《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03班

姓名： 黄澍

学号： B20200202217

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

本项目为《外星人入侵游戏》，需要我们使用pygame包来开发一款2D游戏，它在玩家没消灭一群向下移的外星人后，都将玩家提高一个等级；而等级越高，游戏的节奏越快，难度越大。以及完成一些拓展内容。

关键词：pygame，python

第1章 需求分析

一、外星人入侵项目

游戏界面：
创建一个游戏窗口，显示游戏的画面。
在游戏窗口中显示玩家的飞船、外星人和其他游戏元素。
显示得分、关卡等游戏信息。

玩家控制：
允许玩家使用键盘或鼠标来控制飞船的移动。
允许玩家按下特定键来射击子弹。

外星人生成：
自动生成一群外星人，并将它们放置在屏幕的顶部。

外星人应该能够自动移动，并在边缘触碰到屏幕时改变方向。

飞船与外星人的交互：
当飞船与外星人碰撞时，游戏结束。
当玩家击中外星人时，得分增加，并且被击中的外星人消失。

关卡和难度递增：
实现多个关卡，每个关卡的外星人的数量和移动速度都有所增加。
提高游戏难度，使得外星人的移动更快或者它们的攻击更频繁。

游戏结束：
当玩家飞船被外星人撞毁或者外星人达到屏幕底部时，游戏结束。
显示最终得分和游戏结果，并提供重新开始游戏的选项。

二、拓展练习

练习12-6将飞船放在屏幕左侧进行射击

练习13-2在游戏背景中随机位置绘制星星

练习14-5 将游戏中得到的最高分保存到文件中

第2章 分析与设计

一、系统架构：

该项目采用基于面向对象编程的架构，将不同的游戏元素表示为对象，并通过交互来实现游戏逻辑。

使用Pygame库作为游戏引擎，提供图形显示和用户输入的功能。

二、游戏流程：

游戏开始时，创建游戏窗口并初始化游戏设置。

在游戏主循环中，处理玩家输入、更新游戏状态、绘制游戏画面。

检测碰撞和判断游戏结束的条件。

根据游戏状态的变化，更新得分和关卡等信息。

三、系统模块：

主界面模块：负责创建游戏窗口、设置背景色等

玩家飞船模块：处理玩家输入，控制飞船移动和射击。

外星人模块：生成外星人群、处理外星人移动和碰撞检测。

子弹模块：处理子弹的生成、移动和碰撞检测。

得分和关卡模块：记录玩家得分和当前关卡，根据关卡调整游戏难度。

四、数据库的设计：

本项目的最高分是通过保存在一个名为high_score.json的js文件中来实现保存最高分的。每次都会进行对比，如果不比当前最高分更高则不会保存。

第3章 软件测试

测试用例：

1	用例编号	用例名称	功能特性			预期结果
2	1	测试外星人移动	检查外星人是否到达屏幕边缘			测试类输出“OK”
3	2	测试子弹初始化	检查子弹是否被正确初始化			测试类输出“OK”

测试代码：

```python
import unittest
from unittest.mock import MagicMock
import pygame

from alien import Alien

class AlienTest(unittest.TestCase):
    def setUp(self):
        pygame.init()
        self.screen = pygame.display.set_mode((800, 600))
        self.settings = MagicMock()
        self.ai_game = MagicMock()
        self.ai_game.screen = self.screen
        self.ai_game.settings = self.settings

    def test_check_edges(self):
        alien = Alien(self.ai_game)
        alien.rect.right = self.screen.get_rect().right + 10
        self.assertTrue(alien.check_edges())

        alien = Alien(self.ai_game)
        alien.rect.left = -10
        self.assertTrue(alien.check_edges())

        alien = Alien(self.ai_game)
        alien.rect.right = self.screen.get_rect().right - 10
        self.assertFalse(alien.check_edges())

    def test_update(self):
        alien = Alien(self.ai_game)
        alien.x = 100
        self.settings.alien_speed = 1
        self.settings.fleet_direction = 1

        alien.update()
        self.assertEqual(alien.x, 101)
        self.assertEqual(alien.rect.x, 101)

        self.settings.fleet_direction = -1
        alien.update()
        self.assertEqual(alien.x, 100)
        self.assertEqual(alien.rect.x, 100)
```

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

测试结果：

```
----------------------------------------------------------------------
Ran 2 tests in 1.163s

OK
```

测试代码：

```python
import unittest
import pygame
from bullet import Bullet
from settings import Settings
from ship import Ship

class BulletTest(unittest.TestCase):
    """Tests for the Bullet class."""

    def setUp(self):
        """Set up the test environment."""
        pygame.init()
        self.screen = pygame.display.set_mode((1200, 800))
        pygame.display.set_caption("Bullet Test")
        self.settings = Settings()
        self.ship = Ship(self)
        self.bullet = Bullet(self)

    def test_init(self):
        """Test that bullet object is initialized correctly."""
        self.assertEqual(self.bullet.rect.midtop, self.ship.rect.midtop)
        self.assertAlmostEqual(self.bullet.y, float(self.ship.rect.y))
        self.assertEqual(self.bullet.color, self.settings.bullet_color)

    def test_update(self):
        """Test that bullet is moving up the screen."""
        initial_y = self.bullet.rect.y
        self.bullet.update()
        self.assertLess(self.bullet.rect.y, initial_y)

def test_draw_bullet(self):
    """Test that bullet is drawn to the screen."""
    with unittest.mock.patch('pygame.draw.rect') as mock_draw_rect:
        self.bullet.draw_bullet()
        mock_draw_rect.assert_called()

    def tearDown(self):
        """Clean up the test environment."""
        pygame.quit()

if __name__ == '__main__':
```

```
    unittest.main()
```

测试结果：

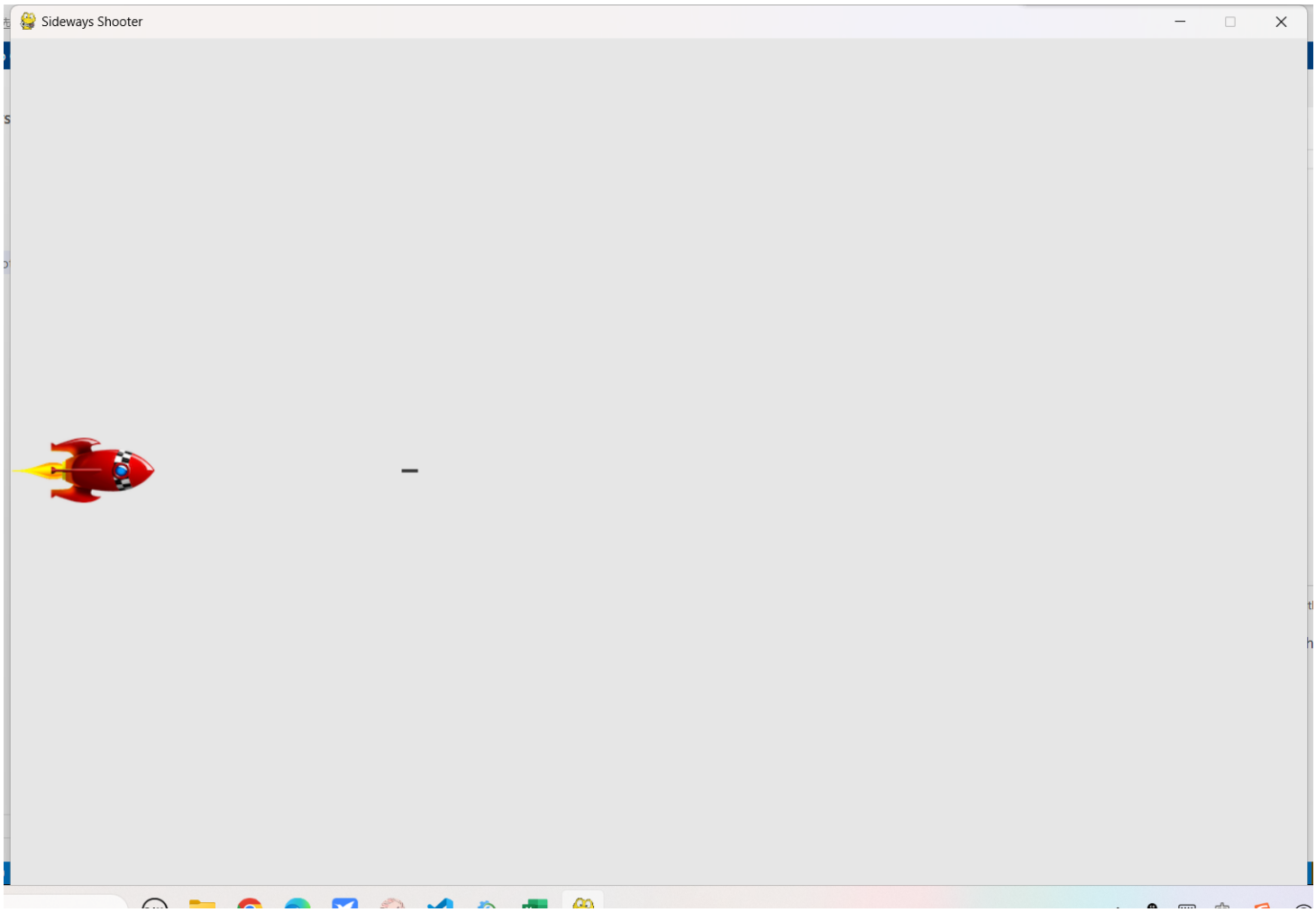
```
----------------------------------------------------------------------
Ran 2 tests in 1.123s

OK
```

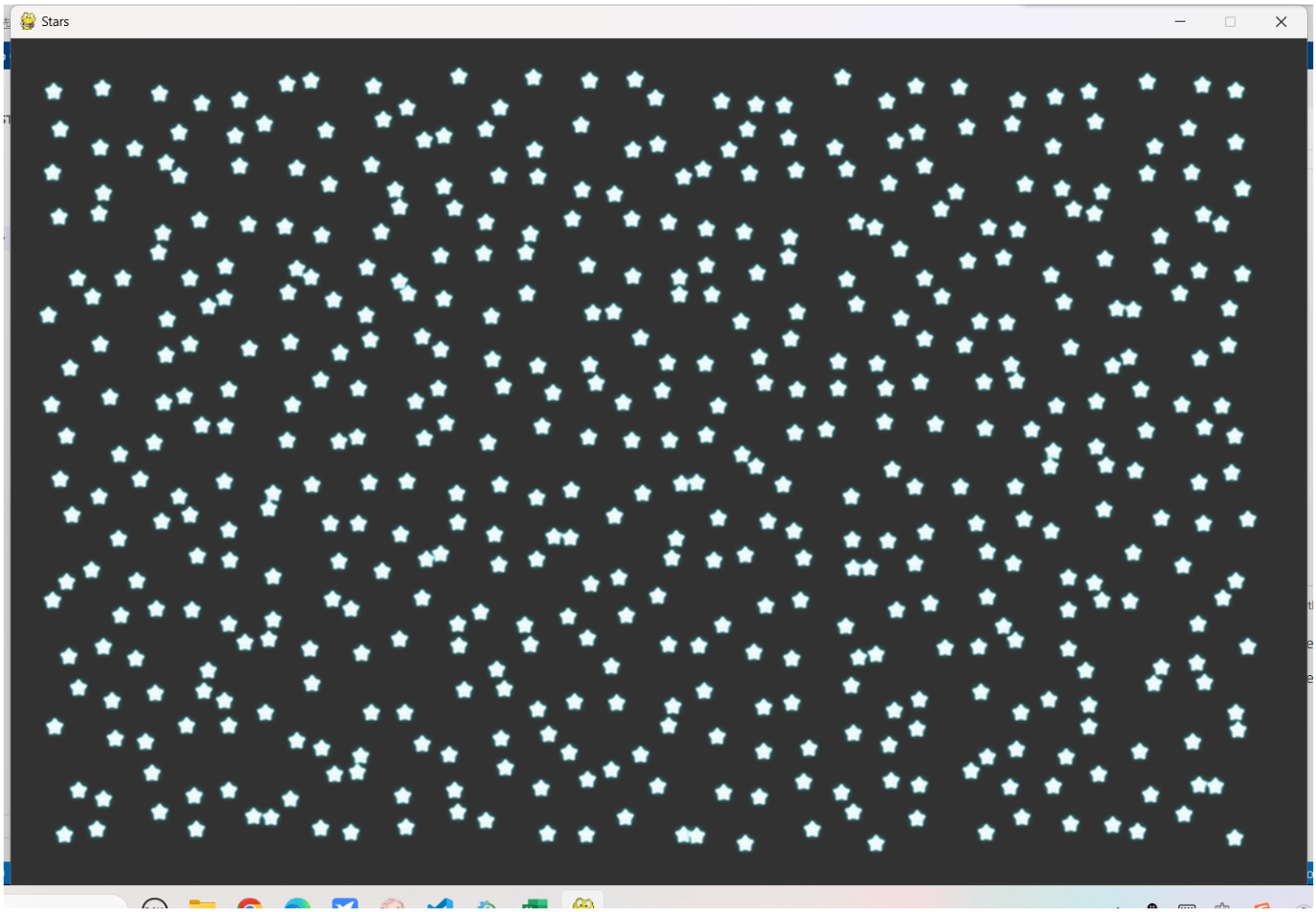
拓展练习：

运行效果：

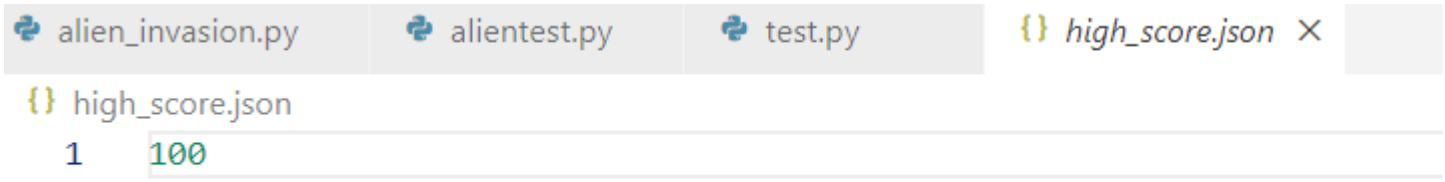
12-6横向射击



13-2随机星星



14-5保留最高分



结论

外星人入侵项目是一款基于 Python 的游戏，旨在实现经典的射击类游戏体验。该项目主要实现了玩家飞船的移动和射击功能、外星人的生成和移动、游戏得分和关卡更新以及游戏的开始和结束逻辑。通过这些功能的实现，项目达到了提供玩家基本的游戏操作体验，并能够根据游戏进展更新游戏状态和难度的目标。

项目的优点在于：
实现了基本的游戏功能，包括玩家飞船的移动和射击、外星人的生成和移动等，为玩家提供了简单而有趣的游戏体验。

代码结构清晰，模块化程度较高，易于阅读和维护。

然而，项目也存在一些不足之处：
缺乏游戏性深度：除了基本的移动和射击操作外，游戏内容相对简单，缺乏更多元的游戏元素和机制，可能会影响长期吸引力。
可扩展性有限：目前的功能相对简单，难以支持较复杂的游戏玩法和关卡设计。

为了改进项目，可以考虑以下方面的改进：
增加游戏元素：例如增加不同类型的外星人，引入道具系统、Boss 战等，以增强游戏的多样性和挑战性。
设计更多关卡：设计多样化的关卡内容，包括不同的地图布局、外星人生成规律和难度设置，以丰富游戏的可玩性和挑战性。
引入音效和动画：通过添加背景音乐、音效和动画效果，提升游戏的沉浸感和娱乐性。

综上所述，外星人入侵项目在初步实现了基本的游戏功能的同时，还有很大的改进空间，可以通过增加游戏元素、设计多样化的关卡和引入更多的视听效果来提升游戏的整体质量和娱乐性。

参考文献

[1] 埃里克·马瑟斯. Python编程从入门到实践第3版[M].人民邮电出版社出版社，2023.