音频剪辑播放器实验报告

71115115 王子卓

问题定义

- 各大安卓应用市场均未有令人满意的音频剪辑软件，这些软件失败的原因有很多种。一些软件编解码速度慢，一些软件浪费系统资源，还有一些软件不能很好的运行在较高的安卓版本。
- 因此本人萌生了开发一个音频剪辑器的念头，旨在提高自己的编程水平并且方便他人。

软件需求

- 制作一个可以播放音频、剪辑音频的安卓软件。
- 该软件必须具备快速的编解码速度(调用FFmpeg库)、人性的用户化界面以及节省用户空间的特性。

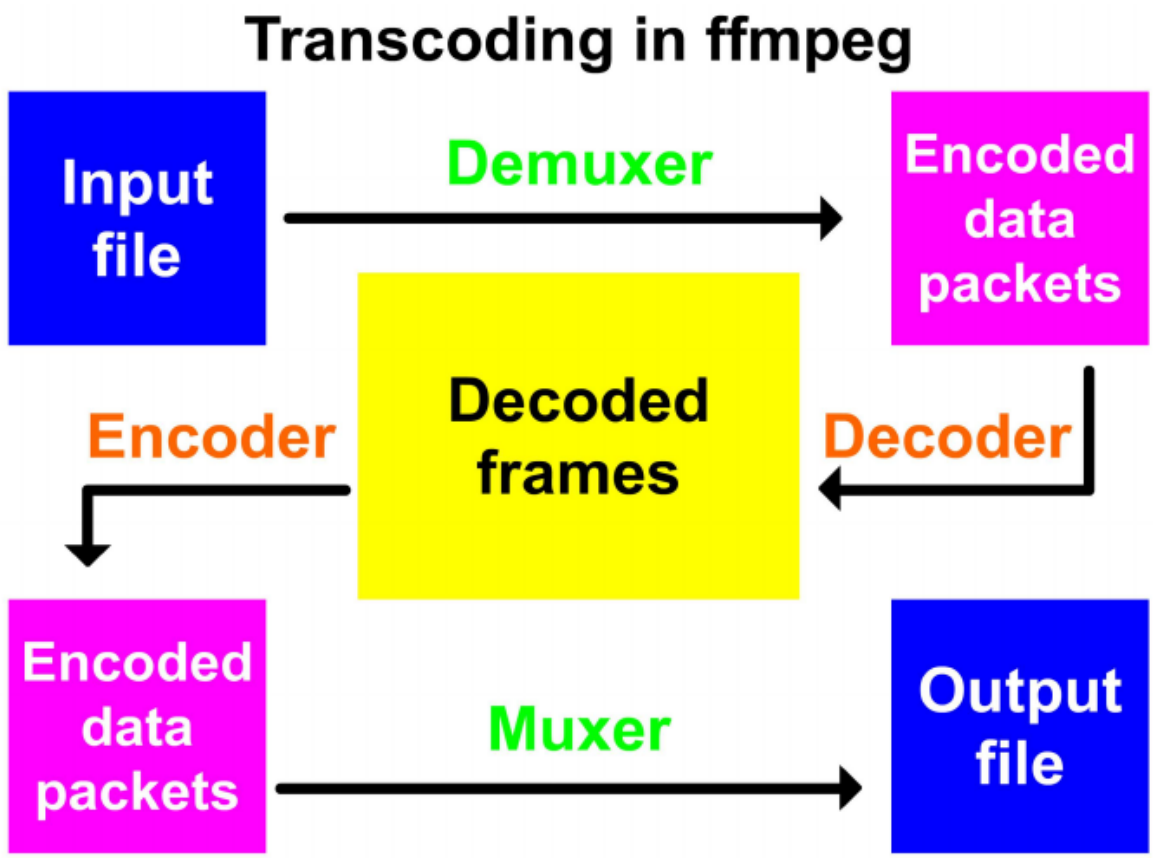
软件体系结构设计

软件重用

- 使用已经较为完善的FFmpeg音频处理库，在已有工作的基础上，充分利用以前别人开发的模块 ，将开发的重点转移到现有系统的特有构成成分。

管道-过滤器风格

- FFmpeg库编辑音频采用了管道-过滤器风格。



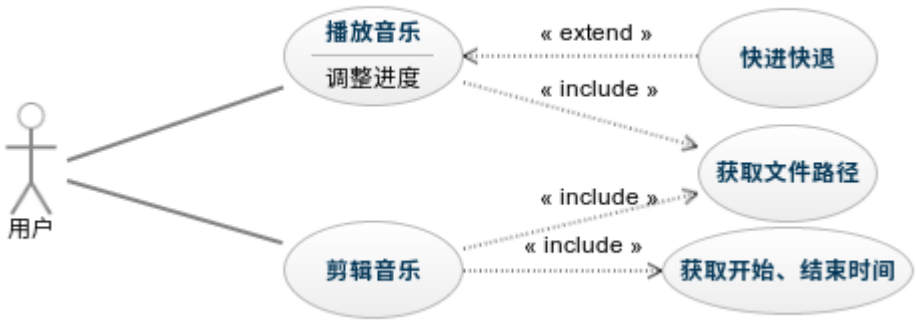
层次系统风格

- FFmpeg构成程序的底层，为作为上层的图形化界面提供剪辑服务。FFmpeg层为上层提供了调用的接口，FFmpeg的底层实现改变不会影响上层的调用，为软件重用提供了支持。

MVC风格

- 用户只操作控制器构件，看视图构件的显示，不去碰模型构件里面的应用程序核心代码。

场景视图



低耦合高内聚

- 采用数据耦合，主活动调用FFmpeg模块，通过简单的数据参数来交换消息。主活动将完成剪辑所需命令参数传递给FFmpeg模块，FFmpeg将执行该命令的结果以字符串的形式动态的更新主界面。
- 采用功能内聚，主活动和FFmpeg模块都实现单一功能，二者协同工作，缺一不可。

信息隐藏

- 本软件对用户隐藏部件的实现细节，更好的处理系统的复杂性和减少各模块之间的耦合。

封装

- 封装FFmpeg模块，使之与主活动之间形成明确的界限，保护内部信息不被破坏。

软件设计

- 用户在点击“打开文件”按钮时，弹出手机中装有的文件管理器，让用户选择文件。选择完毕之后返回音频路径，初始化播放器。
- 用户点击”播放“按钮之后音乐开始播放，分别选择起始时间和终止时间，之后点击”开始“按钮开始剪辑。

软件实现

- 处理信息，当用户要打开文件时，获取路径并且开启一个新线程，用来使播放器更新进度。

```
Handler handler = new Handler(){
        @Override
        public void handleMessage(Message msg){
            if(msg.what == FILE){
                MainActivity.this.directory.setText(uri.getPath().toString());
                player = MediaPlayer.create(MainActivity.this,uri);
                seekBar.setMax(player.getDuration());
                total.setText(getTime(player.getDuration()));
                time = new Thread(){
                    public void run(){
                        while (true){
                            boolean isPlaying = player.isPlaying();
                            while(isPlaying){
                                int pos = player.getCurrentPosition();
                                seekBar.setProgress(pos);

                                Message m = new Message();
                                m.what = UPDATE;
                                handler.sendMessage(m);
                                try {
                                    Thread.sleep(100);
                                } catch (Exception e){
                                    e.printStackTrace();
                                }
                            }
                        }
                    }
                };
                time.start();
                MainActivity.this.total.setText(getTime(player.getDuration()));
            }
            if(msg.what == UPDATE){
                current.setText(getTime(player.getCurrentPosition()));
            }
        }
    };
```

- 调用FFmpeg模块

```
String rawCMD = "-ss "+start.getText()+" -t "+duration.getText()
                            + " -i "+directory.getText()
                            + " -acodec mp3
/storage/emulated/0/MediaSplitter/output.mp3" ;
                    String[] cmd = rawCMD.split(" ");
                    if(cmd.length!=0){
                        try{
                            ffmpeg.execute(cmd,new ExecuteBinaryResponseHandler ()
{
                                @Override
                                public void onStart() {}

                                @Override
                                public void onProgress(String message) {
                                    output.append(message+"\n");
                                }

                                @Override
                                public void onFailure(String message) {
                                    output.append(message+"\n");
                                }

                                @Override
                                public void onSuccess(String message) {
                                    output.append(message+"\n");
                                }

                                @Override
                                public void onFinish() {}
                            });
                        } catch (FFmpegCommandAlreadyRunningException  e){
                            Toast.makeText(MainActivity.this,"一个一个来哦!",
Toast.LENGTH_SHORT).show();
                        }
                    }
```