

Xlua 下载地址: <https://github.com/tencent/xlua>

下载Assets目录即可 也可以全部下载

Branch: master ▾	New pull request	Find file	Clone or download ▾
rebelmx4 and chexiongsheng genNullableCaster 最后一个参数是nil, 其他参数全是nil (fix #527)		Latest commit 04e38ac 5 days ago	
Assets	genNullableCaster 最后一个参数是nil, 其他参数全是nil (fix #527)	5 days ago	
General	1、生成代码工具改进	2 months ago	
Test	调整用例通用性	a year ago	
Tools	hotfix注入支持忽略编译器自动生成代码以及不生成base代理的选项	6 months ago	
WebGLPlugins	webgl xlua.c update	3 months ago	
build	add switch support (#507)	a month ago	
docs	update doc dependence	4 months ago	
.gitignore	1、封装getglobal, setglobal的lua异常;	2 years ago	
.travis.yml	xbuild->msbuild, remove xlua.dll refer	5 months ago	
LICENSE.TXT	更新些信息	a year ago	
README.md	release 图标改为2.1.13	2 months ago	
README_EN.md	release 图标改为2.1.13	2 months ago	

Xlua的映射

- 1 映射到普通的类或者结构体
- 2 映射到结构 interface
- 3 映射到集合 list dict
- 4 映射到LuaTable LuaFunction
- 5 映射到delegate Func Action

实现lua和C#之前的互相调用

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using XLua;
5
6 public class config
7 {
8     public string name;
9     public int age;
10    public int qq;
```

```
11 }
12
13 public class XluaText : MonoBehaviour {
14
15     //lua加载类
16     LuaEnv env = new LuaEnv();
17
18     void OnGUI() {
19         if (GUILayout.Button("C#调用Lua 执行Lua的语法"))
20         {
21             CSharp_DoLua();
22         }
23         if (GUILayout.Button("C#调用Lua File"))
24         {
25             CSharp_DoFile();
26         }
27         if (GUILayout.Button("Lua调用C#"))
28         {
29             LuaCallCSharp();
30         }
31     }
32
33     void CSharp_DoLua()
34     {
35         env.DoString("print 'helloworld'");
36     }
37
38     void CSharp_DoFile()
39     {
40         //调用test0模块
41         env.DoString("require 'test0' ");
42
43         //通过LuaEnv对象 获取全局变量
44         Debug.Log(env.Global.Get<string>("Name"));
45         Debug.Log(env.Global.Get<int>("Age"));
46         Debug.Log(env.Global.Get<bool>("IsBoy"));
47
48         //获取Lua脚本中的表和数据
49         LuaTable table = env.Global.Get<LuaTable>("config");
50         Debug.Log(table.Get<string>("name"));
51         Debug.Log(table.Get<int>("age"));
52         Debug.Log(table.Get<int>("qq"));
```

```

53
54     //映射到自定义的类中
55     config c=env.Global.Get<config>("config");
56     Debug.Log(c.name);
57     Debug.Log(c.age);
58     Debug.Log(c.qq);
59
60     //调用Lua的函数并接受返回值
61     LuaFunction function=env.Global.Get<LuaFunction>("printInfo");
62     object[] datas=function.Call(50);
63     int x = System.Convert.ToInt32(datas[0]);
64     Debug.Log("Lua脚本中方法的返回值是" + x);
65 }
66
67 void LuaCallCSharp()
68 {
69     //调用test.lua.txt 执行
70     env.DoString("require 'test1'");
71 }
72 }
73

```

Hero类型

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Hero {
6      private int hp;
7      public int HP
8      {
9          get { return hp; }
10         set { hp = value; }
11     }
12
13     private string name;
14     public string Name
15     {
16         get { return name; }

```

```

17         set { name = value; }
18     }
19
20     public Hero(int hp,string name)
21     {
22         this.hp = hp;
23         this.name = name;
24     }
25
26     public void PrintInfo()
27     {
28         Debug.Log(name + "血量是:" + hp);
29     }
30
31     public int AddHP(int _hp)
32     {
33         Debug.Log("Lua脚本传递的参数是" + _hp);
34         this.hp += _hp;
35         return this.hp;
36     }
37 }

```

test0.lua.txt

```

1  --创建一张表
2  config={name="张无忌",age=30,qq="110"}
3
4  --创建3个全局变量
5  Name="赵敏"
6  Age=20
7  IsBoy=false
8
9  function printInfo(a)
10     print("C#传递的参数值是"..a)
11     return 1000
12 end
13
14 print("test0.lua执行完毕")
15

```



test1.lua.txt

```
1 local hero=CS.Hero(200,"张三") --调用CS脚本中的类的构造方法
2 hero:PrintInfo() --打印CS 对象中的数据
3
4 print("C#里hero的血量被增加到",hero:AddHP(20)) --调用带参数的有返回值的CS
  方法
5
6
7 local obj=CS.UnityEngine.GameObject("luaobj")
8 obj.transform.position=CS.UnityEngine.Vector3(100,100,100)
```

默认把所有lua文件存放在Resources文件夹下

文件格式修改为.txt

设置UTF8编码格式

(D:) > soft > UnityClass > XLua > Assets > Resources				
新建文件夹				
名称	修改日期	类型	大小	
 test0.lua.txt	2019/1/22 17:41	文本文档	1 KB	
 test0.lua.txt.meta	2019/1/23 9:51	META 文件	1 KB	
 test1.lua.txt	2019/1/23 10:19	文本文档	1 KB	
 test1.lua.txt.meta	2019/1/23 10:14	META 文件	1 KB	

Lua实现热更新流程

添加一个CS文件 负责加载Lua文件中的变量和函数 存储

并在Unity生命周期函数中对应调用

C#脚本

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using XLua;
```

```

5  using System;
6
7  [CSharpCallLua]
8  public class LuaCallBack : MonoBehaviour {
9
10     LuaEnv env = new LuaEnv();
11     //test2.lua.txt
12     public TextAsset textAsset;
13
14     private Action luaAwake;
15     private Action luaStart;
16     private Action luaUpdate;
17     private Action luaDestory;
18     private Action luaOnGUI;
19
20     private LuaTable table;
21     public GameObject other;
22     public AnimationClip clip;
23
24     void Awake()
25     {
26
27         //创建表 用于接受Lua中的数据 变量和函数
28         table = env.NewTable();
29         LuaTable meta = env.NewTable();
30         meta.Set("__index", env.Global);
31         table.SetMetaTable(meta);
32         meta.Dispose();
33
34         env.DoString(textAsset.text, "textLua", table);
35
36         table.Set("self", this); //设置Lua场景中Self关键字为this
37         table.Set("object1", other); //添加场景引用 lua中使用object1来表
示other游戏对象
38         table.Set("clip", clip);
39
40         //Get对应生命周期的方法
41         table.Get("awake", out luaAwake);
42         table.Get("start", out luaStart);
43         table.Get("update", out luaUpdate);
44         table.Get("ondestory", out luaDestory);
45         table.Get("ongui", out luaOnGUI);

```

```

46
47     if (luaAwake != null)
48         luaAwake();
49
50
51 }
52
53 void Start () {
54     if (luaStart != null)
55         luaStart();
56 }
57
58 // Update is called once per frame
59 void Update () {
60     if (luaUpdate != null)
61         luaUpdate();
62 }
63
64 void OnGUI()
65 {
66     if (luaOnGUI != null)
67         luaOnGUI();
68 }
69 }
70

```

lua脚本

```

1  local speed=2
2  local collider=nil
3  local anim=nil
4
5  function start()
6      --调用Debug语句 打印游戏物体名字
7      CS.UnityEngine.Debug.Log(self.name)
8
9      collider=self.gameObject:GetComponent(typeof(CS.UnityEngine.Collider))
10     collider.enabled=false
11
12     --获取Animation组件 添加新动画剪辑 设置播放模式并且播放

```

```

12      --
13      anim=self.gameObject:GetComponent(typeof(CS.UnityEngine.Animation))
14      --anim:AddClip(clip,"GG")
15      --clip.wrapMode=CS.UnityEngine.WrapMode.Loop
16      --anim:Play("GG")
17
18      --动态加载实例化
19      for i=0,4 do
20          prefab=CS.UnityEngine.Resources.Load("abc")
21          CS.UnityEngine.Object.Instantiate(prefab)
22      end
23
24      --给其他引用的物体添加新组件
25      object1:AddComponent(typeof(CS.UnityEngine.Animation))
26
27      end
28
29      function update()
30          local r=CS.UnityEngine.Vector3.up*speed
31          self.transform:Rotate(r)
32      end
33
34      function ondestroy()
35      end
36
37      function ongui()
38          --给UI添加按钮
39          if CS.UnityEngine.GUILayout.Button("按钮") then
40              end
41          end
42      end

```

实现人物的移动和跳跃 发射线 UI按钮逻辑判断

实现摄像机的平滑跟随

人物脚本

```

1  local speed=0.1
2  local h=0
3  local v=0
4  local dir=CS.UnityEngine.Vector3.zero

```



```
5 local rigid
6 local transform
7 local color
8
9 function start()
10
11     rigid=self.gameObject:AddComponent(typeof(CS.UnityEngine.Rigidbody))
12     transform=self.transform
13     color=CS.UnityEngine.Color.blue
14 end
15
16 function update()
17     h=CS.UnityEngine.Input.GetAxis("Horizontal")
18     v=CS.UnityEngine.Input.GetAxis("Vertical")
19     dir=CS.UnityEngine.Vector3(h,0,v)
20     self.transform:Translate(dir*speed)
21     Jump()
22     drawLine()
23 end
24
25 function ongui()
26     if CS.UnityEngine.GUILayout.Button("颜色") then
27
28         render=self.gameObject:GetComponent(typeof(CS.UnityEngine.Renderer))
29         render.material.color=color
30     end
31 end
32
33 function drawLine()
34
35     CS.UnityEngine.Physics.Raycast(transform.position,transform.forward,10
36 )
37
38     CS.UnityEngine.Debug.DrawLine(transform.position,transform.forward*10)
39 end
40
41 function Jump()
42     if CS.UnityEngine.Input.GetMouseButtonDown(1)then
43         rigid:AddForce(CS.UnityEngine.Vector3.up*300)
44     end
45 end
```

摄像机脚本

```
1 local dir
2 local distance
3 local target
4 local pos
5
6 function start()
7     target=CS.UnityEngine.GameObject.Find("Cube").transform
8     dir=self.transform.position-target.position
9 end
10
11 function update()
12     pos=target.position+dir
13
14     self.transform.position=CS.UnityEngine.Vector3.Lerp(self.transform.position,pos,
15         CS.UnityEngine.Time.deltaTime)
16 end
```

实现C#调用Lua脚本的父类

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using XLua;
5 using System;
6
7
8 [CSharpCallLua]
9 public class CCallLuaBase : MonoBehaviour {
10     //Lua脚本文件
11     public TextAsset textAsset;
12     //子类需要往luaTable里设置的数据
13     public Dictionary<string, object> datas = new Dictionary<string,
14         object>();
15
16     //接受Lua脚本中不同方法的委托
```

```

16     protected Action luaAwake;
17     protected Action LuaStart;
18     protected Action LuaUpdate;
19     protected Action LuaDestroy;
20     protected Action LuaOnGUI;
21
22     //Xlua框架中加载和存储Lua数据的类型对象
23     private LuaEnv env = new LuaEnv();
24     private LuaTable table;
25
26     public virtual void SetLuaData() { }
27
28     void Awake()
29     {
30         table = env.NewTable();
31         LuaTable meta = env.NewTable();
32         meta.Set("__index", env.Global);
33         table.SetMetaTable(meta);
34         meta.Dispose();
35
36         env.DoString(textAsset.text, this.GetType().Name, table);
37
38         SetLuaData();
39         table.Set("self", this);
40         foreach (var it in datas)
41         {
42             table.Set(it.Key, it.Value);
43         }
44
45         table.Get("awake", out luaAwake);
46         table.Get("start", out LuaStart);
47         table.Get("update", out LuaUpdate);
48         table.Get("ongui", out LuaOnGUI);
49         table.Get("ondestroy", out LuaDestroy);
50
51         if (luaAwake != null)
52             luaAwake();
53     }
54
55     void Start () {
56         if (LuaStart != null)
57             LuaStart();

```

```

58     }
59
60     void Update () {
61         if (LuaUpdate != null)
62             LuaUpdate();
63     }
64
65     void OnGUI()
66     {
67         if (LuaOnGUI != null)
68             LuaOnGUI();
69     }
70
71     void OnDestroy()
72     {
73         if (LuaDestroy != null)
74             LuaDestroy();
75     }
76 }

```

子类继承

```

1  public class LuaCallBack : CCallLuaBase {
2
3      public GameObject other;
4
5      public override void SetLuaData()
6      {
7          datas.Add("self", this);
8          datas.Add("obj1", other);
9      }
10 }

```