# 我们的飞机大战游戏还有一些问题，如游戏结束不够友好，没有再玩功能，还有一个就是不是完全面向对象，我们这里先做第一个，就是添加再玩功能

## 1.将所有的类定义放到一个叫做sprites.py模块中

|  |
| --- |
|  |

## 2.首先需要解决游戏面向对象的问题，新建一个main.py模块,在里面新建一个Game类，把funs模块里面的函数全部剪切到Game类中，需要适当修改代码。

## 3.把menu模块里面的方法全部剪切到main模块里面的函数全部剪切到main模块中，适当修改

## 4.把draw\_end\_text函数和quit\_menu函数删除，因为我们用不到他们。

## 5.在Game类的初始化方法里面添加如下代码

|  |
| --- |
|  |

## 6.给Game类添加一个load\_datas方法，把原来各个精灵模块的类定义前面的代码统统剪切到这里，适当修改把他们变为Game类的类属性，load\_datas方法的代码如下

|  |
| --- |
| def load\_datas(self):            # 加载玩家图片          self.player\_img1 = pg.image.load(pic\_path+"my1.png")          self.player\_mini\_img1 = pg.transform.scale(self.player\_img1,(20,19))          self.player\_mini\_img1.set\_colorkey(BLACK)          self.shooting\_sound = pg.mixer.Sound(sound\_path+'pew-gunshot-13.wav')          self.enemies\_images = [] #保存敌机图片对象的列表          self.enemies\_list = [              'dj1.png',              'dj2.png',              'dj3.png'          ]          # 加载敌机          for img in self.enemies\_list:              enemies\_img = pg.image.load(pic\_path+img)              enemies\_img = pg.transform.scale(enemies\_img,(80,60))              self.enemies\_images.append(enemies\_img)          self.enemy\_bullet\_img = pg.transform.scale(pg.image.load(pic\_path+'enemy\_bullet.png'),(15,25))          self.ene\_shoot\_sound = pg.mixer.Sound(sound\_path+'enemy\_bullet.wav')          #加载爆炸图片          self.explosion\_anim = {}          self.explosion\_anim['sm'] = []          self.explosion\_anim['lg'] = []          self.explosion\_anim['player'] = []          for i in range(8): #敌机，火山石爆炸              filename = 'dd{}.png'.format(i+1)              img = pg.image.load(pic\_path + filename)              img.set\_colorkey(BLACK)              #大爆炸              img\_lg = pg.transform.scale(img,(75,75))              self.explosion\_anim['lg'].append(img\_lg)              #小爆炸              img\_sm = pg.transform.scale(img,(32,32))              self.explosion\_anim['sm'].append(img\_sm)              # 玩家爆炸              filename = 'sonic{}.png'.format(i+1)              img2 = pg.image.load(pic\_path + filename)              img2.set\_colorkey(BLACK)              self.explosion\_anim['player'].append(img2)          # 加载熔岩图片          self.lava\_list = []          self.lava\_pics = ['lava\_med.png','lava\_small1.png','lava\_small2.png','lava\_small3.png']          for pic in self.lava\_pics:              lava = pg.image.load(pic\_path+pic)              self.lava\_list.append(lava)          #导弹音效          self.missile\_sound = pg.mixer.Sound(sound\_path+'237071-Rocket\_Launcher-02.wav')          # 加载盾牌和闪电          self.powerup\_images = {}          self.powerup\_images['shield'] = pg.image.load(pic\_path+'shield.png')          self.powerup\_images['gun'] = pg.image.load(pic\_path+'bolt2.png') |

## 7.把原来主程序的main方法的代码风格了，给Game模块新建一个new方法，粘贴main方法的新建精灵组和精灵代码的代码，然后调用run方法如下

|  |
| --- |
| def new(self):          #0.播放背景音乐          pg.mixer.music.stop()          pg.mixer.music.load(sound\_path+'battle.ogg')          pg.mixer.music.set\_volume(0.8)          pg.mixer.music.play(-1)          # 1.创建精灵组          # 所有精灵的精灵组          self.all\_sprites = pg.sprite.Group()          # 我方子弹精灵组          self.bullets = pg.sprite.Group()          # 敌机精灵组          self.enemies = pg.sprite.Group()          # 敌人子弹精灵组          self.enemy\_bullets = pg.sprite.Group()          # 补给精灵组          self.powers = pg.sprite.Group()          # 熔岩碎石精灵组          self.lavas = pg.sprite.Group()          #2.创建精灵           # 3.创建玩家飞机          self.player1 = Plane(self,self.player\_img1,WIDTH/2,HEIGHT-30,pg.K\_LEFT,pg.K\_RIGHT,pg.K\_UP,pg.K\_DOWN)          # 4.将他们添加到精灵组,这是必须的，因为只有精灵组才有绘制方法          self.all\_sprites.add(self.player1)            # 创建敌机          for i in range(4):              self.new\_enemy() # 这个方法可以创建敌机并且添加到all\_sprites和enemies精灵组里面          # 创建熔岩碎石          for i in range(4):              self.new\_lava()          # 调用精灵组的更新方法          self.all\_sprites.update()          # 3.调用run方法          self.run() |

## 8.给Game类添加一个run方法，把除了初始化以外的代码粘贴过来，注意这里需要包running改为新建属性playing，run方法是游戏主循环，如果它退出，说明本次游戏结束game over，但是程序并不会退出，他会显示一个game over界面，你可以按任意键重新开始游戏，也可以点击x退出程序

|  |
| --- |
| # run方法是游戏的主循环      def run(self):          global height          self.playing = True          while self.playing:              self.clock.tick(FPS)              for event in pg.event.get():                  if event.type == pg.QUIT:                      if self.playing:                          self.playing = False                      self.running = False              # 背景图片向下滚动              self.screen.blit(self.bg,(0,height))              height += 2              if height > -168:                  height = -936              if self.player1.lives == 0: # 判断是否是游戏结束                  self.player1.shield = 0                  self.playing = False              self.all\_sprites.update()       # 调用精灵组的更新方法              # 调用精灵组的更新方法              self.all\_sprites.draw(self.screen)              self.bullet\_hit\_enemy()              # 子弹打中敌人的碰撞检测              self.plane\_get\_power(self.player1)        #我方飞机获取补给的碰撞检测              self.enemy\_hit\_me(self.player1)           #敌机子弹打中我方飞机              self.plane\_crash(self.player1)            #我方飞机和敌机碰撞检测              self.bullet\_hit\_lava()               #双方子弹打熔岩碎石的碰撞检测              self.lava\_hit\_both\_planes(self.player1)   #熔岩碎石碰撞双方飞机的碰撞检测              self.bullet\_vs\_enemy\_bullet()         # 双方子弹的碰撞检测              self.draw\_screen\_text(self.player1) # 绘制血条和飞机架数              pg.display.update() |

## 9.给Game类添加一个exit\_or\_replay方法，当游戏结束，显示一个界面让用户选择重新玩游戏还是退出，会播放game over音乐

|  |
| --- |
|  |

# 魔改后的代码只有3给模块，main.py是入口模块，constants.py是一些游戏设置，sprites.py是有些精灵类，魔改后的代码完整代码如下

## constants.py

|  |
| --- |
| # 实战图片和音效图片路径  pic\_path = './res/images/'  sound\_path = './res/sounds/'  WIDTH = 480      #窗口宽度  HEIGHT = 600     #窗口高度  SIZE = (WIDTH,HEIGHT) #  FPS = 30   # 帧率  # 定义颜色  BLACK = (0,0,0)  WHITE = (255,255,255)  RED = (255,0,0)  GREEN = (0,255,0)  BLUE = (0,0,255)  YELLOW = (255,255,0)  height = -936  POWERUP\_TIME = 5000  #飞机的火力持续时间  BAR\_LENGTH =100  #血条长度  BAR\_HEIGHT = 10  #血条高度 |

## sprites.py

|  |
| --- |
| import pygame as pg  from constants import \*  import random as rnd  # 加载玩家飞机图片  class Bullet(pg.sprite.Sprite):      def \_\_init\_\_(self, x,y) -> None:          super().\_\_init\_\_()          self.image = pg.transform.scale(pg.image.load(pic\_path+'pd333.png'),(15,40))          # self.image = pg.transform.scale(pg.image.load(pic\_path+'enemy\_bullet2.png'),(15,40))          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.bottom = y          self.rect.centerx = x          self.speed = -10      def update(self, ) -> None:         self.rect.y += self.speed         if self.rect.bottom < 0:             self.kill() # 子弹出界了就销毁  # 敌机类  class Enemy(pg.sprite.Sprite):      def \_\_init\_\_(self,game) -> None:          self.game = game          pg.sprite.Sprite.\_\_init\_\_(self) # 调用父类构造函数          self.image\_orig = rnd.choice(self.game.enemies\_images) # 随机获取一张图片          self.image\_orig.set\_colorkey(BLACK)          self.image = self.image\_orig.copy()          self.rect = self.image.get\_rect()          self.radius = int(self.rect.width\*.90/2)          self.rect.x = rnd.randrange(0,WIDTH-self.rect.width)          self.rect.y = rnd.randrange(-150,-100)          self.speedy = rnd.randrange(2,5)          self.speedx = rnd.randrange(-3,3)          self.shoot\_delay = 1000          self.last\_shot = pg.time.get\_ticks()      def update(self):          self.rect.x += self.speedx          self.rect.y += self.speedy          if rnd.randrange(10) >= 6:              self.enemy\_shoot()          # 超出范围敌机重生          if(self.rect.top > HEIGHT+10) or (self.rect.left < -25) or (self.rect.right > WIDTH+20):              self.rect.x = rnd.randrange(0,WIDTH-self.rect.width)              self.rect.y = rnd.randrange(-100,-40)              self.speedy = rnd.randrange(1,5)          # 碰到两边会反弹          if self.rect.left < 0 :              self.speedx = -self.speedx          if self.rect.right > WIDTH:              self.speedx = -self.speedx      def enemy\_shoot(self):          now = pg.time.get\_ticks()          if now - self.last\_shot > self.shoot\_delay:              self.last\_shot = now              # d敌机创建子弹              enemy\_bullet = EnemyBullet(self.game,self.rect.centerx,self.rect.bottom)              self.game.all\_sprites.add(enemy\_bullet)              self.game.enemy\_bullets.add(enemy\_bullet)              self.game.ene\_shoot\_sound.play()    class EnemyBullet(pg.sprite.Sprite):      def \_\_init\_\_(self,game,x,y) -> None:          self.game = game          pg.sprite.Sprite.\_\_init\_\_(self) # 这里不能用super(),必须用pg.sprite.Sprite.\_\_init\_\_(self)          self.image = self.game.enemy\_bullet\_img          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.centerx = x          self.rect.top = y          self.speedy = 5          self.radius = 5      def update(self):          self.rect.y += self.speedy          if self.rect.top > HEIGHT: # 超出屏幕的子弹会非销毁              self.kill()    # 爆炸类  class Explosion(pg.sprite.Sprite):      def \_\_init\_\_(self,game,center,size):          self.game = game          pg.sprite.Sprite.\_\_init\_\_(self)          self.size = size          self.image = self.game.explosion\_anim[size][0]          self.rect = self.image.get\_rect()          self.rect.center = center          self.frame = 0          self.last\_update = pg.time.get\_ticks()          self.frma\_rate = 75      def update(self) -> None:          now = pg.time.get\_ticks()          if now - self.last\_update > self.frma\_rate:              self.last\_update = now              self.frame += 1              if self.frame == len(self.game.explosion\_anim[self.size]):                  self.kill()              else:                 center = self.rect.center # 获取上一帧的中心点                 self.image = self.game.explosion\_anim[self.size][self.frame]                 self.rect = self.image.get\_rect()                 self.rect.center = center   # 设置到当前帧的中心点    class Lava(pg.sprite.Sprite):      def \_\_init\_\_(self,game):          pg.sprite.Sprite.\_\_init\_\_(self)          self.image = rnd.choice(game.lava\_list)          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.x = rnd.randint(0,WIDTH)          self.rect.y = rnd.randint(0,HEIGHT)          self.radius = 10          self.speedy = 8      def update(self):          self.rect.y += self.speedy          if self.rect.top > HEIGHT:              self.rect.x =  rnd.randint(0,WIDTH)              self.rect.y = rnd.randint(0,HEIGHT)  class Missile(pg.sprite.Sprite):      def \_\_init\_\_(self, x,y) -> None:          super().\_\_init\_\_()          self.image = pg.transform.scale(pg.image.load(pic\_path+'missile3.png'),(20,55))          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.bottom = y          self.rect.centerx = x          self.speed = -10      def update(self, ) -> None:         self.rect.y += self.speed         if self.rect.bottom < 0:             self.kill() # 子弹出界了就销毁  class Plane(pg.sprite.Sprite):      """"玩家飞机类"""      def \_\_init\_\_(self,game, playerImg,center,bottom,K\_LEFT,K\_RIGHT,K\_UP,K\_DOWN) -> None:          self.game = game          super().\_\_init\_\_()          self.image = pg.transform.scale(playerImg,(50,38))          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.centerx = center          self.rect.bottom = bottom          self.speed = 5          self.shield = 100 #血量          self.redius = 20 #杀伤力          self.shoot\_delay = 250 #子弹延迟          self.last\_shot = pg.time.get\_ticks() #最后一次射击时间          self.lives = 3 # 飞机架数          self.hidden = False          self.hide\_timer = pg.time.get\_ticks()          self.power = 3          self.power\_timer = pg.time.get\_ticks() # 火力时间          self.K\_LEFT = K\_LEFT          self.K\_RIGHT = K\_RIGHT          self.K\_UP = K\_UP          self.K\_DOWN = K\_DOWN      def update(self) -> None:          # super().update()          if self.power >=2 and pg.time.get\_ticks() - self.power\_timer > POWERUP\_TIME:              self.power -= 1              self.power\_timer = pg.time.get\_ticks()          if self.hidden and  pg.time.get\_ticks() - self.hide\_timer > 1000:              self.hidden = False              self.rect.centerx = WIDTH/2              self.rect.bottom = HEIGHT - 30          self.shoot()    # 是自动发射子弹的          self.move()     # 设置玩家移动边界        def move(self):          keys = pg.key.get\_pressed() # 获取所有按下的键          if keys[self.K\_RIGHT]:              if self.rect.right > WIDTH: # 右边越界                  self.rect.right = WIDTH              else:                  self.rect.centerx += self.speed          if keys[self.K\_LEFT]:              if self.rect.left < 0 :# 左边越界                  self.rect.left = 0              else:                  self.rect.centerx -= self.speed          if keys[self.K\_UP]:              if self.rect.y < 10:                  self.rect.top = 10              else:                  self.rect.top -= self.speed          if keys[self.K\_DOWN]:              if self.rect.bottom > HEIGHT-10:                  self.rect.bottom = HEIGHT-10              else:                   self.rect.bottom += self.speed      def shoot(self):          now = pg.time.get\_ticks() # 获取现在的时间          if now - self.last\_shot > self.shoot\_delay:              self.last\_shot = now # 保存最新的时间              #单火力              if self.power ==1:                  bullet0 = Bullet(self.rect.centerx,self.rect.top)                  # bullet0 = Missile(self.rect.centerx,self.rect.top)                    # # 子弹需要添加到2个精灵组                  self.game.all\_sprites.add(bullet0)                  self.game.bullets.add(bullet0)                  self.game.shooting\_sound.set\_volume(0.7)                  self.game.shooting\_sound.play()              #双火力              if self.power ==2:                  bullet1 = Bullet(self.rect.left,self.rect.centery)                  bullet2 = Bullet(self.rect.right,self.rect.centery)                  self.game.all\_sprites.add(bullet1)                  self.game.bullets.add(bullet1)                  self.game.all\_sprites.add(bullet2)                  self.game.bullets.add(bullet2)                  self.game.shooting\_sound.set\_volume(0.7)                  self.game.shooting\_sound.play()              #三火力              if self.power >=3:                  bullet1 = Bullet(self.rect.left,self.rect.centery)                  bullet2 = Bullet(self.rect.right,self.rect.centery)                  missile1 = Missile(self.rect.centerx,self.rect.top)                  self.game.all\_sprites.add(bullet1)                  self.game.bullets.add(bullet1)                  self.game.all\_sprites.add(bullet2)                  self.game.bullets.add(bullet2)                  self.game.all\_sprites.add(missile1)                  self.game.bullets.add(missile1)                  self.game.shooting\_sound.set\_volume(0.7)                  self.game.shooting\_sound.play()                  self.game.missile\_sound.play()      def powerup(self):          self.power += 3          self.power\_timer = pg.time.get\_ticks()      def hide(self):          self.hidden = True          self.hide\_timer = pg.time.get\_ticks()          self.rect.center = (WIDTH/2,HEIGHT+200)  class Power(pg.sprite.Sprite):      def \_\_init\_\_(self, game,center) -> None:          pg.sprite.Sprite.\_\_init\_\_(self)          self.type = rnd.choice(['shield','gun'])          self.image = game.powerup\_images[self.type]          self.image.set\_colorkey(BLACK)          self.rect = self.image.get\_rect()          self.rect.center = center          self.speedy = 2      def update(self):          self.rect.y += self.speedy          if self.rect.top > HEIGHT:              self.kill() |

## main.py

|  |
| --- |
| import pygame as pg  import sys  from constants import \*  from sprites import \*  class Game:      def \_\_init\_\_(self) -> None:          pg.init()       #初始化pygame模块          pg.mixer.init() #初始化pygame模块的音效模块          self.game\_title = 'Kenny飞机大战游戏'          self.font\_name = pg.font.match\_font('arial') # 获取系统里面安装了的字体名称          self.screen = pg.display.set\_mode(SIZE)          pg.display.set\_caption(self.game\_title)          self.clock = pg.time.Clock()          self.score = 0          self.bg = pg.image.load(pic\_path+'startfield.jpg')          self.bg = pg.transform.scale(self.bg,(WIDTH,1536))          self.running = True          self.load\_datas()        def load\_datas(self):            # 加载玩家图片          self.player\_img1 = pg.image.load(pic\_path+"my1.png")          self.player\_mini\_img1 = pg.transform.scale(self.player\_img1,(20,19))          self.player\_mini\_img1.set\_colorkey(BLACK)          self.shooting\_sound = pg.mixer.Sound(sound\_path+'pew-gunshot-13.wav')          self.enemies\_images = [] #保存敌机图片对象的列表          self.enemies\_list = [              'dj1.png',              'dj2.png',              'dj3.png'          ]          # 加载敌机          for img in self.enemies\_list:              enemies\_img = pg.image.load(pic\_path+img)              enemies\_img = pg.transform.scale(enemies\_img,(80,60))              self.enemies\_images.append(enemies\_img)          self.enemy\_bullet\_img = pg.transform.scale(pg.image.load(pic\_path+'enemy\_bullet.png'),(15,25))          self.ene\_shoot\_sound = pg.mixer.Sound(sound\_path+'enemy\_bullet.wav')          #加载爆炸图片          self.explosion\_anim = {}          self.explosion\_anim['sm'] = []          self.explosion\_anim['lg'] = []          self.explosion\_anim['player'] = []          for i in range(8): #敌机，火山石爆炸              filename = 'dd{}.png'.format(i+1)              img = pg.image.load(pic\_path + filename)              img.set\_colorkey(BLACK)              #大爆炸              img\_lg = pg.transform.scale(img,(75,75))              self.explosion\_anim['lg'].append(img\_lg)              #小爆炸              img\_sm = pg.transform.scale(img,(32,32))              self.explosion\_anim['sm'].append(img\_sm)              # 玩家爆炸              filename = 'sonic{}.png'.format(i+1)              img2 = pg.image.load(pic\_path + filename)              img2.set\_colorkey(BLACK)              self.explosion\_anim['player'].append(img2)          # 加载熔岩图片          self.lava\_list = []          self.lava\_pics = ['lava\_med.png','lava\_small1.png','lava\_small2.png','lava\_small3.png']          for pic in self.lava\_pics:              lava = pg.image.load(pic\_path+pic)              self.lava\_list.append(lava)          #导弹音效          self.missile\_sound = pg.mixer.Sound(sound\_path+'237071-Rocket\_Launcher-02.wav')          # 加载盾牌和闪电          self.powerup\_images = {}          self.powerup\_images['shield'] = pg.image.load(pic\_path+'shield.png')          self.powerup\_images['gun'] = pg.image.load(pic\_path+'bolt2.png')      def new(self):          #0.播放背景音乐          pg.mixer.music.stop()          pg.mixer.music.load(sound\_path+'battle.ogg')          pg.mixer.music.set\_volume(0.8)          pg.mixer.music.play(-1)          # 1.创建精灵组          # 所有精灵的精灵组          self.all\_sprites = pg.sprite.Group()          # 我方子弹精灵组          self.bullets = pg.sprite.Group()          # 敌机精灵组          self.enemies = pg.sprite.Group()          # 敌人子弹精灵组          self.enemy\_bullets = pg.sprite.Group()          # 补给精灵组          self.powers = pg.sprite.Group()          # 熔岩碎石精灵组          self.lavas = pg.sprite.Group()          #2.创建精灵           # 3.创建玩家飞机          self.player1 = Plane(self,self.player\_img1,WIDTH/2,HEIGHT-30,pg.K\_LEFT,pg.K\_RIGHT,pg.K\_UP,pg.K\_DOWN)          # 4.将他们添加到精灵组,这是必须的，因为只有精灵组才有绘制方法          self.all\_sprites.add(self.player1)            # 创建敌机          for i in range(4):              self.new\_enemy() # 这个方法可以创建敌机并且添加到all\_sprites和enemies精灵组里面          # 创建熔岩碎石          for i in range(4):              self.new\_lava()          # 调用精灵组的更新方法          self.all\_sprites.update()          # 3.调用run方法          self.run()      # run方法是游戏的主循环      def run(self):          global height          self.playing = True          while self.playing:              self.clock.tick(FPS)              for event in pg.event.get():                  if event.type == pg.QUIT:                      if self.playing:                          self.playing = False                      self.running = False              # 背景图片向下滚动              self.screen.blit(self.bg,(0,height))              height += 2              if height > -168:                  height = -936              if self.player1.lives == 0: # 判断是否是游戏结束                  self.player1.shield = 0                  self.playing = False              self.all\_sprites.update()       # 调用精灵组的更新方法              # 调用精灵组的更新方法              self.all\_sprites.draw(self.screen)              self.bullet\_hit\_enemy()              # 子弹打中敌人的碰撞检测              self.plane\_get\_power(self.player1)        #我方飞机获取补给的碰撞检测              self.enemy\_hit\_me(self.player1)           #敌机子弹打中我方飞机              self.plane\_crash(self.player1)            #我方飞机和敌机碰撞检测              self.bullet\_hit\_lava()               #双方子弹打熔岩碎石的碰撞检测              self.lava\_hit\_both\_planes(self.player1)   #熔岩碎石碰撞双方飞机的碰撞检测              self.bullet\_vs\_enemy\_bullet()         # 双方子弹的碰撞检测              self.draw\_screen\_text(self.player1) # 绘制血条和飞机架数              pg.display.update()        def exit\_or\_replay(self):          pg.mixer.music.load(sound\_path+'g-over.ogg')          pg.mixer.music.play(-1)          if not self.running:              return          self.screen.blit(pg.image.load(pic\_path+"menu.png"),(0,0))          self.draw\_text("Game Over",48,WIDTH/2,HEIGHT/2)          self.draw\_text("Press Any Key To Play Again",22,WIDTH/2,HEIGHT\*3/4)          pg.display.flip()          waiting = True          while waiting:              self.clock.tick(FPS)              for e in pg.event.get():                  if e.type == pg.QUIT:                      waiting = False                      self.running = False                  if e.type == pg.KEYUP:                      waiting = False        #绘制文本的函数      def draw\_text(self,text,size,x,y):          font = pg.font.Font(self.font\_name,size)          text\_surface = font.render(text,True,WHITE)          text\_rect = text\_surface.get\_rect()          text\_rect.midtop = (x,y)          self.screen.blit(text\_surface,text\_rect)      def draw\_screen\_text(self,player1):          self.draw\_text(str(self.score),18,WIDTH/2,10) #显示分数          self.draw\_shield\_bar(5,5,player1.shield)          self.draw\_lives(10,20,player1.lives,self.player\_mini\_img1)        # 绘制血条      def draw\_shield\_bar(self,x,y,pct):          pct = max(pct,0)          fill = (pct/100) \* BAR\_LENGTH          outline\_rect = pg.Rect(x,y,BAR\_LENGTH,BAR\_HEIGHT)          fill\_rect = pg.Rect(x,y,fill,BAR\_HEIGHT)          pg.draw.rect(self.screen,GREEN,fill\_rect)          pg.draw.rect(self.screen,WHITE,outline\_rect,2)      def draw\_lives(self,x,y,lives,img):          for i in range(lives):              img\_rect = img.get\_rect()              img\_rect.x = x + 30\*i              img\_rect.y = y              self.screen.blit(img,img\_rect)      # 创建敌机的函数      def new\_enemy(self):          enemy = Enemy(self)          self.all\_sprites.add(enemy)          self.enemies.add(enemy)      def bullet\_hit\_enemy(self):          """我方子弹打中敌人的函数"""          # 先进行我方子弹和敌机的碰撞检测          hits = pg.sprite.groupcollide(self.enemies,self.bullets,True,True,pg.sprite.collide\_mask)          for hit in hits:              self.score += 50-hit.radius              pg.mixer.Sound(sound\_path+"exp.wav").play()              #创建一个爆炸对象需要调用Explosion类              expl = Explosion(self,hit.rect.center,'lg')              # 将爆炸对象添加到所有精灵组              self.all\_sprites.add(expl)              if rnd.random()> 0.9:                  pow = Power(self,hit.rect.center)                  self.all\_sprites.add(pow)                  self.powers.add(pow)              # 每消灭一个敌机，又会创建一个敌机              self.new\_enemy()        # 我方飞机获取补给的方法，元素碰撞检测      def plane\_get\_power(self,player):          sound = pg.mixer.Sound(sound\_path+'FX054\_cut.wav')          hits = pg.sprite.spritecollide(player,self.powers,True)          for hit in hits:              if hit.type == 'shield':                  sound.play()                  player.shield += rnd.randrange(20,40)                  if player.shield >=100:                      player.shield = 100 # 血量不能超过100              elif hit.type == 'gun':                  sound.play()                  player.powerup()        #敌机子弹打中我方飞机      def enemy\_hit\_me(self,player):          hits = pg.sprite.spritecollide(player,self.enemy\_bullets,True,pg.sprite.collide\_mask)          for h in hits:              player.shield -= h.radius \*2 # 被打中会掉血              expl = Explosion(self,h.rect.center,'sm') # 创建爆炸对象，添加到小爆炸里集合面              self.all\_sprites.add(expl) # 将爆炸对象添加到所有精灵组              if player.shield <=0: # 血量掉光了就死掉了                  pg.mixer.Sound(sound\_path+'exp.wav').play() #播放爆炸音效                  dead\_expl = Explosion(self,player.rect.center,'player')                  self.all\_sprites.add(dead\_expl)                  player.hide() # 调用这个方法后几秒钟就会显示player                  player.lives -= 1 # 死掉了，就要减少一条命                  player.shield = 100 # 把player的血量设置位100，那么他就相当于新创建的了      # 我方飞机和敌机的碰撞检测      def plane\_crash(self,player):          hits = pg.sprite.spritecollide(player,self.enemies,True,pg.sprite.collide\_mask)          for hit in hits:              pg.mixer.Sound(sound\_path+"exp.wav").play()              #创建一个爆炸对象需要调用Explosion类              expl = Explosion(self,hit.rect.center,'lg')              # 将爆炸对象添加到所有精灵组              self.all\_sprites.add(expl)              if rnd.random()> 0.9:                  pow = Power(self,hit.rect.center)                  self.all\_sprites.add(pow)                  self.powers.add(pow)              # 每消灭一个敌机，又会创建一个敌机              self.new\_enemy()      #创建熔岩碎石      def new\_lava(self):          if rnd.random() >= 0.6:              lava = Lava(self)              self.all\_sprites.add(lava)              self.lavas.add(lava)      # 子弹到熔岩碎石的碰撞检测，应该双方的子弹都功能打熔岩      def bullet\_hit\_lava(self):          hits = pg.sprite.groupcollide(self.bullets,self.lavas,True,True,pg.sprite.collide\_mask)          for hit in hits:              expl = Explosion(self,hit.rect.center,'lg')              self.all\_sprites.add(expl)              self.new\_lava()          hits = pg.sprite.groupcollide(self.enemy\_bullets,self.lavas,True,True,pg.sprite.collide\_mask)          for hit in hits:              expl = Explosion(self,hit.rect.center,'lg')              self.all\_sprites.add(expl)              self.new\_lava()      # 熔岩碎石撞击双方飞机，我方是单人玩家，没有玩家精灵组      def lava\_hit\_both\_planes(self,player):          hits = pg.sprite.spritecollide(player,self.lavas,True,pg.sprite.collide\_mask)          for hit in hits:              expl = Explosion(self,hit.rect.center,'sm')              self.all\_sprites.add(expl)              player.shield -= hit.radius          for enemy in self.enemies:              hits = pg.sprite.spritecollide(enemy,self.lavas,True,pg.sprite.collide\_mask)              for hit in hits:                  expl = Explosion(self,hit.rect.center,'lg')                  self.all\_sprites.add(expl)                  enemy.kill()                  self.new\_enemy()      # 双方子弹的碰撞检测      def bullet\_vs\_enemy\_bullet(self):          hits = pg.sprite.groupcollide(self.bullets,self.enemy\_bullets,True,True,pg.sprite.collide\_mask)          for hit in hits:              expl = Explosion(self,hit.rect.center,'sm')              self.all\_sprites.add(expl)      def main\_menu(self):          #加载菜单音乐          pg.mixer.music.load(sound\_path + 'tgfcoder-FrozenJam-SeamlessLoop.ogg')          #循环播放菜单音乐          pg.mixer.music.play(-1)          # 加载开始图片          start\_img = pg.image.load(pic\_path+'menu.png')          start\_img = pg.transform.scale(start\_img,SIZE)          self.screen.blit(start\_img,(0,0))          pg.display.update()          while True:              event = pg.event.poll() # 只获取一个事件              if event.type == pg.KEYDOWN:                  if event.key == pg.K\_RETURN: # 回车键                      break              elif  event.type == pg.QUIT:                  pg.quit()                  quit()              else:                  self.draw\_text("Press [Enter] To Begin",30,WIDTH/2,HEIGHT/2)                  self.draw\_text("[↑]",30,WIDTH/2,2\*HEIGHT/3-40)                  self.draw\_text("[←] [↓]  [→] ",30,WIDTH/2,2\*HEIGHT/3)                  pg.display.update()            pg.display.update()        def menu\_display(self):          self.main\_menu()    if \_\_name\_\_ == '\_\_main\_\_':     g = Game()     g.menu\_display()     while g.running:         g.new()         g.exit\_or\_replay()     pg.quit()     sys.exit() |

# 第一次魔改成功，我会继续学习研究，会尝试再次魔改，尽量把它变为一个完美的游戏