# 代码分析

## 从socket连接到登录

SocketManger 用于管理多个SocketConnect对象

SocketConnect 用于连接一个socket, 发送消息给socket, 接收来自socket的消息

ProtocolHelper 用于封装协议中的协议发送数据,及解析接收到的协议数据 (但是这里的功能并不统一,目前只有登录是放在这里的,其他的模块在各自的模块中处理了.)

1. 连接到服务端:

从 调用 SocketManger.Connect 方法开始, 调用 SocketConnect.Connect 方法, 调用 (SocketConnect.clientSocket 字段的) Socket.BeginConnect 方法.

1. 连接成功后

完成1的连接后,会回调 SocketConnect.ConnectCallBack 方法, 这里会启动线程(运行ReceiveData方法)并接受来自服务端的数据.

1. 数据接收

首先接收到的是token数据,并调用ShakeHands 进行处理. 然后

1. 发送处理后的 token 协议到服务端.(并设置状态为Connected)
2. 发送登录协议到服务端 (不是选中的SocketConnect 不发送登录协议)
3. 启动心跳

到这里为止连接成功,并已经发送了登录协议.

4. 登录协议发送过程

从3-2中发送的协议从调用 ProtocolHelper.Send\_login 开始.判断 isGame 分别调用 send\_Game\_login(游戏登录) 和 send\_socketlobby\_login(大厅登录).

5. 大厅登录(游戏登录与大厅登录类似,具体仅有协议内容不同,这里就不列了)

大厅登录分3中登录情况:

1. OnSocketConnectTriggerAction 这种是一次性登录,登录后立刻置空,用于特殊情况,例如手机验证码发送,微信验证码发送等(因为后台在登录失败后会断开连接,需要重新连接)
2. 回调lua里的登录内容(这个不用了,改成全部lua的时候,就不需要了)
3. 使用具体的登录方式登录,这种是登录游戏后能够重连并自动发送的登录, 因为游戏有重连功能,所有在重连后必须要自动发送登录协议. 具体的登录方式在点击按钮的时候使用 LoginType 记录下来,并在连接到socket的时候,对类型进行判断发送不同的协议.

socket连接到登录 流程分析 完.