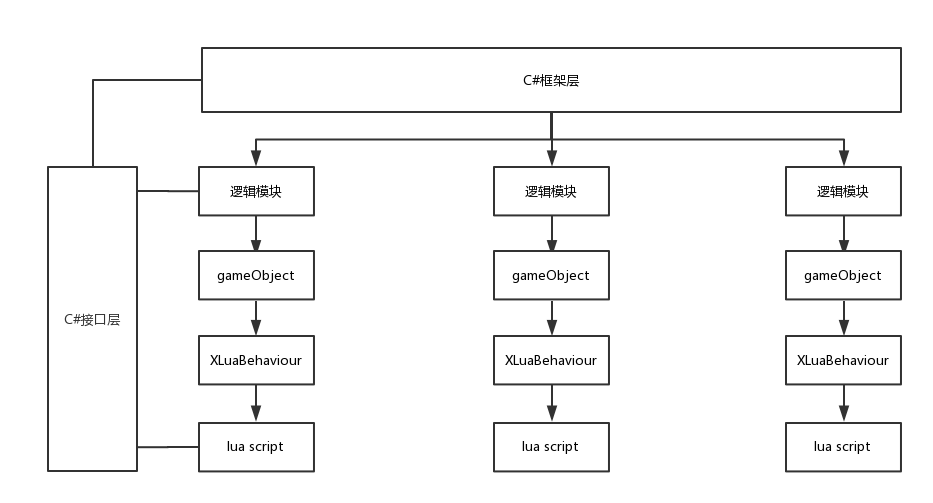
**Unity+Lua交互结构**



**XluaComponent**

Lua框架管理类

**XluaLooper**

把Lua主循环分发到lua层。所有GameObject Update周期都在lua层的Update里处理。

唯一性，绑定到EIFramework上。

**XluaBehaviour**

模拟MonoBehaviour,传递到lua层。

一个GameObject对应一个XluaBehaviour。

一个XluaBehaviour对应一份lua脚本，加载后返回LuaTable引用。

？？：

1、lua层里，每个LuaTable默认为局部变量？互不相通，需要通过XluaBehaviour接口进行交互。

2、lua层里，每个LuaTable定义为全局变量？互相感知，也可通过XluaBehaviour接口进行交互。

**接口层**

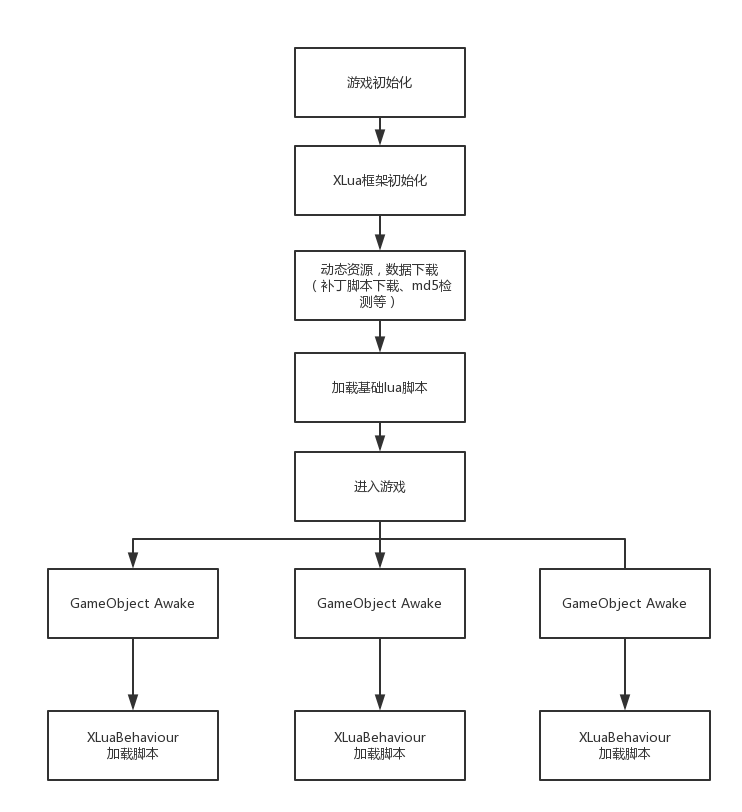
提供数据、协议、网络通信等接口给lua层。

使用XLua ，UnityEngine、System等方法，默认能够让lua直接调用， 那是否需要对lua层进行限制？只提供有限接口，保证lua层作用的边界清晰？

**逻辑模块**

EINode?

**脚本加载**



**脚本的加载和热更**

脚本资源来源：

1、persistentDataPath?

2、AB打包？

加载流程:

游戏开始时，预先加载部分脚本（公有方法、配置等）。到了游戏中， 再进行GameObject上的脚本加载，开始对应的逻辑执行。

**延伸下来会有两种方向：**

1. C#层为主导？

初始化、对象管理、功能模块管理等游戏流程，在C#层实现。具体到功能模块里，再把内容映射到lua，lua层进行对应的业务逻辑编写。

1. lua层为主导？

C#层只提供框架层和接口层，初始化、对象管理、功能模块管理等游戏流程全部写在lua层。