**前门3D虚拟购物街**

**接口需求规格说明书**

**编写人：**

**编写日期：** 年 月 日

文档修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本编号 | 说明 | 日期 | 变更人 | 批准日期 | 批准人 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1.引言 5](#_Toc263633130)

[1.1 编写目的 5](#_Toc263633131)

[1.2 项目名称 5](#_Toc263633132)

[1.3 名词定义 5](#_Toc263633133)

[2.项目描述 5](#_Toc263633134)

[2.1 功能简介 5](#_Toc263633135)

[2.2 运行环境 6](#_Toc263633136)

[3.接口需求 7](#_Toc263633137)

[3.1客户端与服务器端接口 7](#_Toc263633138)

[3.1.1 用户登入登出系统接口 7](#_Toc263633139)

[3.1.2 用户动作消息同步系统接口 13](#_Toc263633140)

[3.1.3 用户聊天系统接口 17](#_Toc263633141)

[3.1.4 用户组队系统接口 21](#_Toc263633142)

[3.1.5 用户好友系统接口 25](#_Toc263633143)

[3.2 客户端本地接口 28](#_Toc263633144)

[3.2.1 登录界面接口 28](#_Toc263633145)

[3.2.2 换装界面接口 29](#_Toc263633146)

[3.2.3 游戏主界面接口 30](#_Toc263633147)

[3.2.4 当前对象界面接口 31](#_Toc263633148)

[3.2.5 选定对象界面接口 32](#_Toc263633149)

[3.2.6 小地图界面接口 33](#_Toc263633150)

[3.2.7 组队界面接口 33](#_Toc263633151)

[3.2.8 好友列表界面接口 34](#_Toc263633152)

[3.2.9 系统选项界面接口 35](#_Toc263633153)

[3.2.10 聊天界面接口 35](#_Toc263633154)

[4.附录 36](#_Toc263633155)

[4.1 引用文件 36](#_Toc263633156)

[4.2 参考资料 36](#_Toc263633157)

# 1.引言

## 1.1 编写目的

项目接口需求规格说明书的编制是为了使用户和项目开发者双方对该项目的初始规定有一个共同的理解，使之成为整个开发工作的基础。

本说明书的预期读者为客户、业务或需求分析人员、测试人员、用户文档编写者。

## 1.2 项目名称

前门3D虚拟购物街

## 1.3 名词定义

无

# 2.项目描述

## 2.1 功能简介

用户可以通过WEB页面访问三维虚拟前门商业街实景，网站的虚拟消费者形象将会设身处地的进入前门商业街，以了解前门商业街、店铺方位等。消费者可以真实感觉到商铺的实际位置，点击进入该商铺的页面，进行网上购物等。

角色模拟功能在线用户可以设定自身模拟的角色，可以设定性别和基本的外型，人格特质等。在前门大街中可选择自身的职业，平台提供SNS交互设计，使用户在人际脉络中，结识陌生的朋友来获取声望。虚拟世界中，文字交流、动作交流，以及好友间的动态消息，都成为维护和扩展交际圈的方式。

## 2.2 运行环境

硬件环境：

最低配置： CPU：奔腾Ⅲ1G

内存：256M

显卡：32M显存，支持16位真彩显示

推荐配置： CPU：Intel 酷睿 i7

内存：2G DDR3

显卡：1G GDDR5显存

软件环境：

操作系统：简体中文Windows2000/XP/vista/Win7，DirectX 8.1或更高。

浏览器：建议使用IE浏览器（包括IE内核的第三方浏览器）以获得最佳效果。

# 3.接口需求

## 3.1客户端与服务器端接口

### 3.1.1 用户登入登出系统接口

用户输入用户名和密码，服务器端进行验证，成功后用户将会登录到前门大街的3D场景中，服务器端会将所有其他在线用户的信息发送给新用户，同时新用户的信息也会转发给所有其他的在线用户，这样保证所有在线用户互相可见，如果用户下线，服务器也会将这一消息发送给其他所有在线用户，实现这一功能的部分重要接口代码节选如下。

客户端：

// Spawn local player object

private void SpawnLocalPlayer() {

int n = spawnPoints.Length;

Transform spawnPoint = spawnPoints[random.Next(n)];

localPlayerObject.transform.position = spawnPoint.transform.position;

localPlayerObject.transform.rotation = spawnPoint.transform.rotation;

// Start sending our transform to other players

localPlayerObject.SendMessage("StartSending");

}

//Get the remote user list and spawn all remote players that have already joinded //before us

private void SpawnRemotePlayers() {

SmartFoxClient client = NetworkController.GetClient();

foreach (User user in client.GetActiveRoom().GetUserList().Values) {

int id = user.GetId();

if (id!=client.myUserId) SpawnRemotePlayer(user);

}

}

//Spawn remote player object

private void SpawnRemotePlayer(User user) {

// Just spawn remote player at a very remote point

UnityEngine.Object remotePlayer = Instantiate(remotePlayerPrefab, new Vector3(-10000, -10000, -10000), new Quaternion(0,0,0,1));

//Give remote player a name like "remote\_<id>" to easily find him then

remotePlayer.name = "remote\_"+user.GetId();

//Start receiving trasnform synchronization messages

(remotePlayer as Component).SendMessage("StartReceiving");

// Force this player to send us transform

ForceRemotePlayerToSendTransform(user);

}

private void UserLeaveRoom(int userId) {

//Just destroy the corresponding object

GameObject obj = GameObject.Find("remote\_"+userId);

if (obj!=null) Destroy(obj);

}

服务器端：

bool CClientNetWork::Create( ACE\_INET\_Addr &ConnectNetAddr)

{

if(m\_pNetClientConnector)

{

return false;

}

ConnectNetAddr.set\_port\_number(CLIENT\_NET\_GATE\_PORT);

ACE\_NEW\_RETURN(m\_pNetClientConnector, CClientConnector, false);

m\_pNetClientConnector->toServer(INTER\_SERVER\_NET);

if(m\_pNetClientConnector->connect(ConnectNetAddr) == -1)

{

return false;

}

// proactor thread

auto\_ptr<Thread\_Args> thread\_args(new Thread\_Args (this));

static bool bCreateClientNetWorkTread = false;

if ( !bCreateClientNetWorkTread )

{

bCreateClientNetWorkTread = true;

if (ACE\_Thread\_Manager::instance ()->spawn ( CClientNetWork::run\_svc, // Pointer to function entry point.

ACE\_static\_cast (void \*, thread\_args.get ()),

// <run\_svc> parameter.

THR\_DETACHED | THR\_SCOPE\_SYSTEM) == -1)

{

return false;

}

}

// wait

//ACE\_OS::sleep(mt);

for( int i = 0; i < 40; i++ )

{

ACE\_Time\_Value mt;

mt.msec(50);

if( ConnectOk() )

{

return true;

}

ACE\_OS::sleep( mt );

}

return true;

}

### 3.1.2 用户动作消息同步系统接口

用户在游戏中的任何动作行为信息都会发送到服务器端，服务器端会将这一消息转发给所有其他在线用户，这样在线用户就会显示出其他用户连贯的动作行为，比如说：走，跑，跳等行为，实现这一功能的部分重要接口代码节选如下。

客户端：

// Send transform to all other users

public void DoSend() {

SmartFoxClient client = NetworkController.GetClient();

SFSObject data = new SFSObject();

data.Put("\_cmd", "t");

//We put \_cmd = "t" here to know that this object contains transform sync data.

data.Put("x", this.position.x);

data.Put("y", this.position.y);

data.Put("z", this.position.z);

data.Put("rx", this.rotation.x);

data.Put("ry", this.rotation.y);

data.Put("rz", this.rotation.z);

data.Put("w", this.rotation.w);

// We send data using SendObject method here. To optimize this you can use //SendXtMessage method with custum formatted method

// Also an extension on the server side could decide which users really need to receive //the transform

client.SendObject(data);

}

private void SendAnimationMessageToRemotePlayerObject(SFSObject data, User fromUser) {

int userId = fromUser.GetId();

if (userId!=smartFoxClient.myUserId) { // If it's not myself

//Find user object with such Id

GameObject user = GameObject.Find("remote\_"+userId);

//If found - send him animation message

if (user) user.SendMessage("PlayAnimation", data.GetString("mes"));

}

}

服务器端：

int CServerConnector::validate\_connection(const ACE\_Asynch\_Connect::Result& result, const ACE\_INET\_Addr &remote, const ACE\_INET\_Addr&)

{

if (!result.success ())

{

ACE\_Time\_Value delay (m\_nRetryDelay);

m\_nRetryDelay \*= 2;

if (m\_nRetryDelay > MAX\_RETRY\_DELAY)

m\_nRetryDelay = MAX\_RETRY\_DELAY;

ACE\_Proactor::instance()->schedule\_timer (\*this, 0, delay);

return -1;

}

m\_nRetryDelay = INITIAL\_RETRY\_DELAY;

return 0;

}

void CServerConnector::AddConnector(CAIoServiceHandler\* pHandler)

{

m\_pConnectorHandler = (pHandler == NULL) ? m\_pConnectorHandler : pHandler;

DMsgServerConnect connectmsg;

connectmsg.create(m\_nServerType, 0, 0);

ACE\_Message\_Block\* pMessageBlock = connectmsg.Create(connectmsg.GetDataInfo(), connectmsg.GetSize());

if (ServerRecvQueue()->put(pMessageBlock) == -1)

{

pMessageBlock->release();

}

}

### 3.1.3 用户聊天系统接口

在线用户之间的聊天，任何用户发送的消息都会先发送到服务器端，服务器端对消息内容进行关键字过滤的操作后进行转发，转发的目的地依据消息中所标明的目的地字段的值，系统广播信息则会由服务器向所有用户发送，实现这一功能的部分重要接口代码节选如下。

客户端：

public void OnPublicMessage(string message, User fromUser, int roomId)

{

int userId = fromUser.GetId();

if (userId!=smartFoxClient.myUserId) { // If it's not myself

string mes = fromUser.GetName()+": "+message;

// Send chat message to the Chat Controller

SendMessage("AddChatMessage", mes);

//Find user object with such Id

GameObject user = GameObject.Find("remote\_"+userId);

//If found - send him bubble message

if (user) {

user.SendMessage("ShowBubble", mes);

}

}

}

}

服务器端：

void NetMessage::Init(void)

{

m\_usMessageSize = 0;

m\_Info.usMessageType = MESSAGE\_TYPE\_NONE;

m\_Info.uServerID.ServerType = INTER\_SERVER\_NONE;

m\_Info.uServerID.ServerNet = 0;

m\_Info.uServerID.ServerLogic = 0;

m\_Info.uiToSocket = INVALID\_SOCKET;

}

ACE\_Message\_Block\* NetMessage::Create(const char\* pMsgBuf, unsigned int uSize)

{

MESSAGE\_STRUCT strcMessage;

strcMessage.usByteOrdor = 0;

strcMessage.usMessageSize = uSize + MESSAGE\_HEAD\_SIZE;

strcMessage.usMessageType = GetType();

strcMessage.uServerID = m\_Info.uServerID;

strcMessage.uiToSocket = GetSocket();

memcpy(strcMessage.szBuffer, pMsgBuf, uSize);

// message data size

ACE\_Message\_Block\* pMessageBlock = 0;

if (!pMsgBuf || uSize >= MESSAGE\_MAX\_SIZE)

{

return pMessageBlock;

}

int nMessageSize = uSize + PACKET\_HEAD\_SIZE + MESSAGE\_HEAD\_SIZE;

///+4 for test for a error for logic server memcpy read one byte more!

ACE\_NEW\_RETURN (pMessageBlock, CNetPoolBlock (nMessageSize + 4), NULL);

ACE\_CDR::mb\_align(pMessageBlock);

pMessageBlock->copy((char\*)&strcMessage, nMessageSize);

return pMessageBlock;

}

### 3.1.4 用户组队系统接口

用户邀请其他在线用户加入队伍的时候，会发送组队邀请消息给服务器，服务器对被邀请人进行判断是否已经加入队伍，如果已经加入其他队伍了，直接返回不能组队的消息，如果被邀请人没有加入队伍则向被邀请人发送请求组队的消息，如果被邀请人同意的话，则服务器会返回确认的消息，实现这一功能的部分重要接口代码节选如下。

客户端：

public void OnTeamMessage(string message, User SendUser, User ToUser)

{

int userId = SendUser.GetId();

// If it's not myself

if (userId!=smartFoxClient.myUserId) {

string mes = SendUser.GetName()+": "+message+" To: "+ToUser.GetName();

// Send team message to the Server Controller

SendMessage("AddTeamMessage", mes);

}

服务器端：

MESSAGE\_TYPE NetMessageFactory::GetMessageType(ACE\_Message\_Block\* pBlock)

{

MESSAGE\_TYPE type = MESSAGE\_TYPE\_NONE;

if (!pBlock)

{

return type;

}

MESSAGE\_STRUCT\* pInfo = (MESSAGE\_STRUCT\*)(pBlock->base());

if (pInfo)

{

type = (MESSAGE\_TYPE)(pInfo->usMessageType);

}

return type;

}

MY\_SOCKET NetMessageFactory::GetMessageSocket(ACE\_Message\_Block\* pBlock)

{

MY\_SOCKET sock = INVALID\_SOCKET\_HANDLER;

if (!pBlock)

{

return sock;

}

MESSAGE\_STRUCT\* pInfo = (MESSAGE\_STRUCT\*)(pBlock->base());

if (pInfo)

{

sock = pInfo->uiToSocket;

}

return sock;

}

ServerID NetMessageFactory::GetToServer(ACE\_Message\_Block\* pBlock)

{

ServerID idServer;

idServer.ServerType = 0;

idServer.ServerNet = 0;

idServer.ServerLogic = 0;

if (!pBlock)

{

return idServer;

}

MESSAGE\_STRUCT\* pInfo = (MESSAGE\_STRUCT\*)(pBlock->base());

if (pInfo)

{

idServer = pInfo->uServerID;

}

return idServer;

}

### 3.1.5 用户好友系统接口

用户将添加好友或者删除好友的消息发送给服务器，服务器会对数据库进行相应的操作，从而在好友列表里面增加或者删除好友，实现这一功能的部分重要接口代码节选如下。

客户端：

public void OnFriendMessage(string message, User SendUser, User ToUser)

{

int userId = SendUser.GetId();

// If it's not myself

if (userId!=smartFoxClient.myUserId) {

string mes = SendUser.GetName()+": "+message+" To: "+ToUser.GetName();

// Send team message to the Server Controller

SendMessage("AddFriendMessage", mes);

}

服务器端：

int CNetServerSendTask::svc(void)

{

int m\_nCount = 0;

int n = 0;

int nCheck = 10000;

int nTimeBegin = ::GetTickCount();

CNetServerClientHandler\* pConnector = NULL;

ACE\_Message\_Block\* pMessageBlock = NULL;

MESSAGE\_TYPE typeMessage = MESSAGE\_TYPE\_NONE;

NetMessage\* pMessage = NULL;

MESSAGE\_STRUCT\* pInfo = NULL;

MY\_SOCKET sock = 0;

while (true)

{

if (getq(pMessageBlock) != -1)

{

// net server message process

//typeMessage =

NetMessageFactory::GetMessageType(pMessageBlock);

// send message to client connector

sock = NetMessageFactory::GetMessageSocket(pMessageBlock);

pConnector = NetServerClientAcceptor()->GetConnector(sock);

if (pConnector)

{

pInfo = (MESSAGE\_STRUCT\*)(pMessageBlock->base());

pConnector->WriteMessage(pMessageBlock);

}

else

{

pMessageBlock->release();

}

}

}

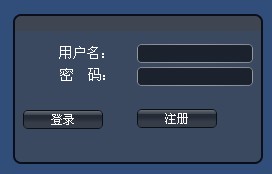
return 0;

}

## 3.2 客户端本地接口

### 3.2.1 登录界面接口

提供给用户输入用户名和密码的图形界面，用户输入后点击登录进入服务器端验证，验证成功后登录游戏场景，否则给出错误提示。



### 3.2.2 换装界面接口

提供用户选择角色以及角色形象打扮的图形界面，用户可以随意搭配出自己喜欢的形象出来，从而打造有个性的游戏生活。



### 3.2.3 游戏主界面接口

在游戏运行中实时显示在游戏界面中的各个窗口，包括当前自己对象的窗口，选定对象的窗口，小地图窗口，组队列表窗口，好友列表窗口，系统选项窗口以及聊天窗口。



### 3.2.4 当前对象界面接口

用户点击当前用户窗口，会弹出GUI窗口，显示离开队伍（只有当前已经加入队伍情况下，点击才会生效）和解散队伍（只有当前在一个队伍中并且自己是队长的情况下，点击才会生效）。



### 3.2.5 选定对象界面接口

用户在游戏中点击其他用户，其他用户的信息会显示在选定对象窗口，用户再点击选定对象窗口，会弹出GUI，显示发送消息、加为好友、组队邀请，加入黑名单操作。



### 3.2.6 小地图界面接口

用户可以查看当前地图中用户分布的情况，可以点击小地图某个可点范围，实现跳转功能。



### 3.2.7 组队界面接口

显示当前组队的情况，可以对队伍成员进行操作，包括发送消息、添加好友、跟随、踢出队伍（只有当前用户为队长才可以实现）、设为队长（只有当前用户为队长才可以实现）操作。



### 3.2.8 好友列表界面接口

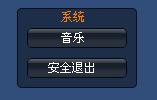
显示当前的好友列表以及被列为黑名单的列表，用户如果点击好友列表中的成员将会弹出GUI窗口，显示发送消息、删除操作，而用户点击黑名单列表中的成员弹出的GUI窗口之显示删除操作。





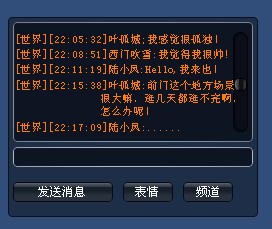
### 3.2.9 系统选项界面接口

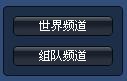
包括声音的设置以及安全退出的选项。



### 3.2.10 聊天界面接口

用户可以和别的在线用户进行私聊，或者和在同一频道内的所有用户进行聊天，频道可以更换世界频道和组队频道，还可以选择静态图片的表情发送。





# 4.附录

## 4.1 引用文件

《前门3D虚拟购物街项目委包合同》

《前门3D虚拟购物街项目总体技术方案》

《前门3D虚拟购物街项目概要设计方案》

## 4.2 参考资料

《软件需求（第2版）》 Karl E.Wiegers著

《网络游戏服务器编程》 （韩）全洪 著

《虚拟现实技术》 胡小强 编著

《SQL完全手册》（荷）范德兰斯　著