# 这一节我们来学习逐帧动画和飞机类的开发，内容如下

# 1.逐帧动画和飞机类说明p52

|  |
| --- |
|  |
|  |
|  |
|  |

# 2.飞机类的基本功能说明p53

|  |
| --- |
|  |
|  |

# 3.用逐帧动画显示英雄飞机p54

|  |
| --- |
|  |
|  |

## 代码实现，

## 1.在game\_items.py模块里面定义一个飞机类，继承自GameSprite

|  |
| --- |
|  |
| **注意：笔记有错，指示面板的定义应该在定义飞机之后** |

## 2.测试，在game.py模块里面添加测试代码，找到创建英雄飞机的代码，把它从GameSprite修改位Plane

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

### 说明飞机类初步实现功能

# 4.减少动画帧数的说明p55

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# 5.飞机的火焰动画完善p56

## 1、先在game\_items.py模块里面添加应该常量FRAME\_INTERVAL

|  |
| --- |
|  |

## 2.然后在game.py模块的游戏循环外面定义应该控制逐帧动画的播放速度的变量frame\_counter

|  |
| --- |
|  |

## 3.在游戏执行的地方，也就是既不是暂停状态，也不是游戏结束状态的地方添加控制代码

|  |
| --- |
|  |

## 4.然后回到game\_items.py模块里面的Plane类修改它的updat方法代码

|  |
| --- |
|  |

## 5.测试一下，效果好很多

# 6.飞机类的属性和方法说明p57

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

# 7.优化飞机属性p58

## 根据上面的分析，我们对飞机的属性进行修改

|  |
| --- |
|  |
|  |

# 8.飞机的重置和更新p59

## 思路

|  |
| --- |
|  |
|  |
|  |

## 代码实现

## 1.修改飞机类的代码，添加上面的属性和方法

|  |
| --- |
|  |

## 2.给飞机类添加一个重置方法，reset\_plane

|  |
| --- |
|  |

## 3.修改飞机类的update方法

|  |
| --- |
|  |

# 9.测试改进版的飞机p60

## .测试代码

## 1. 我们需要在game.py模块中改进英雄飞机的创建方法，使用改进后的Plane类，这里出于测试需要我们给英雄飞机180的血量，实际上他是没有的

|  |
| --- |
|  |

### 测试一下，是能够创建飞机的

|  |
| --- |
|  |

## 2.先把英雄飞机变为类成员

|  |
| --- |
|  |

## 3. 然后我们模拟英雄飞机被击中以及被摧毁的效果

|  |
| --- |
|  |

### 效果：是能够看到爆炸效果的，而且会自动重置，说明重置功能也没有问题

# 10.敌人飞机类的功能描述p61

|  |
| --- |
|  |
|  |
|  |

# 11.定义敌人飞机类p62

## 1.回到game\_item.py模块，我们来新建一个敌机类继承自飞机类，他的构造函数笔记特别，需要根据输入的类型来创建对应的飞机

|  |
| --- |
|  |

## 2.给敌机类添加一个reset\_plane方法，他其实是重写父类的方法，需要先调用父类的方法，然后再编写自己的数据重置代码

|  |
| --- |
|  |

# 12.根据关卡不同创建不同数量的飞机的方法说明p63

## 上面我们以及定义了敌机类，现在我们先来测试一下，看看他能够成功创建敌机，我们先来看看思路

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

# 13.根据关卡不同创建不同数量的飞机p64

## 明白了思路之后，可以进行代码实现

## 1.回到game.py模块，在Game类中添加create\_enemies方法

|  |
| --- |
|  |

## 2.在游戏的\_\_init\_\_方法里面调用上面的代码

|  |
| --- |
|  |

## 测试，敌机是创建了，但是由于没有调整位置，所有的敌机都重叠堆放在左上角，不过说明还是可以创建敌机的

|  |
| --- |
|  |

## 这里的代码只会执行一次，如果需要测试算法能够创建中大飞机，我们还需要在start方法里面游戏事件里面的加法代码里面添加创建飞机的代码

|  |
| --- |
|  |

## 效果：是可以创建的

|  |
| --- |
|  |

### 我们需要解决的就是敌机随机出现的位置的计算问题

# 14.敌人飞机出现位置的随机算法p65

## 思路说明

|  |
| --- |
|  |
|  |
|  |

# 15.敌人飞机出现位置的随机算法代码实现p66

## 1.打开game\_items.py模块，进入Enemy类的reset\_plane方法，添加如下代码

|  |
| --- |
|  |

## 注意：上面的代码只是方便我们测试敌机出现的随机位置，在实际项目中敌机创建的时候是先在屏幕外面创建，然后慢慢移动进入屏幕的

## 2.然后需要在Enemy类的\_\_init\_\_方法中调用这个重置方法修改刚刚创建的飞机的位置

|  |
| --- |
|  |

## 3.运行game.py测试一下，

|  |
| --- |
|  |

## 4.为了更好的测试，我们在对按键b的事件处理方法中修改爆炸代码，先吧模拟英雄牺牲的代码注释掉，然后添加炸飞机代码，其实很简单，只需要把所有飞机的血量设置为0即可，因为在飞机类的update方法里面已经设置好了爆炸图片以及它的播放条件。利用逐帧动画播放爆炸效果，而敌机类是继承自飞机类的，它可以继承父类的方法

|  |
| --- |
|  |

效果：

|  |
| --- |
|  |
|  |

### 说明这个方法可以正常工作

# 16.敌机飞行p67

## 思路

|  |
| --- |
|  |
|  |
|  |

## 代码实现

## 1.修改Enemy类的reset\_plane方法，把他们的初始位置设置到屏幕外面，然后根据最大速度给他们随机生成飞行速度

|  |
| --- |
|  |

## 2.在Enemy类中重写父类的update方法，其实就是在父类的基础上添加飞机出界处理

|  |
| --- |
|  |

## 3.把加分的测试代码删除

# 17.爆炸效果的bug修复p68

## 英雄飞机爆炸图片代码的加载代码的range里面应该是range(1,5)而不是(1,4)因为range是包头不包尾，需要取4，就必须把尾改为5

|  |
| --- |
|  |

## 在Enemy类里面的构造函数的range里面的尾数也需要修改

|  |
| --- |
|  |

# 18.英雄飞机的处理步骤说明p69

|  |
| --- |
|  |
|  |

# 19.飞机功能说明p70

## 英雄飞机发功能如下

|  |
| --- |
|  |
|  |

# 20.定义英雄飞机类p71

## 1.打开game\_item.py模块，在上面的常量区域定义两个常量炸弹数量和默认英雄飞机的位置

|  |
| --- |
|  |

## 2.在game\_item.py模块中定义英雄飞机类

|  |
| --- |
|  |

## 3.在game.py模块中，把创建英雄飞机的代码修一下，用Hero类来创建，而且不需要设置位置

|  |
| --- |
|  |

## 4.测试一下,英雄飞机的位置其实是在中间靠近底部的位置

|  |
| --- |
|  |

# 21.让控制面板显示飞机的炸弹数量p72

## 这里要解决应该问题，就是以前屏幕上面的炸弹数量是我们写死的，这是不对的，我们应该根据英雄飞机的炸弹数量来显示，需要在创建英雄飞机后修改炸弹数量，我们把创建面板的代码移动到创建英雄飞机的上面，然后在创建英雄飞机后修改炸弹数量

|  |
| --- |
|  |

# 22.移动英雄飞机需要的监听机制p73

## 思路分析

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# 23.get\_pressed方法的基本使用p74

## 进入game.py模块，在start方法里面的else代码块里面添加连续按键函数对右箭头按键的事件处理代码

|  |
| --- |
|  |

### 效果，你一直按着，飞机就一直往右移动

# 24.飞机移动效果的完善p75

## 了解了上面的判断原理之后，我们来移动英雄飞机

|  |
| --- |
|  |

## 代码实现

## 1.我们回到game.py模块，把长按键的处理代码修改一下

|  |
| --- |
|  |

## 2.回到game\_item.py模块中，在Hero类中重写父类的update方法

|  |
| --- |
|  |

### 效果：现在能够中有移动英雄飞机了，但是还没有做出界处理

## 3.其实英雄飞机的移动是有条件的，首先传递进来的args的长度必须是3，其次英雄飞机的血量必须大于0，并且我们来做出界处理我们来修改代码

|  |
| --- |
|  |

### 效果：现在英雄飞机无论怎么移动都不会超出有些窗口了

# 25.引爆所有敌机的需求说明p76

## 思路

|  |
| --- |
|  |

# 26.引爆所有敌机的代码实现p77

## 1.给英雄飞机条件一个blowup方法，实现炸掉屏幕说是使用敌机以及计算分数，判断是否升级得到

|  |
| --- |
|  |

## 2.然后在game.py里面的按键b处理方法中把以前的测试代码删除，添加英雄飞机炸掉全部敌机的方法调用并且把分数添加到HUDPanel的score属性中，而且需要判断是否升级如果升级，需要增加敌机数量

|  |
| --- |
|  |

## 测试一下，发现有点问题，在屏幕之外的敌机也给炸了，这个不符合英雄逻辑，我们修改一下英雄飞机的blowup方法，添加对敌机位置的判断，只有他在屏幕里面才能炸他

|  |
| --- |
|  |

## 效果

|  |
| --- |
|  |

# 这一节学习到此为止，这一节之后的所有模块的最新代码如下

## game.py

|  |
| --- |
| #游戏主程序  import pygame  import random  from game\_items import \*  from game\_hud import \*  from game\_music import \*  class Game(object):      def \_\_init\_\_(self) -> None:         #1.游戏主窗口         self.main\_window = pygame.display.set\_mode(SCREEN\_RECT.size)         #设置窗口标题         pygame.display.set\_caption('飞机大战')         #2.游戏状态属性         self.is\_game\_over = False #游戏结束标记         self.is\_pause = False    #游戏暂停标记         #3.创建精灵组，3个精灵组：all\_group,enemies\_group,supplies\_group         self.all\_group = pygame.sprite.Group() # 存放所有精灵的精灵组         self.enemies\_group = pygame.sprite.Group() # 敌人精灵组         self.supplies\_group = pygame.sprite.Group() # 补给精灵组或者道具精灵组           #4.创建精灵         #创建背景精灵，需要2个精灵否则无法实现连续滚动效果，默认向下移动,我们将他们添加到all\_group中         #写法1      #    BackgroundSprite(False,self.all\_group) #默认在窗口里面显示      #    BackgroundSprite(True,self.all\_group)  #默认在窗口顶部外面显示         #写法2         self.all\_group.add( BackgroundSprite(False),BackgroundSprite(True))           #创建游戏控制面板         self.hud\_panel = HUDPanel(self.all\_group)         #创建英雄精灵,有速度的         self.hero = Hero(self.all\_group)         #在创建英雄飞机后修改炸弹数量、         self.hud\_panel.show\_bomb(self.hero.bomb\_count)           # 初始化敌机         self.creat\_enemies()         #5.音乐播放器        def reset\_game(self):        """重置游戏"""        self.is\_game\_over = False #游戏结束标记        self.is\_pause = False    #游戏暂停标记        self.hud\_panel.reset\_panel() #重置显示面板      def creat\_enemies(self):          """创建敌机"""          # 先获取敌人精灵组的精灵数量          count = len(self.enemies\_group.sprites())          # 再定义精灵需要添加到的精灵组，有all\_group和enemies\_group          groups = (self.all\_group,self.enemies\_group)          #根据游戏级别和敌机数量来创建敌机          if self.hud\_panel.level ==1 and count == 0: # 关卡1              for i in range(16):                  Enemy(0,3,\*groups)          elif self.hud\_panel.level ==2 and count == 16: # 关卡2              # 1.先给敌机加速              for enemy in self.enemies\_group.sprites():                  enemy.max\_speed = 5  # 此时只有一种敌机              # 2.创建敌机              for i in range(8):                  Enemy(0,5,\*groups) # 创建8架小飞机              for i in range(2):                  Enemy(1,1,\*groups) # 创建2架中飞机          elif self.hud\_panel.level ==3 and count == 26: # 关卡3              # 1.先给敌机加速              for enemy in self.enemies\_group.sprites():                  enemy.max\_speed = 7 if enemy.kind == 0 else 3 #此时有两种敌机              # 2.创建敌机              for i in range(8):                  Enemy(0,5,\*groups) # 创建8架小飞机              for i in range(2):                  Enemy(1,1,\*groups) # 创建2架中飞机              for i in range(2):                  Enemy(2,1,\*groups) # 创建2架大飞机      def creat\_supplies(self):          """创建道具"""          pass      def start(self):          """开始游戏"""          #创建游戏时钟          clock = pygame.time.Clock()          # 定义应该逐帧动画计算器，用来控制逐帧动画的播放速度          frame\_counter = 0          #游戏主循环          while True:              # 游戏循环的第一件事就是判断游戏是否结束，当玩家的飞机数量是0，就可以              self.is\_game\_over = self.hud\_panel.lives\_count == 0              #监听事件，调用event\_handler方法              if self.event\_handler(): #返回True说明用户点击了关闭按钮                  self.hud\_panel.save\_best\_score() #游戏介绍之前需要把最好成绩保存到record.txt文件中                  return              # 根据游戏状态切换界面显示的内容              if self.is\_game\_over:                  self.hud\_panel.panel\_pause(True,self.all\_group) # Game over了，游戏需要停止注意第一个参数和暂停不一样              elif self.is\_pause:                  self.hud\_panel.panel\_pause(False,self.all\_group) #如果暂停标志位真，就暂停游戏              else:                  self.hud\_panel.panel\_resume(self.all\_group) # 从暂停状态恢复                  # 获取当前时刻的按键元组                  keys = pygame.key.get\_pressed()                    # 我们利用一个比较简便的方法来判断水平移动的方向，                  # 就是用keys[pygame.K\_RIGHT]-keys[pygame.K\_LEFT]                  move\_hor = keys[pygame.K\_RIGHT]-keys[pygame.K\_LEFT]                  # 同理判断垂直移动的方向，就是用keys[pygame. pygame.K\_DOWN]-keys[ pygame.K\_UP]                  move\_ver = keys[pygame.K\_DOWN]-keys[ pygame.K\_UP]                  # 控制逐帧动画播放速度                  frame\_counter = (frame\_counter + 1) % FRAME\_INTERVAL                  self.all\_group.update(frame\_counter == 0,move\_hor,move\_ver)  # 这里也需要传递移动方向              #绘制所有精灵              self.all\_group.draw(self.main\_window)              #刷新界面              pygame.display.update()              #设置游戏的刷新频率              clock.tick(60)      def event\_handler(self):          """事件监听"""          for event in pygame.event.get():              if event.type == pygame.QUIT: #处理点击窗口的关闭按钮的退出事件                  return True              elif event.type == pygame.KEYDOWN and event.key==pygame.K\_ESCAPE: # 按esc键程序也会退出                  return True              elif event.type == pygame.KEYDOWN and event.key==pygame.K\_SPACE: # 按空格键                  #如果游戏已经结束                  if self.is\_game\_over:                      self.reset\_game() #重置游戏                  else:                      self.is\_pause = not self.is\_pause # 如果没有结束，就可以进行暂停和继续的切换              # 当游戏没有暂停也没有结束，用户按下b键才有效，否则没有效果              # 引爆炸弹监听事件，用户按下b键会引爆一个炸弹把所有敌机都炸掉，然后炸弹数量要减少1              if not self.is\_game\_over and not self.is\_pause:                  if event.type == pygame.KEYDOWN and event.key==pygame.K\_b:                     score = self.hero.blowup(self.enemies\_group) # 炸弹屏幕上所有敌人并且获取总分                     # 更新面板的炸掉显示数量                     self.hud\_panel.show\_bomb(self.hero.bomb\_count)                     # 加分并且检查是否升级                     if self.hud\_panel.increase\_score(score):                         # 如果升级，需要创建更多敌机                         self.creat\_enemies()            return False #其他情况都不会退出程序      def check\_collide(self):          """碰撞检测"""          pass    if \_\_name\_\_ == '\_\_main\_\_':      #初始化游戏      pygame.init()      #开始游戏      Game().start()      #释放游戏资源      pygame.quit() |

## game\_item.py

|  |
| --- |
| #封装游戏元素对应的类  from typing import Any  import pygame  import random    # 定义全局常量  #1.游戏主窗口区域  SCREEN\_RECT = pygame.Rect(0,0,480,700) #游戏屏幕矩形  FRAME\_INTERVAL = 10 # 逐帧动画间隔帧数  HERO\_BOMB\_COUNT = 3 # 英雄飞机的炸弹数量  HERO\_DEFAULT\_MID\_BUTTOM = (SCREEN\_RECT.centerx,SCREEN\_RECT.bottom - 90) #英雄默认初始位置  class GameSprite(pygame.sprite.Sprite):      res\_path = './res/images/'      def \_\_init\_\_(self,image\_name,speed, \*groups) -> None:          super().\_\_init\_\_(\*groups)          self.image = pygame.image.load(self.res\_path + image\_name) #加载图片          self.rect = self.image.get\_rect() # 获取图片的矩形区域并且保存起来          self.speed = speed # 接收传递进来的速度值      def update(self, \*args) -> None:          # super().update(\*args)          """默认在垂直方向移动"""          self.rect.y += self.speed  class BackgroundSprite(GameSprite):      # 需要在初始化的时候给他传递应该状态,is\_alt==True说明这个背景精灵需要显示在窗口的顶部      def \_\_init\_\_(self, is\_alt ,\*groups) -> None:          super().\_\_init\_\_('background.png', 1, \*groups)          if is\_alt:              self.rect.y = -self.rect.h      def update(self, \*args) -> None:          super().update(\*args)          # 图片滚出边界的处理：如果图片已经滚动到了窗口的底部，          # 立即让他回到窗口的最上面，然后重新往下滚动          if self.rect.y > self.rect.h:              self.rect.y = -self.rect.h  class StatusButton(GameSprite):      def \_\_init\_\_(self, image\_names, \*groups) -> None:          """image\_names是一个元组，元组的0下标必须是一张暂停的图片，元组的1下标必须是一张播放的图片"""          super().\_\_init\_\_(image\_names[0],0,\*groups)          # 准备2张图片供切换          self.images = [pygame.image.load(self.res\_path+name) for name in image\_names]      def switch\_status(self,is\_pause):          """切换状态的方法"""          #根据状态切换图片          self.image =self.images[1 if is\_pause  else 0] # 仿三目运算符  #标签精灵类  class Label(pygame.sprite.Sprite):      font\_path = './res/font/MarkerFelt.ttc' # 字体文件的存放路径      def \_\_init\_\_(self,text,size,color, \*groups) -> None:          super().\_\_init\_\_(\*groups)          # 创建字体，使用我们提供的字体文件          self.font = pygame.font.Font(self.font\_path,size)          self.color = color # 文字颜色          # 精灵属性          self.image = self.font.render(text,True,self.color)          self.rect = self.image.get\_rect()      def set\_text(self,text):          """设置字体方法"""          self.image = self.font.render(text,True,self.color)          self.rect = self.image.get\_rect()  # 飞机类  class Plane(GameSprite):      def \_\_init\_\_(self, hp,speed,value,wav\_name,normal\_names,hurt\_name,destroy\_names, \*groups) -> None:          """初始化方法          hp:            血量          speed:         飞行速度          value:         分值，也就是摧毁一个得到多少分          wav\_name:      对应的音效文件          normal\_names:  正常状态下的逐帧动画图片列表          hurt\_name:     受伤图片，此时飞机并没有被摧毁          destroy\_names: 飞机爆炸效果逐帧动画的图片列表          \*groups:       需要添加到的精灵组          """          super().\_\_init\_\_(normal\_names[0], speed, \*groups)          self.hp = hp          self.max\_hp = hp          self.value = value          self.wav\_name = wav\_name          # 正常状态图像列表          self.normal\_images = [pygame.image.load(self.res\_path+name) for name in normal\_names] #列表推导式          # 正常状态图像索引          self.normal\_index  = 0          # 加载受伤图片          self.hurt\_image = pygame.image.load(self.res\_path+hurt\_name)          # 加载摧毁图片列表          self.destroy\_images = [pygame.image.load(self.res\_path+name) for name in destroy\_names]          # 摧毁图片索引          self.destroy\_index = 0      # 重置飞机的方法      def reset\_plane(self):          #1.利用max\_hp重置hp          self.hp = self.max\_hp          #2.把normal\_index归零]          self.normal\_index = 0          #3.把destroy\_index归零          self.destroy\_index = 0          #4.把图片改为正常图片的第一张          self.image = self.normal\_images[0]      # update方法      def update(self,\*args):          """更新状态，准备下一次需要显示的内容"""          # 判断是否需要更新，只需要判断args[0]是否位True，注意\*args是可变长参数          if not args[0]: # 如果为False就不更新，直接return              return          #True就更新          #根据血量hp来切换需要显示的图片，有几种情况          if self.hp == self.max\_hp: # 说明没有受伤，显示正常图片即可              self.image = self.normal\_images[self.normal\_index]              # 计算下一张图片索引，注意下标不能越界              count = len(self.normal\_images)              self.normal\_index = (self.normal\_index + 1)%count # 利用取余数的方法来处理越界问题          elif self.hp >0 : # 受伤了，但是还没有被摧毁              self.image = self.hurt\_image #显示为受伤状态          else: #进入摧毁中的状态，播放摧毁动画              #判断是否播放到了最后一张图片，如果是就是完全被摧毁了              if self.destroy\_index < len(self.destroy\_images):                  self.image = self.destroy\_images[self.destroy\_index]                  self.destroy\_index += 1                else: #完全摧毁，需要重置飞机                  self.reset\_plane()  # 敌机类  class Enemy(Plane):      def \_\_init\_\_(self, kind,max\_speed, \*groups) -> None:          self.kind = kind          self.max\_speed = max\_speed          if kind == 0: # 小敌机              super().\_\_init\_\_(1, max\_speed, 1000, 'enemy1\_down.wav',                               ["enemy1.png"], "enemy1.png",                               ['enemy1\_down%d.png'% i for i in range(1,5)],                               \*groups)          elif kind == 1: # 中敌机              super().\_\_init\_\_(6, max\_speed, 6000, 'enemy2\_down.wav',                               ["enemy2.png"], "enemy2\_hit.png",                               ['enemy2\_down%d.png'% i for i in range(1,5)],                               \*groups)          elif kind == 2: # 大敌机              super().\_\_init\_\_(15, 1, 15000, 'enemy3\_down.wav',                               ["enemy3\_n1.png","enemy3\_n2.png"], "enemy3\_hit.png",                               ['enemy3\_down%d.png'% i for i in range(1,7)],                               \*groups)            # 随机设置飞机位置，要不然飞机就重叠在一起了          self.reset\_plane()        def reset\_plane(self):         super().reset\_plane() #先调用父类的重置方法         """敌机的数据重置代码"""         x = random.randint(0,SCREEN\_RECT.w - self.rect.w)         y = random.randint(0,SCREEN\_RECT.h - self.rect.h) - SCREEN\_RECT.h # 设置到屏幕外面         self.rect.topleft = (x,y)         #设置速度         self.speed = random.randint(1,self.max\_speed)      def update(self, \*args): #其实就是在父类的基础上添加飞机出界处理          super().update(\*args)   # 调用父类的update方法          #判断是否被摧毁，如果没有就继续更新位置          if self.hp > 0:              self.rect.y += self.speed          if self.rect.y >= SCREEN\_RECT.h: #出界了，就要重新设置位置              self.reset\_plane()  class Hero(Plane):      def \_\_init\_\_(self, \*groups) -> None:          self.is\_power = False                     # 是否无敌          self.bomb\_count = HERO\_BOMB\_COUNT         # 英雄飞机的炸弹数量          self.bullet\_kind = 0                      # 子弹类型          self.bullets\_group = pygame.sprite.Group() # 子弹精灵组          # 调用父类的初始化方法          super().\_\_init\_\_(1000, 5, 0, 'me\_down.wav',['me1.png','me2.png'],'me1.png',                      ['me\_destroy\_%d.png' % i for i in range(1,5)], \*groups)          # 设置英雄飞机默认位置          self.rect.midbottom = HERO\_DEFAULT\_MID\_BUTTOM      def update(self, \*args):          """          在这里args[0]是一个标记表明是否需要更新帧动画,          args[1]就是水平移动的方向基数，          args[2]就是垂直移动方向基数          """          super().update(\*args)          # 为了保证代码的正确执行，需要两个条件：首先传递进来的args的长度必须是3，其次英雄飞机的血量必须大于0          if len(args) != 3 or self.hp <= 0: #不符合条件就返回              return          # 根据args[1]来设置英雄飞机的水平移动方向          self.rect.x += args[1] \* self.speed          self.rect.x = 0 if self.rect.x < 0 else self.rect.x #左边出界处理          if self.rect.right > SCREEN\_RECT.right:               self.rect.right = SCREEN\_RECT.right # 右边出界处理          # 根据args[2]来设置英雄飞机的垂直移动方向          self.rect.y += args[2] \* self.speed          self.rect.y = 0 if self.rect.y < 0 else self.rect.y # 上边出界处理          if self.rect.bottom > SCREEN\_RECT.bottom:               self.rect.bottom = SCREEN\_RECT.bottom # 下边出界处理      def blowup(self,enemies\_group):          # 先判断有没有炸弹          if self.bomb\_count <= 0 or self.hp <=0:              return 0 # 没有炸弹， 炸不了,返回0分            self.bomb\_count -= 1  # 每炸一次，炸弹数需要减少1          score = 0          count = 0          # 炸掉使用敌机并且获取分数          for enemy in enemies\_group.sprites():              # 在炸飞机之前需要先判断它是否在屏幕里面,如果是才能炸              if enemy.rect.bottom > 0:                  score += enemy.value # 累计得分                  enemy.hp = 0 # 搞死敌人                  count += 1 # 计算器累计            print('炸毁了 %d 架敌机' % count)          # 返回获取的累计分数          return score |

## game\_hud.py

|  |
| --- |
| #游戏面板模块  import pygame  from game\_items import \*  #面板类  class HUDPanel(object):      margin =10      white =(255,255,255)      gray = (64,64,64)      reward\_score = 100000  #每增加100000分奖励一架飞机      level2\_score = 10000     #达到10000分就进入第二关      level3\_score = 50000     #达到50000分就进入第三关      record\_filename = 'record.txt' # 保存最高分数的文件的路径      def \_\_init\_\_(self,display\_group) -> None:          """          面板初始化方法：          display\_group是面板中的精灵要被添加到的精灵组          """          self.score = 0          self.lives\_count = 3          self.level = 1          self.best\_score = 0          # 加载上一次游戏的最好成绩          self.load\_best\_score()            # 图像精灵          #1.状态精灵          self.status\_sprite = StatusButton(('pause\_nor.png','resume\_nor.png'),display\_group)          self.status\_sprite.rect.topleft = (self.margin,self.margin)          #2.炸弹精灵          self.bomb\_sprite = GameSprite('bomb.png',0,display\_group)          # 调整bomb\_sprite的位置          self.bomb\_sprite.rect.x = self.margin          self.bomb\_sprite.rect.bottom = SCREEN\_RECT.bottom - self.margin          #3.生命计数精灵          self.lives\_sprite = GameSprite('life.png',0,display\_group)          # szlives\_sprite的位置          self.lives\_sprite.rect.right = SCREEN\_RECT.right-self.margin          self.lives\_sprite.rect.bottom = SCREEN\_RECT.bottom - self.margin          #4.创建标签精灵          # 4.1数字标签          # 4.1.1 分数标签          self.score\_label = Label("%d" % self.score,32,self.gray,display\_group)          self.score\_label.rect.midleft = (self.status\_sprite.rect.right + self.margin,                                             self.status\_sprite.rect.centery)            # 4.1.2 炸弹计数标签          self.bomb\_label = Label("X 3",32,self.gray,display\_group )          self.bomb\_label.rect.midleft = (self.bomb\_sprite.rect.right+self.margin,                                             self.bomb\_sprite.rect.centery)          # 4.1.3 生命值标签,这个标签的位置笔记特别，它会占用lives\_sprite的位置，          # 然后lives\_sprite的位置需要使用它的位置来重新计算          self.lives\_label = Label("X %d" % self.lives\_count,32,self.gray,display\_group)          self.lives\_label.rect.midright = (SCREEN\_RECT.right - self.margin,                                            self.bomb\_label.rect.centery)            #修改 lives\_sprite的位置          self.lives\_sprite.rect.right = self.lives\_label.rect.left - self.margin          #注意：在游戏正常运行时，最好成绩标签，状态标签和  提示标签是隐藏的，游戏状态，他们才显示          # 4.1.4 最好成绩标签          self.best\_label = Label("Best: %d" % self.best\_score,36,self.white)            # 4.1.5 状态标签          self.status\_label = Label("Game Playing",48,self.white)            # 4.1.6 提示标签          self.tip\_label = Label("Press SpaceBar To Pause Game",22,self.white)          # 显示炸弹数量的方法      def show\_bomb(self,count):          # 修改文字          self.bomb\_label.set\_text("X %d" % count)          # 修改位置          self.bomb\_label.rect.midleft = (self.bomb\_sprite.rect.right + self.margin,                                          self.bomb\_sprite.rect.centery)        def show\_lives(self):          self.lives\_label.set\_text("X %d" % self.lives\_count)          #调整 lives\_label的位置          self.lives\_label.rect.midright = (SCREEN\_RECT.right - self.margin,                                            self.bomb\_label.rect.centery)            #调整 lives\_sprite的位置          self.lives\_sprite.rect.right = self.lives\_label.rect.left - self.margin      # 定义一个增加分数的方法      def increase\_score(self,enemy\_score):          #1.先用一个变量来累计每一次的得分          score = self.score + enemy\_score # 这个分数需要一直增加          #2.每累计一次，就需要判断是否奖励一条命，是很有技巧的,其实还有别的方法，如：只要这个score能被100000整除就可以了          if score // self.reward\_score !=self.score // self.reward\_score:             self.lives\_count +=1 #奖励一条命，也就是多一架飞机             self.show\_lives() # 获取到奖励后需要更新生命值          # 更新自己的分数          self.score = score          #3.每累计一次，都要和上一次的最好成绩比较,只要这个成绩比原来的最好成绩大，就把它设置为最好成绩，否则不设置          self.best\_score = score if score > self.best\_score else self.best\_score          #4.每累计一次，都要判断是否升级          if  score < self.level2\_score:              level = 1          elif score < self.level3\_score:              level = 2          else:              level = 3          is\_upgrade = level !=self.level          self.level = level          # 5.每累计一次，都需要修改分数标签的内容和位置          self.score\_label.set\_text("%d" % self.score)          self.score\_label.rect.midleft = (self.status\_sprite.rect.right + self.margin,                                             self.status\_sprite.rect.centery)          return is\_upgrade        # 保存最好成绩的方法      def save\_best\_score(self):          with open(self.record\_filename,'w') as f:              f.write('%d' % self.best\_score)      def load\_best\_score(self):           try:              with open(self.record\_filename,'r') as f:                  self.best\_score = int(f.read())                  print(f'上一次最高分数是{self.best\_score}')           except (FileNotFoundError,ValueError):               print('读取record.txt文件出错,文件不存在或者内容为空...')        # 暂停游戏的方法,必须在游戏没有介绍的前提下才有效      def panel\_pause(self,is\_game\_over,display\_group):          #1.首先需要判断是否把最好成绩标签，状态标签和  提示标签添加到了精灵组，如果是，直接返回          if display\_group.has(self.best\_label,self.status\_label,self.tip\_label):              return          # 2.根据游戏是否介绍来调整标签的文字          text = "Game Over!" if is\_game\_over else "Game Paused!"          tip = 'Press SpaceBar To'          tip += ' Play Again' if is\_game\_over else " Continue"          # 3.给标签设置对应的文字          self.status\_label.set\_text(text)          self.best\_label.set\_text("Best Score: %d" % self.best\_score)          self.tip\_label.set\_text(tip)          # 4.调整标签位置          self.best\_label.rect.center = SCREEN\_RECT.center          self.status\_label.rect.midbottom = (self.best\_label.rect.centerx,                                              self.best\_label.rect.y - 2 \* self.margin)            self.tip\_label.rect.midtop = (self.best\_label.rect.centerx,                                      self.best\_label.rect.bottom + 8 \* self.margin)            # 5.添加标签到精灵组          display\_group.add(self.best\_label,self.status\_label,self.tip\_label)          # 6.状态精灵切换状态          self.status\_sprite.switch\_status(True)      # 从暂停状态恢复      def panel\_resume(self,display\_group):          # 1.隐藏 最好成绩标签，状态标签和  提示标签添只需要把他们从精灵组里面删除即可          display\_group.remove(self.best\_label,self.status\_label,self.tip\_label)          # 2.需要切换状态精灵的状态          self.status\_sprite.switch\_status(False)      # 重置面板内容的功能      def reset\_panel(self):          #重置数据          self.score = 0          self.lives\_count = 3          #重置精灵数据          self.increase\_score(0) # 可以在这个方法里面计算level，设置最好成绩标签，状态标签和  提示标签添等等          self.show\_lives() #重置生命值          self.show\_bomb(3) #重置炸弹数量 |

## game\_music.py 暂时为空

# 下一节我们来学习碰撞检测