

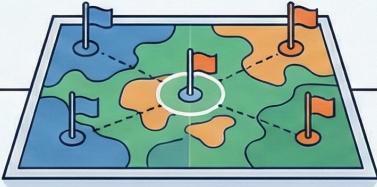
FlagWars 项目结构文档

在线夺旗游戏：FlagWars 技术架构解析

项目概览



FlagWars：一款在线多人
夺旗战略游戏
玩家通过部署兵力占领领土，
目标是击败所有对手。



核心架构



服务器核心：server.py 模块
通过 WebSocket 管理游戏逻辑、
玩家、房间、地图和战斗。

核心游戏特色



支持实时对战



战争迷雾



房间系统



排行榜及观战模式



数据持久化：database.py 模块

使用 SQLite 数据库管理用户账户、游戏记录
和会话。

表名称	主要功能
users	存储用户信息和凭证
games	保存历史游戏记录
user_sessions	管理用户登录会话

技术栈亮点



后端
Tornado



前端
Brython



通信
WebSocket



数据库
SQLite

典型用户流程



注册/登录



加入房间



开始游戏



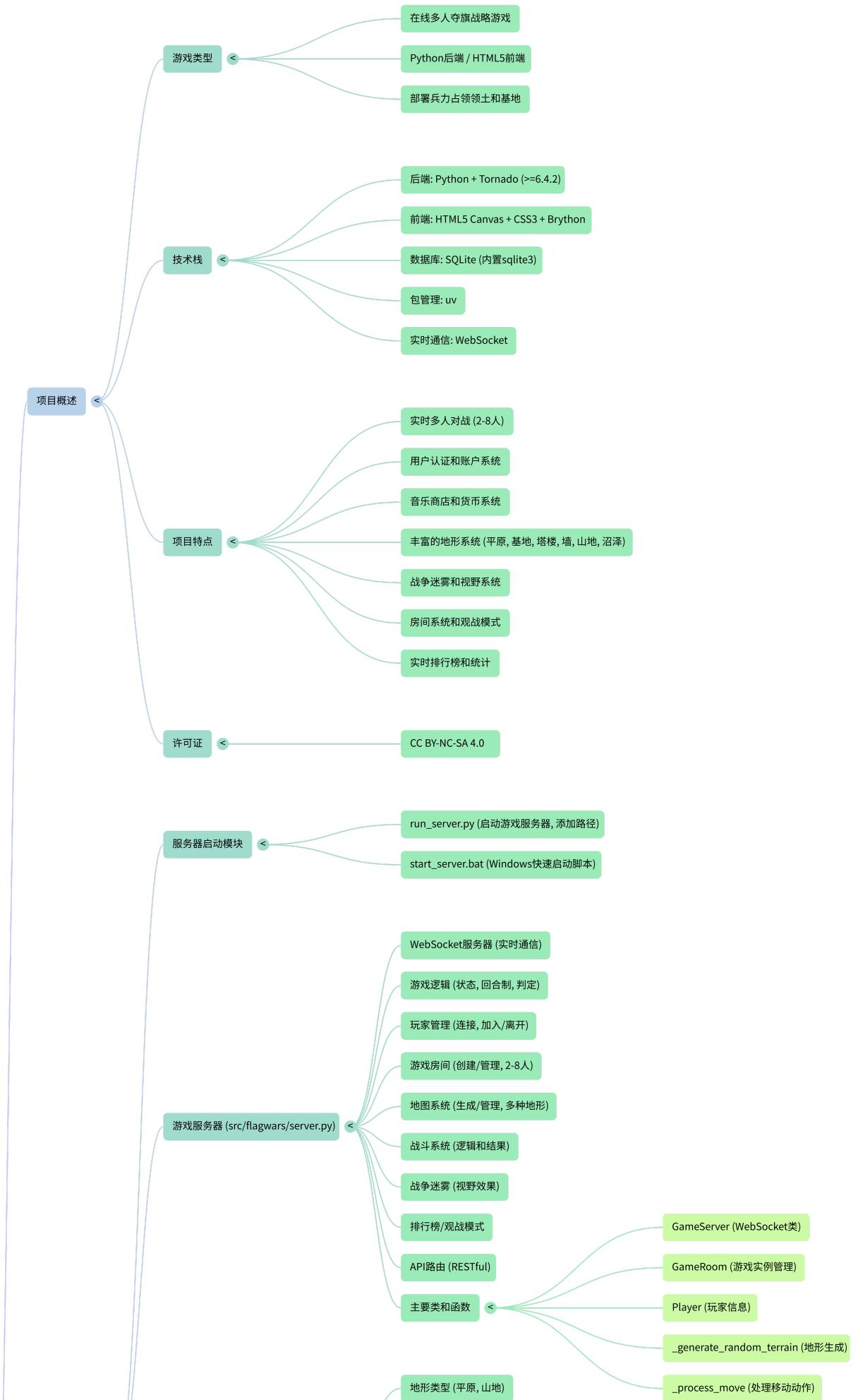
游戏结束

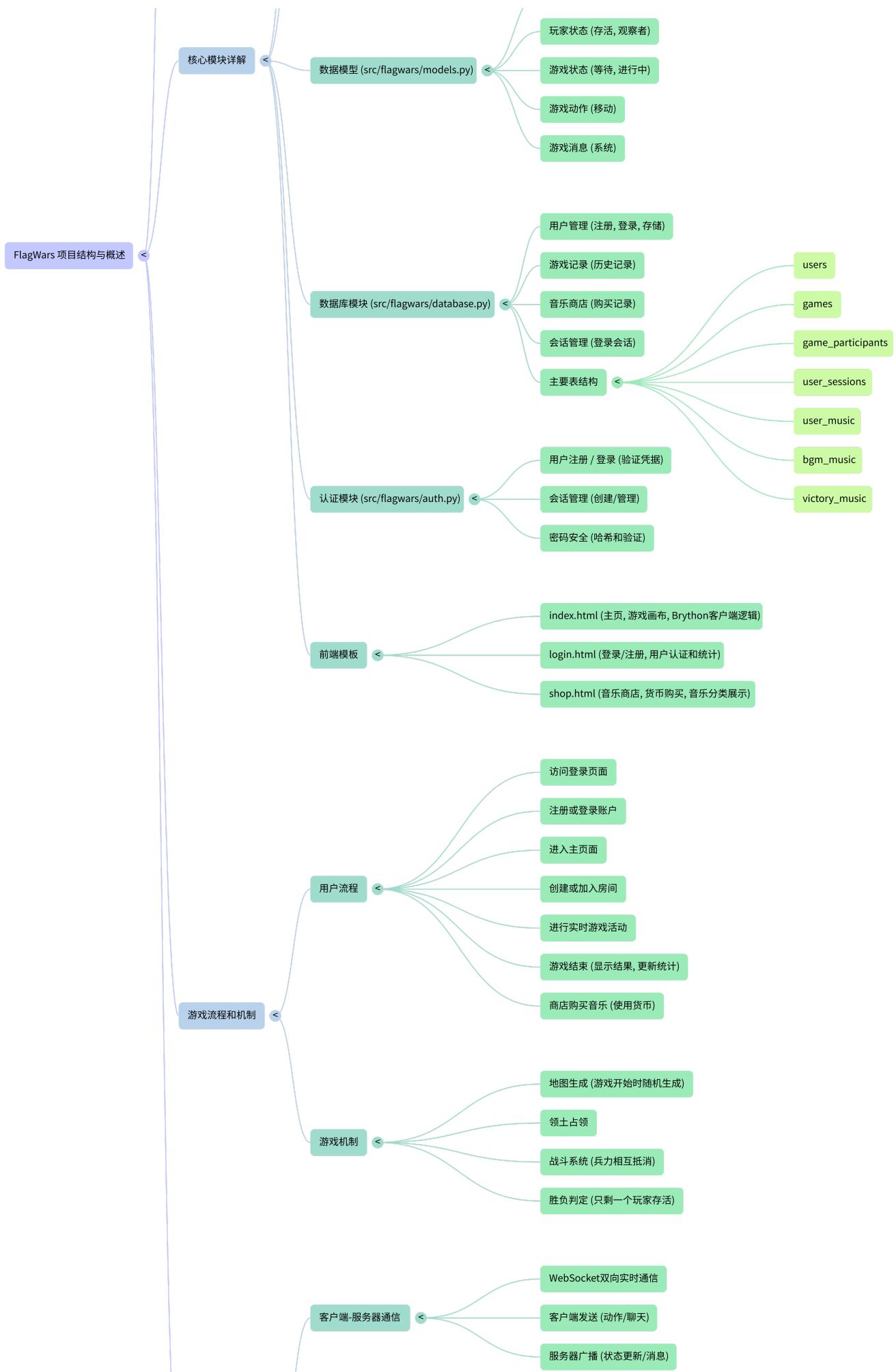


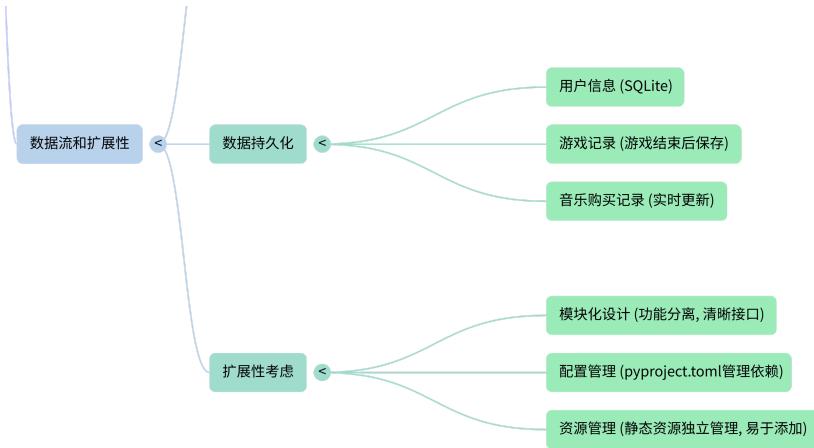
商店购物

© NotebookLM

一图流：







1. 项目概述

FlagWars 是一个现代化的基于 Python 后端和 HTML5 前端的在线多人夺旗战略游戏。玩家通过在地图上部署兵力来占领领土和基地，最终目标是击败所有对手成为最后的胜利者。

1.1 技术栈

- 后端: Python + Tornado Web 框架 ($>= 6.4.2$)
- 前端: HTML5 Canvas + CSS3 + Brython (Python in Browser)
- 数据库: SQLite (通过 Python 内置 sqlite3 模块)
- 包管理: uv (现代 Python 包管理器)
- 实时通信: WebSocket (通过 Tornado 实现)
- 许可证: CC BY-NC-SA 4.0

1.2 项目特点

- 实时多人对战 (2-8人房间)
- 完整的用户认证和账户系统
- 音乐商店和货币系统
- 丰富的地形系统 (平原、基地、塔楼、墙、山地、沼泽)
- 战争迷雾和视野系统
- 房间系统和观战模式
- 实时排行榜和游戏统计

2. 目录结构

```

1 | FlagWars/
2 |   └── docs/                               # 项目文档目录
3 |     └── project-structure.md             # 本文档
4 |   └── icons/                            # 图标资源目录
5 |     └── flag.png                         # 旗帜图标

```

```

6   |   ├── icc.png          # 游戏图标
7   |   ├── image.ico        # 网站图标
8   |   └── slog.png         # 游戏标志
9   ├── music/              # 音乐资源目录
10  |   ├── home/            # 背景音乐
11  |   |   ├── Electric-Heartbeat.mp3
12  |   |   ├── Moonlight-and-Marmalade.mp3
13  |   |   └── Whispers-of-Strategy.mp3
14  |   └── vict/             # 胜利音乐
15  |       ├── folk-vict.mp3
16  |       ├── mario-vict.mp3
17  |       ├── royal-vict.mp3
18  |       └── weird-horn-vict.mp3
19  ├── src/                # 源代码目录
20  |   └── flagwars/         # 主应用包
21  |       ├── __init__.py
22  |       ├── auth.py        # 用户认证模块
23  |       ├── database.py    # 数据库操作模块
24  |       ├── models.py      # 数据模型定义
25  |       ├── server.py      # 游戏服务器主模块
26  |       └── templates/      # HTML模板目录
27  |           ├── index.html # 游戏主页面
28  |           ├── login.html # 登录/注册页面
29  |           └── shop.html  # 音乐商店页面
30  ├── pyproject.toml        # 项目配置文件
31  ├── README.md            # 英文项目说明文档
32  ├── README_ch.md          # 中文项目说明文档
33  ├── LICENSE               # 许可证文件 (CC BY-NC-SA 4.0)
34  ├── DEPENDENCIES_LICENSE.md # 依赖项许可证说明
35  ├── TODO.md               # 开发任务和完成状态清单
36  ├── run_server.py          # 服务器启动脚本
37  └── start_server.bat      # Windows批处理启动脚本

```

3.核心模块详解

3.1 服务器启动模块

`run_server.py`

- 负责启动游戏服务器
- 将src目录添加到Python路径
- 调用flagwars.server.main()启动服务器

`start_server.bat`

- Windows批处理脚本，用于快速启动服务器
- 执行`run_server.py`脚本

3.2 游戏服务器模块 (`src/flagwars/server.py`)

这是项目的核心模块，包含以下主要功能：

- **WebSocket服务器**: 实现实时双向通信
- **游戏逻辑**: 处理游戏状态、回合制逻辑、胜负判定
- **玩家管理**: 管理连接的玩家、处理玩家加入/离开
- **游戏房间**: 创建和管理游戏房间，支持2-8人游戏
- **地图系统**: 生成和管理游戏地图，支持多种地形
- **战斗系统**: 处理战斗逻辑和结果
- **战争迷雾**: 实现视野系统和战争迷雾效果
- **排行榜**: 实时显示玩家统计和排名
- **观战模式**: 支持以观众身份观看游戏
- **API路由**: 提供RESTful API接口

主要类和函数：

- `GameServer`: WebSocket服务器类，处理客户端连接
- `GameRoom`: 游戏房间类，管理单个游戏实例
- `Player`: 玩家类，存储玩家信息和状态
- `_generate_random_terrain`: 生成地形地图的函数
- `_process_move`: 处理玩家移动动作的函数

3.3 数据模型模块 (`src/flagwars/models.py`)

定义了游戏中使用的所有数据结构：

- **地形类型**: 平原、山地、沼泽等
- **玩家状态**: 存活、死亡、观察者等
- **游戏状态**: 等待中、进行中、已结束等
- **游戏动作**: 移动等
- **游戏消息**: 系统消息等

3.4 数据库模块 (`src/flagwars/database.py`)

负责所有数据库操作：

- **用户管理:** 注册、登录、用户信息存储
- **游戏记录:** 保存游戏历史记录
- **音乐商店:** 管理用户音乐购买记录
- **会话管理:** 管理用户登录会话

主要表结构：

- `users`: 用户信息表
- `games`: 游戏记录表
- `game_participants`: 游戏参与者表
- `user_sessions`: 用户会话表
- `user_music`: 用户音乐拥有情况表
- `bgm_music`: 背景音乐表
- `victory_music`: 胜利音乐表

3.5 认证模块 (`src/flagwars/auth.py`)

处理用户认证相关功能：

- **用户注册:** 创建新用户账户
- **用户登录:** 验证用户凭据
- **会话管理:** 创建和管理用户会话
- **密码安全:** 密码哈希和验证

3.6 前端模板

`index.html`

- 游戏主页面
- 包含游戏画布和控制界面
- 使用Brython在浏览器中运行Python代码
- 实现游戏客户端逻辑

login.html

- 用户登录和注册页面
- 提供用户认证界面
- 显示用户统计信息

shop.html

- 音乐商店页面
- 允许用户使用游戏内货币购买音乐
- 分类展示背景音乐和胜利音乐

4. 游戏流程

4.1 用户流程

1. 用户访问网站，进入登录页面
2. 用户注册或登录账户
3. 登录成功后，进入游戏主页面
4. 用户可以创建或加入游戏房间
5. 游戏开始后，玩家进行实时活动
6. 游戏结束后，显示结果并更新统计
7. 用户可以使用获得的货币在商店购买音乐

4.2 游戏机制

1. 地图生成: 游戏开始时随机生成地形地图
2. 领土占领: 占领的领土
3. 战斗系统: 玩家的兵力相互抵消
4. 胜负判定: 当只剩下一个玩家存活时，游戏结束

5. 数据流

5.1 客户端-服务器通信

- 使用WebSocket进行实时双向通信
- 客户端发送游戏动作和聊天消息
- 服务器广播游戏状态更新和消息

5.2 数据持久化

- 用户信息存储在SQLite数据库中
- 游戏记录在游戏结束后保存
- 音乐购买记录实时更新

6. 扩展性考虑

6.1 模块化设计

- 各功能模块分离，便于独立开发和维护
- 清晰的接口定义，便于功能扩展

6.2 配置管理

- 通过pyproject.toml管理项目依赖和配置
- 可轻松添加新的依赖和配置项

6.3 资源管理

- 静态资源(图标、音乐)独立管理
- 可轻松添加新的游戏资源