·山西大学自动化与软件学院

课程　 软件质量保证与测试

作业名称 软件项目走查，开发组:游戏-飞机大战

学生班级/组别 软工2004班 2组

开发小组成员 张继 赵海乐 张伟 杨向征

分工

指导教师 郭新峰

2023年 4月 6日

参考资料

《零基础入门学习python》



目录

[飞机大战 2](#_Toc121679267)

[一.游戏设定 2](#_Toc121679268)

[二.我方飞机 3](#_Toc121679269)

[# 分别定义moveUp()、moveDown()、moveLeft()和moveRight()控制我方飞机上、下、左、右移动: 4](#_Toc121679270)

[三.敌方飞机 4](#_Toc121679271)

[四.发射子弹 6](#_Toc121679272)

[五.发放补给包 7](#_Toc121679273)

[六.主模块 8](#_Toc121679274)

[# 生成我方飞机 9](#_Toc121679275)

[# 生成敌方小型飞机 10](#_Toc121679276)

[# 生成敌方中型飞机 10](#_Toc121679277)

[# 生成敌方大型飞机 10](#_Toc121679278)

[# 中弹图片索引 10](#_Toc121679279)

[# 统计得分 10](#_Toc121679280)

[# 标志是否暂停游戏 10](#_Toc121679281)

[# 设置难度级别 10](#_Toc121679282)

[# 全屏炸弹 10](#_Toc121679283)

[# 每30秒发放一个补给包 10](#_Toc121679284)

[# 超级子弹定时器 10](#_Toc121679285)

[# 标志是否使用超级子弹 11](#_Toc121679286)

[# 解除我方无敌状态定时器 11](#_Toc121679287)

[# 生命数量 11](#_Toc121679288)

[# 用于阻止重复打开记录文件 11](#_Toc121679289)

[# 游戏结束画面 11](#_Toc121679290)

[# 用于切换图片 11](#_Toc121679291)

[# 用于延迟 11](#_Toc121679292)

[# 根据用户的得分增加难度 12](#_Toc121679293)

[# 增加3架小型敌机、2架中型敌机和1架大型敌机 12](#_Toc121679294)

[# 增加5架小型敌机、3架中型敌机和2架大型敌机 12](#_Toc121679295)

[# 增加5架小型敌机、3架中型敌机和2架大型敌机 12](#_Toc121679296)

[# 增加5架小型敌机、3架中型敌机和2架大型敌机 12](#_Toc121679297)

[# 检测用户的键盘操作 12](#_Toc121679298)

[# 绘制全屏炸弹补给并检测是否获得 13](#_Toc121679299)

[# 绘制超级子弹补给并检测是否获得 13](#_Toc121679300)

[# 发射子弹 13](#_Toc121679301)

[# 检测子弹是否击中敌机 13](#_Toc121679302)

[# 绘制大型敌机 13](#_Toc121679303)

[# 绘制血槽 14](#_Toc121679304)

[# 当生命大于20%显示绿色，否则显示红色 14](#_Toc121679305)

[# 即将出现在画面中，播放音效 14](#_Toc121679306)

[# 毁 灭 14](#_Toc121679307)

[# 绘制中型敌机： 14](#_Toc121679308)

[# 绘制血槽 14](#_Toc121679309)

[# 当生命大于20%显示绿色，否则显示红色 14](#_Toc121679310)

[# 毁 灭 14](#_Toc121679311)

[# 绘制小型敌机： 15](#_Toc121679312)

[# 毁 灭 15](#_Toc121679313)

[# 检测我方飞机是否被撞 15](#_Toc121679314)

[# 绘制我方飞机 15](#_Toc121679315)

[# 毁 灭 15](#_Toc121679316)

[# 绘制全屏炸弹数量 15](#_Toc121679317)

[# 绘制剩余生命数量 15](#_Toc121679318)

[# 绘制得分 15](#_Toc121679319)

[# 绘制游戏结束画面 15](#_Toc121679320)

[# 背景音乐停止 15](#_Toc121679321)

[# 停止全部音效 15](#_Toc121679322)

[# 停止发放补给 16](#_Toc121679323)

[# 读取历史最高得分 16](#_Toc121679324)

[# 如果玩家得分高于历史最高得分，则存档 16](#_Toc121679325)

[# 绘制结束画面 16](#_Toc121679326)

[# 检测用户的鼠标操作 16](#_Toc121679327)

[# 获取鼠标坐标 16](#_Toc121679328)

[# 如果用户点击“重新开始” 16](#_Toc121679329)

[# 调用main函数，重新开始游戏 16](#_Toc121679330)

[# 如果用户点击“结束游戏” 16](#_Toc121679331)

[# 绘制暂停按钮 16](#_Toc121679332)

[# 切换图片 16](#_Toc121679333)

# 飞机大战

# 一.游戏设定

游戏界面如下图所示游戏的基本设定:

敌方共有大中小3款飞机，分为高中低三种速度; 子弹的射程并非全屏,而大概是屏幕长度的80%;

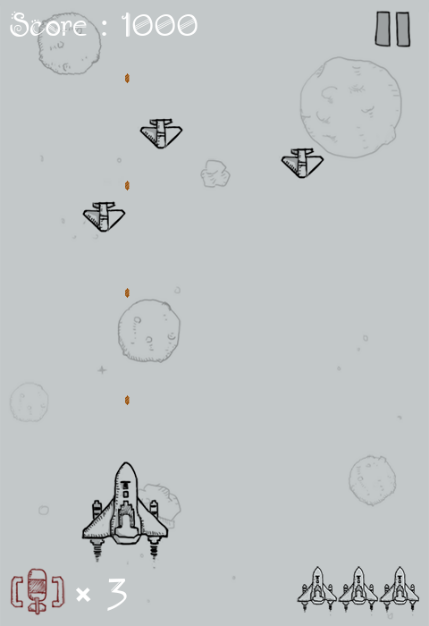
消灭小飞机需要1发子弹,中飞机需要8发,大飞机需要20发子弹; 每消灭一架小飞机得1000分,中飞机6000分,大飞机10000分;

每隔30秒有一个随机的道具补给,分为两种道具，全屏炸弹和双倍子弹; 全屏炸弹最多只能存放3枚,双倍子弹可以维持18秒钟的效果;

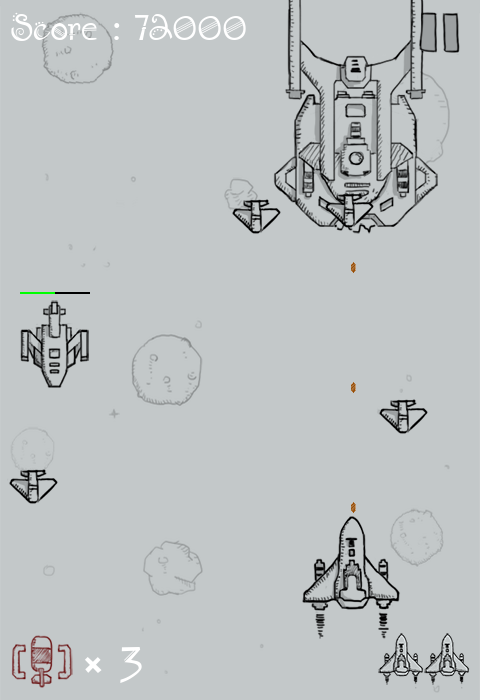
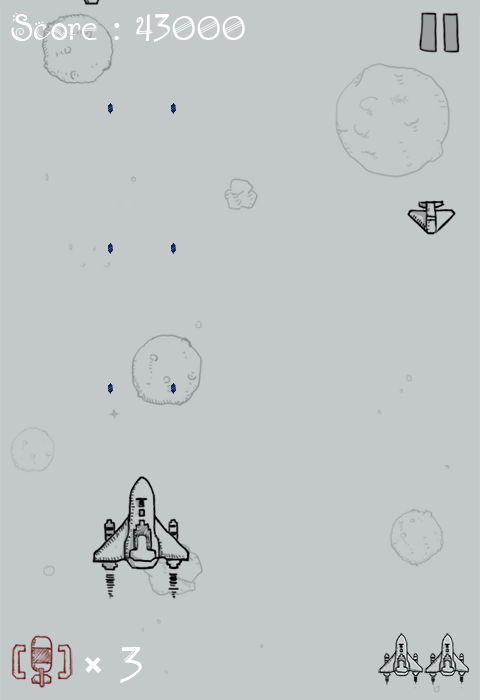
【空格触发炸弹上限为3个，超出不计】

游戏将根据分数来逐步提高难度,难度的提高表现为飞机数量的增多以及速度的加快。

另外还对游戏做了一些改进,比如为中飞机和大飞机增加了血槽的显示,这样玩家可以直观地知道敌机快被消灭了没有;我方有 三次机会,每次被敌人消灭,新诞生的飞机会有3秒钟的安全期;游戏结束后会显示历史最高分数。



**双倍子弹触发**

****

# 二.我方飞机

首先创建一个myplane. py模块来定义我方飞机:

[import](#_bookmark0) pygame

[class](#_bookmark1) MyPlane(pygame.sprite.Sprite): def init (self, bg\_size):

pygame.sprite.Sprite. init (self)

self.image1 = pygame.image.load("images/me1.png").convert\_alpha() self.image2 = pygame.image.load("images/me2.png").convert\_alpha() self.destroy\_images = []

self.destroy\_images.extend([\ pygame.image.load("images/me\_destroy\_1.png").convert\_alpha(), \ pygame.image.load("images/me\_destroy\_2.png").convert\_alpha(), \ pygame.image.load("images/me\_destroy\_3.png").convert\_alpha(), \ pygame.image.load("images/me\_destroy\_4.png").convert\_alpha() \

])

self.rect = self.image1.get\_rect()

self.width, self.height = bg\_size[0], bg\_size[1] self.rect.left, self.rect.top = \

(self.width - self.rect.width) // 2, \ self.height - self.rect.height - 60

self.speed = 10 self.active = True self.invincible = False

self.mask = pygame.mask.from\_surface(self.image1)

## # 分别定义moveUp()、moveDown()、moveLeft()和moveRight()控制我方飞机上、下、左、右移动:

def moveUp(self):

if self.rect.top > 0: self.rect.top -= self.speed

else:

self.rect.top = 0

def moveDown(self):

if self.rect.bottom < self.height - 60: self.rect.top += self.speed

else:

self.rect.bottom = self.height - 60

def moveLeft(self):

if self.rect.left > 0: self.rect.left -= self.speed

else:

self.rect.left = 0

def moveRight(self):

if self.rect.right < self.width: self.rect.left += self.speed

else:

self.rect.right = self.width

def reset(self):

self.rect.left, self.rect.top = \

(self.width - self.rect.width) // 2, \ self.height - self.rect.height - 60

self.active = True self.invincible = True

# 三.敌方飞机

敌机分为小、中、大三个尺寸,它们的速度依次是快、中、慢,在游戏界面的上方位置创造位置随机的敌机,可以让它们不在同一排出现。将敌机的定义写在enemy. py模块中:

[import](#_bookmark49) pygame

[from](#_bookmark50) random import \*

[class](#_bookmark51) SmallEnemy(pygame.sprite.Sprite): def init (self, bg\_size):

pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/enemy1.png").convert\_alpha() self.destroy\_images = []

self.destroy\_images.extend([ \ pygame.image.load("images/enemy1\_down1.png").convert\_alpha(), \ pygame.image.load("images/enemy1\_down2.png").convert\_alpha(), \ pygame.image.load("images/enemy1\_down3.png").convert\_alpha(), \ pygame.image.load("images/enemy1\_down4.png").convert\_alpha() \

])

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

self.width, self.height = bg\_size[0], bg\_size[1] self.speed = 2

self.active = True self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-5 \* self.height, 0)

self.mask = pygame.mask.from\_surface(self.image)

def move(self):

if self.rect.top < self.height: self.rect.top += self.speed

else:

self.reset()

def reset(self): self.active = True

self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-5 \* self.height, 0)

[class](#_bookmark83) MidEnemy(pygame.sprite.Sprite): energy = 8

def init (self, bg\_size): pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/enemy2.png").convert\_alpha() self.image\_hit = pygame.image.load("images/enemy2\_hit.png").convert\_alpha() self.destroy\_images = []

self.destroy\_images.extend([ \ pygame.image.load("images/enemy2\_down1.png").convert\_alpha(), \ pygame.image.load("images/enemy2\_down2.png").convert\_alpha(), \ pygame.image.load("images/enemy2\_down3.png").convert\_alpha(), \ pygame.image.load("images/enemy2\_down4.png").convert\_alpha() \

])

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

self.width, self.height = bg\_size[0], bg\_size[1] self.speed = 1

self.active = True self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-10 \* self.height, -self.height)

self.mask = pygame.mask.from\_surface(self.image) self.energy = MidEnemy.energy

self.hit = False

def move(self):

if self.rect.top < self.height:

self.rect.top += self.speed else:

self.reset()

def reset(self): self.active = True

self.energy = MidEnemy.energy self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-10 \* self.height, -self.height)

[class](#_bookmark121) BigEnemy(pygame.sprite.Sprite): energy = 20

def init (self, bg\_size): pygame.sprite.Sprite. init (self)

self.image1 = pygame.image.load("images/enemy3\_n1.png").convert\_alpha() self.image2 = pygame.image.load("images/enemy3\_n2.png").convert\_alpha() self.image\_hit = pygame.image.load("images/enemy3\_hit.png").convert\_alpha() self.destroy\_images = []

self.destroy\_images.extend([ \ pygame.image.load("images/enemy3\_down1.png").convert\_alpha(), \ pygame.image.load("images/enemy3\_down2.png").convert\_alpha(), \ pygame.image.load("images/enemy3\_down3.png").convert\_alpha(), \ pygame.image.load("images/enemy3\_down4.png").convert\_alpha(), \ pygame.image.load("images/enemy3\_down5.png").convert\_alpha(), \ pygame.image.load("images/enemy3\_down6.png").convert\_alpha() \

])

self.rect = self.image1.get\_rect()

self.width, self.height = bg\_size[0], bg\_size[1] self.speed = 1

self.active = True self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-15 \* self.height, -5 \* self.height)

self.mask = pygame.mask.from\_surface(self.image1) self.energy = BigEnemy.energy

self.hit = False

def move(self):

if self.rect.top < self.height: self.rect.top += self.speed

else:

self.reset()

def reset(self): self.active = True

self.energy = BigEnemy.energy self.rect.left, self.rect.top = \

randint(0, self.width - self.rect.width), \ randint(-15 \* self.height, -5 \* self.height)

# 四.发射子弹

接下来定义子弹,子弹分为两种: 一种是普通子弹一次只发射一颗; 另一种是补给发放的超级子弹一次可以发射两颗。

我们将子弹定义为独立的模块bullet.py:

[import](#_bookmark162) pygame

[class](#_bookmark163) Bullet1(pygame.sprite.Sprite): def init (self, position):

pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/bullet1.png").convert\_alpha() self.rect = self.image.get\_rect()

self.rect.left, self.rect.top = position self.speed = 12

self.active = False

self.mask = pygame.mask.from\_surface(self.image)

def move(self):

self.rect.top -= self.speed

if self.rect.top < 0: self.active = False

def reset(self, position):

self.rect.left, self.rect.top = position self.active = True

[class](#_bookmark183) Bullet2(pygame.sprite.Sprite): def init (self, position):

pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/bullet2.png").convert\_alpha() self.rect = self.image.get\_rect()

self.rect.left, self.rect.top = position self.speed = 14

self.active = False

self.mask = pygame.mask.from\_surface(self.image)

def move(self):

self.rect.top -= self.speed

if self.rect.top < 0: self.active = False

def reset(self, position):

self.rect.left, self.rect.top = position self.active = True

# 五.发放补给包

游戏设计每30秒随机发放一个补给包,可能是超级子弹,也可能是全屏炸弹。补给包有自己的图像和运动轨迹,不妨单独为其定义一个模块supply.py:

[import](#_bookmark203) pygame

[from](#_bookmark204) random import \*

[class](#_bookmark205) Bullet\_Supply(pygame.sprite.Sprite):

def init (self, bg\_size): pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/bullet\_supply.png").convert\_alpha() self.rect = self.image.get\_rect()

self.width, self.height = bg\_size[0], bg\_size[1] self.rect.left, self.rect.bottom = \

randint(0, self.width - self.rect.width), -100 self.speed = 5

self.active = False

self.mask = pygame.mask.from\_surface(self.image)

def move(self):

if self.rect.top < self.height: self.rect.top += self.speed

else:

self.active = False

def reset(self): self.active = True

self.rect.left, self.rect.bottom = \

randint(0, self.width - self.rect.width), -100

[class](#_bookmark228) Bomb\_Supply(pygame.sprite.Sprite):

def init (self, bg\_size): pygame.sprite.Sprite. init (self)

self.image = pygame.image.load("images/bomb\_supply.png").convert\_alpha() self.rect = self.image.get\_rect()

self.width, self.height = bg\_size[0], bg\_size[1] self.rect.left, self.rect.bottom = \

randint(0, self.width - self.rect.width), -100 self.speed = 5

self.active = False

self.mask = pygame.mask.from\_surface(self.image)

def move(self):

if self.rect.top < self.height: self.rect.top += self.speed

else:

self.active = False

def reset(self): self.active = True

self.rect.left, self.rect.bottom = \

randint(0, self.width - self.rect.width), -100

# 六.主模块

所有的模块编写完毕，接下来就该实现主模块了

[#](#_bookmark251) main.py [import](#_bookmark252) pygame [import](#_bookmark253) sys

[import](#_bookmark254) traceback [import](#_bookmark255) myplane [import](#_bookmark256) enemy [import](#_bookmark257) bullet [import](#_bookmark258) supply

[from](#_bookmark259) pygame.locals import \* [from](#_bookmark260) random import \*

[pygame.init()](#_bookmark261) [pygame.mixer.init()](#_bookmark262)

[bg\_size](#_bookmark263) = width, height = 480, 700

[screen](#_bookmark264) = pygame.display.set\_mode(bg\_size) [pygame.display.set\_caption(](#_bookmark265)"飞机大战 – zj`s Game")

[background](#_bookmark266) = pygame.image.load("images/background.png").convert()

[BLACK](#_bookmark267) = (0, 0, 0)

[WHITE](#_bookmark268) = (255, 255, 255)

[GREEN](#_bookmark269) = (0, 255, 0)

[RED](#_bookmark270) = (255, 0, 0)

[#](#_bookmark271) 载入游戏音乐[pygame.mixer.music.load(](#_bookmark272)"sound/game\_music.ogg") [pygame.mixer.music.set\_volume(](#_bookmark273)0.2)

[bullet\_sound](#_bookmark274) = pygame.mixer.Sound("sound/bullet.wav") [bullet\_sound.set\_volume(](#_bookmark275)0.2)

[bomb\_sound](#_bookmark276) = pygame.mixer.Sound("sound/use\_bomb.wav") [bomb\_sound.set\_volume(](#_bookmark277)0.2)

[supply\_sound](#_bookmark278) = pygame.mixer.Sound("sound/supply.wav") [supply\_sound.set\_volume(](#_bookmark279)0.2)

[get\_bomb\_sound](#_bookmark280) = pygame.mixer.Sound("sound/get\_bomb.wav") [get\_bomb\_sound.set\_volume(](#_bookmark281)0.2)

[get\_bullet\_sound](#_bookmark282) = pygame.mixer.Sound("sound/get\_bullet.wav") [get\_bullet\_sound.set\_volume(](#_bookmark283)0.2)

[upgrade\_sound](#_bookmark284) = pygame.mixer.Sound("sound/upgrade.wav") [upgrade\_sound.set\_volume(](#_bookmark285)0.2)

[enemy3\_fly\_sound](#_bookmark286) = pygame.mixer.Sound("sound/enemy3\_flying.wav") [enemy3\_fly\_sound.set\_volume(](#_bookmark287)0.2)

[enemy1\_down\_sound](#_bookmark288) = pygame.mixer.Sound("sound/enemy1\_down.wav") [enemy1\_down\_sound.set\_volume(](#_bookmark289)0.2)

[enemy2\_down\_sound](#_bookmark290) = pygame.mixer.Sound("sound/enemy2\_down.wav") [enemy2\_down\_sound.set\_volume(](#_bookmark291)0.2)

[enemy3\_down\_sound](#_bookmark292) = pygame.mixer.Sound("sound/enemy3\_down.wav") [enemy3\_down\_sound.set\_volume(](#_bookmark293)0.5)

[me\_down\_sound](#_bookmark294) = pygame.mixer.Sound("sound/me\_down.wav") [me\_down\_sound.set\_volume(](#_bookmark295)0.2)

[def](#_bookmark296) add\_small\_enemies(group1, group2, num): for i in range(num):

e1 = enemy.SmallEnemy(bg\_size) group1.add(e1)

group2.add(e1)

[def](#_bookmark301) add\_mid\_enemies(group1, group2, num): for i in range(num):

e2 = enemy.MidEnemy(bg\_size) group1.add(e2) group2.add(e2)

[def](#_bookmark306) add\_big\_enemies(group1, group2, num): for i in range(num):

e3 = enemy.BigEnemy(bg\_size) group1.add(e3) group2.add(e3)

[def](#_bookmark311) inc\_speed(target, inc): for each in target:

each.speed += inc

[def](#_bookmark314) main():

pygame.mixer.music.play(-1)

## # 生成我方飞机

me = myplane.MyPlane(bg\_size)

enemies = pygame.sprite.Group()

## # 生成敌方小型飞机

small\_enemies = pygame.sprite.Group() add\_small\_enemies(small\_enemies, enemies, 15)

## # 生成敌方中型飞机

mid\_enemies = pygame.sprite.Group() add\_mid\_enemies(mid\_enemies, enemies, 4)

## # 生成敌方大型飞机

big\_enemies = pygame.sprite.Group() add\_big\_enemies(big\_enemies, enemies, 2)

# 生成普通子弹bullet1 = [] bullet1\_index = 0

BULLET1\_NUM = 4

for i in range(BULLET1\_NUM): bullet1.append(bullet.Bullet1(me.rect.midtop))

# 生成超级子弹bullet2 = [] bullet2\_index = 0

BULLET2\_NUM = 8

for i in range(BULLET2\_NUM // 2): bullet2.append(bullet.Bullet2((me.rect.centerx - 33, me.rect.centery))) bullet2.append(bullet.Bullet2((me.rect.centerx + 30, me.rect.centery)))

clock = pygame.time.Clock()

## # 中弹图片索引

e1\_destroy\_index = 0

e2\_destroy\_index = 0

e3\_destroy\_index = 0

me\_destroy\_index = 0

## # 统计得分

score = 0

score\_font = pygame.font.Font("font/font.ttf", 36)

## # 标志是否暂停游戏

paused = False

pause\_nor\_image = pygame.image.load("images/pause\_nor.png").convert\_alpha() pause\_pressed\_image = pygame.image.load("images/pause\_pressed.png").convert\_alpha() resume\_nor\_image = pygame.image.load("images/resume\_nor.png").convert\_alpha() resume\_pressed\_image = pygame.image.load("images/resume\_pressed.png").convert\_alpha() paused\_rect = pause\_nor\_image.get\_rect()

paused\_rect.left, paused\_rect.top = width - paused\_rect.width - 10, 10 paused\_image = pause\_nor\_image

## # 设置难度级别

level = 1

## # 全屏炸弹

bomb\_image = pygame.image.load("images/bomb.png").convert\_alpha() bomb\_rect = bomb\_image.get\_rect()

bomb\_font = pygame.font.Font("font/font.ttf", 48) bomb\_num = 3

## # 每30秒发放一个补给包

bullet\_supply = supply.Bullet\_Supply(bg\_size) bomb\_supply = supply.Bomb\_Supply(bg\_size) SUPPLY\_TIME = USEREVENT

pygame.time.set\_timer(SUPPLY\_TIME, 30 \* 1000)

## # 超级子弹定时器

DOUBLE\_BULLET\_TIME = USEREVENT + 1

## # 标志是否使用超级子弹

is\_double\_bullet = False

## # 解除我方无敌状态定时器

INVINCIBLE\_TIME = USEREVENT + 2

## # 生命数量

life\_image = pygame.image.load("images/life.png").convert\_alpha() life\_rect = life\_image.get\_rect()

life\_num = 3

## # 用于阻止重复打开记录文件

recorded = False

## # 游戏结束画面

gameover\_font = pygame.font.Font("font/font.TTF", 48)

again\_image = pygame.image.load("images/again.png").convert\_alpha() again\_rect = again\_image.get\_rect()

gameover\_image = pygame.image.load("images/gameover.png").convert\_alpha() gameover\_rect = gameover\_image.get\_rect()

## # 用于切换图片

switch\_image = True

## # 用于延迟

delay = 100 running = True

while running:

for event in pygame.event.get(): if event.type == QUIT:

pygame.quit() sys.exit()

elif event.type == MOUSEBUTTONDOWN:

if event.button == 1 and paused\_rect.collidepoint(event.pos): paused = not paused

if paused:

pygame.time.set\_timer(SUPPLY\_TIME, 0) pygame.mixer.music.pause() pygame.mixer.pause()

else:

pygame.time.set\_timer(SUPPLY\_TIME, 30 \* 1000) pygame.mixer.music.unpause() pygame.mixer.unpause()

elif event.type == MOUSEMOTION:

if paused\_rect.collidepoint(event.pos): if paused:

paused\_image = resume\_pressed\_image else:

paused\_image = pause\_pressed\_image

else:

if paused:

paused\_image = resume\_nor\_image else:

paused\_image = pause\_nor\_image

elif event.type == KEYDOWN: if event.key == K\_SPACE:

if bomb\_num:

bomb\_num -= 1 bomb\_sound.play()  for each in enemies:

if each.rect.bottom > 0:

each.active = False

elif event.type == SUPPLY\_TIME: supply\_sound.play()

if choice([True, False]): bomb\_supply.reset()

else:

bullet\_supply.reset()

elif event.type == DOUBLE\_BULLET\_TIME: is\_double\_bullet = False pygame.time.set\_timer(DOUBLE\_BULLET\_TIME, 0)

elif event.type == INVINCIBLE\_TIME: me.invincible = False pygame.time.set\_timer(INVINCIBLE\_TIME, 0)

## # 根据用户的得分增加难度

if level == 1 and score > 50000: level = 2 upgrade\_sound.play()

## # 增加3架小型敌机、2架中型敌机和1架大型敌机

add\_small\_enemies(small\_enemies, enemies, 3)

add\_mid\_enemies(mid\_enemies, enemies, 2)

add\_big\_enemies(big\_enemies, enemies, 1) # 提升小型敌机的速度inc\_speed(small\_enemies, 1)

elif level == 2 and score > 300000: level = 3

upgrade\_sound.play()

## # 增加5架小型敌机、3架中型敌机和2架大型敌机

add\_small\_enemies(small\_enemies, enemies, 5)

add\_mid\_enemies(mid\_enemies, enemies, 3)

add\_big\_enemies(big\_enemies, enemies, 2) # 提升小型敌机的速度inc\_speed(small\_enemies, 1)

inc\_speed(mid\_enemies, 1)

elif level == 3 and score > 600000: level = 4

upgrade\_sound.play()

## # 增加5架小型敌机、3架中型敌机和2架大型敌机

add\_small\_enemies(small\_enemies, enemies, 5)

add\_mid\_enemies(mid\_enemies, enemies, 3)

add\_big\_enemies(big\_enemies, enemies, 2) # 提升小型敌机的速度inc\_speed(small\_enemies, 1)

inc\_speed(mid\_enemies, 1)

elif level == 4 and score > 1000000: level = 5

upgrade\_sound.play()

## # 增加5架小型敌机、3架中型敌机和2架大型敌机

add\_small\_enemies(small\_enemies, enemies, 5)

add\_mid\_enemies(mid\_enemies, enemies, 3)

add\_big\_enemies(big\_enemies, enemies, 2) # 提升小型敌机的速度inc\_speed(small\_enemies, 1)

inc\_speed(mid\_enemies, 1)

screen.blit(background, (0, 0))

if life\_num and not paused:

## # 检测用户的键盘操作

key\_pressed = pygame.key.get\_pressed()

if key\_pressed[K\_w] or key\_pressed[K\_UP]: me.moveUp()

if key\_pressed[K\_s] or key\_pressed[K\_DOWN]:

me.moveDown()

if key\_pressed[K\_a] or key\_pressed[K\_LEFT]: me.moveLeft()

if key\_pressed[K\_d] or key\_pressed[K\_RIGHT]: me.moveRight()

## # 绘制全屏炸弹补给并检测是否获得

if bomb\_supply.active: bomb\_supply.move()

screen.blit(bomb\_supply.image, bomb\_supply.rect) if pygame.sprite.collide\_mask(bomb\_supply, me):

get\_bomb\_sound.play() if bomb\_num < 3:

bomb\_num += 1 bomb\_supply.active = False

## # 绘制超级子弹补给并检测是否获得

if bullet\_supply.active: bullet\_supply.move()

screen.blit(bullet\_supply.image, bullet\_supply.rect) if pygame.sprite.collide\_mask(bullet\_supply, me):

get\_bullet\_sound.play() is\_double\_bullet = True

pygame.time.set\_timer(DOUBLE\_BULLET\_TIME, 18 \* 1000) bullet\_supply.active = False

## # 发射子弹

if not (delay % 10): bullet\_sound.play() if is\_double\_bullet:

bullets = bullet2

bullets[bullet2\_index].reset((me.rect.centerx - 33, me.rect.centery)) bullets[bullet2\_index + 1].reset((me.rect.centerx + 30, me.rect.centery)) bullet2\_index = (bullet2\_index + 2) % BULLET2\_NUM

else:

bullets = bullet1 bullets[bullet1\_index].reset(me.rect.midtop) bullet1\_index = (bullet1\_index + 1) % BULLET1\_NUM

## # 检测子弹是否击中敌机

for b in bullets: if b.active:

b.move() screen.blit(b.image, b.rect)

enemy\_hit = pygame.sprite.spritecollide(b, enemies, False, pygame.sprite.collide\_mask) if enemy\_hit:

b.active = False for e in enemy\_hit:

if e in mid\_enemies or e in big\_enemies: e.hit = True

e.energy -= 1

if e.energy == 0: e.active = False

else:

e.active = False

## # 绘制大型敌机

for each in big\_enemies: if each.active:

each.move() if each.hit:

screen.blit(each.image\_hit, each.rect) each.hit = False

else:

if switch\_image: screen.blit(each.image1, each.rect)

else:

screen.blit(each.image2, each.rect)

## # 绘制血槽

pygame.draw.line(screen, BLACK, \

(each.rect.left, each.rect.top - 5), \ (each.rect.right, each.rect.top - 5), \ 2)

## # 当生命大于20%显示绿色，否则显示红色

energy\_remain = each.energy / enemy.BigEnemy.energy if energy\_remain > 0.2:

energy\_color = GREEN else:

energy\_color = RED pygame.draw.line(screen, energy\_color, \

(each.rect.left, each.rect.top - 5), \ (each.rect.left + each.rect.width \* energy\_remain, \ each.rect.top - 5), 2)

## # 即将出现在画面中，播放音效

if each.rect.bottom == -50: enemy3\_fly\_sound.play(-1)

else:

## # 毁 灭

if not (delay % 3):

if e3\_destroy\_index == 0: enemy3\_down\_sound.play()

screen.blit(each.destroy\_images[e3\_destroy\_index], each.rect) e3\_destroy\_index = (e3\_destroy\_index + 1) % 6

if e3\_destroy\_index == 0: enemy3\_fly\_sound.stop() score += 10000 each.reset()

## # 绘制中型敌机：

for each in mid\_enemies: if each.active:

each.move()

if each.hit:

screen.blit(each.image\_hit, each.rect) each.hit = False

else:

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

## # 绘制血槽

pygame.draw.line(screen, BLACK, \

(each.rect.left, each.rect.top - 5), \ (each.rect.right, each.rect.top - 5), \ 2)

## # 当生命大于20%显示绿色，否则显示红色

energy\_remain = each.energy / enemy.MidEnemy.energy if energy\_remain > 0.2:

energy\_color = GREEN else:

energy\_color = RED pygame.draw.line(screen, energy\_color, \

(each.rect.left, each.rect.top - 5), \ (each.rect.left + each.rect.width \* energy\_remain, \ each.rect.top - 5), 2)

else:

## # 毁 灭

if not (delay % 3):

if e2\_destroy\_index == 0: enemy2\_down\_sound.play()

screen.blit(each.destroy\_images[e2\_destroy\_index], each.rect) e2\_destroy\_index = (e2\_destroy\_index + 1) % 4

if e2\_destroy\_index == 0:

score += 6000 each.reset()

## # 绘制小型敌机：

for each in small\_enemies: if each.active:

each.move() screen.blit(each.image, each.rect)

else:

## # 毁 灭

if not (delay % 3):

if e1\_destroy\_index == 0: enemy1\_down\_sound.play()

screen.blit(each.destroy\_images[e1\_destroy\_index], each.rect) e1\_destroy\_index = (e1\_destroy\_index + 1) % 4

if e1\_destroy\_index == 0: score += 1000 each.reset()

## # 检测我方飞机是否被撞

enemies\_down = pygame.sprite.spritecollide(me, enemies, False, pygame.sprite.collide\_mask) if enemies\_down and not me.invincible:

me.active = False

for e in enemies\_down: e.active = False

## # 绘制我方飞机

if me.active:

if switch\_image: screen.blit(me.image1, me.rect)

else:

screen.blit(me.image2, me.rect)

else:

## # 毁 灭

if not (delay % 3):

if me\_destroy\_index == 0: me\_down\_sound.play()

screen.blit(me.destroy\_images[me\_destroy\_index], me.rect) me\_destroy\_index = (me\_destroy\_index + 1) % 4

if me\_destroy\_index == 0: life\_num -= 1 me.reset()

pygame.time.set\_timer(INVINCIBLE\_TIME, 3 \* 1000)

## # 绘制全屏炸弹数量

bomb\_text = bomb\_font.render("× %d" % bomb\_num, True, WHITE) text\_rect = bomb\_text.get\_rect()

screen.blit(bomb\_image, (10, height - 10 - bomb\_rect.height)) screen.blit(bomb\_text, (20 + bomb\_rect.width, height - 5 - text\_rect.height))

## # 绘制剩余生命数量

if life\_num:

for i in range(life\_num): screen.blit(life\_image, \

(width - 10 - (i + 1) \* life\_rect.width, \ height - 10 - life\_rect.height))

## # 绘制得分

score\_text = score\_font.render("Score : %s" % str(score), True, WHITE) screen.blit(score\_text, (10, 5))

## # 绘制游戏结束画面

elif life\_num == 0:

## # 背景音乐停止

pygame.mixer.music.stop()

## # 停止全部音效

pygame.mixer.stop()

## # 停止发放补给

pygame.time.set\_timer(SUPPLY\_TIME, 0)

if not recorded: recorded = True

## # 读取历史最高得分

with open("record.txt", "r") as f: record\_score = int(f.read())

## # 如果玩家得分高于历史最高得分，则存档

if score > record\_score:

with open("record.txt", "w") as f: f.write(str(score))

## # 绘制结束画面

record\_score\_text = score\_font.render("Best : %d" % record\_score, True, (255, 255, 255)) screen.blit(record\_score\_text, (50, 50))

gameover\_text1 = gameover\_font.render("Your Score", True, (255, 255, 255)) gameover\_text1\_rect = gameover\_text1.get\_rect()

gameover\_text1\_rect.left, gameover\_text1\_rect.top = \ (width - gameover\_text1\_rect.width) // 2, height // 3

screen.blit(gameover\_text1, gameover\_text1\_rect)

gameover\_text2 = gameover\_font.render(str(score), True, (255, 255, 255)) gameover\_text2\_rect = gameover\_text2.get\_rect() gameover\_text2\_rect.left, gameover\_text2\_rect.top = \

(width - gameover\_text2\_rect.width) // 2, \ gameover\_text1\_rect.bottom + 10

screen.blit(gameover\_text2, gameover\_text2\_rect)

again\_rect.left, again\_rect.top = \ (width - again\_rect.width) // 2, \ gameover\_text2\_rect.bottom + 50

screen.blit(again\_image, again\_rect)

gameover\_rect.left, gameover\_rect.top = \ (width - again\_rect.width) // 2, \ again\_rect.bottom + 10

screen.blit(gameover\_image, gameover\_rect)

## # 检测用户的鼠标操作

# 如果用户按下鼠标左键

if pygame.mouse.get\_pressed()[0]:

## # 获取鼠标坐标

pos = pygame.mouse.get\_pos()

## # 如果用户点击“重新开始”

if again\_rect.left < pos[0] < again\_rect.right and \ again\_rect.top < pos[1] < again\_rect.bottom:

## # 调用main函数，重新开始游戏

main()

## # 如果用户点击“结束游戏”

elif gameover\_rect.left < pos[0] < gameover\_rect.right and \ gameover\_rect.top < pos[1] < gameover\_rect.bottom:

# 退出游戏pygame.quit() sys.exit()

## # 绘制暂停按钮

screen.blit(paused\_image, paused\_rect)

## # 切换图片

if not (delay % 5):

switch\_image = not switch\_image

delay -= 1

if not delay:

delay = 100

pygame.display.flip() clock.tick(60)

[if](#_bookmark796) name == " main ": try:

main()

except SystemExit: pass

except:

traceback.print\_exc() pygame.quit()

input()

**测试注意事项：** 所有的模块应该放在同一个文件夹，最后运行主模块

# 七.改进对照与问题解决

## 1.首先面对的是pygame包的导入和使用，这里提供思路：

**安装Pygame**

Windows使用pip命令进行安装，如果使用conda 则需要在conda prompt中对已激活环境进行安装

pip install pygame

**导入Pygame模块**

**import** pygame

我们在使用Pygame设计一款游戏或动画的时候，需要包括三个部分：

**一、初始设置**

首先需要导入模块，创建游戏屏幕，然后初始化一些重要的变量

**二、游戏循环**

游戏循环使用while循环，使游戏持续更新屏幕和处理事件，除非用户退出程序，否则就会一直执行

**三、退出程序**

当用户想要停止程序的时候，使用一种方式来结束程序

最简单的就例如：去绘制一个圆

**import** pygame

*# 初始设置*

pygame**.**init() *# 初始化pygame*

screen **=** pygame**.**display**.**set\_mode((800,600)) *# Pygame窗口*

pygame**.**display**.**set\_caption("Pygame绘制图形") *# 标题*

keep\_going **=** True

RED **=** (255,0,0) *# 红色，使用RGB颜色*

radius **=** 20 *# 半径*

*# 游戏循环*

**while** keep\_going:

**for** event **in** pygame**.**event**.**get(): *# 遍历事件*

**if** event**.**type **==** pygame**.**QUIT: *# 退出事件*

keep\_going **=** False

pygame**.**draw**.**circle(screen,RED,(200,300),radius)

pygame**.**display**.**update() *# 刷新屏幕*

*# 退出程序*

pygame**.**quit()

1.初始设置

这里我们首先导入了pygame模块，然后在使用pygame.init()对其进行初始化。pygame.display.set\_mode((800,600))创建了一个800\*600像素的显示窗口，保存到变量screen中，以后的程序中就可以直接调用变量screen即可。

pygame.display.set\_caption("Pygame绘制图形")给屏幕窗口添加一个标题。

在Pygame中创建的游戏窗口，或者加载到窗口的图形都称为Surface。Surface screen即显示窗口，是绘制其他所有图形的主要窗口。

我们定义了一个布尔类型的标志，来控制程序的持续运行，想要暂停，只需要改变keep\_going的值为False。

2.游戏循环

while keep\_going:如果keep\_going为True游戏一直持续进行，持续运行Pygame窗口，直到用户选择退出。

for event in pygame.event.get():这里使用for循环遍历Pygame中的所有事件，if event.type == pygame.QUIT:如果用户按下关闭按钮，退出程序，这样变量keep\_going变为False，游戏结束。

游戏持续进行的时候，即用户没有触发关闭窗口的事件。我们开始在Pygame窗口绘制其他Surface图像，pygame.draw.circle(screen,RED,(200,300),radius)在屏幕窗口(200,300)位置上绘制半径为20的，填充颜色为RED的圆。

第一个参数screen，指定要在那个Surface上绘制图形。pygame.draw除了可以绘制圆（circle），还可以绘制矩形、线段等形状。

**3.加载图片**

将我们保存的图片加载到Surface屏幕上。注意图片保存的位置和我们.py文件在同一个目录下。

pic = pygame.image.load("logo\_lofi.png")从一个文件载入图像，保存到变量pic中，在以后的程序中通过pic来引用这个图片。

将图片绘制到Surface屏幕上并且更新绘制窗口，确保所有内容出现在屏幕上。

screen**.**blit(pic,(100,100)) *# 在Surface上绘制图片*

pygame**.**display**.**update()

blit()方法将我们从文件中加载的图片pic绘制到screen屏幕的（100，100）位置上。

## 2.写到键盘控制，发现无法正常运行

def **cotral():**

for **event** in **pygame.event.get():**

if **event.type** == **quit:**

**print("exit")**

elif **event.type** == **KEYDOWN:**

if **event.type** == **K\_LEFT** or **event.type** == **K\_a:**

**print("left")**

elif **event.type** == **K\_RIGHT** or **event.type** == **K\_d:**

**print("right")**

elif **event.type** == **K\_SPACE:**

**print("space")**

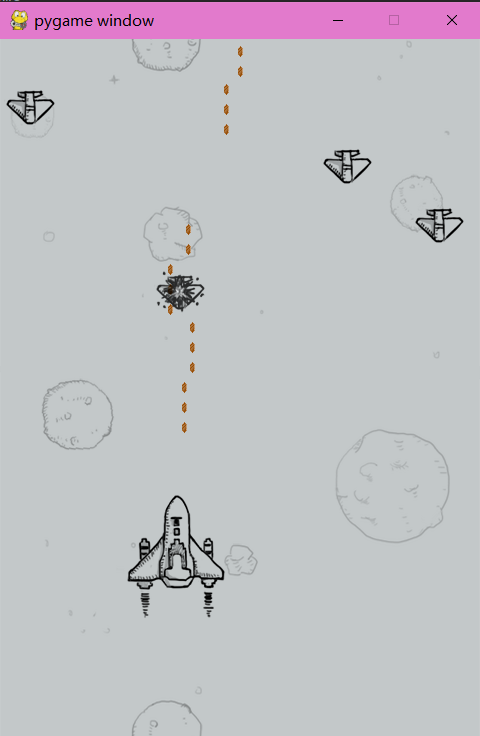
else**:**

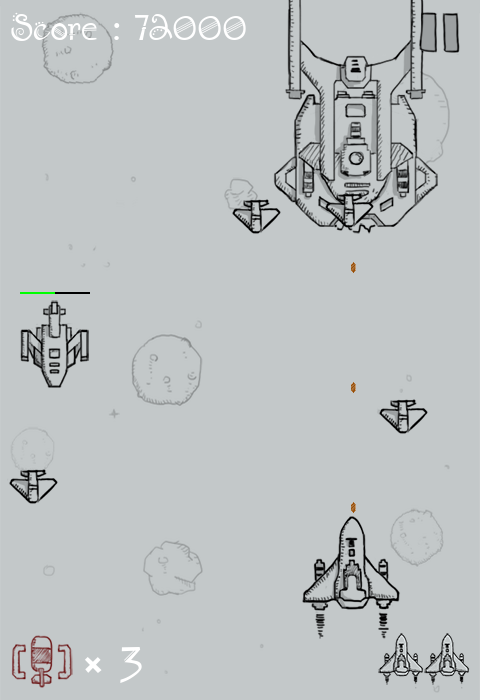
**print("战机损毁...")**

在查阅资料后了解到需要将event.type改为使用event.key 按键事件类型方法使用不当，导致在按下按键后，战机无实际反应。

## 3.在处理这些问题之后，开始优化游戏逻辑，未考虑用户体验，在全组范围内综合讨论与分析后，决定作出以下更改：

1>增加战机类型，并设置按得分使得难度递增变化，由原来的一种类型变更为现在的大中小三型，并按照合适比例设置了战机血条显示，参考主模块文档。下面是对照图

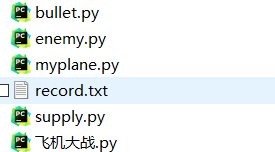


****

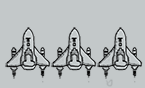
2>增加计分板，在每次游戏结束总得分以文本形式将得分记录在本地

3>新增暂停，重新开始退出等交互按键





4>设置全屏炸弹及生命值，并显示在游戏框内，添加随机补给包，强化子弹补给以及炸弹补给



5>死亡动画并设置复活后无敌时间

6>新增游戏音效

7>优化我方战机移动策略，由固定纵轴变更为可以在画布上以水平和垂直四个方向进行运动