COMP小组作业，代码编写部分思路于总结  
  
大环境配置：

游戏使用Python中的pygame库制作。导入Pygame库的操作：在命令提示符中使用pip install pygam即可。

游戏设定：

**星露谷物语联机版。代码部分是前端设计，后端设计负责联网部分，即玩家可以通过联网，此外还需要**

代码简介：

以下是从头开始编写代码时的思路。参考www.bilibili.com/video/BV1ia411d7yW的星露谷物语游戏制作视频（最后会有总结）

1. **设置**

**Group**是pygame中的关键特性，帮助我们draw and update任何类型的sprite

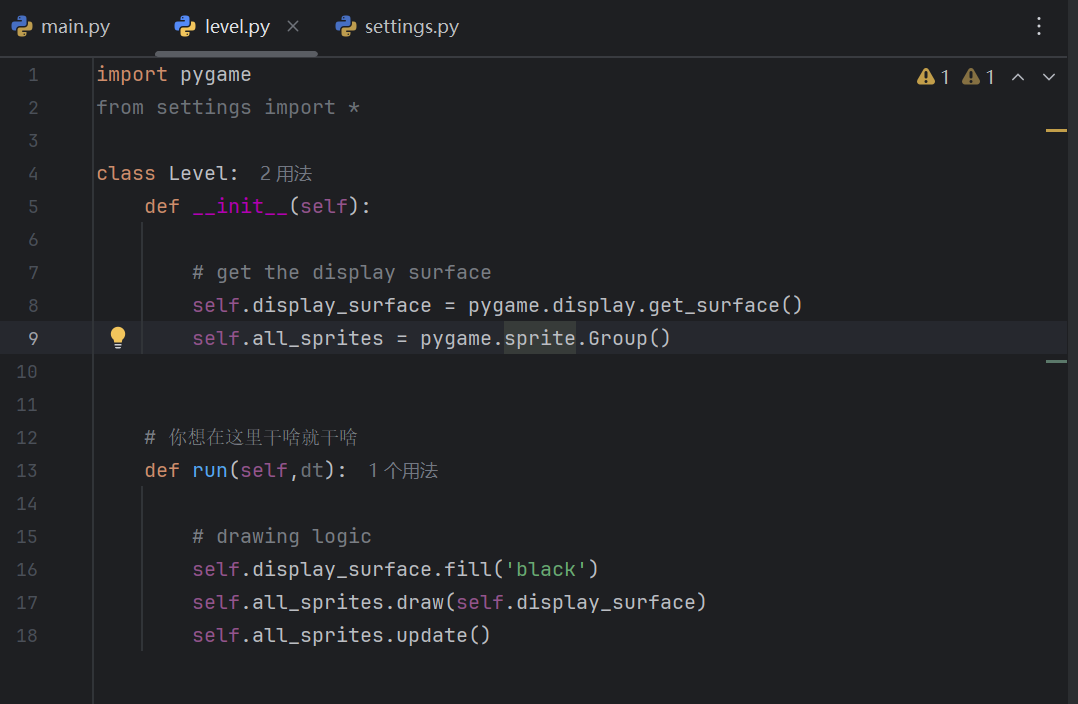
**Sprite**是我在游戏内互动的实体们：玩家，地砖，空气，等等上述都是sprite

Main.py是游戏的运行代码，它包含了标题名，规定了游戏的运行与退出操作，它导入自setting（界面设置和游戏内精灵的参数）和level（运行时干什么事）。

Setting.py规定了游戏窗口大小，以及游戏内参数的设定（没错，比如物品的价格，出入口的位置）

Level.py 直到目前为止定义了run方法，即运行时所要做的工作，所以未来会导入自很多的.py文件（如第二章的player）。run方法又调用了init方法

至此我们编写完成了main，setting，level的框架



**二．创建基本播放器（player.py）**

在player.py中创建Player类，继承自pygame.sprite.Sprite，因为player属于sprite

下面代码提供了self，pos（即玩家位置），Group（因为sprite属于组）。

def \_\_init\_\_(self, pos, group):

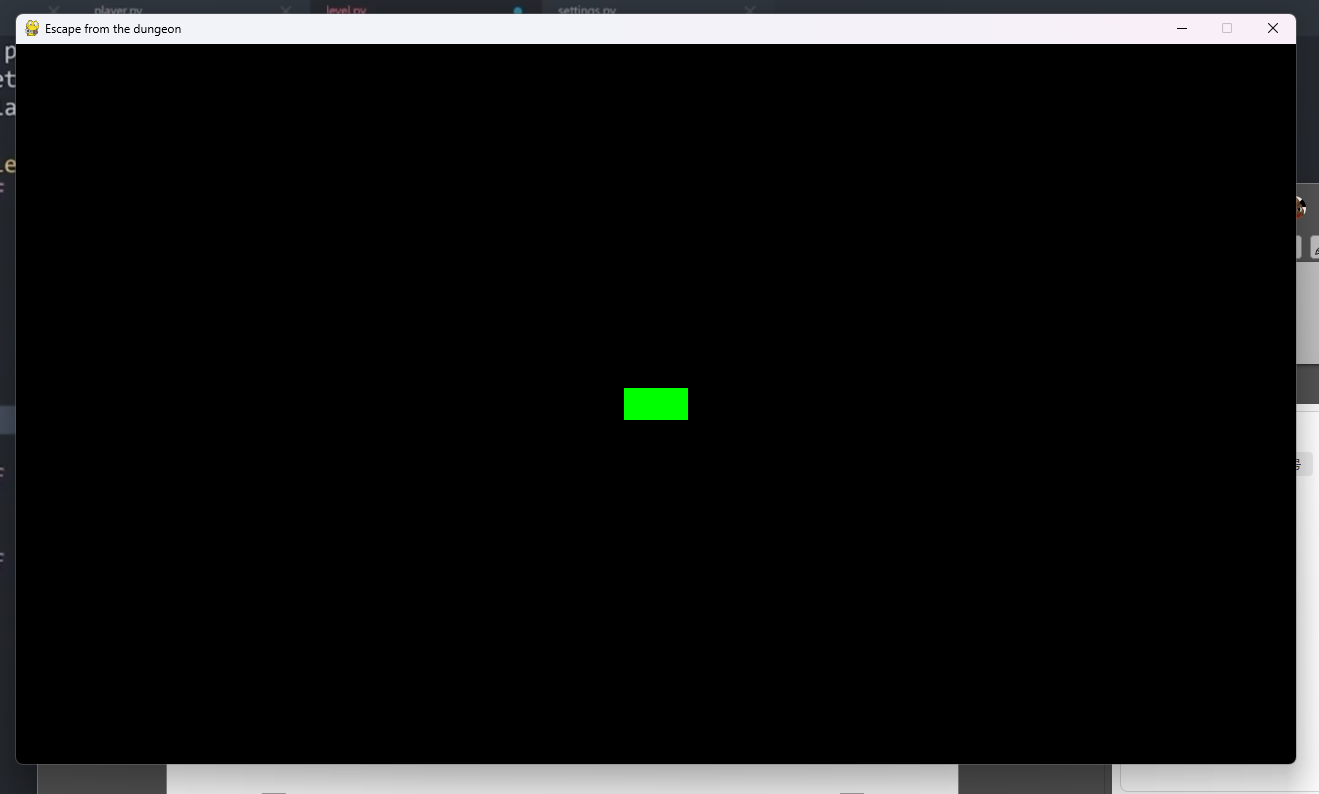
下面代码创建了一个基础的播放器，设置了大小为64x32，绿色，位置在中心

class Player(pygame.sprite.Sprite): #因为玩家也是sprite  
 def \_\_init\_\_(self, pos, group):  
 super().\_\_init\_\_(group)  
  
 self.image = pygame.Surface((64, 32))  
 self.image.fill('green')  
 self.rect = self.image.get\_rect(center = pos)

并且在level.py中添加下列代码使得播放器进入运行循环中

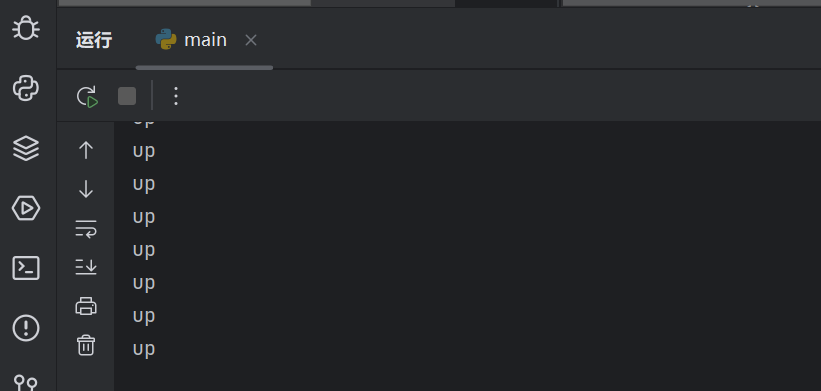
self.setup()  
  
def setup(self):  
 self.player = Player((640,360), self.all\_sprites)

**至此成功运行代码，屏幕中成功出现绿色框框，但是不能进行互动操作！！**



创建input方法并对我们的输入指令产生回应

def input(self):  
 keys = pygame.key.get\_pressed()  
  
 if keys[pygame.K\_UP]:  
 print('up')

当我在运行后按任一方向键，控制台开始打印up。

改变Input方法，接下来是将打印变成真正的位移，首先变更输入指令后导致的结果

#当输入向上的指令时，人物向上位移一个单位，状态更改为up，并且通过self.direction.x = 0使得停止按键时人物就不位移

if keys[pygame.K\_UP] and 1==1:  
 self.direction.y = -1  
 self.status = 'up'  
elif keys[pygame.K\_DOWN] and 1==1:  
 self.direction.y = 1  
 self.status = 'down'  
else:  
 self.direction.y = 0  
  
if keys[pygame.K\_RIGHT]:  
 self.direction.x = 1  
 self.status = 'right'  
elif keys[pygame.K\_LEFT]:  
 self.direction.x = -1  
 self.status = 'left'  
else:  
 self.direction.x = 0

然后我们创建move方法，通过下面公式（方向\*速度\*变量）计算了人物的运动

def move(self,dt):  
 self.pos += self.direction \* self.speed \* dt  
 self.rect.center = self.pos

**最后是update方法，我们在level.py只调用它，它保证先执行了input方法，后执行了move(dt)方法（当然后面还会添加别的操作），每次输入后紧接着移动。这样设计是为了模块化，我们对操作逻辑有修改时（比如把上下左右换成wsad）就可以只修改input方法了！**

def update(self, dt):  
 self.input()  
 self.move(dt)  
 #self.more....

额外调整：首先要归一化向量（pygame自带），使得对角线移动速度与单方向移动一致。然后考虑到游戏内水平和垂直碰撞体积不一样，对两个方向的移动做拆分便于后期调整。

#首先归一化向量，保证玩家在对角线移动的速度等于单向移动而非根号2倍，是pygame的内置功能  
if self.direction.magnitude() > 0:  
 self.direction = self.direction.normalize()

# horizontal movement  
self.pos.x += self.directio.x \* self.speed \* dt  
self.rect.centerx = self.pos.x  
# vertical movement  
self.pos.y += self.direction.y \* self.speed \* dt  
self.rect.centery = self.pos.y

**总结**

**至此播放器设置完成，运行后能实现绿色对象（self）的移动，并实现了归一化和水平垂直方向输入的拆分。**

**三，导入图形播放器**

即导入资源包，使用了<https://limezu.itch.io/moderninteriors>网站的系列资源包以代替原本的sprout lands的资源包。

在player.py中加入import\_assets方法，有animation变量（字典）和一个循环。变量存储了所有可能的动作，循环内不断读取移动所需的动作的路径，并对animations变量进行配对从而与输入动作进行匹配。

#人物（player）动作方法，进行路径读取

def import\_assets(self):  
 self.animations = {'up': [], 'down': [], 'left': [], 'right': [],  
 'right\_idle': [], 'left\_idle': [], 'up\_idle': [], 'down\_idle': [],  
 'right\_hoe': [], 'left\_hoe': [], 'up\_hoe': [], 'down\_hoe': [],  
 'right\_axe': [], 'left\_axe': [], 'up\_axe': [], 'down\_axe': [],  
 'right\_water': [], 'left\_water': [], 'up\_water': [], 'down\_water': []}  
  
 #这是负责读取的循环（在support方法中延申）  
 for animation in self.animations.keys():  
 full\_path = '../graphics/character/' + animation  
 self.animations[animation] = import\_folder(full\_path)

注意，这里的hoe，axe，water等动作可能不会保留，会进行后续的更改。

随后创建support.py。创建import\_folder(path)方法，即在player,py下import\_assets方法中的循环中的animations这个字典所配对的路径。形式为两个循环嵌套

返回player，我们现在要把小绿框替换为人物模型

原代码

#一般设置  
self.image = pygame.Surface((32, 64))  
self.image.fill('green')  
self.rect = self.image.get\_rect(center = pos)

更改后：

首先在init方法中补充两个变量方便测试：status（即input方法中人物在对应输入后应该切换为的状态，目前用’down\_idle’来测试，即人物面向下时的空闲状态对应图片）和帧索引（初次定义，设置为0）

self.import\_assets()  
self.status = 'down\_idle'  
self.frame\_index = 0

然后更改一般设置

#一般设置  
self.image = self.animations[self.status][self.frame\_index]  
self.rect = self.image.get\_rect(center = pos)

