# 3 系统实现

## 3.1人员安排

“NEUIM”简易即时通讯应用系统实现小组

组长：

张翰韬

组员：

蒙龙

赖煜华

董子圣

金鑫一

罗康

曾志强

|  |  |  |  |
| --- | --- | --- | --- |
| 工作范围： | 开发环境 | 实现效果 | 人员分配 |
| 后端 | Python  Websocket  sqlight3 | 实现数据结构定义，数据库的连接，与前端沟通 | 张翰韬  董子圣 |
| 服务器 | 云服务器环境 | 租用服务器，部署基本环境 | 赖煜华 |
| 前端 | Unity  C# | UI制作  与服务器端沟通 | 蒙龙 |
| 网络接口部分 | 前后端交换数据约定设计  Json格式编写 | 通过网络协议利用云服务器实现前端和后端信息的交换 | 罗康  曾志强  金鑫一 |

## 3.2 系统通用类实现

把经常用到的操作封装在通用类中，需要用到哪种操作的时候直接调用封装好的类就可以实现操作。这样可以实现代码的可重用性，大大节省编写程序的时间。本系统将所有通用类都放入Scrip文件夹中，以便形成自己的命名空间，如图3.1所示。

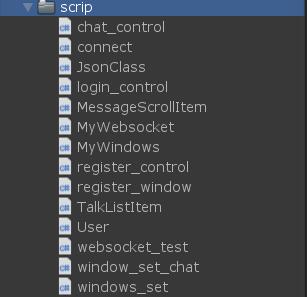


图3.1系统通用类管理

### 3.2.1 聊天控制类chat\_control实现

聊天控制类由chat\_control类实现，该类主要封装了与聊天控制有关的控件的一些对象和方法，例如输入栏控件类型的变量message\_intput，是一个可以输入信息的输入栏对象。除此之外还有执行聊天控制有关操作的对象和方法，例如按钮控件类型的变量send\_button。另外，chat\_control类提供了几个聊天控制有关的函数，对应函数名以及执行的操作如表3.1所示

表3.1 chat\_control类中几个聊天控制函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| private void Init() | 完成界面的初始化 |
| public void UpdateMessageScroll(string id) | 切换聊天框使用，由TalkListItem调用 |
| public void SendMessage() | 发送信息 |

### 3.2.2 服务器连接相关类connect实现

服务器连接类由connect类实现。connect类主要封装了与服务器连接有关的一些控件对象和方法，例如inputField（输入域）类型的公有变量hostInput、portInput和textInput，分别可以输入本地、端口信息以及传送文本信息。除此之外还有Text类型的控件变量connetText,txtStr，用来实现服务器连接可交互界面的有关操作。connet类提供了几个服务器连接有关的函数，对应函数名以及执行的操作如表3.2所示

表3.2 connect类中几个服务器连接相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新函数 |
| public void Connection() | 链接服务器并接收 |
| private void ReceiveCb(IAsyncResult ar) | 从服务器接收数据 |
| public void Send() | 给服务器发送消息 |

### 3.2.3 登录控制类login\_control实现

登录控制类由login\_control类实现。login\_control主要封装了登录操作有关的一些控件对象和方法。在Unity环境下，存在inputField类型的公有变量id\_input和password\_input，可以在两个输入域中输入账号名和密码。另外,还有gameobject（游戏对象）类型的两个变量id\_missing与password\_missing,可以用来辅助实现账号密码错误情况的一些动态效果和反馈。此外，还有私有MyWebSocket变量myWebSocket以及私有string类型变量id与password，这些私有变量用来表示不能被直接访问登录信息。Login\_control类提供了几个登录控制有关的函数，对应函数名以及执行的操作如表3.3所示

表3.3 login\_control类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |
| public void LoginOnClick() | 按钮点击 |
| public void LoginOnOpen(WebSocket web) | 链接的回调 |
| public void LoginOnMessage(WebSocket web, string message) | 收到信息的回调 |
| public void LoginOnError(WebSocket ws, Exception ex) | 登录错误处理 |

### 3.2.4 消息类MessageScrollItem实现

消息类由MessageScrollItem类来实现，MessageScrollItem类主要封装了与消息有关的一些控件对象和方法，例如Unity的Text类对象messageText以及Image类对象iconImage。分别用来表示消息的内容和头像。MessageScrollItem类提供了几个消息操作有关的函数，对应函数名以及执行的操作如表3.4所示

表3.4 MessageScrollItem类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| public void SetMessage(string message) | 消息设置 |
| public void SetIcon(User.Icon icon) | 消息头像设置 |
| static public int CountChinese(string message) | 计算字符串中中文字的数量 |
| public bool IsTooLong(string message) | 判断字符串是否需要分割 |
| public void DivideMessage(ref string message, ref string tail\_message) | 字符串分割 |

### 3.2.5 服务器交互类MyWebSocket实现

服务器交互类由MyWebSocket类实现，MyWebSocket类主要封装了一些与服务器交互功能有关（或需要服务器参与的）的方法函数以及一些变量，如获取、发送数据、创建单例等有关函数以及address、meg\_send、meg\_receive等变量。几个服务器操作有关的对应函数名以及执行的操作如表3.5所示

表3.5 MyWebSocket类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| private MyWebSocket(string address = "ws://60.205.178.57:8888/main") | 创建websocket单例 |
| public bool Connect() | 开始链接 |
| public void Close() | 断开链接 |
| public bool IsConnected() | 判断是否开始链接 |
| private void OnMessageReceived(WebSocket ws, string message) | 当有消息从服务器发过来时调用 |
| public void Login(string id, string password) | 登录函数 |
| public void Register(string nickname, string password, string email) | 注册函数 |
| public void SendMessage(string message) | 发送消息 |
| public void AddFriend() | 添加好友 |
| public void SearchFriend() | 搜索好友 |
| public void DeleteFriend() | 删除好友 |
| void OnOpen(WebSocket webSocket) | 链接成功回调 |
| public static string GetMD5(string password) | 密码转换为MD5 小写 |
| void OnError(WebSocket ws, Exception ex) | 发生错误回调函数 |
| void OnClosed(WebSocket webSocket, UInt16 code, string message) | 链接断开回调函数 |

### 3.2.6 主窗口类MyWindow实现

主窗口类由MyWindow类实现，主要封装了一些窗口初始化预设相关的成员变量和窗口操作方式对应的函数。成员变量有uint型的常量SWP\_SHOWWINDOW = 0x0040，int型的常量GWL\_STYLE = -16，WS\_BORDER = 1，WS\_POPUP = 0x800000，SW\_SHOWMINIMIZED = 2。MyWindow类提供了几个主窗口操作有关的函数，对应函数名以及执行的操作如表3.6所示

表3.6 MyWindow类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| static public void SetWindowsSize(int cx,int cy) | 设置窗口大小 |
| static public void Minimized() | 最小化窗口 |
| static public void SetWindowsNoFrame() | 设置无边框窗口 |
| static public void WindowsMove(RectTransform move\_area, RectTransform[] exclude\_area = null) | 移动窗口 |

### 3.2.7 注册控制类register\_control实现

注册控制类由register\_control类实现，主要封装了注册需要的成员变量和函数，包括InputField类型的name\_input、password\_input、email\_input，分别表示可以输入名字、密码、邮箱的输入栏。GameObject类型的warning成员变量和Text类型的warning\_text则用于错误信息的显示和反馈。register\_control类提供了几个注册控制操作有关的函数，对应函数名以及执行的操作如表3.7所示

表3.7 register\_control类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |
| public void RegisterOnClick() | 注册流程函数 |

### 3.2.8 注册窗口类register\_window实现

注册窗口窗口类由register\_window类实现，主要封装了注册窗口有关的成员变量和函数，成员变量包括Image类型的quit\_button、minimize\_button，分别为退出按钮和最小化按钮的图像对象。以及GameObject类型的move\_area，为移动区域操作游戏类对象。register\_window类对应函数名以及执行的操作如表3.8所示

表3.8 register\_window类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |

### 3.2.9 对话单元类TalkListItem实现

对话单元类由TalkListItem类实现，主要封装了对话单元包含的成员变量和对话单元操作所对应的成员函数。例如Text类型的成员变量nameText，表示可输入姓名的对象控件。TalkListItem类对应函数名以及执行的操作如表3.9所示

表3.9 TalkListItem类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| public void Set(string id = null,string name = null,string message = null, string time = null) | 初始化的过程中 |
| public void OnClick() | 通过chat\_control.Instance完成message\_scroll的更新 |

### 3.2.10 用户类User实现

用户类由User类实现，主要封装了一些成员类、成员变量以及成员函数。成员类有朋友类Friend、对话类Talk和消息类Message;成员变量例如id，name以及由成员类对象构成的队列friendList与talkList。User类提供了几个用户有关的函数，对应函数名以及执行的操作如表3.10所示

表3.10 User类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| public void Init(string id,string name, Icon icon) | 在登录界面的Login中调用，完成初始化 |
| public Talk FindTalk(string id) | 获得对应id的本地Talk聊天记录 |
| public Friend FindFriend(string id) | 获得对应id的朋友 |

### 3.2.11 服务器交互测试类websocket\_test实现

服务器交互测试类由websocket\_test类实现，主要封装了测试服务器交互性能的一些变量和函数。成员变量例如InputField类型的hostInput和portInput等，用于读取输入的本地地址和端口等等，websocket\_test类提供了几个注册控制操作有关的函数，对应函数名以及执行的操作如表3.11所示

表3.11 websocket\_test类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |
| void OnOpen(WebSocket web) | 是链接的回调 |
| void OnMessage(WebSocket web,string message) | 收到信息的回调 |
| void OnClose(WebSocket web) | 连接断开 |
| public void Connect() | 链接服务器 |
| public void Send() | 向服务器发送数据 |

### 3.2.12 聊天窗口设置类window\_set\_chat实现

聊天窗口设置类由window\_set\_chat类实现，主要封装了设置聊天窗口有关的一些控件对象和函数。例如Image控件类型的quit\_button、minimize\_button和chat\_up\_background，表示退出、最小化按钮和聊天背景控件。window\_set\_chat类提供了几个聊天窗口设置有关的函数，对应函数名以及执行的操作如表3.12所示

表3.12 websocket\_test类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |
| public void Quit() | 退出功能实现 |
| public void Minimized() | 最小化功能实现 |

### 3.2.13主窗口设置类windows\_set实现

主窗口设置类由windows\_set类实现，主要封装了主窗口设置有关的一些控件对象和函数。例如Image空间类型的school\_background\_image、quit\_button\_image以及minimized\_button\_image，分别表示背景图片(东大校园背景)、退出按钮图片以及最小化按钮图片，windows\_set类提供了几个主窗口设置有关的函数，对应函数名以及执行的操作如表3.13所示

表3.13 websocket\_test类中几个登录控制相关函数列表

|  |  |
| --- | --- |
| **函数名** | **函数需要执行的操作** |
| void Start() | 在第一帧更新前调用该函数 |
| void Update() | 更新有关信息 |
| public void Quit() | 退出功能实现 |
| public void Minimized() | 最小化功能实现 |
| public void LoginOnClick() | 点击登录事件相关操作 |
| public void RegisterOnClick() | 点击注册事件相关操作 |

## 3.3 登录注册模块实现

### 3.3.1 登录注册模块实现简介

分为登录和注册两个部分，注册目标包括

1.作为第一次使用必要条件，即用户在第一次使用“NEUIM”的时候必须进行注册

2.唯一标识的确认，即在注册时需要输入唯一邮箱

3.在客户端中首先完成基本的输入合法性的判断，例如密码，用户名，邮箱格式等输入是否合法。

4.注册完成后传送至服务器。用户输入完成后，客户端将输入打包成Json文件发送给服务器，服务器判断该邮箱是否已经合法进行返回，注册成功后返回账号。

5.后续开发，在开发工作基本完成且时间的充裕的前提下可以考虑在之后的版本迭代中加入邮箱验证码验证的功能。

登陆目标包括

1.登录前的检查，若是第一次使用“NEUIM”，则先跳转至注册。

2.正确性检验，检验账号是否存在，账号对应密码是否正确。

3.登录成功后传送至服务器。用户输入完成后，客户端将输入打包成Json文件发送给服务器，服务器将用户信息、好友信息和聊天记录更新等内容返回账号。

4. 修改密码，用户可以通过输入注册时的邮箱作为验证，邮箱正确后进行密码重置，可以考虑在之后的版本迭代中添加邮箱验证码的功能。

### 3.3.2 登录注册模块相关类实现

登录注册模块主要用到登录控制类login\_control、主窗口设置类windows\_set、服务器交互类MyWebSocket、注册控制类register\_control、注册窗口类register\_window等。登录界面由登录控制类和主窗口设置类实现，登录控制类提供登录相关以及跳转至其他界面的一些方法函数，例如账号密码检测、是否为第一次使用的检测和修改密码等等，主窗口设置类提供了一些初始化窗口的参数以及它们的初始值，还有改变窗口的一些简单操作函数。注册控制类和注册窗口类则是控制注册相关界面和操作。例如，用户在第一次使用“NEUIM ”的时候首先需要进行注册，在注册时需要输入唯一邮箱，在客户端中首先完成基本的输入合法性的判断，例如密码，用户名，邮箱格式等输入是否合法，此时就要用到注册控制类的判断函数。详细的函数可参照3.2系统通用类实现中对应类成员变量和成员函数定义与解释。

### 3.3.3 登录注册模块文件及跳转关系

（1）模块涉及代码文件列表如表3.13所示。

表3.13 登录注册模块涉及代码文件列表

|  |  |  |
| --- | --- | --- |
| **文件名** | **文件路径** | **文件说明** |
| Login.scene | D:\Unity\asset\scene | 前端Unity场景编辑文件，实现登录界面。 |
| Login\_control.cs | D:\Unity\asset\scrip | 后端C#代码文件，实现登录相关的方法，例如用户登录，修改密码 |
| Register.scene | D:\Unity\asset\scene | 前端Unity场景编辑文件，实现注册界面。 |
| Register\_control.cs | D:\Unity\asset\scrip | 后端C#代码文件，实注册相关的方法，例如注册时输入账号，判断两次密码是否一致 |

（2）模块涉及页面跳转关系图，如图3.2所示。

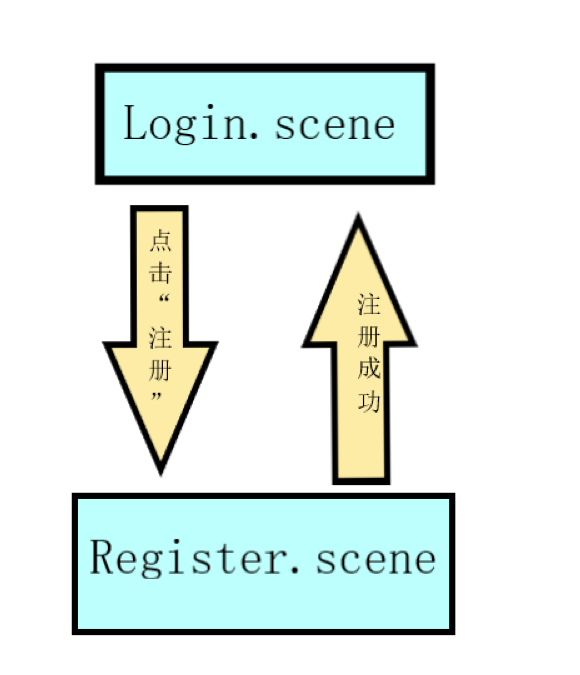


图3.2 登录注册模块涉及界面跳转关系图

### 3.3.4 登录注册模块程序流程

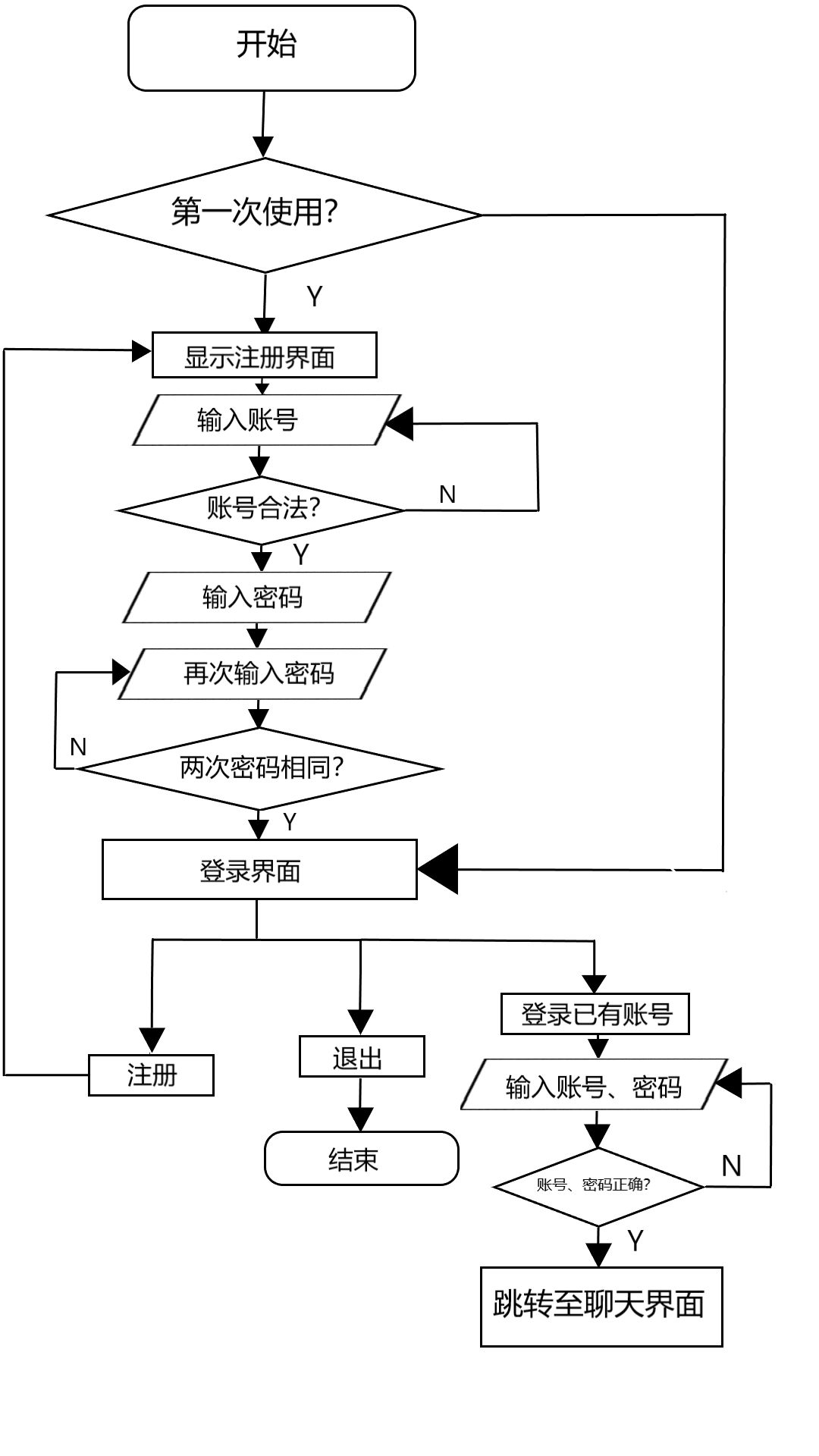


图3.3 登录控制模块程序流程图

### 3.3.5 登录控制模块实现界面



图3.4 登录控制模块实现界面

## 3.4 好友模块实现

### 3.4.1 好友模块实现简介

好友模块主要有用户搜索、添加好友、删除好友、建立聊群、退出聊群5个部分组成

（1）用户搜索：

用户可以通过输入账号来对其他用户获取群聊进行搜索，客户端将搜索内容打包成搜索请求发送给服务器，服务器通过查询数据库返回相关信息，客户端进行显示。

（2）添加好友：

用户在搜索界面搜索到用户和群组之后可以发送添加好友或申请好友的请求。服务器将请求消息发送给对应的用户或群组管理人员，该请求的实现方式可以参考消息转发的实现原理。

（3）删除好友：

用户可以在好友界面删除对应的好友，删除好友之后不能在打开对应的消息窗口。服务器将该好友从信息从数据库中删除，并将删除消息转发给被删除用户。

（4）建立群聊：

用户可以建立群聊，创建者作为群聊的管理者，管理员负责对其他加入群聊申请进 行处理。可以考虑在后续的迭代中添加更多的群聊功能，例如禁言，设置多个管理员等。

（5）退出群聊：

用户可以在好友界面退出群聊，删除好友之后不能在打开对应的消息窗口。服务器将该群聊从信息从数据库中删除，并将退出消息转发给群聊管理员。管理人员退出群聊后对应的群聊也将注销。

### 3.4.2 好友模块相关类实现

好友模块主要用到了服务器交互测试类websocket\_test、服务器连接相关类connect、消息类MessageScrollItem、服务器交互类MyWebSocket、主窗口类MyWindow、对话单元类TalkListItem、用户类User和聊天窗口设置类window\_set\_chat等。用户搜索、添加好友、删除好友、建立聊群与删除聊群均用到了这些通用类。详细的函数可参照3.2系统通用类实现中对应类成员变量和成员函数定义与解释。

### 3.4.3 好友模块文件

模块涉及代码文件列表如表3.14所示。

表3.14好友模块涉及代码文件列表

|  |  |  |
| --- | --- | --- |
| **文件名** | **文件路径** | **文件说明** |
| chat.scene | D:\Unity\asset\scene | 前端Unity场景编辑文件，实现聊天界面，好友界面也属于聊天场景文件的一部分 |
| Window\_set\_chat.cs | D:\Unity\asset\scrip | 后端C#代码文件，实现好友的方法，例如添加好友、建立聊群 |

### 3.4.4 好友模块程序流程

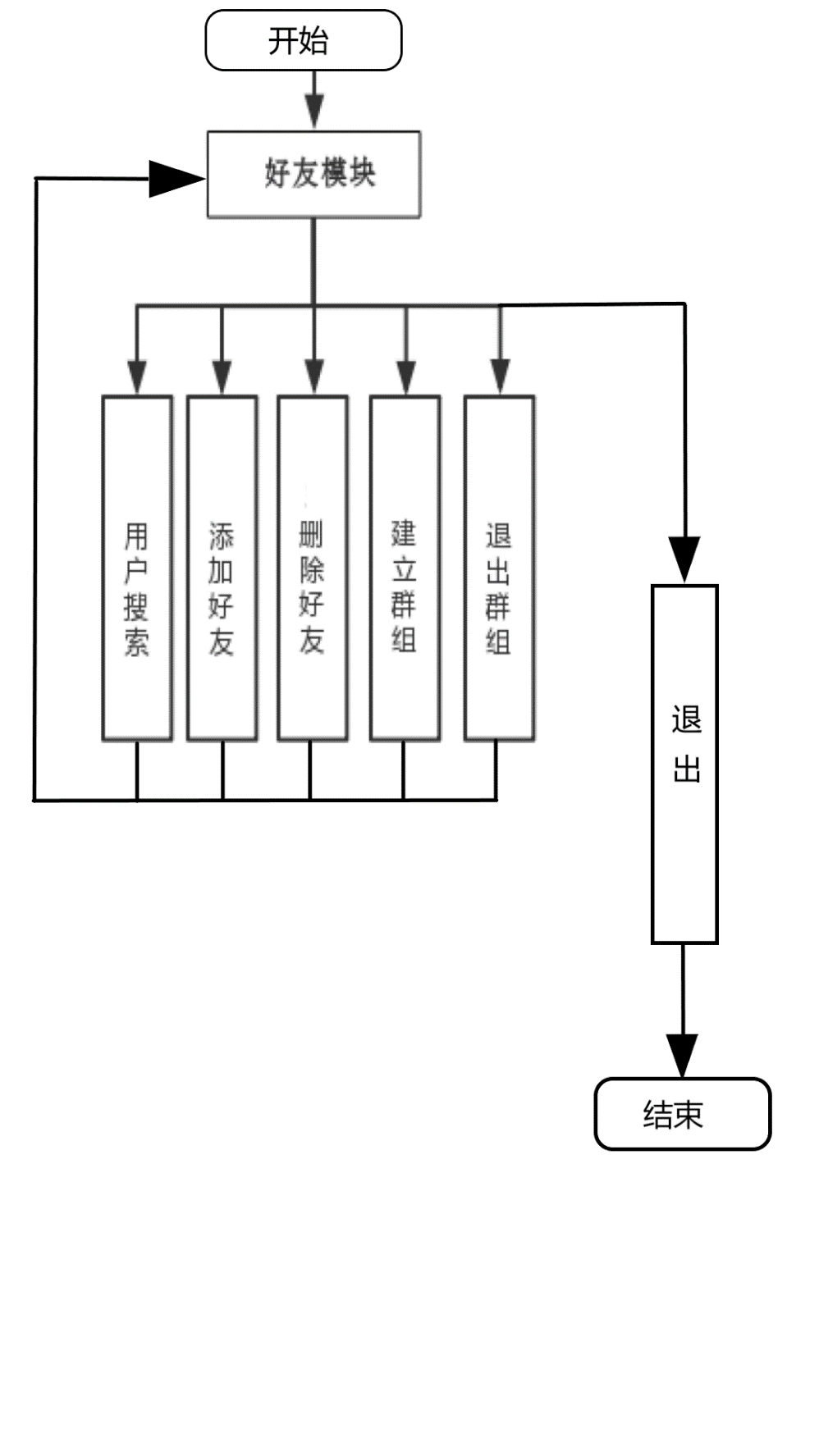


图3.5 好友控制模块程序流程图

### 3.4.5 好友模块实现界面

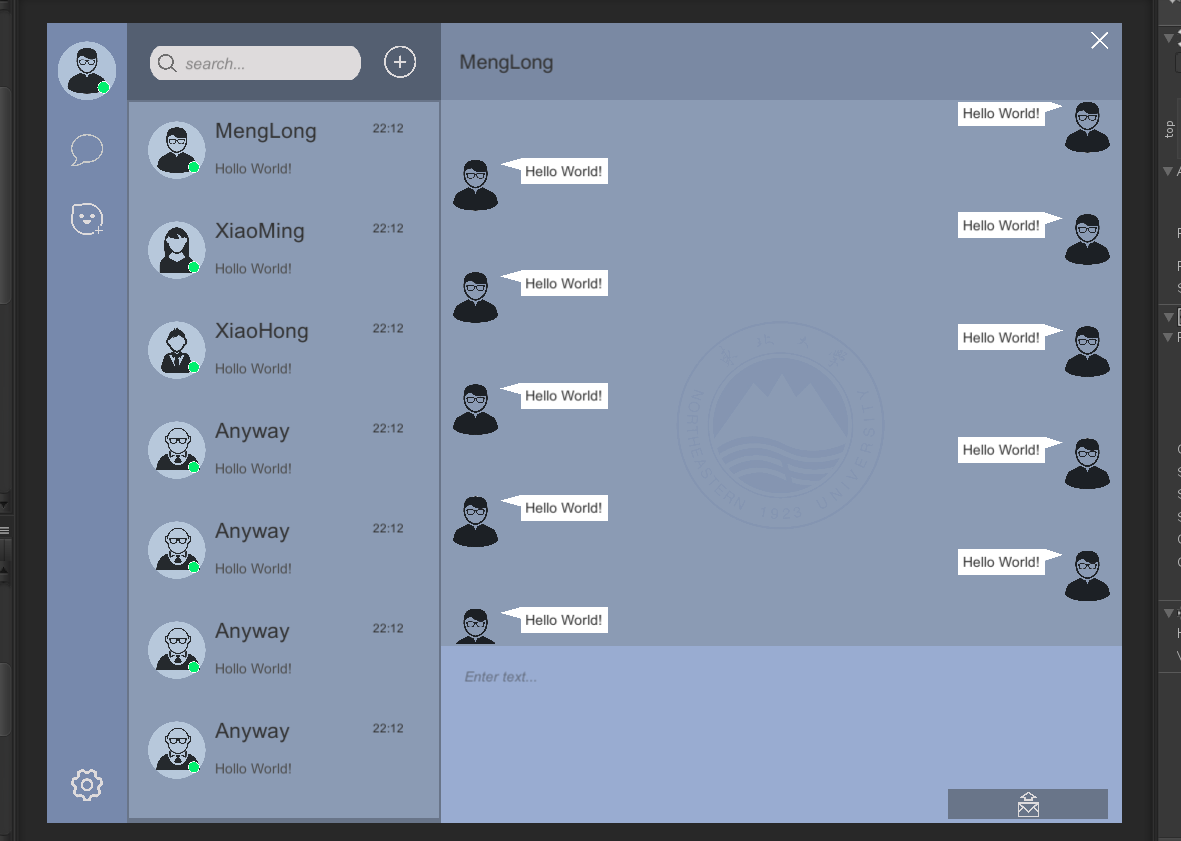


图3.6 好友控制模块实现界面

## 3.5 聊天模块实现

### 3.5.1 聊天模块实现简介

聊天模块主要分为好友聊天、多人聊天、发送文件和显示好友状态四个部分

（1）好友聊天：

用户在好友界面通过双击好友打开对应的聊天界面，可以发送文字和表情，可以考虑在未来的版本迭代中添加文件图片等发送。客户端将相关的信息打包发送给服务器，当目标用户在线时立即进行转发，当用户不在线时进行暂存，当用户上线后立即进行转发。客户端收到消息之后显示为未读消息。

（2）多人聊天：

多人聊天与好友聊天的功能类似。用户在好友界面通过双击群聊打开对应的聊天界面，可以发送文字和表情，可以考虑在未来的版本迭代中添加文件图片等发送。

（3）发送文件：

考虑在开发时间充裕的情况下在未来的版本中添加文件发送的功能，发送的文档可以暂存在服务器中并可以进行多次下载，下载后的文件可以保存在本地。

（4）显示好友状态：

在好友列表中显示好友当前是否在线，可以考虑在未来的迭代版本中添加更多的状态，例如忙碌，请勿打扰等。当用户登录时，服务器返回的消息中包含好友的状态信息，等好友的状态改变时服务器向客户端转发进行状态更新。

### 3.5.2 聊天模块相关类实现

聊天模块主要用到了聊天控制类chat\_control、服务器连接相关类connect、消息类MessageScrollItem、服务器交互类MyWebSocket、主窗口类MyWindow、对话单元类TalkListItem、用户类User和聊天窗口设置类window\_set\_chat等。好友聊天和多人聊天是要用到用户类、聊天窗口设置类、对话单元类、聊天控制类来实现应有的功能。发送文件不仅要用到服务器连接相关类，还要考虑到服务器交互类才可以实现。而显示状态则也需要用户类，并通过聊天窗口设置类的有关操作函数来展现可视化界面。

### 3.5.3 聊天模块文件

模块涉及代码文件列表如表3.15所示。

表3.15好友模块涉及代码文件列表

|  |  |  |
| --- | --- | --- |
| **文件名** | **文件路径** | **文件说明** |
| chat.scene | D:\Unity\asset\scene | 前端Unity场景编辑文件，实现聊天界面 |
| Window\_set\_chat.cs | D:\Unity\asset\scrip | 后端C#代码文件，实现聊天的方法，例如好友聊天，多人聊群 |

### 3.5.4 聊天模块程序流程

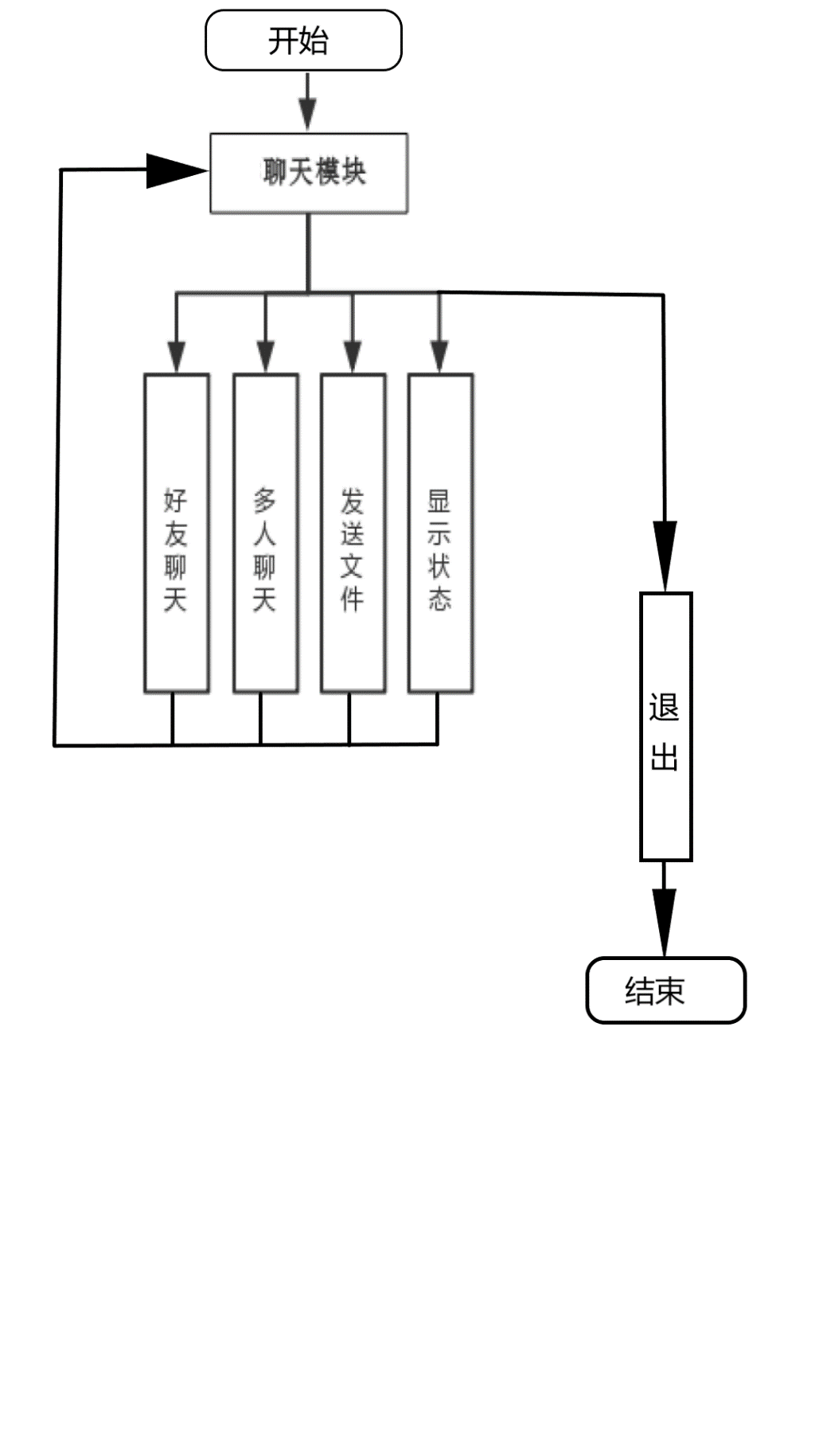


图3.7 聊天控制模块程序流程图

### 3.5.5 聊天模块实现界面

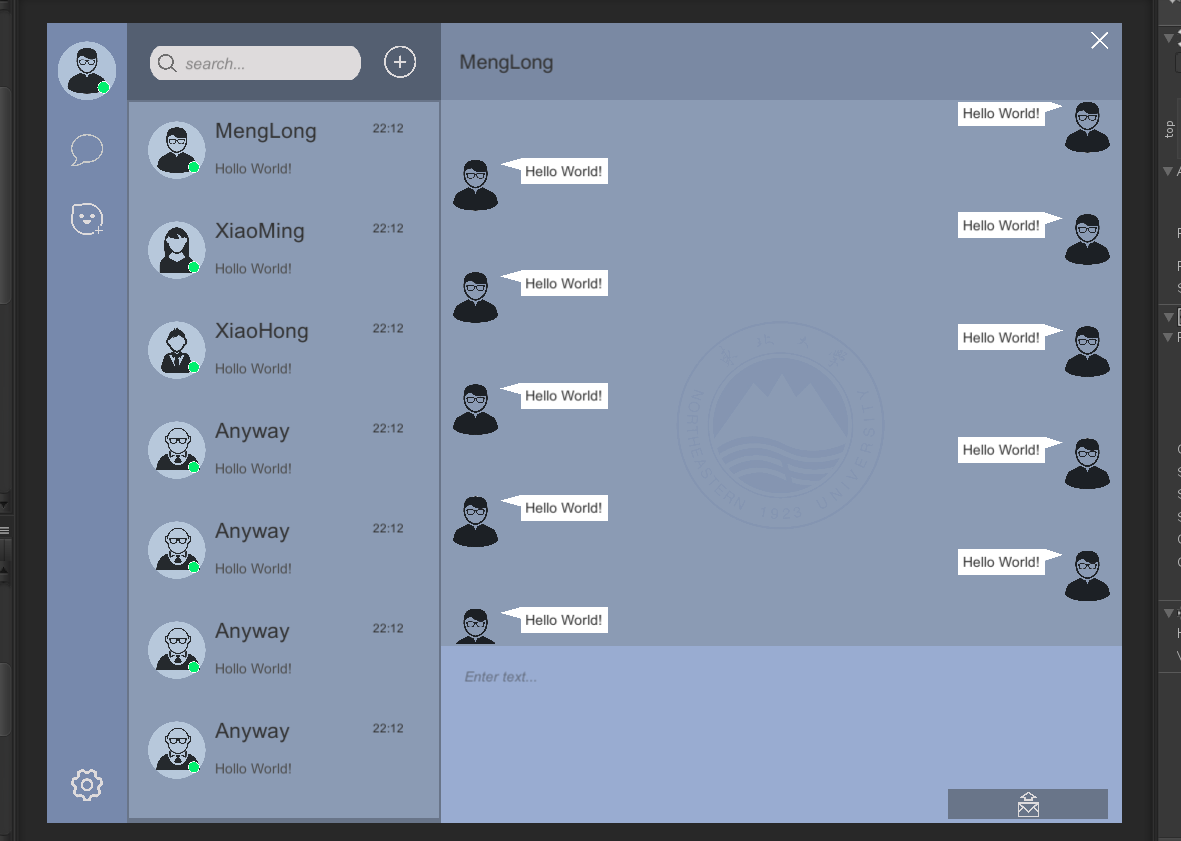


图3.8 聊天控制模块实现界面

## 3.6 实现过程中的需求和设计变动情况说明

“NEUIM”主要包括前端和后端两个部分，其中前端为用户客户端部分，后端为云服务器数据库等部分。客户端中包含软件的用户交互逻辑，发送请求，界面显示，UI设计显示，动画特效等。后端包括云服务器，数据保存，消息转发，响应请求等。

Json格式的设计，将消息，用户账号，用户密码，接受者账号一并合成一个json字

## 3.7 实现小结

在开始确定要设计什么软件是就讨论了很久，在接下来的几天里我们讨论系统实现的设计，人员分工。对于这次大作业不仅考验我们的编程能力，更加考验我们的团队协作能力，在前端和后端的通信之间我们出了一些问题，对于websocket这个东西都不是很了解，刚开始都不知道怎么上手，以致于在代码上产生了很多疑问。在查阅了好多教程之后我们才知道可以利用http通信，用传输json文件来进行前后端的交流，在开发的过程中我们多次因为数据库结构的问题而被迫修改代码，随着越来越多功能的加入，在整合的过程中发现接口对接是无法输出想要的内容，为此也是对代码进行了一顿猛改，好在最后前后端的对接如愿完成了，软件也最终开发完成。在这次开发过程中也是走了一些弯路，但好在最后都一一解决了，我们也学习到了很多东西，对于软件的设计，前期的沟通很重要，这直接影响了后期的开发速度，因此在对软件进行开发过程中前期的沟通一定要做好。