

平顶山学院

课程设计报告

课程名称: C#高级程序设计

设计题目 3D 坦克大战联网对战版之服务端

院（系）: 计算机学院（软件学院）

专业年级: 软件工程（游戏开发工程师）2017 级

学 号: 171530425

姓 名: 徐 可 可

指导教师: 彭 伟 国

2019 年 12 月 16 日

目 录

1 系统需求分析	3
1.1 项目意义	3
1.2 系统需求分析	3
1.2.1 系统需求调查研究	3
1.2.2 系统需求调查结果	4
1.2.3 系统功能设计要求	5
2 系统概要设计	6
2.1 登录注册功能	6
3 系统详细设计	7
3.1 记事本	7
3.1 注册功能	7
3.1 登录功能	7
3.2 系统调试及解决方法	8
4 系统运行结果	9
5 项目评价	13
参考文献	13
附录：源代码	14

1 系统需求分析

1.1 项目意义

对于我本人来说,开发调试一款较为复杂的项目,可以充分锻炼我各方面的能力。

首先锻炼的就是我对所使用的开发语言,C#掌握的能力。其次,由于该游戏基于 Unity 开发,也涉及到网络和数据库以及团队协作开发的操作,所以也会锻炼我对 Unity、SQL、Git、TCP 等的掌握和综合应用能力。

另外由于本游戏是开源的,任何人都能够获取到本游戏的源代码,并且里面不乏有一些优秀的设计之处,所以能够给其他开发者提供借鉴。

1.2 系统需求分析

1.2.1 系统需求调查研究

开发一款网络游戏,必不可少的要开发服务端,而开发服务端并不容易,首先就是需要有好的架构。



图 1-1 百度搜索服务器端架构

软件架构的分析,可以通过不同的层面入手。比较经典的软件架构描述,包含了以下几种架构:运行时架构、逻辑架构、物理架构、数据架构、开发架构。



图 1-2 常见的架构描述

1.2.2 系统需求调查结果

服务器端软件的本质，是一个会长期运行的程序，并且它还要服务于多个不定时，不定地点的网络请求。所以这类软件的特点是要非常关注稳定性和性能。这类程序如果需要多个协作来提高承载能力，则还要关注部署和扩容的便利性；同时，还需要考虑如何实现某种程度容灾需求。由于多进程协同工作，也带来了开发的复杂度，这也是需要关注的问题。

功能约束，是架构设计决定性因素。一个万能的架构，必定是无能的架构。一个优秀的架构，则是正好把握了对应业务领域的核心功能产生的。游戏领域的功能特征，于服务器端系统来说，非常明显的表现为几个功能的需求：

- 对于游戏数据和玩家数据的存储
- 对玩家客户端进行数据广播
- 把一部分游戏逻辑在服务器上运算，便于游戏更新内容，以及防止外挂。

针对以上的需求特征，在服务器端软件开发上，我们往往会关注软件对电脑内存和 CPU 的使用，以求在特定业务代码下，能尽量满足承载量和响应延迟的需求。最基本的做法就是“时空转换”，用各种缓存的方式来开发程序，以求在 CPU 时间和内存空间上取得合适的平衡。在 CPU 和内存之上，是另外一个约束因素：网卡。网络带宽直接限制了服务器的处理能力，所以游戏服务器架构也必定要考虑这个因素。

对于游戏服务器架构设计来说，最重要的是利用游戏产品的需求约束，从而优化出对此特定功能最合适的“时一空”架构。并且最小化对网络带宽的占用。

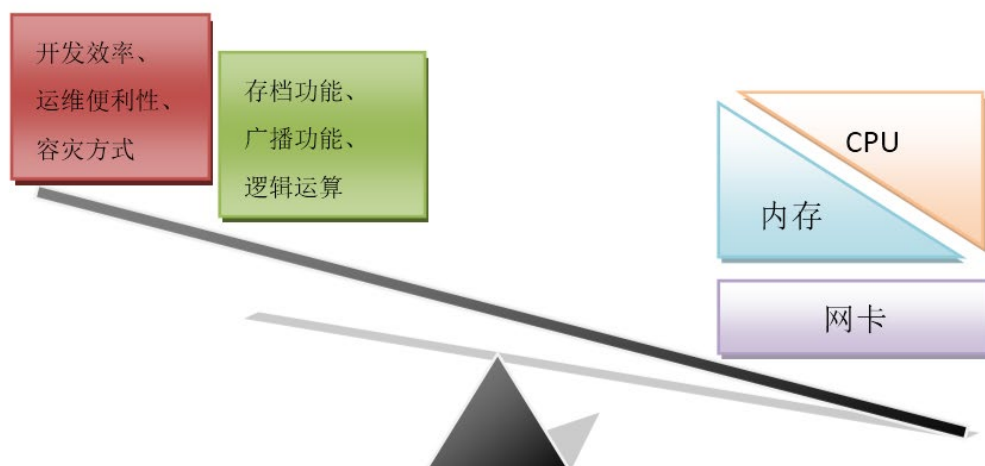


图 1-3 游戏服务器的分析模型

1.2.3 系统功能设计要求

从服务端的角度看，一个玩家会经历连接、登录、获取数据、操作交互、保存数据和退出六个阶段，如图 1-4 所示。

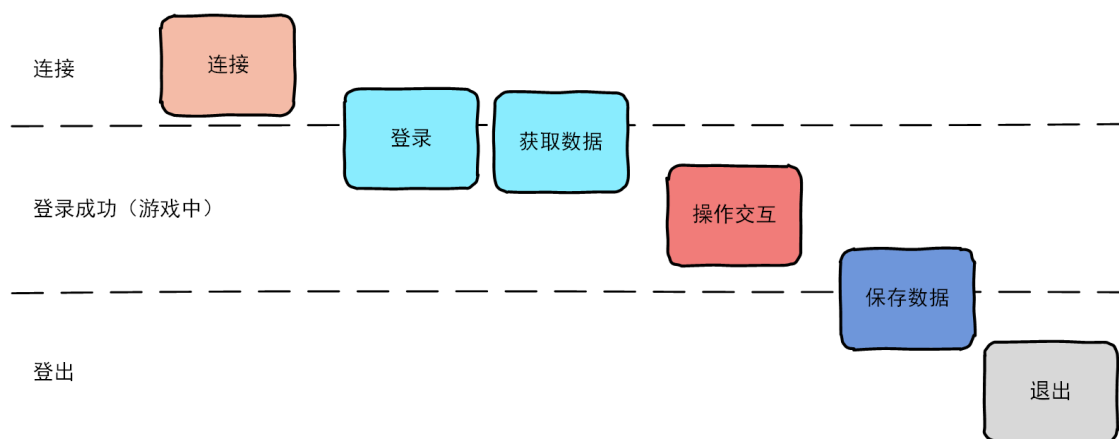


图 1-4 游戏流程

- **连接阶段：**客户端调用 **Connect** 连接服务端即为连接阶段。连接后双端即可通信，但服务端还不知道玩家控制的是哪个角色。于是客户端需要发送一条登录协议，协议中包含用户名、密码等信息，待检验通过后服务端会将网络连接与游戏角色对应起来，从数据库中获取该角色的数据后，才算登录成功。
- **交互阶段：**双端互通协议。**MsgMove**、**MsgAttack**，记事本程序的保存文本功能，都发生在这一阶段。
- **登出阶段：**玩家下线，服务端把玩家的数据保存到数据库中。对于保存玩家数据的时机，不同的服务端会有不同实现。有些服务端采用定时存储的方式，每隔几分钟把在线玩家的数据写回数据库；有些服务端采用下线时存储的方式，只有在玩家下线时才保存数据。上述方式各有优缺点，定时存储相对于下线时存储安全，在服务端突然挂掉的情况下，能够挽回一部分在线玩家数据，但也因为要频繁写数据库，性能较差。本系统采用玩家下线时才保存数据的方式。

对应于上述几个步骤，一个连接会有“连接但未登录”和“登录成功”两种

状态，如表 1-1 所示。

表 1-1 连接状态

状态	说明
连接但未登录	客户端连接(Connect)服务端，服务端还不知道该客户端对应哪个游戏角色。玩家需要输入用户名、密码，服务端验证后从数据库读取角色数据，把连接和角色关联起来
登录成功	连接和角色关联后，玩家可以操作游戏角色，比如打副本、吃药水

2 系统概要设计

2.1 登录注册功能

在服务端程序中为 proto 文件夹添加 LoginMsg.cs 和 NotepadMsg.cs 两个文件，用于定义登录和记事本相关的协议，如图 2-1 所示。

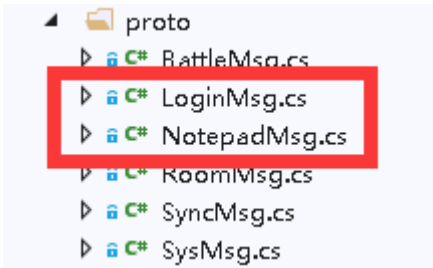


图 2-1 添加 LoginMsg.cs 和 NotepadMsg.cs 两个文件

LoginMsg 中包含了注册、登录和踢出三条协议。MsgRegister 即注册协议，客户端需要发送 id 和 pw 字段，指定要注册的用户名和密码。服务端处理消息后，也会给客户端回应 MsgRegister 协议，如果服务端回应的 result 为 0，代表注册成功，如果为 1。

MsgLogin 即登录协议，客户端也需要发送 id 和 pw 字段，指定要登录的用户名及其密码。服务端收到消息后，会判断密码是否正确，然后加载玩家数据，回应客户端。如果服务端回应的 result 为 0,代表登录成功，如果为 1，代表登录失败。

MsgKick 是由服务端推送的“强制下线”协议。游戏中常有多个客户端同时登录同一个账号的情况，后登录的客户端会把早前登录客户端踢下线。服务端会给早前登录的客户端推送 MsgKick 协议，指明被踢下线的原因。

3 系统详细设计

3.1 记事本

在 NotepadMsg.cs 中编写读取和保存记事本的协议，客户端发送 MsgGetText 协议后，服务端会返回带有 test 字段的同名协议，返回记事本文本。编辑完文本后，玩家点击保存按钮，客户端会发送 MsgSaveText 协议，并将修改后的文本以 text 字段发送给服务端。服务端收到后，更新文本，并返回同名协议。如果 result 为 0，代表保存成功。

3.1 注册功能

在服务端程序中添加 LoginMsgHandle.cs 和 NotepadMsgHandle.cs 两个文件，用于处理登录注册和记事本的协议，如图 3-1 所示。

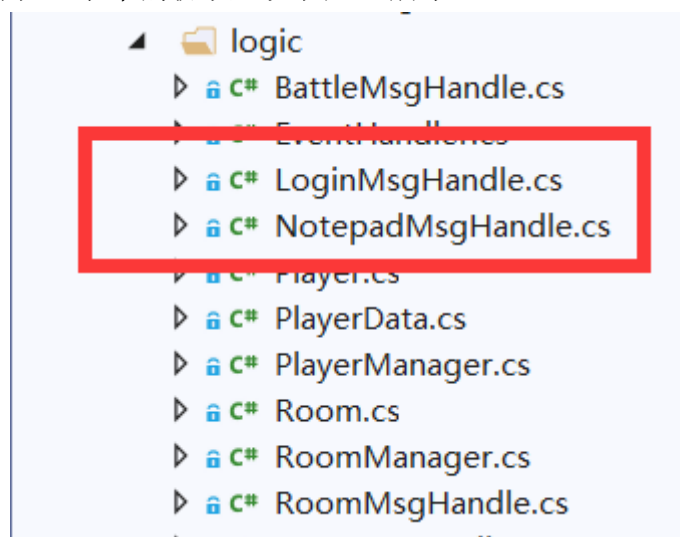


图 3-1 添加 LoginMsgHandle.cs 和 NotepadMsgHandle.cs 两个文件

在 LoginMsgHandle 中编写 MsgHandler 类(partial class MsgHandler)，添加处理注册协议的方法 MsgRegister。MsgRegister 会调用 DbManager.Register 向 account 表写入账号信息，再使用 DbManager.CreatePlayer 向 game 表写入默认的角色信息。最后调用 NetManager.Send 返回协议给客户端。

3.1 登录功能

添加处理登录协议的方法 MsgLogin,它相对复杂，因为要处理下面几项任务。

- 1) 验证密码：通过 DbManager.CheckPassword 验证用户名和密码，如果密码错误，返回 result=1 给客户端。
- 2) 状态判断：如果该客户端已经登录，不能重复登录。
- 3) 踢下线：通过 PlayerManager.IsOnline 判断该账户是否已经登录，如果已经登录，需要先把它踢下线。程序会通过 PlayerManager.GetPlayer(msg.id) 获取已登录的玩家对象，给它发送 MsgKick 协议，通知被踢下线的客户端。最后调用 NetManager.Close 关闭 Socket 连接。
- 4) 读取数据：通过 DbManager.GetPlayerData 从数据库中读取玩家数据。

- 5) 构建 Player: 根据读取到的数据, 构建 player 对象, 并把它添加到 PlayerManager 的列表中, 将客户端信息 ClientState 和 player 对象关联起来。

3.2 系统调试及解决方法

实现过程中曾出现以下调试错误信息:

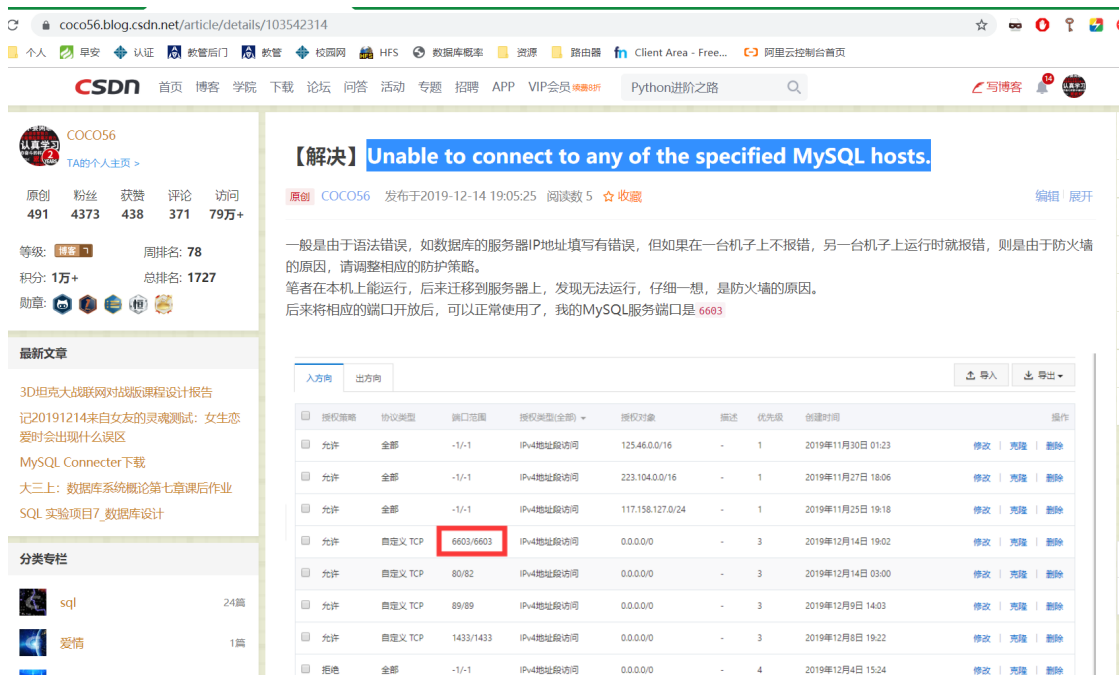
(1)、Unable to connect to any of the specified MySQL hosts.

解决方法:

通过百度搜索自己的曾经写的博文, 成功解决问题。

The screenshot shows a Baidu search result for the query "coco56 Unable to connect to any of the specified MySQL hosts." The search bar at the top contains the query and a "百度一下" button. Below the search bar, there are tabs for "网页", "资讯", "视频", "图片", "知道", "文库", "贴吧", "采购", "地图", and "更多". The search results are displayed below the tabs, showing several links to CSDN and Baidu Zhidao. The first result is from CSDN, titled "...connect to any of the specified MySQL hosts. - ... CSDN博客", with a date of 2017年8月6日 and a reading count of 5268. The second result is also from CSDN, titled "...connect to any of the specified MySQL hosts.解决... CSDN博客", with a date of 2018年4月24日. The third result is from 博客园, titled "...to connect to any of the specified MySQL hosts. - ... 博客园", with a date of 2012年4月20日. The fourth result is also from 博客园, titled "...connect to any of the specified MySQL hosts 的一种... 博客园", with a date of 2011年11月4日. The fifth result is from 百度知道, titled "...connect to any of the specified MySQL hosts 是怎么回事_百...", with a date of 2019年9月2日. The search results are displayed in a list format with links, titles, dates, and reading counts. On the right side of the page, there is a "相关" (Related) section with several icons and a "搜索" (Search) section with a search bar and a "1" button.

3-2 百度搜索结果



3-3 本人自己之前写的博客

4 系统运行结果

打开服务端应用程序，会看到如图 4-1 所示的界面和提示，此时代表连接数据库成功，并成功监听相应的端口，如果出错，则会闪退。

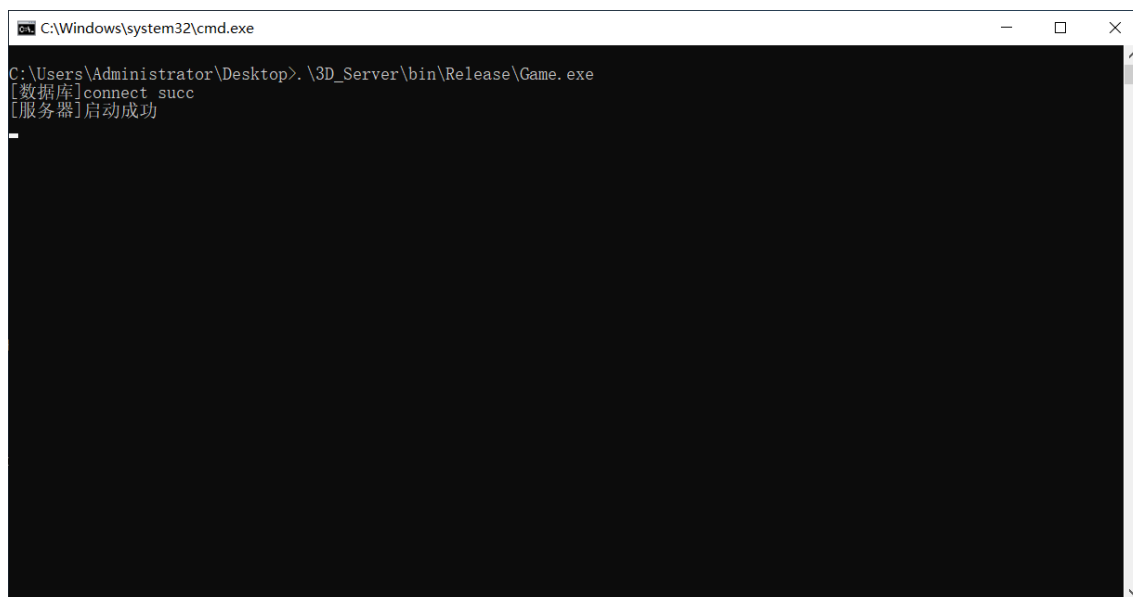
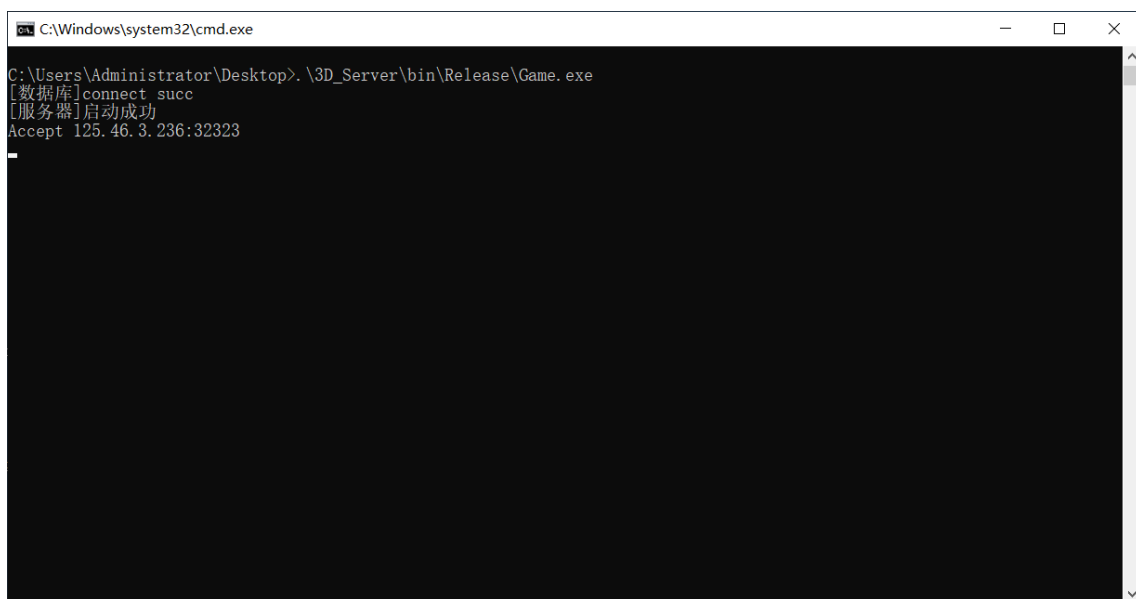


图 4-1 启动成功时的提示

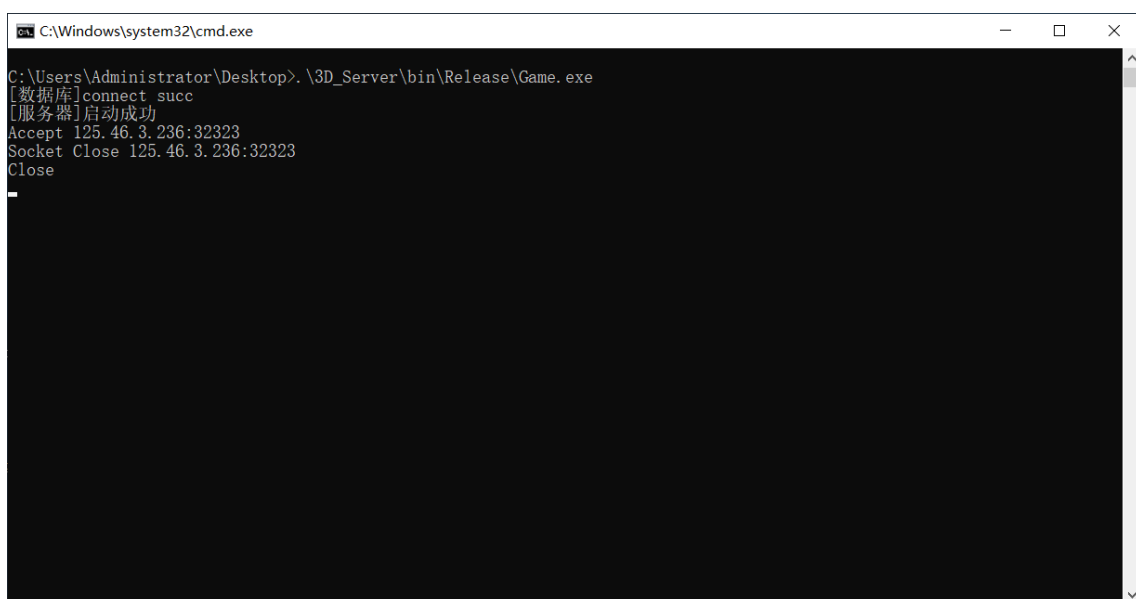
当有新客户端到达时，会有如图 4-2 所示的提示，图 4-2 所示的 Accept 125.46.3.236:32323 意为客户端的 IP 为 125.46.3.236 端口号为 32323。



```
C:\Windows\system32\cmd.exe
C:\Users\Administrator\Desktop>. \3D_Server\bin\Release\Game.exe
[数据库]connect succ
[服务器]启动成功
Accept 125.46.3.236:32323
```

图 4-2 有新客户端到达时的提示

当有客户端关闭时会有如图 4-3 所示的提示，图 4-3 所示的 Socket Close 125.46.3.236:32323 意为客户端的 IP 为 125.46.3.236 端口号为 32323。



```
C:\Windows\system32\cmd.exe
C:\Users\Administrator\Desktop>. \3D_Server\bin\Release\Game.exe
[数据库]connect succ
[服务器]启动成功
Accept 125.46.3.236:32323
Socket Close 125.46.3.236:32323
Close
```

图 4-3 客户端关闭时的提示

当客户端成功登录时，会自动进入房间。提示如图 4-4 所示。另外当客户端打开房间列表面板后，面板左侧会显示玩家的战绩（总胜利次数和总失败次数），因此进入房间时会请求查询战绩（MsgGetAchieve）；客户端面板右侧显示了房间列表，因此还会请求获取房间列表（MsgGetRoomList）；

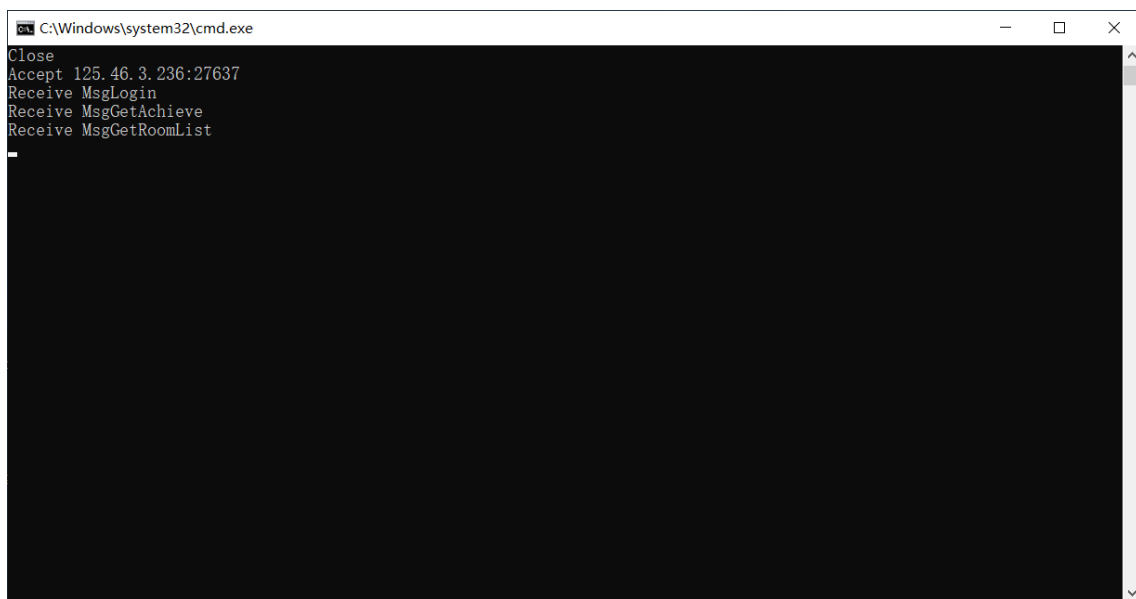


图 4-4 客户端请求获取房间列表的提示

如果玩家拿着手机进入没有信号的山区，或者有人拿剪刀剪断网线，都会导致链路不通。但 TCP 本身的心跳机制太 " 鸡肋 "，要经过 2 个小时的时间才能主动释放资源，游戏程序一般都会自行实现心跳机制。具体来说就是，客户端会定时（如 30 秒）给服务端发送 PING 协议，服务端收到后会回应 PONG 协议。正常情况下，客户端每隔一段时间（如 30 秒）必然会收到服务端的 PONG 协议（就算网络不通畅，最慢 120 秒也总该收到了吧）。如果客户端很长时间（如 120 秒）没有收到 PONG 协议，很大概率是网络不通畅或服务端挂掉，客户端程序可以释放 Socket 资源。其实对于客户端来说，释放不释放关系不大，毕竟只有一个 Socket。但对服务端来说却很重要，因为服务端可能保持着数以万计的连接，当游戏在线人数很多时，只有及时释放资源，才能让玩家正常玩游戏（不然，内存爆满服务器挂掉大家都玩不了）。所以客户端会定时向服务端发送 MsgPing 协议，服务端收到后也会回应 MsgPong 协议。

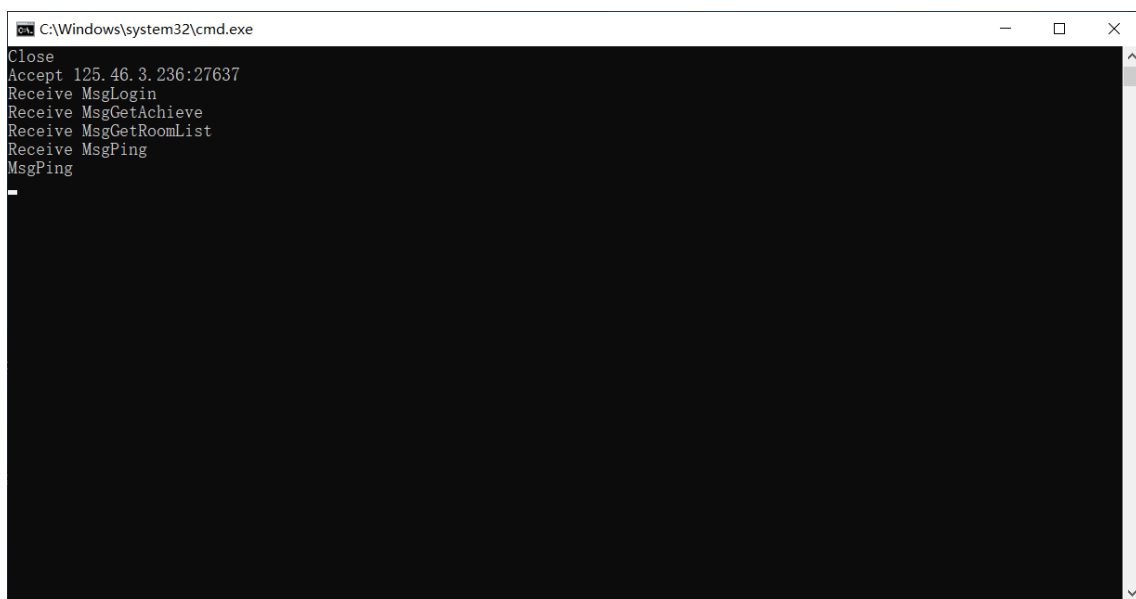
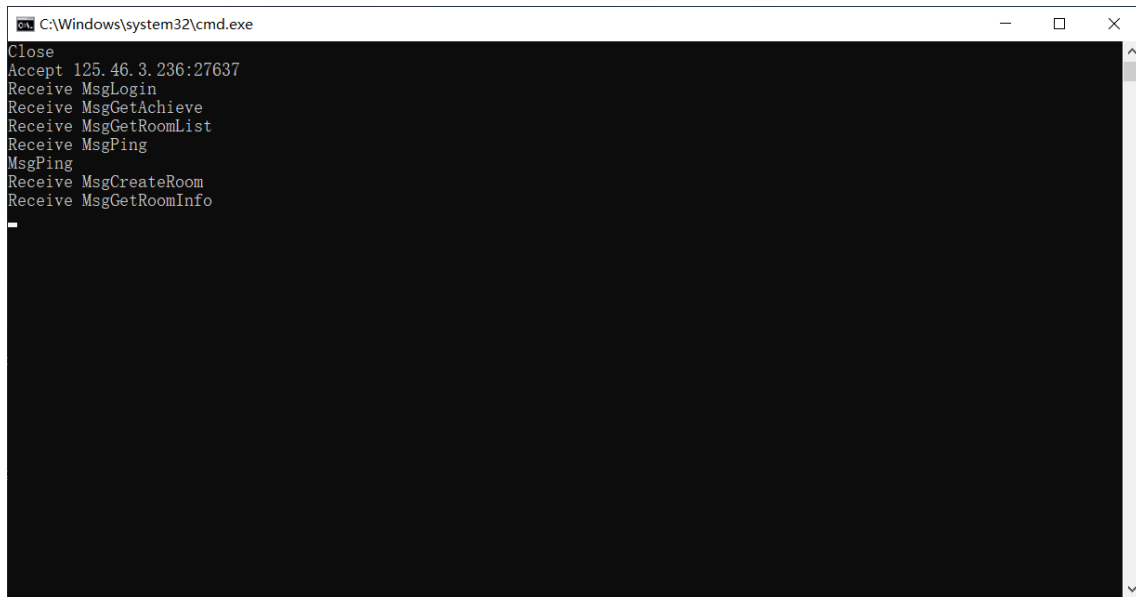


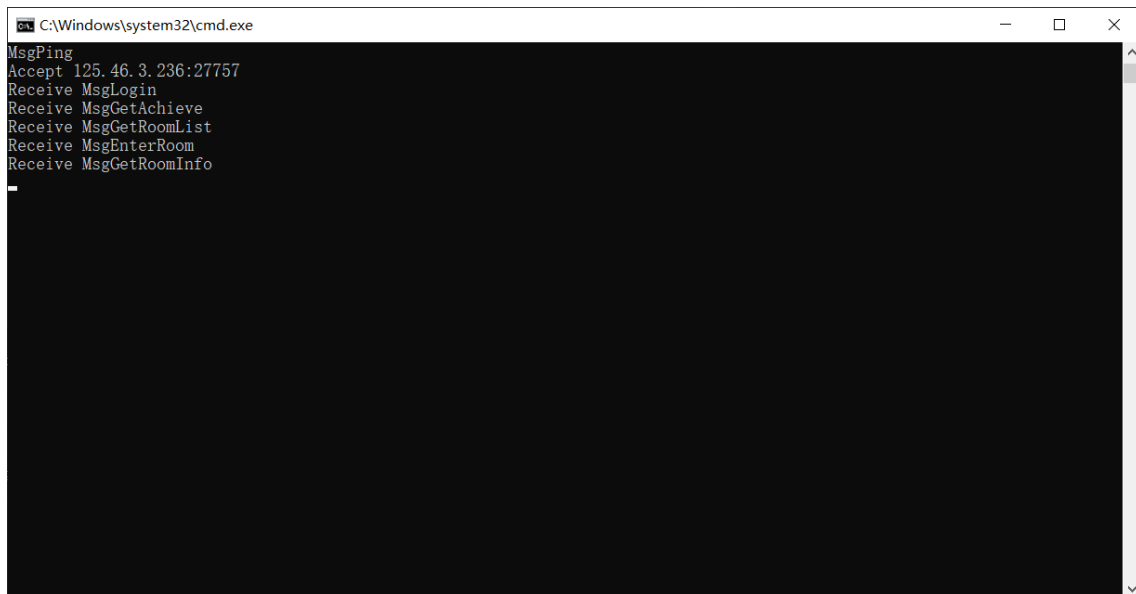
图 4-5 客户端与服务端进行“乒乓”时的提示

客户端登录后面板中有“新建房间”和“加入房间”按钮，涉及 MsgCreateRoom 和 MsgEnterRoom 两条协议；若玩家加入房间，需要获取房间信息(MsgGetRoomInfo 协议)；



```
C:\Windows\system32\cmd.exe
Close
Accept 125.46.3.236:27637
Receive MsgLogin
Receive MsgGetAchieve
Receive MsgGetRoomList
Receive MsgPing
MsgPing
Receive MsgCreateRoom
Receive MsgGetRoomInfo
-
```

图 4-6 客户端创建房间并获取房间信息的提示



```
C:\Windows\system32\cmd.exe
MsgPing
Accept 125.46.3.236:27757
Receive MsgLogin
Receive MsgGetAchieve
Receive MsgGetRoomList
Receive MsgEnterRoom
Receive MsgGetRoomInfo
-
```

图 4-7 客户端进入房间并获取房间信息的提示

战斗过程中，程序会通过 MsgSyncTank,MsgFire,MsgHit 等协议去同步坦克的位置、炮弹位置等信息。当某个阵营取得胜利，服务端会广播 MsgBattleResult 协议，通知客户端哪个阵营获得了胜利。

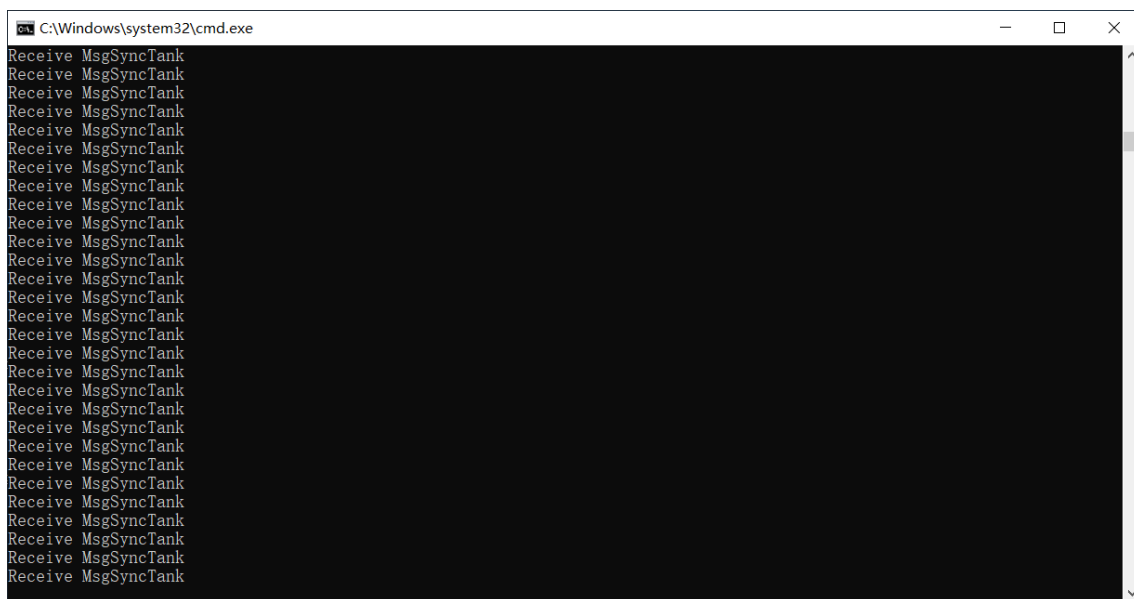


图 4-8 客户端发送同步坦克信息时的提示

5 项目评价

本游戏服务器的设计完成了基本的设计要求，但还有一些地方需要优化，比如如何处理高并发，以及如何支持在服务器集群上运行。另外为了方便用户登录，也可以接入第三方服务，比如接入 QQ 快捷登录，这样用户登录起来就比较方便了，无需专门注册我们的游戏账号，直接用 QQ 扫码登录或注册就可以了。

具体地：

- 由于本游戏不分区不分服，所以在设计服务器的时候，应按世界服的思想去设计，即服务器是一个 n 多台物理机的集群。当用户登陆服务器，创建房间时，可能根据负载均衡算法，它可以在任何一台服务器上面。这样，不管用户登陆到哪一台服务器上面了，都可以获得自己的数据。可以使用 **redis** 来做数据共享。
- 在同一局游戏中，应要求所有人都在同一个房间中，可以规定在同一个房间中的用户，必须登陆到同一台物理服务器上面。在创建房间完成之后，其他人根据房间号查找房间的时候，就可以根据房间号，获取这个房间所在的服务器 **ip** 和端口，判断一个当前用户登陆的服务器 **ip** 与房间所在的服务器 **ip** 是否相同，如果相同，就不做切换，如果不一样，客户端就使用 **ip** 和端口，连接到房间所在的服务器上面。
- 创建房间成功之后，接下来的操作都要保证它的顺序性，所以房间需要有一个它自己的消息队列。可以把每个房间到达服务器的消息封装为一个任务，把这个任务放到消息队列中，然后有一个任务执行者去按顺序执行这些任务。
- 在用户登录时一般都是需要接第三方登陆（如使用 QQ 登录），登陆这一块是 **http** 操作，我们统一提供一个 **web** 服务，用来做登陆验证。因为在登陆时，调用第三方的 **http** 服务，这个过程可能很慢，如果放在逻辑服务器的话，可能会卡业务逻辑任务。因为可能不同的玩家业务请求可能同在一个线程中，如果有任务卡了，那么这个任务以后新来的请求就会卡住，导致消息延迟。

参考文献

[1] 李心蕊. 独立游戏的设计艺术和文化特性研究[D]. 齐齐哈尔：齐齐哈尔大学，2016.

- [2] 刘晓扬. 基于社交网络游戏的行为干预研究[D]. 哈尔滨: 哈尔滨工业大学, 2013.
- [3] 王琛瑜. 移动游戏界面视觉风格的研究与设计应用[D]. 上海: 东华大学, 2017.
- [4] 李绍龙. 网络游戏用户行为分析和应用[D]. 北京: 北京交通大学, 2009.
- [5] 郭振兴. 手机“废城赛车”游戏软件的技术研发[D]. 天津: 天津大学, 2015.
- [6] 杨观. 基于 Unity 的游戏逻辑引擎的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2015.
- [7] 周晓风. 基于 MAYA 和 VIRTTOOLS 的虚拟仓储物流系统的研究与实现[D]. 上海: 上海交通大学, 2011.
- [8] 贺瞿. 基于动力学原理的 Maya 粒子特效插件[D]. 上海: 上海交通大学, 2011.
- [9] 刘佳. 关于 photoshop 应用的几个小技巧[J]. 信息与电脑(理论版), 2013, 5(03):187-188.
- [10] 赵娟. 地图编辑器在苹果手机游戏设计中的开发与应用[D]. 北京: 北京工业大学, 2013.
- [11] 张海藩. 软件工程导论(第6版)[M]. 清华大学出版社, 2016.
- [12] 许鹏森. 游戏软件黑盒测试方法研究与应用[D]. 武汉: 华中科技大学, 2008.

附录：源代码

- 3D_Server\DBConfiguration.cs
 - 1) // 将此文件更名为 DBConfiguration.cs, 并对以下变量进行赋值
 - 2)
 - 3) namespace Game
 - 4) {
 - 5) class DBConfiguration
 - 6) {
 - 7) public static string db= "game";
 - 8) public static string ip = "127.0.0.1";
 - 9) public static int port = 3306;
 - 10) public static string user = "root";
 - 11) public static string pw = "aa123bb456";
 - 12) }
 - 13) }
- 3D_Server\Program.cs
 - 1) namespace Game
 - 2) {
 - 3) class MainClass
 - 4) {

```

5)         public static void Main (string[] args)
6)         {
7)             if(!DbManager.Connect(DBConfiguration.db, DBConfigurat
            ion.ip, DBConfiguration.port, DBConfiguration.user, DBConfiguratio
            n.pw)){
8)                 return;
9)             }
10)
11)             NetManager.StartLoop(82);
12)         }
13)     }
14) }

```

● 3D_Server\script\db\DbManager.cs

```

1) using System;
2) using MySql.Data.MySqlClient;
3) using System.Text.RegularExpressions;
4) using System.Web.Script.Serialization;
5)
6) public class DbManager {
7)     public static MySqlConnection mysql;
8)     static JavaScriptSerializer Js = new JavaScriptSerializer();
9)
10)    //连接 mysql 数据库
11)    public static bool Connect(string db, string ip, int port, str
        ing user, string pw)
12)    {
13)        //创建 MySqlConnection 对象
14)        mysql = new MySqlConnection();
15)        //连接参数
16)        string s = string.Format("Database={0};Data Source={1}; po
            rt={2};User Id={3}; Password={4}",
17)            db, ip, port, user, pw);
18)        mysql.ConnectionString = s;
19)        //连接
20)        try
21)        {
22)            mysql.Open();
23)            Console.WriteLine("[数据库]connect succ ");
24)
25)            return true;
26)        }
27)        catch (Exception e)
28)        {

```



```

29)         Console.WriteLine("[数据库]connect fail, " + e.Message);
30)         return false;
31)     }
32) }
33)
34) //测试并重连
35) private static void CheckAndReconnect(){
36)     try{
37)         if(mysql.Ping()){
38)             return;
39)         }
40)         mysql.Close();
41)         mysql.Open();
42)         Console.WriteLine("[数据库] Reconnect!");
43)     }
44)     catch(Exception e){
45)         Console.WriteLine("[数据库] CheckAndReconnect fail " + e.Message);
46)     }
47)
48) }
49)
50) //判定安全字符串
51) private static bool IsSafeString(string str)
52) {
53)     return !Regex.IsMatch(str, @"[-
54)     ;|,|\V|\(|\)|\[|\]|\\|}|%|@|\*|!|\'|");
55) }
56)
57) //是否存在该用户
58) public static bool IsAccountExist(string id)
59) {
60)     CheckAndReconnect();
61)     //防 sql 注入
62)     if (!DbManager.IsSafeString(id)){
63)         return false;
64)     }
65)     //sql 语句
66)     string s = string.Format("select * from account where id='{
67)     {0}';", id);
68)     //查询
69)     try
70)     {

```

```

69)         MySqlCommand cmd = new MySqlCommand (s, mysql);
70)         MySqlDataReader dataReader = cmd.ExecuteReader ();
71)         bool hasRows = dataReader.HasRows;
72)         dataReader.Close();
73)         return !hasRows;
74)     }
75)     catch(Exception e)
76)     {
77)         Console.WriteLine("[数据
库] IsSafeString err, " + e.Message);
78)         return false;
79)     }
80) }
81)
82) //注册
83) public static bool Register(string id, string pw)
84) {
85)     CheckAndReconnect();
86)     //防 sql 注入
87)     if(!DbManager.IsSafeString(id)){
88)         Console.WriteLine("[数据
库] Register fail, id not safe");
89)         return false;
90)     }
91)     if(!DbManager.IsSafeString(pw)){
92)         Console.WriteLine("[数据
库] Register fail, pw not safe");
93)         return false;
94)     }
95)     //能否注册
96)     if (!IsAccountExist(id))
97)     {
98)         Console.WriteLine("[数据库] Register fail, id exist");
99)         return false;
100)    }
101)    //写入数据库 User 表
102)    string sql = string.Format("insert into account set id
='{0}' ,pw='{1}';", id, pw);
103)    try
104)    {
105)        MySqlCommand cmd = new MySqlCommand(sql, mysql);
106)        cmd.ExecuteNonQuery();
107)        return true;
108)    }

```

```

109)         catch(Exception e)
110)         {
111)             Console.WriteLine("[数据
库] Register fail " + e.Message);
112)             return false;
113)         }
114)     }
115)

116)     //创建角色
117)     public static bool CreatePlayer(string id)
118)     {
119)         CheckAndReconnect();
120)         //防 sql 注入
121)         if(!DbManager.IsSafeString(id)){
122)             Console.WriteLine("[数据
库] CreatePlayer fail, id not safe");
123)             return false;
124)         }
125)         //序列化
126)         PlayerData playerData = new PlayerData ();
127)         string data = Js.Serialize(playerData);
128)         //写入数据库
129)         string sql = string.Format ("insert into player set id
='{0}' ,data ='{1}';", id, data);
130)         try
131)         {
132)             MySqlCommand cmd = new MySqlCommand (sql, mysql);
133)             cmd.ExecuteNonQuery ();
134)             return true;
135)         }
136)         catch (Exception e)
137)         {
138)             Console.WriteLine("[数据
库] CreatePlayer err, " + e.Message);
139)             return false;
140)         }
141)     }
142)

143)     //检测用户名密码
144)     public static bool CheckPassword(string id, string pw)
145)     {
146)         CheckAndReconnect();
147)         //防 sql 注入

```

```

148)         if(!DbManager.IsSafeString(id)){
149)             Console.WriteLine("[数据
           库] CheckPassword fail, id not safe");
150)             return false;
151)         }
152)         if(!DbManager.IsSafeString(pw)){
153)             Console.WriteLine("[数据
           库] CheckPassword fail, pw not safe");
154)             return false;
155)         }
156)         //查询
157)         string sql = string.Format("select * from account wher
           e id='{0}' and pw='{1}';", id, pw);
158)
159)         try
160)         {
161)             MySqlCommand cmd = new MySqlCommand (sql, mysql);
162)
163)             MySqlDataReader dataReader = cmd.ExecuteReader();
164)             bool hasRows = dataReader.HasRows;
165)             dataReader.Close();
166)             return hasRows;
167)         }
168)         catch(Exception e)
169)         {
170)             Console.WriteLine("[数据
           库] CheckPassword err, " + e.Message);
171)             return false;
172)         }
173)
174)         //获取玩家数据
175)         public static PlayerData GetPlayerData(string id)
176)         {
177)             CheckAndReconnect();
178)             //防 sql 注入
179)             if(!DbManager.IsSafeString(id)){
180)                 Console.WriteLine("[数据
           库] GetPlayerData fail, id not safe");
181)                 return null;
182)             }
183)
184)             //sql

```

```

185)         string sql = string.Format("select * from player where
            id='{0}';", id);
186)         try
187)         {
188)             //查询
189)             MySqlCommand cmd = new MySqlCommand (sql, mysql);
190)             MySqlDataReader dataReader = cmd.ExecuteReader();
191)             if(!dataReader.HasRows)
192)             {
193)                 dataReader.Close();
194)                 return null;
195)             }
196)             //读取
197)             dataReader.Read();
198)             string data = dataReader.GetString("data");
199)             //反序列化
200)             PlayerData playerData = Js.Deserialize<PlayerData>
            (data);
201)             dataReader.Close();
202)             return playerData;
203)         }
204)         catch(Exception e)
205)         {
206)             Console.WriteLine("[数据
            库] GetPlayerData fail, " + e.Message);
207)             return null;
208)         }
209)     }
210)
211)     //保存角色
212)     public static bool UpdatePlayerData(string id, PlayerData
            playerData)
213)     {
214)         CheckAndReconnect();
215)         //序列化
216)         string data = Js.Serialize(playerData);
217)         //sql
218)         string sql = string.Format("update player set data='{0
            }' where id='{1}';", data, id);
219)         //更新
220)         try
221)         {
222)             MySqlCommand cmd = new MySqlCommand (sql, mysql);
223)             cmd.ExecuteNonQuery ();

```

```

224)             return true;
225)         }
226)         catch (Exception e)
227)         {
228)             Console.WriteLine("[数据
           库] UpdatePlayerData err, " + e.Message);
229)             return false;
230)         }
231)     }
232) }

```

- 3D_Server\script\logic\BattleMsgHandle.cs

```

1) using System;
2)
3) public partial class MsgHandler {
4)     public static void MsgMove(ClientState c, MsgBase msgBase){
5)         //MsgMove msgMove = (MsgMove)msgBase;
6)         //Console.WriteLine(msgMove.x);
7)         //msgMove.x++;
8)         //NetManager.Send(c, msgMove);
9)     }
10) }

```

- 3D_Server\script\logic\EventHandler.cs

```

1) using System;
2)
3) public partial class EventHandler
4) {
5)     public static void OnDisconnect(ClientState c){
6)         Console.WriteLine("Close");
7)         //Player 下线
8)         if(c.player != null){
9)             //离开战场
10)            int roomId = c.player.roomId;
11)            if(roomId >= 0){
12)                Room room = RoomManager.GetRoom(roomId);
13)                room.RemovePlayer(c.player.id);
14)            }
15)            //保存数据
16)            DbManager.UpdatePlayerData(c.player.id, c.player.data)
           ;
17)            //移除
18)            PlayerManager.RemovePlayer(c.player.id);
19)        }
20)    }
21) }

```

```

22)     public static void OnTimer(){
23)         CheckPing();
24)         RoomManager.Update();
25)     }
26)
27)     //Ping 检查
28)     public static void CheckPing(){
29)         //现在的时间戳
30)         long timeNow = NetManager.GetTimeStamp();
31)         //遍历, 删除
32)         foreach(ClientState s in NetManager.clients.Values){
33)             if(timeNow - s.lastPingTime > NetManager.pingInterval*
4) 4){
34)                 Console.WriteLine("Ping Close " + s.socket.RemoteE
ndPoint.ToString());
35)                 NetManager.Close(s);
36)                 return;
37)             }
38)         }
39)     }
40) }

```

● 3D_Server\script\logic>LoginMsgHandle.cs

```

1) using System;
2)
3) public partial class MsgHandler {
4)
5)     //注册协议处理
6)     public static void MsgRegister(ClientState c, MsgBase msgBase)
{
7)         MsgRegister msg = (MsgRegister)msgBase;
8)         //注册
9)         if(DbManager.Register(msg.id, msg.pw)){
10)             DbManager.CreatePlayer(msg.id);
11)             msg.result = 0;
12)         }
13)         else{
14)             msg.result = 1;
15)         }
16)         NetManager.Send(c, msg);
17)     }
18)
19)     //登陆协议处理
20)     public static void MsgLogin(ClientState c, MsgBase msgBase){
21)         MsgLogin msg = (MsgLogin)msgBase;

```



```

22)         //密码校验
23)         if(!DbManager.CheckPassword(msg.id, msg.pw)){
24)             msg.result = 1;
25)             NetManager.Send(c, msg);
26)             return;
27)         }
28)         //不允许再次登陆
29)         if(c.player != null){
30)             msg.result = 1;
31)             NetManager.Send(c, msg);
32)             return;
33)         }
34)         //如果已经登陆, 踢下线
35)         if(PlayerManager.IsOnline(msg.id)){
36)             //发送踢下线协议
37)             Player other = PlayerManager.GetPlayer(msg.id);
38)             MsgKick msgKick = new MsgKick();
39)             msgKick.reason = 0;
40)             other.Send(msgKick);
41)             //断开连接
42)             NetManager.Close(other.state);
43)         }
44)         //获取玩家数据
45)         PlayerData playerData = DbManager.GetPlayerData(msg.id);
46)         if(playerData == null){
47)             msg.result = 1;
48)             NetManager.Send(c, msg);
49)             return;
50)         }
51)         //构建 Player
52)         Player player = new Player(c);
53)         player.id = msg.id;
54)         player.data = playerData;
55)         PlayerManager.AddPlayer(msg.id, player);
56)         c.player = player;
57)         //返回协议
58)         msg.result = 0;
59)         player.Send(msg);
60)     }
61) }

```

- 3D_Server\script\logic\NotepadMsgHandle.cs

```

1) public partial class MsgHandler {
2)
3)     //获取记事本内容

```

```

4)    public static void MsgGetText(ClientState c, MsgBase msgBase){
5)        MsgGetText msg = (MsgGetText)msgBase;
6)        Player player = c.player;
7)        if(player == null) return;
8)        //获取 text
9)        msg.text = player.data.text;
10)       player.Send(msg);
11)    }
12)
13)    //保存记事本内容
14)    public static void MsgSaveText(ClientState c, MsgBase msgBase)
    {
15)        MsgSaveText msg = (MsgSaveText)msgBase;
16)        Player player = c.player;
17)        if(player == null) return;
18)        //获取 text
19)        player.data.text = msg.text;
20)        player.Send(msg);
21)    }
22) }

```

● 3D_Server\script\logic\Player.cs

```

1) using System;
2)
3) public class Player {
4)     //id
5)     public string id = "";
6)     //指向 ClientState
7)     public ClientState state;
8)     //构造函数
9)     public Player(ClientState state){
10)        this.state = state;
11)    }
12)    //坐标和旋转
13)    public float x;
14)    public float y;
15)    public float z;
16)    public float ex;
17)    public float ey;
18)    public float ez;
19)
20)    //在哪个房间
21)    public int roomId = -1;
22)    //阵营
23)    public int camp = 1;

```

```

24)    //坦克生命值
25)    public int hp = 100;
26)
27)    //数据库数据
28)    public PlayerData data;
29)
30)    //发送信息
31)    public void Send(MsgBase msgBase){
32)        NetManager.Send(state, msgBase);
33)    }
34) }

```

- 3D_Server\script\logic\PlayerData.cs

```

1) public class PlayerData{
2)    //金币
3)    public int coin = 0;
4)    //记事本
5)    public string text = "new text";
6)    //胜利数
7)    public int win = 0;
8)    //失败数
9)    public int lost = 0;
10) }

```

- 3D_Server\script\logic\PlayerManager.cs

```

1) using System.Collections.Generic;
2)
3) public class PlayerManager
4) {
5)    //玩家列表
6)    static Dictionary<string, Player> players = new Dictionary<string, Player>();
7)    //玩家是否在线
8)    public static bool IsOnline(string id){
9)        return players.ContainsKey(id);
10)    }
11)    //获取玩家
12)    public static Player GetPlayer(string id){
13)        if(players.ContainsKey(id)){
14)            return players[id];
15)        }
16)        return null;
17)    }
18)    //添加玩家
19)    public static void AddPlayer(string id, Player player){
20)        players.Add(id, player);

```

```

21)    }
22)    //删除玩家
23)    public static void RemovePlayer(string id){
24)        players.Remove(id);
25)    }
26) }

```

- 3D_Server\script\logic\Room.cs

```

1) using System;
2) using System.Collections.Generic;
3)
4) public class Room {
5)     //id
6)     public int id = 0;
7)     //最大玩家数
8)     public int maxPlayer = 6;
9)     //玩家列表
10)    public Dictionary<string, bool> playerIds = new Dictionary<string, bool>();
11)    //房主 id
12)    public string ownerId = "";
13)    //状态
14)    public enum Status {
15)        PREPARE = 0,
16)        FIGHT = 1 ,
17)    }
18)    public Status status = Status.PREPARE;
19)    //出生点位置配置
20)    static float[, ] birthConfig = new float[2, 3, 6] {
21)        //阵营 1 出生点
22)        {
23)            {262.3f, -8.0f, 342.7f, 0, -151.0f, 0f},//出生点 1
24)            {229.7f, -5.5f, 354.4f, 0, -164.2f, 0f},//出生点 2
25)            {197.1f, -3.6f, 347.7f, 0, -193.0f, 0f},//出生点 3
26)        },
27)        //阵营 2 出生点
28)        {
29)            {-80.3f, 9.5f, 114.6f, 0, -294.0f, 0f},//出生点 1
30)            {-91.1f, 15.5f, 139.1f, 0, -294.2f, 0f},//出生点 2
31)            {-62.3f, 1.2f, 76.1f, 0, -315.4f, 0f},//出生点 3
32)        },
33)    };
34)    //上一次判断结果的时间
35)    private long lastjudgeTime = 0;
36)

```

```

37)    //添加玩家
38)    public bool AddPlayer(string id){
39)        //获取玩家
40)        Player player = PlayerManager.GetPlayer(id);
41)        if(player == null){
42)            Console.WriteLine("room.AddPlayer fail, player is null
    ");
43)            return false;
44)        }
45)        //房间人数
46)        if(playerIds.Count >= maxPlayer){
47)            Console.WriteLine("room.AddPlayer fail, reach maxPlaye
r");
48)            return false;
49)        }
50)        //准备状态才能加人
51)        if(status != Status.PREPARE){
52)            Console.WriteLine("room.AddPlayer fail, not PREPARE");
53)            return false;
54)        }
55)        //已经在房间里
56)        if(playerIds.ContainsKey(id)){
57)            Console.WriteLine("room.AddPlayer fail, already in thi
s room");
58)            return false;
59)        }
60)        //加入列表
61)        playerIds[id] = true;
62)        //设置玩家数据
63)        player.camp = SwitchCamp();
64)        player.roomId = this.id;
65)        //设置房主
66)        if(ownerId == ""){
67)            ownerId = player.id;
68)        }
69)        //广播
70)        Broadcast(ToMsg());
71)        return true;
72)    }
73)
74)    //分配阵营
75)    public int SwitchCamp() {
76)        //计数
77)        int count1 = 0;

```

```

78)         int count2 = 0;
79)         foreach(string id in playerIds.Keys) {
80)             Player player = PlayerManager.GetPlayer(id);
81)             if(player.camp == 1) {count1++;}
82)             if(player.camp == 2) {count2++;}
83)         }
84)         //选择
85)         if (count1 <= count2){
86)             return 1;
87)         }
88)         else{
89)             return 2;
90)         }
91)     }
92)
93)     //是不是房主
94)     public bool isOwner(Player player){
95)         return player.id == ownerId;
96)     }
97)
98)     //删除玩家
99)     public bool RemovePlayer(string id) {
100)         //获取玩家
101)         Player player = PlayerManager.GetPlayer(id);
102)         if(player == null){
103)             Console.WriteLine("room.RemovePlayer fail, player
is null");
104)             return false;
105)         }
106)         //没有在房间里
107)         if(!playerIds.ContainsKey(id)){
108)             Console.WriteLine("room.RemovePlayer fail, not in
this room");
109)             return false;
110)         }
111)         //删除列表
112)         playerIds.Remove(id);
113)         //设置玩家数据
114)         player.camp = 0;
115)         player.roomId = -1;
116)         //设置房主
117)         if(ownerId == player.id){
118)             ownerId = SwitchOwner();
119)         }

```

```

120)         //战斗状态退出
121)         if(status == Status.FIGHT){
122)             player.data.lost++;
123)             MsgLeaveBattle msg = new MsgLeaveBattle();
124)             msg.id = player.id;
125)             Broadcast(msg);
126)         }
127)         //房间为空
128)         if(playerIds.Count == 0){
129)             RoomManager.RemoveRoom(this.id);
130)         }
131)         //广播
132)         Broadcast(ToMsg());
133)         return true;
134)     }
135)
136)     //选择房主
137)     public string SwitchOwner() {
138)         //选择第一个玩家
139)         foreach(string id in playerIds.Keys) {
140)             return id;
141)         }
142)         //房间没人
143)         return "";
144)     }
145)
146)     //广播消息
147)     public void Broadcast(MsgBase msg){
148)         foreach(string id in playerIds.Keys) {
149)             Player player = PlayerManager.GetPlayer(id);
150)             player.Send(msg);
151)         }
152)     }
153)
154)     //生成 MsgGetRoomInfo 协议
155)     public MsgBase ToMsg(){
156)         MsgGetRoomInfo msg = new MsgGetRoomInfo();
157)         int count = playerIds.Count;
158)         msg.players = new PlayerInfo[count];
159)         //players
160)         int i = 0;
161)         foreach(string id in playerIds.Keys){
162)             Player player = PlayerManager.GetPlayer(id);
163)             PlayerInfo playerInfo = new PlayerInfo();

```



```

164)         //赋值
165)         playerInfo.id = player.id;
166)         playerInfo.camp = player.camp;
167)         playerInfo.win = player.data.win;
168)         playerInfo.lost = player.data.lost;
169)         playerInfo.isOwner = 0;
170)         if(isOwner(player)){
171)             playerInfo.isOwner = 1;
172)         }
173)
174)         msg.players[i] = playerInfo;
175)         i++;
176)     }
177)     return msg;
178) }
179)
180) //能否开战
181) public bool CanStartBattle() {
182)     //已经是战斗状态
183)     if (status != Status.PREPARE){
184)         return false;
185)     }
186)     //统计每个队伍的玩家数
187)     int count1 = 0;
188)     int count2 = 0;
189)     foreach(string id in playerIds.Keys) {
190)         Player player = PlayerManager.GetPlayer(id);
191)         if(player.camp == 1){ count1++; }
192)         else { count2++; }
193)     }
194)     //每个队伍至少要有 1 名玩家
195)     if (count1 < 1 || count2 < 1){
196)         return false;
197)     }
198)     return true;
199) }
200)
201) //初始化位置
202) private void SetBirthPos(Player player, int index){
203)     int camp = player.camp;
204)
205)     player.x = birthConfig[camp-1, index,0];
206)     player.y = birthConfig[camp-1, index,1];
207)     player.z = birthConfig[camp-1, index,2];

```

```

208)         player.ex = birthConfig[camp-1, index,3];
209)         player.ey = birthConfig[camp-1, index,4];
210)         player.ez = birthConfig[camp-1, index,5];
211)     }
212)
213)     //玩家数据转成 TankInfo
214)     public TankInfo PlayerToTankInfo(Player player){
215)         TankInfo tankInfo = new TankInfo();
216)         tankInfo.camp = player.camp;
217)         tankInfo.id = player.id;
218)         tankInfo.hp = player.hp;
219)
220)         tankInfo.x = player.x;
221)         tankInfo.y = player.y;
222)         tankInfo.z = player.z;
223)         tankInfo.ex = player.ex;
224)         tankInfo.ey = player.ey;
225)         tankInfo.ez = player.ez;
226)
227)         return tankInfo;
228)     }
229)
230)     //重置玩家战斗属性
231)     private void ResetPlayers(){
232)         //位置和旋转
233)         int count1 = 0;
234)         int count2 = 0;
235)         foreach(string id in playerIds.Keys) {
236)             Player player = PlayerManager.GetPlayer(id);
237)             if(player.camp == 1){
238)                 SetBirthPos(player, count1);
239)                 count1++;
240)             }
241)             else {
242)                 SetBirthPos(player, count2);
243)                 count2++;
244)             }
245)         }
246)         //生命值
247)         foreach(string id in playerIds.Keys) {
248)             Player player = PlayerManager.GetPlayer(id);
249)             player.hp = 100;
250)         }
251)     }

```

```

252)
253) //开战
254) public bool StartBattle() {
255)     if(!CanStartBattle()){
256)         return false;
257)     }
258)     //状态
259)     status = Status.FIGHT;
260)     //玩家战斗属性
261)     ResetPlayers();
262)     //返回数据
263)     MsgEnterBattle msg = new MsgEnterBattle();
264)     msg.mapId = 1;
265)     msg.tanks = new TankInfo[playerIds.Count];
266)
267)     int i=0;
268)     foreach(string id in playerIds.Keys) {
269)         Player player = PlayerManager.GetPlayer(id);
270)         msg.tanks[i] = PlayerToTankInfo(player);
271)         i++;
272)     }
273)     Broadcast(msg);
274)     return true;
275) }
276)
277) //是否死亡
278) public bool IsDie(Player player){
279)     return player.hp <= 0;
280) }
281)
282) //定时更新
283) public void Update(){
284)     //状态判断
285)     if(status != Status.FIGHT){
286)         return;
287)     }
288)     //时间判断
289)     if(NetManager.GetTimeStamp() - lastjudgeTime < 10f){
290)         return;
291)     }
292)     lastjudgeTime = NetManager.GetTimeStamp();
293)     //胜负判断
294)     int winCamp = Judgment();

```

```

295)         //尚未分出胜负
296)         if(winCamp == 0){
297)             return;
298)         }
299)         //某一方胜利，结束战斗
300)         status = Status.PREPARE;
301)         //统计信息
302)         foreach(string id in playerIds.Keys) {
303)             Player player = PlayerManager.GetPlayer(id);
304)             if(player.camp == winCamp){player.data.win++;}
305)             else{player.data.lost++;}
306)         }
307)         //发送 Result
308)         MsgBattleResult msg = new MsgBattleResult();
309)         msg.winCamp = winCamp;
310)         Broadcast(msg);
311)     }
312)
313)     //胜负判断
314)     public int Judgment(){
315)         //存活人数
316)         int count1 = 0;
317)         int count2 = 0;
318)         foreach(string id in playerIds.Keys) {
319)             Player player = PlayerManager.GetPlayer(id);
320)             if(!IsDie(player)){
321)                 if(player.camp == 1){count1++;}
322)                 if(player.camp == 2){count2++;}
323)             }
324)         }
325)         //判断
326)         if(count1 <= 0){
327)             return 2;
328)         }
329)         else if(count2 <= 0){
330)             return 1;
331)         }
332)         return 0;
333)     }
334) }

```

- 3D_Server\script\logic\RoomManager.cs
 - 1) using System.Collections.Generic;
 - 2)
 - 3) public class RoomManager

```

4) {
5)     //最大 id
6)     private static int maxId = 1;
7)     //房间列表
8)     public static Dictionary<int, Room> rooms = new Dictionary<int
, Room>();
9)
10)    //创建房间
11)    public static Room AddRoom(){
12)        maxId++;
13)        Room room = new Room();
14)        room.id = maxId;
15)        rooms.Add(room.id, room);
16)        return room;
17)    }
18)
19)    //删除房间
20)    public static bool RemoveRoom(int id){
21)        rooms.Remove(id);
22)        return true;
23)    }
24)
25)    //获取房间
26)    public static Room GetRoom(int id) {
27)        if(rooms.ContainsKey(id)){
28)            return rooms[id];
29)        }
30)        return null;
31)    }
32)
33)    //生成 MsgGetRoomList 协议
34)    public static MsgBase ToMsg(){
35)        MsgGetRoomList msg = new MsgGetRoomList();
36)        int count = rooms.Count;
37)        msg.rooms = new RoomInfo[count];
38)        //rooms
39)        int i = 0;
40)        foreach(Room room in rooms.Values){
41)            RoomInfo roomInfo = new RoomInfo();
42)            //赋值
43)            roomInfo.id = room.id;
44)            roomInfo.count = room.playerIds.Count;
45)            roomInfo.status = (int)room.status;
46)

```

```

47)         msg.rooms[i] = roomInfo;
48)         i++;
49)     }
50)     return msg;
51) }
52)
53) //Update
54) public static void Update(){
55)     foreach(Room room in rooms.Values){
56)         room.Update();
57)     }
58) }
59) }

```

- 3D_Server\script\logic\RoomMsgHandle.cs

```

1) public partial class MsgHandler {
2)
3)     //查询战绩
4)     public static void MsgGetAchieve(ClientState c, MsgBase msgBas
e){
5)         MsgGetAchieve msg = (MsgGetAchieve)msgBase;
6)         Player player = c.player;
7)         if(player == null) return;
8)
9)         msg.win = player.data.win;
10)        msg.lost = player.data.lost;
11)
12)        player.Send(msg);
13)    }
14)
15)    //请求房间列表
16)    public static void MsgGetRoomList(ClientState c, MsgBase msgBa
se){
17)        MsgGetRoomList msg = (MsgGetRoomList)msgBase;
18)        Player player = c.player;
19)        if(player == null) return;
20)
21)        player.Send(RoomManager.ToMsg());
22)    }
23)
24)    //创建房间
25)    public static void MsgCreateRoom(ClientState c, MsgBase msgBas
e){
26)        MsgCreateRoom msg = (MsgCreateRoom)msgBase;
27)        Player player = c.player;

```

```

28)         if(player == null) return;
29)         //已经在房间里
30)         if(player.roomId >=0 ){
31)             msg.result = 1;
32)             player.Send(msg);
33)             return;
34)         }
35)         //创建
36)         Room room = RoomManager.AddRoom();
37)         room.AddPlayer(player.id);
38)
39)         msg.result = 0;
40)         player.Send(msg);
41)     }
42)
43)     //进入房间
44)     public static void MsgEnterRoom(ClientState c, MsgBase msgBase
    ){
45)         MsgEnterRoom msg = (MsgEnterRoom)msgBase;
46)         Player player = c.player;
47)         if(player == null) return;
48)         //已经在房间里
49)         if(player.roomId >=0 ){
50)             msg.result = 1;
51)             player.Send(msg);
52)             return;
53)         }
54)         //获取房间
55)         Room room = RoomManager.GetRoom(msg.id);
56)         if(room == null){
57)             msg.result = 1;
58)             player.Send(msg);
59)             return;
60)         }
61)         //进入
62)         if(!room.AddPlayer(player.id)){
63)             msg.result = 1;
64)             player.Send(msg);
65)             return;
66)         }
67)         //返回协议
68)         msg.result = 0;
69)         player.Send(msg);
70)     }

```



```

71)
72)    //获取房间信息
73)    public static void MsgGetRoomInfo(ClientState c, MsgBase msgBa
    se){
74)        MsgGetRoomInfo msg = (MsgGetRoomInfo)msgBase;
75)        Player player = c.player;
76)        if(player == null) return;
77)
78)        Room room = RoomManager.GetRoom(player.roomId);
79)        if(room == null){
80)            player.Send(msg);
81)            return;
82)        }
83)        player.Send(room.ToMsg());
84)    }
85)
86)    //离开房间
87)    public static void MsgLeaveRoom(ClientState c, MsgBase msgBase
    ){
88)        MsgLeaveRoom msg = (MsgLeaveRoom)msgBase;
89)        Player player = c.player;
90)        if(player == null) return;
91)
92)        Room room = RoomManager.GetRoom(player.roomId);
93)        if(room == null){
94)            msg.result = 1;
95)            player.Send(msg);
96)            return;
97)        }
98)
99)        room.RemovePlayer(player.id);
100)        //返回协议
101)        msg.result = 0;
102)        player.Send(msg);
103)    }
104)
105)    //请求开始战斗
106)    public static void MsgStartBattle(ClientState c, MsgBase m
    sgBase){
107)        MsgStartBattle msg = (MsgStartBattle)msgBase;
108)        Player player = c.player;
109)        if(player == null) return;
110)        //room
111)        Room room = RoomManager.GetRoom(player.roomId);

```

```

112)         if(room == null){
113)             msg.result = 1;
114)             player.Send(msg);
115)             return;
116)         }
117)         //是否是房主
118)         if(!room.isOwner(player)){
119)             msg.result = 1;
120)             player.Send(msg);
121)             return;
122)         }
123)         //开战
124)         if(!room.StartBattle()){
125)             msg.result = 1;
126)             player.Send(msg);
127)             return;
128)         }
129)         //成功
130)         msg.result = 0;
131)         player.Send(msg);
132)     }
133) }

```

● 3D_Server\script\logic\SyncMsgHandle.cs

```

1) using System;
2)
3) public partial class MsgHandler {
4)
5)     //同步位置协议
6)     public static void MsgSyncTank(ClientState c, MsgBase msgBase)
7)     {
8)         MsgSyncTank msg = (MsgSyncTank)msgBase;
9)         Player player = c.player;
10)        if(player == null) return;
11)        //room
12)        Room room = RoomManager.GetRoom(player.roomId);
13)        if(room == null){
14)            return;
15)        }
16)        //status
17)        if(room.status != Room.Status.FIGHT){
18)            return;
19)        }
20)        //是否作弊
21)        if(Math.Abs(player.x - msg.x) > 5 ||

```

```

21)         Math.Abs(player.y - msg.y) > 5 ||
22)         Math.Abs(player.z - msg.z) > 5){
23)             Console.WriteLine("疑似作弊 " + player.id);
24)         }
25)         //更新信息
26)         player.x = msg.x;
27)         player.y = msg.y;
28)         player.z = msg.z;
29)         player.ex = msg.ex;
30)         player.ey = msg.ey;
31)         player.ez = msg.ez;
32)         //广播
33)         msg.id = player.id;
34)         room.Broadcast(msg);
35)     }
36)
37)     //开火协议
38)     public static void MsgFire(ClientState c, MsgBase msgBase){
39)         MsgFire msg = (MsgFire)msgBase;
40)         Player player = c.player;
41)         if(player == null) return;
42)         //room
43)         Room room = RoomManager.GetRoom(player.roomId);
44)         if(room == null){
45)             return;
46)         }
47)         //status
48)         if(room.status != Room.Status.FIGHT){
49)             return;
50)         }
51)         //广播
52)         msg.id = player.id;
53)         room.Broadcast(msg);
54)     }
55)
56)     //击中协议
57)     public static void MsgHit(ClientState c, MsgBase msgBase){
58)         MsgHit msg = (MsgHit)msgBase;
59)         Player player = c.player;
60)         if(player == null) return;
61)         //targetPlayer
62)         Player targetPlayer = PlayerManager.GetPlayer(msg.targetId
63)     );
        if(targetPlayer == null){

```

```

64)         return;
65)     }
66)     //room
67)     Room room = RoomManager.GetRoom(player.roomId);
68)     if(room == null){
69)         return;
70)     }
71)     //status
72)     if(room.status != Room.Status.FIGHT){
73)         return;
74)     }
75)     //发送者校验
76)     if(player.id != msg.id){
77)         return;
78)     }
79)     //状态
80)     int damage = 35;
81)     targetPlayer.hp -= damage;
82)     //广播
83)     msg.id = player.id;
84)     msg.hp = player.hp;
85)     msg.damage = damage;
86)     room.Broadcast(msg);
87) }
88) }

```

- 3D_Server\script\logic\SysMsgHandler.cs

```

1) using System;
2)
3) public partial class MsgHandler {
4)     public static void MsgPing(ClientState c, MsgBase msgBase){
5)         Console.WriteLine("MsgPing");
6)         c.lastPingTime = NetManager.GetTimeStamp();
7)         MsgPong msgPong = new MsgPong();
8)         NetManager.Send(c, msgPong);
9)     }
10) }

```

- 3D_Server\script\net\ByteArray.cs

```

1) using System;
2)
3) public class ByteArray {
4)     //默认大小
5)     const int DEFAULT_SIZE = 1024;
6)     //初始大小
7)     int initSize = 0;

```

```

8)    //缓冲区
9)    public byte[] bytes;
10)   //读写位置
11)   public int readIdx = 0;
12)   public int writeIdx = 0;
13)   //容量
14)   private int capacity = 0;
15)   //剩余空间
16)   public int remain { get { return capacity-writeIdx; }}
17)   //数据长度
18)   public int length { get { return writeIdx-readIdx; }}
19)
20)   //构造函数
21)   public ByteArray(int size = DEFAULT_SIZE){
22)       bytes = new byte[size];
23)       capacity = size;
24)       initSize = size;
25)       readIdx = 0;
26)       writeIdx = 0;
27)   }
28)
29)   //构造函数
30)   public ByteArray(byte[] defaultBytes){
31)       bytes = defaultBytes;
32)       capacity = defaultBytes.Length;
33)       initSize = defaultBytes.Length;
34)       readIdx = 0;
35)       writeIdx = defaultBytes.Length;
36)   }
37)
38)   //重设尺寸
39)   public void ReSize(int size){
40)       if(size < length) return;
41)       if(size < initSize) return;
42)       int n = 1;
43)       while(n<size) n*=2;
44)       capacity = n;
45)       byte[] newBytes = new byte[capacity];
46)       Array.Copy(bytes, readIdx, newBytes, 0, writeIdx-readIdx);
47)       bytes = newBytes;
48)       writeIdx = length;
49)       readIdx = 0;
50)   }
51)

```

```

52) //写入数据
53) public int Write(byte[] bs, int offset, int count){
54)     if(remain < count){
55)         ReSize(length + count);
56)     }
57)     Array.Copy(bs, offset, bytes, writeIdx, count);
58)     writeIdx+=count;
59)     return count;
60) }
61)
62) //读取数据
63) public int Read(byte[] bs, int offset, int count){
64)     count = Math.Min(count, length);
65)     Array.Copy(bytes, 0, bs, offset, count);
66)     readIdx+=count;
67)     CheckAndMoveBytes();
68)     return count;
69) }
70)
71) //检查并移动数据
72) public void CheckAndMoveBytes(){
73)     if(length < 8){
74)         MoveBytes();
75)     }
76) }
77)
78) //移动数据
79) public void MoveBytes(){
80)     if(length > 0) {
81)         Array.Copy(bytes, readIdx, bytes, 0, length);
82)     }
83)     writeIdx = length;
84)     readIdx = 0;
85) }
86)
87) //打印缓冲区
88) public override string ToString(){
89)     return BitConverter.ToString(bytes, readIdx, length);
90) }
91)
92) //打印调试信息
93) public string Debug(){
94)     return string.Format("readIdx({0}) writeIdx({1}) bytes({2}
)",

```

```

95)         readIdx,
96)         writeIdx,
97)         BitConverter.ToString(bytes, 0, capacity)
98)     );
99) }
100) }

```

- 3D_Server\script\net\ClientState.cs

```

1) using System.Net.Sockets;
2)
3) public class ClientState
4) {
5)     public Socket socket;
6)     public ByteArray readBuff = new ByteArray();
7)     //Ping
8)     public long lastPingTime = 0;
9)     //玩家
10)    public Player player;
11) }

```

- 3D_Server\script\net\MsgBase.cs

```

1) using System;
2) using System.Web.Script.Serialization;
3)
4) public class MsgBase{
5)     public string protoName = "null";
6)
7)     //编码器
8)     static JavaScriptSerializer Js = new JavaScriptSerializer();
9)
10)    //编码
11)    public static byte[] Encode(MsgBase msgBase){
12)        string s = Js.Serialize(msgBase);
13)        return System.Text.Encoding.UTF8.GetBytes(s);
14)    }
15)
16)    //解码
17)    public static MsgBase Decode(string protoName, byte[] bytes, int offset, int count){
18)        string s = System.Text.Encoding.UTF8.GetString(bytes, offset, count);
19)        MsgBase msgBase = (MsgBase)Js.Deserialize(s, Type.GetType(protoName));
20)        return msgBase;
21)    }
22)

```

```

23)    //编码协议名 (2 字节长度+字符串)
24)    public static byte[] EncodeName(MsgBase msgBase){
25)        //名字 bytes 和长度
26)        byte[] nameBytes = System.Text.Encoding.UTF8.GetBytes(msgB
ase.protoName);
27)        Int16 len = (Int16)nameBytes.Length;
28)        //申请 bytes 数值
29)        byte[] bytes = new byte[2+len];
30)        //组装 2 字节的长度信息
31)        bytes[0] = (byte)(len%256);
32)        bytes[1] = (byte)(len/256);
33)        //组装名字 bytes
34)        Array.Copy(nameBytes, 0, bytes, 2, len);
35)
36)        return bytes;
37)    }
38)
39)    //解码协议名 (2 字节长度+字符串)
40)    public static string DecodeName(byte[] bytes, int offset, out
int count){
41)        count = 0;
42)        //必须大于 2 字节
43)        if(offset + 2 > bytes.Length){
44)            return "";
45)        }
46)        //读取长度
47)        Int16 len = (Int16)((bytes[offset+1] << 8 )| bytes[offset]
);
48)        if(len <= 0){
49)            return "";
50)        }
51)        //长度必须足够
52)        if(offset + 2 + len > bytes.Length){
53)            return "";
54)        }
55)        //解析
56)        count = 2+len;
57)        string name = System.Text.Encoding.UTF8.GetString(bytes, o
ffset+2, len);
58)        return name;
59)    }
60) }

```

● 3D_Server\script\net\NetManager.cs

```

1) using System;

```



```

2) using System.Net;
3) using System.Net.Sockets;
4) using System.Collections.Generic;
5) using System.Reflection;
6)
7) class NetManager
8) {
9)     //监听 Socket
10)    public static Socket listenfd;
11)    //客户端 Socket 及状态信息
12)    public static Dictionary<Socket, ClientState> clients = new Dictionary<Socket, ClientState>();
13)    //Select 的检查列表
14)    static List<Socket> checkRead = new List<Socket>();
15)    //ping 间隔
16)    public static long pingInterval = 30;
17)
18)    public static void StartLoop(int listenPort)
19)    {
20)        //Socket
21)        listenfd = new Socket(AddressFamily.InterNetwork,
22)                               SocketType.Stream, ProtocolType.Tcp);
23)        //Bind
24)        IPAddress ipAdr = IPAddress.Parse("0.0.0.0");
25)        IPEndPoint ipEp = new IPEndPoint(ipAdr, listenPort);
26)        listenfd.Bind(ipEp);
27)        //Listen
28)        listenfd.Listen(0);
29)        Console.WriteLine("[服务器]启动成功");
30)        //循环
31)        while(true){
32)            ResetCheckRead(); //重置 checkRead
33)            Socket.Select(checkRead, null, null, 1000);
34)            //检查可读对象
35)            for(int i = checkRead.Count-1; i>=0; i--){
36)                Socket s = checkRead[i];
37)                if(s == listenfd){
38)                    ReadListenfd(s);
39)                }
40)                else{
41)                    ReadClientfd(s);
42)                }
43)            }
44)            //超时

```

```

45)         Timer();
46)     }
47) }
48)
49) //填充 checkRead 列表
50) public static void ResetCheckRead(){
51)     checkRead.Clear();
52)     checkRead.Add(listenfd);
53)     foreach (ClientState s in clients.Values){
54)         checkRead.Add(s.socket);
55)     }
56) }
57)
58) //读取 Listenfd
59) public static void ReadListenfd(Socket listenfd){
60)     try{
61)         Socket clientfd = listenfd.Accept();
62)         Console.WriteLine("Accept " + clientfd.RemoteEndPoint.
ToString());
63)         ClientState state = new ClientState();
64)         state.socket = clientfd;
65)         state.lastPingTime = GetTimeStamp();
66)         clients.Add(clientfd, state);
67)     }catch(SocketException ex){
68)         Console.WriteLine("Accept fail" + ex.ToString());
69)     }
70) }
71)
72) //关闭连接
73) public static void Close(ClientState state){
74)     //消息分发
75)     MethodInfo mei = typeof(EventHandler).GetMethod("OnDiscon
nect");
76)     object[] ob = {state};
77)     mei.Invoke(null, ob);
78)     //关闭
79)     state.socket.Close();
80)     clients.Remove(state.socket);
81) }
82)
83) //读取 Clientfd
84) public static void ReadClientfd(Socket clientfd){
85)     ClientState state = clients[clientfd];
86)     ByteArray readBuff = state.readBuff;

```

```

87)         //接收
88)         int count = 0;
89)         //缓冲区不够，清除，若依旧不够，只能返回
90)         //当单条协议超过缓冲区长度时会发生
91)         if(readBuff.remain <=0){
92)             OnReceiveData(state);
93)             readBuff.MoveBytes();
94)         };
95)         if(readBuff.remain <=0){
96)             Console.WriteLine("Receive fail , maybe msg length > buff capacity");
97)             Close(state);
98)             return;
99)         }
100)        try{
101)            count = clientfd.Receive(readBuff.bytes, readBuff.writeIdx, readBuff.remain, 0);
102)        }catch(SocketException ex){
103)            Console.WriteLine("Receive SocketException " + ex.ToString());
104)            Close(state);
105)            return;
106)        }
107)        //客户端关闭
108)        if(count <= 0 ){
109)            Console.WriteLine("Socket Close " + clientfd.RemoteEndPoint.ToString());
110)            Close(state);
111)            return;
112)        }
113)        //消息处理
114)        readBuff.writeIdx+=count;
115)        //处理二进制消息
116)        OnReceiveData(state);
117)        //移动缓冲区
118)        readBuff.CheckAndMoveBytes();
119)    }
120)
121)    //数据处理
122)    public static void OnReceiveData(ClientState state){
123)        ByteArray readBuff = state.readBuff;
124)        //消息长度
125)        if(readBuff.length <= 2) {
126)            return;

```

```

127)        }
128)        //消息体长度
129)        int readIdx = readBuff.readIdx;
130)        byte[] bytes =readBuff.bytes;
131)        Int16 bodyLength = (Int16)((bytes[readIdx+1] << 8 )| b
        ytes[readIdx]);
132)        if(readBuff.length < bodyLength){
133)            return;
134)        }
135)        readBuff.readIdx +=2;
136)        //解析协议名
137)        int nameCount = 0;
138)        string protoName = MsgBase.DecodeName(readBuff.bytes,
        readBuff.readIdx, out nameCount);
139)        if(protoName == ""){
140)            Console.WriteLine("OnReceiveData MsgBase.DecodeNam
        e fail");
141)            Close(state);
142)            return;
143)        }
144)        readBuff.readIdx += nameCount;
145)        //解析协议体
146)        int bodyCount = bodyLength - nameCount;
147)        if(bodyCount <= 0){
148)            Console.WriteLine("OnReceiveData fail, bodyCount <
        =0 ");
149)            Close(state);
150)            return;
151)        }
152)        MsgBase msgBase = MsgBase.Decode(protoName, readBuff.b
        ytes, readBuff.readIdx, bodyCount);
153)        readBuff.readIdx += bodyCount;
154)        readBuff.CheckAndMoveBytes();
155)        //分发消息
156)        MethodInfo mi = typeof(MsgHandler).GetMethod(protoNam
        e);
157)        object[] o = {state, msgBase};
158)        Console.WriteLine("Receive " + protoName);
159)        if(mi != null){
160)            mi.Invoke(null, o);
161)        }
162)        else{
163)            Console.WriteLine("OnReceiveData Invoke fail " + p
        rotoName);

```

```

164)         }
165)         //继续读取消息
166)         if(readBuff.length > 2){
167)             OnReceiveData(state);
168)         }
169)     }
170)
171)     //发送
172)     public static void Send(ClientState cs, MsgBase msg){
173)         //状态判断
174)         if(cs == null){
175)             return;
176)         }
177)         if(!cs.socket.Connected){
178)             return;
179)         }
180)         //数据编码
181)         byte[] nameBytes = MsgBase.EncodeName(msg);
182)         byte[] bodyBytes = MsgBase.Encode(msg);
183)         int len = nameBytes.Length + bodyBytes.Length;
184)         byte[] sendBytes = new byte[2+len];
185)         //组装长度
186)         sendBytes[0] = (byte)(len%256);
187)         sendBytes[1] = (byte)(len/256);
188)         //组装名字
189)         Array.Copy(nameBytes, 0, sendBytes, 2, nameBytes.Length);
190)         //组装消息体
191)         Array.Copy(bodyBytes, 0, sendBytes, 2+nameBytes.Length, bodyBytes.Length);
192)         //为简化代码, 不设置回调
193)         try{
194)             cs.socket.BeginSend(sendBytes, 0, sendBytes.Length, 0, null, null);
195)         }catch(SocketException ex){
196)             Console.WriteLine("Socket Close on BeginSend" + ex.ToString());
197)         }
198)     }
199)
200)     //定时器
201)     static void Timer(){
202)         //消息分发

```

```

203)         MethodInfo mei = typeof(EventHandler).GetMethod("OnTi
            mer");
204)         object[] ob = {};
205)         mei.Invoke(null, ob);
206)     }
207)
208)         //获取时间戳
209)         public static long GetTimeStamp() {
210)             TimeSpan ts = DateTime.UtcNow - new DateTime(1970, 1,
                1, 0, 0, 0, 0);
211)             return Convert.ToInt64(ts.TotalSeconds);
212)         }
213)     }

```

● 3D_Server\script\proto\BattleMsg.cs

```

1) //坦克信息
2) [System.Serializable]
3) public class TankInfo{
4)     public string id = ""; //玩家 id
5)     public int camp = 0; //阵营
6)     public int hp = 0; //生命值
7)
8)     public float x = 0; //位置
9)     public float y = 0;
10)    public float z = 0;
11)    public float ex = 0; //旋转
12)    public float ey = 0;
13)    public float ez = 0;
14) }
15)
16) //进入战场（服务端推送）
17) public class MsgEnterBattle:MsgBase {
18)     public MsgEnterBattle() {protoName = "MsgEnterBattle";}
19)     //服务端回
20)     public TankInfo[] tanks;
21)     public int mapId = 1; //地图，只有一张
22) }
23)
24) //战斗结果（服务端推送）
25) public class MsgBattleResult:MsgBase {
26)     public MsgBattleResult() {protoName = "MsgBattleResult";}
27)     //服务端回
28)     public int winCamp = 0; //获胜的阵营
29) }
30)

```

```

31) //玩家退出（服务端推送）
32) public class MsgLeaveBattle:MsgBase {
33)     public MsgLeaveBattle() {protoName = "MsgLeaveBattle";}
34)     //服务端回
35)     public string id = ""; //玩家 id
36) }
● 3D_Server\script\proto>LoginMsg.cs
1) //注册
2) public class MsgRegister:MsgBase {
3)     public MsgRegister() {protoName = "MsgRegister";}
4)     //客户端发
5)     public string id = "";
6)     public string pw = "";
7)     //服务端回（0-成功，1-失败）
8)     public int result = 0;
9) }
10)
11) //登陆
12) public class MsgLogin:MsgBase {
13)     public MsgLogin() {protoName = "MsgLogin";}
14)     //客户端发
15)     public string id = "";
16)     public string pw = "";
17)     //服务端回（0-成功，1-失败）
18)     public int result = 0;
19) }
20)
21) //踢下线（服务端推送）
22) public class MsgKick:MsgBase {
23)     public MsgKick() {protoName = "MsgKick";}
24)     //原因（0-其他人登陆同一账号）
25)     public int reason = 0;
26) }
● 3D_Server\script\proto\notepadMsg.cs
1) //获取记事本内容
2) public class MsgGetText:MsgBase {
3)     public MsgGetText() {protoName = "MsgGetText";}
4)     //服务端回
5)     public string text = "";
6) }
7)
8) //保存记事本内容
9) public class MsgSaveText:MsgBase {
10)     public MsgSaveText() {protoName = "MsgSaveText";}

```

```

11)    //客户端发
12)    public string text = "";
13)    //服务端回 (0-成功 1-文字太长)
14)    public int result = 0;
15) }
● 3D_Server\script\proto\RoomMsg.cs
1)    //查询成绩
2)    public class MsgGetAchieve:MsgBase {
3)        public MsgGetAchieve() {protoName = "MsgGetAchieve";}
4)        //服务端回
5)        public int win = 0;
6)        public int lost = 0;
7)    }
8)
9)    //房间信息
10)   public class RoomInfo{
11)       public int id = 0;        //房间 id
12)       public int count = 0;    //人数
13)       public int status = 0;   //状态 0-准备中 1-战斗中
14)   }
15)
16) //请求房间列表
17) public class MsgGetRoomList:MsgBase {
18)     public MsgGetRoomList() {protoName = "MsgGetRoomList";}
19)     //服务端回
20)     public RoomInfo[] rooms;
21) }
22)
23) //创建房间
24) public class MsgCreateRoom:MsgBase {
25)     public MsgCreateRoom() {protoName = "MsgCreateRoom";}
26)     //服务端回
27)     public int result = 0;
28) }
29)
30) //进入房间
31) public class MsgEnterRoom:MsgBase {
32)     public MsgEnterRoom() {protoName = "MsgEnterRoom";}
33)     //客户端发
34)     public int id = 0;
35)     //服务端回
36)     public int result = 0;
37) }
38)

```



```

39) //玩家信息
40) public class PlayerInfo{
41)     public string id = "lpy";    //账号
42)     public int camp = 0;         //阵营
43)     public int win = 0;          //胜利数
44)     public int lost = 0;         //失败数
45)     public int isOwner = 0;      //是否是房主
46) }
47)
48) //获取房间信息
49) public class MsgGetRoomInfo:MsgBase {
50)     public MsgGetRoomInfo() {protoName = "MsgGetRoomInfo";}
51)     //服务端回
52)     public PlayerInfo[] players;
53) }
54)
55) //离开房间
56) public class MsgLeaveRoom:MsgBase {
57)     public MsgLeaveRoom() {protoName = "MsgLeaveRoom";}
58)     //服务端回
59)     public int result = 0;
60) }
61)
62) //开战
63) public class MsgStartBattle:MsgBase {
64)     public MsgStartBattle() {protoName = "MsgStartBattle";}
65)     //服务端回
66)     public int result = 0;
67) }

```

● 3D_Server\script\proto\SyncMsg.cs

```

1) //同步坦克信息
2) public class MsgSyncTank:MsgBase {
3)     public MsgSyncTank() {protoName = "MsgSyncTank";}
4)     //位置、旋转、炮塔旋转
5)     public float x = 0f;
6)     public float y = 0f;
7)     public float z = 0f;
8)     public float ex = 0f;
9)     public float ey = 0f;
10)    public float ez = 0f;
11)    public float turretY = 0f;
12)    public float gunX = 0f;
13)    //服务端补充
14)    public string id = "";    //哪个坦克

```

```

15) }
16)
17) //开火
18) public class MsgFire:MsgBase {
19)     public MsgFire() {protoName = "MsgFire";}
20)     //炮弹初始位置、旋转
21)     public float x = 0f;
22)     public float y = 0f;
23)     public float z = 0f;
24)     public float ex = 0f;
25)     public float ey = 0f;
26)     public float ez = 0f;
27)     //服务端补充
28)     public string id = "";           //哪个坦克
29) }
30)
31) //击中
32) public class MsgHit:MsgBase {
33)     public MsgHit() {protoName = "MsgHit";}
34)     //击中谁
35)     public string targetId = "";
36)     //击中点
37)     public float x = 0f;
38)     public float y = 0f;
39)     public float z = 0f;
40)     //服务端补充
41)     public string id = "";           //哪个坦克
42)     public int hp = 0;               //被击中坦克血量
43)     public int damage = 0;          //受到的伤害
44) }

```

- 3D_Server\script\proto\SysMsg.cs

```

1) public class MsgPing:MsgBase {
2)     public MsgPing() {protoName = "MsgPing";}
3) }
4)
5) public class MsgPong:MsgBase {
6)     public MsgPong() {protoName = "MsgPong";}
7) }
214)

```

- 3D_Server\script\SQL\game.sql

```

1) drop database if exists game;
2)
3) CREATE DATABASE IF NOT EXISTS game;
4)

```

```

5) use game;
6)
7) SET NAMES utf8mb4;
8) SET FOREIGN_KEY_CHECKS = 0;
9)
10) -- -----
11) -- Table structure for account
12) -- -----
13) DROP TABLE IF EXISTS `account`;
14) CREATE TABLE `account` (
15)   `id` text CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT N
      ULL,
16)   `pw` text CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL,
17)   PRIMARY KEY (`id`(20)) USING BTREE
18) ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_a
      i_ci ROW_FORMAT = Dynamic;
19)
20) -- -----
21) -- Table structure for player
22) -- -----
23) DROP TABLE IF EXISTS `player`;
24) CREATE TABLE `player` (
25)   `id` text CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT N
      ULL,
26)   `data` text CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NUL
      L,
27)   PRIMARY KEY (`id`(20)) USING BTREE
28) ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_a
      i_ci ROW_FORMAT = Dynamic;
29)
30) SET FOREIGN_KEY_CHECKS = 1;

```

课程设计评分表

课题名称： C#高级程序设计

项 目	评 价
系统功能及代码编写（权重 40%）	
信息检索工具（权重 30%）	
项目的测试（权重 10%）	
项目的评价（权重 10%）	
课程设计报告质量（权重 10%）	
综合成绩	

教师签名： _____

日 期： _____