

带有弹幕与审核功能的互动软件

# 设计说明书

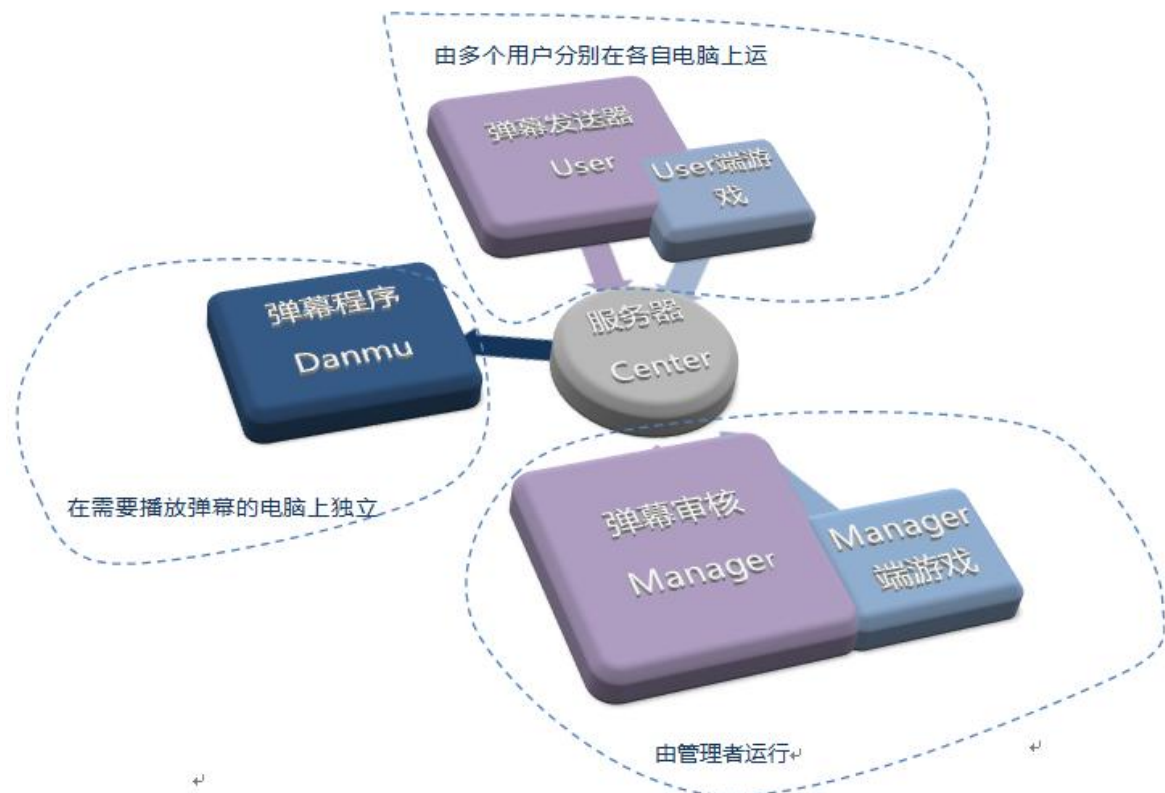
郑 悦 物理系 基科 32 2013012267

白 可 物理系 基科 31 2013012245

张格菲 物理系 基科 31 2013012236

## 功能设计

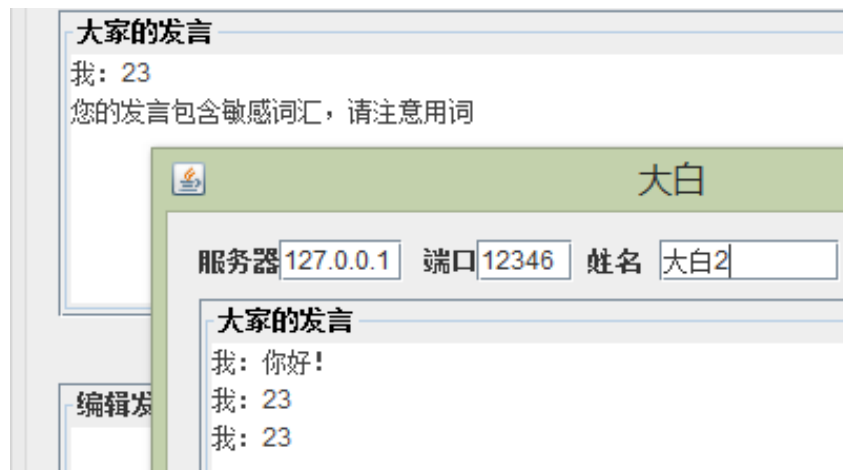
整个软件分为四个主要部分，运行在服务器上的程序、由用户使用的部分、由管理者使用的部分和弹幕播放程序。分别在不同的电脑上运行，由服务器提供信息的交互。能够为各种需要使用弹幕投射的场合，提供弹幕发送的客户端、审核端、弹幕控制端和服务器。其中包含游戏、聊天、内容审核的功能。



## 主要功能

### 1. 审核

管理者对用户发送的弹幕进行审核，通过审核后弹幕出现在弹幕显示端和用户的聊天界面上



## 2. 弹幕

用户可以向屏幕发送两种弹幕，滚动弹幕或者置顶弹幕，滚动弹幕在屏幕上从右向左滚动一次后显示完毕。置顶弹幕在屏幕中央上端显示一定时间后消失。

同时弹幕窗口是透明悬浮的，可以让用户在看视频的同时看到实时的弹幕。



## 3. 游戏

游戏部分提供了所有用户可以一同参与的抽奖功能。游戏分为两个部分，分别由 **user** 程序和 **manager** 程序运行。**Manager** 程序可以控制每轮游戏何时开始，当 **manager** 发送开始信息后，用户可以在自己的游戏程序中选择选项。倒计时结束后由服务器同一发出结果，公布用户是否胜出。

在抽奖活动没有开始时，**user** 和 **manager** 程序都处于等待状态



**Manager** 发出开始游戏的信号以后用户可以开始选择相应的瓶子，每个用户只能选择一个，选定以后不可以改变





类的成员变量和部分方法变量说明在类图上

部分不方便标在类图上的方法变量说明如下

**Class Danmu**

**Public void Closes():**

若与服务器相连，则关闭弹幕线程，与服务器断开连接

**public Danmu()**

构造函数，初始化与用户交互的图形界面，给各个按钮加上监听器

**public boolean connect(String service, int port )**

与服务器连接，service 表示 IP 地址，port 表示接口，若连接成功则返回 true，否则返回 false；

**public synchronized boolean close()**

关闭本对象自己的线程和输入输出流

**public class DanmuThread extends Thread**

内部类，启动弹幕线程

**public class DanmuThread extends Thread**

**Showout.myPanel 类:**

**Public myPanel ( )** //不含参数的构造函数

**Public myPanel(gundongProcessor string1, zhidingProcessor string2 )**

//string1 用来存放滚动弹幕，string2 用来存放置顶函数

**public int available1 ( )** //返回下一个可以显示滚动弹幕的行数

**public int available2( )** //返回下一个可以显示置顶弹幕的行数

**public void paint( Graphics g )** //每 10ms 调用一次 paint 函数，画出此时所有弹幕的位置

**public void run( )** //每 10s 调用一次 paint 函数，直到程序结束

**Show.zhidingProcessor 类**

**public void add( String s )** //在类中加入新的弹幕

**public void add( zhidingdanmu s )** //在类中加入新的弹幕

**public void remove( inti )** //在类中移除序号为 i 的弹幕

**public zhidingProcessor( )** //构造函数

**public int size( )** //返回类中存放弹幕个数

**public zhidingdanmu get( inti )** //返回序号为 i 的弹幕

**Show.gundongProcessor 类**

**public void add( String s )** //在类中加入新的弹幕

**public void add(gundongdanmu s )** //在类中加入新的弹幕

**public void remove( inti )** //在类中移除序号为 i 的弹幕

**public gundongProcessor( )** //构造函数

**public int size( )** //返回类中存放弹幕个数

**public gundongdanmu get( inti )** //返回序号为 i 的弹幕


**showout.danmu 类**

**public danmu( String s )** //构造新的弹幕

**showout.gundongdanmu 类**

```
public gundongdanmu( String s )//构造新的滚动弹幕  
showout.zhidingdanmu 类  
publiczhidingdanmu( String s )//构造新的置顶弹幕
```

#### ▲ MResultthread

- △ win : int
- △ frame : Rollframe
- △ m : Mass
- △ path : int[]
- ▲  MResultthread(Rollframe, Mass)
- ▲ run() : void
- begin(int, int[]) : void //set win,begin the result method in Rollframe

#### ▲ Rollframe

-  wid : int
-  hit : int
- c : Clickmess
- name : String
-  Rollframe(ObjectOutputStream //构造函数，需要传递由 socket 得到的输出流
- invisiblet1() : void //设置 waiting 动画不可见
- visiblet1() : void //设置 waiting 动画为可见
- t1flash() : void //运行 waiting 界面
- invisiblet3() : void //设置 waiting 动画不可见
- visiblet3() : void //设置 waiting 动画为可见
- resultflash(int, int[]) : void //运行结果界面
- mresultflash(int, int[]) : void //构造函数，需要传递由 socket 得到的输出流
- settablebutton() : void //设置允许使用 button
- setunablebutton() : void //设置不允许使用 button
- setchosenbutton(int) : void //显示被选择的 button
- refreshbutton() : void //更新 button 图标
- managerbutton() : void //控制 manager 开始游戏的 button
- setbeginable() : void //设置为可以使用 begin 按钮



# 主要技术难点和算法设计

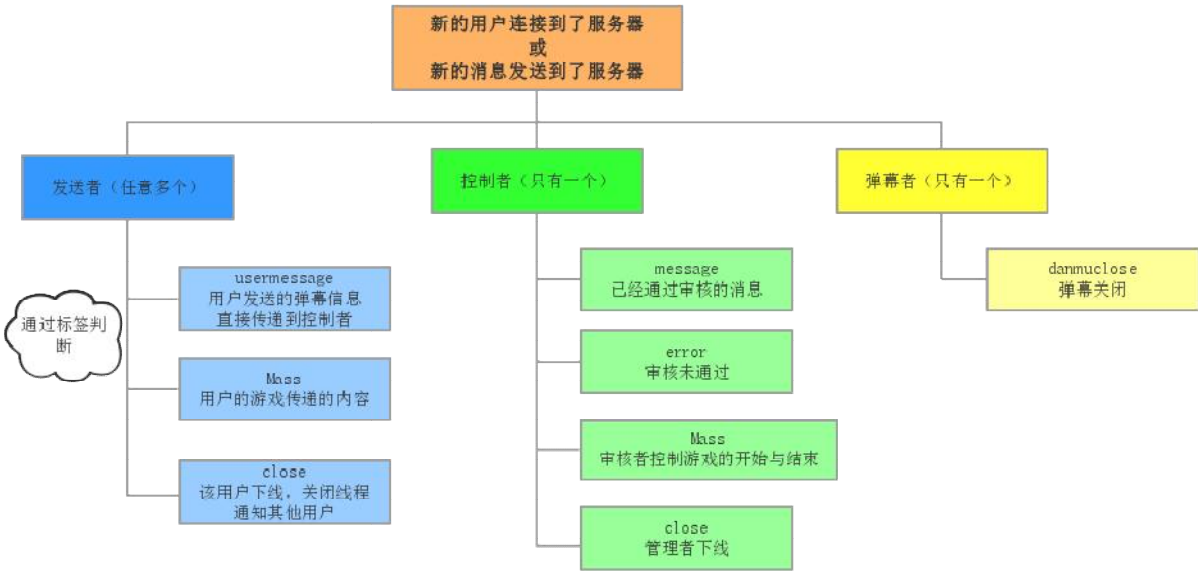
## 【服务器】

服务器作为中转站，需要同时接收来自三个客户端的消息。如何对他们进行很好地分辨。

因此，所有使用者在首次连接服务器时，先在服务器上注册好自己信息，每进入一个新的使用者，服务器就专门给它新建一条线程用来接收该用户传递来的消息。

其次，每个客户端都会传递包括“需要发送的文本信息”“上线”“下线”等多种多样的信息。且部分传递内容不仅仅局限于文本内容，还涉及到游戏内容。

因此传递的是对象，使用 objectstream 进行传递，在程序中，我将传递对象类称作“trans”，trans 中可以包含任意可以进行序列化的内容、对象。每次传递的 trans 都附带有自己的标签. 服务器收到该对象后，先判断它是从哪一个线程来的（发送者、审核者、弹幕者），首先进行解析, 具体过程如下图所示。



图片稍小，大图在附件中，名称为“服务器接收消息”

## 【自动审核】

自动审核因为敏感词库较大，因此不能逐条在信息中查看是否出现，且中文字不同与英文，因此使用 KMP 也十分低效，因此我采用了 DFA 算法，具体过程如下：

DFA 算法的实质是，由 当前事件与状态得到下一个状态，运用在此信息审核中，实质上是一系列相互嵌套的 Map。如图所示，这样同时保存了两个敏感词，在过滤时，如输入“学习”，遇见“学”字，isend=0，进入“学” Map，学 MAP 中没有“习”，因此学习不是敏感词。

```

Map=
{
    学={
        isEnd=0,
        分={
            isEnd=0
            绩={
                isEnd=1
            }
            高={
                IsEnd=1
            }
        }
    }
}

```

### 【用户端】

具有多个用户端，“发送者”，“审核者”，“弹幕者”，每种的功能都有不同。因为与用户交互，因此需要有较好的容错能力，因此，我们来看看什么情况下它会报错：

#### “使用者”

1. 输入错误的 IP 地址或端口，找不到服务器
2. 服务器还没有启动
3. 输入的端口方式有问题
4. 已经连接后的重复连接
5. 没有连接服务器时就开始发送消息
6. “发送者”连接但是“审核者”此时没有连接。
7. 输入内容不可为空

用户在收到每一条信息的时候都会判断是否为自己发送的，如果是自己发送的，在屏幕上的名称会变成“我”，方便识别。

#### “审核者”

审核者在没有消息传递的时候，屏幕上显示“没有人发送消息”，一旦有人发送，就会立刻在屏幕上自动弹出，十分方便。

当屏幕上没有用户所发送的消息时，无论是审核者点“拒绝”或“同意”都无效，不会对用户发送冗余信息。

但是仅仅有界面的显示和信息的传递是不够的，我们还需要考虑用户在下线时，关闭窗口时的一系列操作。因为关闭的不仅仅是用户线程，在服务器中，也有该注册用户的线程，因此，用户下线时，应该通知服务器关闭此线程。停止 run 函数，保证线程安全。

### 【弹幕】

1. 实现了隐藏的关闭按钮：因为看弹幕的同时如果有关闭按钮的存在影响



视野，所以实现了平时隐藏，在鼠标移动到关闭按钮的位置上才显示的关闭按钮。

2. 实现了悬浮弹幕窗口的拖动功能：由于 jframe 类在实现透明悬浮后会变成固定不动的窗口，所以写了弹幕窗口的拖动功能。弹幕显示窗口可以拖过用户的拖动随意挪动位置，更方便用户的操作。
3. 多线程：使用了三个线程来实现弹幕的功能，一个线程实现主窗口显示，一个线程用来实现弹幕的滚动，一个线程不断向服务器获取新的弹幕。同时实现了获取弹幕和显示新获取的弹幕的操作互斥。

算法设计：

1. 弹幕分布算法：由于屏幕上最多每秒显示 10-20 条弹幕，而用户每秒发的弹幕数量不会超出这个量太多，所以设置了弹幕的缓冲区，若某一时刻弹幕数量比较集中，可以在之后的几秒内将这些弹幕都显示出去。
2. 弹幕速度随机分布，同时保证同种弹幕不会相互重叠

### 【游戏】

1. 如果单独完成一个用户的操作，程序是顺序运行的，会比较简单。但由于需要保证所有用户统一一致的行动，就需要始终通过 server 统一 manager 和各个用户的状态。各种状态的信息种类很多，信息传递的顺序也对结果有影响，因此比较不容易管理
2. 不同的 user 登录时间可能不同，需要保证在不同时间登录的用户都可以正常的进行游戏，不会因为先前的信息不完整受到影响
3. 游戏的每一个状态都是在未知的时间停止的，这个时间由操作者决定。一些状态比如显示结果动画的“result thread”，单次运行时间就可能超过了操作者一次操作的时间，因此如果只是用循环条件来控制停止线程，就不能实现操作

结构和算法

1. user 和 manager 程序都使用统一的 Mass 类传递信息。Mass 类中包含了一个表示状态的 tag: int，所有程序都通过 tag 指定的状态运行，保证了各个客户端的统一
2. 每一个动画和显示都由独立的线程运行，以便在状态改变时可以随时停止不受影响。游戏程序本身是 user 或者 manager 上运行的独立线程，而每个游戏线程中包含了 3 个独立状态的线程：WaitingThread，SendThread 和 ResultThread
3. 小球的滚动相当于一个 random walk，有 15 步，16 个结果对应 16 个 button，每个 button 的概率不同，越靠近中间概率越大。小球滚动的路径由服务器利用 random 函数计算出