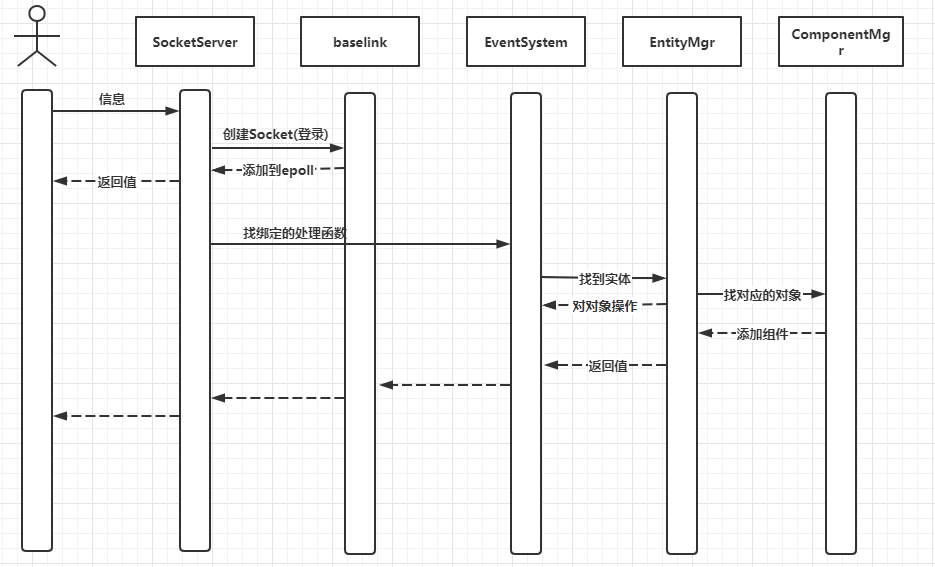
**服务器精英训练营**

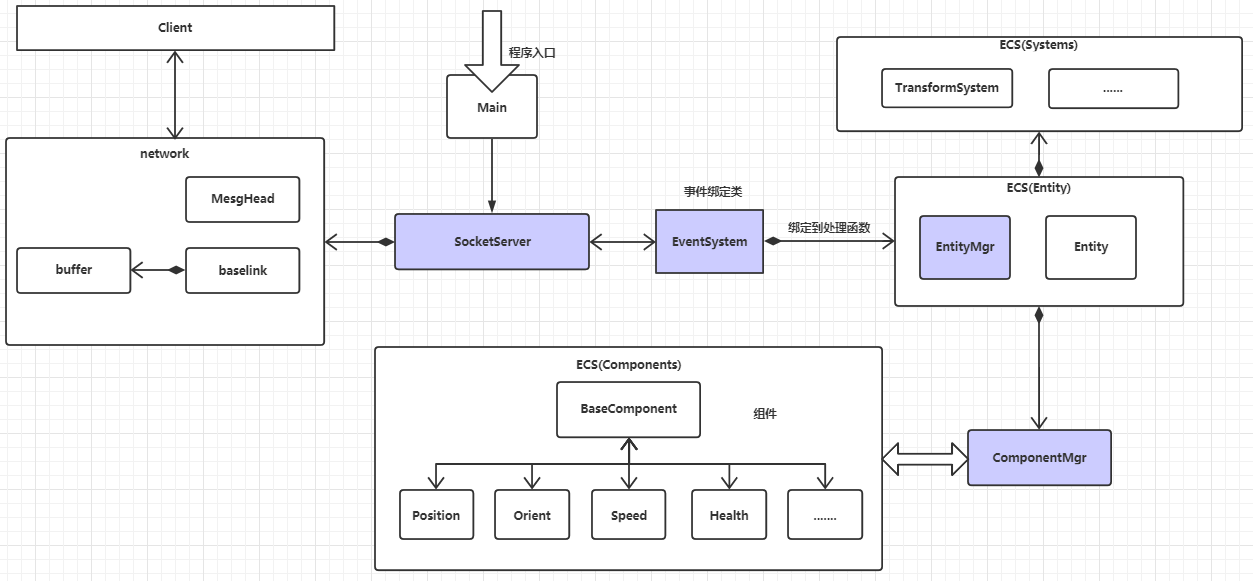
##### 第三周作业：

**1、服务器端：**

时序图：



类图：（紫色=单例类）



**设计思路：**

这周主要事件都用来写后端了，主要主要是ECS，在搞懂这个框架上花费了挺多时间的，但是效果确实不错，在后面写代码的时候感受到了这个框架的拓展性很高，各个模块之间耦合很少，同时也把这个框架推广到了组内成员，感觉大家用的效果还不错。用EventSystem来绑定处理函数。

**Server**

main.cpp: 函数入口，单例类初始化

SocketServer.cpp: 服务器主要流程，目前epoll没有封装，直接在SocketServer中， 所以会有很多初始化epoll的代码

Player.cpp: 用户数据类，主要是用户所有的基本操作和数据，在客户端将用户数据&方法分离后感觉效果并不好，所以这边暂时数据和方法是耦合的，接下来同样可以使用ECS框架来改进

EventSystem.cpp: 事件处理类，目前想到的还是对于消息的处理，但未来可能会加入其他事件的管理。用模板类(MsgHandler.h)实现了 MsgID&处理函数的绑定，对于一类消息创建一个map来绑定事件与方法（一个msgID可以看作一个事件）。同时由于EventSystem.cpp是一个单例管理类，相当于一个全局的事件管理类，未来所有的事件可能主要都是与这个类有关。目前的实现形式是将MsgHandler返回（由于模板类的原因，没办法返回对于的函数指针）。未来需要改进的是将对于Msg的处理也放到EventSystem中，而不是在SocketServer中。（个人认为这种管理方式比demo中将事件处理类放在server/client中作为成员变量来的好）

EntityMgr.cpp：实体管理单例类，管理所有的Entity，包括创建与多个实体的逻辑处理

**Network:**

baselink.cpp: 对于socket的一些封装，和socket有关的基本都在里面。与demo中的TCP类相对应

buffer.cpp: 对于buffer的封装，包括头/尾指针的管理与缓冲区的管理，目前没写环形缓冲区

MesgHead.cpp: 包头类，对于包头的处理(encode&decode)，包头信息在MesgInfo中对应（现在是定死的 msgID、uID、packLen）

**Components:**

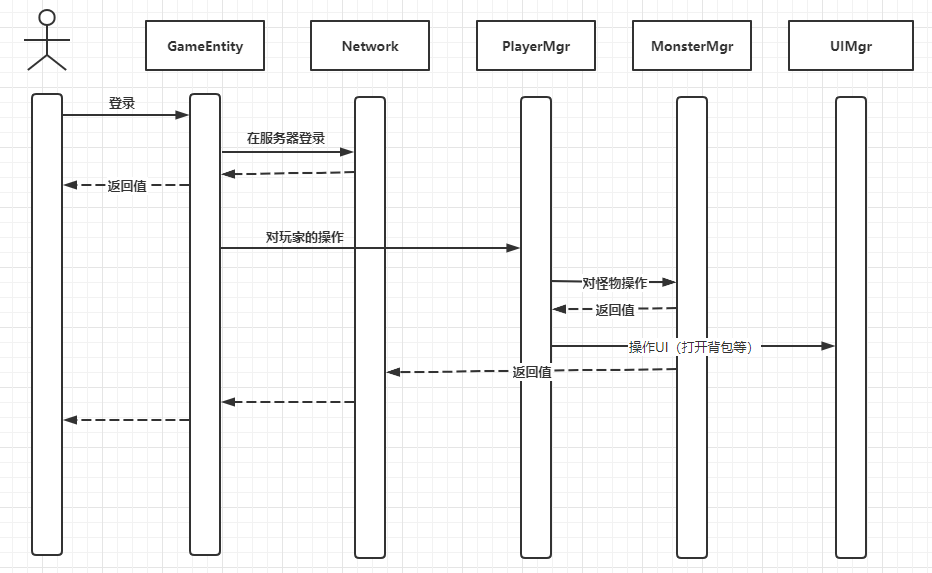
ComponentMgr.cpp：管理所有的组件(组件类而不是实例)，方便实例化组件

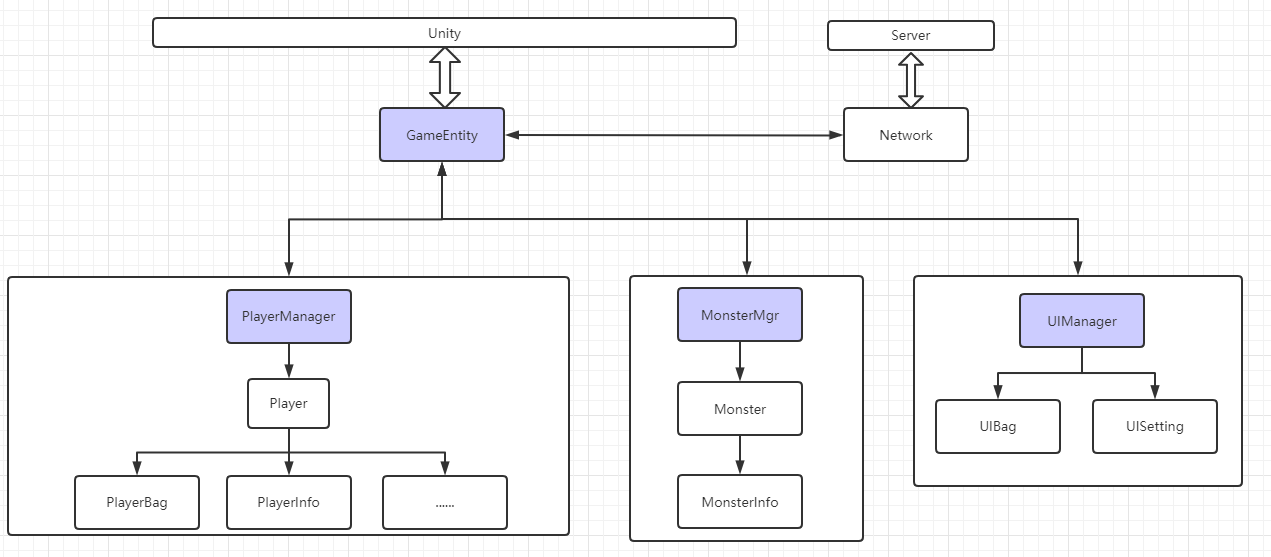
TransoformComponent.cpp：和移动相关的几个组件

**System:**

TransoformSystem.cpp: transform的逻辑System

**2、客户端：**





**设计思路：**

客户端主要逻辑就和上周差不太多，主要是对几个单例类封装，用来处理逻辑。GameEntity是对unity的唯一接口，用来控制所有对象的生命周期和流程，用三个单例类来控制对应的对象，这样拓展性高一点。后面的工作主要是同步，使用服务器传来的数据刷新玩家数据，客户端不断同步服务器的数据来渲染。

预测是保留一个玩家前几帧的数据，看前几帧玩家的移动，来做加权平均来预测玩家对象的数据来渲染。