#### 一、xLua的安装

1、新建任意的Unity工程，将xLua.zip解压。

2、将xLua/Assets中的所有内容，放入Unity工程的Assets文件夹中。

3、将xLua/Tools文件夹，拷贝到Unity工程Assets文件夹旁边。（即Tools和Assets在同一级目录）

xLua安装完毕。

xLua主要包含两个文件夹，其中Plugins是动态链接库，由于Lua本身是用C语言写的，所以需要对各种常用平台都编译一个不同的so或dll动态链接库。可以在Plugins文件夹中看到大量常用平台。

另一个XLua文件夹，里面是Xlua本身的代码，还有文档和范例，我们主要是通过Examples来对Xlua进行学习。

#### 二、测试和熟悉xLua

xLua自带丰富的案例，位于XLua/Examples文件夹中，按顺序依次打开示例场景进行学习即可。具体听老师的讲解。

这些案例涵盖了大量C#的高级用法，后续范例把事件回调、async异步、泛型等用法导入到Lua中协同工作。这些案例更多是为了演示插件语法的完整性，在实际游戏开发中并不常见。

注意，在学习时要重点关注基础的案例，把基础案例学明白、搞透彻，才能理解后续热更新的步骤。

#### 范例说明

|  |  |
| --- | --- |
| 01\_Helloworld | 最简单的Lua示例，参考意义不大 |
| 02\_U3DScripting | 最重要的范例，必须掌握。参考价值很大，但实用时需要进一步修改。 |
| 03\_UIEvent | 与上一个范例类似，说明在XLua中，Lua函数可以直接用作C#委托参数，即直接作为UI按钮等事件的回调函数使用。 |
| 04\_LuaObjectOrented | 复杂案例，涉及到C#接口、元表操作。要求尽量看懂，不要求掌握 |
| 05\_NoGc | 使用特殊语法，在Lua和C#之间传递值类型时不产生GC，不要求掌握 |
| 06\_Coroutine | Lua协程与Unity协程混写。Lua协程在本课程中属于超纲内容。 |
| 08\_Hotfix | “热修补”是XLua的一大特色功能，老师会科普解释这一功能的设计思路和使用方式。 |

#### Addressable资源管理与热更新

Addressable的内容请看另一个文档《Addressable资源管理与热更新》。在相关范例的基础上，我们再结合XLua来进一步实践。

#### 用Lua改写《飞龙射击》的主角脚本

请参考Dragon\_HotUpdate热更新完整工程。

重点内容：

LuaScripts文件夹下

|  |  |
| --- | --- |
| LuaScript.cs | C#脚本与Lua脚本的桥梁 |
| PlayerCha.lua.txt | 用Lua改写的角色脚本 |
| PlayerController.lua.txt | 用Lua改写的输入控制脚本 |
| GameSys.cs | 1. 管理所有的Lua脚本源码，Lua用脚本名称标识 2. 管理编译后的Lua对象 |
| TempLoadLua.cs | 临时脚本，读取Lua的TextAsset源码，保存到GameSys里 |

经过这一步骤，实现了通过Lua脚本控制主角逻辑。

为了实现Lua源代码管理，以及Lua脚本之间的互相访问，实现了GameSys。

#### 热更新Lua脚本

主要思路就是在开始场景中，把Lua脚本作为资源加载，并保存到GameSys里。

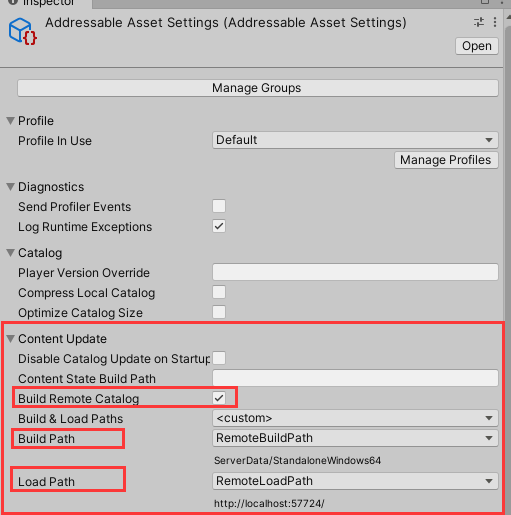
参考Scripts\Start\StartScene.cs

|  |
| --- |
| var luas = Addressables.LoadAssetsAsync<TextAsset>(  new List<string> { "PlayerCha.lua", "PlayerController.lua" }  , null, Addressables.MergeMode.Union);  yield return luas;  // 等待加载完毕  foreach (var textAsset in luas.Result)  {  Debug.Log("加载Lua成功 " + textAsset.name);  GameSys.Instance.AddLuaSource(textAsset.name, textAsset.text);  } |

之后记得勾选Lua文件的Addressables标签，加入到热更新资源列表里。

#### Addressables的资源升级

必须要为热更新资源建立Catalog（目录）才支持资源升级，修改Addressable的System Settings里的Content Update设置，见下图：



修改后，整个发布流程必须重新走一遍。包括重新打资源包，重新发布游戏的执行文件（需全部删除原来的游戏文件再打包，不能覆盖）。

成功开启Catalog功能以后，就可以尝试修改脚本，发布补丁了。

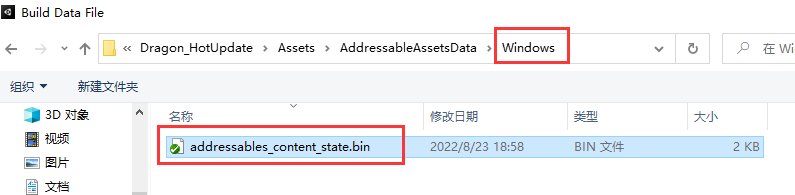
方法为：

1、先修改Lua脚本，改动一些明显的参数。

2、在打包按钮处选择Update a Previous Build：



然后找到打包的bin文件（里面记录了当前资源包的状态。以此出发，到下一个补丁的状态）



3、查看Log，如果打包成功，在ServerData里会多出很多资源文件。里面包含了原来的资源包和补丁资源。

4、再次运行游戏，就会看到Lua已经被更新到最新版了。