Project_Report

项目报告

叶顾燊 余在畅 梁景育

2025.1.12

1. 引言

本项目为SI100B 最终项目
项目文件储存在 https://github.com/Ye-on-fire/SI100B_Project
我们制作了一款**roguelike**游戏。玩家初始出生在休息站，可以和最左边的npc对话获取游戏的操作方式和目标，**和篝火对话**可以选择花钱升级或者开始进入闯关环节，玩家开始身上并没有金钱，在闯关过程中玩家还能回到休息站中进行升级，休息站中还有一个**大语言模型驱动**的医师角色，和她对话可以治疗。关于战斗，采用**即时动作制玩法**，玩家击败小怪可以获得金钱，小怪难度会随着关卡数的上升而上升，当达到10关后会进入boss关，**击败boss**即可通关。

2.项目实施

2.1 场景

1. 大型场景

- 游戏中的每个场景都是大型场景，都有着可以跟随玩家移动而进行视角移动的摄像头。
- 镜头移动的原理：每当满足镜头移动的条件时，所有物体（含地砖、NPC、玩家）都会获得一个偏移量，该偏移量等于在这一帧玩家试图移动的距离，在渲染时所有物体的位置都会加上这个偏移量的相反数。因此，玩家移动了 (dx, dy) + (−dx, −dy) = (0, 0), 即在原地不动，其它物体全部移动偏移量的相反数，达到了镜头移动的效果。

2. 场景数量

- 游戏中有12个场景，一个休息站（初始房间）场景，十个战斗场景，一个boss房，战斗场景中会生成障碍物，地图，以及越来越多强度更高敌人。当玩家击败本场景内的所有敌人

后，可通过门进入下一个战斗场景或休息站，通过前十个战斗场景后会进入第十一个boss房场景，即boss房，进行最终的boss战。
- 场景由地图砖块与障碍物构成，在每一帧中，程序会一个一个地渲染地图砖块，拼接起来的砖块形成了一个完整的场景。所有场景均由 `game_collections.py` 中的 `SceneLike` 类实现 。

3. 互动物品

- 游戏中有三类互动物品，分别为障碍物，门，以及小篝火。
- 障碍物包含树木，灌木，残破石柱，石碑，石块，玩家无法跨越这些障碍物。
- 门出现在每个战斗场景的上方，场景中存在敌人时门保持关闭，当所有敌人被击败后门会打开，此时玩家可以通过门进入下一个战斗场景。
- 每若干个战斗场景中会出现一个小篝火，当场景中所有敌人被击败后，小篝火会变得可交互，触碰小篝火会将玩家传送至休息站。

2.2 角色

1. 主要角色

- 在我们的游戏中，**主要角色具有初始100点的血量，100点的体力条，10点攻击力**，其中体力条经过一段时间冷却后会随着时间恢复。血量与体力条分别以绿色和黄色进度条的形式显示在左上角。玩家可在休息站处用金币升级对应属性。
- 玩家可以通过 **WASD** 四个按键操控主要角色四处移动，通过门传送到下一个场景，主要角色的 移动通过检测按键按下与抬起这两种事件的方法来实现，当按键被按下，角色就会持续朝指定方向移动，按键抬起后，角色不再朝指定方向移动，角色在遇到障碍物时将无法继续向前移动。向左和向右移动时角色分别会播放对应的跑步动画。
- 角色通过按下**SPACE键**向前进行翻滚，**翻滚会消耗10点体力**，翻滚过程中有短暂的无敌效果，角色具有对应的翻滚动画。
- 角色通过按下**J键**对前方一片区域进行攻击，**攻击会消耗20点体力，造成攻击力数值的伤害**，角色具有对应的攻击动画。
- 角色通过按下**K键**对身后一片区域进行攻击，**身后斩会消耗30点体力，造成攻击力数值的伤害**，角色具有对应的身后斩动画。
- 角色被敌人攻击时会扣除对应血量，角色具有对应的受击动画。
- 角色在接近npc时通过**E键**进行交互。
- 角色的生命值小于等于0时角色死亡，游戏结束，角色具有死亡动画。
- 角色的所有设置在 `game_objects.py` 中的 `Player` 类中实现。

2. 友好NPC

- 在休息站（初始房间）中一共有三个友好npc，分别是指引者，医师和大篝火，靠近npc时按下**E键**交互，按下**1键**、**2键**、**3键**做出选择，按下**esc键**退出对话。
- 与指引者交互时，她会告诉你游戏玩法。
- 与医师交互时，你可以在弹出的输入框输入文字与医师进行交谈（英文），当你提出需要回血时（关键词"need heal"）他会将你的血量回满.
- 与大篝火交互时，你可以选择再次进入战斗场景或进行升级属性。
- 所有NPC的设置在 `game_objects.py` 中的 `FriendlyNpc` , `Tutor` , `Bonfire` 和 `Healer` 类中实现。

3. 简单敌人

- 场景中会随机刷新出一种简单敌人：骷髅兵，随着战斗房间的深入，骷髅兵的攻击力和血量会逐渐提升。
- 当玩家靠近骷髅兵时，它会加速向玩家移动，并且具有向左和向右对应的移动动画。
- 当玩家出现在骷髅兵的攻击范围内，它会以固定频率向玩家发起进攻，每次攻击对玩家造成**10点伤害**，该敌人具有对应的攻击动画。
- 当骷髅兵的**血量小于等于**0时，它会立刻原地死亡，从场景中消失，并且具有死亡动画。
- 骷髅兵的设置在 `game_objects.py` 中的 `Skeleton` 类中实现

4. 特殊敌人

- 第十一个战斗场景中会刷新出特殊敌人：boss。boss具有相对高的血量，击败boss后游戏取得胜利，进入胜利界面。
- boss具有三种攻击方式，具有对应的攻击动画。
  - 当玩家站在与boss房的边缘时，boss会瞬移到地图的另一边，并且发射一排具有一定伤害的子弹。（可靠翻滚躲避）
  - 当玩家站在地图中间时，boss会在玩家周围生成一些向玩家发射的子弹。（可靠翻滚躲避）
  - **当boss的血量降低到50%以下时**，boss会在玩家周围召唤**两个骷髅兵**。
- 当boss**血量小于等于0**时，boss死亡，**游戏结束**。

2.3 游戏机制

1. 核心机制

- 玩家通过操纵主角在战斗场景中通过移动、翻滚以及攻击击败敌人不断获得金钱并且进入更深的战斗房间，房间的难度会越来越高。当通过所有战斗房间后玩家取得胜利。期间玩家需要往返休息站通过与npc的交互升级自己的基础属性，治疗自己的血量，从而更轻松得击败更多的敌人并取得胜利。

2. 碰撞系统

- 程序通过 `pygame` 内置的碰撞检测方法来检测玩家与实体间的碰撞。全部的碰撞检测全部由`game_collections.py` 中的 `EntityLike` 类实现
- 当玩家与互动物品、npc、敌人、等实体碰撞时，碰撞检测方法会向事件队列发送相对应的事件，在事件队列中被相继处理。

3. 资源系统

- 玩家击败敌人就可获得金币，金币可在大篝火处消耗从而来提升玩家的基础属性，即血量，攻击力和体力。
- 玩家的血量也是一种资源，玩家需要时刻分配好自己的血量，确保自己不被怪物击杀的同时赚取足够多的金钱返回休息站恢复血量。
- 资源系统的设置在 `game_collections.py` 中的 `ResouceManager` 类中实现。

2.4 大语言模型智能体系统

1. 对话系统

- 与引领者，大篝火，医师的交互均为对话。其中医师的对话系统运用了大语言模型智能体系统，玩家可以自由地与医师沟通，他会捕捉关键字如"need heal"为玩家治疗血量，医师的设置在 `game_objects.py` 中`Healer`类中实现。

2. 决策系统

- 与引领者，大篝火，医师的对话均包含决策系统。
  - 与引领者对话可以选择是否听取游戏机制。
  - 与大篝火对话可以选择升级还是进入战斗房间。
  - 在大篝火处的升级系统可以选择升级哪一项玩家的基础属性。
  - 与医师的对话中可以选择是否让他帮我们治疗。

2.5 游戏性

1. 主菜单

- 我们的游戏有三种菜单，开始菜单、失败界面与胜利界面。
- 开始菜单为游戏开始时出现，按下任意键进入游戏。
- 失败界面为玩家死亡时出现，按下**R键**重新开始，按下**esc键**游戏结束。

- 胜利界面为玩家通过所有战斗房间后取得胜利出现，按下**R键**重新开始，按下**esc键**游戏结束。

2. 背景音乐

- 我们的游戏在主菜单、失败界面、胜利界面、城市场景、野外场景与 Boss 房都会播放不同的BGM，BGM的实现在不同的`ScenceLike`对象中实现。

2.6代码

仓库中`assets`，`maps`均为素材文件。用编辑器打开`SI00B_Project`后运行`main.py`，或直接通过在 SI100B_Project 目录打开终端并输入`python main.py`运行，即可进入游戏。程序采用面向对象编程的编程方式，代码中难以理解的地方全都添加了注释。下面罗列程序中的各个文件的相应作用：

- `base`该文件夹包含了本次课程提供的模
- `game_collections.py`该文件定义了
  - 实体类`EntityLike`，类对象具有贴图以及碰撞框，并可以进行碰撞检测。
  - 状态类`State`， 主要用于管理动画和角色的状态，负责动画每一帧的刷新，并且检测传输动画队列的可被打断性，循环次数以及每一帧角色的状态（如无敌）。
  - 动画角色类`AnimatedSprite`，继承了`EntityLike`的属性，类对象包含一个`imageset`用于储存角色的所有动作的动画，包含一个`State`类对象管理角色的状态。
  - 贴图类`Tile`，继承了`EntityLike`的属性，类对象用于显示地图以及具有碰撞体积的障碍物。
  - 场景类`SceneLike`，主要提供相机坐标，图层控制，场景的进入与退出以及BGM。类对象用于控制相机的绘制，图层的层数，角色被传送到哪一个场景中以及不同场景对应的BGM的播放。
  - 文本类`TextEntity`， 类对象用于给交互系统（如升级系统，对话系统，菜单）提供文本。
  - 资源管理器类`RecourceManager`，用于管理全局资源（如角色金币数）。
- `game_constans.py`该文件定义了所有代码的`uuid`，用于传入监听者队列中。
- `game_objects.py`该文件定义了所有游戏中出现的对象，均属于`AnimatedSprite`的子类。
  - 玩家类`Player`
  - 敌人类`Enemy`
  - 骷髅兵类`Skeleton`
  - boss类`Boss`
  - 友好NPC类`FriendlyNPC`,包含

        - 引领者类 `Tutor`
        - 篝火类 `Bonfire`
        - 医师类 `Healer`，与医师的对话包含大语言模型智能体系统。
- `scene.py` 该文件用于管理游戏中的所有场景，其中包含
    - 地图生成器类 `MapGenerator` 用于随机生成不同难度的战斗场景。
    - 休息站（初始房间）类 `Home` 用于显示休息站（初始房间） 。
    - 主菜单类 `MainMenu` 用于显示主菜单。
    - 游戏结束类 `GameOver` 用于显示游戏结束。
    - 门类 `Door` 用于显示场景中的门。
    - 小篝火类 `BonfireDoor` 用于显示场景中的小篝火。
    - 场景管理器类 SceneManager 用于管理和切换场景。
- `main.py` 该文件是游戏主程序，包含了游戏主循环，负责运行游戏。

---

3. 创意

3.1 即时动作游戏

- 角色与各种NPC以及敌人有丰富的动画系统，攻击翻滚和移动的即时反馈给予玩家更强的体验感与交互感，敌人的攻击系统更给玩家带来了趣味性和挑战性。

3.2 大语言模型的加入

- NPC医师的对话系统由大语言模型驱动，玩家可以和其自由交流，也可以让他帮助恢复游戏内血量，给游戏带来了更高的自由度和不确定性。

3.3 随机地图生成器

- `MapGenerator` 地图生成器类可以生成随机地图，地图上会随机分布障碍物(灌木，树木等)和对应场景的地砖，给游戏增加的探索性与不确定性。

3.4 等级系统

- 每次玩家进入战斗场景时，怪物的等级都会增长，这会使它们的各个属性值都得到增长，并增加它们掉落的金币。等级系统旨在增加游戏的挑战性，促使玩家升级不同的基础属性，注意管理自己的血量资源与金币资源，不同属性的分配也增加了游戏的成长性和决策需要。

3.5属性显示

- 玩家的血量和体力都会人性化地显示在窗口左上角，金钱数放在了左下角以方便玩家进行资源管理。血量和体力均以进度条的形式展示，增加了直观性。

小组贡献

- 叶顾燊：角色，NPC，大语言模型，敌人，场景，BGM，菜单，测试，美工
- 余在畅：相机，资源，撰写report和README，测试
- 梁景育：大语言模型，测试