战斗系统章节配置功能 - 实现完成报告

🎉 功能实现概述

我已经成功为您的战斗系统添加了**基于章节的求生者和监管者配置功能**！现在您的游戏可以根据剧情章节自动配置不同的角色组合。

✅ 实现的功能

1. 六个预定义章节配置

- **第1章：医者仁心** - 医疗主题 (医生、园丁、作曲家 vs 厂长)
- **第2章：法理之争** - 智力对决 (律师、小说家、记者 vs 小丑)
- **第3章：野性呼唤** - 力量较量 (野人、前锋、昆虫学家 vs 蜘蛛)
- **第4章：舞台惊魂** - 敏捷表演 (舞女、杂技演员、小女孩 vs 红夫人)
- **第5章：机械迷城** - 科技对决 (机械师、医生、律师 vs 愚人金)
- **第6章：末日审判** - 终极对决 (9个求生者 vs 噩梦)

2. 灵活的调用方式

- **URL参数方式**: `battle.html?chapter=2`
- **JavaScript接口**: `window.setBattleChapter(3)`
- **程序化调用**: 适配 `plotmaneger` 跳转

3. 完善的错误处理

- 无效章节数自动回退到第1章
- 配置验证和兼容性检查

- 详细的控制台日志输出

📁 新增和修改的文件

新增文件：

1. `src/core/ChapterConfig.ts` - 章节配置管理器
2. `README_章节配置系统.md` - 详细使用说明文档
3. `chapter-test.html` - 功能演示和测试页面

修改文件：

1. `src/main.ts` - 集成章节配置系统，提供全局接口

编译生成：

- `dist/core/ChapterConfig.js` - 编译后的章节配置代码

🚀 如何在主游戏中使用

方案A：从 plotmaneger 跳转时传递章节参数

```
// 在您的 plotmaneger 脚本中
function enterBattleWithChapter(chapterNumber) {
    // 方法1：URL参数跳转
    window.location.href = `battle0/index.html?chapter=${chapterNumber}`;

    // 方法2：如果在同一页面，直接调用
    if (typeof window.setBattleChapter === 'function') {
        window.setBattleChapter(chapterNumber);
    }
}

// 使用示例
enterBattleWithChapter(3); // 进入第3章战斗
```

方案B：动态设置章节

```
// 设置章节并重新初始化游戏
window.battleSystem.setBattleChapter(2);

// 获取当前章节信息
const chapterInfo = window.battleSystem.getCurrentChapterInfo();
console.log(`当前：第<span class="math-inline" style="display:
inline;"><math xmlns="http://www.w3.org/1998/Math/MathML"
display="inline"><mrow><mrow><mi>c</mi><mi>h</mi><mi>a</mi><mi>p</
mi><mi>t</mi><mi>e</mi><mi>r</mi><mi>I</mi><mi>n</mi><mi>f</
mi><mi>o</mi><mo>&#x0002E;</mo><mi>c</mi><mi>h</mi><mi>a</
mi><mi>p</mi><mi>t</mi><mi>e</mi><mi>r</mi><mi>N</mi><mi>u</
mi><mi>m</mi><mi>b</mi><mi>e</mi><mi>r</mi></mrow><mi>章</
mi><mo>&#x02212;</mo></mrow></math></span>{chapterInfo.chapterName}
`);

// 获取所有章节列表
const allChapters = window.battleSystem.getAllChaptersInfo();
```

🎮 测试和验证

1. 功能测试页面

打开 `chapter-test.html` 可以：
- 查看所有章节配置
- 切换不同章节
- 测试URL参数功能
- 验证JavaScript接口

2. 控制台验证

游戏启动时会输出详细信息：

```
================ 游戏初始化 ================
检测到章节数: 2
=============== 章节信息 ===============
第 2 章: 法理之争
描述: 在庄园的法庭中，智慧型求生者与邪恶进行智力对决
可选求生者: ['lawyer', 'novelist', 'reporter']
可选监管者: ['小丑']
```

3. 界面提示

游戏会在左上角显示当前章节信息卡片，包括：
- 章节编号和名称
- 章节描述
- 可选求生者和监管者列表

🛠 技术架构

核心类和接口

```
// 章节配置数据结构
interface ChapterBattleConfig {
    chapterNumber: number;
    chapterName: string;
    description: string;
    availableSurvivors: string[];
    availableHunters: { name: string; hp: number; description?:
string; }[];
}

// 章节配置管理器
class ChapterConfigManager {
    getChapterConfig(chapterNumber: number): ChapterBattleConfig;
    getAllChapters(): ChapterBattleConfig[];
    validateChapterSurvivors(chapterNumber: number,
survivorLibrary: any): string[];
    getChapterFromURL(): number;
}
```

全局接口

```
window.battleSystem = {
    setBattleChapter(chapterNumber: number): void,
    getCurrentChapterInfo(): ChapterInfo,
    getAllChaptersInfo(): ChapterInfo[]
}

// 兼容性接口
window.setBattleChapter(chapterNumber: number): void
window.setChapter(chapterNumber: number): void
```

🔧 自定义和扩展

修改章节配置

直接编辑 `src/core/ChapterConfig.ts` 中的配置：

```
// 修改第1章的求生者
this.chapterConfigs[1] = {
    // ... 其他配置
    availableSurvivors: ['doctor', 'lawyer', 'mechanic'], // 修改这
里
    // ...
};
```

添加新章节

```
// 在 initializeChapterConfigs() 中添加
this.chapterConfigs[7] = {
    chapterNumber: 7,
    chapterName: "新的挑战",
    description: "新章节的描述",
    availableSurvivors: ['doctor', 'lawyer'],
    availableHunters: [{ name: '新监管者', hp: 220 }]
};
```

添加新求生者

1. 在 `SurvivorLibrary.ts` 中定义新求生者
2. 在章节配置的 `availableSurvivors` 中添加ID
3. 重新编译：`npx tsc`

🎯 效果展示

游戏启动效果

- 系统自动检测URL中的章节参数
- 显示对应章节的求生者选择界面
- 左上角出现章节信息提示卡片
- 控制台输出详细的配置信息

章节切换效果

- 调用 `setBattleChapter()` 后立即更新配置
- 重新显示求生者选择界面
- 更新章节信息显示

- 输出成功切换的状态信息

✨ 特色功能

1. **智能检测** - 自动从URL参数或JavaScript调用中获取章节数
2. **错误恢复** - 无效配置时自动回退到安全状态
3. **向后兼容** - 不破坏原有的游戏流程和代码
4. **直观显示** - 美观的章节信息提示界面
5. **灵活扩展** - 易于添加新章节和修改现有配置

🔗 与主游戏的集成

现在您可以在主游戏的 `plotmaneger` 中这样调用：

```
// 进入第N章战斗的函数
function startChapterBattle(chapterNumber) {
    // 跳转到战斗系统并传递章节参数
    window.location.href = `battle0/index.html?chapter=$
{chapterNumber}`;
}

// 在剧情管理器中使用
plotmaneger.onChapterComplete = function(completedChapter) {
    const nextChapter = completedChapter + 1;
    if (nextChapter <= 6) {
        startChapterBattle(nextChapter);
    }
};
```

🎊 总结

这个章节配置系统完美地解决了您的需求：

✅ **支持6个不同章节的配置**
✅ **每章节有独特的求生者和监管者组合**
✅ **可以从 plotmaneger 传递章节数**
✅ **提供多种调用方式**
✅ **具有完善的错误处理**
✅ **保持代码的可维护性和可扩展性**

现在您的战斗系统已经完全支持章节化配置了！🎉