Whiteboard-Online 技术文档

刘冰雁　00904151
张逸伦　00904159
陈雨禾　00948222
潘楚中　1000010314

2012 年 6 月 29 日

目 录

Whiteboard-Online 是我们四人小组开发的基于互联网的电子白板软件，不同的设备之间能够在同一个共享的电子白板上进行涂鸦等操作。

1 程序架构

为保证程序的兼容性同时尽量提供优秀的UI系统，我们使用了Andriod兼容库。这是Google官方推出的在Android 1.6以上系统可以用的提供了Fragment等高级控件的库。

为了实现可以在互联网上实时通信的电子白板软件。我们对软件进行了如下的架构设计：

首先，由于条件的限制，我们使用的是阿里云的服务器。这台服务器仅支持http协议，所以我们需要实现基于http的实时通信协议。

客户端方面，为了有效的传递数据，我们选用base64的方式对数据进行编码。这种编码在损失1/3的容量的代价下将二进制数据全部变成非常易于处理和传输的文本格式数据。

由于服务器性能的珍贵，我们选择使用重客户端轻服务器的开发模式。服务器端只负责对数据的暂存和转发，数据对其完全透明。为此我们需要在客户端设计对数据的打包和解析的全部过程。

进一步，我们抽象出来Action这个抽象类型，所有的需要通信的操作（绘图、撤销、清屏、加文字、聊天、名称广播）都对应一种Action的子类。Action类的所有子类需要都实现编码至base64、从base64解码、执行自身，这几种（抽象）方法。这样我们就统一了对各种操作的编解码与执行进程。

最后，由于传输的流量限制和对不同设备兼容性的考虑，我们使用矢量图的方式。同时为了能够在将来实现回放功能，我们维护了一个列表，用来储存所有的Action。

2 服务器设计

服务器使用node.js实现。Node.js是一种异步编程语言。我们核心的管理结构是以房间区分的。所以不同的房间可以取得不同的数据。服务器仅在此层面上根据房间名进行分发，对数据不做任何处理。

实时通信是本系统的核心，这里使用的了Node.js的异步特性。我们让客户端向服务器发送一个http请求。如果有消息就立刻返回，如果没有，就讲此连接挂起，直到有消息时候返

回或者超时终止。超时终止后会重新发送请求，确保信息的实时性。通过这种方法我们实现了实时的通信。

3 客户端设计

客户端的设计主要分为数据结构与显示效果两个方面。

3.1 数据结构

数据结构方面，我们必须要设计出一种合适的数据结构，使得它可以存储用户所有的操作，并且进行同一的编码和解码；此外，每个用户都需要本地存储所有用户的所有操作，并且记录其发生的顺序，以便实现任意撤销等高级功能。

为了满足以上要求，我们抽象出了一个Action类，它包括了随手画、橡皮、清屏、撤销、重做、显示文字、发送消息等操作。我们采用usrID和localID对Action进行编号，其中usrID是用户进入房间时服务器分配的唯一ID，用来识别用户，房间内每进入一名用户，usrID++；localID是由客户端本地给Action分配的编号，用来识别同一用户的不同Action，用户每进行一次操作，localID++。localID的大小代表了Action发生的先后顺序。usrID和localID是所有Action类的子类对象所共有的属性，我们称之为公共信息；与之相对的是私有信息，其形式与内容视其所属的操作类型而定。

用户每执行一次操作，客户端就将此次操作按照我们设定的规则进行编码，然后发送到服务器；服务器得到用户发来的操作后，无差别的广播给同一房间中的所有用户。客户端收到编码后的操作，会根据操作所属的子类来决定解码的方式，将编码还原为Action类的某一子类。

在知道了每个Action的公共信息与私有信息之后，我们便可以实现所有预期的功能了。其中随手画、橡皮、清屏、显示文字等功能只需要获得私有信息即可正常显示，而撤销、重做等功能则需要识别其他操作的公共信息才能发挥作用。为方便起见，每个客户端都在本地创建并维护了一个Action列表，按照从服务器接收到的Action进行排序。此Action列表的前后关系能够反映同一用户的Action先后关系，即自动按照此用户的localID进行排序。这对于我们查找某一用户的某个Action来说是非常方便的。

3.2 显示效果

在显示效果方便，我们遇到的主要难点是多点触控缩放特效的编写以及如何高效、流畅的对矢量图进行缩放。

处理多点触控时，我们对触摸事件进行响应，检测当前触摸点的个数及各个点的位置。若当前触摸点的个数与之前相比不变，则根据各触摸点的平均位置平移画板，根据各触摸点到其平均位置的距离之和缩放画板。若画板的放大率超过了预先设定的范围，或者画板移出了边界，我们还编写了让画板”弹回”合适位置和大小的特效。

随之而来的一个问题是，在缩放时如何高效的重绘画板，使得用户几乎感受不到任何延迟。实际上，若对每次触摸事件，都要重新绘制画板，则缩放的流畅度是非常低的。于是我们采取了如下方案：当用户的触摸时间未终止（即仍然与屏幕有接触）时，不对画板进行重绘，而是把此次缩放前的画板按像素放大；当所有手指都离开屏幕时，才进行一次矢量图重绘，精确绘出屏幕显示区域内的画板状况。

按照上述方式，我们成功的编写出了一套流畅度媲美iOS，并且在Android 2.1以上系统全部通用的多点触控缩放系统。

此外，我们设计了一套兼容Android 2.1以上系统的UI界面，在不同版本的系统里具有相同的显示效果，非常美观。