移动互联网课程项目报告

16307130117 陈澄钧

16307130345 曾瑞莹

项目名称

- LOOP循环播放器

项目介绍

项目功能

- 音频的同步播放与录制
- 音频循环播放
- 多音轨播放
- 保存、删除音频文件
- 从本地服务器上传下载音频文件

团队分工

- 策划：陈澄钧，曾瑞莹
- 服务端功能：曾瑞莹
- 客户端逻辑代码：陈澄钧
- 客户端界面：陈澄钧，曾瑞莹
- DEMO录制：张云哲
- 工作量占比：0.5 0.5
- 报告由两人讨论完成

项目设计

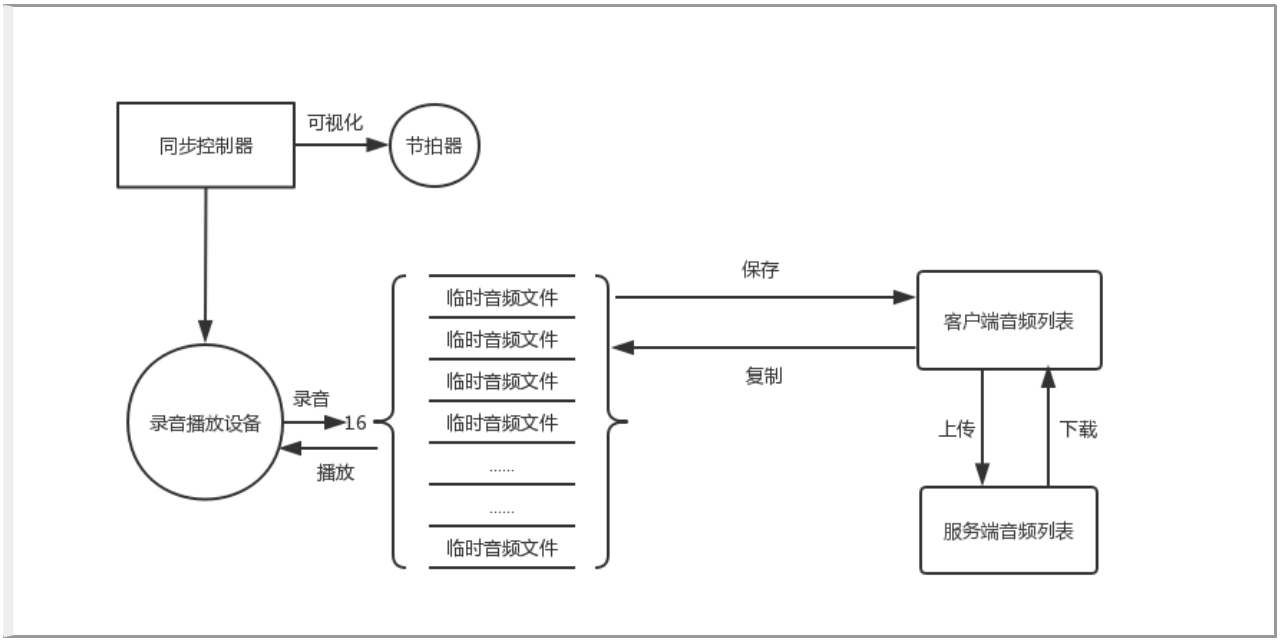
技术难点

- 创建web项目，通过 `Eclipse` 连接 `Tomcat` 搭建本地服务器，实现上传下载功能。*(曾瑞莹)*
  - 服务器端使用servlet重载 `doPost` 函数，借助 `commons-fileupload` 和 `commons-io` 实现接收客户端文件并写入指定文件夹中。考虑到会有相同命名的文件，因此每个音频文件的文件名由一段随机字符加上原文件名组成。同时，将该文件名写入 `index.txt` 文件中，以便于客户端遍历以及获取。
  - 客户端借助一个处理网络请求的开源项目 `OkHttp3` 实现各种网络请求。查看所有文件时则直接访问服务端的 `index.txt` ，下载时在通过文件名访问到该音频资源。由于在安卓模拟器中，访问“localhost”或是回环地址“127.0.0.1”均是访问模拟器本身，在模拟器上访问“10.0.0.2”才是访问本机。
- 节拍器的实现：对计时器进行改造可实现。*(曾瑞莹)*
- 界面跳转的实现：若有播放器在播放音频时，应先释放该播放器资源再退出该界面。*(曾瑞莹)*
- VI设计：为了让整个app更有一种独立完整的app的感觉，因此我们在UI方面还进行了VI设计，选用了主色调"#2f4f4f"，辅助色"#263f3f"以及其他的一些点缀颜色，同时，由于android控件样式比较局限，一些效果没法实现，因此我们自行通过PS来得到部分图片从而实现了整个整体的VI设计，包括logo、图标、按钮、选中效果、节拍器效果等。不算难点，但还是很有工作量的！*(曾瑞莹)*

- 拖拽框的拖动功能：因为该app一个很重要的点便是用户操作的便捷性，因此对于用户将用户音频转移到播放单元格这个功能的实现上，我们采用了拖动实现的方式。*(曾瑞莹)*
  - 在主界面的拖拽框中我们可以看到用户音频的list，长按你想拖动的音频名，它会出现一个镜像的shadow，将其拖至你想放置的单元格便实现了转移。
  - 此功能的实现难点是在于item的获得，因为该过程其实是拖拽框中的点击事件以及主界面中的拖拽事件的一个结合的过程，我们是在list的长按事件中，生成一个对象的shadow，然后将其赋值给activity中的一个对象指针，并调用activity中的touch事件，让该touch事件拖动的对象是这个shadow。之后再计算释放坐标，联动Gridview完成整个拖动功能。
- 播放单元格：为了最优化用户使用，我们最后选择了网格的形式，让其每一格作为一个播放单元格。*(陈澄钧)*
  - 整体是由Gridview(这个有很多坑啊，貌似是自带的bug比如在getview时第一个格子会重复且不按顺序地执行多次)实现的，但不比普通的Gridview(就是里面一些Buttom啥的)，在每一格中需要实现一个圆形播放进度条，以及在内部实现播放控制功能。最后我们实现的方式是自定义一个item类，表示每一格播放单元格，在里面封装好所有音频播放相关功能以及进度条功能，然后在adapter里面对其进行设置监听。
  - 其中，单元格还必须判断此格中是否有音频（因为刚开始每个格子中都是没有音频的，通过在每个格子中录音或是把音频文件从拖拽框拖至指定单元格获得）。在这里我们采取的方法是通过文件名来判断。我们设置让播放单元格访问的音频文件放置于Temp文件夹中(客户音频文件是在另一个文件夹中)，在这个文件夹中的音频名对应的是单元格的position，即（节拍数)-(position).wav (此处还要载入一个map来得到每个position对应的节拍数)，而保存和从客户音频列表复制过来时只需针对指定的单元格文件进行复制和重命名操作即可。此时，单元格来检测是否有音频只需判断该单元格position对应的音频文件是否存在即可。具体还有一些其他难点记不太清了因为写的时候是期中左右吧离现在已经几个月了。
- 同步的播放：*(陈澄钧)*
  - 对于同步播放，首先必须要实现的就是音频对齐。由于客户点击播放按钮的时间是不可控的，因此若要让每个音频都能在同一时间开始播放，就需要有一个外部的控制器。我们采用的是一个在activity里设置的节拍器，由计时器控制，对拍子进行循环计数。而此时，如果要让activity去控制每一个播放器的播放，会遇到许多问题（例如资源占用过大导致延迟以及临时音频的增删导致的控制遗漏）。所以我们采取了在每一个音频格的类中添加handler，让其对activity里的拍子进行监听，以此来达到控制的效果。
  - 同时，必须保证音频在每一次循环中的播放时间相同，因此要对音频进行等比例的压缩控制，这里便涉及到了小节数问题，下文再进行细说。
- 录音的同步与录音质量：此处的实现方式与播放同步类似，由于同时只能进行一次录音，因此只需在主程序中添加一个Handler来监听即可。*(陈澄钧)*
  - 由于考虑到客户若是手动结束录音会造成操作上的延迟，因此我们实现了自动结束录音的功能。由于我们已经默认了录音的音乐性，因此认为录制的音频必须满足某节奏型，即其拍子必须是每小节拍子数的整数倍，所以我们实现了让客户设置小节播放速率（bpm）以及录制小节数的功能，以此来实现音频录制的自动结束。
  - 至于录音质量，经过评测感觉wav格式的小段音频录制质量较高，同时曾尝试了一些降噪算法，发现只是整体降低了录音的音量，并没有很实际的降噪功能，因此只在录制麦克风的设置上进行了优化(在电脑模拟器上效果不是很明显，手机端可能会好一点，具体优化方式是参考网上的，比较硬件我们也不是很理解)。
- 循环质量的优化：*(陈澄钧)*
  - 一开始对循环播放的实现我们采用的是mediaplayer中的自带的loop功能，但后来发现，其loop功能在每次结束后需要重新加载音频文件，此时便会产生不可避免的延迟，而这就极大影响了loop的效果。于是我们讨论后决定采用多播放器循环的方法来实现loop功能，即在每一个播放单元格中设立两个mediaplayer，当一个在播放时，另一个会先对音频文件进行加载，而当一进入播放结束后的下一个小节时，便可以马上开始播放。这其中涉及到了对资源的释放问题，由于有多个mediaplayer占用了较大资源，因此对播放完的mediaplayer要做到及时释放，此时用到了一个指针，在交替的时候会指向即将播放完的mediaplayer，从而实现一结束立即释放。
  - 还有一点比较困难的是对于同步播放中的暂停与开始以及音频删减，需要控制多个player，类似于一个多线程问题需要添加很多锁(flag)，比较繁琐此处就不赘述。
  - 同时为了提高音频交替的衔接性(因为在实际演奏中，乐器的声音并不是发出后马上消失，会有余音)，如果直接进行切换，会在衔接过程中有一种空白感，因此我们让每一个录音的时长比理论计算的时长延长了一定时间(很短的)，来使一段音频结束后有种(余音绕梁)的感觉。

- 音频复用：第一点中提到了让每一个音频在每次循环中的播放时间相同，而若能实现这个便能让以不同播放速率录制的音频能在同一种速率下进行同步播放，即音频复用。*(陈澄钧)*
  - 此时最主要要解决的难点就是如何计算理论播放时间，因为一旦有理论播放时间，便可通过其和音频本身的长度来计算得到音频的理论播放速度，通过设置播放速度来实现音频的播放同步。而要计算理论播放时间，需要的是当前的bpm(播放速率)以及音频的小节数，播放速率可以直接由播放设置获得，而小节数便关联到了录音时所设置的小节数。因此我们在录音阶段巧妙地将音频录制设置的小节数以字符形式加到了音频文件名中，然后在显示的时候通过字符串处理隐去，而在读取音频文件时却可以对通过文件名进行解析，得到该音频的小节数，这样便实现了真正的音频复用。

设计框图



项目效果

- 见附件demo1，demo2，demo3

项目使用

使用环境

- 安卓模拟器：Nexus 5 API 25(其他版本的模拟器不能吻合UI)
- Tomcat 8.5

使用步骤

- 将 `TestWeb.war` 放入Tomcat的webapps目录下
- 在bin目录下运行命令 `startup.bat` 启动项目
- 访问"localhost:8080/TestWeb/TestWeb"检验项目是否启动成功
- 若项目启动成功，将 `Loop.apk` 拖拽入模拟器中进行安装，安装完后在模拟器中设置该项目的权限
- 参照demo进行操作