pygame.init() #初始化pygame

screen.fill(color) #设置背景颜色

screen.blit(a,(x,y)) #将图片a放置在坐标x，y处

screen=pygame.display.set\_mode((500,500),FULLSCREEN,32) #全屏

screen=pygame.display.set\_mode((500,500),RESIZABLE,32) #可切换分辨率

screen=pygame.display.set\_mode((500,500),0,32) #窗口化，0与32皆可不要

#可利用拖动改变窗口大小，需要设置RESIZABLE

#利用上述语句生成屏幕对象，以后在屏幕对象上作图

pygame.display.set\_caption("Sugar") #设定运行程序的程序名

pygame.display.update() #使图像显示出来，无此项则完全黑屏

mycolor=pygame.Color(0,255,0,128)

#以RGB参数设置颜色，第四个参数为透明度，0是全透明，255为完全不透明。默认255

for event in pygame.event.get():

if event.type==QUIT:

exit()

#在程序中必须有类似语句才能保证不会因循环过多而导致崩溃

pygame.draw.circle(surface,color ,center ,radius,width)

在程序中画一个圆，需要五组参数，其中最后一个参数可以不写，默认完全填充，0无效

pygame.draw.circle(screen,(0,0,255,128),(500,500),200 ,100)

pygame.draw.line(surface,color,start,end,width)

#在start和end间绘制一条直线,width默认为1

pygame.draw.line(surface,color,(146,0),(291,106),1)

pygame.draw.lines(surface,color,closed,pointlist,width)

#依次连接各个点，若closed参数为真，则第一个和最后一个也会连，反之不连

pygame.draw.lines(surface,color,0,((100,0),(100,100),(200,0)),1)

pygame.draw.polygon(screen,color,tuple,width)

在程序中画多边形，按照给出点的顺序依次连接，线条宽度参数可不写

pygame.draw.polygon(screen,(0,0,255),((146,0),(291,106),(236,277)),10)

pygame.draw.ellipse(surface,color,position,width)

#绘制椭圆，position指定了围绕椭圆形能绘制的最小的矩形。

pygame.draw.ellipse(surface,color,(0,0,200,500),width)

pygame.draw.rect(surface,color,position,width)

#绘制矩形，position是一个Rect对象或矩形左上x，y坐标，右下x，y坐标。

pygame.draw.rect(surface,color,(0,0,200,500),width)

myrect=pygame.Rect(50,60,200,300)

#矩形区域，前两个参数为矩形左上角x，y坐标，后两个为宽和高

a.colliderect(b)

#碰撞检测，a与b为两个Rect对象，若有碰撞则输出1，否则为0

soundplay=pygame.mixer.Sound('./aaaa.wav')

soundplay.play()

time.sleep(3)

soundplay.stop()

#添加音乐，似乎只能打开wav格式；mp3无法打开。

pixel=pygame.PixelArray(screen) #利用像素点上色，但会锁定surface对象

pixel[200][300]=(0,0,0)

del pixel #删除像素对象之后自动解锁surface对象，即不可用blit

#像素上色无法再程序范围以外进行，否则会报错

pygame.time.Clock().tick(30) #设置FPS，设定过大无效果，放在update之后

a=pygame.image.load(r'C:\Users\Username\Desktop\plane.png')

#将某张图片赋予变量a，用于日后移动、显示等操作

x, y = pygame.mouse.get\_pos() #得到鼠标的位置

mouse=pygame.image.load("hero.png").convert\_alpha() #可保留透明通道

.PNG 格式能保留透明通道，.JPG格式不可以

#对于event.type中的KEYUP，也可以设置不同的键位从而达到相应操纵效果，例如改善飞机游戏中的操纵性，例如：

elif event.type == KEYUP:

if event.key == K\_LEFT or event.key == K\_RIGHT:

move\_x = 0

字体相关

#显示中文必须在文件头部加上如下注释，并且所选则字体要支持中文，否则会出现乱码

# -\*- coding: utf-8 -\*-

a=pygame.font.get\_fonts()

#可得到系统可用的所有字体，返回list参数

a=pygame.font.Font('freesansbold.ttf',80) #设置字体，大小

b=a.render('Hello World!',1,Blue,White)

#设置显示内容，是否抗锯齿，字体颜色，背景颜色(可不写),

#为surface对象，用blit显示

a=pygame.font.Font('freesansbold.ttf',80).render('HelloWorld!',1,Green)

#可用以上语句代表简写

TitleFro=pygame.font.SysFont('默陌信笺手写体',50)

#可以使用系统自带字体，字体名后不需要添加.ttf后缀。一般系统都有arial字体。

#若找不到则默认使用第一个字体

color\_table.set\_at((x,y), (g, r, b))

#可以对surface对象某一像素单独修改颜色，如上所示。

#可以利用clock对象来实现对时间间隔的测量，如下：

a=pygame.time.Clock() #设置一个clock对象

time.sleep(2) #进行任意代码运行

s=a.tick() #s值即为上次tick到这次的时间间隔，单位ms，为int型

#不同的计时器之间分开计算各自的时间间隔

#利用这个函数可以使图像不管帧率多少都能在一定时间内移动相同的距离，示例如下：

s=clock.tick()

x+=velocity\*s/1000 #此处s单位为ms所以要除以1000

顺畅移动相关

#新的键盘按键检测和流畅运动可以参考以下代码：此代码仅做到

move=[0,0]

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

    if pressed\_keys[K\_RIGHT]:

        move[0]=-1

    if pressed\_keys[K\_LEFT]:

        move[0]=+1

    if pressed\_keys[K\_UP]:

        move[1]=1

    if pressed\_keys[K\_DOWN]:

        move[1]=-1

#顺畅移动的本质是用状态量去控制，比如：

elif event.type==pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

self.ship.moving\_left=True

self.ship.moving\_right=False

if event.key == pygame.K\_RIGHT:

  self.ship.moving\_right=True

    self.ship.moving\_left=False

if event.key == pygame.K\_UP:

self.ship.moving\_up=True

self.ship.moving\_down=False

if event.key == pygame.K\_DOWN:

      self.ship.moving\_down=True

   self.ship.moving\_up=False

elif event.type==pygame.KEYUP:

if event.key == pygame.K\_LEFT:

self.ship.moving\_left=False

   if event.key == pygame.K\_RIGHT:

     self.ship.moving\_right=False

   if event.key == pygame.K\_UP:

       self.ship.moving\_up=False

   if event.key == pygame.K\_DOWN:

      self.ship.moving\_down=False