

个人资料



房燕良

+ 加关注 发私信



访问： 125183次

积分： 1743

等级： BLOG > 4

排名： 第12304名

原创： 27篇 转载： 0篇

译文： 1篇 评论： 160条

引擎技术交流QQ群



游戏引擎能吃吗

(264656505)

文章搜索

常用网址

Unity3D官网  
Unreal Engine官网  
游戏蛮牛

文章分类

Unity3D (19)  
Unreal Engine (2)  
自研引擎 (2)  
其它 (5)

CSDN专家精选，微信开发学习路线大有看头！ 【博乐】点评美文，得C币 iOS开发前沿与Swift探秘 Swift初级教程大汇总

## 原 《炉石传说》架构设计赏析(7)：使用Google.ProtocolBuffers处理网络消息

分类： Unity3D

2014-12-15 11:38 4592人阅读 评论(5) 收藏 举报

unity3d google protocolbuffe 网络 网络游戏

目录(?) [+]

这段时间琢磨了一下Unity3D网络游戏开发中的网络消息处理。网络游戏的服务端一般都是自主开发的，所以对应的网络消息处理也要自己开发。客户端/服务端之间的消息传到目前使用JSON和Google.ProtocolBuffers是两种常见的做法。打开炉石的代码看了看它的处理方式，感觉代码写的还是很好的，把它的思路分析一下，与大家分享。

### 整体机制描述

我们想要达到的目标大概是这样的：

- 有N个网络消息，每个消息对应一个Proto中的message描述；
- 每个消息对应一个数字ID；
- 底层在收到消息是，将其解析成为Google.ProtocolBuffers.IMessage对象，这个对象的具体类型应该是前面那个message生成的代码；
- 发送消息就简单了，一个IMessage类型，可以直接执行序列化；

炉石使用Google.ProtocolBuffers处理网络消息发送

发送的机制很简单，首先使用IMessage接口生成的message类构造一个消息对象，例如：

ConnectAPI.SendPing()



CSDN 手机客户端

快速回复

```
[csharp]
01. public static void SendPing()
02. {
03.     Ping.Builder body = Ping.CreateBuilder();
04.     QueueGamePacket(0x73, body);
05.     s_lastGameServerPacketSentTime = DateTime.Now;
06. }
```

底层会构造一个“PegasusPacket”数据包对象，添加到发送队列之中，这个数据包对象主要包含3部分：消息ID，消息大小，具体消息数据。详见PegasusPacket.Encode()函数：

```
[csharp]
01. public override byte[] Encode()
02. {
03.     if (!(this.Body is IMessageLite))
```

## 文章存档

2015年08月 (1)

2015年07月 (3)

2015年04月 (1)

2015年03月 (1)

2015年02月 (1)



## 阅读排行

《炉石传说》架构设计赏

(13888)

你不再需要TinyXML, 推

(11247)

使用Unity3D的50个技巧

(10071)

分享一个小工具: Excel转

(7735)

2009年混合语言编程总结

(7620)

《炉石传说》架构设计赏

(7036)

Unity3D-RPG项目实战 (

(4863)

Unity 4.6的使用匿名dele

(4715)

《炉石传说》架构设计赏

(4589)

Unity3D-RPG项目实战 (

(4539)

## 最新评论

次世代关卡制作流程: 使用Unre:  
风吹夏天: 大神回来了分享一个小工具: Excel表快速转  
valliant303: 解压缩看到一个  
excel2json.exe的可执行文件,  
但是没看到你提到的例子表格,  
我该怎么转换自己...《炉石传说》架构设计赏析(2):  
ZeroDeep: QQ群多少啊分享一个小工具: Excel表快速转  
胖小个花: 感谢分享了哈, 随便  
试了下可以的。不过我要组织好  
excel的格式。我想问下楼主, 如  
果我的excel里...分享一个小工具: Excel表快速转  
房燕良: @liyingwill123:已经上  
传:  
http://download.csdn.net/detai...分享一个小工具: Excel表快速转  
liyingwill123: 分享个编译的版本  
吧, 不可能为了用这个我还要装  
个visual Studio啊次世代关卡制作流程: 使用Unre:  
王静娜: 学习了, 给个赞次世代关卡制作流程: 使用Unre:  
jump fire: 好厉害的感觉。  
(๑\*۞\*๑)次世代关卡制作流程: 使用Unre:  
Json-Niu: 好高大上, 学习了。次世代关卡制作流程: 使用Unre:  
717606641: 学习了, 感谢博主

```
04.     {
05.         return null;
06.     }
07.     IMessageLite body = (IMessageLite) this.Body;
08.     this.Size = body.SerializedSize;
09.     byte[] destinationArray = new byte[8 + this.Size];
10.     Array.Copy(BitConverter.GetBytes(this.Type), 0, destinationArray, 0, 4);
11.     Array.Copy(BitConverter.GetBytes(this.Size), 0, destinationArray, 4, 4);
12.     body.WriteTo(CodedOutputStream.CreateInstance(destinationArray, 8, this.Size));
13.     return destinationArray;
14. }
```

## 消息接收与解析

接下来我们重点看一下消息的接收与解析机制。首先因为TCP是流式的, 所以底层应该检测数据包头, 并收集到一个完整的数据包, 然后再发送到上层解析, 这部分逻辑是

在"ClientConnection<PacketType>.BytesReceived()"中实现的。当收到完整数据包时, 会在主线程中触发"OnPacketCompleted"事件, 实际上会调用到"ConnectAPI.PacketReceived()", 其内部主要是调用了"ConnectAPI.QueuePacketReceived()", 这个函数负责将TCP层接收到的byte[]解析成对应的IMessage对象。

重点来了! 由于网络层发过来的数据包, 只包含一个消息ID, 那么客户端就需要解决从ID找到相应的消息Type的问题。想象中无非有两种方式去做: 1是手动记录每个ID对应的Type; 2是搞一个中间的对应关系的类, 附上自定义的Attribute, 然后在使用反射机制自动收集这些类, 其实和前者也差不多。炉石采用了第一种方式。整体机制是这样的:

- 客户端每个消息对应一个PacketDecoder的派生类对象;
- ConnectAPI类使用一个字典, 用来保存<消息ID, Decoder对象>之间的对应关系:  
ConnectAPI.s\_packetDecoders: SortedDictionary<Int32, ConnectAPI.PacketDecoder>;
- 如果每个消息都要写一个Decoder, 而其内部代码由完全一致, 岂不是很蛋疼?! 好吧, 我们用模板来实现, 详见后续分析;
- 在ConnectAPI.ConnectInit()初始化的时候, 创建Decoder对象, 并保存到上述dict之中, 类似这样:  
s\_packetDecoders.Add(0x74, new DefaultProtoBufPacketDecoder<Pong, Pong.Builder>());
- 最后在上述的收到完整数据包的函数中, 根据数据包中记录的消息ID, 去查找Decoder, 然后调用其方法得到具体的消息对象, 类似这样:

```
[csharp]
01. if (s_packetDecoders.TryGetValue(packet.Type, out decoder))
02. {
03.     PegasusPacket item = decoder.HandlePacket(packet);
04.     if (item != null)
05.     {
06.         queue.Enqueue(item);
07.     }
08. }
09. else
10. {
11.     Debug.LogError("Could not find a packet decoder for a packet of type " + packet.Type);
12. }
```

最后我们看一下, Decoder模板的实现技巧。首先消息解析的具体操作是有Google.ProtocolBuffers生成的代码去实现的, 所以具体操作流程是完全一致的, 这些写到基类的静态模板函数中:

```
[csharp]
01. public abstract class PacketDecoder
02. {
03.     // Methods
04.     public abstract PegasusPacket HandlePacket(PegasusPacket p);
05.     public static PegasusPacket HandleProtoBuf(TMessage, TBuilder)
        (PegasusPacket p) where TMessage: IMessageLite<TMessage, TBuilder> where TBuilder: IBuilderLite<
```



```
06.     {
07.         byte[] body = (byte[]) p.Body;
08.         TBuilder local2 = default(TBuilder);
09.         TBuilder local = (local2 == null) ? Activator.CreateInstance<TBuilder>
           () : default(TBuilder);
10.         p.Body = local.MergeFrom(body).Build();
11.         return p;
12.     }
13. }
```

其次，使用一个模板派生类，实现HandlePacket()这个虚函数，主要的目的只是把TMessage和TBuilder这两个类型传给那个静态函数而已：

```
[csharp]
01. public class DefaultProtobufPacketDecoder<TMessage, TBuilder> : ConnectAPI.PacketDecoder where
02. {
03.     // Methods
04.     public override PegasusPacket HandlePacket(PegasusPacket p)
05.     {
06.         return ConnectAPI.PacketDecoder.HandleProtoBuf<TMessage, TBuilder>(p);
07.     }
08. }
```

OK，炉石是使用使用ProtocolBuffers来处理网络消息的机制就是这样，是不是已经很清晰啦！

版权声明：本文为博主原创文章，未经博主允许不得转载。



- ^ 上一篇
 在Unity3D的Legacy动画系统中应用Root Motion
- v 下一篇
 Unity3D脚本中的Awake()和Start()的本质区别

顶
 2

踩
 2

主题推荐

- 架构设计
- 网络
- class
- 博客
- pre

猜你在找

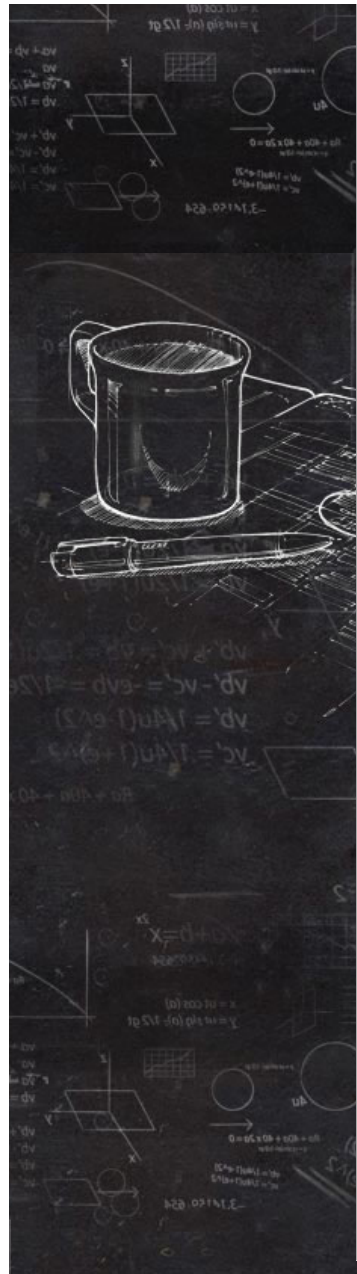
- 深入浅出Unity3D——第一篇
 ■ 《炉石传说》架构设计赏析2Scene管理
 ■ 实战进阶学习Unity3d游戏开发
 ■ 《炉石传说》架构设计赏析4Asset管理
 ■ 基于Unity的游戏开发（下）
 ■ 《炉石传说》架构设计赏析5卡牌&技能的静态数据
 ■ Android入门实战教程
 ■ 《炉石传说》架构设计赏析1游戏启动流程
 ■ Qt基础与Qt on Android入门
 ■ 并发网络服务程序——架构设计关注点

准备好了么？跳吧！

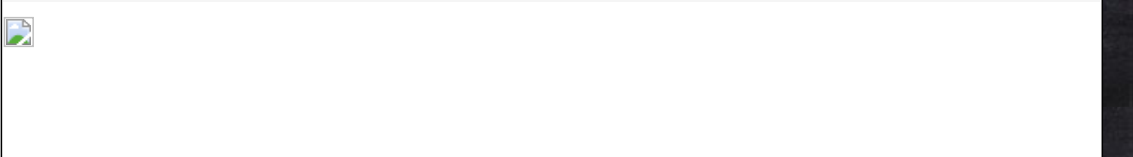
更多职位尽在 CSDN JOB

- 计算机桌面及网络维护工程师
 我要跳槽
 ■ 产品经理（网络直播平台）
 我要跳槽
- 北京盛华合创科技有限公司
 5-8K/月
 上海骏梦网络科技有限公司
 35-50K/月





■ 社交网络产品经理	我要跳槽	■ 高级网络工程师	我要跳槽
人瑞网络科技有限公司	30-60K/月	北京夏墨咨询服务有限公司	15-30K/月



查看评论

3楼 [lirentai](#) 2015-01-07 00:48发表

楼主uml图和各个类的描述都很清晰，我在架构上是小白，希望能在楼主讲解里面看到更多的对架构上的优劣分析：)

2楼 [silveryw](#) 2014-12-18 11:44发表

LZ分析项目的思路很老道，我想问那个类似UML图是用什么工具画的？我是做unity的新手，我这边也反编了代码在学习，但是遇到yield return部分，代码就会比较难分析。。有什么好的反编译C#的工具推荐吗？

Re: [房燕良](#) 2014-12-18 17:37发表

回复[silveryw](#)：UML图是用Microsoft Visio画的。反编译C#我用的Reflector，可以到我的QQ群里面来下载。

1楼 [u010384128](#) 2014-12-15 14:35发表

亲，给来个可以运行的源代码呗！

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目											
全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker		
OpenStack	VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	
WAP	jQuery	BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora XML
LBS	Unity	Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra		
CloudStack	FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App			
SpringSide	Maemo	Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP		
HBase	Pure	Solr	Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap			

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 [webmaster@csdn.net](mailto:webmaster@csdn.net) 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持  
京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

