服务器模块

文档说明：该文档是初步的服务器模块构思，包含数据VO的设计，游戏中几大主要模块的构思，由于没做过实际的服务器开发，所以后期开发过程中可能会改动比较多，更详细的服务器逻辑设计请参考服务器模块对应的思维导图。

日期：20140110

**一：启动模块**（ServerLaunchModule）

1.1：检测是否已有服务器运行（ServerCheck）

1.2：扫描本地主机信息（主机名，IP，Mac地址）（ScanLocationMsg）

1.2.1:该信息数据结构描述为：

struct serverInfo{

String name;//主机名

String ip\_addr;//IP地址

String mac\_addr;//mac地址

}

1.2.2：暂时使用String保存，以后可能会更改数据类型

1.3：检测当前主机网络是否已经打开并加入到局域网（ServerNetCheck）

1.4：启动出错处理模块（ServerLaunchErrModule）

**二：初始化网络连接模块**（ServerInitModule）

2.1：初始化网络连接队列（NetworkQueueInit）

2.1.1：15个空的等待客户端连接的网络队列

2.1.2：队列元素的数据结构初步设定为

struct network{

socket m\_socket;//soket套接字，

string name; //客户端设定的角色名称

}

其中socket结构可能会重写，使其包含客户端的IP，mac，端口等必要信息

2.2：开放连接请求服务器端口（ConnectionPortInit）

2.2.1：该阶段仅开放连接请求端口

2.3：初始化服务器监听（ConnectionListenerInit）

2.3.1：监听连接请求

2.4：接受客户端连接请求

2.4.1：使用2.1.1中的队列接入客户端的网络连接

**三：初始化公共房间模块**（PublicRoomInit）

3.1:初始化三个阵营数组（红，蓝，等待），数组元素数据结构和网络连接结构类似

struct camp{

socket m\_socket;//soket套接字，

string name; //客户端设定的角色名称

}

3.2：开放房间活动端口

3.3：使第一个连接的玩家成为房主（从连接队列里取第一个），如果房主半途退出房间，则取队列的下一个成为房主

3.4：处理玩家阵营选择变化（CampChange）,并将变化发送给房间内的所有玩家

3.5：接受房主地图，模式，时间修改，并广播到局域网

3.6：接受房主开始游戏指令

四：**初始化游戏模块**（GameInitModule）

4.1:根据阵营人数及对应武器选择初始化对应的角色人物数据结构，角色的数据结构初定如下

class Role{

string name;//角色名

int camp;//所属阵营

int hp;//血量

int kill;//击杀数

int dead;//死亡

int assist;//助攻

int kda;//游戏结束后的kda

int curWeapon;//当前正使用的武器

int gun;//选择的枪支类型 自动步枪/手枪

int ammo;//当前枪支剩余弹药数

//下列属性供扩展所用

int knife;//选择的匕首类型

int bomb;//炸弹类型

long outputHarm;//输出伤害

long affordHarm;//承受伤害

long recoverHp;//恢复血量

int bulletCount;//使用的弹药总量

int emptyBullet;//空弹数

double nullBullet;//空弹率

...

}//其中用于标识的属性全部定义为枚举常量

4.2：初始化随机血药包和弹药包模块（RadomGoodsModule）

struct radomGoods{

int type;//类型 血药包/弹药包

int num;//数量

int location；//出现的位置

}

4.3：初始化战斗模块（FightModule）,包括伤害计算，角色状态变化等

4.4：初始化死亡模块（DeadModule），包括战绩改变，击杀提示广播等

4.5：初始化复活模块（ReliveModule）

4.6：初始化全局计时器（TimeModule）

4.7：初始化地图模块（MapModule）

**五：聊天模块**（ChatModule）

5.1：包括公共聊天和阵营聊天

5.2：使用三个队列来分别处理公共聊天，红方阵营聊天，蓝方阵营聊天 5.3：该模块负责所有的聊天接收与分发

**六：战绩统计模块**（ResultModule）

6.1：该模块负责所有战绩统计，包括游戏中和游戏结束后

6.2：每次战绩变化时，都重新统计战绩，并分发给所有客户端

**七：全局网络连接模块**（ConnectionModule）

7.1：服务器的所有数据由此为出口和入口

7.2：该模块管理所有网络连接的最底层的连接和断开

**八：全局错误模块**（ServerErr）

8.1：定义和管理服务器出现的错误

**九：全局Log模块**（ServerLogModule）

9.1:负责服务器的log打印

**十：其他**

10.1：开发前期不会过多考虑服务器性能和优化问题，以实现功能为重

10.2：整个游戏完成所有功能后才会考虑优化问题

10.3：开发过程中，可能会把一些服务器的需求写到客户端上，视实际开发为准