

# 项目报告

2024 月 1 月 12 日

(edit2025.02.09: 移除了个人信息, 但由于是 pdf 转 word 所以可能部分格式会有问题)

## 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/azaneNH37/LibraryOfRedundancy>。

我们制作了一款 Roguelike 类型的卡牌策略游戏, 主要灵感来源于 MoonProject 开发的《废墟图书馆》(以及融合了其他部分游戏中的元素), 并且在较为忠实的实现了原作的几乎全部功能的前提下, 融合了 rouge 模式, 使游戏角色可以在游戏过程中成长, 同时利用 LLM 模型(大概率的)获取你想要的卡牌, 以此构建属于你自己的卡组 and 风格, 战胜强大的敌人。(并挖掘世界的真相?)

愿您找到想要的书

## 2 项目实施

### 2.0 分工

(NH37 似乎并不打算把这次的 report 当正式的论文写)

(本部分由 NH37 写的, 所以具有完全的主观性, 哼)

(但反正审核没意见~)

#### @NH37

负责了整体架构的搭建, 渲染部分的全部代码, 肉鸽系统的实现以及 LLM 部分的思路构建(?), 以及永远解不完的资源包(海澜之包舔三遍, 每遍都有新发现), 还有 report 的撰写(具体是哪部分从文风就能看出来罢 by bmwang)

负责模块: AudioSystem, AnimeSystem, Data, EventSystem, RenderSystem (它是最大的!), RougeSystem, SceneSystem, (GameControlSystem, GameDatamanager (其中一些(大部分)) GameInitialSetting, LLMStorage)

#### @bmwang

负责了全部战斗数据部分的代码, 以及本 report 的架构, 以及并不勤劳的测试员?(但是在终端玩废墟图书馆真的很辛苦欸)

负责模块：ReceptionSystem（其中曾经有一个长达 3k 行的文件）

## @2Bpencil

负责了 LLM 部分的全部代码，以及勤劳（？）的测试员，以及部分美术资源的提取。哦对了，还有驯服 llama3.2（事实上，一个 3B 的小模型还能干什么呢）。

负责模块：LLMSystem（包含情感强烈的 prompt 部分）

（以下内容（2.1-2.5）仅为踩分点展示，与本项目核心游戏逻辑存在一定差异，因此不对该部分游戏性做保证）

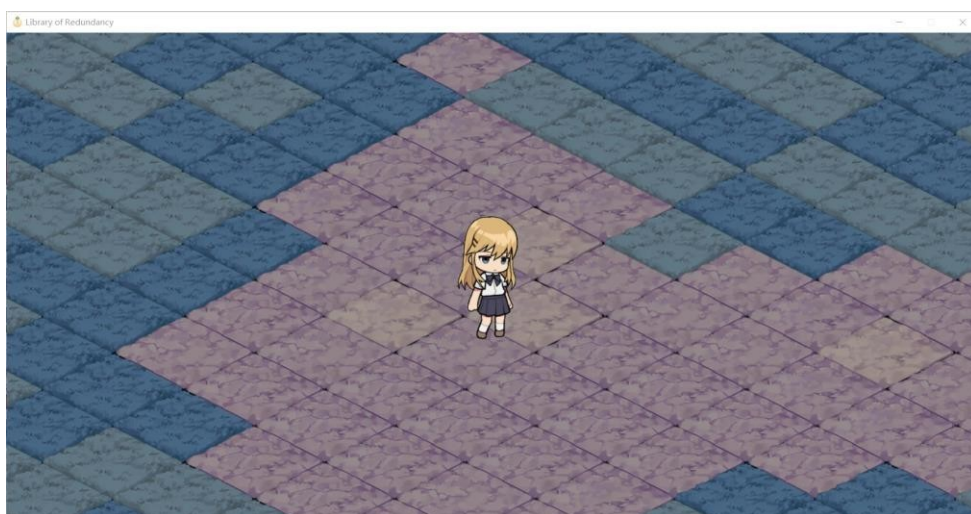
## 2.1 场景（Scene 15%）

### 2.1.1 One large scene with the camera following （10%）

根据踩分点要求，本项目仿照游戏《超时空方舟》的地图设计制作了大型地图。鉴于在制作该部分时，本项目的底层架构已制作完毕，因此此处仅能简要介绍基于本项目底层之上的架构（关于本项目底层架构参见下文）。

基础逻辑：对于每一个位于 UIMap 对象下的 UIMapObject，为其赋值一个地图坐标。在每个 DataFrame 中，读入键盘事件操控 UIMap 的 Camera 字段修正其注视的地图中心坐标；在每个 RenderFrame 中，遍历 UIMap 的子物体，计算所有 UIMapObject 与注视中心的相对坐标，若位于 screen 范围内则绘制。（考虑到地图大小，本项目并未对 UIMapObject 基于地图坐标进行分块处理，但依然能保证较高的流畅度）（关于地图边界：本项目并未对地图边界做阻挡，因为有彩蛋 这也是计划的一部分.jpg 绝对不是因为懒（逃））

下图为角色位于地图时的待机状态（露西可爱捏）



### 2.1.2 One interactive object (e.g. obstacle) (5%)

早上好，中午好以及晚上好，我的朋友。

为了拿到这 5 分只好“请”坎诺特先生出来客串一下 NPC，坎诺特先生同时包含了以下功能：One interactive object, A friendly NPC, Collision system, Dialogue system, Decision system（哇 好多分）

以下是解读哦~~

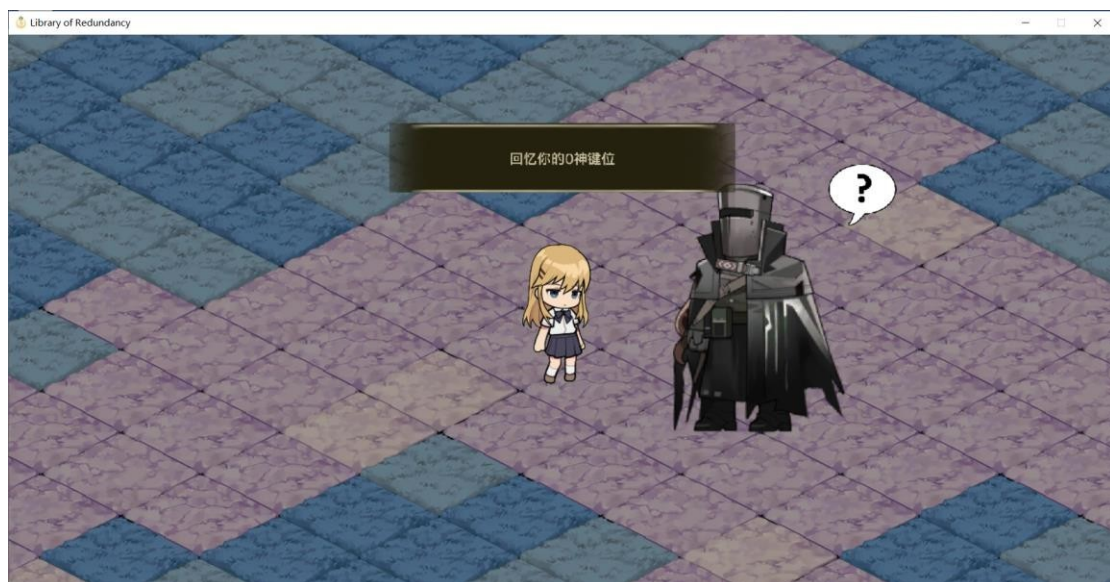
**One interactive object:** 当角色靠近坎诺特先生的时候，坎诺特头顶上会冒出问号，这个时候就可以与他进行交互，进入商店（广义上的商店）进行操作（怎么操作呢？游戏会告诉你的~）

**A friendly NPC:**（因为项目没有非敌对 NPC 了，所以你不能抢他的商店哦~）他都给你免费发角色升级了，还帮你跟笨蛋 GPT 斗智斗勇生成卡牌了，还不是 friendly NPC 么？快说谢谢坎诺特先生！（截图在 2.2.2）

**Collision system:** 本项目忠实还原了《超时空方舟》的 2.5D 设计，角色不能进入（？）坎诺特先生，但可以在两者碰撞箱不接触的情况下在他的上部或下部经过；并且当角色位于坎诺特先生上部时，角色会显示在坎诺特先生的后方；反之亦然。（请看 VCR↓在 2.3.2）（为了演示（包括最终成品）这一点适当放大了双方的碰撞箱）

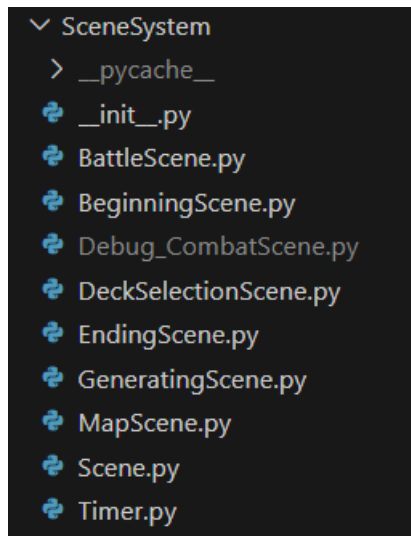
**DialogSystem s DecisonSystem:** 参见 2.5.1s2.5.2（坎诺特先生其实是司书哦 点头.jpg）

下图为角色靠近坎诺特先生时触发的提示



### 2.1.3 ≥ Two different scenes

哦天哪 这个项目到底有多少场景！（详见创意部分）



## 2.2 角色（Characters 15%）

（此处所指的角色是为了拿分写出来的角色，关于本项目游戏性体现的角色参见创意）

### 2.2.1 A main character (5%)

（扭曲之地传到任何奇怪的地方都很合理罢）

把露西抓过来当主角色，主要能力是走路（？）（绝对不是因为 SD 小人的走路动画便于抄袭哦）

下图为帧（反正一跑就看到动画了 不放.gif）



### 2.2.2 友好 NPC

还是坎诺特先生哦~ 不认识他的可以看 2.1.2 哦

下图只是在炫耀坎诺特先生给的两个金被动



我去！随手 **roll** 出来个三黄 换图换图（并不是同一局哦）



下图只是在展示坎诺特的突然深情（但能不能别塞闪避骰了）





结果还是塞了两个（）（关于卡牌生成参见 llm 部分和创意哦）

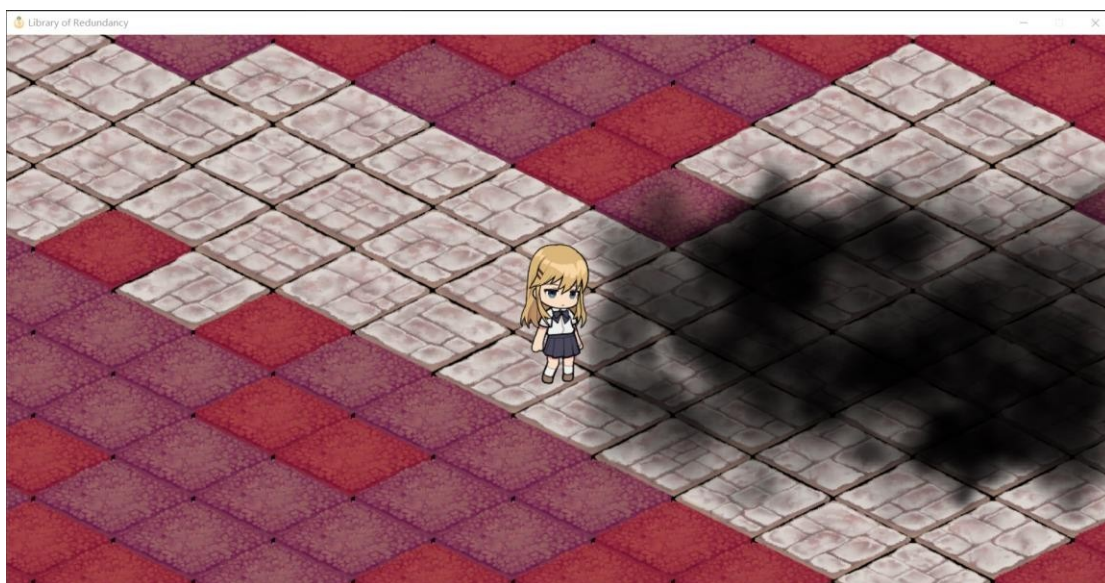


### 2.2.3 A simple enemy (5%)

哦 天哪 我们的敌人似乎都不太简单 但如果你是在说这个黑雾 它确实挺简单的

当角色深陷黑雾时，会进入 CardSelectionScene，并最终进入 BattleScene 战斗（人话：撞到黑雾碰撞箱）

然后你就可以打八重缪罗威尔红扭曲阳了！



### 2.2.4 A special enemy

并非“A”

见创意

## 2.3 游戏机制 20 分

### 2.3.1 核心机制 10 分

玩家通过操作主要角色移动来触发各类事件，在友好 NPC 附近按口（欸这个字符怎么不见了呢？ *by NH37*）（是 F 啦，游戏里有提示的）可与其交互，与代表敌方的黑色迷雾碰撞会触发战斗。（以上是用来踩得分点的哦 但不可否认确实让游戏性得到了一定提升 *by NH37*）玩家需要击败敌人以进入下一层，每一层中玩家都可以通过友好 NPC 获得一定程度的提升。包括新的可用卡牌，新的被动能力，已经及（抓到错别字了哦 *by NH37*（你不也没改对 *by bmwang*））基础数值的提升

战斗采用回合制，以骰子拼点为核心机制，以拼点胜利打出的伤害为一般输出手段，需要玩家合理配置与运用从 NPC 处获得的卡牌，以获取战斗的胜利。每回合开始后，需要按一下空格以确定双方的速度以及敌方的策略，然后可以配置己方的策略。点击一颗己方的速度骰，在下方选择想使用的书页，再点击想打的敌方速度骰即可完成一颗速度骰的配置。右键点击速度骰可以取消其配置。配置完成后再按一下空格即可开始战斗。具体的战斗机制比较复杂，我们设计了很多类型的骰子，很多的特殊效果，很多的 buff 与 debuff 以及很多的被动能力，详见创意部分 3.1.1。如果不想/不会自己配置策略，也可以按 P 键进行自动装卡，不保证其智能（保证其不智能）。根据背景设定，本游戏中一场战斗也被称为接待，一回合也被称为一幕。

（看不懂的会摁空格和 P 就行了 但你大抵通不了关（） *by NH37*）

### 2.3.2 碰撞系统 5 分

描述见 2.1.2 下面是一大堆截图







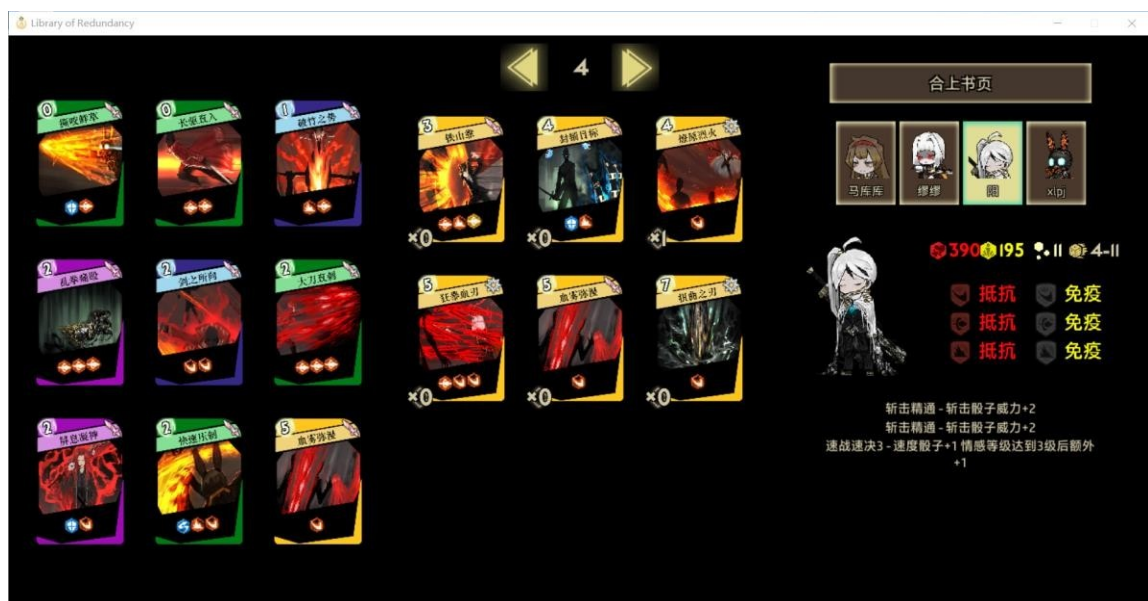
哦天哪，我该怎么表现你只可到此，不可向左一步



### 2.3.3 资源系统 5 分

本项目并没有传统意义上的金币啊之类的资源，你所要管理的资源是你的卡牌池，以及对于角色天赋和升级的合理选择。（根据我对这一条理解，只要搞个任意全局数据存储就行了）

展示展示牌库（王之宝库）





## 2.4 游戏性（指具有游戏性质的 **by NH37**）5 分

### 2.4.1 Main Menu 3 分

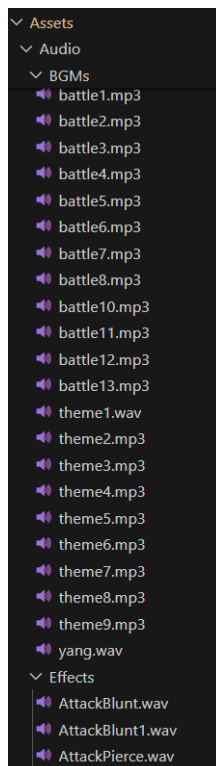
主界面有很多动画和彩蛋哦（搓手手）

（开发中画面 不代表最终效果 主要指分队效果）



### 2.4.2 BGM 2 分

如图 好听的 **ender lilies** 的原声带真的是神罢



## 2.5 LLM 的运用

### 2.5.1 LLM 对话

在游玩过程中，玩家可以通过对话框与 LLM 对话，LLM 会根据玩家输入的话返回一段有趣或者有哲理的话。（偶尔发癫。 **By NH37**）

为了使 LLM 的说话风格较为冷淡但幽默，程序会让 LLM 阅读一个语料库，并让 LLM 学习并模仿语料库中的说话风格。

LLM 对话是通过 LLMSystem 文件夹中的 ChatWithGPT.py 模块来实现的。

### 2.5.2 LLM 策略

LLM 策略主要体现在让 LLM 生成游戏卡牌。具体来说，在玩家需要从 NPC 处获取卡牌时，玩家可以在对话框输入一段话，LLM 会根据玩家输入的一段话实时生成若干张卡牌（这些卡牌不是预先制作的）。

LLM 策略是通过 LLMSystem 文件夹中的 KeywordMatch.py 和 InteractionWithGPT.py 两个模块来实现的。前者用于从玩家输入的一段话中提取关键词，后者用于让 LLM 根据关键词生成卡牌。（至于生成的卡牌到底有多吻合关键词 我只能说 根据长期测试整体吻合度良好 但不同批次生成中方差较大 只能说是 **3B** 小模型魅力时刻了 **by NH37**）

## 2.6 代码 5pts


**(NH37 正在疯狂补充这段内容)**

可读性是大大滴有的，设计也是大大滴有的（省流， **by NH37**）

仓库中 SI100B\_LibraryOfRedundancy 为游戏文件夹。用编辑器打

开

SI100B\_LibraryOfRedundancy 后运行 main.py，即可进入游戏。

程序采用面向对象编程的编程方式。代码总长度非常非常非常（由 **NH37** 补充）长，部分难以理解的地方添加了注释，所有的类与函数都采用了清晰且规范的命名，能够很好的展示其作用。（真公式  **by NH37**）

下面罗列项目根目录下的各个文件的作用：

- 1、GameContronSystem.py 游戏 **Scenes** 总控
- 2、GameDataManager.py 全局数据存储
- 3、GameInitialSetting.py 初始化
- 4、LLMStorage.py llm 部分的隔离控制
- 5、main.py 该文件是游戏主程序，进行整个系统的初始化，包含了游戏主

循环，负责运行游戏。（进行了个锤子 纯套皮 by NH37）

下面罗列根目录下各个文件夹的作用：

（为什么不是文件呢？）

	261 个文件，25 个文件夹
类型:	类型均为 文件夹
位置:	全部位于 E:\Project\SI100B_FinalProject_LibraryOfR
大小:	1.70 MB (1,791,299 字节)
占用空间:	2.11 MB (2,215,936 字节)

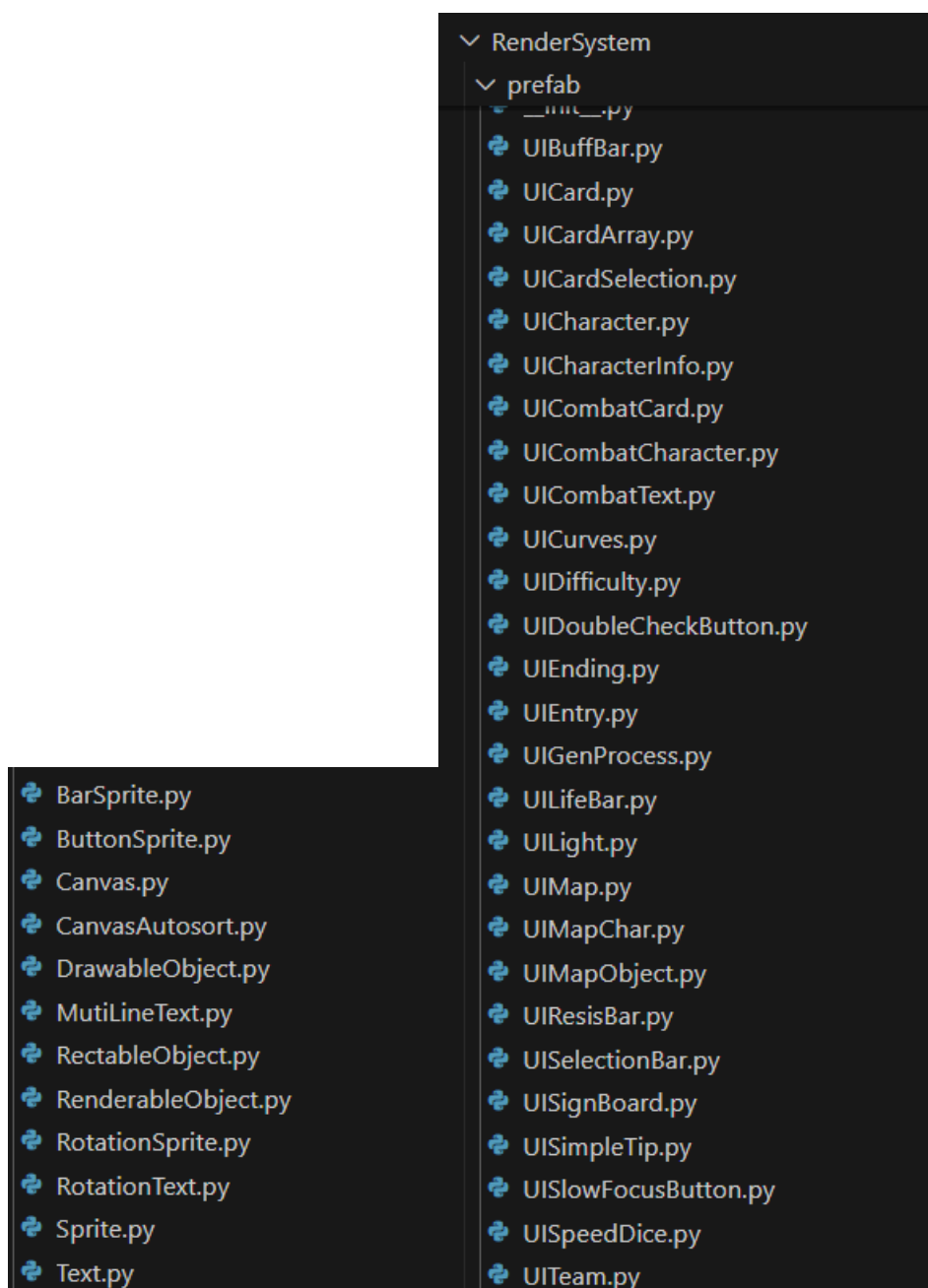
会累死的。

### 2.6.1 RenderSystem

渲染部分的核心模块。

本项目设计思路与 Unity 等游戏引擎相似（更像是直接抄袭了 unity 的 UI 系统）（劣质抄袭），在 Scene 下存放 GameObject，并利用父子关系构建 GameObject 树。所有可以被渲染在屏幕上的组件均继承自 RenderableObject（集成了接受事件 InteractableInterface 和可进行动画 IAnime 的接口），并通过父子关系计算位置，缩放和数据存储。不同之处在于，考虑到本项目渲染部分和战斗数据部分是分离开发的，因此单独将渲染部分隔离进 sceneRender（继承自 Canvas）中，在 Scene 中处理数据部分和两者之间的链接。此外，仅有 Canvas 组件能够做父物体，以简化代码中的变量字段，并使架构更加简单清晰。（绝对不是懒哦~）

构建的基础模板元素和预制件如下（开发中，不代表最终文件夹结构）



SceneRender 与 Scenes 基本一一对应，除 BattleScene 拆分了 mainScene 和 combatScene。

### 2.6.2 SceneSystem

Scene 主要功能是作为一个可以独立进行游戏全部逻辑的单元，内置了数据帧和渲染帧的刷新，以及进入和退出场景的多种方法，主要进行数据的处理以及控制渲染的进行。（至于为什么自实现 **frame**，绝对不是因为忘了 **clock** 这个东西哦 这是真的。。。吗？）

（为什么说的这么少呢 因为确实没啥好 **briefly introduce** 的 全是不同场景



需要不同的逻辑)

### 2.6.3 AnimeSystem

所有 Anime 本质上是一个可以按动画帧（与渲染帧刷新频率相同）对特定 Object 进行操作的类。AnimeManager 管理全部 Anime，并控制动画的 tick 和 destroy。鉴于此，可以很方便的做 timer，callback 之类的组件。

Anime 内置了对于 pos, scale, active, alpha, sprite 的动画，只要控制关键帧，就可以自动计算运行全部帧。

一个简单的按钮点击动画

```
class ButtonClickAnime(Anime):
    def __init__(self, obj):
        super().__init__(obj, AnimeInfo(4).disableSprite().add(AnimeType.Scale, 0, Vector2(0.8, 0.8))
            .add(AnimeType.Scale, 4, Vector2.One()).finish())
```

一个简单(?)的 AOE 动画

```
class AoeAnime_FrenziedBloodBlade_Attacker(AoeAttackerAnime):
    def __init__(self, obj, combatRenderer, tempRenderer, info):
        self.tempRenderer = tempRenderer
        self.combatRenderer = combatRenderer
        self.rcdPos = obj.localPosition()
        _ani = AnimeInfo(len(info["receiver"])*8+3).add(AnimeType.Pos, 0, obj.localPosition())
        _kp = -3
        _order = list(range(len(info["receiver"])))
        random.shuffle(_order)
        for index in _order:
            _kp += 8
            _ani.add(AnimeType.Pos, _kp, self.combatRenderer.findChild(info["receiver"][index]).localPosition())
            .add(AnimeType.Pos, _kp+3, self.combatRenderer.findChild(info["receiver"][index]).localPosition())
            _ani.add(AnimeType.Pos, _kp+6, obj.localPosition())
        Anime.__init__(self, obj, _ani.disableSprite().finish())
    def OnKFrame(self, typ, frame):
        if typ == AnimeType.Pos:
            if self.frame % 8 == 5 or self.frame % 5 == 0:
                scale, rotation, mid = getLinearResult(self.rcdPos, self.Object.localPosition(), 416)
                self.tempRenderer.Add(BloodTrace("bt", self.tempRenderer.screen, 10, mid(), rotation), 5,
                    AnimeInfo(20).disableSprite().add(AnimeType.Scale, 0, Vector2(scale().x, 1.5))
                        .add(AnimeType.Scale, 20, Vector2(scale().x, 0)).finish())
                for i in range(3):
                    self.tempRenderer.Add(Blood("bld", self.tempRenderer.screen, 100,
                        self.Object.localPosition()+Vector2(random.randint(-70, 70), random.randint(-70, 70))), 5,
                        AnimeInfo(50).disableSprite().enableAlpha().add(AnimeType.Alpha, 0, 255).add(AnimeType.Alpha, 50, 0)
                            .add(AnimeType.Scale, 0, Vector2.One()).add(AnimeType.Scale, 50, Vector2(2, 2)))
                AudioManager.AddEffect("AttackSlash")
            self.rcdPos = self.Object.localPosition()
```

### 2.6.4 EventSystem

InteractableInterface 反正都被 RenderableObject 继承了

没啥好说的，集成了 pygame 的事件获取和处理，提供

了

PointEnter, OnHover, PointExit, OnClick, OnRightClick, OnKeyClick 等基础方法，用 preventRay 控制是否监听事件及向下传递事件

哦 还有万恶的(ui 真丑啊) tkinter 的输入框。

(剩下的我负责的部分在底部哦~)

### 2.6.5 LLMSystem

该部分负责 LLM 的对话与策略生成。

- 1、ChatWithGPT.py 创建了 Chater 类，负责与 LLM 进行对话。
- 2、KeywordMatch.py 创建了 MatchKeyword 类，用于提取玩家输入的话的关键词。该模块中使用了 spacy 库，以中文模型“zh\_core\_web\_md”为基础计算出句中每个词的向量，并与给定的关键词库中的每个词作比较，最终得出最匹配这段话的若干个关键词。
- 3、InteractionWithGPT.py 创建了 LLMSystem 类，负责让 LLM 根据关键词生成卡牌。该模块首先会告知 LLM 卡牌生成规则，然后将会将一些关键词转化为卡牌的大致数据，并告知 LLM。在 LLM 生成卡牌后，程序会将 LLM 生成的自然语言卡牌转化为 dict，并检查生成的卡牌有无问题，如有问题则会让 LLM 重新生成，直至满足要求。最后，该模块会将生成的若干张卡存储到名为 DYNAMICGENPAGE.json 的文档中。
- 4、LLMSystem 的文件夹中还有一个文件夹，名为 ToGPT，里面是给 LLM 传达的 system 信息，负责告诉 LLM 卡牌生成规则与所需完成的任务。

### 2.6.6 ReceptionSystem

该部分为战斗系统的后端，负责处理战斗系统中的各种数据，并将特定格式的 dict 传递给前端以供显示。

- 1、Buffs.py、DiceBuff.py、PageBuff.py 负责存储与实现各个状态效果、骰子与书页的特殊效果。
- 2、Dice.py、Page.py 分别创建了 Dice 与 Page 类，用于实例化以存储信息。文件中的函数还负责处理骰子与书页的拼点与单方面攻击。
- 3、SpeedDice.py 创建了 SpeedDice 类，负责处理速度骰的操作。
- 4、Emotion.py 创建了 Emotion 类，负责处理人物情感硬币与情感等级的有关操作。
- 5、Character.py 创建了 Character 类，负责存储与修改人物信息，进行与人物有关的操作。
- 6、Team.py 创建了 Team 类，负责整个阵营的初始化，回合末结算等操作。
- 7、CombatSystem.py 创建了 BattleSystem 类，做为战斗系统的总控，负责整个战斗的初始化，敌方策略生成等操作，以及为前端提供各个接口。
- 8、以 Reception 开头的 5 个 py 文件创建了若干 Character 的子类，存储各种接待的特殊机制与常规数据。
- 6、PageCoding.py 提供了 Page 实例与 json 文件互相转化的两个函数，用于将由 llm 生成的卡转为 Page 实例接入系统。
- 10、GenerateDict.py 用于将战斗系统的信息汇总整和到一个约定格式的 dict 中，以供前端显示。

### 2.6.7 AudioSystem

简单封装了一下 python 的 sound 和 music 的部分

支持了延迟 sound，简易音效添加和 bgm 的自动随机切换之类的有用的功能。

(Assets\Audio\AudioLis.json 有配表)

### 2.6.8 RogueSystem

啊 是万恶的发牌员 叉腰！

古希腊掌管 rouge 中随机生成的相关内容的神（所以不是坎诺特先生发放的慈善福利哦），同时存储了全局的 stage 信息。

### 2.6.G Assets

Audio 文件夹 音频及配表

Font 文件夹 各种好看的字体

GData 文件夹 存了游戏的 stage 信息（就是肉鸽的层），预制的部分卡（王之宝库的部分财产），角色升级的配置文件和被动能力的文件 都是拿 json 配的  
(长膜通信是古老且优雅的.jpg)

Image 文件夹 如标题，内部管理一坨（主要原因在于顶层放了一堆 asset 后期懒得改了）

### 2.6.10 Data

此 Data 非彼 Data 这里主要放的是一些常量和有用的类型，函数

比如自实现的 Vector2，算贝塞尔曲线的，color 的常量，单例的装饰器之类的

(至于为什么叫 Data，那你该问创项目时的我 ( ))

## 3 创意

### 3.1 前端

(怎么感觉全是创意呢？) (并非创意，只是架构的搬运工)

(但你就说是不是自己设计的流程罢)

(本项目是没有用模板的哦)

介绍整体代码的时候应该已经大致能呈现了关于渲染端整体架构的创意了，这里写点啥呢？

先介绍介绍 **Scenes** 么？开始界面（记得多点点），地图界面（我去，我喜欢走路），商店界面（升级角色和生成卡牌的），选卡界面（配置你的卡组），战斗界面（玩过《废墟图书馆》的肯定就看懂了，没玩过的记得看 3.2 开始的后端中对战斗机制的描述哦）

再介绍介绍开发思路罢。首先本项目的核心代码量均在 **BattleScene** 中，考虑到人员分工，因此将显示端与数据端分离开发，两者之间的数据传递通过特定结构的包进行。

然后是关于肉鸽系统，灵感来源于原版角色天赋虽然可以任意拼接，但是不能叠加同名天赋，因此决定做一个可以叠加天赋的版本；由此衍生出一个角色逐渐随层数加深逐渐成长的设计，成长维度也是多方面的（包括血条，混乱条，骰子数，伤害抗性，速度等等）；虽然由于开发时间的限制并没有做成一个平衡性非常好，耐玩性非常强的肉鸽，但是已经是一个有难度区分，并且有多种攻克思路的肉鸽（真的。。吗？）（另：三种难度真的区分程度良好，并且是通过一些优雅的数值调整进行的（自夸 ing））

最后是代码设计中的一些小感想罢，感觉代码整体架构在我所有搭过的项目中应该是最好了（果然人是会进步的呢），至少这次没有出现需求一个功能需要对底层结构进行根本性调整的时候（包括为了拿分临时赶工出来的 **MapScene**）。啊，感觉这几周代码敲的有点多了，想要一一道来感觉不太可能了，现在也没办法一点点说清楚敲代码过程中有一些比较漂亮（至少我觉得）的实现了（比如不同种动画的实现小细节（**aoe**，跳币，加载场景，战斗动画生成（这个只是复杂，架构不是很好，但是它有 **500** 多行欸）），自排序的画布，**Scene** 切换的管理和动画，以及对于各种奇形怪状的 **SD** 的人物部分的位置进行 **focus**，很方便的临时 **Object** 绘制，以及各种地方的动画音效交互设计的小细节之类的）。所以！我有很好的创意，但这里空间太小，我写不下。

以及关于之前提到的监听的装饰器之类的设计思路，其实项目创立时有考虑过用 **EventBus** 之类的思路去进行类之间的信息传递，但最终还是选择了类似 **UI** 的这种思路，由于 **UI** 元素间都有明确的继承关系，所以 **listener** 的重要性也就下降了，最终也没有采用这种思路。

嗯，然后就是附加的 **20** 分，首先是 **consistency**：本作首先已经还原了图书馆原著的绝大部分功能，因此战斗系统已经较为成熟了（未还原的部分主要是各种异想体书页，那只能说有各种奇奇怪怪的机制和动画，实现难度会高很多很多）；

其次本作的肉鸽系统也具有完整的游戏性，表现在不好好选被动选升级喜欢乱按 **P** 键就会被对面制裁，认真去构建一个卡组体系，对强度有清晰的认知就可以有效降低难度（另：简单模式（绿）和普通模式（白）的难度已经降的很低了，想挑战自己右转困难模式（红））；包括敌人的选择，不同的敌人具有



显著的机制差异，不是堆强度（事实上不好好玩也堆不起来）就能无脑过的；还有关于 LLM 生成卡，只能说，会强调和组合关键词的真的能拿到比预先安排的卡强的多的卡。

关于 **depth**，其实没什么好说的，熟练条件下通关普通难度大概 **2h**，而在此之前，你还要学习本作 **2** 种伤害类型（体力，混乱），**3** 种书页（近战，远程，**aoe**），**4** 个主要机制（卡牌，拼点，情感，肉鸽），**5** 种骰子（斩击，突刺，打击，防御，闪避），**5** 种迥异的敌人（**5** 层接待，挑的全是有强机制的），**16** 个主要 **buff** 及对应的体系，配了 **400** 多行表的角色升级（虽然应该归功于 **json** 的格式化，但它真的很多），以及 **80** 多种被动能力。至于困难模式的诞生，只要小小的调整一点点数值，嗯，祝你好运。

包括本作的美术设计，在还原了原作绝大部分 **UI** 的基础上，添加了一些其他游戏的很好看的 **UI** 元素，整个界面交互起来还是比较舒服的（个人体感），很多能想到的小细节也都加上去了（大部分 **UI** 和图像都是拆原作美术包的，因此艺术性必有保证的）；以及音乐，**debug** 的时候打太快了觉得音乐太多，真正打实机演示的时候发现思考时间一上去，一场战斗 **5** 首 **BGM** 都不够捏。。。

以及战斗机制要好好看哦，不然打不过不要怪策划~

## 3.2 后端

### 3.2.1 战斗机制

战斗中，玩家会操控至多五名角色，每名角色拥有两个条，血条与混乱抗性条（如图 1）。本游戏中有斩击，突刺，打击三种伤害，每名角色针对这三种伤害有不同的抗性（如图 2 右侧），分为免疫，抵抗，耐性，一般，脆弱，致命 6 个等级。实战时，玩家应根据敌方抗性选择合适的书页。

血条清空会导致角色死亡，混乱抗性条清空会导致角色在当幕与下一幕陷入混乱。陷入混乱的角色无法行动，且所有抗性变为致命。解除混乱时，恢复所有混乱抗性。

光芒是本游戏中的费用，大部分书页需要光芒打出，所需数量显示在书页左上角（如图 2），角色目前拥有的光芒数显示在角色上方（如图 1）。

速度骰（角色头顶的骰子）决定了该角色最多能使用几张书页。双方速度骰相互指定对方为目标的情况称为拼点，一方指另一方称为单方面攻击。使用己方速度骰指向敌方时，若己方速度大于敌方，则会强制改变其目标，使其与自己拼点。速度高的骰子在结算时优先级更高。

双方书页进行拼点时若书页中都有骰子，则骰子会进行拼点。书页中红色的骰子为攻击型骰子，攻击型骰子相互拼点时，大的一方打出基础值等同与点

数的伤害与混乱伤害。

蓝色的骰子为防御型骰子，分为格挡骰与闪避骰。格挡骰为低风险低回报的防御骰，与攻击骰拼点可抵消等量的伤害，若拼赢，还会打出等同于差值的反震伤害，反震伤害只有混乱伤害且无视抗性。与防御骰拼点拼赢时打出全额的反震伤害。

闪避骰则是高风险高回报，与攻击骰拼点拼输会承受全额伤害，但拼赢可恢复等同于骰子数值的混乱抗性，且可以重复投掷。与格挡骰拼赢时恢复混乱抗性，与闪避骰无视点数强制拼平。

当一方的书页中没有骰子了或是书页进行单方面攻击，骰子会进行单方面攻击。攻击型骰子直接打出全额的伤害与混乱伤害，而未使用的防御骰会进入



图 2



图 1

角色的防御书页。当角色防御书页有骰子时，遭受书页单方面攻击会使用防御书页与其拼点。

书页右侧会显示书页的详细信息（如图 2），所有骰子的类型与最大最小值，特殊效果以及书页本身的特殊效果。

### 3.2.2 拓展战斗机制

书页的默认类型为近战，也有其他的类型的书页。

远程书页结算优先级高于近战。当远程书页中的攻击骰拼输时不会立刻受到伤害，而是会将对方的攻击骰移至对方书页末尾重新投掷。

群体攻击书页非常的强力，也通常具有较高的费用，分为交锋与清算。使用群体书页指向主目标后还会随机选择所有其他敌方角色的一颗随机的速度骰作为攻击目标。交锋书页有多颗骰子，会与对方书页的对应位置的骰子进行拼点，胜利则摧毁该骰子并造成伤害。清算书页则只有一颗较大的骰子，会与敌方书页中所有骰子的总和进行比较，胜利则摧毁敌方所有骰子并造成伤害。

反击骰是一种特殊的骰子，会在战斗开始时直接插入防御书页。拼点胜利

后，反击骰还能重复投掷。

背景设定上，光芒是情感具象化的产物。当骰子投出最大值最小值，拼点胜利或失败以及杀死敌方角色时，人物会获得正面或负面情感硬币。回合结束时情感硬币总数达到一定数量会提升情感等级，最高五级。提升情感硬币的当幕会回满光芒并额外抽取一张书页。未提升则只会恢复一点光芒并抽取一张书页。

我们还添加了 16 种 **buff** 或 **debuff**，它们在游戏中会显示它们的效果，故不在这里介绍。骰子的特殊效果共有 16 种模板，书页特殊效果则有 20 种模板。此外，人物还有 76 种被动能力与 30 项升级可在 NPC 处抽取。这边分内容过多，且游戏内描述较清晰，也不在这里介绍。

### 3.2.3 接待

游戏内提供了叶工坊，殷红迷雾，RRR 公司，六协二科，扭曲阳五场接待，也可以创建不同的自定义接待。

五场接待中既有群架也有单体 **boss**，还有很多特殊机制。如殷红迷雾半血后会开启 **ego** 状态，获得威力增幅，并会使用大点数群体攻击书页。扭曲阳会在一阶段使用大点数的闪避反击骰，以阻止玩家攻击他。玩家必须先击败他的两只手，随后阳会合体进入二阶段。血量低时，阳会解体，并使用修复书页，如果你不与他的修复书页拼点，他将恢复手的血量。阳本体并不会使用其他攻击书页，只会在一定幕数后使用大点数群体书页，合体会加快其倒计时。

### 3.2.4 代码

我们的项目战斗数据处理的部分庞大到使其足以被称为后端，体量上即是创意。后端的代码设计并没有太多亮点，但也有些创意。

角色的各项增益，**buff** 的结算采用了时点的设计思路，在如回合结束，拼点胜利等地方添加时点，统一执行 **buff** 的结算。

不同种类的 **buff** 运用了 **python** 的子类。充分运用了子类的优势，整合相同的部分，分写不同的部分，节省码量，提高可读性。

## 3.3 LLM 部分

### 3.3.1 关键词提取

由于给出的 LLM 提取关键词的能力极不稳定，因此模块用 **Python** 的库中的语言模型来实现关键词提取。

**KeywordMatch.py** 模块中，使用了 **Python** 的 **spacy** 库，并使用了 **spacy** 库中的中文模型“**zh\_core\_web\_md**”来计算词向量，同时用了 **sklearn** 库来计算词向量

间的余弦相似度，以此来匹配与这句话最相关的若干个关键词。

只用上述方法是无法让模块理解否定词的。例如：玩家输入“不强大”，模块会匹配“强大”而不是“弱小”。为解决这一问题，模块中还加入了否定词识别，如果检测到否定词，那就将这个否定词对应的关键词转化为反义词，以此来提升关键词提取的精准度。

### 3.3.2 实时生成卡牌

`InteractionWithGPT.py` 模块实现了战斗中实时生成卡牌。该模块会接收 `KeywordMatch.py` 模块传入的关键词，然后根据关键词调整卡牌相关参数，然后将这些参数的具体要求告诉 LLM，并让 LLM 以自然语言生成卡牌。

这里生成的卡牌不是预先制作好的卡牌，而是在告知 LLM 游戏规则与卡牌格式要求后，让 LLM 自由发挥生成的卡牌。在每场游戏中，玩家都会收到来自 LLM 生成的独一无二的卡牌。

收到 LLM 生成的自然语言卡牌信息后，模块会将自然语言转化成能够被其他模块读取的 `dict` 形式，并且会检查卡牌的格式，数值。如果卡牌的格式有误或者数值不合理，模块会指出 LLM 的具体错误，并且要求 LLM 重新生成一张卡牌。如此循环，直至生成一张正确的卡牌。

为了使得 LLM 生成的卡牌强度合理，模块会量化计算一张卡牌的强度与不稳定度，并且会根据这两个数值判定卡牌的设计是否合理。

LLM 生成卡牌的速度较慢，因此，在卡牌的生成过程中，模块会实时向前端汇报卡牌的生成进度，并在前端以进度条的形式显示。

## 3.4 无终

*郊区比我所听说的要更加荒凉。*

最后的最后还是想写点什么，虽然大抵没什么人看，但毕竟算是第一个真正正做完的项目，还是有点纪念意义的。

截至提交，本项目应该还没有完全实现预想中的全部功能，但毕竟凡事总有遗憾么。或者说，正是这种瑕疵，才可能会孕育出更多更好的作品。

当然，在这里自然还要感谢我的队友，没有他们的协助，这个项目的完成度绝不可能如此之高；以及要感谢 SI100B 的各位助教，他们很认真的解答了我们有关 LLM 的诸多困惑，并及时的提醒我们要踩住的得分点。

最后的最后的最后，在《Library of Redundancy》中玩的开心！

日常是命运赐予生灵最大的善意，平凡，即是喜乐。