崂山冒险游戏完整设计文档

1. 项目概述

这是一个基于控制台的回合制RPG冒险游戏，玩家扮演勇者探索崂山地图，与各种怪物战斗，收集技能和道具，最终挑战强大的龙王BOSS。

2. 核心文件结构

```
崂山冒险游戏/
├── main.cpp                # 程序入口点
├── Game.h                  # 游戏主控制器
├── Player.h                # 玩家类定义
├── Map.h                   # 地图系统
├── Room.h                  # 房间类
├── Enemy.h                 # 敌人类及工厂
├── Battle.h                # 战斗系统
├── Skill.h / Skill.cpp     # 技能系统
├── Item.h / Item.cpp       # 道具系统
├── SaveGame.h              # 存档系统
├── Colors.h / Colors.cpp   # 控制台颜色输出
├── AsciiMap.h              # ASCII艺术地图渲染
├── AudioManager.h / AudioManager.cpp  # 音频系统
├── SoundEffects.h          # 音效定义
├── miniaudio.h             # 第三方音频库
└── map_demo.cpp            # 地图演示程序
```

Fence 1

3. 各模块详细说明

3.1 1. 游戏主控制器 (Game.h)

- **功能**: 游戏的总控制器，管理游戏循环、输入处理、游戏状态等
- **主要职责**:
  - 初始化游戏
  - 处理用户输入
  - 管理游戏状态（运行中、胜利、失败）
  - 显示游戏界面
  - 处理房间进入事件
  - 商店系统交互

3.2 2. 玩家系统 (Player.h)

- **功能**: 玩家角色的定义和管理
- **主要职责**:
  - 管理玩家属性（生命值、攻击力、防御力、等级等）
  - 管理技能背包
  - 管理道具背包
  - 经验值和升级系统
  - 金币管理
  - 状态显示

3.3 3. 地图系统 (Map.h, Room.h)

- **功能**: 游戏世界的地图和房间管理
- **主要职责**:
  - 生成8x3的随机地图
  - 管理房间类型（起始点、怪物房、商店、休息房、BOSS房）
  - 玩家位置跟踪
  - 房间访问状态管理
  - ASCII艺术地图渲染 (AsciiMap.h)

3.4 4. 敌人系统 (Enemy.h)

- **功能**: 游戏中敌人的定义和管理
- **主要职责**:
  - 定义多种敌人类型（史莱姆、哥布林、骷髅兵、兽人、龙王）
  - 敌人AI行为
  - 掉落物系统
  - 敌人工厂模式创建

3.5 5. 战斗系统 (Battle.h)

- **功能**: 回合制战斗机制
- **主要职责**:
  - 玩家与敌人的回合制战斗
  - 技能使用
  - 道具使用
  - 逃跑机制
  - 战斗奖励计算

3.6 6. 技能系统 (Skill.h, Skill.cpp)

- **功能**: 玩家和敌人的技能管理
- **主要职责**:
  - 定义多种技能类型（致死打击、生命吸取、复仇等）
  - 技能稀有度系统（普通到传说）
  - 技能效果实现
  - 技能工厂模式创建

3.7 7. 道具系统 (Item.h, Item.cpp)

- **功能**: 游戏中的道具管理
- **主要职责**:
  - 定义多种道具类型（治疗药水、能量凝胶、烤鸡等）
  - 道具效果实现
  - 道具工厂模式创建
  - Buff/Debuff系统（如兴奋剂效果）

3.8 8. 存档系统 (SaveGame.h)

- **功能**: 游戏进度的保存和加载
- **主要职责**:
  - 保存玩家状态

- 保存地图状态
- 读取存档文件
- 删除存档

3.9 9. 视觉系统 (Colors.h, Colors.cpp)

- **功能**: 控制台彩色输出
- **主要职责**:
  - 跨平台颜色支持（Windows和Linux/Mac）
  - 各种颜色输出函数
  - 标题和分隔符美化

3.10 10. 音频系统 (AudioManager.h, AudioManager.cpp, SoundEffects.h)

- **功能**: 游戏音频播放
- **主要职责**:
  - 背景音乐播放
  - 音效播放
  - 音量控制
  - 使用miniaudio库实现

4. 游戏流程

1. **启动游戏**: 显示主菜单，选择新游戏或继续游戏
2. **创建角色**: 输入角色名称，初始化玩家属性
3. **探索地图**: 使用WASD移动，探索不同类型的房间
4. **房间事件**:
   - 怪物房：进入战斗
   - 商店房：购买技能和道具
   - 休息房：恢复生命值
   - BOSS房：挑战最终BOSS
5. **战斗系统**: 回合制战斗，可使用普通攻击、技能、道具或逃跑
6. **升级成长**: 通过击败敌人获得经验值和金币，升级提升属性
7. **存档系统**: 可随时保存游戏进度
8. **游戏目标**: 击败足够数量的怪物后挑战并击败龙王BOSS

5. 技术特点

- **面向对象设计**: 使用C++类和继承实现模块化设计
- **工厂模式**: 技能和道具使用工厂模式创建
- **存档系统**: 文本文件格式保存游戏进度
- **跨平台**: 支持Windows和Linux/Mac系统
- **色彩输出**: 丰富的控制台颜色效果
- **音频支持**: 背景音乐和音效播放
- **ASCII艺术**: 精美的地图和界面显示

游戏设计详细说明

1. 游戏名称

重返崂山!

2. 核心玩法 ✅ 已实现

- 玩家与敌人进行回合制战斗
- 玩家在自己的回合可以选择:
  - **普通攻击**: 对敌人造成基础伤害，10%暴击几率
  - **技能攻击**: 使用一次性技能，造成更高伤害或特殊效果
  - **道具使用**: 使用血瓶、炸弹等战斗道具
  - **逃跑**: 70%成功率逃离战斗
- 战斗模式类似于《宝可梦》的回合制系统

3. 游戏目标 ✅ 已实现

- 探索ASCII艺术风格的3x8地图
- 击败15只普通敌怪获得足够实力
- 挑战并击败强大的崂山龙王BOSS取得最终胜利

4. 主要特性 ✅ 已实现

4.1 💀 敌人系统

- 多种敌人类型：史莱姆、哥布林、骷髅兵、兽人、龙王BOSS
- 击败敌人获得经验值、金币和技能奖励
- 每个敌人有独特的AI行为模式和风味描述

4.2 🗺️ 地图系统

- **ASCII艺术地图**: 精美的字符艺术地图显示
- **房间类型**:
  - **起始房** (S): 安全的出发点
  - **怪物房** (M): 遭遇敌人触发战斗，清理后显示为已清理
  - **商店房** ($): 购买技能和道具
  - **休息房** (E): 一次性完全恢复生命值
  - **BOSS房** (B): 最终挑战龙王
- **迷雾系统**: 未访问房间显示为?，增加探索乐趣

4.3 ⚔️ 技能系统

- **一次性使用**: 所有技能使用后消失，增加策略性
- 玩家最多可持有10个技能
- **初始技能**: 开局获得3个随机技能
- **获得途径**: 战斗奖励、商店购买

- **技能稀有度**: 普通、不常见、稀有、史诗、传说五个等级

4.4 🎒 道具系统

- **治疗药水**: 恢复50HP
- **能量凝胶**: 恢复30%最大HP
- **急救喷雾**: 恢复70%最大HP
- **烤鸡**: 完全恢复生命值
- **其他特殊道具**: 兴奋剂、黑暗契约、瓶中妖精等
- **道具背包**: 最多携带8个道具
- 支持战斗中和非战斗状态下使用

4.5 💰 经济系统

- 击败敌人获得金币奖励
- 商店购买技能和道具
- 起始金币： 100

4.6 📈 升级系统

- 通过战斗获得经验值
- 每级所需经验值: level × 60
- 升级属性提升: HP+30, 攻击+5, 防御+3

5. 详细游戏设计 - 当前实现

5.1 角色属性系统 ✅ 已实现

- **玩家初始属性**:
  - 生命值: 150 HP
  - 攻击力: 22
  - 防御力: 8
  - 经验值: 0 EXP
  - 等级: 1
  - 金币: 100
- **升级系统**:
  - 每级所需经验值: level × 60
  - 升级时属性提升: HP+30, 攻击+5, 防御+3

5.2 敌人设计 ✅ 已实现

- **史莱姆** (普通): HP=30, 攻击=8, 防御=2, 经验=25, 金币=15
- **哥布林** (普通): HP=45, 攻击=12, 防御=3, 经验=35, 金币=25
- **骷髅兵** (普通): HP=55, 攻击=15, 防御=5, 经验=45, 金币=30
- **兽人** (精英): HP=80, 攻击=20, 防御=8, 经验=70, 金币=50
- **崂山龙王** (BOSS): HP=300, 攻击=35, 防御=15, 经验=200, 金币=200

5.3 技能系统设计 ✅ 已实现

5.3.1 攻击技能:

- **致死打击**: 对生命值较低的敌人造成毁灭性打击
- **生命吸取**: 攻击敌人并吸取生命值
- **复仇**: 低血量时的强力反击
- **死亡标记**: 标记敌人，使其在5回合内受到更多伤害
- **致命一击**: 下次攻击造成额外伤害
- **虚弱打击**: 攻击并使敌人下次受伤增加
- **财富一击**: 伤害基于金币数量
- **运气骰子**: 投掷骰子决定伤害倍率
- **同生共死**: 与敌人共享伤害
- **起死回生**: 血量越少伤害越高
- **欺诈**: 使用敌人的攻击力进行攻击

5.4 地图设计 ✅ 已实现

- **总房间数**: 24个房间 (3×8网格布局)
- **ASCII艺术显示**: 每个房间都有独特的视觉符号
- **房间分布**:
  - 起始房: 1个 (左上角)
  - 怪物房: ~19个 (随机分布)
  - 商店房: 3个 (随机分布)
  - 休息房: 1个 (随机分布)
  - BOSS房: 1个 (右下角)
- **移动系统**: W/A/S/D四方向移动，边界检测

5.5 战斗系统 ✅ 已实现

- **伤害计算**: 最终伤害 = max(1, 攻击力 - 防御力)
- **暴击系统**: 10%几率造成1.5倍伤害
- **回合顺序**: 玩家先手行动
- **战斗选项**: 普通攻击、技能、道具、逃跑
- **逃跑机制**: 70%成功率，失败继续战斗
- **胜利条件**: 敌人HP降为0
- **失败条件**: 玩家HP降为0

5.6 商店系统 ✅ 已实现

- **随机技能**: 100金币购买随机技能（高稀有度）
- **治疗药水**: 恢复50HP，价格30金币
- **能量凝胶**: 恢复30%最大HP，价格25金币
- **急救喷雾**: 恢复70%最大HP，价格80金币
- **烤鸡**: 完全恢复，价格120金币
- 背包容量检查和金币验证

5.7 界面系统 ✅ 已实现

5.7.1 固定界面布局:

- **顶部**: 玩家状态信息 (生命值条、等级、经验、金币)
- **中部**: ASCII艺术地图 (带图例和状态信息)
- **底部**: 操作控制说明

5.7.2 输入控制:

地图探索:

- W/A/S/D: 四方向移动
- M: 显示地图
- P: 显示玩家状态
- I: 查看技能背包
- B: 查看道具背包
- H: 帮助说明
- C: 清屏刷新
- Q: 退出游戏菜单
- X: 手动保存游戏
- T: 音乐开关

战斗界面:

- 1: 普通攻击
- 2: 使用技能 (进入技能选择菜单)
- 3: 使用道具 (进入道具选择菜单)
- 4: 尝试逃跑

5.8 存档系统 ✅ 已实现

- **自动保存**: 游戏进度自动保存
- **手动保存**: 按X键手动保存
- **断点继续**: 支持从存档点继续游戏
- **存档管理**: 主菜单提供存档删除选项

5.9 中文支持 ✅ 已实现

- **UTF-8编码**: 完整支持中文显示
- **中文输入**: 支持中文角色姓名输入
- **控制台设置**: 自动配置Windows控制台编码

6. 技术实现

6.1 编译环境

- **编译器**: MinGW-w64 g++ 13.2.0
- **C++标准**: C++17
- **静态链接**: 无需额外DLL文件
- **编译输出**: game_latest.exe

6.2 核心类设计 - 实际实现

6.2.1 1. Character (角色基类) ✅

```
class Character {
protected:
    int hp, maxHP, attack, defense, level;
    string name;
public:
    virtual void takeDamage(int damage);
    virtual int dealDamage() const;
    virtual bool isAlive() const;
    virtual void displayStats() const = 0;
};
```

Fence 2

6.2.2 2. Player (玩家类) ✅

```
class Player : public Character {
private:
    int exp, gold;
    vector<Skill*> skills;      // 技能列表，使用后删除
    vector<Item*> inventory;    // 道具背包
    static const int MAX_SKILLS = 10;
    static const int MAX_ITEMS = 8;
public:
    void gainExp(int amount);
    void levelUp();
    bool addSkill(Skill* skill);
    bool useSkill(int index, Character& target);  // 一次性使用
    bool addItem(Item* item);
    bool useItem(int index);
    bool useItemInBattle(int index, Character& enemy);
    void displayStats() const override;
    void displaySkillBag() const;
    void displayInventory() const;
    void restoreToFullHP();
};
```

Fence 3

6.2.3 3. Battle (战斗类) ✅

```
class Battle {
private:
    Player* player;
    Enemy* enemy;
    bool battleEnded;
    bool playerWon;
    bool playerEscaped;  // 区分逃跑和死亡
    vector<DropItem> battleRewards;
public:
    bool startBattle();
    bool handleAttack();
    bool handleSkillUse();
    bool handleItemUse();
    bool handleEscape();
    void handleVictory();
    void handleDefeat();
    bool didPlayerEscape() const;
};
```

Fence 4

6.2.4 4. AsciiMapRenderer (地图渲染) ✅

```
class AsciiMapRenderer {
public:
    static void displayAsciiMap(Room* const rooms[HEIGHT][WIDTH],
                                int playerX, int playerY, int monstersDefeated);
private:
    static void generateRoomAscii(RoomAscii& ascii, Room* room, bool hasPlayer);
    static void generateStartRoom(RoomAscii& ascii, bool hasPlayer);
    static void generateMonsterRoom(RoomAscii& ascii, bool hasPlayer, bool cleared);
    static void generateShopRoom(RoomAscii& ascii, bool hasPlayer);
    static void generateEmptyRoom(RoomAscii& ascii, bool hasPlayer);
    static void generateBossRoom(RoomAscii& ascii, bool hasPlayer);
};
```

Fence 5

6.2.5 5. Game (游戏主控制) ✅

```
class Game {
private:
    Player* player;
    Map* gameMap;
    bool gameRunning;
    bool gameWon;
public:
    void run();
    void gameLoop();
    void displayGameInterface();  // 固定布局界面
    void handleMovement(char direction);
    void handleRoomEntry();
    void handleShop();
    void showStartMenu();
    bool startNewGame();
    bool loadGame();
};
```

Fence 6

6.3 设计模式应用 ✅

- **工厂模式**: SkillFactory, EnemyFactory, ItemFactory 创建游戏对象
- **单例模式**: Colors 颜色管理类
- **策略模式**: 不同房间类型的处理策略
- **状态模式**: 游戏状态管理 (菜单、游戏中、战斗等)

7. 游戏特色

7.1 🎨 视觉设计

- **ASCII艺术风格**: 复古终端美学
- **彩色界面**: 使用ANSI颜色codes增强视觉效果
- **清晰布局**: 固定三段式界面布局
- **动态更新**: 实时刷新地图和状态信息

7.2 🎮 游戏体验

- **即时输入**: getch()实现无需回车的快速操作
- **策略深度**: 一次性技能系统要求合理规划资源

- **探索乐趣**: 迷雾系统和随机地图布局
- **成就感**: 清晰的进度指示和奖励反馈

7.3 🔧 技术特色

- **跨平台兼容**: 支持Windows/Linux控制台
- **内存安全**: RAII和智能指针模式
- **模块化设计**: 高内聚低耦合的类结构
- **扩展性强**: 工厂模式便于添加新内容

8. 平衡性分析 - 实测数据

8.1 战斗难度曲线 ✅

- **前期** (1-5只怪): 史莱姆为主，建立信心
- **中期** (6-10只怪): 哥布林、骷髅兵混合，需要技能配合
- **后期** (11-15只怪): 精英兽人出现，考验策略运用
- **BOSS战**: 龙王超高血量，需要充分准备和技能储备

8.2 经济平衡 ✅

- 击败15只普通敌人平均获得: ~400金币
- 商店核心物品成本: 技能100+治疗药水30+能量凝胶25+急救喷雾80=235金币
- 有余量购买多个道具，但需要做选择

8.3 技能获取策略 ✅

- 开局3技能 + 战斗掉落 + 商店购买 = 约8-12个技能获得机会
- 10个技能上限迫使优化技能组合
- 一次性使用增加每次战斗的决策深度

9. 部署说明

9.1 编译方法

```
g++ -std=c++17 -Wall -Wextra -O2 -static-libgcc -static-libstdc++ -o game_latest.exe main.cpp Colors.cpp Item.cpp
```

Fence 7

9.2 运行要求

- Windows 10+ 或 Linux 系统
- 支持UTF-8的控制台环境
- 约1MB磁盘空间

9.3 文件结构

```
claudecode/
├── game_latest.exe     # 主程序
├── compile.bat         # 编译脚本
├── game_save.txt       # 存档文件(自动生成)
├── main.cpp           # 程序入口
├── Colors.cpp         # 颜色系统实现
├── Item.cpp           # 道具系统实现
└── 各种.h头文件       # 类定义
```

Fence 8