|  |
| --- |
| 1.1项目标题：python打外星人小游戏 |
| 1.2参与成员：张思源202005002133 负责实验说明书写以及代码错误排查  周香 202005002145 负责主体代码的导入、运行环境构建以及代码运行 |
| 1.3项目介绍：该项目主要使用python实现打外星人小游戏，游戏操作包括：  1.关闭按键退出 2.开始按钮进入 3.SPACE空格开火 4.左右移动  玩家击败外星人并记录击败个数，获得游戏积分，将游戏积分显示在游戏界面当中。玩家可以在游戏中实现游戏的重置以及最高分的记录。 |
| 1.4详细运行说明：    图1 图2  图1为游戏开始界面，窗口上标显示游戏标题“飞船大战UFO”，点击“start game”开始游戏。  图2为游戏主题界面，玩家通过键盘操作飞船，点击空格进行设计后外星人消失，右上角分数变化，上标数字为游戏关卡（此时游戏是第一轮）。    图3 图4  图3为飞船进行持续射击后，外星人图标下移靠近飞船。图4为第一关结束前界面，在所有外星人被攻击后即将进入第二关，以此类推，关卡数递进外星人的靠近速度和移动速度也不断增加。 |
| 1.5参考资料：  1.CSDN：宇宙飞船打外星人(python-2D游戏开发)  https://blog.csdn.net/wuzhihuaw/article/details/121519898  2.CSDN：飞船打外星人(python)  https://blog.csdn.net/worldinme/article/details/122014204 3.github:https://github.com/LowApe/PythonPractice |
| 1.6 Github链接地址： |

2. 代码和数据

1.alien.py

import pygame

from pygame.sprite import Sprite

class Alien(Sprite):

def \_\_init\_\_(self,ai\_settings,screen):

super().\_\_init\_\_()

# 初始化外星人

self.screen=screen

self.ai\_settings=ai\_settings

# 加载图片

self.image=pygame.image.load('images/alien.bmp')

self.rect=self.image.get\_rect()

# 定位

self.rect.x=self.rect.width

self.rect.y=self.rect.height

# 存储外星人的准确定位

self.x=float(self.rect.x)

def update(self):

self.rect.x+=(self.ai\_settings.alien\_speed\_factor\*self.ai\_settings.fleet\_direction)

def check\_edges(self):

"""如果外星人位于屏幕边缘,就返回True"""

screen\_rect=self.screen.get\_rect()

if self.rect.right>=screen\_rect.right:

return True

elif self.rect.left<=0:

return True

def blitme(self):

self.screen.blit(self.image,self.rect)

2. alien\_invasion.py

import sys

import pygame

from settings import Settings

from ship import Ship

from bullet import Bullet

import game\_functions as gf

from pygame.sprite import Group

from alien import Alien

from game\_stats import GameStats

from button import Button

from scoreboard import Scoreboard

def run\_game():

# 1.初始化游戏并创建一个屏幕对象

pygame.init()

ai\_settings=Settings()

# 创建ship stat 的实例

stats=GameStats(ai\_settings)

screen=pygame.display.set\_mode((ai\_settings.screen\_width,ai\_settings.screen\_height))

# 3.设置游戏标题

pygame.display.set\_caption('Alien Invasion')

# 创建一艘飞船

ship=Ship(ai\_settings,screen)

# 实例外星人

alien=Alien(ai\_settings,screen)

# 创建一个编组存储子弹

bullets=Group()

# 创建一个编组存储外星人

aliens=Group()

gf.create\_fleet(ai\_settings,screen,aliens,ship)

# 创建一个button

play\_button=Button(ai\_settings,screen,'Play')

# 创建一个计分板实例

scoreboard=Scoreboard(ai\_settings,screen,stats)

# 4.开始游戏的主循环

while True:

# 鼠标事件

gf.check\_event(ai\_settings,screen,ship,aliens,bullets,stats,play\_button,scoreboard)

gf.update\_screen(ai\_settings,screen,ship,bullets,aliens,play\_button,stats,scoreboard)

if stats.game\_active:

ship.update()

bullets.update()

gf.update\_bullets(ai\_settings,screen,bullets,aliens,ship,scoreboard

,stats)

gf.update\_aliens(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard)

else:

pygame.mouse.set\_visible(True)

run\_game()

3. bullet.py

import pygame

from pygame.sprite import Sprite

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

# 在（0,0）处创建一个表示子弹的矩形，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.rect.top=ship.rect.top

# 储存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 draw\_bullet(self):

"""在屏幕上绘制子弹"""

pygame.draw.rect(self.screen,self.color,self.rect)

def update(self):

"""子弹向上"""

# y轴减去速度

self.y-=self.speed\_factor

# 实际影响

self.rect.y=self.y

4. button.py

import pygame.font

class Button():

def \_\_init\_\_(self,ai\_settings,screen,msg):

self.screen=screen

self.screen\_rect=self.screen.get\_rect()

self.width,self.height=200,50

self.button\_color=(0,255,0)

self.text\_color=(255,255,255)

self.font=pygame.font.SysFont(None,48)

self.rect=pygame.Rect(0,0,self.width,self.height)

self.rect.center=self.screen\_rect.center

# 按钮只需创建一次

self.prep\_msg(msg)

def prep\_msg(self,msg):

# 将msg渲染为图像，并使其在按钮上居中

self.msg\_image=self.font.render(msg,True,self.text\_color,self.button\_color)

self.msg\_imgae\_rect=self.msg\_image.get\_rect()

self.msg\_imgae\_rect.center=self.rect.center

def draw\_button(self):

self.screen.fill(self.button\_color,self.rect)

self.screen.blit(self.msg\_image,self.msg\_imgae\_rect)

5. game\_functions.py

import sys

import pygame

from ship import Ship

from bullet import Bullet

from alien import Alien

from time import sleep

def check\_event(ai\_settings,screen,ship,aliens,bullets,stats,play\_button,scoreboard):

"""响应按键和鼠标事件"""

# 监视键盘和鼠标事件

for event in pygame.event.get():

if event.type==pygame.QUIT:

sys.exit()

elif event.type==pygame.KEYDOWN:

check\_keydown\_event(ai\_settings,screen,stats,aliens,event,ship,bullets,scoreboard)

elif event.type==pygame.KEYUP:

check\_keyup\_event(event,ship)

elif event.type==pygame.MOUSEBUTTONDOWN:

mouse\_x,mouse\_y=pygame.mouse.get\_pos()

check\_play\_button(ai\_settings,screen,stats,ship,aliens,bullets,play\_button,mouse\_x,mouse\_y,scoreboard)

def check\_play\_button(ai\_settings,screen,stats,ship,aliens,bullets,play\_button,mouse\_x,mouse\_y,scoreboard):

if play\_button.rect.collidepoint(mouse\_x,mouse\_y) and not stats.game\_active:

start\_game(ai\_settings,screen,stats,aliens,bullets,ship,scoreboard)

def check\_keydown\_event(ai\_settings,screen,stats,aliens,event,ship,bullets,scoreboard):

if event.key==pygame.K\_RIGHT:

# 飞船向右移动

ship.moving\_right=True

elif event.key==pygame.K\_LEFT:

# 飞船向左移动

ship.moving\_left=True

elif event.key==pygame.K\_SPACE:

# 开火

fire\_bullet(ai\_settings,screen,ship,bullets)

elif event.key==pygame.K\_q:

# 按键q退出游戏

sys.exit()

elif event.key==pygame.K\_p:

start\_game(ai\_settings,screen,stats,aliens,bullets,ship,scoreboard)

def start\_game(ai\_settings,screen,stats,aliens,bullets,ship,scoreboard):

pygame.mouse.set\_visible(False)

# 初始化游戏设置

ai\_settings.initialize\_dynamic\_settings()

# 重置游戏统计信息

stats.reset\_stats()

scoreboard.prep\_score()

scoreboard.prep\_level()

stats.game\_active=True

# 清空外星人和子弹

aliens.empty()

bullets.empty()

# 重新绘制屏幕

create\_fleet(ai\_settings,screen,aliens,ship)

ship.center\_ship()

def check\_keyup\_event(event,ship):

if event.key==pygame.K\_RIGHT:

ship.moving\_right=False

elif event.key==pygame.K\_LEFT:

ship.moving\_left=False

def update\_bullets(ai\_settings,screen,bullets,aliens,ship,scoreboard

,stats):

for bullet in bullets.copy():

if bullet.rect.top<=0:

bullets.remove(bullet)

check\_bullets\_aliens\_collisions(ai\_settings,screen,bullets,aliens,ship,scoreboard,stats)

def check\_bullets\_aliens\_collisions(ai\_settings,screen,bullets,aliens,ship,scoreboard,stats):

collisions=pygame.sprite.groupcollide(bullets,aliens,True,True)

if len(aliens)==0:

bullets.empty()

ai\_settings.increas\_speed()

stats.level+=1

scoreboard.prep\_level()

create\_fleet(ai\_settings,screen,aliens,ship)

if collisions:

for aliens in collisions.values():

stats.score+=ai\_settings.alien\_points\*len(aliens)

scoreboard.prep\_score()

scoreboard.prep\_ships()

check\_high\_score(stats,scoreboard)

def update\_aliens(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard):

check\_fleet\_edges(ai\_settings,aliens)

aliens.update()

if pygame.sprite.spritecollideany(ship,aliens):

ship\_hit(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard)

check\_aliens\_bottom(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard)

def check\_aliens\_bottom(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard):

for alien in aliens.sprites():

if alien.rect.bottom>=screen.get\_rect().bottom:

ship\_hit(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard)

break

def check\_high\_score(stats,scoreboard):

if stats.high\_score<=stats.score:

stats.high\_score=stats.score

scoreboard.prep\_high\_score()

def ship\_hit(ai\_settings,screen,ship,stats,aliens,bullets,scoreboard):

# 生命减1

if stats.ship\_left>0:

stats.ship\_left-=1

scoreboard.prep\_ships()

else:

stats.game\_active=False

# 清空屏幕

aliens.empty()

bullets.empty()

create\_fleet(ai\_settings,screen,aliens,ship)

# 创建新的外星人和飞船顺序错了

#create\_fleet(ai\_settings,screen,ship,aliens)

ship.center\_ship()

# 暂停0.5

sleep(0.5)

def check\_fleet\_edges(ai\_settings,aliens):

"""移动外星人群"""

for alien in aliens.sprites():

if alien.check\_edges():

check\_fleet\_direction(ai\_settings,aliens)

break;

def check\_fleet\_direction(ai\_settings,aliens):

for alien in aliens.sprites():

alien.rect.y+=ai\_settings.fleet\_drop\_speed

ai\_settings.fleet\_direction\*=-1

def fire\_bullet(ai\_settings,screen,ship,bullets):

if len(bullets)<ai\_settings.bullet\_allowed:

# 创建一颗子弹，并将其加入到编组bullets中

new\_bullet=Bullet(ai\_settings,screen,ship)

bullets.add(new\_bullet)

def create\_fleet(ai\_settings,screen,aliens,ship):

"""绘制需要的外星人"""

alien=Alien(ai\_settings,screen)

alien\_width=alien.rect.width

alien\_height=alien.rect.height

number\_alien\_x=get\_number\_aliens\_x(ai\_settings,alien\_width)

number\_rows=get\_number\_aliens\_y(ai\_settings,alien\_height,ship)

for number\_row in range(number\_rows):

for alien\_number in range(number\_alien\_x):

create\_alien(ai\_settings,screen,aliens,alien\_number,number\_row,alien\_height,alien\_width)

def get\_number\_aliens\_x(ai\_settings,alien\_width):

available\_space=ai\_settings.screen\_width-2\*alien\_width

number\_alien\_x=int(available\_space / (2\*alien\_width))

return number\_alien\_x

def get\_number\_aliens\_y(ai\_settings,alien\_height,ship):

available\_space=ai\_settings.screen\_height-3\*alien\_height

number\_rows=int(available\_space/(2\*alien\_height))

return number\_rows

def create\_alien(ai\_settings,screen,aliens,alien\_number,number\_rows,alien\_height,alien\_width):

alien=Alien(ai\_settings,screen)

alien.rect.x=alien\_width+2\*alien\_width\*alien\_number

alien.rect.y=alien\_height+alien\_height\*2\*number\_rows

aliens.add(alien)

def update\_screen(ai\_settings,screen,ship,bullets,aliens,play\_button,stats,scoreboard):

"""更新屏幕上的图像，并切换新的屏幕"""

screen.fill(ai\_settings.bg\_color)

for bullet in bullets.sprites():

bullet.draw\_bullet()

ship.blitme()

aliens.draw(screen)

scoreboard.show\_score()

if not stats.game\_active:

play\_button.draw\_button()

pygame.display.flip()

6. game\_stats.py

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()

self.game\_active=False

# 在任何情况下都不应该重置最高分

self.high\_score=0

def reset\_stats(self):

self.ship\_left=self.ai\_settings.ship\_limit

self.score=0

self.level=1

7. scoreboard.py

import pygame.font

from pygame.sprite import Group

from ship import Ship

class Scoreboard():

"""显示得分信息的类"""

def \_\_init\_\_(self,ai\_settings,screen,stats):

self.screen=screen

self.screen\_rect=screen.get\_rect()

self.ai\_settings=ai\_settings

self.stats=stats

# 显示得分信息时使用的字体设置

self.text\_color=(30,30,30)

self.font=pygame.font.SysFont(None,48)

# 准备初始化得分图像

self.prep\_score()

self.prep\_high\_score()

self.prep\_level()

self.prep\_ships()

def prep\_score(self):

"""将得分转化为一幅渲染的图像"""

score\_str=str(self.stats.score)

# 分数格式

rounded\_score=int(round(self.stats.score,-1))

score\_str="{:,}".format(rounded\_score)

self.score\_image=self.font.render(score\_str,True,self.text\_color,self.ai\_settings.bg\_color)

# 将文字放到左上角

self.score\_rect=self.score\_image.get\_rect()

self.score\_rect.right=self.screen\_rect.right-20

self.score\_rect.top=20

def prep\_high\_score(self):

high\_score=int(round(self.stats.high\_score,-1))

high\_score\_str="{:,}".format(high\_score)

self.high\_score\_image=self.font.render(high\_score\_str,True,self.text\_color,self.ai\_settings.bg\_color)

# 位置

self.high\_score\_rect=self.high\_score\_image.get\_rect()

self.high\_score\_rect.centerx=self.screen\_rect.centerx

self.high\_score\_rect.top=20

def prep\_level(self):

self.level\_image=self.font.render(str(self.stats.level),True,self.text\_color,self.ai\_settings.bg\_color)

self.level\_rect=self.level\_image.get\_rect()

self.level\_rect.right=self.screen\_rect.right-20

self.level\_rect.top=self.score\_rect.bottom+10

def show\_score(self):

"""在屏幕上显示得分"""

self.screen.blit(self.score\_image,self.score\_rect)

self.screen.blit(self.high\_score\_image,self.high\_score\_rect)

self.screen.blit(self.level\_image,self.level\_rect)

self.ships.draw(self.screen)

def prep\_ships(self):

self.ships=Group()

for ship\_number in range(self.stats.ship\_left):

ship=Ship(self.ai\_settings,self.screen)

ship.rect.x=10+ship\_number\*ship.rect.width

ship.rect.y=10

self.ships.add(ship)

8. settings.py

class Settings():

"""储存游戏的所有设置的类"""

def \_\_init\_\_(self):

"""初始化屏幕的设置"""

# 屏幕的设置

self.screen\_width=1200

self.screen\_height=700

self.bg\_color=(230,230,230)

# 飞船设置

self.ship\_speed\_factor=3

self.ship\_limit=3

#子弹设置

self.bullet\_speed\_factor=3

self.bullet\_width=3

self.bullet\_height=15

self.bullet\_color=60,60,60

self.bullet\_allowed=5

# 外星人设置

self.alien\_speed\_factor=1

self.fleet\_drop\_speed=10

# 1 表示向右 -1 表示想做

self.fleet\_direction=1

# 添加游戏节奏的比例

self.speedup\_scale=1.1

#计分比例

self.aliens\_score=1.5

self.initialize\_dynamic\_settings()

def initialize\_dynamic\_settings(self):

# 初始化随游戏的变化而变化的设置

self.ship\_speed\_factor=3

self.bullet\_speed\_factor=3

self.alien\_speed\_factor=1

self.fleet\_direction=1

# 计分

self.alien\_points=50

def increas\_speed(self):

"""提高速度设置"""

self.ship\_speed\_factor\*=self.speedup\_scale

self.bullet\_speed\_factor\*=self.speedup\_scale

self.alien\_speed\_factor\*=self.speedup\_scale

self.alien\_points=int(self.alien\_points\*self.aliens\_score)

9. ship.py

import pygame

from pygame.sprite import Sprite

class Ship(Sprite):

def \_\_init\_\_(self,ai\_settings,screen):

super().\_\_init\_\_()

"""初始化飞船，并初始化位置"""

self.screen=screen

self.ai\_settings=ai\_settings

# 加载飞船图像

self.image = pygame.image.load('images/ship.bmp')

# 获取飞船的外部的矩形

self.rect = self.image.get\_rect()

# 获取屏幕的矩形

self.screen\_rect=self.screen.get\_rect()

# 将每艘飞船放入到屏幕的中央

self.rect.centerx=self.screen\_rect.centerx

self.rect.bottom=self.screen\_rect.bottom

# 给飞船的属性centerx中存储小数值

self.center=float(self.rect.centerx)

# 左右移动的标志

self.moving\_right=False

self.moving\_left=False

def blitme(self):

"""在指定位置绘制飞船"""

self.screen.blit(self.image,self.rect)

def update(self):

"""根据移动标志调整飞船的位置"""

if self.moving\_right and self.rect.right<self.screen\_rect.right:

self.center+=self.ai\_settings.ship\_speed\_factor

elif self.moving\_left and self.rect.left>0:

self.center-=self.ai\_settings.ship\_speed\_factor

# 根据self.center 更新rect对象

self.rect.centerx=self.center

def center\_ship(self):

self.center=self.screen\_rect.centerx