**南京理工大学(江阴校区)**

**课程大作业**

**（团队总结）**

**课程名称： 大数据分析**

**班级： 网安**

**学生姓名： 王韬/王时青**

**团队互评成绩（排名）： /**

**总成绩：**

|  |
| --- |
| **任课教师评语：**  **签名：**  **年 月** |

# 一、课程大作业完成的基本情况（至少500字）

利用Python课件（武装飞船、外星人、评分）和相关理论与方法，设计和完成一个“外星人入侵”的游戏项目，项目组成包括：游戏中包括武装飞船、外星人和评分，使得游戏有趣、内容丰富、界面友好，并撰写实验报告，实验报告内容包括：项目规划、完整的代码和注释（涵盖这款游戏的所有细节），解释和说明等。

每个小组制作PPT，汇报本小组所制作游戏项目的创新性、有趣性和丰富度。

**对本小组实验的项目规划和实验创新之处进行归纳和总结如下：**

## 项目规划

1. 项目背景

作为一门高效的解释性语言，Python凭借其易学易懂的语法特性，以及简洁明了的动态语言特性，成为了多数平台上写脚本和快速开发应用的编程语言。为了在实践中更好的学习python与应用python，我们小组编写了《入侵外星人》这个游戏。

1. 项目简介

《外星人入侵》是一款俯视角rougelike游戏，在游戏中，玩家采用鼠标为太空飞船标定航行与射击方向，并采取“加速”与“射击”两种操作应对袭来的外星飞船，不断通过击败外星飞船获取物资强化飞船，并最终逃出生天

1. 项目步骤：
2. 在游戏开发的第一个阶段，我们通过建模得出飞船的外形与实际体积，并创建一艘飞船，该飞船将随着玩家鼠标的指向改变方向，并拥有加速和射击两种功能
3. 在游戏开发的第二个阶段，我们建模得出敌方外星飞船的外形与实际体积，并为其通过随机设定出现时机与轨迹，且为其安排射击功能。
4. 在游戏开发的第三个阶段，我们构建敌方外星飞船和我方飞船之间的交互，通过“射击”与“碰撞”两种功能，设计友方，敌方之间的交互性，并引入血量概念
5. 在游戏开发的第四个阶段，我们构建物资与等级概念——友方飞船可以通过射击敌方飞船获取物资并升级，并获取血量与攻击上的强化，同时通过随机给出的选项，得到额外多样的强化
6. 在游戏开发的第五个阶段，敌人不再只有一种——他们会随着友方飞船的等级增长而增长。地图刷新出的不止有一种的敌人，更有不同等级与不同种类的敌人
7. 在游戏开发的第六个阶段，我们完善游戏，通过物理引擎，动态星空背景的方式对太空进行仿真，并增加了背景音乐，音效等功能
8. 在游戏开发的第七个阶段，我们对游戏进行平衡性测试，使得游戏更具有可玩性。
9. 在游戏开发的第八个阶段，我们对游戏进行打包，使其可以在不同系统中实现功能

## 创新之处（包括有趣性和丰富度）

我们研发的《外星人入侵》是一款俯视角rougelike游戏，在游戏中，玩家采用鼠标为太空飞船标定航行与射击方向，并采取“加速”与“射击”两种操作应对袭来的外星飞船，不断通过击败外星飞船获取物资强化飞船，并最终逃出生天。我们通过对python语言和游戏程序的了解，对传统的《雷霆战机》式游戏进行了改造，具体增加了如下创新，极大的增加了游戏的有趣性与丰富度：

1. 为了贴合游戏的星空背景，放弃使用静态贴图，采用动态星空贴图，通过算法在一定规则下生成亮度不同的星星，并在通过星星在飞船移动下不同速率的位移达成近乎立体的感官。
2. 为了更好的仿真飞船发射与航行的过程，固定炮塔在飞船前段且引擎在飞船后端，通过鼠标作为导向，从而达成更好的仿真模拟与更高的可玩性
3. 为了更好贴合游戏的科幻背景与星战风格，在尝试过多种科幻贴图后，发现与游戏的极简基调星空背景不符，且存在放大后失真现象，故采用无贴图式几何建模，更好的与科幻背景相融合并减少在不同大小，不同分辨率背景下的失真情况。
4. 察觉到在固定游戏关卡下，存在在多次游玩下对固定敌人，固定关卡与固定友方产生乏味，进而失去可玩性。故而采用rougelike的游戏方式，通过随机性使得每次游戏产生不一样的游戏情况，进而增加游戏在多次重复下的耐玩程度
5. 添加物理引擎。为了拟真宇宙飞船在星空中的真实举动，通过添加物理引擎的方式模拟了敌军飞船与我军飞船在被子弹击中时的击退效果与发射子弹时的后坐力因素。添加了物理引擎的仿真因素。增加了游戏的仿真性与趣味性
6. 仿真飞船加速，在飞船加速过程中，通过物理引擎改变飞船速度，并非简单控制速度，而是通过加速度——速度的形势控制速度，加大游戏的可玩性与趣味性。

# 二、实验的过程和主要数据记录

## 1.武装飞船的设计与实现

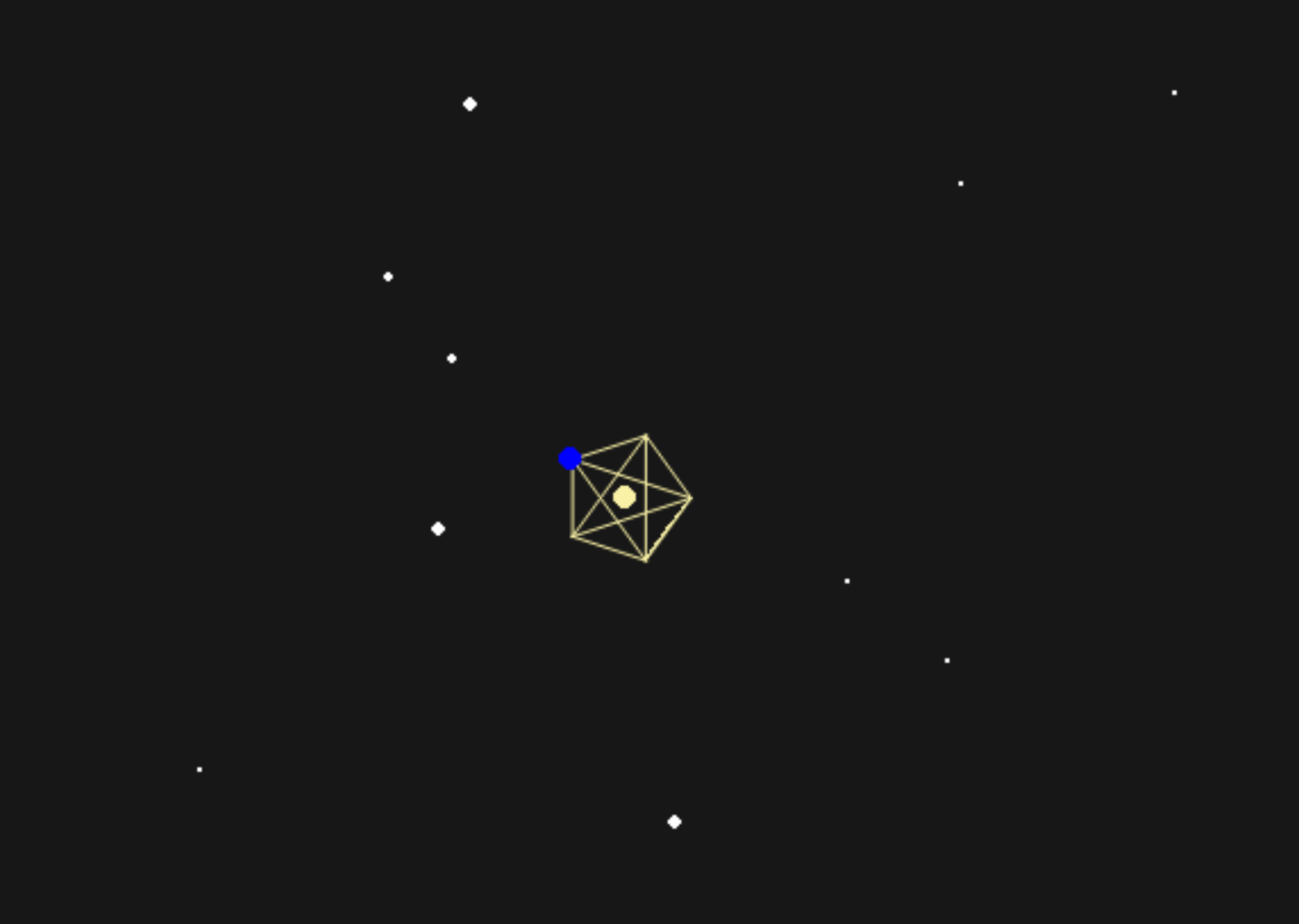
在宇宙飞船的外观设计上，我们一开始采取贴图模型式（如下图为选取样例之一）



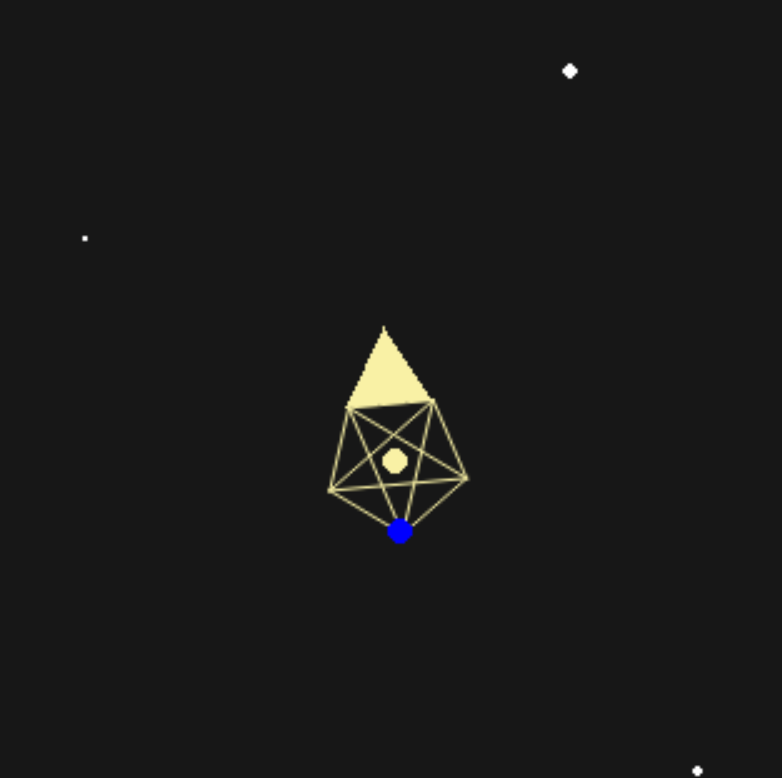
而后发现在选择星空作为背景的情况下，网络上已有的贴图无法在当前技术层面上做到与原背景完美贴合并做到风格差异，且在成比例放大时会出现锯齿状与模糊化（如下图），不美观且无法与命中体积，动态背景贴合，故采用计算生成矢量图模型而非贴图模型，能够更加美观且完美贴合命中体积，动态背景。



最终矢量图效果图：（科幻极简风格）未加速



加速状态飞船图：



在宇宙飞船的功能设计上，我们一开始设计了传统空战式的操作——即通过位移键平移调整方向，射击键射击固定方向，这样在移动上不拟真，且无法与现实中的方向——加速引擎系统产生贴合，且在射击上只有一个方向的自由度。无法彻底实现自由射击，这无疑是与我们的想法背道而驰的。故在不断的测试与改进中，我们大胆的放弃了方向键平移位移系统，转而采用鼠标方向控制左转/右转，将所有操作（包括加速/射击）全移植至鼠标上。

在飞船的移动功能中，我们模拟现实中的物理法则，建立物理引擎，采取加速度方式进行移动，在加速与转向过程中，只改变飞船的加速度通过物理引擎改变速度。可以做到仿真与趣味性。

在飞船的射击过程与被射击过程中，我们除开模拟飞船被击中时的生命值损失，而且模拟现实中的物理法则，建立物理引擎，从而计算仿真出飞船在遭受子弹后的击退效果与射击时的后坐力效果。

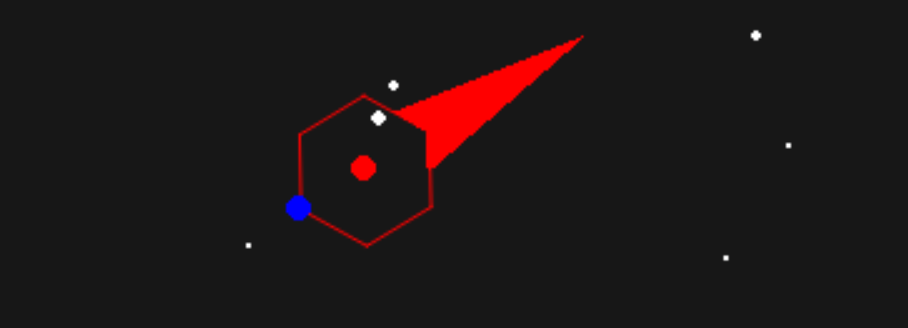
在飞船的升级过程中，我们除了增加飞船本身的属性值，每次升级，还会随机给飞船三个强化选项。三个选项依托于随机数。从而在多次游戏中，游戏不会因熟悉而单调乏味，反而会常玩常新，极大增加游戏的耐玩程度。

## 2.外星人的设计与实现

在外星飞船的外观设计上，我们一开始采取贴图模型式（如下图为选取样例之一）



而后发现在选择星空作为背景的情况下，网络上已有的贴图无法在当前技术层面上做到与原背景完美贴合并做到风格差异，且在成比例放大时会出现锯齿状与模糊化，无法与命中体积贴合，故采用计算生成矢量图模型而非贴图模型，能够更加美观且完美贴合命中体积与动态背景。（最终效果图如下）



在外星飞船的功能设计上，我们设计了三种敌人——可以发射子弹和不能发射子弹，近身突袭的，以及后期出现的可以产生飞船的母舰，从而增加了游戏的层次感与多样性，并使得刷新出的敌人可以根据我方等级的改变而改变，增加了游戏后期的挑战性。

在外星飞船的攻击中，我们模拟现实中的物理法则，建立物理引擎，采取加速度方式进行移动，在飞船向我方飞船前进的同时，也会使得子弹被我方子弹抵消，自身被我方子弹击退。增加了玩家的真实感与操作感。同时，外星飞船的设计为在大地图中随机刷新，并环绕在友方飞船周围出现。

在飞船的升级过程中，刷新的外星飞船本身的属性值并非固定不变，而是会随着友军的升级而变强，友军只能根据升级新获得的功能去对抗新的敌军，增加了游戏后期的挑战性

## 3.游戏评分的设计与实现

游戏评分主要通过升级与击杀敌人实现。升级会获得固定评分，同时，击杀敌人会获得基于敌军等级的分数。

## 4.其他的设计与实现

游戏地图主要采用算法随机生成星空，通过算法随机生成不同亮度的星星去模拟星空，同时通过不同亮度星星不同速度来实现星空的伪立体化。

游戏的音效与音乐设置主要采取pygame内置函数，并对采取音效适当微调，使得游戏更有打击感，同时游戏可以在设置中关闭音效。

# 三、实验总结和心得体会

## 1.实验中遇到的问题及解决方案

在实验中，我们遇到了以下问题：

1. 武装飞船和外星人的贴图风格难以统一并且现有外星人贴图过于单一粗糙，我们的解决方案是借鉴了游戏《几何竞技场(Geometry Arena)》的画风，通过pygame.draw之中的函数来绘制几何图形以显现宇宙飞船和外星人的
2. 因为宇宙飞船固定在屏幕中央，在屏幕上它只能转动而并不会移动，因此在实际游玩的体验当中有些难以体会到飞船的实际的移动速度，我们的解决方案是给飞船加上尾焰，并且设计成尾焰是逐渐变长的，以符合飞船逐渐加速的状态。并且添加基于飞船的速度而动态改变的游戏星空背景，使得飞船的移动有了更加实际的参照物。而且，我们为飞船的加速添加了引擎的音效，使得其更加真实可感。
3. 考虑到飞船需要足够的速度来躲避外星人的子弹，因此我们给予其的加速度并不能太小，所以很容易出现飞船一直加速而无法找到外星人的情况。我们的解决方案是添加一个减速因子，在每一个时刻给飞船和外星人的速度乘上一个介于0到1之间的值，同时，也给予外星人加速的能力，并且进一步设置外星人的AI，使得其会在与飞船距离大于设定值的情况下，向飞船加速靠近。
4. 考虑到我们的外星人是同时生成的，而在它的武器冷却时间是相同的情况下，所有飞船的射击时刻将是同时的，我们的解决方案是将冷却时间和外星人等级相关，同时在外星人等级确定之中引入相关性的因素。
5. 考虑到外星人都是单纯的正多边形并且只会射击，较为单调，我们的解决方案是根据等级改变外星人的颜色，并且增加了一种可以发射外星人飞船的母舰，作为敌人，使得游戏后期更为紧张。

## 2.团队协作及每个人对完成实验的具体贡献（分工与合作）

王韬主要负责游戏评分、结束菜单、设置菜单以及游戏音效的实现。

王时青主要负责开始菜单、升级菜单、游戏之中ui、技能树以及飞船和背景显示的实现。

王韬和王时青共同完成了游戏运行部分的实现。

## 心得体会

在本学期的python大作业中，我们在老师的教导下，学习了python的有关知识。但纸上得来终觉浅，绝知此事要躬行。单纯的学习是苍白无力的。学习不仅仅是理论知识的学习，还需要实践。在实践中，我们小组不仅仅根据老师传输的知识，而且通过网络搜索，github源码学习等方式进一步学习pygame知识。且接触并熟悉游戏开发流程。借此，我们展开编写了俯视角rougelike类游戏《外星人入侵》。在本次课程的学习中，我们不仅学习到了pygame模块的应用方式，而且强化学习了学习编程语言的方法。

# 四、附录

## 1．完整的代码和注释（主程序，boat对象，其余代码另附）

（主函数）

1. **import** pygame
2. **import** sys
3. **from** math **import** sqrt
4. **from** settings **import** Settings  # 加载选项
5. **from** menu **import** Start\_menu, Choose\_menu  # 构造开始菜单
6. **from** boat **import** Boat, Player  # 构建游戏玩家太空船
7. **from** alien **import** Alien, alien\_summon,Summoner  # 构造敌人
8. **from** game\_ui **import** Game\_UI  # 当前血量显示
9. **from** setting\_menu **import** Setting\_menu  # 设置菜单
10. **from** over\_menu **import** Over\_menu  # 结束菜单
11. **import** background  # 随机算法形成动态星空背景
13. **def** level\_up(screen, sett, player):  # 升级菜单：升级强化
14. menu = Choose\_menu(screen, sett)  # 打开菜单
15. **while** True:
16. **for** event **in** pygame.event.get():
17. **if** event.type == pygame.QUIT:
18. sys.exit()
19. **if** event.type == pygame.KEYDOWN:
20. **if** event.key == pygame.K\_a **or** event.key == pygame.K\_LEFT:
21. menu.left()
22. **if** event.key == pygame.K\_d **or** event.key == pygame.K\_RIGHT:
23. menu.right()
24. **if** event.key == pygame.K\_RETURN:
25. # 调用player实现升级，但是，要是修改的是alien的属性该怎么办？
26. player.level\_up(sett, menu.get())
27. sett.choice\_list.remove(menu.get())
28. **return**
29. # 进行升级选择
30. screen.fill(sett.bg\_color)
31. menu.display()
32. pygame.display.flip()

35. **def** overmenu(score, sett, screen):
36. ovmu = Over\_menu(sett.screen\_width, sett.screen\_height, screen,
37. pygame.font.SysFont('Comic Sans MS', 50), pygame.font.SysFont('Comic Sans MS', 24), score)
38. filepath = r".\music\alien-spaceship\_daniel\_simion.mp3"  # 打开mp3文件播放结束背景音乐
39. pygame.mixer.init()
40. pygame.mixer.music.load(filepath)  # 加载MP3文件
41. pygame.mixer.music.play(loops=-1)  # 播放mp3文件
42. pygame.mixer.music.set\_volume(1 **if** sett.set\_music **else** 0)
43. # 因为升级会改变sett之中的一些参量，所以在一局游戏结束之后，需要将其再一次的初始化
44. sett.\_\_init\_\_()
45. **while** True:
46. pygame.mixer.music.set\_volume(1 **if** sett.set\_music **else** 0)
47. **for** event **in** pygame.event.get():
48. **if** event.type == pygame.QUIT:
49. sys.exit()
50. **if** event.type == pygame.KEYDOWN:
51. **if** event.key == pygame.K\_w **or** event.key == pygame.K\_UP:
52. ovmu.up()
53. **if** event.key == pygame.K\_s **or** event.key == pygame.K\_DOWN:
54. ovmu.down()
55. **if** event.key == pygame.K\_RETURN:
56. **if** ovmu.count == 0:
57. pygame.mixer.music.stop()
58. run\_game(screen, sett)
59. **elif** ovmu.count == 1:
60. sys.exit()  # 选择重新开始或者退出游戏
61. screen.fill(sett.bg\_color)
62. # screen.blit(text\_surface,text\_rect)
63. ovmu.display()
64. pygame.display.flip()
66. # 游戏菜单
67. **def** run\_game(screen, sett):
68. # 记录游戏分数
69. score = 0
70. # 生成游戏背景对象
71. bkg = background.Background(sett, 50)
72. # 加载游戏bgm
73. filepath = r".\music\alien-spaceship\_daniel\_simion.mp3"  # 打开游戏运行音乐
74. **if** sett.set\_music:
75. pygame.mixer.init()
76. pygame.mixer.music.load(filepath)  # 加载MP3文件
77. pygame.mixer.music.play(loops=-1)  # 播放mp3文件
78. pygame.mixer.music.set\_volume(1 **if** sett.set\_music **else** 0)
79. lenth = 50 / sqrt(3)  # 50为三角形边长
80. player\_color = (249, 241, 165)  # 设置游戏颜色
81. alien\_color = (249, 165, 241)  # 设置外星人颜色
82. # 生成玩家飞船
83. player = Player(lenth, (sett.screen\_width / 2, sett.screen\_height / 2), [0, 1, 2, 3, 4, 0, 2, 4, 1, 3], 0)
84. summmor = Summoner(player.cen\_pos,10)
85. # 记录玩家飞船是否处于设计状态和加速状态
86. attack\_flag = False
87. speedup\_flag = False
88. # 友方子弹列表
89. player\_bullet\_list = []
90. # 敌方子弹列表
91. alien\_bullet\_list = []
92. # 敌方飞船列表
93. alien\_list = [alien\_summon(500, player.cen\_pos, side\_num=i,level=player.level) **for** i **in** range(3, 10)]
94. summmor\_list = []
95. # 升级所需要的经验
96. level\_xp = 100
97. # 标记玩家子弹准备时间
98. bullet\_time = 0
99. # 加载游戏ui显示
100. ui = Game\_UI(sett)
101. # 火箭声音
102. rocket\_sound\_path = "./music/Rocket\_Sound.mp3"
103. rocket\_channel = pygame.mixer.Channel(0)
104. flag = True
105. **while** True:
106. # 判定是否应当播放火箭声音
107. **if** speedup\_flag **and** sett.set\_sound:
108. # 判定当前是否已经在播放火箭声音
109. **if** **not** rocket\_channel.get\_busy():
110. rocket\_channel.play(pygame.mixer.Sound(rocket\_sound\_path))
111. **else**:
112. rocket\_channel.stop()
113. # 将背景写入缓冲区，并使背景根据玩家速度来更新星星的位置
114. bkg.display(screen, sett, player.v)
115. # 通过clock来实现60帧
116. pygame.time.Clock().tick(60)
117. # 当外星人清空时，添加新的外星人
118. **if** len(alien\_list) == 0 **and** len(summmor\_list)==0:
119. **for** i **in** range(7):
120. alien\_list.append(alien\_summon(500, player.cen\_pos, side\_num=10,level=player.level))  # 加入外星人
121. # 到达固定等级之后开始生成召唤师
122. **for** i **in** range(player.level-7):
123. summmor\_list.append(Summoner(player.cen\_pos,player.level))
124. # 处理事件
125. **for** event **in** pygame.event.get():
126. **if** event.type == pygame.QUIT:
127. sys.exit()
128. # if event.type==pygame.KEYDOWN:
129. #     if event.key==pygame.K\_a or event.key==pygame.K\_LEFT:
130. #         player.v=(-1,0)\*player.speed
131. #         # player.cen\_pos
132. #     if event.key==pygame.K\_d or event.key==pygame.K\_RIGHT:
133. #         player.v=(1,0)\*player.speed
134. #         # pass
135. #     if event.key==pygame.K\_w or event.key==pygame.K\_UP:
136. #         player.v=(0,1)\*player.speed
137. #         # menu.up()
138. #     if event.key==pygame.K\_s or event.key==pygame.K\_DOWN:
139. #         player.v=(0,-1)\*player.speed
140. #         # menu.down()
141. # 按下鼠标的情况
142. **if** event.type == pygame.MOUSEBUTTONDOWN:
143. # 为了贴近现实逻辑，把飞船设置成头部开炮，尾部引擎控制方向
144. **if** event.button == 1:  # 鼠标左键点击——射击
145. attack\_flag = True
146. bullet\_time = 0
147. **if** event.button == 3:  # 鼠标右键点击——加速
148. speedup\_flag = True
149. # 鼠标抬起的情况
150. **if** event.type == pygame.MOUSEBUTTONUP:
151. **if** event.button == 1:
152. attack\_flag = False
153. **if** event.button == 3:
154. speedup\_flag = False
155. # 子弹时间充分，进行射击
156. **if** bullet\_time >= player.cooldown:
157. **for** b **in** player.attack():
158. player\_bullet\_list.append(b)
159. bullet\_time = 0
160. # 玩家意欲攻击时，进行攻击准备
161. **if** attack\_flag:
162. bullet\_time += 1
163. # 玩家意欲加速时，进行加速
164. **if** speedup\_flag:
165. player.speedup()
166. # 敌方射击
167. **for** alien **in** alien\_list:
168. t = alien.attack()
169. **if** t **is** **not** None:
170. **for** b **in** t:
171. alien\_bullet\_list.append(b)
172. **for** summmor **in** summmor\_list:
173. **for** a **in** summmor.summon():
174. alien\_list.append(a)
175. # 将玩家飞船写入缓冲区
176. player.display(screen, pygame.mouse.get\_pos(), player.v)
177. # 将外星飞船写入缓冲区
178. **for** alien **in** alien\_list:
179. alien.display(screen, player.cen\_pos, player.v)
180. # 对外星人进行加速
181. **if** alien.need\_speedup(player.cen\_pos):
182. alien.speedup()
183. # 判定是否需要移除外星人
184. **if** alien.death():
185. score = score + alien.level \* 10
186. # 击杀外星人之后增加对应的经验
187. player.xp += alien.xp
188. alien\_list.remove(alien)
189. **for** summmor **in** summmor\_list:
190. summmor.display(screen, player.cen\_pos, player.v)
191. **if** summmor.need\_speedup(player.cen\_pos):
192. summmor.speedup()
193. **if** summmor.death():
194. score += summmor.level \* 50
195. player.xp += summmor.xp
196. summmor\_list.remove(summmor)
197. # 将子弹写入缓冲区
198. **for** b **in** alien\_bullet\_list:
199. b.display(screen, player.v)
200. # 对子弹进行碰撞检测
201. **if** pygame.sprite.collide\_circle(player, b):
202. b.hit(player)
203. player.attacked(hiter=b)
204. **if** b.death():
205. alien\_bullet\_list.remove(b)
206. **for** b **in** player\_bullet\_list:
207. b.display(screen, player.v)
208. **for** alien **in** alien\_list:
209. **if** pygame.sprite.collide\_circle(alien, b):
210. b.hit(alien)
211. alien.attacked(b)
212. **for** summmor **in** summmor\_list:
213. **if** pygame.sprite.collide\_circle(summmor, b):
214. b.hit(summmor)
215. summmor.attacked(b)
216. # 子弹之间进行碰撞检测
217. **for** b1 **in** alien\_bullet\_list:
218. **if** pygame.sprite.collide\_circle(b1, b):
219. b1.hit(b)
220. **if** b.death():
221. player\_bullet\_list.remove(b)
222. # 对玩家飞船和敌方飞船进行碰撞检测
223. **for** alien **in** alien\_list:
224. **if** pygame.sprite.collide\_circle(alien, player):
225. player.attacked(alien)
226. alien.attacked(player)
227. **for** summmor **in** summmor\_list:
228. **if** pygame.sprite.collide\_circle(summmor, player):
229. player.attacked(summmor)
230. summmor.attacked(player)
231. # 需要升级
232. **if** player.xp >= level\_xp:
233. score = score + 100
234. player.xp -= level\_xp
235. level\_xp += 100
236. attack\_flag = False
237. speedup\_flag = False
238. level\_up(screen, sett, player)
239. # ui界面输出到缓冲区
240. ui.display(screen, player, sett,level\_xp)
241. pygame.display.flip()
242. # 如果玩家死亡，跳转到结束菜单
243. **if** player.death():
244. overmenu(score, sett, screen)

247. **def** run\_setting(screen,sett):  # 设置
248. # 生成一个设置菜单类，并确定其中的字体
249. menu = Setting\_menu(sett.screen\_width, sett.screen\_height, screen,
250. pygame.font.SysFont('Comic Sans MS', 50), pygame.font.SysFont('Comic Sans MS', 24), sett)
251. **while** True:
252. # 处理时间
253. **for** event **in** pygame.event.get():
254. **if** event.type == pygame.QUIT:
255. sys.exit()
256. **if** event.type == pygame.KEYDOWN:
257. # 上下移动选择
258. **if** event.key == pygame.K\_w **or** event.key == pygame.K\_UP:
259. menu.up()
260. **if** event.key == pygame.K\_s **or** event.key == pygame.K\_DOWN:
261. menu.down()
262. **if** event.key == pygame.K\_RETURN:
263. # 对对应选项进行更新
264. **if** menu.count == 0 **or** menu.count == 1:
265. menu.switch(sett)
266. # 退出
267. **else**:
268. **return**
269. screen.fill(sett.bg\_color)
270. menu.display()
271. pygame.display.flip()

274. **def** show\_menu():
275. # 初始化pygame
276. pygame.init()
277. # 加载运行图标
278. surface = pygame.image.load("./icon.png")
279. pygame.display.set\_icon(surface)
280. # 设定窗口标题
281. pygame.display.set\_caption("Alien Invasion")
282. # 初始化一个设置类
283. sett = Settings()
284. # 设置屏幕长和宽
285. screen = pygame.display.set\_mode((sett.screen\_width, sett.screen\_height))
286. # 加载bgm
287. filepath = r".\music\alien-spaceship\_daniel\_simion.mp3"  # 打开始界面MP3
288. pygame.mixer.init()
289. pygame.mixer.music.load(filepath)  # 加载MP3文件
290. pygame.mixer.music.play(loops=-1)  # 播放mp3文件，并设置其为循环播放
291. pygame.mixer.music.set\_volume(1 **if** sett.set\_music **else** 0) # 设置音量
292. # 加载开始菜单类，并定义字体
293. menu = Start\_menu(sett.screen\_width, sett.screen\_height, screen,
294. pygame.font.SysFont('Comic Sans MS', 50), pygame.font.SysFont('Comic Sans MS', 24))
295. **while** True:
296. # pygame.mixer.music.set\_volume(1 if sett.set\_music else 0)
297. # 处理事件
298. **for** event **in** pygame.event.get():
299. **if** event.type == pygame.QUIT:
300. sys.exit()
301. **if** event.type == pygame.KEYDOWN:
302. # 处理上下选择的情况
303. **if** event.key == pygame.K\_w **or** event.key == pygame.K\_UP:
304. menu.up()
305. **if** event.key == pygame.K\_s **or** event.key == pygame.K\_DOWN:
306. menu.down()
307. # 选中的情况
308. **if** event.key == pygame.K\_RETURN:
309. # 选中开始游戏
310. **if** menu.count == 0:
311. **print**("game start")
312. pygame.mixer.music.stop()
313. run\_game(screen, sett)
314. # 选中游戏设置
315. **elif** menu.count == 1:
316. **print**("settings")
317. run\_setting(screen,sett)
318. **if** **not** sett.set\_music:
319. pygame.mixer.music.stop()
320. **elif** **not** pygame.mixer.music.get\_busy():
321. pygame.mixer.music.play(loops=-1)
322. # 选中退出
323. **elif** menu.count == 2:
324. sys.exit()
325. # 填充背景颜色
326. screen.fill(sett.bg\_color)
327. # 将菜单输出到缓冲区之中
328. menu.display()
329. # 绘制缓冲区
330. pygame.display.flip()

333. **if** \_\_name\_\_ == "\_\_main\_\_":
334. # 开始游戏
335. show\_menu()

Boat对象（包含部分物理引擎）：

1. **from** math **import** atan2, cos, pi, sin, sqrt
2. **import** pygame
3. **from** bullet **import** Bullet

6. **class** Boat():  # 太空船——（玩家与敌军的原型）
7. **def** \_\_init\_\_(self, lenth=100.0, cen\_pos=(0.0, 0.0), shape=[0, 1, 2], theta=0.0, color=(291, 241, 165), v=(0, 0),
8. bullet\_v=10, a=0.1, cooldown=5, health=100, bullet\_number=1, bullet\_scatter=pi / 3, bullet\_type=0,
9. damage=5, delta\_vk=0.1):
10. self.lenth = lenth  # 定义大小
11. self.cen\_pos = cen\_pos  # 中心位置
12. self.shape = shape#形状
13. self.color = color#颜色
14. self.theta = theta  # 初始角度
15. self.omega = 0.05  # 限定每帧转过的角度
16. self.v = v  # 飞船速度
17. self.bullet\_v = bullet\_v  # 子弹出膛速度
18. self.a = a # 飞船加速度
19. self.cooldown = cooldown  # 射击时子弹冷却时间
20. self.speed = 1
21. self.fire\_lenth = 0  # 火焰长度
22. self.side\_num = max(shape) + 1
23. self.rel = cos(pi / self.side\_num) \* self.lenth
24. self.rel2 = self.rel \* sqrt(3)
25. self.rect = pygame.Rect(self.cen\_pos[0] - self.rel2 / 2, self.cen\_pos[1] - self.rel2 / 2, self.rel2, self.rel2)
26. self.behit = 0  # 被撞击后会改变的参量
27. self.bullet\_damage = 5
28. self.bullet\_number = bullet\_number
29. self.bullet\_scatter = bullet\_scatter
30. self.bullet\_type = bullet\_type
31. self.health = health
32. self.shielded = 0
33. self.blutting = 1
34. self.damage = damage
35. self.delta\_vk = delta\_vk
37. **def** display(self, screen, to\_pos, playerv):
38. m = self.side\_num
39. self.cen\_pos = (self.cen\_pos[0] + self.v[0] - playerv[0], self.cen\_pos[1] + self.v[1] - playerv[1])
40. alpha = atan2(to\_pos[0] - self.cen\_pos[0], to\_pos[1] - self.cen\_pos[1])
41. self.v = (self.v[0] \* 0.99, self.v[1] \* 0.99)
42. **if** alpha < 0: alpha += 2 \* pi
43. **if** min(2 \* pi - abs(alpha - self.theta), abs(alpha - self.theta)) <= self.omega:
44. self.theta = alpha
45. **elif** 0 < alpha - self.theta < pi **or** -2 \* pi < alpha - self.theta < -pi:
46. self.theta += self.omega
47. **if** self.theta >= 2 \* pi:
48. self.theta -= 2 \* pi
49. **else**:
50. self.theta -= self.omega
51. **if** self.theta < 0:
52. self.theta += 2 \* pi
53. point\_list = []
54. theta = self.theta
55. **for** i **in** range(m):
56. point\_list.append((self.cen\_pos[0] + sin(theta) \* self.lenth, self.cen\_pos[1] + cos(theta) \* self.lenth))
57. theta += 2 \* pi / m
59. poly\_point\_list = [point\_list[i] **for** i **in** self.shape]
60. width = 5
61. pygame.draw.aalines(screen, self.color, True, poly\_point\_list)
62. # 绘制中心点
63. pygame.draw.circle(screen, self.color, self.cen\_pos, width)
64. # 绘制炮管
65. pygame.draw.circle(screen, (0, 0, 255), point\_list[0], width)
66. # 绘制尾焰
67. fire\_point\_list = []
68. **if** (len(point\_list) % 2 == 0):
69. n = len(point\_list) // 2
70. fire\_point\_list.append(
71. ((point\_list[n - 1][0] + point\_list[n][0]) / 2, ((point\_list[n - 1][1] + point\_list[n][1]) / 2)))
72. fire\_point\_list.append(point\_list[n])
73. fire\_point\_list.append(
74. ((point\_list[n + 1][0] + point\_list[n][0]) / 2, ((point\_list[n + 1][1] + point\_list[n][1]) / 2)))
75. theta = pi + self.theta
76. lenth = self.lenth + self.fire\_lenth
77. fire\_point\_list.append((self.cen\_pos[0] + sin(theta) \* lenth, self.cen\_pos[1] + cos(theta) \* lenth))
78. **else**:
79. n = len(point\_list) // 2
80. fire\_point\_list.append(point\_list[n + 1])
81. fire\_point\_list.append(point\_list[n])
82. theta = pi + self.theta
83. lenth = self.fire\_lenth
84. fire\_point\_list.append(((point\_list[n][0] + point\_list[n + 1][0]) / 2 + sin(theta) \* lenth,
85. (point\_list[n][1] + point\_list[n + 1][1]) / 2 + cos(theta) \* lenth))
86. self.fire\_lenth \*= 0.95
87. pygame.draw.polygon(screen, self.color, fire\_point\_list)
89. **def** speedup(self):
90. self.v = (self.v[0] + sin(self.theta) \* self.a, self.v[1] + cos(self.theta) \* self.a)
91. self.fire\_lenth = self.fire\_lenth + 4
93. **def** attack(self):
94. b\_list = []
95. **if** self.bullet\_number == 1:
96. b\_list.append(
97. Bullet(v=(self.bullet\_v \* sin(self.theta) + self.v[0], self.bullet\_v \* cos(self.theta) + self.v[1]),
98. theta=self.theta, color=self.color, pos=(
99. self.cen\_pos[0] + sin(self.theta) \* self.lenth, self.cen\_pos[1] + cos(self.theta) \* self.lenth),
100. damage=self.bullet\_damage))
101. **else**:
102. match self.bullet\_type:
103. case 0:
104. # 对应散射的情况
105. theta = self.theta - self.bullet\_scatter / 2
106. theta = (theta + 2 \* pi) % (2 \* pi)
107. delta\_theta = self.bullet\_scatter / (self.bullet\_number - 1)
108. **for** i **in** range(self.bullet\_number):
109. b\_list.append(
110. Bullet(v=(self.bullet\_v \* sin(theta) + self.v[0], self.bullet\_v \* cos(theta) + self.v[1]),
111. theta=theta, color=self.color, pos=(self.cen\_pos[0] + sin(self.theta) \* self.lenth,
112. self.cen\_pos[1] + cos(self.theta) \* self.lenth),
113. damage=self.bullet\_damage))
114. theta += delta\_theta
115. theta %= 2 \* pi
116. case 1:
117. # 平射
118. **if** self.bullet\_number % 2 == 1:
119. # 奇数的情况，则中心点发出弹药
120. b\_list.append(Bullet(v=(
121. self.bullet\_v \* sin(self.theta) + self.v[0], self.bullet\_v \* cos(self.theta) + self.v[1]),
122. theta=self.theta, color=self.color, pos=(
123. self.cen\_pos[0] + sin(self.theta) \* self.lenth,
124. self.cen\_pos[1] + cos(self.theta) \* self.lenth), damage=self.bullet\_damage))
125. # 再分开处理两边的情况
126. num = self.bullet\_number // 2
127. theta = 2 \* pi / self.side\_num
128. theta = (self.theta - theta + 2 \* pi) % (2 \* pi)
129. p0 = (self.cen\_pos[0] + sin(theta) \* self.lenth, self.cen\_pos[1] + cos(theta) \* self.lenth)
130. p1 = (
131. self.cen\_pos[0] + sin(self.theta) \* self.lenth, self.cen\_pos[1] + cos(self.theta) \* self.lenth)
132. **for** i **in** range(num):
133. b\_list.append(Bullet(v=(
134. self.bullet\_v \* sin(self.theta) + self.v[0], self.bullet\_v \* cos(self.theta) + self.v[1]),
135. theta=self.theta, color=self.color, pos=(
136. p0[0] \* (num - i) / num + p1[0] \* i / num, p0[1] \* (num - i) / num + p1[1] \* i / num),
137. damage=self.bullet\_damage))
138. theta = 2 \* pi / self.side\_num
139. theta = (self.theta + theta) % (2 \* pi)
140. p0 = (self.cen\_pos[0] + sin(theta) \* self.lenth, self.cen\_pos[1] + cos(theta) \* self.lenth)
141. **for** i **in** range(num):
142. b\_list.append(Bullet(v=(
143. self.bullet\_v \* sin(self.theta) + self.v[0], self.bullet\_v \* cos(self.theta) + self.v[1]),
144. theta=self.theta, color=self.color, pos=(
145. p0[0] \* (num - i) / num + p1[0] \* i / num, p0[1] \* (num - i) / num + p1[1] \* i / num),
146. damage=self.bullet\_damage))
147. # 反冲力
148. delta\_v = 0.5
149. self.v = (self.v[0] - sin(self.theta) \* delta\_v, self.v[1] - cos(self.theta) \* delta\_v)
150. **return** b\_list
152. **def** attacked(self, hiter):
153. self.behit += 1
154. self.health -= hiter.damage
155. # 子弹撞击后产生的速度改变量
156. delta\_v = (hiter.v[0] - self.v[0], hiter.v[1] - self.v[1])
157. self.v = (self.v[0] + hiter.delta\_vk \* delta\_v[0], self.v[1] + hiter.delta\_vk \* delta\_v[1])
159. # pygame.mixer.music.stop()#播放mp3文件
160. filepath = r".\music\player\_hit.wav"  # 打开mp3文件
161. # pygame.mixer.music.pygame.mixer.Channel(0).load(filepath)#加载MP3文件
162. # pygame.mixer.music.play(loops=0,start=0,fade\_ms=0)#播放mp3文件
164. pygame.mixer.Channel(1).play(pygame.mixer.Sound(filepath))
166. **def** death(self):
167. **if** self.health > 0:
168. **return** False
169. **return** True  # 死亡发生事情


173. **class** Player(Boat):
174. **def** \_\_init\_\_(self, lenth=100.0, cen\_pos=(0, 0), shape=[0, 1, 2], theta=0, color=(249, 241, 165), v=(0, 0),
175. bullet\_v=10, a=0.1, cooldown=5, health=100,full\_health=100, bullet\_number=1, bullet\_scatter=pi / 3, bullet\_type=0):
176. super().\_\_init\_\_(lenth, cen\_pos, shape, theta, color, v, bullet\_v, a, cooldown, health, bullet\_number,
177. bullet\_scatter, bullet\_type)
178. self.level = 1
179. self.xp=0
180. self.full\_health=health
182. **def** level\_up(self, sett, index):
183. self.level += 1
184. # 增加最大血量上限
185. self.full\_health=int(self.full\_health\*1.2)
186. # 升级时回满血量
187. self.health=self.full\_health
188. # 以下部分需要在python3.10以上使用
189. match index:
190. case 0:
191. # 子弹伤害增加
192. self.bullet\_damage \*= 2
193. sett.choice\_list.append(10)
194. case 1:
195. # 发射速度加快
196. # 在cooldown值小于1的情况下，这样的升级将不再有效
197. self.cooldown /= 2
198. case 2:
199. # 子弹发射数量上升，散射
200. self.bullet\_number \*= 2
201. # 进一步提升数量
202. sett.choice\_list.append(4)
203. # 减小子弹散射
204. sett.choice\_list.append(5)
205. case 3:
206. # 子弹可以弹射
207. # 这需要给子弹赋予属性
208. **return**
209. case 4:
210. # 子弹发射数量进一步上升
211. self.bullet\_number \*= 2
212. case 5:
213. # 减小子弹散射
214. self.bullet\_scatter /= 2
215. sett.choice\_list.append(6)
216. case 6:
217. # 取消子弹散射
218. self.bullet\_type = 1
219. case 7:
220. # 增加选择数量
221. sett.choice\_count += 1
222. sett.choice\_list.append(8)
223. case 8:
224. # 进一步增加选择数量
225. sett.choice\_count += 1
226. case 9:
227. # 进行吸血
228. sett.vampire = 0.1
229. sett.choice\_list.append(11)
230. case 10:
231. # 子弹伤害上升
232. self.bullet\_damage \*= 2
233. case 11:
234. # 吸血能力加强
235. sett.vampire \*= 2
236. case 12:
237. # 转向速度提升
238. self.omega \*= 2
239. case 13:
240. # 推进速度提升
241. self.a \*= 2

**学生签名：王韬，王时青**

2022.12.