**手势识别项目总结**

# 项目结构：

分为Android客户端和WPF windows 服务端。

# 版本控制：

GitHub

# Repository地址：

Android端：https://github.com/BigDipper7/Gesture-Recognition-Android-Client-Side

WPF端：https://github.com/BigDipper7/Gesture-Recognition-WPF-Server-Side

# WPF 服务端：

## 功能：

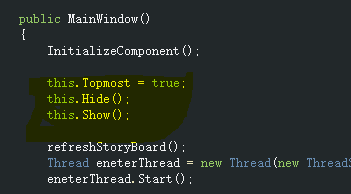
1. 根据Android端传来数据显示对应动画并播放音效
2. 根据传感器手势显示对应动画并播放音效

## 采用技术：

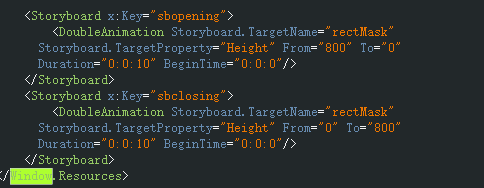
1. 采用WPF来制作全屏显示的windows 应用程序。
2. 动画采用Storyboard 和 DoubleAnimation 来实现动画显示，DoubleAnimation控制黑色幕布进行伸缩遮盖天空背景。
3. 服务端客户端通讯采用以WPF作为服务端的方式的WebSocket进行消息通讯，消息体为纯文本，未做加密。
4. 声音播放采用线程安全的MediaPlayer的方式，可播放多种格式。
5. 状态转化控制选用状态机

## 关键技术介绍：

1. 采用WPF进行windows编程，根据需求，在程序启动时设置为：程序全屏、永远保持在其他应用最前端。同时整体程序画布设置为固定的1280 \* 800 的大小，为landscape的方式。

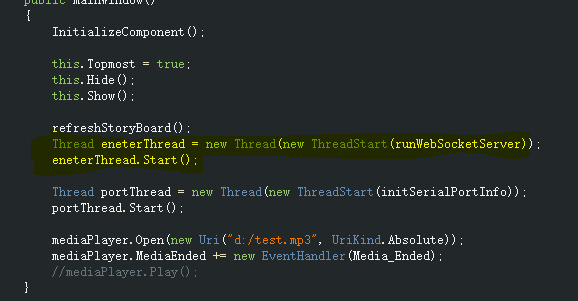


1. 采用Storyboard和DoubleAnimation，制作幕布遮盖和开启的线性动画。其中为了处理方便以及考虑到Storyboard动画控制端的特殊性，同时设置了开启和关闭两个动画，经由状态机根据命令来处理动画的start、stop、pause、resume的操作，同时需要注意的是当动画resume时会根据开启动画方向不同需要同时设定动画的starttime。



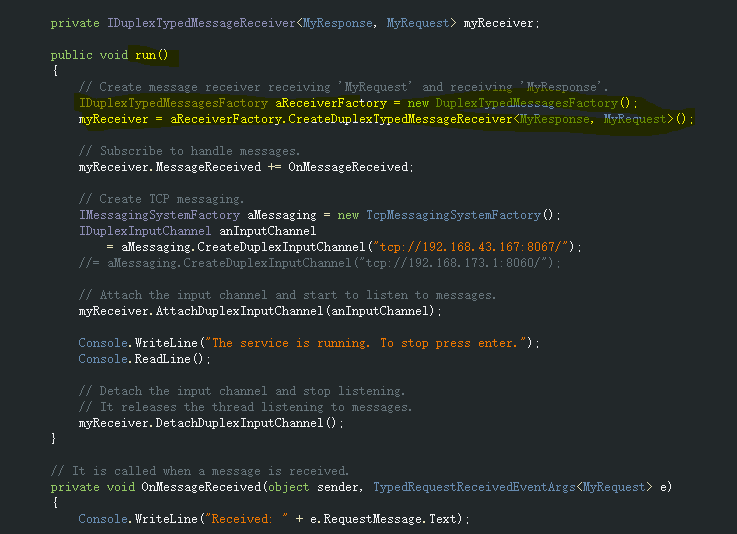
1. Android端和WPF端采用WebSocket的通讯方式。消息体数据采用纯文本，不加密。建立以WPF作为服务端的中心网络，支持一机多联，同时由于状态机存在保证多用户之间的操作的安全性。

其中WPF端websocket采用了Fleck 框架，简单快捷，在启动windows窗体的时候同时开子线程启动Websocket服务器，但是也需要用特殊手段在子线程中通知前端windows窗体的动画改变。

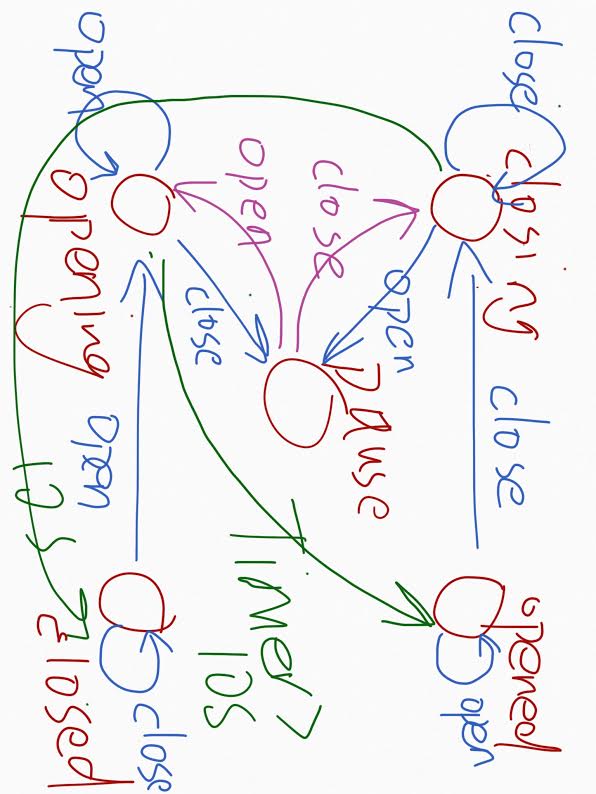




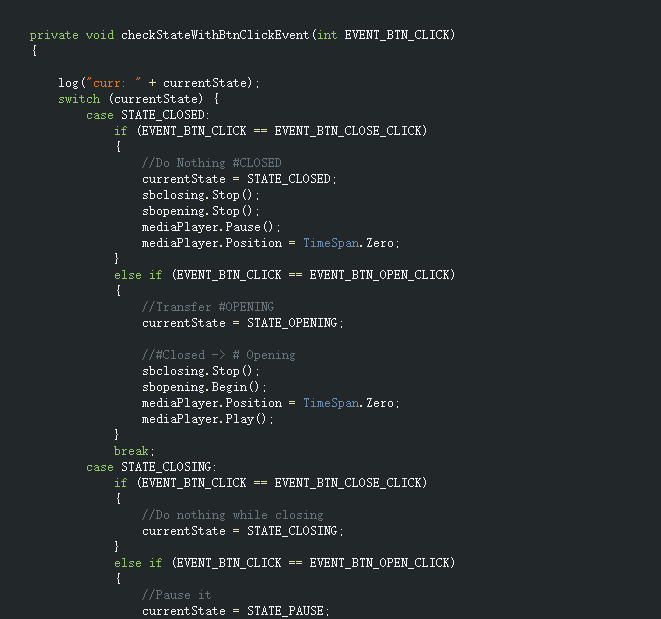
曾采用Eneter框架实现.Net和Android端通讯，但是由于框架的极其不稳定，重构为Fleck框架。



1. 状态机保证状态转化的准确性，共五种状态两种操作，状态转化图如下



在状态机中实现动画开启暂停。



1. COM3接口的串口数据接收。直接采用C#的SerialPort直接接收串口数据。同时将接收到的消息处理判断后输入状态机函数，实现状态迁移。同时也采用程序启动时开启子线程尝试连接COM3的串口，连接成功后持续监听该端口。

## 项目部署：

1. 先从GitHub上拉取repository。
2. 导入VS（推荐VS2010），检查reference下是否导入了Fleck框架和Eneter框架（为保险起见，最好都导入），Fleck可以采用NuGet安装，Eneter采用手动import的方式即可。
3. 编译及打包发布
4. 注意：由于客户方要求，声音文件需多次变动，而打包发布后的程序会将声音文件一同打包编码，所以选用hardcode到路径：“d:\\test.mp3”为我们的目标音频，建议采用10s音频，不添加音频程序不报错，但播放动画时无声音。

# Android客户端：

## 功能：

1. 和服务端建立Websocket连接，并向其发送开启关闭的纯文本指令。

## 采用技术：

1. 采用java\_websocket进行Websocket的建立和链接以及消息接受。
2. 设计逻辑回显错误以及服务链接状态显示。

## 关键技术介绍：

Android比较简单，略。

## 项目部署：

1. 从Github上拉取repository。
2. 导入AndroidStudio。
3. 编译及打包发布加签。