飞机大战项目总结

小组成员：杨立伦、姜裕、王文慧、
黄思雨、王盼、吴凌霄、徐睿

汇报人　：杨立伦

目录

Content

01

第一章　项目需求

项目需求

目标

实现飞机大战游戏，添加开始和结束界面，并且增加功能

游戏需求

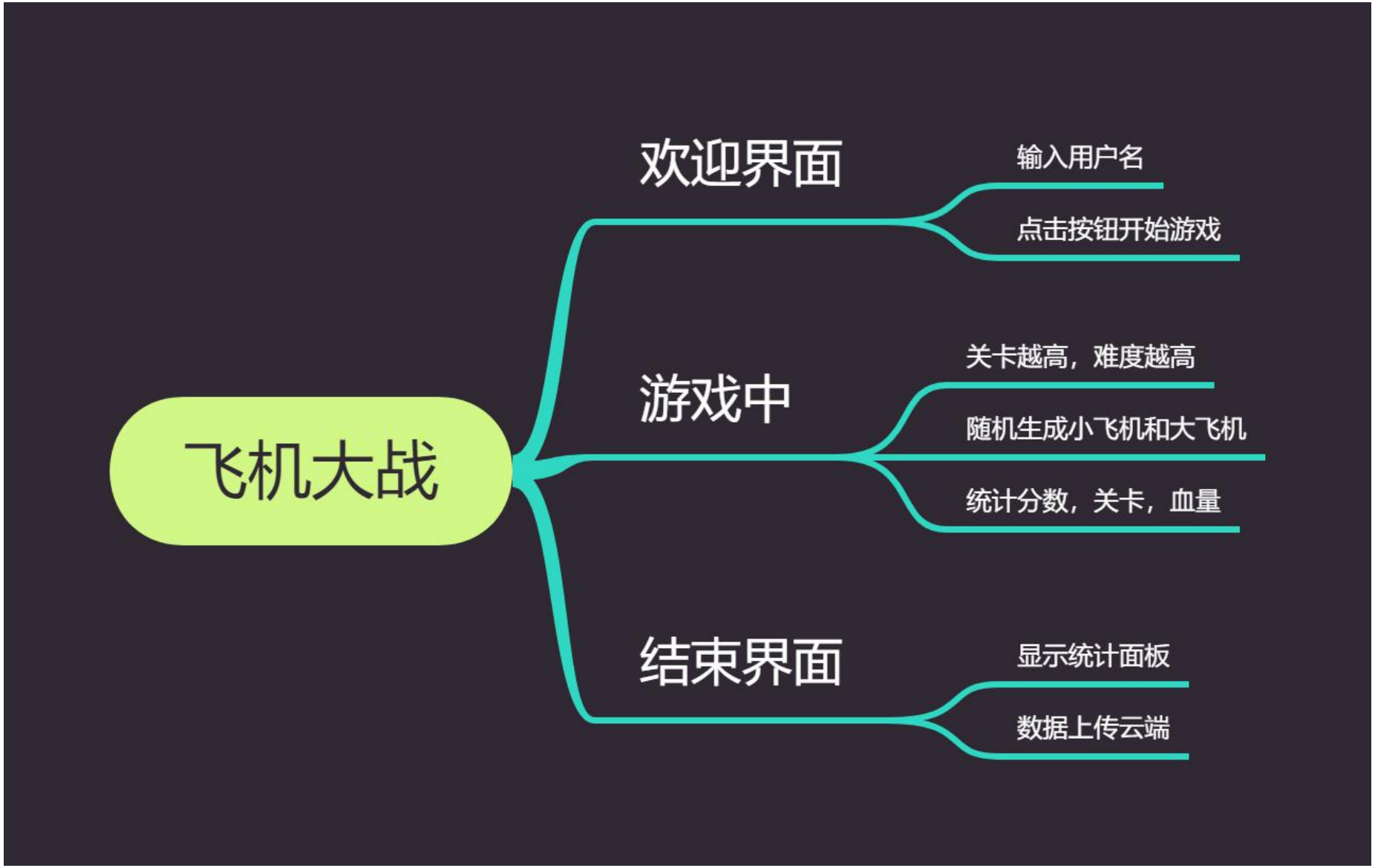
玩家输入用户名进入游戏开始界面，点击开始游戏进入第一关，开始通关。玩家控制一台英雄飞机，空格键发射子弹，通过上下左右移动，击毁敌机，全部击败进入下一关，难度升级；如碰到敌机则减少一次生命值，共有两次复活机会，直至复活机会用完，游戏结束。

游戏记录

游戏后台记录使用java编写，通过http协议与python开发的飞机大战游戏进行数据通信。

数据库使用MySQL，在游戏结束时会保存玩家当前分数

项目需求


飞机大战
欢迎界面
输入用户名
点击按钮开始游戏
游戏中
关卡越高，难度越高
随机生成小飞机和大飞机
统计分数，关卡，血量
结束界面
显示统计面板
数据上传云端

02

第二章　环境准备

环境准备

软件名称	描述
python3.8 编程语言	开发游戏主语言
pygame游戏库	使用到第三方游戏库增加开发效率
requests请求库	发送http请求保存，读取游戏记录
pycharm	集成开发环境
adobe audition	音频编辑，用于编辑游戏音效
adobe photoshop	图片编辑，用户编辑游戏素材
jdk1.8	用于编写游戏数据存储后台
idea	java集成开发软件
spring-boot	后台开发框架
mysql5.8	存储玩家记录数据库
navicat	数据库管理软件

03

第三章　详细设计

详细设计 协同合作

杨立伦。负责策划游戏，协调工作，
编写核心模块，对接模块

黄思雨。记录开发过程，总结问题，
编写PPT文案

姜裕，徐睿。开发人员，编写部分
模块，参与调试

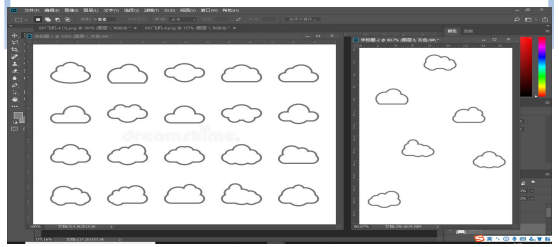
王盼，美工。负责对游戏素材制作

王文慧，吴凌霄 游戏测试人员，
负责测试游戏bug，并及时反馈

详细设计 设计分块

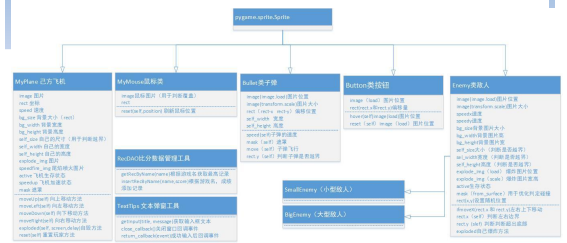
素材设计

对可能会使用的素材文件进行创作，编辑



模块设计

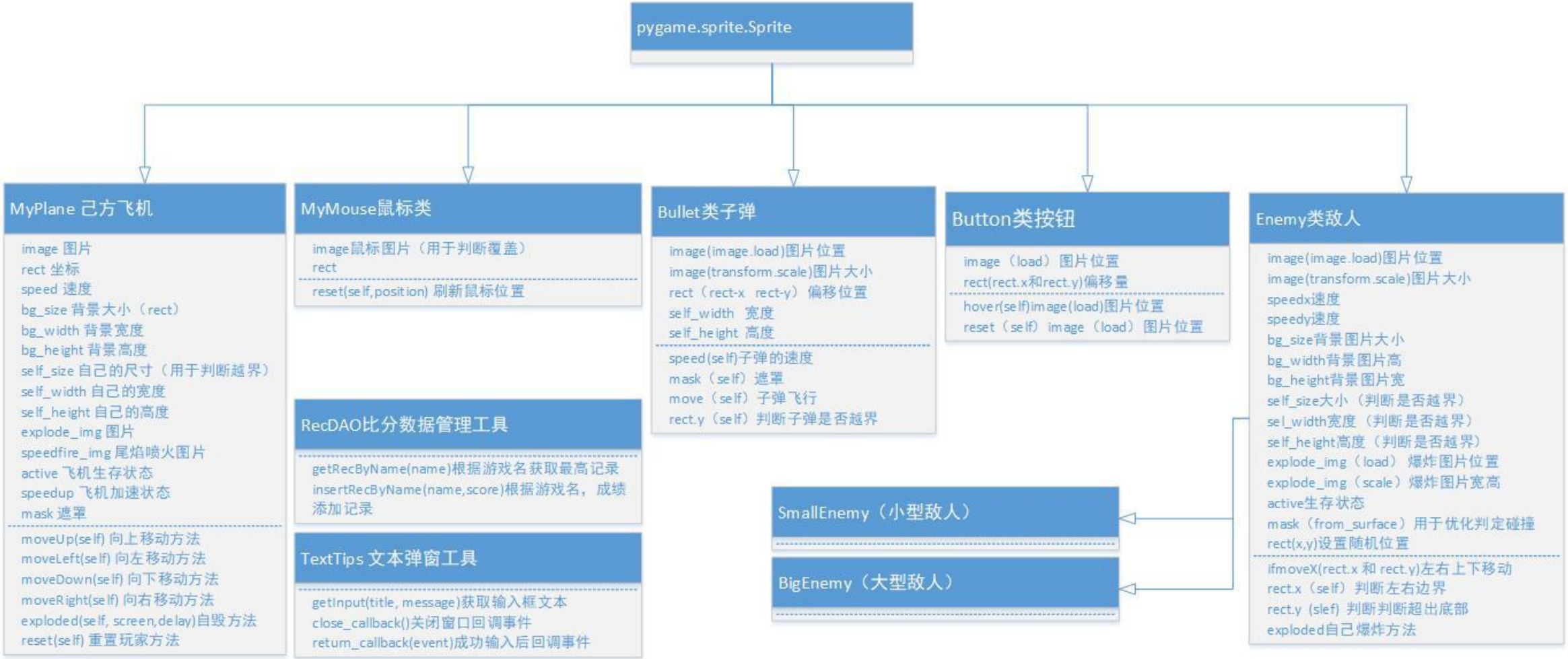
对系统模块结构，继承关系进行详细设计

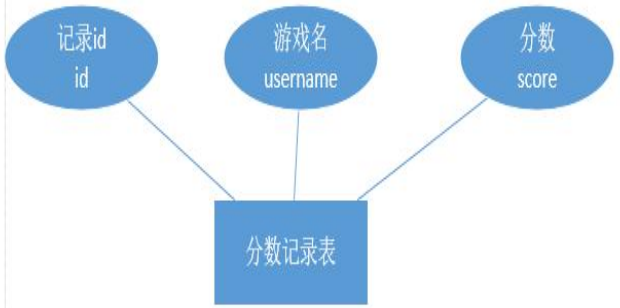


数据库设计

对数据库表进行设计

详细设计 UML类图


pygame.sprite.Sprite
MyPlane 己方飞机
image 图片
rect 坐标
speed 速度
bg_size 背景大小（rect）
bg_width 背景宽度
bg_height 背景高度
self_size 自己的尺寸（用于判断越界）
self_width 自己的宽度
self_height 自己的高度
explode_img 图片
speedfire_img 尾焰喷火图片
active 飞机生存状态
speedup 飞机加速状态
mask 遮罩
moveUp(self) 向上移动方法
moveLeft(self) 向左移动方法
moveDown(self) 向下移动方法
moveRight(self) 向右移动方法
exploded(self, screen,delay)自毁方法
reset(self) 重置玩家方法
MyMouse鼠标类
image鼠标图片（用于判断覆盖）
rect
reset(self,position) 刷新鼠标位置
RecDAO比分数据管理工具
getRecByName(name)根据游戏名获取最高记录
insertRecByName(name,score)根据游戏名，成绩添加记录
TextTips 文本弹窗工具
getInput(title, message)获取输入框文本
close_callback()关闭窗口回调事件
return_callback(event)成功输入后回调事件
Bullet类子弹
image(image.load)图片位置
image(transform.scale)图片大小
rect（rect-x rect-y）偏移位置
self_width 宽度
self_height 高度
speed(self)子弹的速度
mask（self）遮罩
move（self）子弹飞行
rect.y（self）判断子弹是否越界
Button类按钮
image（load）图片位置
rect(rect.x和rect.y)偏移量
hover(self)image(load)图片位置
reset（self）image（load）图片位置
SmallEnemy（小型敌人）
BigEnemy（大型敌人）
Enemy类敌人
image(image.load)图片位置
image(transform.scale)图片大小
speedx速度
speedy速度
bg_size背景图片大小
bg_width背景图片高
bg_height背景图片宽
self_size大小（判断是否越界）
sel_width宽度（判断是否越界）
self_height高度（判断是否越界）
explode_img（load）爆炸图片位置
explode_img（scale）爆炸图片宽高
active生存状态
mask（from_surface）用于优化判定碰撞
rect(x,y)设置随机位置
ifmoveX(rect.x 和 rect.y)左右上下移动
rect.x（self）判断左右边界
rect.y（slef）判断判断超出底部
exploded自己爆炸方法



分数记录表字段

后台接口设计

查询所有记录

- 请求路径: `getAllRec`
- 请求类型: `post`

参数:

字段	类型	说明	备注
无参数			

返回值:

字段	类型	说明	备注
id	int	编号	无
username	String	游戏名	无
score	int	分数值	无

示例json:

```
[
        {
        "id":编号,
        "username":"游戏名",
        "score":分数值
        }
]
```

根据游戏名查询最高分数

- 请求路径: `getRecByUsername`
- 请求类型: `post`

参数:

字段	类型	说明	备注
username	String	s	游戏名

返回值:

字段	类型	说明	备注
id	int	编号	无
username	String	游戏名	无
score	int	分数值	无

示例json:

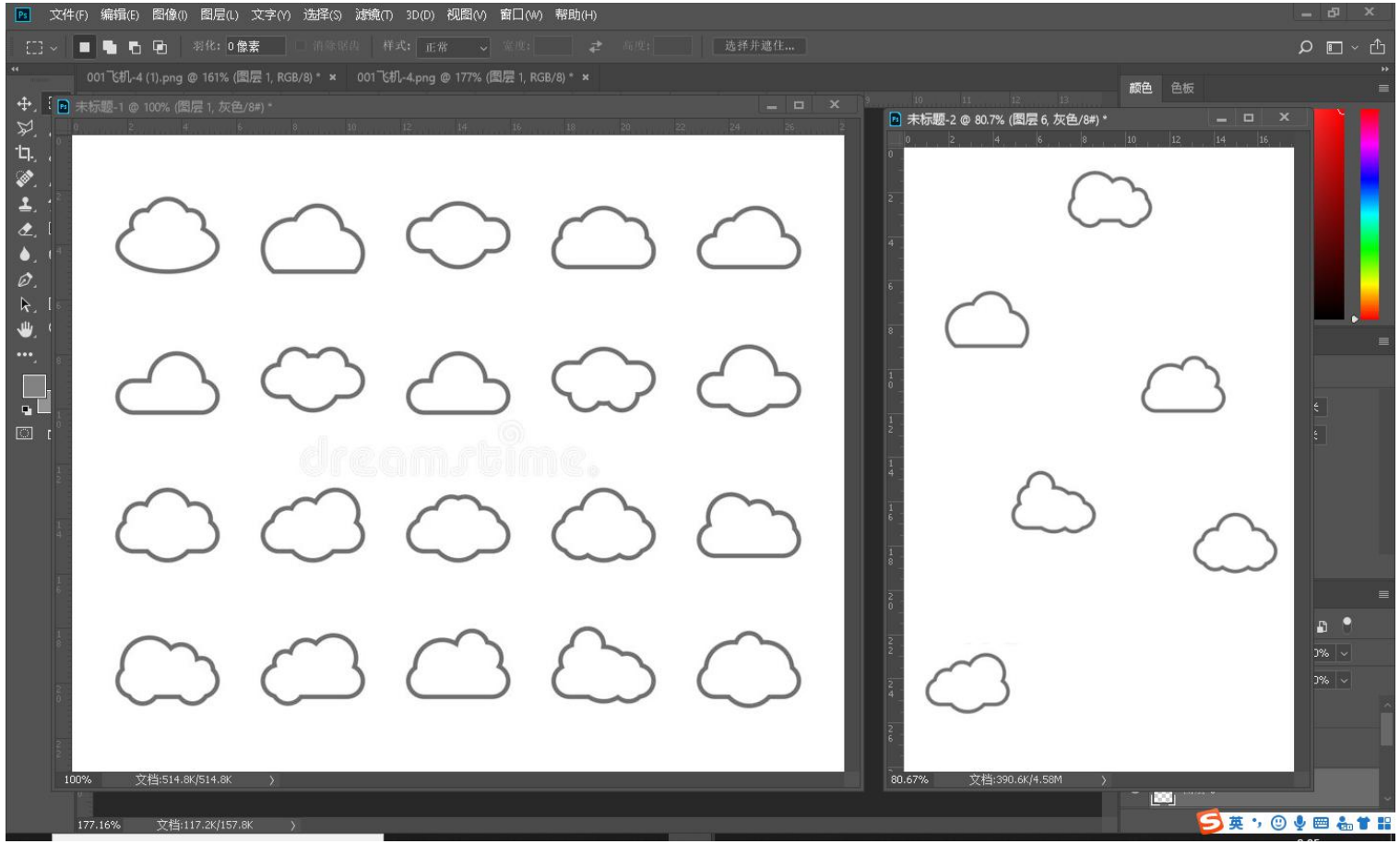
```
{
"id":编号,
"username":"游戏名",
"score":分数值
}
```

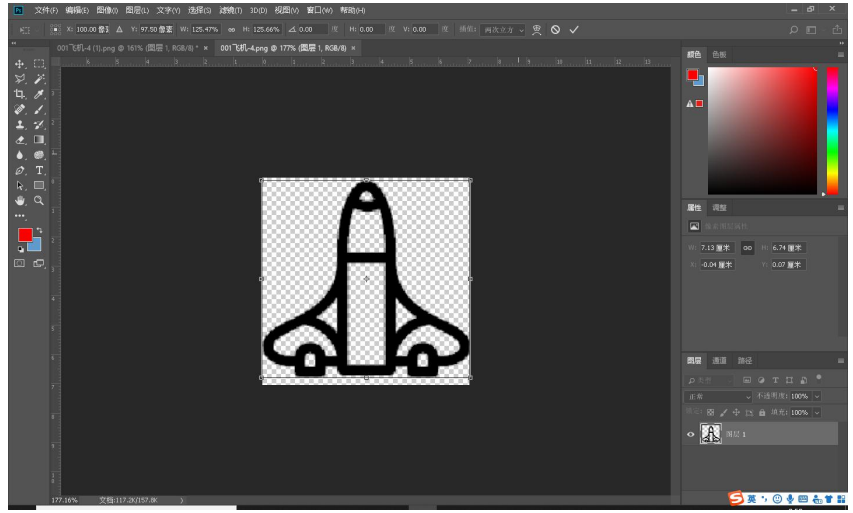
04

第四章　实现过程

实现过程

制作背景和图标





实现过程

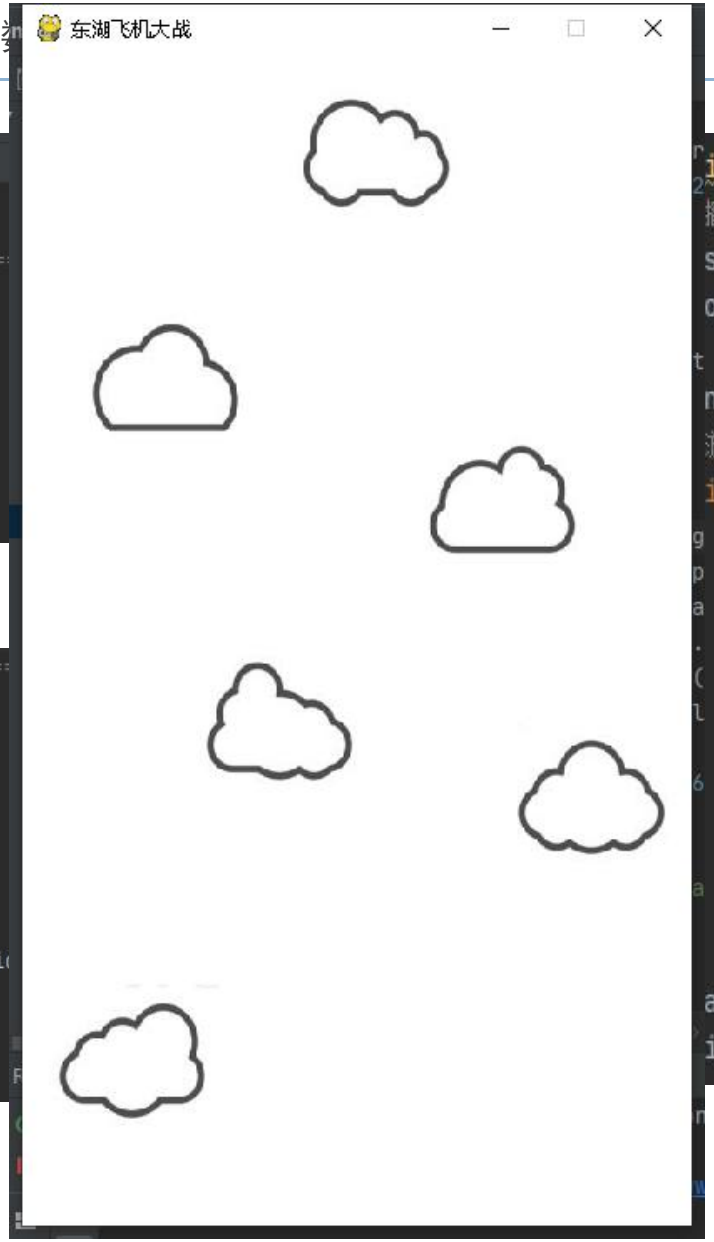
制作主函数

```python
import pygame
import sys
# 配置设置==========
# 游戏宽度高度
width = 400
height = 700
# 背景图片地址
background_url = './static/background.png'
background_music_url = './static/Highscore.wav'
# 游戏标题
game_title = '东湖飞机大战'
```

main配置变量

```python
# 初始化==========
pygame.init()
# 混音器初始化
pygame.mixer.init()
# 设置画布的宽高
screen = pygame.display.set_mode((width,height))
pygame.display.set_caption(game_title)
# 加载背景
background = pygame.image.load(background_url)
background = pygame.transform.scale(background,(wi
# 加载背景音乐
music = pygame.mixer.Sound(background_music_url)
music.set_volume(0.2)
```

main游戏初始化




```python
in():
播放背景音乐
sic.play(-1)
ock = pygame.time.Clock()

nning = True
游戏循环
ile running:
  for e in pygame.event.get():
      if e.type == pygame.QUIT:
          pygame.quit()
          sys.exit()
  screen.blit(background,(0,0))
  pygame.display.flip()
  # 时钟控制
  clock.tick(60)


ame__ == '__main__':
in()
```

main主函数

实现过程

编写己方飞机类

```python
class MyPlane(pygame.sprite.Sprite):
    # 己方飞机构造
    def __init__(self,image,self_size,bg_size,point,speed):
        pygame.sprite.Sprite.__init__(self)
        # 加载图片，并且缩放到指定大小
        self.image = pygame.image.load(image)
        self.image = pygame.transform.scale(self.image,(self_size[0],self_size[1]))
        # 设置坐标，以及x，y位置
        self.rect = self.image.get_rect()
        self.rect.x = point[0]
        self.rect.x = point[1]
        # 设置速度
        self.speed = speed
        # 保存背景的大小，另存为宽度和高度
        self.bg_size = bg_size
        self.bg_width = bg_size[0]
        self.bg_height = bg_size[1]
        # 保存自己的大小，用于判断越界
        self.self_size = self_size
```

myplane构造方法

```python
    # 上移动方法
    def moveUp(self):
        if self.rect.y <= 0:
            self.rect.y = 0
        else:
            self.rect.y -= self.speed

    # 下移动方法
    def moveDown(self):
        if self.rect.y >= self.bg_height - self.self_size[1]:
            self.rect.y = self.bg_height - self.self_size[1]
        else:
            self.rect.y += self.speed

    # 左移动方法
    def moveLeft(self):
        if self.rect.x <= 0:
            self.rect.x = 0
        else:
            self.rect.x -= self.speed
    # 右移动方法
    def moveRight(self):
        if self.rect.x >= self.bg_width - self.self_size[0]:
            self.rect.x = self.bg_width - self.self_size[0]
        else:
```

myplane移动方法

实现过程

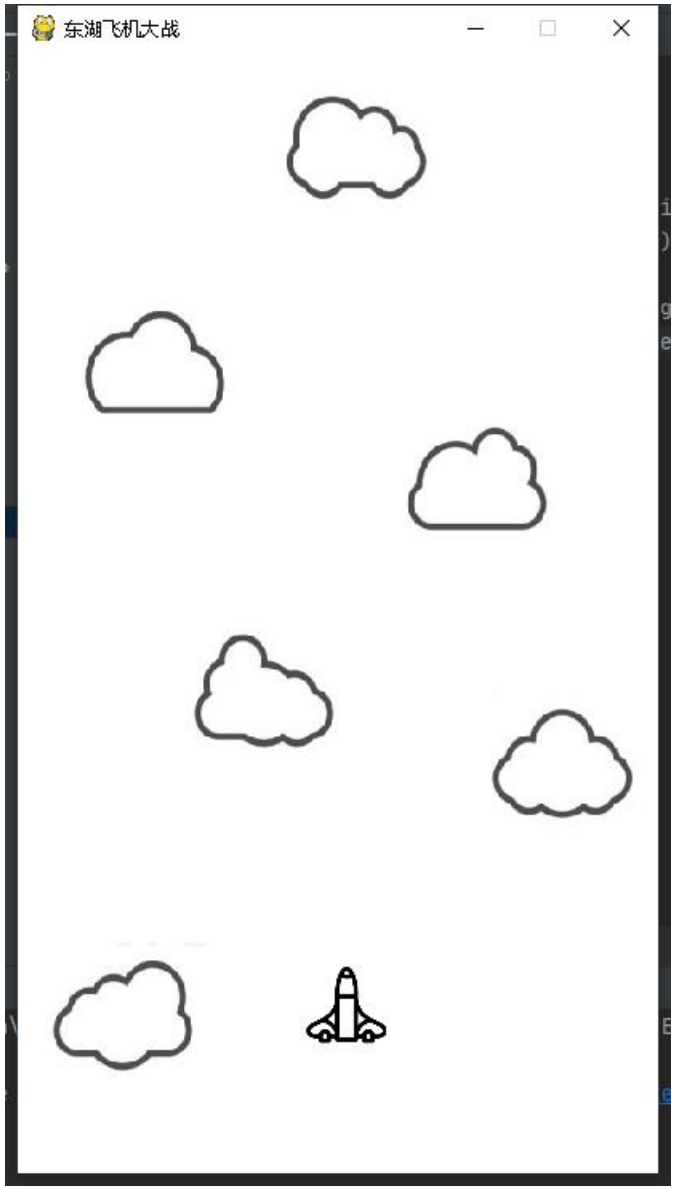
在main中添加己方飞机

```python
def main():
    # 播放背景音乐
    music.play(-1)
    clock = pygame.time.Clock()

    # 生成我方飞机
    myplane = MyPlane.MyPlane(myplane_img_url,(50,50),(width,height),(0,0),10)
```

在mian中绘制我方飞机

```python
    running = True
    # 游戏循环
    while running:
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        # 监听键盘事件
        key_press = pygame.key.get_pressed()
        if key_press[pygame.K_UP]:
            myplane.moveUp()
        if key_press[pygame.K_DOWN]:
            myplane.moveDown()
        if key_press[pygame.K_LEFT]:
            myplane.moveLeft()
        if key_press[pygame.K_RIGHT]:
            myplane.moveRight()
```

在main中添加监听事件，移动己方飞机



实现过程

编写敌方飞机类Enemy

Enemy.py MyPlane.py main.py

```python
import pygame
import random

class Enemy(pygame.sprite.Sprite):
    def __init__(self,image,self_size,bg_size,speed,speedy):
        pygame.sprite.Sprite.__init__(self)
        # 加载图片，并且缩放到指定大小
        self.image = pygame.image.load(image)
        self.image = pygame.transform.scale(self.image, (self_size[0], self_size[1]))
        # 设置速度
        self.speedx = speed
        self.speedy = speedy
        # 保存背景的大小，另存为宽度和高度
        self.bg_size = bg_size
        self.bg_width = bg_size[0]
        self.bg_height = bg_size[1]
        # 保存自己的大小，用于判断越界
        self.self_size = self_size
        self.self_width = self_size[0]
        self.self_height = self_size[1]
        # 设置随机位置
        self.rect = self.image.get_rect()
        self.rect.x = random.randint(0,self.bg_width-self.self_width)
        self.rect.y = 0
```

编写敌方小飞机类

```python
# 敌机移动事件
def move(self):
    # 左右上下移动
    self.rect.x += self.speedx
    self.rect.y += self.speedy
    # 判断左右碰到边界
    if self.rect.x >= self.bg_width - self.self_width:
        self.speedx *= -1
    if self.rect.x <= 0:
        self.speedx *= -1
    # 判断超出最底部
    if self.rect.y >= self.bg_height:
        self.reset()
```

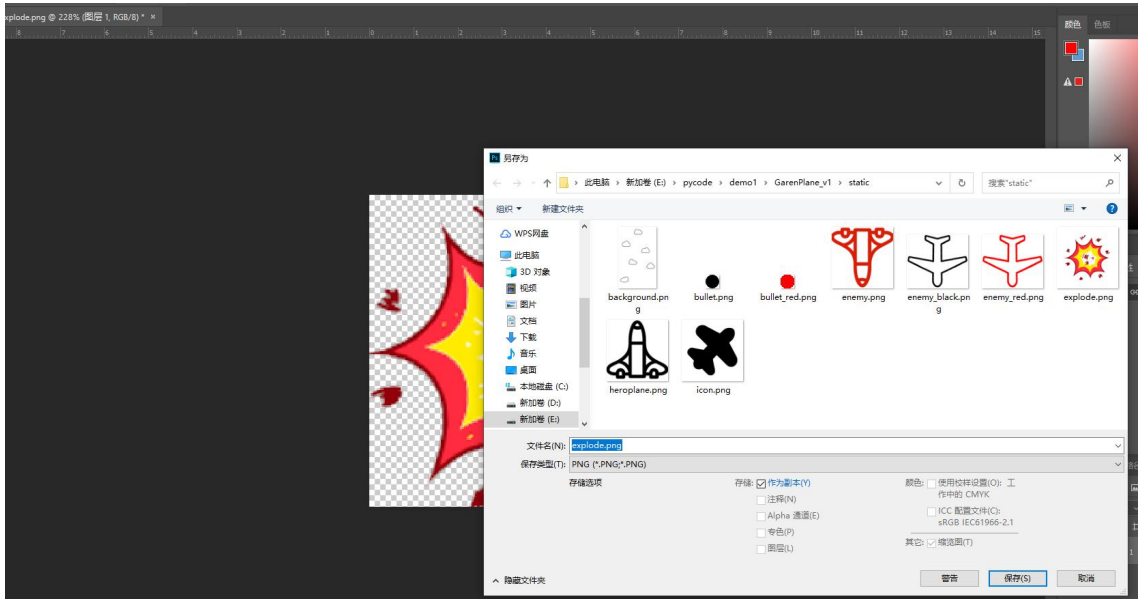
敌方小飞机移动模块

```python
# 添加n架小飞机
def add_small_enemy(small_enemies,enemys_all,num):
    for i in range(num):
        temp_enemy = Enemy.Enemy(enemy_img_url, (50, 50), (width, height), 3, 2,False)
        small_enemies.add(temp_enemy)
        enemys_all.add(temp_enemy)
```

封装添加敌机方法

实现过程

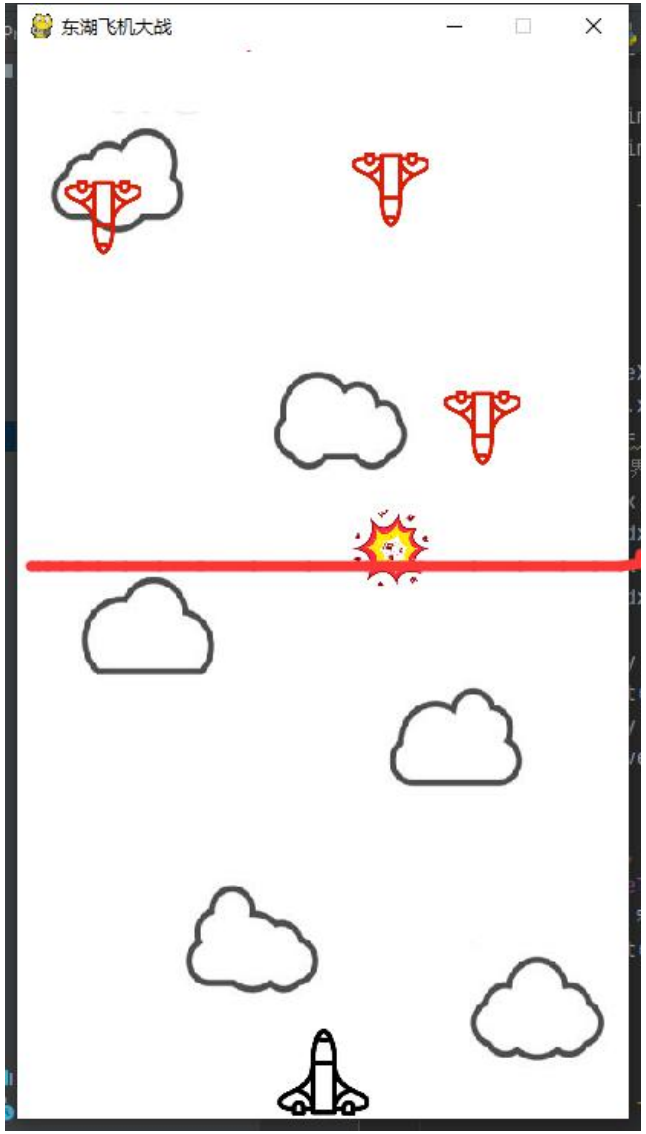
制作爆炸效果



制作爆炸效果素材

```
46          if self.rect.y >= self.bg_height:
47              self.reset()
48          if self.rect.y >= 300:
49              self.active = False
50
51      # 自己爆炸方法
```

设置一条线，碰到则爆炸（debug）



效果

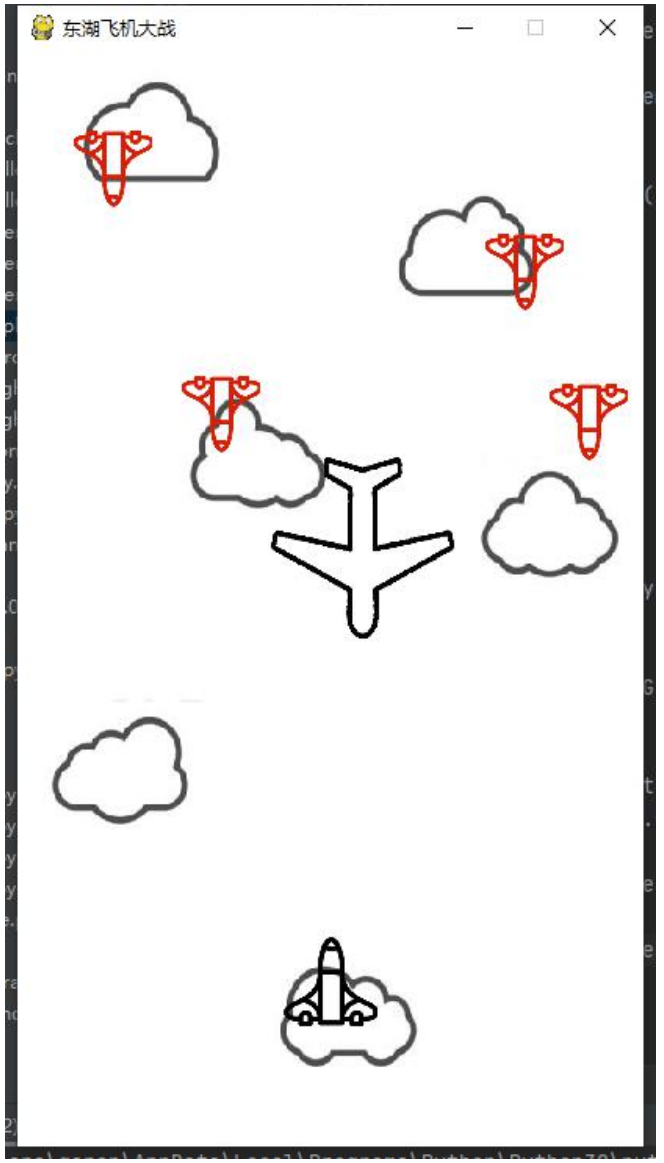
实现过程


```python
# 添加n架大飞机
def add_big_enemy(enemys_all,num,speed,ifmove,movespeed):
    for i in range(num):
        temp_enemy = Enemy.BigEnemy(bigenemy_img_url, (120, 120), (width, height), movespeed, speed, ifmove)
        enemys_all.add(temp_enemy)
```

生成大飞机

```python
    # 更新飞机
    # 己方
    screen.blit(myplane.image, myplane.rect)
    # 小飞机
    for temp_enemy in small_enemies:
        if temp_enemy.active:
            screen.blit(temp_enemy.image, temp_enemy.rect)
            temp_enemy.move()
        else:
            temp_enemy.exploded(screen, delay)
    # 大飞机
    for temp_enemy in big_enemies:
        if temp_enemy.active:
            screen.blit(temp_enemy.image, temp_enemy.rect)
            temp_enemy.move()
        else:
            temp_enemy.exploded(screen,delay)
```

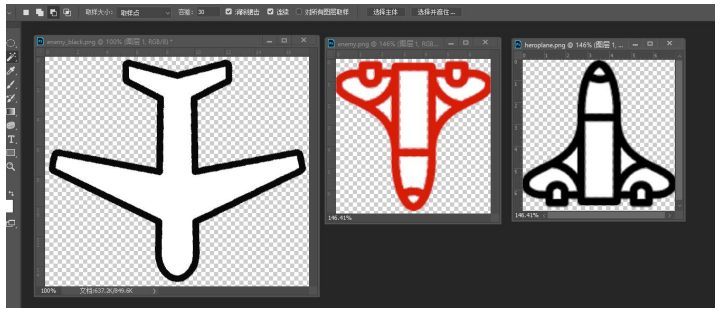
更新所有飞机



效果

实现过程

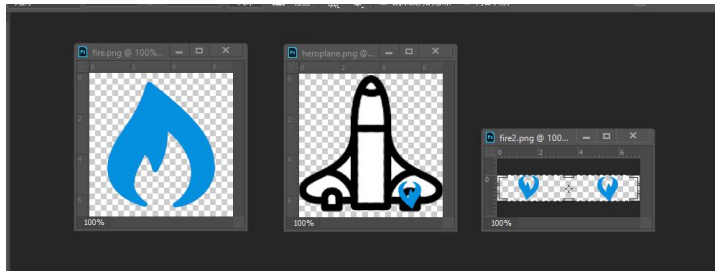
额外功能：前进加速喷火



对飞机素材进行填充

```
35
36      # 上移动方法
37      def moveUp(self):
38          if self.rect.y <= 0:
39              self.rect.y = 0
40          else:
41              self.rect.y -= self.speed
42              self.speedup = True
43
```

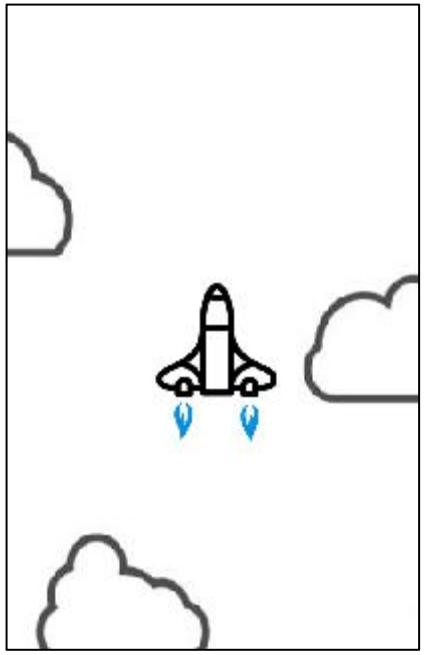
如果在按上，则加速状态为true



制作加速效果图

```
# 更新飞机
# 己方
screen.blit(myplane.image, myplane.rect)
# 如果在加速
if myplane.speedup:
    screen.blit(myplane.speedfire_img, (myplane.rect.x,myplane.rect.y+myplane.self_height))

# 小飞机
```

如果在加速，喷火

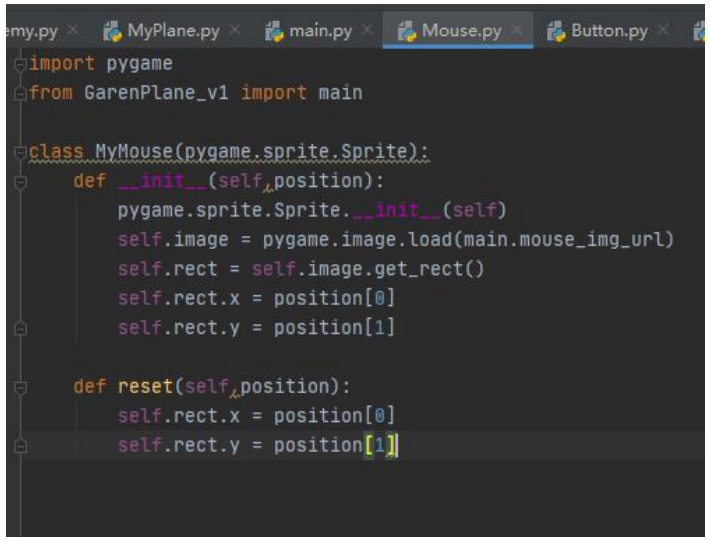


效果

实现过程

进场界面制作

制作进场界面



```python
import pygame
from GarenPlane_v1 import main

class MyMouse(pygame.sprite.Sprite):
    def __init__(self,position):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(main.mouse_img_url)
        self.rect = self.image.get_rect()
        self.rect.x = position[0]
        self.rect.y = position[1]

    def reset(self,position):
        self.rect.x = position[0]
        self.rect.y = position[1]
```

鼠标精灵，用于判断点击按钮精灵

实现过程

记录游戏分数

```
# 输入游戏名
username = TextTips.getInput('东湖飞机大战','请输入游戏名')
print(username)
```

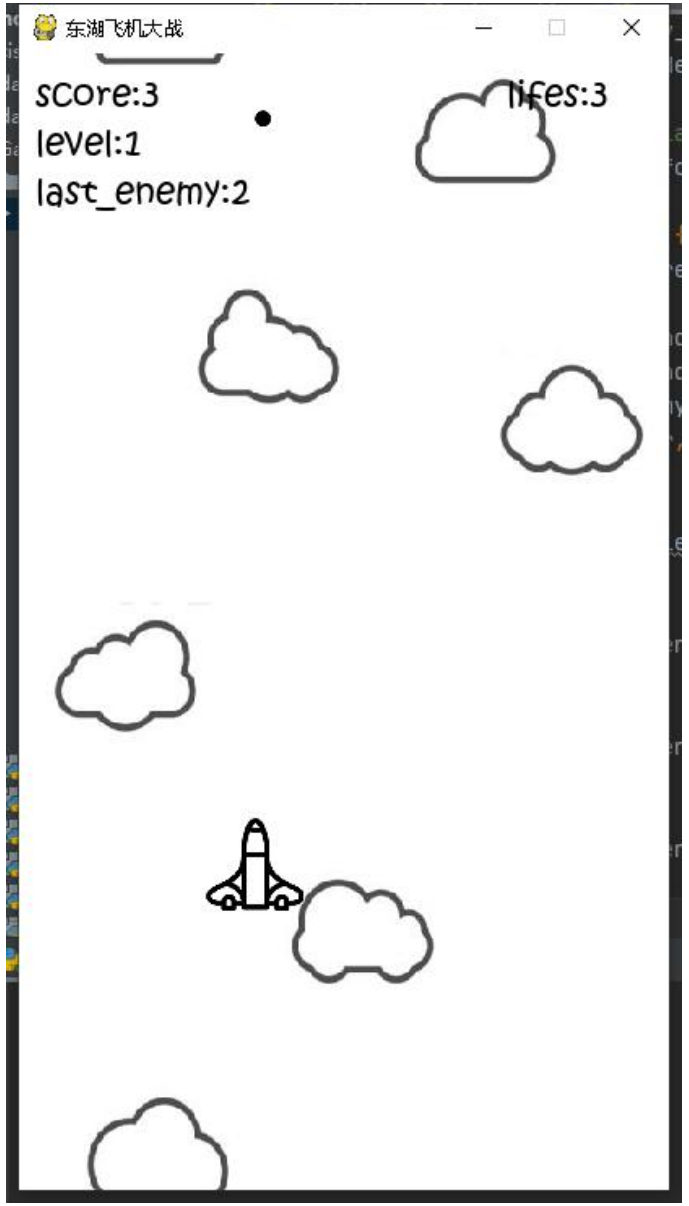
输入游戏名，用于存储数据

```
# 分数
score = 0
# 生命
my_hp = 3
# 剩余敌军
last_enemy = 0;
```

添加统计信息

```
# 显示积分面板
def showscore():
    global last_enemy,level

    score_text = f"score:{score}"
    score_render = font.render(score_text,True,(0,0,0))

    hp_text = f"lifes:{my_hp}"
    hp_render = font.render(hp_text, True, (0, 0, 0))

    last_enemy_text = f"last_enemy:{last_enemy}"
    last_enemy_render = font.render(last_enemy_text, True, (0, 0, 0))

    level_text = f"level:{level}"
    level_render = font.render(level_text, True, (0, 0, 0))

    screen.blit(score_render,(10,10))
    screen.blit(level_render, (10, 40))
    screen.blit(last_enemy_render, (10, 70))
    screen.blit(hp_render, (width-100,10))
```

显示统计信息面板



实现过程

制作结束界面

制作结束界面

```
# 结束界面==========================================================================
end_img_url = './static/background_end.png'
end_img = pygame.image.load(end_img_url)
end_img = pygame.transform.scale(end_img,(width,height))
```

编写结束界面背景

```
# 显示结束统计
def showendscore():
    global last_enemy, level,score,username,high_score

    score_text = f"score:{score}"
    score_render = font2.render(score_text, True, (0, 0, 0))

    highscore_text = f"high-score:{high_score}"
    highscore_render = font2.render(highscore_text, True, (0, 0, 0))

    name_text = f"name:{username}"
    name_render = font2.render(name_text, True, (0, 0, 0))

    screen.blit(score_render, (50,350))
    screen.blit(highscore_render, (50, 390))
    screen.blit(name_render, (50, 430))
```

编写结束统计面板

实现过程

编写分数

```java
//    根据游戏名查询最高分数
    @RequestMapping("/getRecByUsername")
    @ResponseBody
    public Recording getRecByName(String username){
        List<Recording> recordings = recordingDao.queryAll(new Recording
        Recording Rec_max = null;
        int score_max = 0;
        for(Recording r : recordings){
            if(r.getScore() > score_max){
                Rec_max = r;
                score_max = r.getScore();
            }
        }
        return Rec_max;
    }

//    添加分数记录
    @RequestMapping("/addRec")
    @ResponseBody
    public boolean addRec(Recording recording) { return recordingDao.ins
}
```

java编写记录服务器


```
取最高纪录
Name(name):
ttp://127.0.0.1:28081/getRecByUsername'
= requests.post(url,{"username":name})
encoding = 'utf-8'
on.loads(request.text)
bj.get('score')


成绩添加记录
cByName(name,score):
ttp://127.0.0.1:28081/addRec'
= requests.post(url,{"username":name,"score":score})
encoding = 'utf-8'
equest.text == 'true'
```

编写python存储数据方法

```
350         # 获取历史最高分
351         high_score = RecDao.getRecByName(us
352         print(high_score)
353
354         # 上传当前成绩
355         if RecDao.insertRecByName(username,
356             print(f'上传成绩成功:{username} -
357
```

取,

05

第五章 总结

总结

想要了解整个游戏，就得从0开始制作整个游戏。走完整个流程下来，收获了很多。首先是团队合作的力量。一起学习，分析老师的代码，仿照着写一些测试，成功了之后互相分享知识点。
其次是面向对象的思维，最早的版本是全过程编写，发现很麻烦，碰撞也不好检测。没办法只好用sprite重写，这个时候就体会到了使用别人库的优点了。

学习Python的这短时间以来，觉得Python还是比较简单，容易上手的，就基本语法而言，但是有些高级特性掌握起来还是有些难度，需要时间去消化。
Python给我最大的印象就是语法简洁，就像写伪代码一样，很多其他语言要用很多行才能实现的操作Python可能几行就搞定了
其次就是python运行的方便，不需要编译，并且打包很方便，安装第三方库也很方便，自带pip，在以后的学习中我还会继续学习python~

谢谢观看

THANK YOU FOR WATCHING

汇 报 人 ： 杨 立 伦