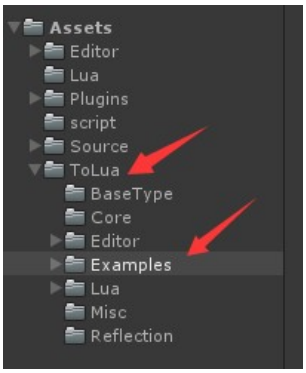
学习tolua#·20多个例子

初始项目搭建

- clone官方库
- 新建unity工程
- 依次把官方库里的Assets和Unity5.x/Assets拷贝到项目Assets里

打开unity工程， 开始逐个学习例子，例子目录:



1. hello world

- 新建luaState
- 执行字符串命令print ('hello tolua#')

```
    void Awake()
    {
        LuaState lua = new LuaState();
        lua.Start();
        string hello =
            @"
                print('hello tolua#')
            ";

        lua.DoString(hello, "HelloWorld.cs");
        lua.CheckTop();
        lua.Dispose();
        lua = null;
    }
```

unity的console里找到输出结果（找的不是很容易呀）

2. run scripts from file

- 添加源码建search path
- 读取执行lua源文件

脚本内容

ScriptsFromFile.lua

```
print("This is a script from a utf8 file")
print("tolua: 你好!")
```

添加脚本搜索路径到luaState里。

```
lua = new LuaState();
lua.Start();
//如果移动了ToLua目录，自己手动修复吧，只是例子就不做配置了
string fullPath = Application.dataPath + "\\ToLua/Examples/02_ScriptsFromFile";
lua.AddSearchPath(fullPath);
```

执行方法1，dofile：执行脚本内容

```
lua.DoFile("ScriptsFromFile.lua");
```

执行方法2，require：执行**一次**脚本内容

```
lua.Require("ScriptsFromFile");
```

输出结果

```
[ScriptsFromFile.lua:1]:This is a script from a utf8 file
[ScriptsFromFile.lua:2]:tolua: 你好!
```

3. call lua function

- lua加载执行代码，定义函数
- c#调用lua定义的函数
- 释放c#保存的lua函数

定义函数test.luaFunc

```
private string script =
    @"  function luaFunc(num)
            return num + 1
        end

        test = {}
        test.luaFunc = luaFunc
    ";
```

```
lua = new LuaState();
lua.Start();
lua.DoString(script);
```

C#获取lua的function

```
//Get the function object
func = lua.GetFunction("test.luaFunc");
```

执行方式1：

```
//有gc alloc
object[] r = func.Call(123456);
Debugger.Log("generic call return: {0}", r[0]);
```

源码中说要少用，Call实现

```
//慎用
public object[] Call(params object[] args)
{
    BeginPCall();
    int count = args == null ? 0 : args.Length;

    if (!luaState.LuaCheckStack(count + 6))
    {
        EndPCall();
        throw new LuaException("stack overflow");
    }

    PushArgs(args);
    PCall();
    object[] objs = luaState.CheckObjects(oldTop);
    EndPCall();
    return objs;
}
```

执行方式2：和1差不多，对返回值特殊处理了

```
        func.BeginPCall();
        func.Push(123456);
        func.PCall();
        int num = (int)func.CheckNumber();
        func.EndPCall();
        return num;
```

释放：

```
    func.Dispose();
    func = null;
```

4. c# access lua variable

- c# 用 luaState来存取lua全局对象

```
    varTable = {1,2,3,4,5}
    varTable.default = 1
```

```
lua["Objs2Spawn"] = 5;
lua.DoString(script);

//通过LuaState访问
Debugger.Log("Read var from lua: {0}", lua["var2read"]);
Debugger.Log("Read table var from lua: {0}", lua["varTable.default"]);
```

5. lua coroutine

- tolua #改写了lua里的协程

6. lua coroutine

- tolua # 是还实现了一些函数：Yield、WaitForEndOfFrame等

7. LuaThread

- tolua # 把lua协程包装成LuaThread，方便c#控制协程的执行。

8. lua access c# array

- lua访问c#的array

c#里构造一个array，通过函数传参的方式传给lua

```
int[] array = { 1, 2, 3, 4, 5};
func = lua.GetFunction("TestArray");

func.BeginPCall();
func.Push(array);
func.PCall();
```

lua里使用接口访问c#的array

```
function TestArray(array)
    local len = array.Length

    for i = 0, len - 1 do
        print('Array: '..tostring(array[i]))
    end

    local t = array:ToTable()

    for i = 1, #t do
        print('table: '.. tostring(t[i]))
    end

    local iter = array:GetEnumerator()

    while iter:MoveNext() do
        print('iter: '..iter.Current)
    end

    local pos = array:BinarySearch(3)
    print('array BinarySearch: pos: '..pos..' value: '..array[pos])

    pos = array:IndexOf(4)
    print('array indexof bbb pos is: '..pos)

    return 1, '123', true
end
```

9. lua access c# Dictionary

- lua读写c#的Dictinary

也是tolua#导出了一些接口，具体看例子就是了。
这个例子里有放置一个tolua#导出自定义c#类接口的样例，可以看看。

10. lua access c# Enum

- lua访问c#的Enum，Enum在lua里可以获得字符串名字，也可以转成整数。

11. lua access c# delegate

- lua访问c#的delegate和event

12. lua access unity GameObject

- lua访问UnityEngine.GameObject

例子是在场景中不停的加白色粒子

代码很简单，生成一个gameobject，添加个粒子组件。

```
lua.DoString(@"
    local go = UnityEngine.GameObject('go', typeof(UnityEngine.Camera))
    go:AddComponent(typeof(UnityEngine.ParticleSystem))
    ", "TestGameObject.cs");
```

tolua#生成的LuaBinder.cs导出了大量的接口

```
    {
        lua = new LuaState();
        lua.LogGC = true;
        lua.Start();
        LuaBinder.Bind(lua);
        lua.DoString(script, "TestGameObject.cs");
```

```
public static void Bind(LuaState L)
{
    float t = Time.realtimeSinceStartup;
    L.BeginModule(null);
    LuaInterface_DebuggerWrap.Register(L);
    L.BeginModule("UnityEngine");
    UnityEngine_ComponentWrap.Register(L);
    UnityEngine_TransformWrap.Register(L);
    UnityEngine_MaterialWrap.Register(L);
    UnityEngine_LightWrap.Register(L);
```

```
public static void Register(LuaState L)
{
    L.BeginClass(typeof(UnityEngine.Component), typeof(UnityEngine.Object));
    L.RegFunction("GetComponent", GetComponent);
    L.RegFunction("GetComponentInChildren", GetComponentInChildren);
    L.RegFunction("GetComponentsInChildren", GetComponentsInChildren);
    L.RegFunction("GetComponentInParent", GetComponentInParent);
    L.RegFunction("GetComponentsInParent", GetComponentsInParent);
    L.RegFunction("GetComponents", GetComponents);
    L.RegFunction("CompareTag", CompareTag);
    L.RegFunction("SendMessageUpwards", SendMessageUpwards);
    L.RegFunction("SendMessage", SendMessage);
    L.RegFunction("BroadcastMessage", BroadcastMessage);
    L.RegFunction("New", _CreateUnityEngine_Component);
    L.RegFunction("__eq", op_Equality);
    L.RegFunction("__tostring", ToLua.op_ToString);
    L.RegVar("transform", get_transform, null);
    L.RegVar("gameObject", get_gameObject, null);
    L.RegVar("tag", get_tag, set_tag);
    L.EndClass();
}
```

13. 演示luaClient的使用

luaClient继承MonoBehaviour。

14. lua 针对c#的out 类型参数处理

- out类型参数，转换成返回值，lua支持多个返回值。

c#里代码

```
Camera camera = Camera.main;
Ray ray = camera.ScreenPointToRay(Input.mousePosition);
RaycastHit hit;
bool flag = Physics.Raycast(ray, out hit, 5000, 1 << LayerMask.NameToLayer("Default"));
```

lua里代码

```
local box = UnityEngine.BoxCollider

function TestPick(ray)
    local _layer = 2 ^ LayerMask.NameToLayer('Default')
    local flag, hit = UnityEngine.Physics.Raycast(ray, nil, 5000, _layer)
    -- local flag, hit = UnityEngine.Physics.Raycast(ray, RaycastHit.out, 5000, _layer)

    if flag then
        print('pick from lua, point: '..tostring(hit.point))
    end
end
```

这一句去掉，日志里有错误信息与preload有关

15. 演示protobuf的使用

- c#导出，c#自己使用
- protoc-gen-lua生成，给lua使用。

```
protected virtual void OpenLibs()
{
    luaState.OpenLibs(LuaDLL.luaopen_pb);
    luaState.OpenLibs(LuaDLL.luaopen_struct);
    luaState.OpenLibs(LuaDLL.luaopen_lpeg);
```

16. 延时int64使用

- tolua#提供了个int64的扩展库，把int64分成两个int32了。

```
local str = tostring(int64.new(3605690779, 30459971))
local n2 = int64.new(str)
local l, h = int64.tonum2(n2)
print(str..':'..tostring(n2)..' low:'..l..' high:'..h)
```

17. tolua模拟继承

例子里使用tolua.setpeer扩展包装transform。能够提升性能。

```
function LuaTransform.Extend(u)
    local t = {}
    local _position = u.position
    tolua.setpeer(u, t)

    t.__index = t
    local get = tolua.initget(t)
    local set = tolua.initset(t)

    local _base = u.base

    --重写同名属性获取
    get.position = function(self)
        return _position
    end
```

```
local transform = LuaTransform.Extend(node)
```

这样获取transform的position时，就不用每次都从tramsform里查找position属性了。

```
for i = 1, 200000 do
    transform.position = transform.position
end
```

```
for (int i = 0; i < 200000; i++)
{
    Vector3 v = transform.position;
    transform.position = v;
}
```

20万次赋值，耗时统计对比

使用方式	耗时/ms
c#	20.7
lua extend	46.0
lua	160

18. 将lua打包成资源包使用

- 菜单命令Lua/build bundle files not jit，先打包输出

到/Assets/StreamingAssets目录

- 使用assetBuddle和WWW加载资源。

```
#else
        string main = "file:///" + streamingPath + "/" + LuaConst.osDir + "/" + LuaConst.osDir;
#endif
        WWW www = new WWW(main);
        yield return www;

        AssetBundleManifest manifest = (AssetBundleManifest)www.assetBundle.LoadAsset("AssetBundleManifest");
        List<string> list = new List<string>(manifest.GetAllAssetBundles());
#else
```

例子里是加载的本地资源

19. 使用cjson

- tolua#集成了第三方库cjson

开启

```
//cjson 比较特殊，只new了一个table，没有注册库，这里注册一下
protected void OpenCJson()
{
    luaState.LuaGetField(LuaIndexes.LUA_REGISTRYINDEX, "_LOADED");
    luaState.OpenLibs(LuaDLL.luaopen_cjson);
    luaState.LuaSetField(-2, "cjson");

    luaState.OpenLibs(LuaDLL.luaopen_cjson_safe);
    luaState.LuaSetField(-2, "cjson.safe");
}
```

使用例子

```
local json = require 'cjson'

function Test(str)
    local data = json.decode(str)
    print(data.glossary.title)
    s = json.encode(data)
    print(s)
end
```

20. 使用utf8

- tolua#扩展的utf8库，使用的lua版本是5.1的，并没有utf8的库。

21. 使用C# string

- lua使用c#的字符串

```
function Test()
    local str = System.String.New('男儿当自强')
    local index = str:IndexOfAny('儿自')
    print('and index is: '..index)
    local buffer = str:ToCharArray()
    print('str type is: '..type(str)..' buffer[0] is ' .. buffer[0])
    local luastr = tolua.tolstring(buffer)
    print('lua string is: '..luastr..' type is: '..type(luastr))
    luastr = tolua.tolstring(str)
    print('lua string is: '..luastr)
end
```

22. 使用反射

- lua里使用c#的Reflection机制

反射理解：通过实例获取类定义，或者通过字符串名字获取类定义，然后调用
获得的类定义中的函数。
例子有些复杂，没细看，应该用不到。

23. 使用C# List

例子太长，不看了。

24. 测试函数重载

C#支持重载函数，这个例子用于测试这个。
没细看。

24. 一些性能测试的例子

里面的第一个例子是这种类型的

```
for i = 1, 200000 do
    transform.position = transform.position
end
```

这个在lua里的耗时是c#里的8倍，可以优化成2倍。
见第17个例子。

25. 测试lua堆栈，和一些出错情况。

以后细看。

« 上一篇：protoc-gen-lua
» 下一篇：lua入门

posted @ 2016-11-29 14:08 treert 阅读(10163) 评论(0) 编辑 收藏

系统推荐博文:
· 使用Slua框架开发Unity项目的重要步骤
· topameng/tolua
· Unity3D 预备知识：C#与Lua相互调用
· Unity3D中tolua的"安装部署和使用"教程
· 基于Unity3D的Android游戏添加google广告的方法——使用AdMob

最新知识库文章:
· 程序员：用代码改变世界
· 阿里云的这群疯子
· 为什么说 Java 程序员必须掌握 Spring Boot ？
· 在学习中，有一个比掌握知识更重要的能力
· 如何招到一个靠谱的程序员
» 更多知识库文章...