《Python 程序设计基础》程序设计作品说明书

题目： 外星人入侵地球

学院：计算机科学与工程学院

姓名： 张栝棋

学号： B20210302112

指导教师： 周景

起止日期：2023.11.10-2023.12.10

摘要

本次项目是关于外星人入侵地球的一款游戏，用户可以通过按开始按钮来开始外星人入侵地球的游戏，极大地缓解了用户的心理压力，给生活增添了更多活力。

关键词：外星人入侵地球，游戏，解压

第 1 章 需求分析

在游戏《外星人入侵》中，玩家控制着一艘最初出现在屏幕底部中央的飞船。玩家可以使用箭头键左右移动飞船，还可使用空格键进行射击。游戏开始时，一群外星人出现在天空中，他们在屏幕中向下移动。玩家的任务是射杀这些外星人。玩家将所有外星人都消灭干净后，将出现一群新的外星人，他们移动的速度更快。只要有外星人撞到了玩家的飞船或到达了屏幕底部，玩家就损失一艘飞船。玩家损失三艘飞船后，游戏结束。

第 2 章 分析与设计

1.添加飞船图像

下面将飞船加入到游戏中。为了在屏幕上绘制玩家的飞船，我们将加载一幅图像，

2. 关键代码实现：

Window.py(设置屏幕的初始化函数）：

```
import sys
import pygame


from settings import Settings
from ship import Ship


def run_game():
    # 初始化 pygame、设置和屏幕对象
    pygame.init()
    ai_settings=Settings()
    screen = pygame.display.set_mode(
        (ai_settings.screen_width,ai_settings.screen_height))# 设置窗口大小
    pygame.display.set_caption("Alien Invasion 1.0")# 窗口名称


    # 设置背景色
    bg_color = (230,230,230)


    # 创建飞船
    ship = Ship(screen)
    # 开始游戏的主循环
    while True:
        # 监视键盘和鼠标事件
        for event in pygame.event.get():
            if event.type == pygame.QUIT:# 检测玩家单击游戏窗口的关闭按钮，调用 sys.exit()
退出
                sys.exit()
```

```
        # 每次循环时都重绘屏幕
        screen.fill(ai_settings.bg_color)
        ship.blitme()
        # 让最近绘制的屏幕可见
        pygame.display.flip()

run_game()
```

game_functions.py（按键的设置）：

```
import sys
import pygame
def check_events(ship):
    """响应按键和鼠标事件"""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                ship.moving_right = True
            elif event.key == pygame.K_LEFT:
                ship.moving_left = True

        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_RIGHT:
                ship.moving_right = False
            elif event.key == pygame.K_LEFT:
                ship.moving_left = False
```

```
def update_screen(ai_settings,screen,ship):
    """更新屏幕上的图像，并切换到新屏幕"""
    # 每次循环时都重绘屏幕
    screen.fill(ai_settings.bg_color)
    ship.blitme()
    # 让最近绘制的屏幕可见
    pygame.display.flip()
```

bullet.py（飞船发射子弹函数）：

```
    import pygame
    from pygame.sprite import Sprite

    class Bullet(Sprite):
        """一个对飞船发射子弹管理的类"""
        def __init__(self,ai_settings,screen,ship):
            """在飞船说出的位置创建一个子弹对象"""
            super().__init__()
            self.screen = screen

            # 在（0，0）创建一个表示子弹的矩阵，再设置正确的位置
            self.rect = pygame.Rect(0,0,ai_settings.bullet_width,ai_settings.bullet_height)
            self.rect.centerx = ship.rect.centerx
            self.rext.top = ship.rect.top

            # 存储用小数表示的子弹位置
            self.y = float(self.rect.y)
```

```
            self.color = ai_settings.bullet_color
            self.speed_factor = ai_settings.bullet_speed_factor

        def uppdate(self):
            """向上移动子弹"""
            # 更新表示子弹位置的小数值
            self.y -= self.speed_factor
            # 更新表示子弹的 rect 的位置
            self.rect.y = self.y

        def draw_bullet(self):
            """在屏幕上绘制子弹"""
            pygame.draw.rect(self.screen, self.color, self.rect)
```

Alien.py(外星人初始化函数):

```
import pygame
    from pygame.sprite import Sprite

    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().__init__()
            self.screen = screen
            self.ai_settings = ai_settings
            # 加载外星人图像，并设置其 rect 属性
            self.image = pygame.image.load('images/alien.bmp')
            self.rect = self.image.get_rect()
```

```
        # 每个外星人最初都在屏幕左上角附近
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height
        # 存储外星人的准确位置
        self.x = float(self.rect.x)

    def blitme(self):
        """在指定位置绘制外星人"""
        self.screen.blit(self.image, self.rect)
```

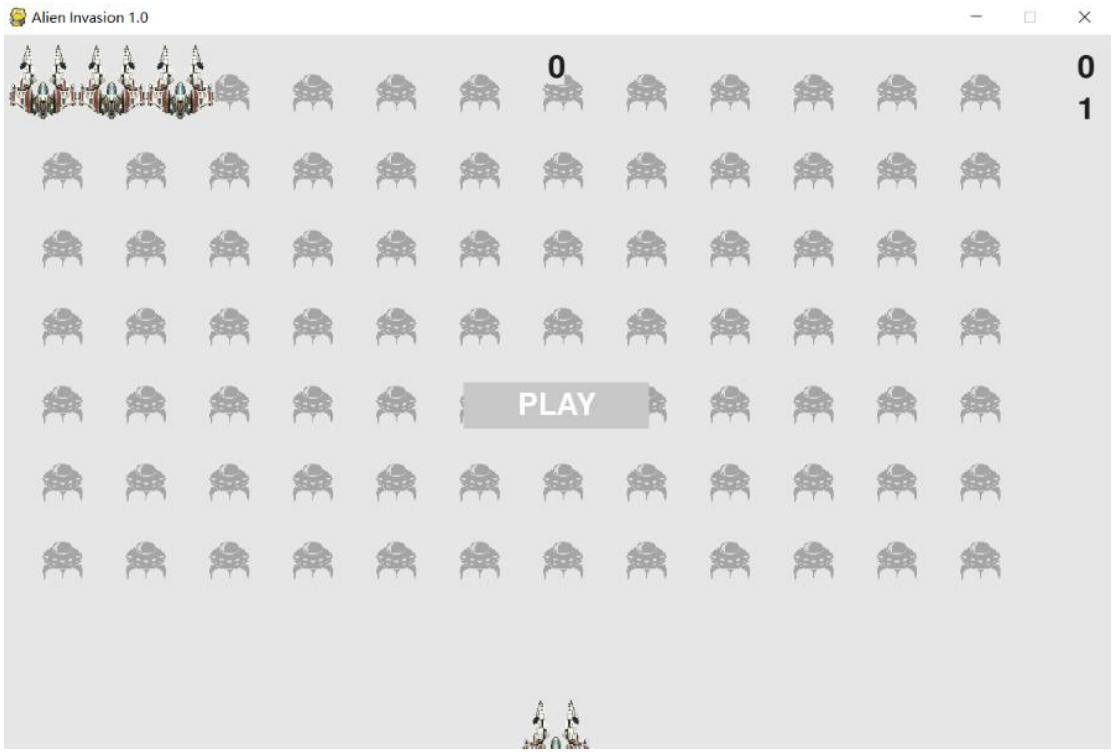
game_stats.py(游戏的开始与结束函数):

```
class GameStats():
    """跟踪游戏的统计信息"""
    def __init__(self, ai_settings):
        """初始化统计信息"""
        self.ai_settings = ai_settings
        self.reset_stats()

        # 游戏刚启动时处于活动状态
        self.game_active = True

    def reset_stats(self):
        """初始化在游戏运行期间可能变化的统计信息"""
        self.ships_left = self.ai_settings.ship_limit
        ```
```

最后在 window.py 和 game_functions.py 添加条件
```

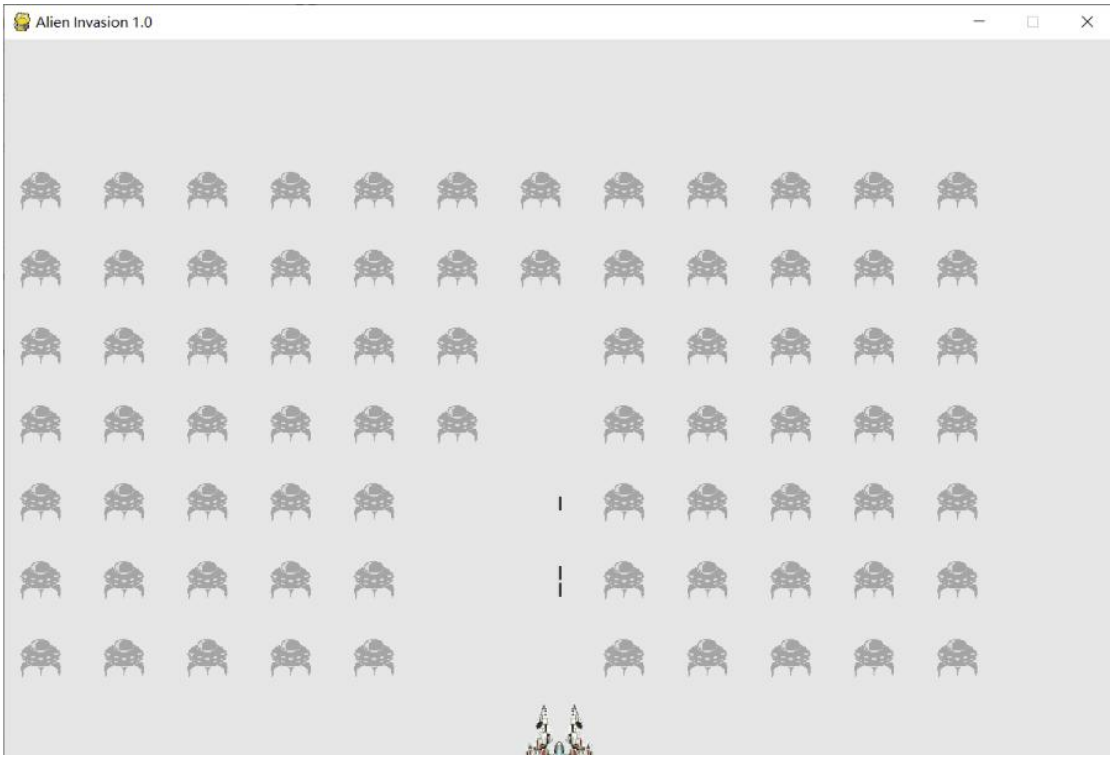
game_functions.py

```python
def ship_hit(ai_settings, stats, screen, ship, aliens, bullets):
 """响应飞船被外星人撞到"""
 if stats.ships_left > 0:
        # 将 ships_left 减 1
        stats.ships_left -= 1
        --snip--
        #暂停一会儿
        sleep(0.5)
 else:
        stats.game_active = False
```

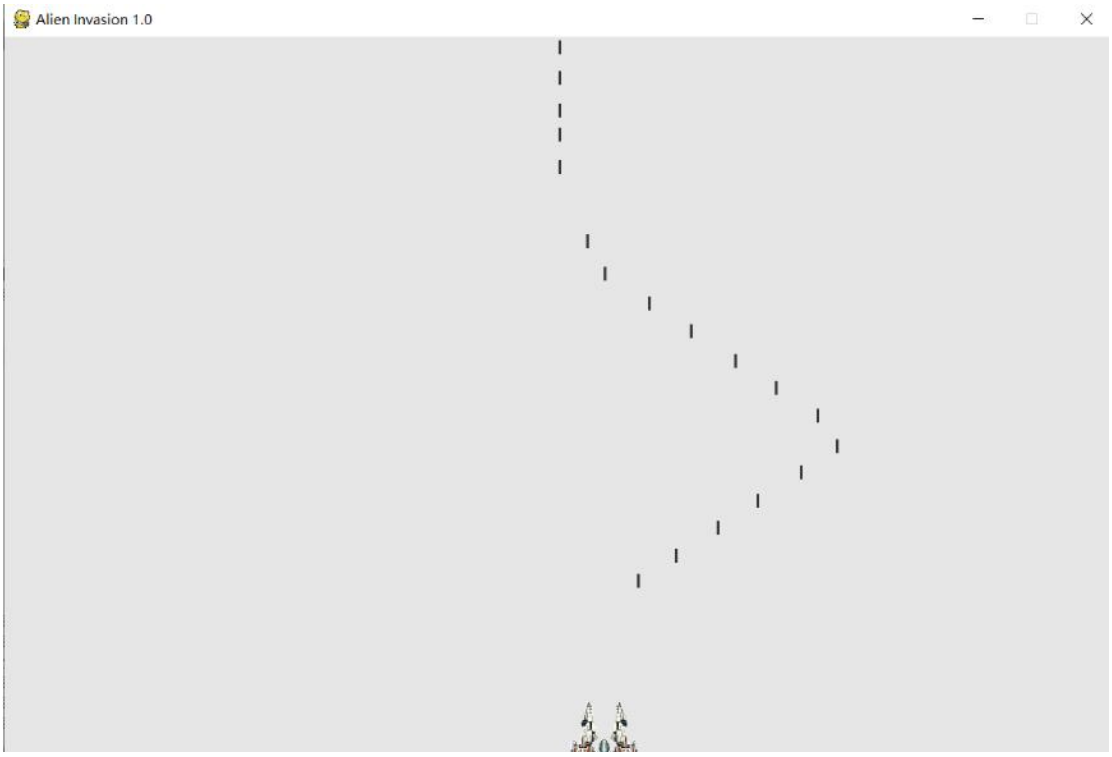
3. 数据结果以及展示：



点击屏幕中心的 play 按钮开始游戏，飞船应共有三条命，三条命都失去就代表游戏结束，右上角还有计分系统



子弹击落的效果



子弹的运行轨迹

第 3 章 软件测试

单元测试用例

用飞船剩余生命值来表示输入的结果

#	测试目标	输入	预期结果	测试结果
1	Ship	3	Start	Start
2	Ship	2	Start	Start
3	Ship	1	Start	Start
4	Ship	0	Over	Over

在生命值大于等于 1 且小于等于 3 时，游戏继续

在生命值到达 0 时，飞船死亡，游戏结束

结论

该项目主要做了一个外星人入侵地球的 python 游戏，玩家可以操控飞船来击打外星人，左右前后移动，飞船一共有三次生命，在碰到外星人是会减一条，生命消耗殆尽就会死亡，也就意味着游戏结束。

参考文献

1.《Python 游戏编程基础》 - 作者：Daniel Mulkey

2.《Python 编程：游戏设计入门》 - 作者：Sean Tracy, Bryan Helmig

3.《Python 与 Pygame 游戏开发》 - 作者：Will McGugan

4.《Python 编程的 AI 游戏开发》 - 作者：Danny Kodicek