《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3班

姓名： 潘振宇

学号： B20210302304

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

Python外星人入侵游戏是一款基于Pygame库开发的2D游戏，玩家需要控制一艘飞船来抵御外星人的入侵。游戏中，玩家可以移动飞船并发射子弹来摧毁外星人，同时需要躲避外星人的攻击。游戏具有多个关卡和难度递增的挑战，玩家可以通过击败外星人和获得高分来提升自己的等级

关键词：Python Pygame

第1章 需求分析

1. 游戏界面：包括游戏窗口、飞船、外星人、子弹等元素的绘制。
2. 控制操作：玩家可以通过键盘控制飞船的移动，按下空格键发射子弹。
3. 碰撞检测：检测子弹与外星人的碰撞，以及飞船与外星人的碰撞。
4. 计分系统：记录玩家的得分，并显示最高分和当前等级。
5. 关卡设置：游戏具有多个关卡，每个关卡的外星人数量和速度都不同。
6. 难度递增：随着游戏的进行，外星人的数量和速度会逐渐增加，增加游戏的挑战性。

第2章 分析与设计

1. 游戏初始化：设置游戏窗口大小、背景颜色等参数，并创建游戏对象。
2. 游戏循环：在游戏循环中，监听玩家的操作，更新游戏元素的位置和状态，并进行碰撞检测和计分。
3. 绘制元素：使用Pygame库提供的绘制函数，将游戏元素绘制到游戏窗口中。
4. 碰撞检测：使用Pygame库提供的碰撞检测函数，检测子弹与外星人的碰撞，以及飞船与外星人的碰撞。
5. 计分系统：记录玩家的得分，并根据得分更新最高分和当前等级。
6. 关卡设置：根据当前等级设置外星人的数量和速度，控制游戏的难度递增。

第3章 软件测试

一：外星人移动的测试用例时，我们可以考虑以下几个方面：

1. 外星人向右移动的测试用例：
    - 创建一个外星人对象，并设置其初始位置。
    - 调用外星人的向右移动方法。
    - 断言外星人的位置是否正确地向右移动了。
2. 外星人向左移动的测试用例：

- 创建一个外星人对象，并设置其初始位置。
- 调用外星人的向左移动方法。
- 断言外星人的位置是否正确地向左移动了。

3. 外星人到达屏幕边缘时改变方向的测试用例：
   - 创建一个外星人对象，并设置其初始位置。
   - 设置外星人的移动方向为向右。
   - 调用外星人的向右移动方法，直到外星人到达屏幕边缘。
   - 断言外星人的位置是否正确地到达了屏幕边缘，并且移动方向是否正确地改变为向左。

```
import unittest
from alien import Alien, Settings

class AlienMovementTestCase(unittest.TestCase):
    """测试外星人移动的测试用例"""

    def setUp(self):
        """在每个测试方法运行之前创建外星人对象"""
        self.settings = Settings()
        self.alien = Alien(self.settings, None)

    def test_alien_moves_right(self):
        """测试外星人向右移动"""
        initial_x = self.alien.rect.x
        self.alien.update()
        self.assertEqual(self.alien.rect.x, initial_x +
self.settings.alien_speed_factor)

    def test_alien_moves_left(self):
        """测试外星人向左移动"""
        initial_x = self.alien.rect.x
        self.alien.ai_settings.fleet_direction = -1
        self.alien.update()
        self.assertEqual(self.alien.rect.x, initial_x -
self.settings.alien_speed_factor)

    def test_alien_changes_direction_at_edge(self):
        """测试外星人到达屏幕边缘时改变方向"""
        self.alien.rect.x = self.settings.screen_width - self.alien.rect.width
        self.alien.ai_settings.fleet_direction = 1
        self.alien.update()
        self.assertEqual(self.alien.ai_settings.fleet_direction, -1)

unittest.main()
```

二：射击测试代码

1. 确保玩家的飞船能够发射子弹：
   - 创建一个玩家飞船对象，并设置其初始位置。
   - 模拟玩家按下发射子弹的按键。
   - 断言玩家飞船发射了一颗子弹。

2. 确保子弹能够正确地向上移动：
    - 创建一个子弹对象，并设置其初始位置。
    - 调用子弹的向上移动方法。
    - 断言子弹的位置是否正确地向上移动了。
3. 确保子弹能够击中外星人：
    - 创建一个子弹对象和一个外星人对象，并设置它们的初始位置。
    - 将子弹移动到外星人的位置。
    - 断言子弹是否成功击中了外星人，并触发了相应的效果。

```
import unittest
from ship import Ship
from bullet import Bullet
from alien import Alien
from game_functions import check_bullet_alien_collision

class ShootingTestCase(unittest.TestCase):
    """测试射击功能的测试用例"""

    def setUp(self):
        """在每个测试方法运行之前创建玩家飞船和外星人对象"""
        self.ship = Ship()
        self.bullet = Bullet(self.ship)
        self.alien = Alien()

    def test_ship_shoots_bullet(self):
        """测试玩家飞船发射子弹"""
        self.ship.shoot_bullet()
        self.assertTrue(self.bullet.active)

    def test_bullet_moves_up(self):
        """测试子弹向上移动"""
        initial_y = self.bullet.rect.y
        self.bullet.update()
        self.assertEqual(self.bullet.rect.y, initial_y - self.bullet.speed_factor)

    def test_bullet_hits_alien(self):
        """测试子弹击中外星人"""
        self.bullet.rect.x = self.alien.rect.x
        self.bullet.rect.y = self.alien.rect.y
        check_bullet_alien_collision(self.bullet, self.alien)
        self.assertFalse(self.bullet.active)
        self.assertFalse(self.alien.active)

unittest.main()
```

三：外星人入侵游戏的初始化界面测试

1. 确保游戏窗口正确创建：
    - 创建一个游戏对象，并初始化游戏窗口。
    - 断言游戏窗口的宽度、高度和背景颜色是否正确。

2. 确保游戏标题正确显示：
    - 创建一个游戏对象，并设置游戏标题。
    - 断言游戏窗口的标题是否正确显示。
3. 确保游戏背景图像正确加载：
    - 创建一个游戏对象，并加载游戏背景图像。
    - 断言游戏背景图像是否成功加载。

```
import unittest
from alien import Alien

class GameInitializationTestCase(unittest.TestCase):
    """测试外星人入侵游戏的初始化界面的测试用例"""

    def test_game_window_creation(self):
        """测试游戏窗口的创建"""
        game = Game()
        game.initialize_window()
        self.assertEqual(game.window_width, 800)  # 假设游戏窗口宽度为800
        self.assertEqual(game.window_height, 600)  # 假设游戏窗口高度为600
        self.assertEqual(game.window_bg_color, (0, 0, 0))  # 假设游戏窗口背景颜色为
黑色

    def test_game_title_display(self):
        """测试游戏标题的显示"""
        game = Game()
        game.initialize_window()
        game.set_title("外星人入侵")
        self.assertEqual(game.window_title, "外星人入侵")  # 断言游戏窗口的标题是否正
确显示为"外星人入侵"

    def test_game_background_image_loading(self):
        """测试游戏背景图像的加载"""
        game = Game()
        game.initialize_window()
        game.load_background_image("background.jpg")  # 假设游戏背景图像
为"background.jpg"
        self.assertIsNotNone(game.background_image)  # 断言游戏背景图像是否成功加载

unittest.main()
```

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。

外星人入侵项目主要实现了以下功能和目标：

理解了！你希望我帮你重写游戏功能列表，并添加改进游戏界面和用户界面的建议。让我们从头开始：

游戏功能：

1. **游戏界面基础**：
    - 创建游戏窗口并加载背景图像。
    - 显示游戏标题和基本的游戏界面布局。
2. **玩家飞船控制**：
    - 实现玩家飞船的控制机制，使其能在游戏界面中自由移动。
    - 允许玩家发射子弹，击败敌人。
3. **外星人敌人**：
    - 添加多个外星人敌人，让它们在屏幕上移动，并向玩家飞船发射子弹。
4. **碰撞检测和击败敌人**：
    - 实现子弹和外星人之间的碰撞检测，当子弹击中外星人时，处理相应的击败敌人逻辑。
    - 让玩家飞船被外星人的子弹击中时游戏结束。
5. **得分系统和显示**：
    - 设计得分系统，使玩家能够通过击败外星人来获得得分。
    - 在界面上显示当前得分，并及时更新。
6. **关卡和难度递增**：
    - 实现多个关卡，每个关卡增加难度，包括外星人移动速度、攻击频率等。
    - 确保难度递增不至于过于剧烈，让玩家能够逐步适应挑战。
7. **游戏结束和重新开始**：
    - 当玩家飞船被外星人的子弹击中时，结束游戏，并显示最终得分。
    - 提供重新开始游戏的选项。

不足之处：

游戏界面和图形效果简单：

游戏界面目前相对简单，缺乏视觉吸引力和深度，没有足够的图形元素或动态效果。 图形效果方面较为简单，缺少动画、特效或细节，需要提升以增强整体的视觉体验。 用户界面单一：

游戏用户界面基本，缺乏交互性和多样性元素，如开始菜单、游戏设置等，可能影响玩家的长期参与。 缺少用户界面的深度设计，例如暂停菜单、选项菜单等，这些可以提升游戏的易用性和吸引力。 **改进：**

加以提升自我的代码算法等各方面的水准

参考文献

Python编程： 从入门到实践 / (美) 埃里克.马瑟斯（Eric Matthes）著；袁国忠译. ——北京： 人民邮电出版社