闪烁。可以考虑在游戏初始化时生成一次，并在后续的更新中保持不变。当玩家的生命数减为零时，游戏立即结束，可以考虑添加一个过渡画面，提供更好的游戏结束体验。目前外星人只能左右移动，可以尝试添加更多种类的外星人，以及更多复杂的移动和攻击行为，提高游戏的多样性。可以进一步美化游戏界面，包括添加更多的动画效果、改进外星人和飞船的图形等。根据项目的规模，可能需要对代码结构进行一些优化，以提高代码的可维护性和可扩展性。 总体而言，外星人大战项目已经完成了基本的功能，可以通过进一步的改进和扩展使游戏更加丰富和有趣。

参考文献

[1]Matthes, E. (2016). Python编程从入门到实践 [Translation: Python Crash Course]. 人民邮电出版社 [People's Posts and Telecommunications Press].

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

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