系统设计师

1. 系统设计文档
   1. 游戏是什么风格的：是一款横版2D的经典RPG游戏，核心在于对话和剧情。主角在游戏中可以被操控着进行基本的跳跃，攻击，使用技能（基本是释放魔法或者释放武器），敌人也可以进行释放武器，和技能来对主角进行攻击。有基本的碰撞机制和生命机制。每次进入一个新的场景，会与对方角色进行一次对话，对话后进行战斗，战斗中间和结束时也可以进行对话。在最终打败boss后，则弹出提示框，游戏结束。（该任务将扫描一个基础游戏的代码，并将其风格总结成文档）
   2. 模块设计：
      1. 图片层：背景图片（包含不同场景），角色图片（半身立绘+角色的sprite sheet，分为敌人和主要角色），魔法图片（类似火球，水球，大剑）
      2. 代码层：
         1. Index.html（网页段的模板，通过使用js来实现游戏）
         2. Logic.js（核心逻辑控制，各个模块的交互）
         3. ui.js （负责场景中的ui逻辑控制）
         4. Player.js（负责主要角色的逻辑控制）
         5. background.js（负责使用背景图片，在对应的场景中切换）
         6. enemy.js（负责使用敌人逻辑）
         7. dialogData.js（存储角色对话）
         8. dialogCtrl.js（负责使用角色的对话，来更新对话框）
      3. 声音层：存储游戏的各种背景音乐，音效
2. 设计过程
   1. 故事讲述，根据用户的提示，生成一个完整而详细的故事链。要求详细地讲述角色到达了什么地方，遇到了谁，说了什么话，进行了什么战斗，得到了什么。
   2. 系统文档设计：根据1的内容，设计各个模块的架构和依赖。确认每个模块负责的内容，再为每一个模块写一个清单要求实现的内容（包括定义的接口，变量名，以及使用的其他模块的变量和方法等），写一个整体的架构依赖文档讲清楚各个模块之间如何交互，输出n个文件。
   3. Index.html模块：读取html文档和架构文档，实现文档的要求和依赖，完成index.html的撰写。
   4. Logic模块：读取logic文档和架构文档。实现文档中的要求和依赖，完成logic.js的撰写
   5. UI模块：读取ui文档和架构文档，实现文档中的要求和依赖，完成ui.js的撰写。
   6. 关卡设计：根据故事，设计不同的关卡，关卡的元素有：角色，对话，战斗，背景。每个关卡涵盖：关卡内容介绍，关卡角色的介绍，以及各自所有的武器和魔法，大致的角色对话内容描述。
   7. 图片生成：根据关卡设计内容，通过ai生成去除背景后的角色半身立绘以及武器，魔法图片（使用工具将背景透明化）。同时给出一个清单list（区分主角和配角）
   8. Player模块：通过对图片list使用找到主角的素材。读取player模块的文档以及整体架构文档，使用主角的素材，完成主角的逻辑代码生成，并完成依赖。（包括音效的播放，假设声音素材都放在sounds文件夹）
   9. Enemy模块：通过对图片list找到敌人的素材。读取player模块的文档以及整体的架构文档，使用敌人的素材，完成敌人的逻辑代码生成，并完成相关的依赖。
   10. 背景生成：同样根据关卡描述，为每个关卡设计一个场景背景的图片（1920x1080）。给出一个名字清单。
   11. Background.js：根据背景的名字清单，以及background的
   12. 对话脚本设计：根据故事内容和关卡设计，对每个关卡写出详细的对话脚本，按照json的格式。（以dialog{scene{role:sentence;role2:sentence;…} {scene2…}…}的格式）
   13. dialogCtrl: 根据对话脚本json文件和关卡设计，完成对话的逻辑，包括角色图片立绘的切换（通过调用player和enemy中的角色图片变量）。完成dialogCtrl.js

**1. StoryTellerAgent**

* 【主要职责】根据提示生成完整且详细的游戏故事链
* 【具体任务细项】：
  + 展开主线剧情，从起点到终点（含Boss战）
  + 详细描述每个场景到达的地点
  + 指定遇到的人物（角色姓名、身份、初步性格）
  + 写清对话内容（谁对谁说什么）
  + 具体描述战斗经过（触发、过程、结果）
  + 描述每个战斗胜利后的奖励或结果
* 【输出要求】：
  + 标准 Markdown 格式文档
  + 按【地点-遇到的人物-对话-战斗-奖励】顺序组织
* 【依赖输入】：
  + 用户提供的初步提示信息

**2. SystemDesignerAgent**

* 【主要职责】根据故事链，设计系统架构、模块划分、接口规范
* 【具体任务细项】：
  + 识别游戏中需要的功能模块（如 Player、Enemy、Dialog、HUD）
  + 每个模块详细列清：
    - 模块职责
    - 主要类和方法
    - 对外暴露接口（方法名、参数、返回值）
    - 调用其他模块的依赖（变量、方法）
  + 输出系统依赖关系图（清晰箭头标注模块之间调用）
* 【输出要求】：
  + 生成：
    - modules\_structure.md（模块定义文档）
    - modules\_interface.md（模块接口清单）
    - system\_dependency.md（系统依赖关系图）
* 【依赖输入】：
  + StoryTellerAgent输出的完整故事链

**3. HtmlDeveloperAgent**

* 【主要职责】基于系统文档，编写index.html
* 【具体任务细项】：
  + 设置基本HTML结构（canvas画布、UI层容器）
  + 引入各模块对应的js文件（按依赖顺序）
  + 引入CSS或样式控制区（如有）
  + 设置基本启动函数
* 【输出要求】：
  + index.html 文件
* 【依赖输入】：
  + modules\_structure.md
  + system\_dependency.md

**4. LogicDeveloperAgent**

* 【主要职责】根据系统设计，开发逻辑核心模块（logic.js）
* 【具体任务细项】：
  + 管理场景切换
  + 管理战斗开始、胜利、失败逻辑
  + 管理技能释放、血量变化
  + 触发对话的逻辑（调用dialogCtrl）
* 【输出要求】：
  + logic.js 文件（有清晰注释）
* 【依赖输入】：
  + modules\_structure.md
  + system\_dependency.md

**5. UiDeveloperAgent**

* 【主要职责】根据UI设计文档开发Hud.js或ui.js
* 【具体任务细项】：
  + 生命条、魔法条绘制与更新
  + 技能图标冷却逻辑
  + 提示框、对话框绘制
* 【输出要求】：
  + ui.js 或 hud.js 文件
* 【依赖输入】：
  + modules\_structure.md
  + system\_dependency.md

**6. LevelDesignerAgent**

* 【主要职责】根据故事链设计具体关卡
* 【具体任务细项】：
  + 定义每个关卡：
    - 背景场景
    - 出现角色（主角、敌人）
    - 预设对话内容（概要）
    - 战斗配置（敌人数量、武器种类）
  + 规划关卡顺序与难度曲线
* 【输出要求】：
  + levels\_design.md 文件（列出所有关卡内容）
* 【依赖输入】：
  + StoryTellerAgent输出（故事链）

**7. AssetGeneratorAgent**

* 【主要职责】根据关卡角色，生成图像资源
* 【具体任务细项】：
  + 为主角生成半身立绘
  + 为敌人生成半身立绘
  + 为角色生成对应武器图标、魔法图标
  + 自动去背景处理
  + 输出清单list（标注主角/敌人区分）
* 【输出要求】：
  + characters\_images/ 文件夹
  + weapons\_magic\_images/ 文件夹
  + assets\_list.txt（包含文件路径+角色分类）
* 【依赖输入】：
  + LevelDesignerAgent输出（关卡设计）

**8. PlayerDeveloperAgent**

* 【主要职责】生成Player角色逻辑模块（player.js）
* 【具体任务细项】：
  + 读取主角图片资源
  + 实现角色基本动作（移动、跳跃、攻击、释放技能）
  + 加入音效（从sounds目录）
  + 支持生命值、魔法值管理
* 【输出要求】：
  + player.js 文件
* 【依赖输入】：
  + SystemDesignerAgent输出（player模块规范）
  + AssetGeneratorAgent输出（图片资源list）

**9. EnemyDeveloperAgent**

* 【主要职责】生成敌人角色逻辑模块（enemy.js）
* 【具体任务细项】：
  + 读取敌人图片资源
  + 实现敌人攻击行为、技能释放逻辑
  + 管理敌人血量与击败逻辑
* 【输出要求】：
  + enemy.js 文件
* 【依赖输入】：
  + SystemDesignerAgent输出（enemy模块规范）
  + AssetGeneratorAgent输出（图片资源list）

**10. BackgroundImageAgent**

* 【主要职责】生成每个关卡场景背景图片
* 【具体任务细项】：
  + 读取每个关卡描述
  + 生成1920x1080的背景图（高质量）
  + 输出背景图片清单（背景名称）
* 【输出要求】：
  + backgrounds/ 文件夹
  + backgrounds\_list.txt
* 【依赖输入】：
  + LevelDesignerAgent输出（关卡设计）

**11. BackgroundManagerAgent**

* 【主要职责】开发背景管理模块（background.js）
* 【具体任务细项】：
  + 加载背景图片
  + 在场景切换时正确切换背景
* 【输出要求】：
  + background.js 文件
* 【依赖输入】：
  + BackgroundImageAgent输出（背景图片清单）
  + SystemDesignerAgent输出（background模块规范）

**12. DialogueWriterAgent**

* 【主要职责】撰写详细关卡对话脚本（JSON格式）
* 【具体任务细项】：
  + 每个场景写详细对话（含主角、敌人）
  + 保持自然对话流（可含分支选项）
  + 符合格式：

json

复制编辑

dialog: {

scene1: { role1: "text", role2: "text" },

scene2: {...}

}

* 【输出要求】：
  + dialogue\_script.json 文件
* 【依赖输入】：
  + StoryTellerAgent输出（故事链）
  + LevelDesignerAgent输出（关卡设计）

**13. DialogueControllerAgent**

* 【主要职责】根据对话脚本，开发dialogCtrl.js
* 【具体任务细项】：
  + 实现对话播放流程
  + 控制角色立绘切换（从player/enemy模块调用）
  + 支持分支跳转
* 【输出要求】：
  + dialogCtrl.js 文件
* 【依赖输入】：
  + DialogueWriterAgent输出（dialogue\_script.json）
  + SystemDesignerAgent输出（dialogCtrl模块规范）