## 10+版本迭代

1. 简单介绍一下这个项目？

Royole是做柔性传感器（FS）和柔性显示器（FD），柔记手写本就是基于柔性传感器（FS，电容式）的手写设备，将写在纸上的笔迹转换为轨迹坐标数据，柔记APP通过蓝牙BLE与手写本连接，将传输到手机上轨迹坐标数据还原为用户笔记。还有笔记分享，转换图片、微视频，回放，笔记拆分，合并，笔记搜索、笔记本管理、安全连接、更换画笔，笔记转文本、笔记云同步等功能。

柔记APP公司唯一个互联网化的APP，使用的技术栈跟主流的互联网APP相同，上架了国内各大主流应用市场以及谷歌市场。大概进行了10多个版本迭代，开发流程相对规范，团队大小适中，也有一定的用户量。

代码量大概10万行，java文件，

1. 项目成员有多少人？你在这个项目担当什么角色？

项目是从17年10月开始的，我是18年4月加入的。前期项目快速迭代，多的时候6个人左右，目前迭代速度变慢，两个人在小功能的开发与维护。

本人在项目中的角色是安卓端开发负责人，主要工作包括需求评审、工作量评估、技术评审、编码实现、代码审核、版本发布等工作。

1. 在这个项目中又遇到什么让你印象深刻的事情？

1）柔派适配，风险评估与处理；

在一个版本迭代过程中，老板希望在一个展会上使用柔派手机来展示柔记APP。 柔派手机是公司的一款折叠手机，因为屏幕可以折叠，所以屏幕有三种分辨率：大屏、小屏（主屏、辅屏）。在App运行过程中折叠手机屏幕，屏幕分辨率变化，Activity会销毁重建，一些用户操作就会被打断。所以必须对柔派进行适配。当时就把柔派手机的适配工作交给一个同事去做。当时觉得就是屏幕适配，所以也不觉得会有什么风险。中间有多次问那位同事，是否有风险，回复说没有问题，直到发版前一周，发现不断有适配导致的问题冒出来。才发现问题不是那么简单，在折叠过程中，由于屏幕分辨率会发生变化，导致Activity会销毁重建，在开发过程中并未考虑Activity销毁重建的情况，并未对运行时数据进行保存，导致一些运行时的状态会丢失；同时有一些用户操作是不允许会被打断，比如绘制过程中，折叠一下，所有的新绘制的笔记都丢失了。整个App总共20个页面左右，这样引发了很多的问题。当时两个开发人员，在没有现成解决方案的情况下，一周时间无论如何都无法完美适配。所以发邮件给项目负责人，告知风险，寻找解决方案。

解决方案：

1. 短期（一周内）适配常用的界面；
2. 适配方法：简单的页面，可打断的页面，走销毁重建的逻辑，在onSaveInstanceState中保存状态数据，在onRestoreInstanceState中恢复状态数据；复杂的页面，有很多中间状态的页面不销毁重建，在onConfigurationChanged回调中对视图元素的布局参数进行调整。

MVP+ViewModel框架，将业务逻辑与运行时数据放到ViewModel中，由于ViewModel在Activity因分辨率变化而导致的销毁重建并不会销毁，所以用ViewModel作为Presenter，保存运行时数据。再配合LiveData可以在Activity销毁重建之后恢复销毁前的状态。

2）笔记本同步，流程设计与实现；

V2.5.0 版本为了适配最新的柔记二代手写本硬件的外观设计。将APP进行了重构，更加强调笔记本的概念。增加了笔记本的管理，如：设置默认笔记本、设置常用笔记本、对笔记本进行排序、修改笔记本名称、封面等。对默认笔记本的同步流程进行梳理，绘制流程图，提供给IOS、服务器端编码实现。这部分的难点在于：默认笔记本多端修改冲突、默认笔记本切换等。

3）手写本连接库，过程改进；

手写本与APP之间是通过蓝牙BLE连接的，这部分的逻辑相对独立，就专门封装了一个SDK，专门处理手写本与APP之间的连接通信。手写本与APP之间的通信包括，扫描手写本、获取存储信息、获取电量信息、分页获取存储数据、删除数据、获取固件版本号、设置/获取手写本的模式（实时or离线）、安全连接逻辑等。这个连接库是一个独立的工程，因为这个连接库在供内部使用之外，还提供给外部如方正公司使用。一开始APP使用这个依赖库的方式是，直接编译出jar，手动拷贝到柔记APP工程中。但这样手动拷贝容易出错，中间就有一次开发人员修改了连接库，但是忘记了拷贝jar到APP工程中，导致出错。后面将依赖库发布到Maven私服上，APP工程通过maven的方式来引用。

4）柔记项目Https安全传输总结；

见 **柔记项目Https安全传输总结.ppt**

1. 项目是从零开始做的吗，你是什么时候加入到这个项目的？

项目是从17年10月开始的，我是18年4月加入的。

1. APP的日活，用户数等有多少

日活用户大概2000左右；每天新增用户数大概是500左右；

## 组件化

1. 为什么要组件化？
2. 业务模块之间耦合度太高，项目难以扩展与维护；
3. 任何一个改动都要编译整个项目，影响开发效率；
4. 组件化有哪些方案？

基于ARouter的组件化方案：路由+接口下沉

基于ActivityRouter的组件化方案：路由+静态方法

得到DDComponentForAndroid：路由+接口下沉

CC、ModularizationArchitecture：组件总线

聚美组件化方案，基于聚美Router：路由+接口下沉

1. 为什么选用ARouter方案？

1）阿里出品，使用者众多，QQ群里交流比较活跃；

2）支持分级按需加载；

3）是一个路由框架，并不是完整的组件化方案，可作为组件化架构的通信引擎；

4）跟其他组件化方案相比，更为灵活；

1. 组件化主要关注点？

a.业务解耦，识别业务边界，整理业务之间的交互通信，通过路由来跳转和暴露服务；

b.用gradle的属性来切换组件模式还是集成模式。

c.服务接口下沉，暴露组件服务。

1. 你是如何实现组件化的？

识别业务边界，拆分组件，组件间的界面跳转通过路由实现

组件的对外服务接口下沉到依赖库

实体下沉到依赖库

一些通用的工具类事件总线等下沉到依赖库；

1. 说说路由的实现原理？？？

如何实现服务的暴露？

如何实现页面的跳转？

1. 你这个项目分了几个模块？

分成了5个：登录、笔记书写、手写识别、商城Web、笔记分享；

登录：注册、登录、修改密码、三方绑定与登录、个人信息等；

笔记：笔记绘制、展示、管理、拆分、合并、移动、更换画笔、回放等；

手写识别：将笔记转换为文本；

Web：商城、小教程、用户协议、隐私策略等需要用web进行展示的功能；

## 引入MVP+ViewModel架构；

1. 介绍一下MVP架构，与MVC、MVVM的区别与联系

MVC：MVC（Model-View-Controller，模型-视图-控制器），一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

* **Model（模型）：**处理应用程序数据及业务逻辑的部分。
* **View（视图）：**处理数据显示的部分，将模型数据展示给用户。
* **Controller（控制器）：**响应用户输入，负责从视图读取数据，并向模型发送数据。

MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注程序的一个方面。具有耦合性低、重用性高、可维护性高等特点。

在Android中，Controller对应的是Activity，而Activity中却又具有操作UI的功能，我们在实际的项目中也会有很多UI操作在这一层，也就是做了很多View中应该做的事情，当然Activity中也包含Controller应该做的事情，比如各种事件的派发回调，以及业务逻辑处理等。导致Activity代码臃肿，各层次之间耦合比较严重。

MVP（Model-View-Presenter，模型-视图-表示器）从MVC演变而来，它们的基本思想有相通的地方：Controller/Presenter负责逻辑的处理，Model提供数据，View负责显示。

* **Model（模型）：**负责存储、检索、操纵数据。
* **View（视图）：**负责绘制UI元素、与用户进行交互。
* **Presenter（控制器）：**作为View与Model交互的中间纽带，处理与用户交互逻辑。

特点：

* View与Model不直接交互，完全解耦
* 模块职责分明，层次清晰
* 实现Presenter的复用
* View与业务分离，可以实现组件化

Model:

* 从网络，数据库，文件，传感器，第三方等数据源读写数据
* 对外部的数据类型进行解析转换为APP内部数据交由上层处理

View

在presenter的控制下修改UI

将业务事件交由presenter处理

View层不存储数据，不与Model直接交互

View是被动的显示：

我接收到用户的输入，现在交给你，你看要怎么处理，如果需要我的协助就告诉我。

我接受到用户的输入，我现在需要你的协助，因为Model只跟你打交道。

Presenter是控制的中心，View只是辅助，View中只处理简单的判断筛选逻辑，不应该有业务相关的操作。

显示用户信息UserInfo

\*View interface:需要View实现的接口，View通过View interface与Presenter进行交互，降低耦合，方便进行单元测试

\*Presenter interface:需要Presenter

MVP的缺点：

* Presenter中有大量View与Model之间的交互，会导致Presenter代码臃肿
* View与Presenter联系紧密，一旦View变动，Presenter也要更改

1. 为什么要引入这个架构？

**Android应用的特点**

* 应用组件不按固定的顺序启动，可以单独启动
* 应用组件随时可能被终止

**应用架构原则**

1. 分离关注点

* 将程序分成不同部分，每个部分解决一个单独的问题
* 程序模块化和信息系统的分层设计都是分离关注点思想的应用

2）模型驱动界面

* 如果 Android 操作系统销毁应用以释放资源，用户不会丢失数据。
* 当网络连接不稳定或不可用时，应用会继续工作。

**编写Android应用应该注意什么**

* 组件之间不可以相互依赖
* 不可以在应用组件存储应用的数据和状态

1.分离关注点

TCP/IP协议将计算机网络分为链路层、网络层、传输层、应用层，每一层关注/处理不同的问题，将复杂的网络问题拆分成若干易于解决的问题。

2.模型驱动界面

由于Activity/Fragment不可靠、随时可能被销毁，应该将应用的业务逻辑与数据与界面拆分开来。当Activity/Fragment恢复时，你的应用能继续运行。

常见的问题是，在Activity中处理所有的业务逻辑，Activity代码臃肿，Activity销毁时，所有的数据状态都都丢失了。

Activity、Fragment只是android系统提供给开发者使用系统服务，比如显示、接收用户输入、打开外设（蓝牙）的“窗口” ，这些“窗口” 有可能随时会关闭，开发者不应该把自己的业务逻辑放到这些不稳定的“窗口” 中。

1. 跟普通的MVP有什么不同的地方？

不同的地方在与我们Android的ViewModel来实现Presenter，Presenter继承自ViewModel。

在Activity或Fragment因配置改变被系统异常销毁时，ViewModel不会被销毁，如果我们将ViewModel作为Presenter，作为应用的业务中心，用它来保存用户数据，我们的数据和状态就不会随着Activity和Fragment的销毁而丢失。

1. 说说ViewModel是怎么实现应用数据保存的？

当屏幕旋转的时候，Activity会被销毁重建，Activity会经过几个生命周期方法，但是这个时候ViewModel还是之前的对象，并没有被重新创建。

当Activity正常结束时，ViewModel.onCleared()方法会被调用，ViewModel对象才会被销毁。

ViewModel的生命周期比Activity的生命周期长，原则上ViewModel不应该持有Activity的引用。

1. 这个所谓的框架，只是用ViewModel来做Presenter，你在这里做了什么工作？为什么叫框架？

## UI适配 pt适配

1. 介绍一下怎么适配的？

一开始是采用百分比适配的方法，百分比适配的原理：

1. 将UI设计稿（1080X1920）缩放到目标屏幕，宽高上每个刻度对应的像素值生成到lay\_x.xml和lay\_y.xml。
2. 系统根据屏幕分辨率选择对应的lay\_x.xml和lay\_y.xml的尺寸值。

问题：

1. 当目标屏幕的宽高比与UI稿的宽高比不一致时，x、y方向缩放系数不一致，界面会出现变形。
2. 新出一种尺寸的屏幕的时候，需要制作一份对应的尺寸资源，否则使用默认的尺寸资源。

今日头条适配方案：根据设计稿的尺寸与目标屏幕的尺寸，计算缩放系数，修改DisplayMatrix的density、densityDpi、xdpi等变量，以实现一套UI尺寸适配所有机型屏幕尺寸。

今日头条适配方案特点：

* x、y方向的缩放系数都是一致的，不存在界面变形的问题
* 只需一套尺寸资源
* 适用于新项目

1. 官方适配方法？

为不同的屏幕分辨率、屏幕密度生成不同尺寸文件，放在不同的value文件夹下。在不同的分辨率的手机上，使用不同资源目录下的资源。

1. 百分比适配，今日头条适配方案对比，开源库AndroidAutoSize

百分比适配：1）需要为每一种屏幕分辨率生成一种尺寸资源；2）当屏幕分辨率与设计稿的尺寸宽高比例不一致时，图片会被拉伸变形；3）每新增一个屏幕分辨率的设备时，需要重新生成尺寸资源；

## 引入Checkstyle，编码规范

1. 为什么要引入Checkstyle？

为了执行统一的编码规范，配合gitlab，可以让团队成员强制执行编码规范检查，不符合编码规范的代码，不可以合并到仓库中。Checkstyle可以自定义编码规范扫描规则。

1. 如何执行code review？

分为两步：

1. 代码提交到gitlab服务器，会自动触发编码规范扫描任务，如果发现有不符合编码规范的代码，不允许合并入仓库。
2. 代码通过自动化编码规范扫描任务之后，

代码提交人员需指定审核人员进行审核，审核通过之后代码才允许合并入仓库。

Gitlab可以指定代码团队成员，如代码维护者、开发者等，可以给不同成员分配不同的权限，如维护者才有权限合并代码。

1. Gradle插件；

自定义编码规范插件。可以在AndroidStudio的gradle task中找到并执行，当然也是可以在命令终端中执行。

1. gitlab、git工具的使用；

gitlab是一个代码仓库管理系统，类似github，以git为代码管理工具为基础搭建起来的web服务，它集成了issue跟踪、代码合并与审核、持续构建与集成等功能。

代码上传到gitlab服务器会触发构建活动，在项目根目录的yaml文件中，编写构建活动脚本。

## 接入LeakCanary Refwatcher 检测ViewModel内存泄漏。

LeakCannary的原理；

如何检测ViewModel的内存泄漏；

## 引入Tinker热修复，Bugly

1. 已经弃用、谷歌Play市场已经禁止，为什么

主要是不利于管控，应用在上架的时会进行合规性审核。合规性的意思是指，是否有暴力黄色内容、是否违反隐私协议等。如果通过动态化技术更新了APP之后，使得它无法管控这些APP。

1. 主流热修复方案原理及对比；
2. Tinker热修复方案原理；

BSDiff生成增量更新文件，APP启动，在onAttachBaseContext方法中，检测到增量更新文件，会合成一个完整的class.dex文件。

1. Java的类加载机制；
2. Android中的类加载器；

## 笔记绘制

二阶贝塞尔曲线

双指缩放

## 蓝牙通信

安全连接逻辑；

自动重连机制；

## 笔记云同步

OkHttp+Retrofit

采用ProtoBuffer，节省流量。

Https证书校验提高数据安全性；

## 生成笔记视频

如何转换？

1. 将轨迹的总轨迹点数除以每秒绘制点数（210），得到转换视频的时长videoTime

基础帧率：15帧/秒

videoTime = totalPoint / 210

1）10秒短视频：

帧率 10\*15 – 17；

每帧的点数 = （tatalPoint + 10 \* 15 – 17 - 1）/ （10 \* 15 – 17）

1. 其他视频
2. > 3min 压缩到3min播放 每帧点数 =Math.round(totalPoint / (360f \* IVideoConfig.FRAME\_RATE));
3. > 1min 快速播放 每帧点数 =（210 / 15) \* 2
4. < 1s 慢速播放 每帧点数 = totalPoint / 20 > 0 ? (int) (totalPoint / 20) : 1;
5. >1s && <=1min 正常播放 每帧的点数 = 210 / 15

根据总点数和每帧点数计算出总帧数。

每间隔一定的时间，将bitmap中的argb数据通过TransYUV.rgb2yuv420sp和TransYUV.rgb2yuv420p转换为yuv编码数据。

然后将yuv编码数据通过MediaCodec，编码成h264格式的视频数据。

最后，通过ffmpeg将h264码流转换成MP4格式的视频文件。

H264具有更高的编码效率。

## 手写识别，跨进程通信

为什么要跨进程？

Android实现进程间通信的方法？

Messenger：

Messenger是基于AIDL实现的，服务端（被动方）提供一个Service来处理客户端（主动方）连接，维护一个Handler来创建Messenger，在onBind时返回Messenger的binder。

双方用Messenger来发送数据，用Handler来处理数据。Messenger处理数据依靠Handler，所以是串行的，也就是说，Handler接到多个message时，就要排队依次处理。

AIDL：

AIDL通过定义服务端暴露的接口，以提供给客户端来调用，AIDL使服务器可以并行处理，而Messenger封装了AIDL之后只能串行运行，所以Messenger一般用作消息传递。

通过编写aidl文件来设计想要暴露的接口，编译后会自动生成响应的java文件，服务器将接口的具体实现写在Stub中，用iBinder对象传递给客户端，客户端bindService的时候，用asInterface的形式将iBinder还原成接口，再调用其中的方法。

文件共享：

对同一个文件先后写读，从而实现传输，Linux机制下，可以对文件并发写，所以要注意同步。顺便一提，Windows下不支持并发读或写。

Bundle/Intent传递数据：

可传递基本类型，String，实现了Serializable或Parcellable接口的数据结构。Serializable是Java的序列化方法，Parcellable是Android的序列化方法，前者代码量少（仅一句），但I/O开销较大，一般用于输出到磁盘或网卡；后者实现代码多，效率高，一般用户内存间序列化和反序列化传输。

ContentProvider：

系统四大组件之一，底层也是Binder实现，主要用来为其他APP提供数据，可以说天生就是为进程通信而生的。自己实现一个ContentProvider需要实现6个方法，其中onCreate是主线程中回调的，其他方法是运行在Binder之中的。自定义的ContentProvider注册时要提供authorities属性，应用需要访问的时候将属性包装成Uri.parse("content://authorities")。还可以设置permission，readPermission，writePermission来设置权限。 ContentProvider有query，delete，insert等方法，看起来貌似是一个数据库管理类，但其实可以用文件，内存数据等等一切来充当数据源，query返回的是一个Cursor，可以自定义继承AbstractCursor的类来实现。

Binder AIDL

广播Broadcast

切换APP语言，采用广播通知不同进程修改APP语言；

这里时是如何实现进程间通信的？

## 团队角色

1. 负责柔记项目，还有参与一些小项目：永兴元定制版柔记、练字系统、签名系统、会议系统、OneUI、一拖多项目、残影修复项目、电梯项目等。
2. 技术培训

RxJava

UI适配&应用程序模版工程介绍

FlexAnimator SDK的使用

Git & gitlab的使用；

1. 问题解决
2. Code Review