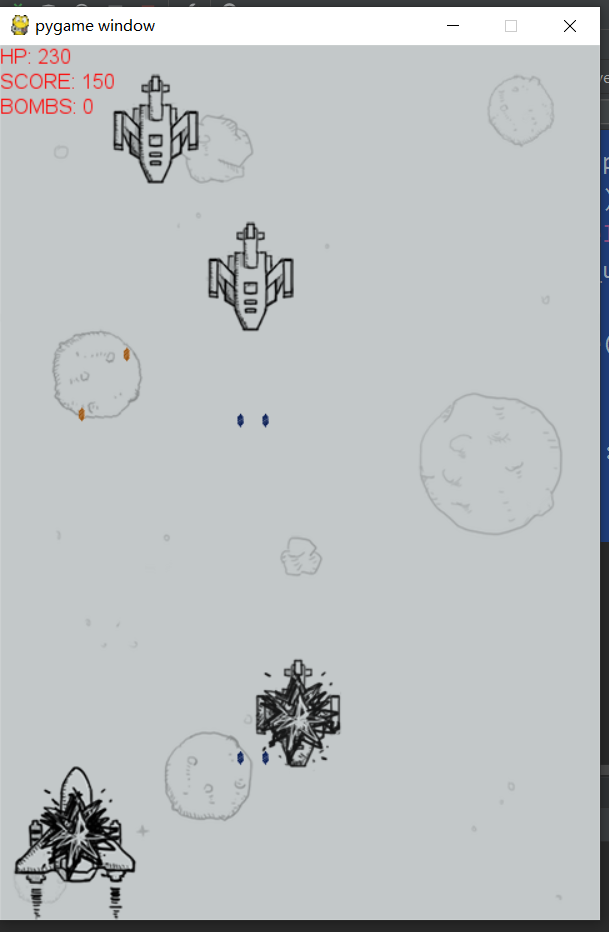
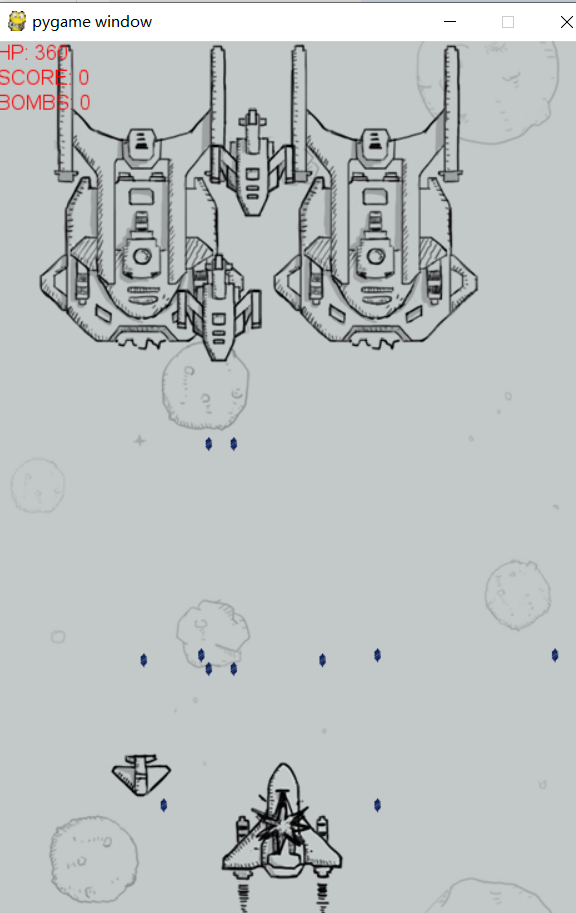
1. 完成小飞机游戏---功能和发给大家的代码保持一致即可

*'''  
飞机上下左右可移动 空格开火 开火间隙合理  
积分每达到一定值，可以使用伤害较高的炸弹，按b释放  
  
敌机随机生成 每种敌机速度，子弹种类不同 共有  
 1.普通弹 2.并排弹 3.散射弹  
  
采用hp生命值机制，非一触即死，hp剩余值不同对应显示不同状态的图片,敌机、英雄生效  
  
敌机、英雄死亡有爆炸动画，动画有时间停留 ，其中  
当英雄死亡时，游戏结束，屏幕出现相应提示，玩家不可操作，画面0.5倍速播放  
  
增加游戏音乐（Double Dragon.mp3）、击中伤害显示、游戏得分、hp显示  
  
'''*import random as r  
import pygame as pg  
from pygame.sprite import Sprite, Group, groupcollide, spritecollide  
  
SCREEN\_SIZE = (480, 700)  
FRAME\_RATES = 60  
CREATE\_ENEMY\_EVENT = pg.USEREVENT  
CLEAR\_DAMAGE\_LIST = pg.USEREVENT + 1  
  
  
def py\_environment(func):  
 def inner(\*args, \*\*kwargs):  
 pg.init()  
 pg.mixer.init()  
 pg.font.init()  
 res = func(\*args, \*\*kwargs)  
 pg.quit()  
 return res  
  
 return inner  
  
  
class GameSprite(Sprite):  
 ImagePath = 'images/'  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.image = None  
 self.pngs = None  
 self.hp = None  
 self.hpFlash = None  
 self.hpCheckPoints = None  
 self.live = None  
 self.deathFlash = None  
 self.flashInterval = FRAME\_RATES // 2  
 self.speed = None  
 self.rect = None  
  
 def get\_image(self, imageName, rect=True):  
  
 image = pg.image.load(GameSprite.ImagePath + imageName)  
 if rect: self.rect = image.get\_rect()  
  
 return image  
  
 def get\_pngs(self, name, downs, downsName=None, setFlash=True):  
 *'''* ***:param*** *name:* ***:param*** *downs: (x,y) all down pngs,y the first bomb png* ***:return****:  
 '''* pngs = [[], []]  
 pngs[0].append(name + '.png')  
 x, y = downs  
 if not downsName: downsName = name + '\_down'  
  
 for i in range(1, y):  
 pngs[0].append(downsName + str(i) + '.png')  
 for i in range(y, x + 1):  
 pngs[1].append(downsName + str(i) + '.png')  
  
 if setFlash:  
 self.deathFlash = pngs[1]  
 self.hpFlash = pngs[0]  
  
 return pngs  
  
 def set\_hp(self, hp):  
  
 if (not self.hpCheckPoints) and self.hpFlash:  
 slices = hp // len(self.hpFlash)  
 self.hpCheckPoints = []  
 for i in range(1, len(self.hpFlash)):  
 self.hpCheckPoints.append((i \* slices, self.hpFlash[-i]))  
 return hp  
  
 def get\_damaged(self, damage=0):  
 if self.hp - damage > 0:  
 self.hp -= damage  
 MainGame.damageFontList.append((-damage, (self.rect.right, self.rect.centery)))  
 for hp, image in self.hpCheckPoints:  
 if self.hp <= hp:  
 self.image = self.get\_image(image, rect=False)  
 break  
 else:  
 self.hp = 0  
 self.kill()  
 MainGame.flashGroup.add(Flash(self.rect.x, self.rect.y,  
 self.deathFlash, self.flashInterval))  
 self.live = False  
  
  
class Flash(GameSprite):  
 def \_\_init\_\_(self, x, y, flashes, interval=FRAME\_RATES // 2):  
 super().\_\_init\_\_()  
 self.frame = 0  
 self.time = 0  
 self.interval = interval  
 self.flashes = flashes  
 self.image = self.get\_image(self.flashes[self.frame])  
 self.rect.x, self.rect.y = x, y  
  
 def update(self, \*args):  
 if self.frame >= len(self.flashes):  
 self.kill()  
 elif self.time % self.interval == 0:  
 self.image = self.get\_image(self.flashes[self.frame], rect=False)  
 self.frame += 1  
 self.time += 1  
  
  
class Hero(GameSprite):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.pngs = self.get\_pngs(name='me1', downs=(4, 3), downsName='me\_destroy\_')  
 self.image = self.get\_image(self.pngs[0][0])  
 self.rect.x, self.rect.bottom = SCREEN\_SIZE[0] // 2, SCREEN\_SIZE[1]  
 self.speed = (3,2)  
  
 self.hp = self.set\_hp(1000)  
 self.deathFlash = self.pngs[1]  
 self.flashInterval = FRAME\_RATES \* 2  
 self.live = True  
  
 self.fireTime = 0  
 self.fireInterval = FRAME\_RATES // 5 # 1秒发射子弹数目  
 self.bombs=0  
 self.score=0  
  
 def move(self, direction=None):  
 if self.hp <= 0: return  
  
 if direction == 'up': # 更新位置  
 self.rect.y -= self.speed[1]  
 elif direction == 'down':  
 self.rect.y += self.speed[1]  
 elif direction == 'left':  
 self.rect.x -= self.speed[0]  
 elif direction == 'right':  
 self.rect.x += self.speed[0]  
  
 if self.rect.top <= 0: # 位置修正  
 self.rect.top = 0  
 elif self.rect.bottom >= SCREEN\_SIZE[1]:  
 self.rect.bottom = SCREEN\_SIZE[1]  
 elif self.rect.left <= 0:  
 self.rect.left = 0  
 elif self.rect.right >= SCREEN\_SIZE[0]:  
 self.rect.right = SCREEN\_SIZE[0]  
  
 def fire(self):  
 if self.fireTime % self.fireInterval == 0:  
 MainGame.bullet\_group.add(Bullet(self.rect.centerx, self.rect.top))  
  
 def bomb(self):  
 if self.bombs>0:  
 self.bombs-=1  
 MainGame.bomb\_group.add(Bomb(self.rect.centerx,self.rect.top-1))  
  
  
 def update(self, \*args):  
 self.fireTime += 1  
 if MainGame.score-self.score>Bomb.score\_cost:  
 self.score=MainGame.score  
 self.bombs+=1  
  
  
class Bullet(GameSprite):  
 def \_\_init\_\_(self, x, y, speed=None, damage=20, isEnemy=False):  
 super().\_\_init\_\_()  
 self.image = self.get\_image('bullet1.png') if not isEnemy else self.get\_image('bullet2.png')  
 self.rect.x, self.rect.y = x, y  
 if speed:  
 self.speed = speed  
 else:  
 self.speed = (0, -4) if not isEnemy else (0, 4)  
 self.damage = damage  
  
 def update(self, \*args):  
 self.rect.x += self.speed[0]  
 self.rect.y += self.speed[1]  
 if self.rect.top <= 0 or \  
 self.rect.left > SCREEN\_SIZE[0] or self.rect.right < 0:  
 self.kill()  
  
  
class Bomb(GameSprite):  
 score\_cost=200  
 def \_\_init\_\_(self,x,y):  
 super().\_\_init\_\_()  
 self.image=self.get\_image('bomb.png')  
 self.rect.x,self.rect.y=x,y  
 self.speed=(0,-2)  
 self.damage=500  
  
 def move(self):  
 self.rect.y+=self.speed[1]  
 if self.rect.bottom<0:self.kill()  
  
 def update(self, \*args):  
 self.move()  
  
  
class enemy(GameSprite):  
 def \_\_init\_\_(self, kind=1):  
 super().\_\_init\_\_()  
 downs = (4, 3) if kind != 3 else (6, 5)  
 self.kind = kind  
 self.score = [50, 100, 200][kind - 1]  
  
 self.pngs = self.get\_pngs(name='enemy' + str(kind), downs=downs)  
 self.image = self.get\_image(self.pngs[0][0])  
 self.rect.x = r.randint(0, SCREEN\_SIZE[0] - self.rect.size[0])  
  
 self.hp = self.set\_hp([100, 300, 900][kind - 1])  
 self.live = True  
 self.speed = [2, 1, 0.5][kind - 1]  
 self.damage = 10 \* self.kind  
  
 self.time = 0  
 self.fireInterval = FRAME\_RATES // (2)  
  
 def get\_damaged(self, damage=0):  
 super().get\_damaged(damage)  
  
 if not self.live:  
 MainGame.score += self.score  
  
 def move(self):  
 if self.live == False: return  
 self.rect.y += self.speed  
 if self.rect.top >= SCREEN\_SIZE[1]:  
 self.kill()  
  
 def fire(self):  
 if self.time % self.fireInterval == 0 and r.random() > 0.7:  
 if self.kind == 1:  
 MainGame.enemy\_bullet\_group.add(Bullet(x=self.rect.centerx, y=self.rect.bottom, isEnemy=True))  
 elif self.kind == 2:  
 MainGame.enemy\_bullet\_group.add(Bullet(x=self.rect.centerx + 10, y=self.rect.bottom, isEnemy=True),  
 Bullet(x=self.rect.centerx - 10, y=self.rect.bottom, isEnemy=True))  
 else:  
  
 MainGame.enemy\_bullet\_group.add(  
 Bullet(speed=(-1, 4), x=self.rect.left, y=self.rect.bottom, isEnemy=True),  
 Bullet(speed=(0, 4), x=self.rect.centerx, y=self.rect.bottom, isEnemy=True),  
 Bullet(speed=(1, 4), x=self.rect.right, y=self.rect.bottom, isEnemy=True), )  
  
 def update(self, \*args):  
 self.time += 1  
 self.move()  
 self.fire()  
  
  
class BackGround(GameSprite):  
 def \_\_init\_\_(self, issecond=False):  
 super().\_\_init\_\_()  
 self.image = self.get\_image('background.png')  
 if issecond:  
 self.rect.y = -self.rect.size[1]  
 self.speed = 1  
  
 def update(self, \*args):  
 self.rect.y += self.speed  
 if self.rect.top == SCREEN\_SIZE[1]:  
 self.rect.top = -self.rect.size[1]  
  
  
class GameInfo(GameSprite):  
 def \_\_init\_\_(self, name):  
 super().\_\_init\_\_()  
 self.pngs = {'gameover': 'gameover.png', 'pause': 'pause\_nor.png'}  
 self.name = name  
 self.image = self.get\_image(self.pngs[name])  
 if name == 'gameover':  
 self.rect.centerx, self.rect.centery = SCREEN\_SIZE[0] // 2, SCREEN\_SIZE[1] // 2  
  
  
class MainGame:  
 heroGroup = Group()  
 enemy\_group = Group()  
 bullet\_group = Group()  
 bomb\_group=Group()  
 enemy\_bullet\_group = Group()  
 backGroundGroup = Group(BackGround(), BackGround(True))  
 flashGroup = Group()  
 InfoGroup = Group()  
  
 damageFontList = []  
 gameover = False  
 score = 0  
  
 def \_\_init\_\_(self):  
 self.window = pg.display.set\_mode(SCREEN\_SIZE)  
 self.clock = pg.time.Clock()  
 self.hero = Hero()  
 MainGame.heroGroup.add(self.hero)  
  
 self.Font = None  
 pg.time.set\_timer(CREATE\_ENEMY\_EVENT, 2000)  
 pg.time.set\_timer(CLEAR\_DAMAGE\_LIST, 200)  
  
 def music\_play(self):  
 pg.mixer\_music.load('Double Dragon.mp3')  
 pg.mixer\_music.play(-1)  
  
 @py\_environment  
 def start\_game(self):  
 self.music\_play()  
 while True:  
 self.clock.tick(FRAME\_RATES)  
 self.update()  
  
 def event\_handler(self):  
 for event in pg.event.get():  
 if event.type == pg.QUIT:  
 print('游戏退出')  
 exit()  
 elif event.type == CREATE\_ENEMY\_EVENT:  
 res = r.random()  
 if res <= 0.5:  
 MainGame.enemy\_group.add(enemy(1))  
 elif res <= 0.95:  
 MainGame.enemy\_group.add(enemy(2))  
 else:  
 MainGame.enemy\_group.add(enemy(3))  
 elif event.type == CLEAR\_DAMAGE\_LIST:  
 MainGame.damageFontList.clear()  
  
 if event.type == pg.KEYDOWN and event.key == pg.K\_b: self.hero.bomb()  
  
 keys\_pressed = pg.key.get\_pressed()  
  
  
 if keys\_pressed[pg.K\_RIGHT]:  
 self.hero.move('right')  
 elif keys\_pressed[pg.K\_LEFT]:  
 self.hero.move('left')  
 elif keys\_pressed[pg.K\_UP]:  
 self.hero.move('up')  
 elif keys\_pressed[pg.K\_DOWN]:  
 self.hero.move('down')  
  
 if keys\_pressed[pg.K\_SPACE]:  
 self.hero.fire()  
  
  
 def collide\_check(self):  
 enemy\_damage\_group=[ MainGame.bullet\_group,MainGame.bomb\_group]  
 for group in enemy\_damage\_group:  
 res = groupcollide(MainGame.enemy\_group, group, False, True)  
 for enemy,damages in res.items():  
 for damage in damages:  
 enemy.get\_damaged(damage.damage)  
  
  
 hero\_damage\_group=[MainGame.enemy\_group,MainGame.enemy\_bullet\_group]  
 for group in hero\_damage\_group:  
 res = groupcollide(group, MainGame.heroGroup, True, False)  
 for enemy in res: self.hero.get\_damaged(enemy.damage)  
  
  
 def game\_over(self):  
 global FRAME\_RATES  
 if len(MainGame.heroGroup) == 0 and not MainGame.gameover:  
 MainGame.InfoGroup.add(GameInfo('gameover'))  
 MainGame.gameover = True  
 FRAME\_RATES //= 2  
  
 def player\_info\_update(self):  
 font = pg.font.SysFont('arial', 16)  
 hpInfo = font.render('HP: {}'.format(self.hero.hp), True, [255, 0, 0])  
 scoreInfo = font.render('SCORE: {}'.format(MainGame.score), True, [255, 0, 0])  
 bombInfo=font.render('BOMBS: {}'.format(self.hero.bombs), True, [255, 0, 0])  
 self.window.blit(hpInfo, (0, 0))  
 self.window.blit(scoreInfo, (0, 20))  
 self.window.blit(bombInfo, (0, 40))  
  
 damageFont = pg.font.SysFont('arial', 20, True)  
 for damage, cordial in MainGame.damageFontList:  
 damageInfo = damageFont.render('hit:{}'.format(damage), True, [255, 0, 0])  
 self.window.blit(damageInfo, cordial)  
  
 def update(self):  
 self.event\_handler()  
 self.collide\_check()  
 self.game\_over()  
  
 groups = [MainGame.backGroundGroup, MainGame.heroGroup, MainGame.enemy\_group,  
 MainGame.bullet\_group, MainGame.enemy\_bullet\_group, MainGame.flashGroup,  
 MainGame.bomb\_group,MainGame.InfoGroup]  
  
 for group in groups:  
 group.update()  
 group.draw(self.window)  
 self.player\_info\_update()  
  
 pg.display.update()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 mainGame = MainGame()  
 mainGame.start\_game()

并排弹



散射弹



命中提示



炸弹

