项目报告

周晴阳，赵麒安
2025 年 1 月 12 日

1. 引言

本项目为 SI100B 课程的最终项目，项目文件已上传到
https://github.com/yang43616768/The_Best_One_team/
本项目分为五个阶段，开始菜单，室外关卡，室内关卡，最终章，以及结束菜单、
这是一个类 rpg 式的地图探索+物品收集+npc 互动的游戏,核心战斗机制为回合制卡牌战斗玩家初始拥有一定血量,攻击,防御,行动力，在玩家与 NPC 的战斗中，需要合理的选择打出的攻击防御牌来击败对方。玩家可以通过战斗或购物获得物品，提升属性，让自己变得更强，也可以在与 NPC 的对话中逐步掌握剧情，体会了解事件真相的乐趣
在玩家击败每一阶段的一些 NPC 并获得他们对应的独特掉落物后,玩家将能够通过传送门前往下一阶段。当然，在准备充分前，你仍然可以继续探索。

2. 项目实施

2.1 场景

2.1.1 场景简述

在本游戏中共有五个场景：
1. 开始菜单：含一张开场图与 BGM，玩家按下任意按键后开始游戏
2. 草地关卡：含 5 个 NPC 与许多的墙体，砖块路和屋顶，以及传送门，以及 BGM，背景为随机生成的草地图像，玩家可以自由探索该场景并与 NPC 互动，最后前往传送门
3. 塔楼场景：含 4 个 NPC 与许多的墙体，砖块路和屋顶，以及传送门，以及 BGM，背景为随机生成的砖墙图像，玩家可以自由探索该场景并与 NPC 互动，最后前往传送门
4. 最终章场景：含 2 个 NPC，以及 BGM，背景为给定的决战背景，玩家在此与他们决战，最后前往传送门
5. 结束菜单：含 11 张 CG，以及逐渐显现的“Thanks for playing!”文字，以及 BGM

图 1：草地场景，以及半透明屋顶

图 2：塔楼场景

图 3：结束 CG

2.1.2 大型场景及其生成原理

我们游戏中有三个主要大地图场景，分别对应室外关卡、室内场景、最终章场景，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。

为了得到地图，我们使用 Map.py。在这里，我们定义了 Map 类用于“生成”地图，我们先得到生成随机数组，并引用 setting.py 中规定的瓦片图像，按照数字与瓦片图像的一一对应关系，我们就得到了一个储存着整个地图的瓦片的列表，而这个列表将在 Scenemanager.py 中被用到。

场景渲染分为 3 块：地图、物体的渲染、摄像头窗口的移动

1.地图

为了得到一张可使用的地图我们的游戏分别用 3 个用 render 方法，引用之前得到的储存着整个地图的瓦片的列表，以此填满整个被分为多个格子的“地图”贴纸，同时，我们在地图上还绘制的合理的路用于引导玩家，然后将整个地图“贴纸”渲染到窗口上。

2.物体的渲染

为了能够在这个地图“贴纸”上渲染出不同的物体，如 player，NPC，墙，对话框等等，我们创建了 location 函数作为可调用的方法渲染要渲染到窗口上的图像、物体。其原理是以得到的地图的左上角为原点，建立一个坐标系，在坐标系上渲染各种基础性物体，同时会判定玩家的状态，以此为依据决定是否渲染对话等额外性物体，再将这些物体组成的“临时”图片渲染到窗口上。

3.镜头窗口的移动

为了实现这一点，我们创建了 update_date 函数。镜头会随玩家移动而移动，以防止玩家走出镜头，且镜头移动到边界后不会继续移动以防止显示黑色的“虚空”，且第一点中渲染的地图和第二点中渲染的各种物体不会随镜头移动而移动。

原理是在 update_date 函数中，我们对镜头的坐标进行判断，在玩家靠近镜头边缘时，通过坐标判定来实时刷新镜头，玩家这一次移动的位移设为（dx，dy），镜头会获得一个相同的（dx，dy）的偏移量，这样 player 看起来在镜头上是不动的。

而地图和物体可不能随镜头移动而移动，为了实现这一点，我们在之前的 render 函数和 location 函数中，无论是地图或是物体的坐标均在原有基础上减去了 camera 的坐标（当然，对话框这些东西则是会固定在镜头上），这就使得镜头移动时这些物体会和地图一样“固定”在原地。

这就实现了镜头移动的效果。

当然，还有对镜头坐标与地图边界的判断，防止镜头移动到地图外显示出黑色的“虚空”。在镜头移动到边界后，玩家继续朝原方向移动只能产生 player 自主移动的效果。而且，玩家若不靠近镜头边缘，仅仅在镜头中央移动也不会使得镜头移动。

2.1.2 可互动的静态场景

Transparent 类：

可变 alpha 值类，作为屋顶使用，通过 check_transparent 函数监测是否与玩家重合，若重

合则提高透明度以实现“玩家在屋顶下能看到屋子内的场景，玩家在屋子外只能看到屋顶”的视觉效果，渲染通过 scenemanager 实现

Portal 类：

通过 check_telepotation 和 check_item 函数实现当玩家与自身重合并按下交互键 e 时,若玩家库存中所需的剧情物品不够,则调用 tp_failed 函数弹出白色窗口提示玩家还缺少哪些物品，若玩家已经具备足够条件，则激活自身的 tp_succeeded,并在主循环中监听此特征值并跳出当前循环切换到下一个 stage

图 4：传送门

Wall 类：

用于作为障碍物判定碰撞体积。我们在这里创造了一个继承 pygame 中的一个精灵组的 Wall 类，其显示为白色半透明(可看到地图），在 Scenemanager.py 的 location 函数中被使用，这使得 wall 可在 stage.py 中被 location 函数以精灵组的形式被渲染(可定义坐标,长宽)。

2.2 角色

图 5：玩家

2.2.1 主要角色 Player

在本游戏中，Player 主要有以下互动行为：

1. 根据坐标在场景中渲染（见镜头与渲染部分）
2. 通过 update 函数实现按 WASD 在场景中移动和墙体碰撞监测:

向着对应方向每刻移动等同于玩家速度大小的距离（这会涉及到场景相机的迁移，在本文其他部分会详细说明），同时在每一个游戏刻内若检测到玩家刚才的移动使玩家将要碰撞到墙体的碰撞体积，执行回退一步的操作来实现墙体碰撞功能。

3. 按 I 键打开/关闭物品背包：

当玩家按下 I 键时切换背包的开闭状态，在 show_inventory 函数中通过预设好的背包背景

（淡蓝色半透明方块）并读取玩家此时的属性和库存将玩家的物品和各种详细信息打印在屏幕上。为了防止游戏后期过多物品显示不全，加入了使用 pgup/pgdn（通过测试，部分电脑还需要按住 Fn 键再按上述按键）来改变文字 y 轴偏移值从而实现视觉上的翻页效果。

4. 获取物品：玩家的 add_item 函数在玩家通过购买或击败 NPC 获取新物品时被调用，效果是为玩家的属性值加上对应物品的属性。

通过 draw_item_message 在当前游戏窗口的中心位置短暂的显示刚才获取的物品的名　字，达到提醒玩家获得新物品的功能

在 stage1 函数下，有两行全物品指令供测试用。

关于玩家与 NPC，Transparent，Portal 类的交互，在他们对应的模块将会详细阐释

图 6：NPC 贴图

2.2.2 可互动实体 NPC：

在本游戏中，NPC 主要有以下互动行为：

1. 根据坐标在场景中渲染（见镜头与渲染部分），每个 NPC 都拥有独立的场景内像素 小人贴图

在与玩家 Player 的欧式距离小于设定数值时，启用所有交互，包括：

2. 聊天气泡循环显示：通过读取 setting.py 中对应 NPC 的聊天气泡列表，每个 NPC 在初始化后都具有独一无二的气泡文本内容。通过 pygame.time 调用当前时间，在主循环中使用 switch_bubble 函数每隔 120 帧切换一次气泡内容并调用 scenemanager 相关函数进行渲染（见 scenemanager）

3. 对话，战斗，商店窗口：在主循环中监听玩家的交互行为，将非交互状态下的所有输入交由 npc 的 handle_input 函数进行处理：使用 self.dialogue/fight/buy_active 标记自身的交互状态，当自身未被激活时，根据对应的功能键 e/f/b 激活交互状态，在主循环中监听到对应交互状态启用时，调用相关函数

3.1 对话相关：通过抓取键盘输入作为对话输入，Esc 为退出对话指令（即关闭对话激活），enter 为提交文本。所有 NPC 通过 setting.py 中预设好的人物性格，剧情内容和游戏通识内容作为 openai 预设输入，并将玩家的对话输入提交至 openai 得到反馈，将问答文本整理在 dialogue_history 列表中并在 scenemanager 中根据 NPC 激活对话状态用调用 draw_dialogue 函数，根据对应的 NPC 聊天背景（同样每个 NPC 有个性化的背景）和对话内容绘制对话窗口。同时，为了防止对话次数过多导致的显示问题，同样也加入了基于监听 pgup/pgdn 的上下翻页功能。

图 7：NPC 对话窗口

3.2 战斗相关：监测到玩家在自己附近按 f 且自身未被击败时，激活战斗状态，并在主循环中根据激活状态绘制对应的战斗窗口：每个回合由以下事件构成：先在 handle_input 函数中监听 q 键和 e 键的点按次数并计数，当次数到达回合行动上限时调用 fight_calculate 计算玩家和 NPC 下一回合的血量值。回合结束，回合数+1，当玩家血量小于等于 0 时触发 fight_failed 函数，退出战斗窗口并初始化所有战斗参数（和打开战斗以前不会有任何区别），当玩家血量大于 0 且 NPC 血量小于 0 时触发 fight_succeeded 函数，将 npc 设置为 defeated 并结算战斗奖励同时初始化战斗参数；战斗窗口的绘制由 scenemanager 根据战斗激活状态调用 draw_fight 函数执行：在屏幕左右分别是玩家和 NPC 的贴图，血量数值，红白条血量百分比图像，中央会显示回合数，玩家的下方会根据 q,e 键的监听次数显示该回合已经打出的攻击牌数目和防御牌数目（为了可视化，采用两张画有盾牌和剑的贴图，贴图数目代表打出的牌数）同时界面正下方有提示文字引导玩家攻击与防御的按键是 e 和 q

战斗部分有独立背景音乐，见背景音乐部分

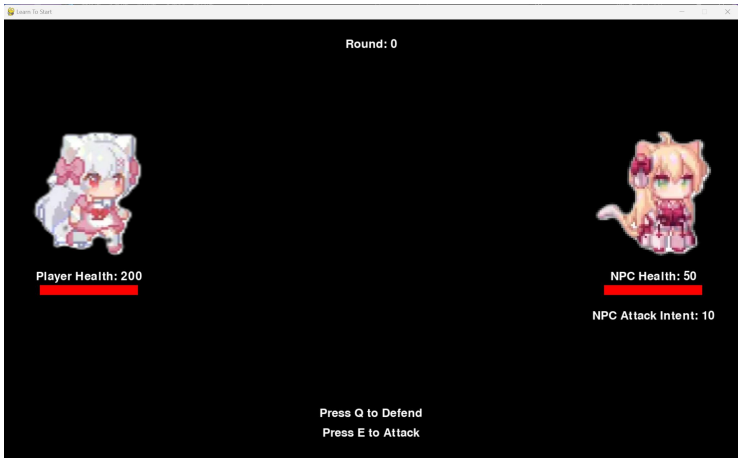

图 8：战斗窗口

3.3 商店相关：监测到玩家在自己附近按 b 时激活商店状态，并在主循环中根据激活状态绘制对应的商店窗口：每个 NPC 读取 setting.py 中自己的商品列表作为商品，调用数字键 1，2，3 作为购买对应物品的快捷键，同时在窗口上绘制半透明的黑色商店背景和物品文字信息，按照“快捷键--物品信息--价格”的结构从上到下显示所有商品，同时当玩家购买了上层物品时会重新排列，即若购买物品 1，则物品 2,3 的快捷键和显示行数将变为 1 2

同样是使用 scenemanager 根据商店激活状态调用 draw_buy 函数进行绘制

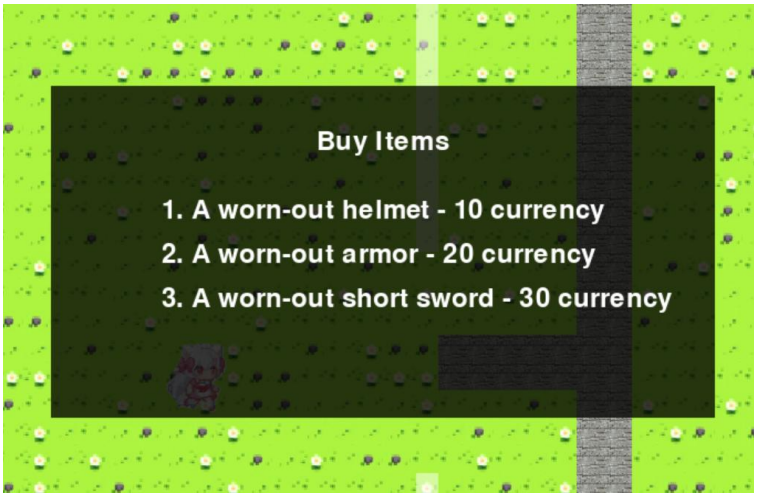

图 9：商店窗口

2.3 游戏机制

2.3.1 大地图

我们通过渲染一张长宽为窗口的两倍大小的地图并在其中布置 NPC 和各种墙体障碍等环境物体来营造可供探索的大地图，玩家可以自由移动并寻找 NPC 进行交互

2.3.2 NPC 交互

战斗系统，商店系统与对话系统是本游戏的核心系统，共同构成了主要的游戏内容。
战斗系统：
本作严谨的测试了流程中 NPC 的战斗数值和独特的掉落物数值,合理的数值设计让玩家不论是“速通流”的依靠技术击败对手还是“养成流”的轻松发育战胜对手都能游刃有余。
商店系统：
大多数 NPC 都有三个可供购买的商店物品，这为玩家提供了更多的选择，让玩家可以通过合理的购买与挑战顺序来让探险的过程更加轻松
对话系统：
每一个 NPC 都有独一无二的对话设定，在与 NPC 的对话中可以感受到剧情的趣味

2.3.3 购物系统

在战斗中,玩家可以获得战斗掉落物品以及金币,并使用金币购买商店中的物品来强化自己。

2.3.4 物品系统:

游戏中所有物品通过 Items 类定义，初始值包括数值列表 statistics 和描述 discription,其中 statistics 包括对玩家 attack,defend,health,moves 的更改量,物品在玩家库存中的添加通过玩家的 add_item 函数实现，而 discription 是我们创作时对各个物品的吐槽和简介，在打开 i 键背包时能够看到所有物品的 discription

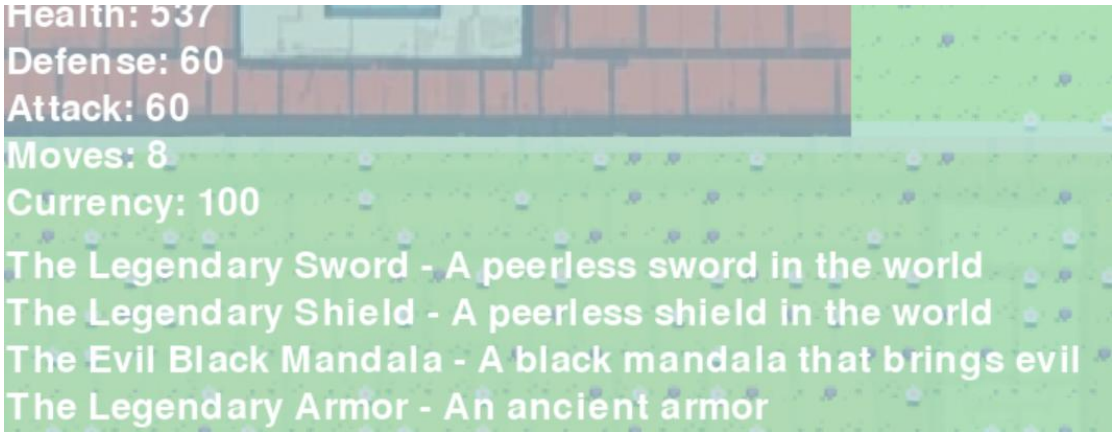

图 10：物品背包

2.4 游戏性

2.4.1 UI 设计

在战斗面板中我们使用了很多可视化的元素来让玩家直观地感受战斗的情况,便于玩家更好的做出战斗决策

2.4.2 美术风格

本游戏中尽量选取美术风格一致的素材并通过一系列后期加工来让其贴合游戏的画风,并形成一套独立有趣的世界观

2.4.3 BGM

在五个阶段中，我们有四首不同的 BGM，同时战斗中还有一支 BGM

2.5 代码

本游戏依赖于 my-game-project 文件夹运行 使用 vscode 打开该文件夹,在工作区直接点开 main.py 并右上角运行即可打开游戏，或直接通过在 my-game-project 目录打开终端并输入 python main.py 运行，即可进入游戏。

1.bubble.py 定义了聊天气泡类，用于处理 npc 气泡的渲染生成和刷新

2. items.py 定义了物品类，用于对玩家属性进行修改，以及在物品背包中显示对应物品名称和详细描述
3. main.py 调用所有 stage 函数来确定游戏运行顺序，启动游戏
4. map.py 用于生成地图瓦片序列
5. npc.py NPC 类用于处理所有玩家与 npc 间的交互行为以及 npc 的渲染行为，包括战斗场景的判定和渲染，商店场景的判定和渲染，对话窗口的判定和渲染，聊天气泡的渲染和刷新
6. player.py 玩家类，用于处理玩家的移动，物品背包界面的开启和渲染，拾起物品的提示
7. portal.py 传送门类，用于检测玩家是否满足传送条件，绘制传送失败菜单，以及进行 stage 的切换
8. SceneManager.py 游戏帧率的刷新，物体在窗口上根据镜头位置渲染，地图的渲染和路径的渲染和生成，根据 NPC 的激活情况调用对应的绘制窗口，根据 Player 位置刷新镜头的位置，处理图层冲突
9. setting.py 游戏各个参数的基础设置
10. stage.py 定义了所有 stage 通用部分并在初始化阶段生成墙体，NPC，BGM，传送门，屋顶
11. transparent.py 根据玩家与自身位置关系调整透明度，用作屋顶
12. wall.py 管理墙体类，用于生成，判定和渲染墙体

3. 创意

3.1 路径与屋顶

通过各种障碍物的视觉效果引导玩家的探索路线，指引玩家发现隐藏的 NPC，这增加了探索的趣味性，也给玩家带来惊喜感。同时，在一张地图内，玩家的探索路径是自由的，也就是说，可以随意选择挑战 NPC 的顺序与购买物品的顺序，这大大增加了本作的自由度

3.2 战斗策略和可变的 NPC 意图

在战斗中，我们秉承着合理化的数值设计，不让游戏难度过于困难同时保留一定程度需要玩家思考的战斗。NPC 的战斗意图列表这一设计便是最好的体现：通过多变的 NPC 意图，让玩家养成观察对手攻击意图来决定攻防出牌决策的习惯，领略数值之美

3.3 养成机制

为了防止线性流程造成的枯燥感，我们增加了许多的物品供玩家获取，在获取物品变得更强的过程中，玩家可以更轻松的击败对手，同时也能阅读到许多有趣的物品描述

3.4 剧情文本

在游戏中，有多处预设的游戏文本，如 NPC 的聊天气泡，对话设定，物品的描述，这丰富了游戏中的直观体验

图 11：NPC 的聊天气泡