Unity3D研究院之C#使用Socket与HTTP连接服务器传输数据包

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif(2013-04-29 11:13:17)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签：  [it](http://search.sina.com.cn/?c=blog&q=it&by=tag) | 分类： [Unity3d](http://blog.sina.com.cn/s/articlelist_1324845813_1_1.html) |

From:http://www.apkbus.com/android-497-1.html(只为本人学习，感谢本文作者)

**1.Socket**

**Socket不要写在脚本上，如果写在脚本上游戏场景一旦切换，那么这条脚本会被释放掉，Socket会断开连接。场景切换完毕后需要重新在与服务器建立Socket连接，这样会很麻烦。所以我们需要把Socket写在一个单例的类中，不用继承MonoBehaviour。这个例子我模拟一下，主角在游戏中移动，时时向服务端发送当前坐标，当服务器返回同步坐标时角色开始同步服务端新角色坐标。**

**Socket在发送消息的时候采用的是字节数组，也就是说无论你的数据是 int float short object 都会将这些数据类型先转换成byte[] ，目前在处理发送的地方我使用的是数据包，也就是把（角色坐标）结构体object转换成byte[]发送，这就牵扯一个问题，如何把结构体转成字节数组，如何把字节数组回转成结构体。请大家接续阅读，答案就在后面，哇咔咔。**

**直接上代码**

**JFSocket.cs 该单例类不要绑定在任何对象上**

/////////////////////////////////////////////////////////////////////////////

**[代码]：**

**using UnityEngine;  
using System.Collections;  
using System;  
using System.Threading;  
using System.Text;  
using System.Net;  
using System.Net.Sockets;  
using System.Collections.Generic;  
using System.IO;  
using System.Runtime.InteropServices;  
using System.Runtime.Serialization;  
using System.Runtime.Serialization.Formatters.Binary;**

**public class JFSocket  
{**

**//Socket客户端对象  
private Socket clientSocket;  
//JFPackage.WorldPackage是我封装的结构体，  
//在与服务器交互的时候会传递这个结构体  
//当客户端接到到服务器返回的数据包时，我把结构体add存在链表中。  
public List<JFPackage.WorldPackage> worldpackage;  
//单例模式  
private static JFSocket instance;  
public static JFSocket GetInstance()  
{  
if (instance == null)  
{  
instance = new JFSocket();  
}  
return instance;  
}**

**//单例的构造函数  
JFSocket()  
{  
//创建Socket对象，这里我的连接类型是TCP  
clientSocket = new Socket (AddressFamily.InterNetwork,SocketType.Stream,ProtocolType.Tcp);  
//服务器IP地址  
IPAddress ipAddress = IPAddress.Parse ("192.168.1.100");  
//服务器端口  
IPEndPoint ipEndpoint = new IPEndPoint (ipAddress, 10060);  
//这是一个异步的建立连接，当连接建立成功时调用connectCallback方法  
IAsyncResult result = clientSocket.BeginConnect (ipEndpoint,new AsyncCallback (connectCallback),clientSocket);  
//这里做一个超时的监测，当连接超过5秒还没成功表示超时  
bool success = result.AsyncWaitHandle.WaitOne( 5000, true );  
if ( !success )  
{  
//超时  
Closed();  
Debug.Log("connect Time Out");  
}else  
{  
//与socket建立连接成功，开启线程接受服务端数据。  
worldpackage = new List<JFPackage.WorldPackage>();  
Thread thread = new Thread(new ThreadStart(ReceiveSorket));  
thread.IsBackground = true;  
thread.Start();  
}  
}**

**private void connectCallback(IAsyncResult asyncConnect)  
{  
Debug.Log("connectSuccess");  
}**

**private void ReceiveSorket()  
{  
//在这个线程中接受服务器返回的数据  
while (true)  
{**

**if(!clientSocket.Connected)  
{  
//与服务器断开连接跳出循环  
Debug.Log("Failed to clientSocket server.");  
clientSocket.Close();  
break;  
}  
try  
{  
//接受数据保存至bytes当中  
byte[] bytes = new byte[4096];  
//Receive方法中会一直等待服务端回发消息  
//如果没有回发会一直在这里等着。  
int i = clientSocket.Receive(bytes);  
if(i <= 0)  
{  
clientSocket.Close();  
break;  
}**

**//这里条件可根据你的情况来判断。  
//因为我目前的项目先要监测包头长度，  
//我的包头长度是2，所以我这里有一个判断  
if(bytes.Length > 2)  
{  
SplitPackage(bytes,0);  
}else  
{  
Debug.Log("length is not > 2");  
}**

**}  
catch (Exception e)  
{  
Debug.Log("Failed to clientSocket error." + e);  
clientSocket.Close();  
break;  
}  
}  
}**

**private void SplitPackage(byte[] bytes , int index)  
{  
//在这里进行拆包，因为一次返回的数据包的数量是不定的  
//所以需要给数据包进行查分。  
while(true)  
{  
//包头是2个字节  
byte[] head = new byte[2];  
int headLengthIndex = index + 2;  
//把数据包的前两个字节拷贝出来  
Array.Copy(bytes,index,head,0,2);  
//计算包头的长度  
short length = BitConverter.ToInt16(head,0);  
//当包头的长度大于0 那么需要依次把相同长度的byte数组拷贝出来  
if(length > 0)  
{  
byte[] data = new byte[length];  
//拷贝出这个包的全部字节数  
Array.Copy(bytes,headLengthIndex,data,0,length);  
//把数据包中的字节数组强制转换成数据包的结构体  
//BytesToStruct()方法就是用来转换的  
//这里需要和你们的服务端程序商量，  
JFPackage.WorldPackage wp = new JFPackage.WorldPackage();  
wp = (JFPackage.WorldPackage)BytesToStruct(data,wp.GetType());  
//把每个包的结构体对象添加至链表中。  
worldpackage.Add(wp);  
//将索引指向下一个包的包头  
index = headLengthIndex + length;**

**}else  
{  
//如果包头为0表示没有包了，那么跳出循环  
break;  
}  
}  
}**

**//向服务端发送一条字符串  
//一般不会发送字符串应该是发送数据包  
public void SendMessage(string str)  
{  
byte[] msg = Encoding.UTF8.GetBytes(str);**

**if(!clientSocket.Connected)  
{  
clientSocket.Close();  
return;  
}  
try  
{  
//int i = clientSocket.Send(msg);  
IAsyncResult asyncSend = clientSocket.BeginSend (msg,0,msg.Length,SocketFlags.None,new AsyncCallback (sendCallback),clientSocket);  
bool success = asyncSend.AsyncWaitHandle.WaitOne( 5000, true );  
if ( !success )  
{  
clientSocket.Close();  
Debug.Log("Failed to SendMessage server.");  
}  
}  
catch  
{  
Debug.Log("send message error" );  
}  
}**

**//向服务端发送数据包，也就是一个结构体对象  
public void SendMessage(object obj)  
{**

**if(!clientSocket.Connected)  
{  
clientSocket.Close();  
return;  
}  
try  
{  
//先得到数据包的长度  
short size = (short)Marshal.SizeOf(obj);  
//把数据包的长度写入byte数组中  
byte [] head = BitConverter.GetBytes(size);  
//把结构体对象转换成数据包，也就是字节数组  
byte[] data = StructToBytes(obj);**

**//此时就有了两个字节数组，一个是标记数据包的长度字节数组，一个是数据包字节数组，  
//同时把这两个字节数组合并成一个字节数组**

**byte[] newByte = new byte[head.Length + data.Length];  
Array.Copy(head,0,newByte,0,head.Length);  
Array.Copy(data,0,newByte,head.Length, data.Length);**

**//计算出新的字节数组的长度  
int length = Marshal.SizeOf(size) + Marshal.SizeOf(obj);**

**//向服务端异步发送这个字节数组  
IAsyncResult asyncSend = clientSocket.BeginSend (newByte,0,length,SocketFlags.None,new AsyncCallback (sendCallback),clientSocket);  
//监测超时  
bool success = asyncSend.AsyncWaitHandle.WaitOne( 5000, true );  
if ( !success )  
{  
clientSocket.Close();  
Debug.Log("Time Out !");  
}**

**}  
catch (Exception e)  
{  
Debug.Log("send message error: " + e );  
}  
}**

**//结构体转字节数组  
public byte[] StructToBytes(object structObj)  
{**

**int size = Marshal.SizeOf(structObj);  
IntPtr buffer = Marshal.AllocHGlobal(size);  
try  
{  
Marshal.StructureToPtr(structObj,buffer,false);  
byte[] bytes = new byte[size];  
Marshal.Copy(buffer, bytes,0,size);  
return bytes;  
}  
finally  
{  
Marshal.FreeHGlobal(buffer);  
}  
}  
//字节数组转结构体  
public object BytesToStruct(byte[] bytes, Type strcutType)  
{  
int size = Marshal.SizeOf(strcutType);  
IntPtr buffer = Marshal.AllocHGlobal(size);  
try  
{  
Marshal.Copy(bytes,0,buffer,size);  
return Marshal.PtrToStructure(buffer, strcutType);  
}  
finally  
{  
Marshal.FreeHGlobal(buffer);  
}**

**}**

**private void sendCallback (IAsyncResult asyncSend)  
{**

**}**

**//关闭Socket  
public void Closed()  
{**

**if(clientSocket != null && clientSocket.Connected)  
{  
clientSocket.Shutdown(SocketShutdown.Both);  
clientSocket.Close();  
}  
clientSocket = null;  
}**

**}**

/////////////////////////////////////////////////////////////////////////////

**为了与服务端达成默契，判断数据包是否完成。我们需要在数据包中定义包头，包头一般是这个数据包的长度，也就是结构体对象的长度。正如代码中我们把两个数据类型 short 和 object 合并成一个新的字节数组。**

**然后是数据包结构体的定义，需要注意如果你在做IOS和Android的话数据包中不要包含数组，不然在结构体转换byte数组的时候会出错。**

**Marshal.StructureToPtr () error : Attempting to JIT compile method**

**JFPackage.cs**

**[代码]：**

**using UnityEngine;  
using System.Collections;  
using System.Runtime.InteropServices;**

**public class JFPackage  
{  
 //结构体序列化  
 [System.Serializable]  
 //4字节对齐 iphone 和 android上可以1字节对齐  
 [StructLayout(LayoutKind.Sequential, Pack = 4)]  
 public struct WorldPackage  
 {  
 public byte mEquipID;  
 public byte mAnimationID;  
 public byte mHP;  
 public short mPosx;  
 public short mPosy;  
 public short mPosz;  
 public short mRosx;  
 public short mRosy;  
 public short mRosz;**

**public WorldPackage(short posx,short posy,short posz, short rosx, short rosy, short rosz,byte equipID,byte animationID,byte hp)  
 {  
 mPosx = posx;  
 mPosy = posy;  
 mPosz = posz;  
 mRosx = rosx;  
 mRosy = rosy;  
 mRosz = rosz;  
 mEquipID = equipID;  
 mAnimationID = animationID;  
 mHP = hp;  
 }**

**}**

**}**

**在脚本中执行发送数据包的动作，在Start方法中得到Socket对象。**

**[代码]：**

public JFSocket mJFsorket; void Start () { mJFsorket = JFSocket.GetInstance(); }

**让角色发生移动的时候，调用该方法向服务端发送数据。**

**[代码]：**

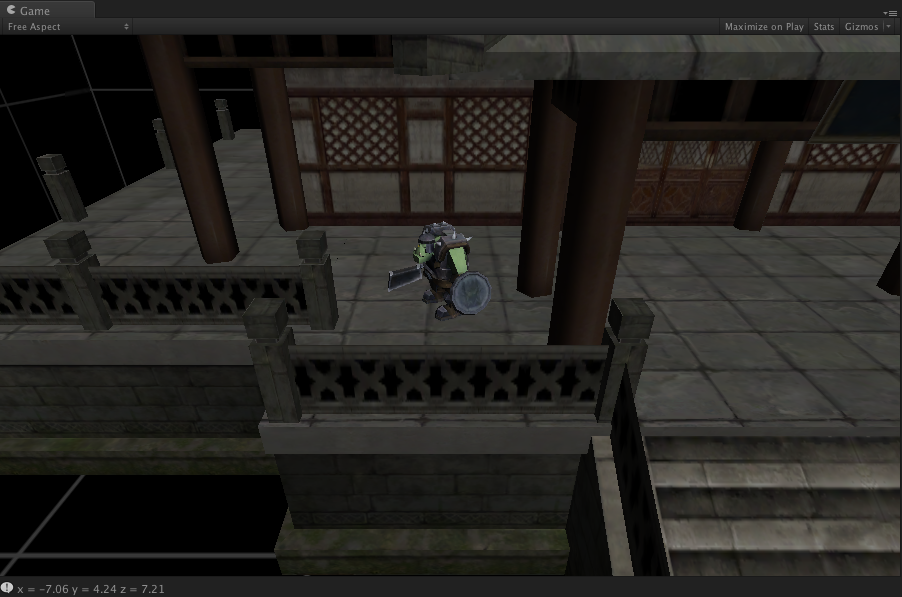
void SendPlayerWorldMessage() { //组成新的结构体对象，包括主角坐标旋转等。 Vector3 PlayerTransform = transform.localPosition; Vector3 PlayerRotation = transform.localRotation.eulerAngles; //用short的话是2字节，为了节省包的长度。这里乘以100 避免使用float 4字节。当服务器接受到的时候小数点向前移动两位就是真实的float数据 short px = (short)(PlayerTransform.x\*100); short py = (short)(PlayerTransform.y\*100); short pz = (short)(PlayerTransform.z\*100); short rx = (short)(PlayerRotation.x\*100); short ry = (short)(PlayerRotation.y\*100); short rz = (short)(PlayerRotation.z\*100); byte equipID = 1; byte animationID =9; byte hp = 2; JFPackage.WorldPackage wordPackage = new JFPackage.WorldPackage(px,py,pz,rx,ry,rz,equipID,animationID,hp); //通过Socket发送结构体对象 mJFsorket.SendMessage(wordPackage); }

**接着就是客户端同步服务器的数据，目前是测试阶段所以写的比较简陋，不过原理都是一样的。**

**[代码]：**

//上次同步时间 private float mSynchronous; void Update () { mSynchronous +=Time.deltaTime; //在Update中每0.5s的时候同步一次 if(mSynchronous > 0.5f) { int count = mJFsorket.worldpackage.Count; //当接受到的数据包长度大于0 开始同步 if(count > 0) { //遍历数据包中 每个点的坐标 foreach(JFPackage.WorldPackage wp in mJFsorket.worldpackage) { float x = (float)(wp.mPosx / 100.0f); float y = (float)(wp.mPosy /100.0f); float z = (float)(wp.mPosz /100.0f); Debug.Log("x = " + x + " y = " + y+" z = " + z); //同步主角的新坐标 mPlayer.transform.position = new Vector3 (x,y,z); } //清空数据包链表 mJFsorket.worldpackage.Clear(); } mSynchronous = 0; } }

**主角移动的同时，通过Socket时时同步坐标喔。。有没有感觉这个牛头人非常帅气 哈哈哈。**

[](http://www.apkbus.com/data/attachment/portal/201303/04/114005vmm7553583b8v835.png)

**对于Socket的使用，我相信没有比MSDN更加详细的了。 有关Socket 同步请求异步请求的地方可以参照MSDN 链接地址给出来了，好好学习吧，嘿嘿。** [**http://msdn.microsoft.com/zh-cn/library/system.net.sockets.socket.aspx**](http://msdn.microsoft.com/zh-cn/library/system.net.sockets.socket.aspx)

**上述代码中我使用的是Thread() 没有使用协同任务StartCoroutine() ，原因是协同任务必需要继承MonoBehaviour，并且该脚本要绑定在游戏对象身上。问题绑定在游戏对象身上切换场景的时候这个脚本必然会释放，那么Socket肯定会断开连接，所以我需要用Thread，并且协同任务它并不是严格意义上的多线程。**

**2.HTTP**

**HTTP请求在Unity我相信用的会更少一些，因为HTTP会比SOCKET慢很多，因为它每次请求完都会断开。废话不说了，我用HTTP请求制作用户的登录。用HTTP请求直接使用Unity自带的www类就可以，因为HTTP请求只有登录才会有，所以我就在脚本中来完成，使用 www 类 和 协同任务**StartCoroutine()。

**[代码]：**

using UnityEngine; using System.Collections; using System.Collections.Generic; public class LoginGlobe : MonoBehaviour { void Start () { //GET请求 StartCoroutine(GET("http://xuanyusong.com/")); } void Update () { } void OnGUI() { } //登录 public void LoginPressed() { //登录请求 POST 把参数写在字典用 通过www类来请求 Dictionary dic = new Dictionary (); //参数 dic.Add("action","0"); dic.Add("usrname","xys"); dic.Add("psw","123456"); StartCoroutine(POST("http://192.168.1.12/login.php",dic)); } //注册 public void SingInPressed() { //注册请求 POST Dictionary dic = new Dictionary (); dic.Add("action","1"); dic.Add("usrname","xys"); dic.Add("psw","123456"); StartCoroutine(POST("http://192.168.1.12/login.php",dic)); } //POST请求 IEnumerator POST(string url, Dictionary post) { WWWForm form = new WWWForm(); foreach(KeyValuePair post\_arg in post) { form.AddField(post\_arg.Key, post\_arg.Value); } WWW www = new WWW(url, form); yield return www; if (www.error != null) { //POST请求失败 Debug.Log("error is :"+ www.error); } else { //POST请求成功 Debug.Log("request ok : " + www.text); } } //GET请求 IEnumerator GET(string url) { WWW www = new WWW (url); yield return www; if (www.error != null) { //GET请求失败 Debug.Log("error is :"+ www.error); } else { //GET请求成功 Debug.Log("request ok : " + www.text); } } }

**如果想通过HTTP传递二进制流的话 可以使用 下面的方法。**

**目前Socket数据包还是没有进行加密算法，后期我会补上。欢迎讨论，互相学习互相进度加油，蛤蛤。**

**下载地址我不贴了，因为没有服务端的东西运行也看不到效果。**