|  |  |
| --- | --- |
| 产品名称 | 密级 |
| WOW!我的英雄最小模型 | 内部资料 |
| 项目编号 | 共页 |
| WJYL20130201 |

《WOW!我的英雄》最小模型设计说明书

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制 | 黄进 | 日期 | 2013-5-10 |
| 评审人 | 黄进 | 日期 | 2013-5-10 |
| 批准 | 刘永明 | 日期 | 2013-5-10 |

玩嘉娱乐工作室

# 目录

# 引言

## 编写目的

此概要设计说明书是为了说明整个《WOW!我的英雄》最小模型游戏的体系架构，以及需求用例的各个功能点在架构中的体现，为系统的详细设计人员进行详细设计时的输入参考文档。

## 背景

软件系统的名称：《WOW!我的英雄》最小模型

开发工具：Unity3D，MonoDevelop

开发语言：C# Script

开发者： 玩嘉娱乐技术部

## 定义

**最小模型：**《WOW!我的英雄》最小模型简称。

## 参考资料

# 总体设计

## 简述

最小模型包括游戏的战斗系统、道具系统、技能系统、关卡系统、AI系统、部分必不可少的特效、UI，不包括数据库、大部分特效、大部分UI、广告及道具收费模块。最小模型的分析设计采用面向对象的技术，应用Visio等工具进行辅助设计。

## 架构设计

### 分层图



### 顶级系统包图



包类说明：

* 玩家操作：触屏、鼠标操作的识别，对玩家意图的判断。
* 游戏逻辑：包括战斗系统、界面、AI系统、技能系统、道具系统、关卡系统等子系统。
* 数据接口：提供对XML数据源的处理和读取方法。
* 数据实体：主要以XML文档的形式存储游戏中使用到的数据，其中包括主角状态、道具、怪物、关卡、技能。

### 游戏逻辑包图



包类说明：

* 战斗系统：包括战斗主体（主角、敌人、Boss），战斗主体的控制、技能接口、AI接口、道具接口，战斗逻辑。
* 界面：最小模型中必不可少的操作界面。
* AI系统：与战斗系统中AI接口协同工作的怪物智能行为。
* 技能系统：与战斗系统中技能接口协同工作的技能逻辑及效果。
* 道具系统：与战斗系统中道具接口协同工作的道具逻辑及效果。
* 关卡系统：负责关卡的初始化、怪物的刷新、胜利条件及奖励。

### 数据库关系图



包类说明：

* 主角状态：包括属性级别、技能级别、关卡进度、金币钻石、道具情况。
* 技能：包括各技能每一级别的等级影响系数。
* 关卡：包括各关卡的刷怪情况、胜利条件、奖励情况。
* 道具：包括各道具的属性加成、效果脚本的指派。
* 属性：主角属性各级别数据。
* 怪物：怪物属性、技能槽、AI槽。

## 接口设计

### 界面框架设计



### 

### 顶级系统接口

#### 玩家操作系统

|  |  |
| --- | --- |
| **OP\_GetCamMoveDelta** | |
| 函数原型 | Vector2 OP\_GetCamMoveDelta() |
| 功能 | 获取一小段时间内玩家操作摄像机移动的方向向量 |
| 参数说明 | 无 |
| 返回值说明 | 摄像机移动的方向向量 |
| 使用方法 | 在摄像机的Update中调用，若检测到不为零向量则移动摄像机 |
| 备注 | 自动屏蔽微小拖拽导致的移动 |

#### 游戏系统

|  |  |
| --- | --- |
| **GS\_GetGameState** | |
| 函数原型 | GameState GS\_GetGameState() |
| 功能 | 获取当前游戏状态 |
| 参数说明 | 无 |
| 返回值说明 | |  |  | | --- | --- | | enum GameState | | | Fight | 战斗状态 | | Pause | 暂停状态 | | Settlement | 结算状态 | |
| 使用方法 | 无 |
| 备注 | 无 |

#### 数据接口

|  |  |
| --- | --- |
| **DB\_GetPlayerData** | |
| 函数原型 | PlayerData DB\_GetPlayerData() |
| 功能 | 获取当前主角状态 |
| 参数说明 | 无 |
| 返回值说明 | PlayerData类，用于存储玩家数据 |
| 使用方法 | 此脚本在主角控制脚本内调用，并用于初始化主角数据。 |
| 备注 | 主角属性、城墙和技能的强化，由于具有较为明显的规律，故仅在数据库中存储等级，而不存储每个级别的相应参数值，这些参数直接在程序中计算出来。 |
| **DB\_GetLevelData** | |
| 函数原型 | LevelData DB\_GetLevelData(String levelId) |
| 功能 | 获取指定关卡数据 |
| 参数说明 | 指定关卡ID |
| 返回值说明 | LevelData类，用于存储关卡数据。 |
| 使用方法 | 此脚本在关卡控制脚本内调用，并用于初始化关卡数据。 |
| 备注 | 无 |
| **DB\_GetWaveData** | |
| 函数原型 | WaveData DB\_GetWaveData (String waveId) |
| 功能 | 获取指怪物波数据 |
| 参数说明 | 指定怪物波ID |
| 返回值说明 | WaveData类，用于存储怪物波数据。 |
| 使用方法 | 此脚本在关卡控制脚本内调用，并用于初始化某一关的怪物波数据。 |
| 备注 | 无 |
| **DB\_GetMonsterData** | |
| 函数原型 | MonsterData DB\_GetMonsterData(String monsterId) |
| 功能 | 获取指定怪物数据 |
| 参数说明 | 指定怪物ID |
| 返回值说明 | MonsterData类，用于存储怪物数据。 |
| 使用方法 | 此脚本在怪物控制脚本内调用，并用于初始化怪物数据。 |
| 备注 | 凡ID结尾者，指须从数据库中调用的数据对象；凡scriptName结尾者，则按其名称从脚本库中添加组件。下同，不再重复。  本函数中，skillIDs用于获取属于该怪物的所有技能ID，用户须根据此ID在数据库中获取相应数据并自行初始化；aiScriptNames用于获取属于该怪物的所有AI行为，用户须根据此名称，从脚本库中为此GameObject添加相应组件。 |
| **DB\_GetSkillData** | |
| 函数原型 | SkillData DB\_GetSkillData(String skillID)  SkillData DB\_GetSkillData(String skillID, int level) |
| 功能 | 获取指定主角或者怪物技能数据 |
| 参数说明 | skillID: 指定技能ID  level: 主角技能的等级 |
| 返回值说明 | SkillData类，用于存储技能数据。 |
| 使用方法 | 此脚本在技能脚本内调用，并用于初始化技能数据。 |
| 备注 | a~f四个变量用于存放影像技能效力的数值，其使用规则由技能表的设计和技能脚本定义。 |
| **DB\_GetItemData** | |
| 函数原型 | ItemData DB\_GetItemData(String itemID) |
| 功能 | 获取指定道具数据 |
| 参数说明 | 指定道具ID |
| 返回值说明 | ItemData类，用于存储道具数据。 |
| 使用方法 | 此脚本在主角脚本内调用，并用于初始化主角道具数据。 |
| 备注 | 之所以要区分道具类型，是因为主角的防具直接绑定在主角身上，而武器与主角是分开的两个图层。 |
| **DB\_SetPlayerData** | |
| 函数原型 | Status DB\_SetPlayerData(PlayerData data) |
| 功能 | 获取指定道具数据 |
| 参数说明 | 最新主角数据 |
| 返回值说明 | |  |  | | --- | --- | | enum Status | | | Error | 错误 | | Ok | 正确 | | Infeasible | 输入参数不正确 | | Outrange | 超出数据范围 | |
| 使用方法 | 用于保存玩家数据 |
| 备注 |  |

# 子系统设计

## 战斗系统

### 子系统说明

战斗系统包括：角色、游戏状态管理、玩家控制器、场景控制器、摄像机控制器。

* 角色：定义游戏角色，实现角色攻击、移动、施放技能等基础战斗行为。
* 游戏状态管理：定义游戏状态，实现游戏状态间的切换。
* 玩家控制器：实现于对玩家角色的控制。
* 场景控制器：实现场景的布置，场景触发器的控制。
* 摄像机控制器：实现摄像机的跟随主角、移动、跳转、战斗特效。

### 类图

#### 角色



注：本图为概念设计图，主要目的为阐述总体构架和设计思想，一些具体元素可能被省略，编码时可根据实际情况进行修订和调整。

#### 游戏状态管理

#### 玩家控制器

#### 场景控制器

#### 摄像机控制器

### 类说明

#### 角色

##### Role



概要：角色基类，继承MonoBehavior（是一个脚本Component），存储角色基础属性，实现移动、转身、站立、受到伤害等。

主要属性描述：

|  |  |
| --- | --- |
| int attack | 攻击 |
| int armor | 防御 |
| int hp | HP |
| int mp | MP |
| float mpRegen | MP恢复 |
| float dodgeRate | 闪避率 |
| float attSpeed | 攻击速度（相对值，范围[0,200]，据此值计算出攻击间隔，单位秒。） |
| float speed | 移动速度（相对值，范围[0,200], 据此值计算出单位时间移动距离，单位像素。） |
| float criticalRate | 暴击率 |
| float attRange | 攻击距离 |
| Size faceDir | 脸朝向（左、右） |
| RoleState state | 角色状态（移动、站立、施放技能） |

主要方法描述：

|  |  |
| --- | --- |
| void Init(int attPower,  int armor,  int hp,  int mp,  float mpRegen,  float dodgeRate,  float attSpeed,  float speed,  float criticalRate,  float attRange,  Size faceDir) | 角色基础属性初始化，注意，为了控制各对象初始化时机，请勿使用继承自MonoBehavior的Start()方法对属性进行初始化。 |
| void Update() | 继承自MonoBehavior，根据state对角色属性进行更新 |
| void MoveDirectly(Size dir) | 直接移动角色（不顾障碍物） |
| void Move(Size dir) | 移动角色 |
| void Stand(Size dir) | 站立 |
| void Turn (Size dir) | 转向 |
| void OnCast (Skill casted) | 施放技能时触发的事件 |
| void ApplyDamage (float damage) | 造成伤害 |
| void CostMp (float cost) | 消耗魔法值 |

##### Character



概要：更高级的角色基类，是角色动画、技能和AI的载体，这些对象在此类中存有引用，并且通过此类实现功能。

主要属性描述：

|  |  |
| --- | --- |
| BoneAnimation charBoneAnim | 角色动画对象的引用 |
| List<Skill> listSkill | 所有技能对象的引用 |
| List<Ai> listAi | 所有AI对象引用 |

主要方法描述：

|  |  |
| --- | --- |
| void Init(int attPower,  int armor,  int hp,  int mp,  float mpRegen,  float dodgeRate,  float attSpeed,  float speed,  float criticalRate,  float attRange,  Size faceDir,  BoneAnimation smRoleAnim) | 增多角色动画对象的初始化 |
| void Update() | 继承自MonoBehavior，根据state对角色属性进行更新 |

##### Player



概述：供玩家角色拓展所用

主要属性描述：略

主要方法描述：略

##### Monster



概述：供怪物角色拓展所用

主要属性描述：略

主要方法描述：略

#### 游戏状态管理

#### 玩家控制器

#### 场景控制器

#### 摄像机控制器

### GUI设计

## 道具系统

### 子系统说明

### 类图

### 类说明

### GUI设计

## 技能系统

### 子系统说明

技能系统包括技能、技能效果、增/减益状态（Buff/DeBuff）。技能类继承与MonoBehavior，是一个脚本组件，它在Character里面保存了引用。一个技能对应一个或多个技能效果，技能在击中判定后出发技能效果内定义的击中事件。



* 技能：主要负责技能的触发、动画、施放、击中判定、角色施放和击中事件的调用。
* 技能效果：技能击中事件的定义，即现实技能击中后的效果。
* 增/减益状态：继承MonoBehavior，是一个脚本组件，可通过技能效果中定义的击中事件添加到物体上。

### 类图

#### 技能



注：本图为概念设计图，主要目的为阐述总体构架和设计思想，一些具体元素可能被省略，编码时可根据实际情况进行修订和调整。

#### 技能效果



注：本图为概念设计图，主要目的为阐述总体构架和设计思想，一些具体元素可能被省略，编码时可根据实际情况进行修订和调整。

#### 增/减益效果

暂缺

### 类说明

#### Skill



概要：技能基类，继承MonoBehavior（是一个脚本Component），主要负责技能的触发、动画、施放、击中判定、角色施放和击中事件的调用。

主要属性描述：

|  |  |
| --- | --- |
| Role caster | 技能施放者的引用 |
| BoneAnimation triggerAnim | 用于击中判定的动画对象 |
| List<BoneAnimation> listSkillAnim | 施放技能时的其他动画对象 |
| int mpCost | 损耗的魔法 |
| float coldDown | 技能冷却时间 |
| event CastSkillEventHandler CastSkillEvent | 用于注册和触发施放技能时的事件 |
| event SkillHittedEventHandler SkillHittedEvent | 用于注册和触发技能击中时的事件（技能效果事件） |

主要方法描述：

|  |  |
| --- | --- |
| Init (Role caster,  BoneAnimation triggerAnim,  string triggerBoneName,  string triggerClipName,  int mpCost,  float coldDown) | 技能的初始化函数 |
| void CastSkillEventHandler (Skill casted) | 施放技能时的事件的处理者（Handler，或者称为委托） |
| bool SkillHittedEventHandler (Role caster, Role hitted) | 技能击中时的事件的处理者（Handler，或者称为委托） |
| bool CastSkill () | 施放技能 |
| void SkillHitted (ColliderTriggerEvent triggerEvent) | 技能击中 |

#### SkillEffect



概要：技能效果基类，计算基础伤害，定义击中事件。

主要属性描述：

|  |  |
| --- | --- |
| float basicDamage | 技能施放者对被攻击者的基础伤害 |

主要方法描述：

|  |  |
| --- | --- |
| bool OnHit (Role caster, Role hitted) | 计算基础伤害，如果击中对象为空则返回fasle |

### GUI设计

## 关卡系统

### 子系统说明

在关卡初始化阶段，主要负责调用数据管理系统将本关数据库数据（磁盘数据）存储到数据集（内存数据）中。同时，完成主角游戏物体的初始化。

在关卡进行阶段，主要负责根据读入数据集的各波数据，在规定的时间内初始化指定的怪物。

在关卡结束阶段，主要负责统计关卡数据，计算奖励，修改主角数据集数据并调用数据管理系统写入主角数据。

### 类图



### 类说明

##### LevelBuilder



概要：在关卡初始化阶段，主要负责调用数据管理系统将本关数据库数据（磁盘数据）存储到数据集（内存数据）中。同时，完成主角游戏物体的初始化。

在关卡进行阶段，主要负责根据读入数据集的各波数据，在规定的时间内初始化指定的怪物。

在关卡结束阶段，主要负责统计关卡数据，计算奖励，修改主角数据集数据并调用数据管理系统写入主角数据。

主要属性描述：

|  |  |
| --- | --- |
| DataManageSystem dataSystem | 数据管理系统对象 |
| PlayerData playerData | 数据集：玩家数据 |
| LevelData levelData | 数据集：关卡数据 |
| List<WaveData> waveDataSet | 数据集：怪物波数据  怪物波数据按顺序存储 |
| HashTable monsterDataSet | 数据集：怪物数据   |  |  |  | | --- | --- | --- | | 哈希表 | | | | Key | string | 怪物id | | Value | MonsterData | 怪物数据 | |
| HashTable ItemDataSet | 数据集：物品数据   |  |  |  | | --- | --- | --- | | 哈希表 | | | | Key | string | 物品id | | Value | ItemData | 物品数据 | |
| HashTable skillDataSet | 数据集：技能数据   |  |  |  | | --- | --- | --- | | 哈希表 | | | | Key | string | 技能id | | Value | SkillData | 技能数据 | |

主要方法描述：

|  |  |
| --- | --- |
| bool SpawnPlayer(string id) | 将指定的玩家GameObject创建并安放在场景中的出生点 |
| bool SpawnMonster(string id) | 将制定的怪物GameObject创建并在场景中指定位置刷新 |
| Awake() | 完成关卡系统的初始化工作，包括：   1. 新建数据管理系统对象。 2. 使用数据管理系统将主角数据及本关卡相关载入数据集。 3. 完成地形的初始化。 4. 完成主角的初始化。 |
| Update() | 按照数据集中的关卡数据，在规定的时间节点和时长内，动态刷新怪物 |
| Summarize() | 关卡结束条件形成后，对关卡进行总结并负责写入主角数据 |

## AI系统

### 子系统说明

系统管理包括系统数据备份和系统操作日志查询功能，只有系统管理员才有权限操作此功能。

系统数据备份：实现将系统数据备份到指定的路径。

操作日志查询：实现查询系统数据的修改历史及登录系统用户的操作记录。

### 类图



### 类说明



备份系统数据：执行数据备份操作，将当前系统数据库备份至指定路径。

记录系统操作日志：得到登录系统人员操作日志实体类数据，将日志信息保存至库中。

查询全部操作日志：执行查询，得到全部的系统操作日志实体类数据。

多条件查询操作日志：根据多条件（起始日期、截止日期、姓名、功能模块、动作类型）查询对象，查询得到系统操作日志实体类数据。



此类的说明同上。



此类的说明同上。



人员操作日志实体类数据。

### 界面设计

系统管理子系统页面包括：数据备份、操作日志两部分。

#### 数据备份页面



#### 操作日志页面



## 数据管理系统

### 子系统说明

数据管理系统包括：数据读取和数据存储两部分。

数据读取：实现玩家、怪物、关卡、道具、AI、技能等数据的读取。

数据存储：实现玩家属性等级、技能等级、城防等级、关卡进度、装备金钱更新等数据的存储。

### 类图



### 类说明

#### PlayerData

概要：用于存储玩家数据。

属性：

|  |  |
| --- | --- |
| int strength | 力量等级 |
| int health | 体质等级 |
| int agility | 敏捷等级 |
| int magic | 魔力等级 |
| int wallArmor | 坚化城墙等级 |
| int wallHealth | 加筑城墙等级 |
| int wrap | 包扎技能等级 |
| int assault | 突袭技能等级 |
| int earthquake | 地裂技能等级 |
| int tornado | 旋风技能等级 |
| List<String> currentItemIDs | 当前道具 |
| List<String> itemIDs | 持有道具 |
| List<String> unLockItemIDs | 已解锁道具 |
| int gold | 金币 |
| int diamond | 钻石 |
| int level | 当前关卡 |

方法：无

#### LevelData

概要：用于存储关卡数据。

属性：

|  |  |
| --- | --- |
| int waveNum | 怪物波数 |
| List<String> waveIDs | 本关各波 |
| int gold | 奖励金币 |

方法：无

#### WaveData

概要：用于存储怪物波数据。

属性：

|  |  |
| --- | --- |
| List<String> monsterIDs | 怪物类型 |
| List<float> rates | 怪物出现几率 |
| int monsterNum | 怪物总数 |
| float time | 下波怪物出现时间 |

方法：无

#### MonsterData

概要：用于存储怪物数据。

属性：

|  |  |
| --- | --- |
| String animName | 怪物动画文件 |
| MonsterType type | 怪物类型（普通、精英、Boss） |
| int attack | 攻击力 |
| int armor | 防御力 |
| int hp | 生命 |
| int range | 攻击距离 |
| int mp | 魔法值 |
| float dodgeRate | 闪避几率 |
| int mpRegen | 每秒魔法值恢复 |
| int attSpeed | 攻速 |
| int speed | 移动速度 |
| float criticlRate | 暴击率 |
| List<String> skillIDs | 技能ID |
| List<String> aiScriptNames | AI |
| int gold | 奖励金币 |

方法：无

#### SkillData

概要：用于存储技能数据。

属性：

|  |  |
| --- | --- |
| String clipName | 动画片段名（主角） |
| float a | 技能参数1（主角+怪物） |
| float b | 技能参数2（主角+怪物） |
| float c | 技能参数3（主角+怪物） |
| float d | 技能参数4（主角+怪物） |
| float e | 技能参数5（主角+怪物） |
| float f | 技能参数6（主角+怪物） |
| int mpCost | MP消耗（主角+怪物） |
| float coldDown | 冷却时间（主角+怪物） |
| String skillScriptName | 技能脚本名字（主角+怪物） |

方法：无

#### ItemData

概要：用于存储道具数据。

属性：

|  |  |
| --- | --- |
| ItemType type | 道具类型（武器或防具） |
| String itemName | 道具名字 |
| String descrition | 道具描述 |
| int attack | 攻击力 |
| int armor | 防御力 |
| float drinkBlood | 吸血（百分比） |
| float citicalRate | 暴击率（百分比） |
| float speed | 跑速增加（百分比） |
| float attackSpeed | 攻速增加（百分比） |
| float mpRegen | MP回复/秒 |
| int hp | HP增加 |
| String skillID | 技能ID |

方法：无

#### DataManageSystem

概要：用于读取玩家、关卡、怪物、道具等数据，并将最新玩家数据写入数据库。代码上主要实现数据接口模块的顶层接口函数。

属性：无

方法：

|  |  |
| --- | --- |
| PlayerData DB\_GetPlayerData() | 获取当前主角状态 |
| LevelData DB\_GetLevelData(String levelId) | 获取指定关卡数据 |
| WaveData DB\_GetWaveData (String waveId) | 获取指怪物波数据 |
| MonsterData DB\_GetMonsterData(String monsterId) | 获取指定怪物数据 |
| SkillData DB\_GetSkillData(String skillID)  SkillData DB\_GetSkillData(String skillID, int level) | 获取指定主角或者怪物技能数据 |
| ItemData DB\_GetItemData(String ItemID) | 获取指定道具数据 |
| ItemData DB\_GetItemData(String itemID) | 获取指定道具数据 |
| Status DB\_SetPlayerData(PlayerData data) | 用于更新主角数据 |

#### DataTool

概要：用于辅助DataManageSystem完成其功能。

属性：待定

方法：待定

# 约束和假定

描述系统设计中最主要的约束，这些是由客户强制要求并在需求说明书写明的。说明系统是如何来适应这些约束的。

　　另外如果本系统跟其它外部系统交互或者依赖其它外部系统提供一些功能辅助，那么系统可能还受到其它的约束。这种情况下，要求清楚地描述与本系统有交互的软件类型（比如某某某数据库软件，某某某EMail软件）以及这样导致的约束（比如只允许纯文本的Email）。

　　实现的语言和平台也会对系统有约束，同样在此予以说明。

　　对于因选择具体的设计实现而导致对系统的约束，简要地描述你的想法思路，经过怎么样的权衡，为什么要采取这样的设计等等。

包括运行环境的约束，简要地说明对本系统的运行环境（包括硬件环境和支持环境）的规定，详细说明参见附录C。

# 数据库设计

## 总体设计图



注：记录是一张表中的一行，字段是一张表中的一列。例如怪物表，有字段MonsterId，MonsterHP等，这些为列，“怪物1”的数据则使用第一行来存储。

本项目使用XML文档搭建数据库，在项目开发阶段，考虑到XML文档的阅读和编辑困难，本项目采用由Excel文档生成XML文档的方式，软件发布之后，即可脱离Excel独立运行：



## 数据表设计

注：表和表字段命名规范：

1. 表名的定义：

规范格式：表名（字母全部大写）

比如，主角数据表定义为：PLAYER。

1. 表字段的定义：

规范格式：表名+字段名称（超过三个字母单词缩写为前三个字母，首字母大写），以避免字段重名出现在不同表中，字段名称2个单词以上的，最后一个单词缩写，其余的仅大写首字母。

比如，表PLAYER中的字段标识Strength定义为：PlayerStr，Wall Armor定义为PlayerWArm。

1. 表字段类型定义：

规范格式：限定使用以下三种类型：int，float，String。其中前两种用来存储数值，字符串类型用来存储ID、文字、文件名等。

### 角色属性表（PLAYER）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| PlayerId | 玩家ID | Yes | int | 自增长主键 |
| PlayerStr | 力量等级 | Yes | int |  |
| PlayerHea | 体质等级 | Yes | int |  |
| PlayerAgi | 敏捷等级 | Yes | int |  |
| PlayerMag | 魔力等级 | Yes | int |  |
| PlayerWArm | 坚化城墙等级 | Yes | int |  |
| PlayerWHea | 加筑城墙等级 | Yes | int |  |
| PlayerSWra | 包扎技能等级 | Yes | int |  |
| PlayerSAss | 突袭技能等级 | Yes | int |  |
| PlayerSEar | 地裂技能等级 | Yes | int |  |
| PlayerSTor | 旋风技能等级 | Yes | int |  |
| PlayerArm | 当前盔甲 | No | int | 物品表中该物品的ID |
| PlayerWea | 当前武器 | No | int | 物品表中该物品的ID |
| PlayerIte | 已拥有道具 | No | String | 格式：”[道具1ID];[道具2ID];……”  例如：”1;2” |
| PlayerUIte | 已解锁道具 | No | String | 格式：”[道具1ID];[道具2ID];……”  例如：”1;2;3;4” |
| PlayerGol | 金币 | Yes | int |  |
| PlayerDia | 钻石 | Yes | int |  |
| PlayerLev | 解锁关卡及评级 | Yes | String | 格式：“[关卡1ID],[评级];[关卡2ID],[评级];……”  例如：”1,A;2,S;3,B” |

### 关卡表（LEVEL）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| LevelId | 关卡ID | Yes | int | 自增长主键 |
| LevelWNum | 怪物波数 | Yes | int |  |
| LevelW1Id | 第一波ID | Yes | int |  |
| LevelGold | 获胜金币奖励 | Yes | int |  |

### 怪物波表（WAVE）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| WaveId | 怪物波ID | Yes | int | 自增长主键 |
| WaveMTyp | 该波怪物类型及出现概率 | Yes | String | 格式：”[怪物1ID],[怪物1概率];[怪物2ID],[怪物2概率];……”  例如：”1,0.5;2,0.5” |
| WaveMNum | 该波怪物数量 | Yes | int |  |
| WaveTim | 下波怪物出现间隔 | No | float | 单位秒，如果该波为本关最后一波则此项为空 |

### 怪物表（MONSTER）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| MonsterId | 怪物ID | Yes | int | 自增长主键 |
| MonsterANam | 动画资源文件名 | Yes | String | 导入Unity后在Assets里面查找 |
| MonsterTyp | 怪物类型 | Yes | int | 导成枚举 |
| MonsterAtt | 攻击力 | Yes | int |  |
| MonsterArm | 防御力 | Yes | int |  |
| MonsterHp | 生命 | Yes | int |  |
| MonsterRan | 攻击距离 | Yes | int |  |
| MonsterMp | 魔法值 | Yes | int |  |
| MonsterDRat | 闪避几率 | Yes | float |  |
| MonsterMReg | 每秒魔法值恢复 | Yes | int |  |
| MonsterASpe | 攻速 | Yes | int |  |
| MonsterSpe | 移动速度 | Yes | int |  |
| MonsterCRat | 暴击率 | Yes | float |  |
| MonsterSki | 技能ID | No | String | 格式：”[技能1ID];[技能2ID];……”  例如：”1;2” |
| MonsterANam | AI | Yes | String | 格式：”[AI1脚本名称];[AI2脚本名称];……”  例如：  ”Common;Rampage” |
| MonsterGol | 奖励金币 | Yes | int |  |

### 技能表（SKILL）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| SkillId | 技能ID | Yes | int | 自增长主键 |
| SkillCNam | 动画片段名（主角） | No | String | 导入Unity后在Assets里面查找 |
| SkillVA | 技能参数1 | No | float |  |
| SkillVB | 技能参数2 | No | float |  |
| SkillVC | 技能参数3 | No | float |  |
| SkillVD | 技能参数4 | No | float |  |
| SkillVE | 技能参数5 | No | float |  |
| SkillVF | 技能参数6 | No | float |  |
| SkillMCos | MP消耗 | Yes | int |  |
| SkillCDow | 冷却时间 | Yes | int |  |
| SkillSNam | 技能脚本名字 | Yes | String |  |

### 物品表（ITEM）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 中文说明 | 非空 | 数据类型 | 备注 |
| ItemId | 物品ID | Yes | int | 自增长主键 |
| ItemTyp | 道具类型 | Yes | int | 导成枚举 |
| ItemNam | 道具名字 | Yes | String | 当为防具时，填写不同防具主角动画文件，当为武器时，填写武器动画文件。 |
| ItemDes | 道具描述 | No | String | 道具购买界面使用 |
| ItemAtt | 攻击力 | No | int |  |
| ItemArm | 防御力 | No | int |  |
| ItemDBlo | 吸血（百分比） | No | float |  |
| ItemCRat | 暴击率（百分比） | No | float |  |
| ItemSpe | 跑速增加（百分比） | No | float |  |
| ItemASpe | 攻速增加（百分比） | No | float |  |
| ItemMReg | MP回复/秒 | No | float |  |
| ItemHp | HP增加 | No | int |  |
| ItemSId | 技能ID | No | String | 格式：”[技能1ID];[技能2ID];……”  例如：”1;2” |

## 数据结构与程序的关系

说明各个数据结构与访问这些数据结构的形式:

# 系统出错处理设计

## 出错信息

用一览表的方式说朗每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。

## 补救措施

说明故障出现后可能采取的变通措施，包括：

1. 后备技术说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术；
2. 降效技术说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录；
3. 恢复及再启动技术说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。

## 系统维护设计

说明为了系统维护的方便而在程序内部设计中作出的安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。 各个程序之间的对应关系，可采用如下的矩阵图的形式；