# 《潘神的迷宫》游戏设计文档

## 服务端

## 包构成

1. **config**：**配置类**，提供配置读取功能，目前主要用于读取数据库路径。
2. **constant**：**常量类**，提供包括武器类型、怪物类型、怪物行为类型、消息类型、消息目标对象类型等常量。
3. **util**：**应用类**，提供其他未聚集成包的一些功能类，如迷宫生成、路径搜索等功能。
4. **database**：**数据库操作类**，提供数据库操作功能，以dao和bl的方式组织。
5. **info**：**信息类**，存储怪物、玩家、武器、场景、用户等信息。
6. **message**：**消息类**，封装需要发送的信息成为消息类对象，生成json串发送；同时将从客户端接受来的json串打包成信息类对象，便于读取、使用。
7. **manager**：**管理类**，分为用户管理和游戏管理两部分，用户管理部分控制注册、登陆功能；游戏管理部分控制场景、玩家角色、武器、敌人等信息的载入、读取、更新等功能。

## 组织架构

图1 服务端组织架构

针对每一个登陆用户，都会有一个GameManager与之对应，存放在OnlineManager的容器中统一管理。GameManager管理中玩家的场景、角色、武器、怪物等信息，这些信息都是通过各个子管理器（SceneManager, PlayerManager, WeaponManager, EnemyManager）控制从数据库中读入、写出，同时与消息类进行转换。

* 处理一个客户端发送的消息流程主要是：

1. 收到客户端消息字符串。
2. 根据OnlineManager中的socket缓存找到对应的PlayerManager。
3. Dispatcher将消息字符串打包成对应的Message对象，根据消息类型和目标类型，发送给PlayerManager中对应的子Manager处理。
4. 子Manager根据消息的具体内容，决定是否生成相应数据或读写数据库。

* 向客户端发送的消息流程主要是：

1. 子Manager生成了需要发送给客户端的Message对象。
2. 将Message对象转换成json字符串，交友Dispatcher发送。

## 客户端

客户端这边的组织架构和服务端比较类似，也是消息的打包、解包、各个Manager的消息处理过程，这部分代码主要位于Assets/Script目录下。关于脚本部分对游戏过程的影响此处就不再介绍了。

## 模块构成

1. **ClientConn**：**客户端连接控制类**，主要通过线程与服务端进行socket连接和通讯，将消息置于消息队列中，等待客户端处理程序读取。通过Dispatcher将读取的消息发送给各个子Manager处理。
2. **Manager**：**管理类**，主要控制系统、玩家、迷宫、怪物、武器等消息的处理和更新服务端的相关数据。
3. **Message**：**消息类**，解析接收到的json串，封装成消息类；将客户端发送给服务端的消息打包成Message，转换成json串后发送。

## 组织架构

该部分组织架构与服务端类似，此处不再赘述。

## 关于路径搜索

* **简介**

首先此处的迷宫在客户端通过生成树算法随机生成，我们保存了迷宫各个格子的可达方向和对应的父节点格子。在迷宫中，每两个格子之间都是可达的。所以，我们通过寻找两个格子在通完树根节点的路径上的公共节点，将两条子路径合并即可生成怪物寻找玩家的路径了。

每当客户端玩家的位置发生变化，我们就通知服务器，服务器根据**当前玩家的位置**和怪物目前行走路径的**下一个节点**生成新的路径。这么做是因为一般情况下，玩家是被追逐的对象，玩家位置发生变化，也极有可能是在当前追逐怪物的路径之上的，所以我们以怪物当前路径上的下一个节点作为搜索路径的起点。如果以怪物目前的位置作为起点，那么服务器计算出来的路径发送给客户端时，考虑到网络延迟，极有可能会让怪物做一些返回之前走过位置的行为。

* **问题**

因为迷宫的道路不是很宽阔，怪物依然会有很大概率发生碰撞，甚至就在某个位置互相阻挡，无法前进。考虑目前玩过的游戏的情况，除了主动避让之外，大多数游戏的怪物是从四面八方围住主角的，碰撞概率比较小，同时也限制了怪物的数量。当然对于那种怪物极多的RPG单机游戏，（据看到的情况）似乎是在保持追逐玩家的大方向的基础上，随机向碰撞的侧方向改变位置。

此处我们的解决方案是，1）当怪物碰撞时，随机选择一个怪物（或根据id大小），将其当前路径上的下一个节点做适当的偏移处理，即让怪物将原来直接走直线的路径变成一个先向侧方向行走再向原来目标节点的侧方向行走，也就是走一个直角。当然如果怪物彼此还在碰撞里，选择将一个怪物的运动学属性打开，碰撞就消失了。2）限制某一个时刻迷宫中生成怪物的最大数量。

上述这个方案还是有一个很大的问题，就是怪物极可能发生重叠。因为时间限制，这个问题没有得到很好处理。设想是需要进行很多的位置和可达性计算，进行怪物之间的位置修正。