存档资料 成绩：

南昌交通学院

课程设计报告书

课程名称 Python网络编程课程设计

题 目 雷霆篮球

分 院 人工智能学院

专业班级 软件工程

学 号

学生姓名

指导教师

2022 年 12 月 18 日

目录

[第一章 引言 3](#_Toc20366)

[1.1 目的 3](#_Toc18849)

[1.2 背景 3](#_Toc32691)

[1.3 需求分析 3](#_Toc272)

[第二章 概要设计 4](#_Toc8428)

[2.1 设计思路 4](#_Toc26989)

[2.2 设计流程图 5](#_Toc7136)

[第三章 系统详细设计 6](#_Toc5140)

[3.1绘制游戏布局 6](#_Toc8832)

[3.2 背景音效载入 6](#_Toc24855)

[3.3游戏游玩规则 7](#_Toc15087)

[3.4 游戏内容设计 7](#_Toc13052)

[第四章 测试与小结 8](#_Toc9931)

[4.1 游戏菜单测试 8](#_Toc12651)

[4.2 游戏规则测试 9](#_Toc12688)

[4.3 开始游戏测试 9](#_Toc6790)

[4.3 游戏补给测试 10](#_Toc1827)

[4.3 角色生命测试 10](#_Toc24507)

[4.4 重新开始测试 11](#_Toc31703)

[4.5 测试小结 11](#_Toc29254)

[第五章 代码设计 11](#_Toc14689)

[5.1模块介绍 11](#_Toc1225)

[5.2子弹类 12](#_Toc8339)

[5.3角色类 13](#_Toc29348)

[5.4多种敌人类 15](#_Toc28123)

[5.5 绘制布局 18](#_Toc17128)

[5.6 绘制敌人 22](#_Toc24283)

[5.7 游戏逻辑 22](#_Toc11300)

[参考文献 38](#_Toc18834)

# 第一章 引言

## 1.1 目的

通过此次课程设计，进一步加深对 python语言和运用的了解，将理论知识运用于开发的实践，并在实践中逐步掌握软件工具的使用。巩固已经学习过的python理论知识；进一步学习程序设计、程序调试的能力；进一步学习面向对象编程的知识和程序调试的能力;增强 Pycharm编程环境的应用能力，掌握并且熟练应用《Python编程案例教程》中所学知识。掌握类与对象之间的关系，使用函数肢解每个功能模块的方法，并运用此方法解决问题。学会程序测试方案的制定，并且实现程序的测试。通过此次实践，积累经验，提高分析和解决问题的能力。

我们小组决定通过编写类雷霆战机游戏来掌握和实践学习到的python知识。

## 1.2 背景

本课程设计是本专业的一门重要实践性教学环节。在学习了专业基础课和《Python程序设计》课程的基础上，本课程设计旨在加深对Python程序设计的认识，对Python语言及其语言生态有一个进一步的掌握和应用，学会运用Python标准库及外接相关库来解决实际问题的基本能力，培养和提高学生分析问题、解决问题的能力，尤其是提高我们学生使用Python为开发语言来进行问题描述、交流与思考的能力，为毕业设计和以后的工程实践打下良好的基础，经过小组讨论分析，我们最终选择了“雷霆战机”为基础的改版作为本次课设主题。

雷霆战机: 随着人类太空科技的飞速发展，希望水晶被越来越多的科学家当做核心能源来开发使用。人类社会也因为水晶资源的争夺，开始逐渐分化成两派。留在地球的普通人成立地球联邦，移居卫星的新人类建立行星同盟，两股势力开始各自发展自己的军备，纷争之势就此展开。跨越十二星座，一场争夺水晶的战争打响，肆意的战火点燃了静谧的星海。

游戏内容与经典的雷霆战机相似, 在面对经典玩法的无趣上我们小组进行美化创新与改变游戏节奏相结合, 让经典玩法焕发光彩。

## 1.3 需求分析

**创新需求**：

面对飞机大战这种经典游戏, 任何玩法上的创新都像在玩火, 所以我们小组决定在美工上进行创新, 让飞机大战以另一种形式展现出来。

**游戏性需求：**

根据调查发现人们不太愿意玩慢节奏的游戏, 所以我们小组决定提升项目的游戏节奏, 让游戏可玩性有所提升。

# 第二章 概要设计

## 2.1 设计思路

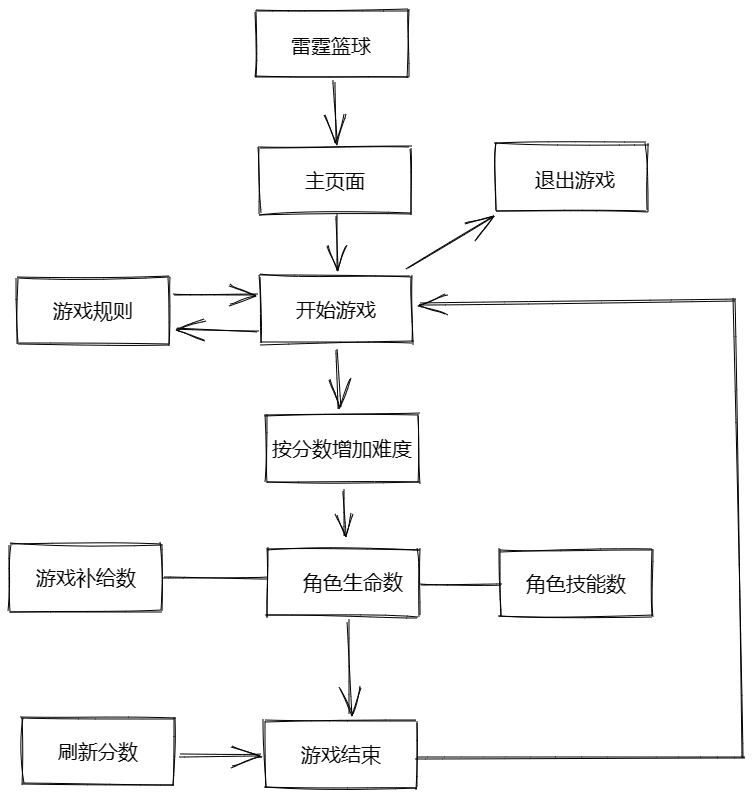
**游戏操作流程设计：**

进入主页面可以选择开始游戏和阅读游戏规则, 开始后WSAD控制角色移动杀敌, 敌人有三种类型会根据玩家得分进行难度调整, 期间每30秒会有补给品强化玩家, 游戏死亡后可以选择重新开始或退出游戏, 游玩的最高分会一直保留刷新。

**游戏界面设计：**

游戏界面采用手机屏幕比例的矩形窗口，给用户沉浸式体验与怀旧的街机体验。

## 2.2 设计流程图



# 系统详细设计

## 3.1绘制游戏布局

游戏界面使用的是pygame进行绘制, 绘制一个700\*480的矩形窗口, 窗口上设有标题雷霆篮球, 且初始化全局背景。

# 初始化pygame  
pygame.init()  
# 初始化pygame的音频模块  
pygame.mixer.init()  
# 定义背景尺寸宽和高  
bg\_size = width, height = 480, 700  
# 初始化窗口  
screen = pygame.display.set\_mode(bg\_size)  
# 设置显示在窗口上的名称  
pygame.display.set\_caption("雷霆篮球")  
# 初始化背景图片  
background = pygame.image.load("images/background.png").convert()  
# 定义RGB颜色  
BLACK = (0, 0, 0)  
GREEN = (0, 255, 0)  
RED = (255, 0, 0)

## 3.2 背景音效载入

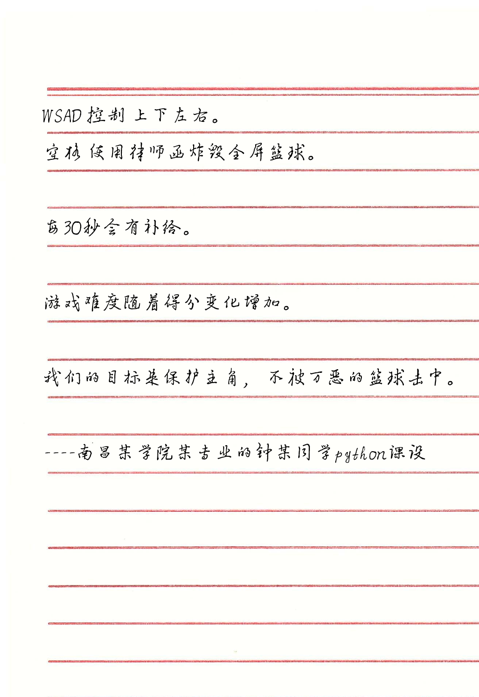
游戏开始时载入BGM只因你太美, 其它游戏音效也装载完毕待触发。

# 载入背景音乐  
pygame.mixer.music.load("sound/game\_music.ogg")  
pygame.mixer.music.set\_volume(0.2)  
# 载入游戏音效  
background\_sound = pygame.mixer.Sound("sound/background\_sound.wav")  
background\_sound.set\_volume(0.1)  
enemy3\_fly\_sound = pygame.mixer.Sound("sound/Organic Rhythm Assault.wav")  
enemy3\_fly\_sound.set\_volume(0.2)  
enemy3\_down\_sound = pygame.mixer.Sound("sound/爆炸.wav")  
enemy3\_down\_sound.set\_volume(0.2)  
bomb\_sound = pygame.mixer.Sound("sound/bomb\_sound.wav")  
bomb\_sound.set\_volume(0.2)  
get\_supply\_sound = pygame.mixer.Sound("sound/get\_bullet\_sound.wav")  
get\_supply\_sound.set\_volume(0.2)

## 3.3游戏游玩规则

规则以图片形式展示在游戏里。

* WSAD控制主角的上下左右
* 空格使用角色技能清空屏幕敌人
* 每 30s 会有补给强化角色
* 游戏随着得分发生难度变化
* 我们的目标是保护主角,不被万恶的篮球击中。



## 3.4 游戏内容设计

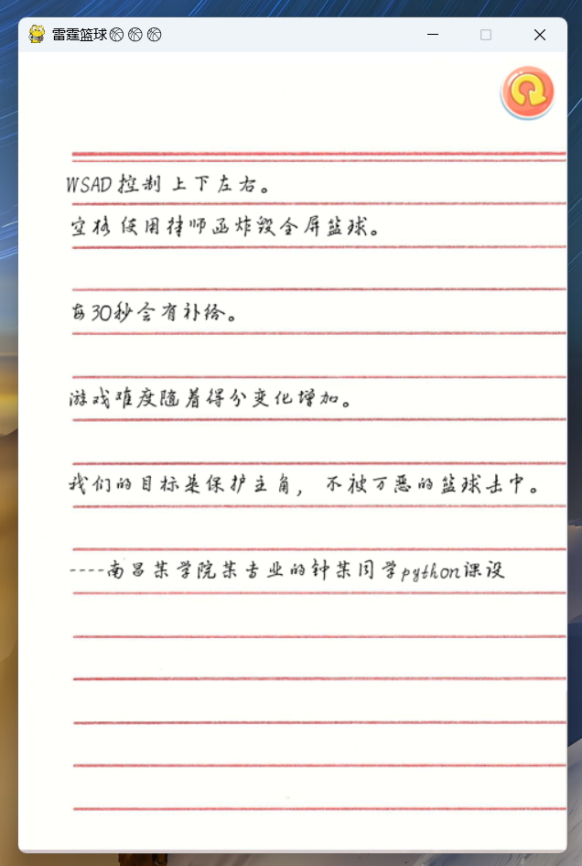
游戏内容与经典的雷霆战机相似, 在面对经典玩法的无趣上我们小组进行美化创新与改变游戏节奏相结合, 让经典玩法焕发光彩。

# 第四章 测试与小结

## 4.1 游戏菜单测试



## 4.2 游戏规则测试



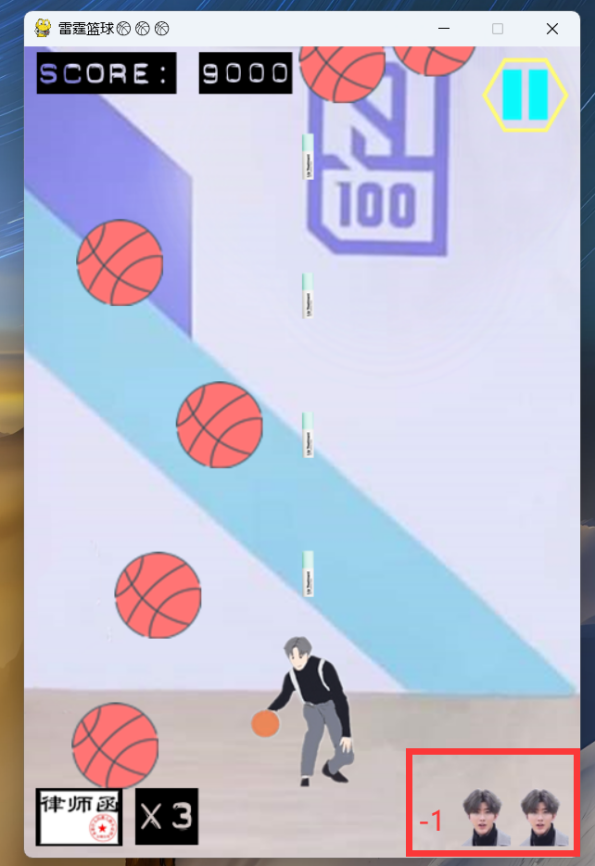
## 4.3 开始游戏测试



## 4.3 游戏补给测试



## 4.3 角色生命测试



## 4.4 重新开始测试



## 4.5 测试小结

菜单正常, 游戏页面渲染正常, 敌人刷新与死亡正常, 角色类攻击与死亡正常, 游戏补给品正常, 一切与代码理论一致, 无明显BUG。

# 第五章 代码设计

## 5.1模块介绍

* Font文件夹: 字体库
* Images文件夹: 图片材料库
* Sound文件夹: 音效材料库
* Bullet.py: 子弹类
* CXK.py: 角色类
* Enemy.py: pygame中的精灵桶,即本项目的各种篮球(敌人)类
* CXKRUN.py: 主体逻辑类

## 5.2子弹类

定义一个Bullet基础子弹类：

class Bullet(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, position):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("images/bullet\_1.png").convert\_alpha()  
 self.rect = self.image.get\_rect()  
 self.rect.left,self.rect.top = position  
 self.speed = 12  
 self.active = False  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 self.rect.top -= self.speed  
  
 if self.rect.top < 0:  
 self.active = False  
  
 def reset(self,position):  
 self.rect.left,self.rect.top = position  
 self.active = True

定义一个Bullet2加强子弹类：

class Bullet2(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, position):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("images/bullet\_2.png").convert\_alpha()  
 self.rect = self.image.get\_rect()  
 self.rect.left,self.rect.top = position  
 self.speed = 14  
 self.active = False  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 def move(self):  
 self.rect.top -= self.speed  
  
 if self.rect.top < 0:  
 self.active = False  
  
 def reset(self,position):  
 self.rect.left,self.rect.top = position  
 self.active = True

## 5.3角色类

定义一个CXK类：

包含角色的上下左右移动

class CXK(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image1 = pygame.image.load("images/CXK\_1.png").convert\_alpha()  
 self.image2 = pygame.image.load("images/CXK\_2.png").convert\_alpha()  
 self.image3 = pygame.image.load("images/CXK\_3.png").convert\_alpha()  
 self.image4 = pygame.image.load("images/CXK\_4.png").convert\_alpha()  
 self.image5 = pygame.image.load("images/CXK\_5.png").convert\_alpha()  
 self.image6 = pygame.image.load("images/CXK\_6.png").convert\_alpha()  
 self.image7 = pygame.image.load("images/CXK\_7.png").convert\_alpha()  
 self.image8 = pygame.image.load("images/CXK\_8.png").convert\_alpha()  
 self.image9 = pygame.image.load("images/CXK\_9.png").convert\_alpha()  
 self.image10 = pygame.image.load("images/CXK\_10.png").convert\_alpha()  
 self.image11 = pygame.image.load("images/CXK\_11.png").convert\_alpha()  
 self.image12 = pygame.image.load("images/CXK\_12.png").convert\_alpha()  
 self.image13 = pygame.image.load("images/CXK\_13.png").convert\_alpha()  
 self.image14 = pygame.image.load("images/CXK\_14.png").convert\_alpha()  
 self.image15 = pygame.image.load("images/CXK\_15.png").convert\_alpha()  
 self.image16 = pygame.image.load("images/CXK\_16.png").convert\_alpha()  
 self.image17 = pygame.image.load("images/CXK\_17.png").convert\_alpha()  
 self.image18 = pygame.image.load("images/CXK\_18.png").convert\_alpha()  
  
 self.destroy\_images = []  
 self.destroy\_images.extend([\  
 pygame.image.load("images/CXK\_down\_1.png").convert\_alpha(),\  
 pygame.image.load("images/CXK\_down\_2.png").convert\_alpha(),\  
 pygame.image.load("images/CXK\_down\_3.png").convert\_alpha(),\  
 pygame.image.load("images/CXK\_down\_4.png").convert\_alpha()])  
  
 self.rect = self.image1.get\_rect()  
 self.width,self.height = bg\_size[0],bg\_size[1]  
 self.rect.left,self.rect.top = (self.width - self.rect.width) /2,self.height - self.rect.height - 60  
 self.speed = 10  
 self.active = True  
 self.invincible = False  
 self.mask = pygame.mask.from\_surface(self.image1)  
  
 def moveUP(self):  
 if self.rect.top>0:  
 self.rect.top -= self.speed  
 else:  
 self.rect.top = 0  
  
 def moveDOWN(self):  
 if self.rect.bottom < self.height - 60:  
 self.rect.top += self.speed  
 else:  
 self.rect.bottom = self.height - 60  
  
 def moveLEFT(self):  
 if self.rect.left > 0:  
 self.rect.left -= self.speed  
 else:  
 self.rect.left = 0  
  
 def moveRIGHT(self):  
 if self.rect.right < self.width:  
 self.rect.left += self.speed  
 else:  
 self.rect.right = self.width  
  
 def reset(self):  
 self.rect.left,self.rect.top = (self.width - self.rect.width) /2,self.height - self.rect.height - 60  
 self.active = True  
 self.invincible = True

## 5.4多种敌人类

基础敌人:

class SmallEnemy(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("images/small\_basketball.png").convert\_alpha()  
  
 self.destroy\_images = []  
 self.destroy\_images.extend([\  
 pygame.image.load("images/small\_basketball\_down\_1.png").convert\_alpha(),\  
 pygame.image.load("images/small\_basketball\_down\_2.png").convert\_alpha(),\  
 pygame.image.load("images/small\_basketball\_down\_3.png").convert\_alpha()])  
  
 self.rect = self.image.get\_rect()  
 self.width,self.height = bg\_size[0],bg\_size[1]  
 self.speed = 2  
 self.active = True  
 self.mask = pygame.mask.from\_surface(self.image)  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-5\*self.height,0)  
  
 def move(self):  
 if self.rect.top < self.height:  
 self.rect.top += self.speed  
 else:  
 self.reset  
  
 def reset(self):  
 self.active = True  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-5\*self.height,0)

中等敌人:

class MidEnemy(pygame.sprite.Sprite):  
  
 energy = 8  
  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image = pygame.image.load("images/mid\_basketball.png").convert\_alpha()  
  
 self.destroy\_images = []  
 self.destroy\_images.extend([\  
 pygame.image.load("images/mid\_basketball\_down\_1.png").convert\_alpha(),\  
 pygame.image.load("images/mid\_basketball\_down\_2.png").convert\_alpha(),\  
 pygame.image.load("images/mid\_basketball\_down\_3.png").convert\_alpha()])  
  
 self.rect = self.image.get\_rect()  
 self.width,self.height = bg\_size[0],bg\_size[1]  
 self.speed = 1  
 self.active = True  
 self.mask = pygame.mask.from\_surface(self.image)  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-10\*self.height,-self.height)  
 self.energy = MidEnemy.energy  
  
 def move(self):  
 if self.rect.top < self.height+5:  
 self.rect.top += self.speed  
 else:  
 self.reset  
  
 def reset(self):  
 self.active = True  
 self.energy = MidEnemy.energy  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-10\*self.height,-self.height)

大型敌人:

class BigEnemy(pygame.sprite.Sprite):  
  
 energy = 20  
  
 def \_\_init\_\_(self, bg\_size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
  
 self.image1 = pygame.image.load("images/big\_basketball\_1.png").convert\_alpha()  
 self.image2 = pygame.image.load("images/big\_basketball\_2.png").convert\_alpha()  
 self.image3 = pygame.image.load("images/big\_basketball\_3.png").convert\_alpha()  
 self.image4 = pygame.image.load("images/big\_basketball\_4.png").convert\_alpha()  
 self.image5 = pygame.image.load("images/big\_basketball\_5.png").convert\_alpha()  
 self.image6 = pygame.image.load("images/big\_basketball\_6.png").convert\_alpha()  
 self.image7 = pygame.image.load("images/big\_basketball\_7.png").convert\_alpha()  
 self.image8 = pygame.image.load("images/big\_basketball\_8.png").convert\_alpha()  
 self.image9 = pygame.image.load("images/big\_basketball\_9.png").convert\_alpha()  
 self.image10 = pygame.image.load("images/big\_basketball\_10.png").convert\_alpha()  
 self.image11 = pygame.image.load("images/big\_basketball\_11.png").convert\_alpha()  
 self.image12 = pygame.image.load("images/big\_basketball\_12.png").convert\_alpha()  
 self.image13 = pygame.image.load("images/big\_basketball\_13.png").convert\_alpha()  
 self.image14 = pygame.image.load("images/big\_basketball\_14.png").convert\_alpha()  
 self.image15 = pygame.image.load("images/big\_basketball\_15.png").convert\_alpha()  
 self.image16 = pygame.image.load("images/big\_basketball\_16.png").convert\_alpha()  
 self.image17 = pygame.image.load("images/big\_basketball\_17.png").convert\_alpha()  
 self.image18 = pygame.image.load("images/big\_basketball\_18.png").convert\_alpha()  
 self.image19 = pygame.image.load("images/big\_basketball\_19.png").convert\_alpha()  
  
 self.destroy\_images = []  
 self.destroy\_images.extend([\  
 pygame.image.load("images/big\_basketball\_down\_1.png").convert\_alpha(),\  
 pygame.image.load("images/big\_basketball\_down\_2.png").convert\_alpha(),\  
 pygame.image.load("images/big\_basketball\_down\_3.png").convert\_alpha()])  
  
 self.rect = self.image1.get\_rect()  
 self.width,self.height = bg\_size[0],bg\_size[1]  
 self.speed = 1  
 self.active = True  
 self.mask = pygame.mask.from\_surface(self.image1)  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-15\*self.height,-5\*self.height)  
 self.energy = BigEnemy.energy  
  
 def move(self):  
 if self.rect.top < self.height+5:  
 self.rect.top += self.speed  
 else:  
 self.reset  
  
 def reset(self):  
 self.active = True  
 self.energy = BigEnemy.energy  
 self.rect.left,self.rect.top =randint(0,self.width - self.rect.width),randint(-15\*self.height,-5\*self.height)

## 5.5 绘制布局

# 初始化pygame  
pygame.init()  
# 初始化pygame的音频模块  
pygame.mixer.init()  
# 定义背景尺寸宽和高  
bg\_size = width, height = 480, 700  
# 初始化窗口  
screen = pygame.display.set\_mode(bg\_size)  
# 设置显示在窗口上的名称  
pygame.display.set\_caption("雷霆篮球")  
# 初始化背景图片  
background = pygame.image.load("images/background.png").convert()  
# 定义RGB颜色  
BLACK = (0, 0, 0)  
GREEN = (0, 255, 0)  
RED = (255, 0, 0)  
  
# 载入背景音乐  
pygame.mixer.music.load("sound/game\_music.ogg")  
pygame.mixer.music.set\_volume(0.2)  
# 载入游戏音效  
background\_sound = pygame.mixer.Sound("sound/background\_sound.wav")  
background\_sound.set\_volume(0.1)  
enemy3\_fly\_sound = pygame.mixer.Sound("sound/Organic Rhythm Assault.wav")  
enemy3\_fly\_sound.set\_volume(0.2)  
enemy3\_down\_sound = pygame.mixer.Sound("sound/爆炸.wav")  
enemy3\_down\_sound.set\_volume(0.2)  
bomb\_sound = pygame.mixer.Sound("sound/bomb\_sound.wav")  
bomb\_sound.set\_volume(0.2)  
get\_supply\_sound = pygame.mixer.Sound("sound/get\_bullet\_sound.wav")  
get\_supply\_sound.set\_volume(0.2)

def ui():  
 # 循环播放背景音乐  
 pygame.mixer.music.play(-1)  
  
 # 初始化界面按键图片并获取图片的矩形位置  
 start\_game\_image = pygame.image.load("images/start\_game.png").convert\_alpha()  
 start\_game\_image\_rect = start\_game\_image.get\_rect()  
 game\_rules\_image = pygame.image.load("images/game\_rules.png").convert\_alpha()  
 game\_rules\_image\_rect = game\_rules\_image.get\_rect()  
 game\_quit\_image = pygame.image.load("images/game\_quit.png").convert\_alpha()  
 game\_quit\_image\_rect = game\_quit\_image.get\_rect()  
  
 # 初始化游戏规则图片并获取图片的矩形位置  
 rules\_image = pygame.image.load("images/游戏玩法.png").convert\_alpha()  
 back\_image = pygame.image.load("images/back.png").convert\_alpha()  
 back\_image\_rect = back\_image.get\_rect()  
  
 # 标志是否在主界面  
 is\_ui = True  
  
 # 帧率  
 clock = pygame.time.Clock()  
  
 # 主界面循环  
 while True:  
 # 获取事件信息  
 for event in pygame.event.get():  
 # 如果点击右上角退出  
 if event.type == QUIT:  
 # 退出程序  
 pygame.quit()  
 sys.exit()  
  
 # 如果是主界面  
 if is\_ui:  
 # 绘制背景  
 screen.blit(background, (0, 0))  
  
 # 更改主界面按键图片的矩形位置并绘制主界面按键  
 start\_game\_image\_rect.left, start\_game\_image\_rect.top = (  
 width - start\_game\_image\_rect.width) // 2, height - 500  
 screen.blit(start\_game\_image, start\_game\_image\_rect)  
  
 game\_rules\_image\_rect = game\_rules\_image.get\_rect()  
 game\_rules\_image\_rect.left, game\_rules\_image\_rect.top = (  
 width - game\_rules\_image\_rect.width) // 2, start\_game\_image\_rect.bottom + 50  
 screen.blit(game\_rules\_image, game\_rules\_image\_rect)  
  
 game\_quit\_image\_rect.left, game\_quit\_image\_rect.top = (  
 width - game\_quit\_image\_rect.width) // 2, game\_rules\_image\_rect.bottom + 50  
 screen.blit(game\_quit\_image, game\_quit\_image\_rect)  
  
 # 检测用户的鼠标操作  
 # 如果用户按下鼠标左键  
 if pygame.mouse.get\_pressed()[0]:  
 # 获取鼠标坐标  
 pos = pygame.mouse.get\_pos()  
 # 如果用户点击”开始游戏“  
 if start\_game\_image\_rect.left < pos[0] < start\_game\_image\_rect.right and start\_game\_image\_rect.top < \  
 pos[1] < start\_game\_image\_rect.bottom:  
 # 调用主函数  
 main()  
 # 如果用户点击”退出游戏“  
 if game\_quit\_image\_rect.left < pos[0] < game\_quit\_image\_rect.right and game\_quit\_image\_rect.top < pos[  
 1] < game\_quit\_image\_rect.bottom:  
 pygame.quit()  
 sys.exit()  
 # 如果用户点击”游戏规则“  
 if game\_rules\_image\_rect.left < pos[0] < game\_rules\_image\_rect.right and game\_rules\_image\_rect.top < \  
 pos[1] < game\_rules\_image\_rect.bottom:  
 # 离开主界面  
 is\_ui = False  
  
 # 进入游戏规则界面  
 else:  
 # 绘制游戏规则图片  
 screen.blit(rules\_image, (0, 0))  
  
 # 停止播放背景音乐  
 pygame.mixer.music.stop()  
 # 循环播放游戏规则音效  
 background\_sound.play(-1)  
  
 # 更改返回按键图片的矩形位置并绘制返回按键  
 back\_image\_rect.left, game\_quit\_image\_rect.top = width - back\_image\_rect.width - 10, 10  
 screen.blit(back\_image, (width - back\_image\_rect.width - 10, 10))  
  
 if pygame.mouse.get\_pressed()[0]:  
 # 获取鼠标坐标  
 pos = pygame.mouse.get\_pos()  
 # 如果用户点击返回图片  
 if back\_image\_rect.left < pos[0] < back\_image\_rect.right and back\_image\_rect.top < pos[  
 1] < back\_image\_rect.bottom:  
 # 背景音乐停止并进入主界面  
 pygame.mixer.stop()  
 ui()  
  
 # 刷新屏幕  
 pygame.display.flip()  
  
 # 设置帧率为60帧  
 clock.tick(60)

## 5.6 绘制敌人

# 定义增加小型敌人的函数  
def add\_small\_enemies(group1, group2, num):  
 for i in range(num):  
 e1 = enemy.SmallEnemy(bg\_size)  
 group1.add(e1)  
 group2.add(e1)  
  
  
# 定义增加中型敌人的函数  
def add\_mid\_enemies(group1, group2, num):  
 for i in range(num):  
 e2 = enemy.MidEnemy(bg\_size)  
 group1.add(e2)  
 group2.add(e2)  
  
  
# 定义增加大型敌人的函数  
def add\_big\_enemies(group1, group2, num):  
 for i in range(num):  
 e3 = enemy.BigEnemy(bg\_size)  
 group1.add(e3)  
 group2.add(e3)  
  
  
# 定义增加敌人移动速度的函数  
def inc\_speed(target, inc):  
 for each in target:  
 each.speed += inc

## 5.7 游戏逻辑

def main():  
 # 循环播放背景音乐  
 pygame.mixer.music.play(-1)  
  
 # 用于计算未暂停时经过的时间  
 TIME = 0  
  
 # 生成CXK  
 me = CXK.CXK(bg\_size)  
  
 # 用于存放敌人  
 enemies = pygame.sprite.Group()  
  
 # 生成小型敌人  
 small\_enemies = pygame.sprite.Group()  
 add\_small\_enemies(small\_enemies, enemies, 40)  
  
 # 生成中型敌人  
 mid\_enemies = pygame.sprite.Group()  
 add\_mid\_enemies(mid\_enemies, enemies, 10)  
  
 # 生成大型敌人  
 big\_enemies = pygame.sprite.Group()  
 add\_big\_enemies(big\_enemies, enemies, 6)  
  
 # 生成普通子弹  
 bullet1 = []  
 bullet1\_index = 0  
 BULLET1\_NUM = 4  
 for i in range(BULLET1\_NUM):  
 bullet1.append(bullet.Bullet(me.rect.midtop))  
  
 # 生成超级子弹  
 bullet2 = []  
 bullet2\_index = 0  
 BULLET2\_NUM = 8  
 for i in range(BULLET2\_NUM // 2):  
 bullet2.append(bullet.Bullet2((me.rect.centerx - 10, me.rect.centery)))  
 bullet2.append(bullet.Bullet2((me.rect.centerx + 10, me.rect.centery)))  
  
 clock = pygame.time.Clock()  
  
 # 爆炸图片索引  
 e1\_destroy\_index = 0  
 e2\_destroy\_index = 0  
 e3\_destroy\_index = 0  
 me\_destroy\_index = 0  
  
 # 统计得分  
 score = 0  
 score\_font = pygame.font.Font("font/font.ttf", 36)  
  
 # 暂停  
 paused = False  
 pause\_nor\_image = pygame.image.load("images/pause\_1.png").convert\_alpha()  
 pause\_pressed\_image = pygame.image.load("images/pause\_2.png").convert\_alpha()  
 resume\_nor\_image = pygame.image.load("images/start\_1.png").convert\_alpha()  
 resume\_pressed\_image = pygame.image.load("images/start\_2.png").convert\_alpha()  
 pause\_rect = pause\_nor\_image.get\_rect()  
 pause\_rect.left, pause\_rect.top = width - pause\_rect.width - 10, 10  
 paused\_image = pause\_nor\_image  
  
 # 设置难度级别  
 level = 1  
  
 # 全屏炸弹  
 bomb\_image = pygame.image.load("images/bomb.png").convert\_alpha()  
 bomb\_rect = bomb\_image.get\_rect()  
 bomb\_font = pygame.font.Font("font/font.ttf", 48)  
 bomb\_num = 3  
  
 # 护盾  
 shield\_image = pygame.image.load("images/shield.png").convert\_alpha()  
 shield\_rect = shield\_image.get\_rect()  
  
 # 每10秒发放一个补给包  
 bullet\_supply = supply.Bullet\_Supply(bg\_size)  
 bomb\_supply = supply.Bomb\_Supply(bg\_size)  
 shield\_supply = supply.Shield\_Supply(bg\_size)  
 SUPPLY\_TIME = USEREVENT  
 pygame.time.set\_timer(SUPPLY\_TIME, 10 \* 1000)  
  
 # 超级子弹定时器  
 DOUBLE\_BULLET\_TIME = USEREVENT + 1  
  
 # 无敌时间计时器  
 INVINCIBLE\_TIME = USEREVENT + 2  
  
 # 暂停计时器  
 PAUSE\_TIME = USEREVENT + 3  
 pygame.time.set\_timer(PAUSE\_TIME, 1 \* 1000)  
  
 # 标志是否使用超级子弹  
 is\_double\_bullet = False  
  
 # 标志是否有护盾  
 is\_protected = False  
  
 # 生命数量  
 life\_image = pygame.image.load("images/life.png").convert\_alpha()  
 life\_rect = life\_image.get\_rect()  
 life\_num = 3  
  
 # 游戏结束画面  
 gameover\_font = pygame.font.Font("font/font.ttf", 48)  
 again\_image = pygame.image.load("images/again.png").convert\_alpha()  
 again\_rect = again\_image.get\_rect()  
 gameover\_image = pygame.image.load("images/gameover.png").convert\_alpha()  
 gameover\_rect = gameover\_image.get\_rect()  
  
 # 用于切换图片  
 CXK\_switch\_image = 1  
 enemy\_switch\_image = 1  
  
 # 用于延迟  
 delay = 100  
 enemy\_delay = 15  
  
 # 用于限制重复打开记录文件  
 recorded = False  
  
 # 标志是否暂停过  
 is\_pause\_time = False  
  
 # 主函数循环  
 while True:  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
  
 # 如果有鼠标点击事件  
 elif event.type == MOUSEBUTTONDOWN:  
 # 如果是鼠标左键点击暂停按键  
 if event.button == 1 and pause\_rect.collidepoint(event.pos):  
 # 更改暂停状态  
 paused = not paused  
 # 如果当前为暂停  
 if paused:  
 # 标志暂停过  
 is\_pause\_time = True  
 # 暂停补给投放  
 pygame.time.set\_timer(SUPPLY\_TIME, 0)  
 # 背景音乐暂停  
 pygame.mixer.music.pause()  
 # 音效暂停  
 pygame.mixer.pause()  
 # 如果当前不是暂停状态  
 else:  
 # 如果曾经暂停过  
 if is\_pause\_time:  
 # 设置补给投放时间为10秒减去之前未暂停时经过的时间  
 pygame.time.set\_timer(SUPPLY\_TIME, (10 - TIME) \* 1000)  
 # 从新标志位未暂停  
 is\_pause\_time = False  
 # 继续播放背景音乐  
 pygame.mixer.music.unpause()  
 # 继续播放音效  
 pygame.mixer.unpause()  
  
 # 检测鼠标移动事件  
 elif event.type == MOUSEMOTION:  
 # 根据鼠标位置更改暂停/继续按键的样式  
 if pause\_rect.collidepoint(event.pos):  
 if paused:  
 paused\_image = resume\_pressed\_image  
 else:  
 paused\_image = pause\_pressed\_image  
 else:  
 if paused:  
 paused\_image = resume\_nor\_image  
 else:  
 paused\_image = pause\_nor\_image  
  
 # 检测键盘按下的事件  
 elif event.type == KEYDOWN:  
 # 如果按下空格键  
 if event.key == K\_SPACE:  
 # 如果炸弹数量不为0  
 if bomb\_num:  
 # 炸弹数量-1  
 bomb\_num -= 1  
 # 使用炸弹音效播放  
 bomb\_sound.play()  
 # 屏幕内所有敌人暴毙  
 for each in enemies:  
 if each.rect.bottom > 0:  
 each.active = False  
  
 # 如果为用户自定义事件发放补给  
 elif event.type == SUPPLY\_TIME:  
 # 随机选择一个补给发放  
 Choice = int(choice([1, 2, 3]))  
 if Choice == 1:  
 bomb\_supply.reset()  
 if Choice == 2:  
 bullet\_supply.reset()  
 if Choice == 3:  
 shield\_supply.reset()  
 # 发放补给后计时清零  
 TIME = 0  
  
 # 双倍子弹结束  
 elif event.type == DOUBLE\_BULLET\_TIME:  
 # 双倍子弹失效  
 is\_double\_bullet = False  
 # 取消双倍子弹计时器  
 pygame.time.set\_timer(DOUBLE\_BULLET\_TIME, 0)  
  
 # 无敌时间结束  
 elif event.type == INVINCIBLE\_TIME:  
 # 不无敌  
 me.invincible = False  
 # 取消无敌时间计时器  
 pygame.time.set\_timer(INVINCIBLE\_TIME, 0)  
  
 # 非暂停时间自动30秒内计时  
 elif event.type == PAUSE\_TIME:  
 if not paused:  
 if TIME < 29:  
 TIME += 1  
 else:  
 TIME = 0  
  
 # 根据用户得分增加难度  
 if level == 1 and score > 50000:  
 level = 2  
 # 增加10小型敌人，6个中型敌人，4个大型敌人  
 add\_small\_enemies(small\_enemies, enemies, 10)  
 add\_mid\_enemies(mid\_enemies, enemies, 6)  
 add\_big\_enemies(big\_enemies, enemies, 4)  
 # 提升小型敌人的速度  
 inc\_speed(small\_enemies, 1)  
  
 if level == 2 and score > 300000:  
 level = 3  
 # 增加10小型敌人，6个中型敌人，4个大型敌人  
 add\_small\_enemies(small\_enemies, enemies, 10)  
 add\_mid\_enemies(mid\_enemies, enemies, 6)  
 add\_big\_enemies(big\_enemies, enemies, 4)  
 # 提升小型敌人的速度  
 inc\_speed(small\_enemies, 1)  
 # 提升中型敌人的速度  
 inc\_speed(mid\_enemies, 1)  
  
 if level == 3 and score > 600000:  
 level = 4  
 # 增加10小型敌人，6个中型敌人，4个大型敌人  
 add\_small\_enemies(small\_enemies, enemies, 10)  
 add\_mid\_enemies(mid\_enemies, enemies, 6)  
 add\_big\_enemies(big\_enemies, enemies, 4)  
 # 提升小型敌人的速度  
 inc\_speed(small\_enemies, 1)  
 # 提升中型敌人的速度  
 inc\_speed(mid\_enemies, 1)  
  
 if level == 4 and score > 1000000:  
 level = 5  
 # 增加10小型敌人，6个中型敌人，4个大型敌人  
 add\_small\_enemies(small\_enemies, enemies, 10)  
 add\_mid\_enemies(mid\_enemies, enemies, 6)  
 add\_big\_enemies(big\_enemies, enemies, 4)  
 # 提升小型敌人的速度  
 inc\_speed(small\_enemies, 1)  
 # 提升中型敌人的速度  
 inc\_speed(mid\_enemies, 1)  
  
 screen.blit(background, (0, 0))  
  
 if life\_num and not paused:  
  
 # 检测用户的键盘操作  
 key\_pressed = pygame.key.get\_pressed()  
  
 if key\_pressed[K\_w] or key\_pressed[K\_UP]:  
 me.moveUP()  
 if key\_pressed[K\_s] or key\_pressed[K\_DOWN]:  
 me.moveDOWN()  
 if key\_pressed[K\_a] or key\_pressed[K\_LEFT]:  
 me.moveLEFT()  
 if key\_pressed[K\_d] or key\_pressed[K\_RIGHT]:  
 me.moveRIGHT()  
  
 # 绘制炸弹补给  
 if bomb\_supply.active:  
 bomb\_supply.move()  
 screen.blit(bomb\_supply.image, bomb\_supply.rect)  
 # 碰撞检测  
 if pygame.sprite.collide\_mask(bomb\_supply, me):  
 get\_supply\_sound.play()  
 if bomb\_num < 3:  
 bomb\_num += 1  
 bomb\_supply.active = False  
  
 # 绘制子弹补给  
 if bullet\_supply.active:  
 bullet\_supply.move()  
 screen.blit(bullet\_supply.image, bullet\_supply.rect)  
 # 碰撞检测  
 if pygame.sprite.collide\_mask(bullet\_supply, me):  
 get\_supply\_sound.play()  
 is\_double\_bullet = True  
 pygame.time.set\_timer(DOUBLE\_BULLET\_TIME, 18 \* 1000)  
 bullet\_supply.active = False  
  
 # 绘制护盾补给  
 if shield\_supply.active:  
 shield\_supply.move()  
 # 碰撞检测  
 screen.blit(shield\_supply.image, shield\_supply.rect)  
 if pygame.sprite.collide\_mask(shield\_supply, me):  
 get\_supply\_sound.play()  
 is\_protected = True  
 shield\_supply.active = False  
  
 # 绘制子弹  
 if not (delay % 10):  
 if is\_double\_bullet:  
 bullets = bullet2  
 bullets[bullet2\_index].reset((me.rect.centerx - 10, me.rect.centery))  
 bullets[bullet2\_index + 1].reset((me.rect.centerx + 10, me.rect.centery))  
 bullet2\_index = (bullet2\_index + 2) % BULLET2\_NUM  
 else:  
 bullets = bullet1  
 bullets[bullet1\_index].reset(me.rect.midtop)  
 bullet1\_index = (bullet1\_index + 1) % BULLET1\_NUM  
  
 # 检测子弹是否击中敌人  
 for b in bullets:  
 if b.active:  
 b.move()  
 screen.blit(b.image, b.rect)  
 # 碰撞检测  
 enemy\_hit = pygame.sprite.spritecollide(b, enemies, False, pygame.sprite.collide\_mask)  
 if enemy\_hit:  
 b.active = False  
 for e in enemy\_hit:  
 if e in mid\_enemies or e in big\_enemies:  
 e.energy -= 1  
 if e.energy == 0:  
 e.active = False  
 else:  
 e.active = False  
  
 # 绘制大型敌人  
 for each in big\_enemies:  
 if each.active:  
 each.move()  
  
 if enemy\_switch\_image == 1:  
 screen.blit(each.image1, each.rect)  
 if enemy\_switch\_image == 2:  
 screen.blit(each.image2, each.rect)  
 if enemy\_switch\_image == 3:  
 screen.blit(each.image3, each.rect)  
 if enemy\_switch\_image == 4:  
 screen.blit(each.image4, each.rect)  
 if enemy\_switch\_image == 5:  
 screen.blit(each.image5, each.rect)  
 if enemy\_switch\_image == 6:  
 screen.blit(each.image6, each.rect)  
 if enemy\_switch\_image == 7:  
 screen.blit(each.image7, each.rect)  
 if enemy\_switch\_image == 8:  
 screen.blit(each.image8, each.rect)  
 if enemy\_switch\_image == 9:  
 screen.blit(each.image9, each.rect)  
 if enemy\_switch\_image == 10:  
 screen.blit(each.image10, each.rect)  
 if enemy\_switch\_image == 11:  
 screen.blit(each.image11, each.rect)  
 if enemy\_switch\_image == 12:  
 screen.blit(each.image12, each.rect)  
 if enemy\_switch\_image == 13:  
 screen.blit(each.image13, each.rect)  
 if enemy\_switch\_image == 14:  
 screen.blit(each.image14, each.rect)  
 if enemy\_switch\_image == 15:  
 screen.blit(each.image15, each.rect)  
 if enemy\_switch\_image == 16:  
 screen.blit(each.image16, each.rect)  
 if enemy\_switch\_image == 17:  
 screen.blit(each.image17, each.rect)  
 if enemy\_switch\_image == 18:  
 screen.blit(each.image18, each.rect)  
 if enemy\_switch\_image == 19:  
 screen.blit(each.image19, each.rect)  
 enemy\_switch\_image = 1  
  
 # 每15帧切换一次图片  
 if not enemy\_delay:  
 enemy\_switch\_image += 1  
 enemy\_delay = 15  
 else:  
 enemy\_delay -= 1  
  
 # 绘制血槽  
 pygame.draw.line(screen, BLACK, (each.rect.left, each.rect.top - 5),  
 (each.rect.right, each.rect.top - 5), 2)  
 energy\_remain = each.energy / enemy.BigEnemy.energy  
 if energy\_remain > 0.2:  
 energy\_color = GREEN  
 else:  
 energy\_color = RED  
 pygame.draw.line(screen, energy\_color, (each.rect.left, each.rect.top - 5),  
 (each.rect.left + each.rect.width \* energy\_remain, each.rect.top - 5), 2)  
  
 # 载入音效  
 if each.rect.bottom == -50:  
 enemy3\_fly\_sound.play(-1)  
 else:  
 # 毁灭  
 if not (delay % 3):  
 if e3\_destroy\_index == 0:  
 enemy3\_down\_sound.play()  
 # 绘制毁灭画面  
 screen.blit(each.destroy\_images[e3\_destroy\_index], each.rect)  
 e3\_destroy\_index = (e3\_destroy\_index + 1) % 3  
 if e3\_destroy\_index == 0:  
 enemy3\_fly\_sound.stop()  
 score += 10000  
 each.reset()  
  
 # 绘制中型敌人  
 for each in mid\_enemies:  
 if each.active:  
 each.move()  
 screen.blit(each.image, each.rect)  
  
 # 绘制血槽  
 pygame.draw.line(screen, BLACK, (each.rect.left, each.rect.top - 5),  
 (each.rect.right, each.rect.top - 5), 2)  
 energy\_remain = each.energy / enemy.MidEnemy.energy  
 if energy\_remain > 0.2:  
 energy\_color = GREEN  
 else:  
 energy\_color = RED  
 pygame.draw.line(screen, energy\_color, (each.rect.left, each.rect.top - 5),  
 (each.rect.left + each.rect.width \* energy\_remain, each.rect.top - 5), 2)  
  
 else:  
 # 毁灭  
 # enemy3\_down\_sound.play()  
 if not (delay % 3):  
 # 绘制毁灭画面  
 screen.blit(each.destroy\_images[e3\_destroy\_index], each.rect)  
 e2\_destroy\_index = (e2\_destroy\_index + 1) % 3  
 if e2\_destroy\_index == 0:  
 score += 6000  
 each.reset()  
  
 # 绘制小型敌人  
 for each in small\_enemies:  
 if each.active:  
 each.move()  
 screen.blit(each.image, each.rect)  
 else:  
 # 毁灭  
 # enemy3\_down\_sound.play()  
 if not (delay % 3):  
 # 绘制毁灭画面  
 screen.blit(each.destroy\_images[e3\_destroy\_index], each.rect)  
 e1\_destroy\_index = (e1\_destroy\_index + 1) % 3  
 if e1\_destroy\_index == 0:  
 score += 1000  
 each.reset()  
  
 # 检测我方飞机是否被撞  
 anemies\_down = pygame.sprite.spritecollide(me, enemies, False, pygame.sprite.collide\_mask)  
 if anemies\_down and not me.invincible:  
 if is\_protected:  
 is\_protected = False  
 me.invincible = True  
 # 无敌状态计时  
 pygame.time.set\_timer(INVINCIBLE\_TIME, 3 \* 1000)  
 else:  
 me.active = False  
 for e in anemies\_down:  
 e.active = False  
  
 # 绘制CXK  
 if me.active:  
 if CXK\_switch\_image == 1:  
 screen.blit(me.image1, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image1)  
 if CXK\_switch\_image == 2:  
 screen.blit(me.image2, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image2)  
 if CXK\_switch\_image == 3:  
 screen.blit(me.image3, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image3)  
 if CXK\_switch\_image == 4:  
 screen.blit(me.image4, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image4)  
 if CXK\_switch\_image == 5:  
 screen.blit(me.image5, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image5)  
 if CXK\_switch\_image == 6:  
 screen.blit(me.image6, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image6)  
 if CXK\_switch\_image == 7:  
 screen.blit(me.image7, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image7)  
 if CXK\_switch\_image == 8:  
 screen.blit(me.image8, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image8)  
 if CXK\_switch\_image == 9:  
 screen.blit(me.image9, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image9)  
 if CXK\_switch\_image == 10:  
 screen.blit(me.image10, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image10)  
 if CXK\_switch\_image == 11:  
 screen.blit(me.image11, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image11)  
 if CXK\_switch\_image == 12:  
 screen.blit(me.image12, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image12)  
 if CXK\_switch\_image == 13:  
 screen.blit(me.image13, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image13)  
 if CXK\_switch\_image == 14:  
 screen.blit(me.image14, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image14)  
 if CXK\_switch\_image == 15:  
 screen.blit(me.image15, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image15)  
 if CXK\_switch\_image == 16:  
 screen.blit(me.image16, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image16)  
 if CXK\_switch\_image == 17:  
 screen.blit(me.image17, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image17)  
 if CXK\_switch\_image == 18:  
 screen.blit(me.image18, me.rect)  
 me.mask = pygame.mask.from\_surface(me.image18)  
 CXK\_switch\_image = 1  
 else:  
 if not (delay % 6):  
 CXK\_switch\_image += 1  
 else:  
 # 毁灭  
 if not (delay % 3):  
 if me\_destroy\_index == 0:  
 enemy3\_down\_sound.play()  
 screen.blit(me.destroy\_images[me\_destroy\_index], me.rect)  
 me\_destroy\_index = (me\_destroy\_index + 1) % 4  
 if me\_destroy\_index == 0:  
 life\_num -= 1  
 me.reset()  
 pygame.time.set\_timer(INVINCIBLE\_TIME, 3 \* 1000)  
  
 # 绘制护盾  
 if is\_protected:  
 screen.blit(shield\_image, (me.rect.left - 20, me.rect.top - 5))  
  
 if not delay:  
 delay = 100  
 else:  
 delay -= 1  
  
 # 剩余炸弹数量  
 bomb\_text = bomb\_font.render("X%d" % bomb\_num, True, BLACK)  
 text\_rect = bomb\_text.get\_rect()  
 screen.blit(bomb\_image, (10, height - 10 - bomb\_rect.height))  
 screen.blit(bomb\_text, (20 + bomb\_rect.width, height - 11 - text\_rect.height))  
  
 # 绘制剩余生命数量  
 if life\_num:  
 for i in range(life\_num):  
 screen.blit(life\_image, (width - 10 - (i + 1) \* life\_rect.width, height - 10 - life\_rect.height))  
  
 # 绘制得分  
 score\_text = score\_font.render("Score: %s" % str(score), True, BLACK)  
 screen.blit(score\_text, (10, 5))  
  
 # 绘制游戏结束画面  
 elif life\_num == 0:  
  
 # 背景音乐停止  
 pygame.mixer.music.stop()  
  
 # 停止全部音效  
 pygame.mixer.stop()  
  
 # 停止发放补给  
 pygame.time.set\_timer(SUPPLY\_TIME, 0)  
  
 if not recorded:  
 recorded = True  
 # 读取历史最高得分记录  
 with open("record.txt", "r") as f:  
 record\_score = int(f.read())  
 if score > record\_score:  
 record\_score = score  
 with open("record.txt", "w") as f:  
 f.write(str(score))  
  
 # 绘制结束界面  
 record\_score\_text = score\_font.render("Best:%d" % record\_score, True, BLACK)  
 screen.blit(record\_score\_text, (50, 50))  
  
 gameover\_text1 = gameover\_font.render("Your Score", True, BLACK)  
 gameover\_text1\_rect = gameover\_text1.get\_rect()  
 gameover\_text1\_rect.left, gameover\_text1\_rect.top = (width - gameover\_text1\_rect.width) // 2, height - 500  
 screen.blit(gameover\_text1, gameover\_text1\_rect)  
  
 gameover\_text2 = gameover\_font.render(str(score), True, BLACK)  
 gameover\_text2\_rect = gameover\_text2.get\_rect()  
 gameover\_text2\_rect.left, gameover\_text2\_rect.top = (  
 width - gameover\_text2\_rect.width) // 2, gameover\_text1\_rect.bottom + 10  
 screen.blit(gameover\_text2, gameover\_text2\_rect)  
  
 again\_rect.left, again\_rect.top = (width - again\_rect.width) // 2, gameover\_text2\_rect.bottom + 50  
 screen.blit(again\_image, again\_rect)  
  
 gameover\_rect.left, gameover\_rect.top = (width - again\_rect.width) // 2, again\_rect.bottom + 10  
 screen.blit(gameover\_image, gameover\_rect)  
  
 # 检测用户的鼠标操作  
 # 如果用户按下鼠标左键  
 if pygame.mouse.get\_pressed()[0]:  
 # 获取鼠标坐标  
 pos = pygame.mouse.get\_pos()  
 # 如果用户点击”重新开始“  
 if again\_rect.left < pos[0] < again\_rect.right and again\_rect.top < pos[1] < again\_rect.bottom:  
 main()  
 # 如果用户点击”结束游戏“  
 if gameover\_rect.left < pos[0] < gameover\_rect.right and gameover\_rect.top < pos[  
 1] < gameover\_rect.bottom:  
 pygame.quit()  
 sys.exit()  
  
 # 绘制暂停按钮  
 screen.blit(paused\_image, pause\_rect)  
  
 pygame.display.flip()  
  
 clock.tick(60)

# 参考文献

[1] Eric Matthes.Python 编程从入门到实践[M].北京：人民邮电出版社，2016.

[2] Magnus Lie Hetland. Python基础教程(第3版)[M].北京：人民邮电出版社，2018.

[3]小甲鱼.Python零基础入门学习[M].北京：清华大学出版社，2016.

[4]黑马程序员.Python快速编程入门[M].北京：人民邮电出版社，2017.

[5]董付国.Python程序设计（第2版）[M].北京：清华大学出版社，2016.

[6]嵩天，礼欣，黄天羽.Python语言程序设计基础（第2版）[M].北京：高等教育出版社，2017.

[7] Wesley Chun.Python 核心编程[M].北京：人民邮电出版社，2016.