变更列表

序号	记录	时间	版本	负责人
1	初版	2020/3/19	V1	王丽芳
2	更新多屏切换动画设计	2020/3/27	V2	王丽芳
3	APP 启动 Flag 增加 Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT 说明，支持已经启动的 activity 显示到其他的 display 上 添加 taskAffinity 的说明	2020/6/19	V3	王丽芳
4	添加被启动应用整体移动的说明	2020/8/3	V4	吴正雪

一、验证环境

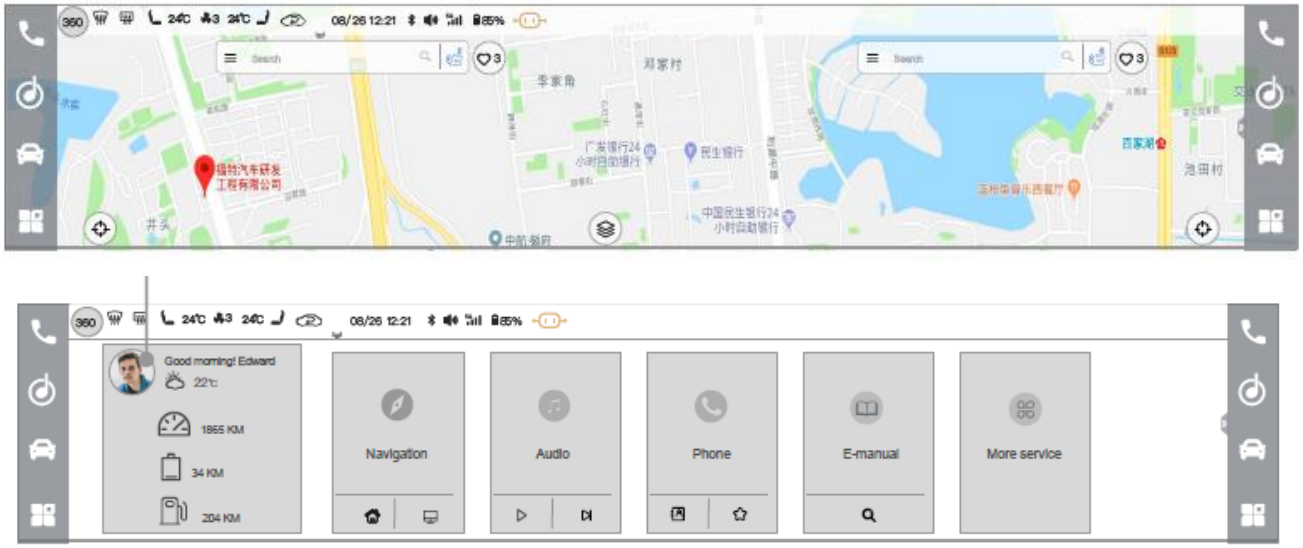
硬件平台：高通 S820A
操作系统：Android 8.1
开发语言： JAVA、C 、C++
屏幕尺寸：13 寸 2384*1080(还没有拿到 27 寸的 display)

二、独立驾驶模式切换方案

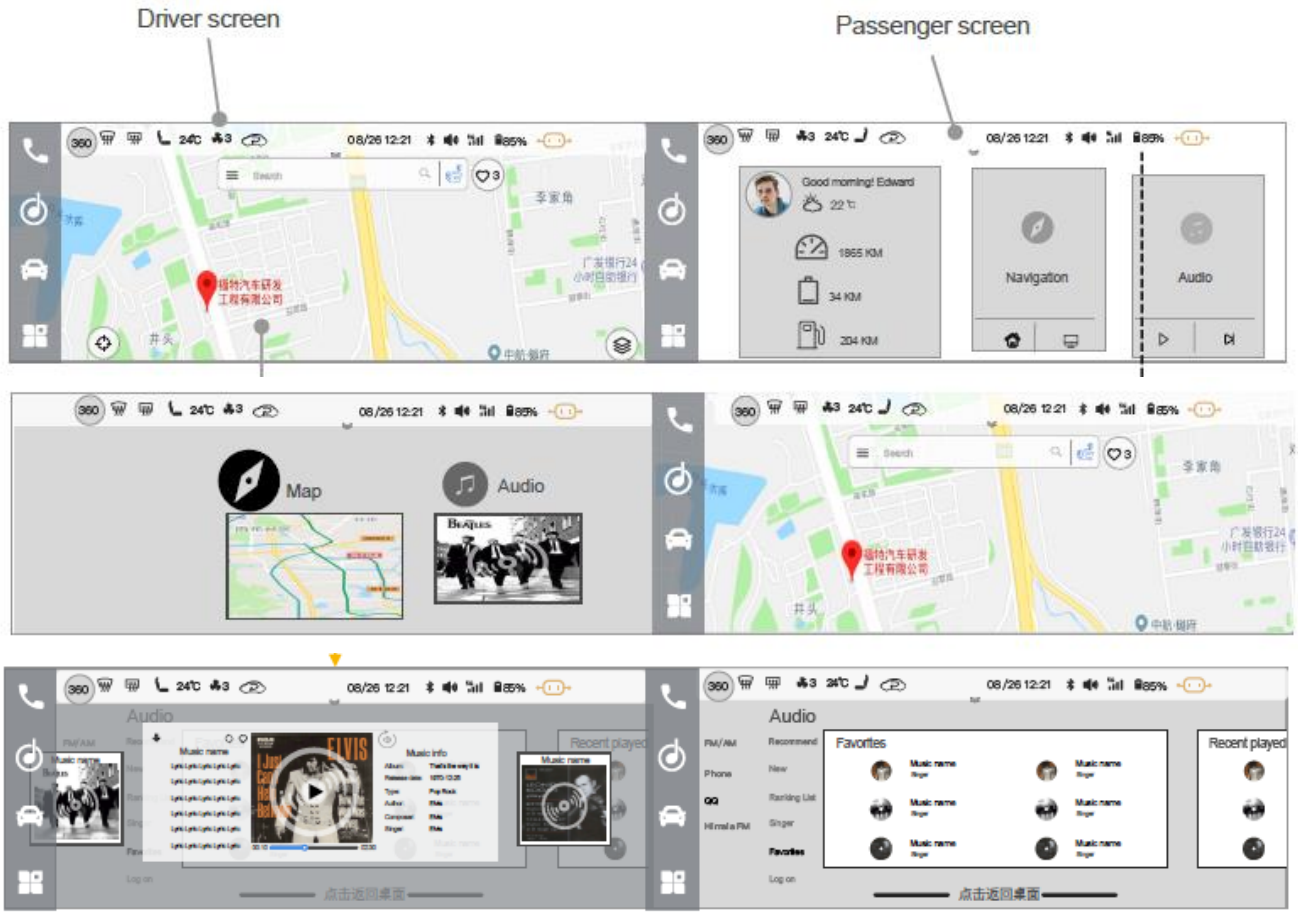
需求

1、独自驾驶模式(Solo mode)

2、合作驾驶模式(Co-pilot mode)



3、独立模式(Individual mode)



多屏方案

为了满足独立驾驶模式和其他模式的切换需求，以及切换动画效果，切换过程对系统的性能负载（memory/CPU/GPU）的影响，我们设计如下的方案：

1. 系统启动的时候会创建一个虚拟屏（virtual display），不可见，并启动 second launcher 到这个 display 上

2. 当进入独立驾驶模式时，根据切换动画效果，展示 virtual display，动画效果参考 relayout_ok.mp4； 由于在设计的切换效果中，小屏上的应用也有动画展示效果，这个要进一步确认如何触发小屏的动画。

3. 同时系统对大屏上的应用进行 stack resize（可以根据应用是否需要 relayout 来配置是否对应用进行 resize），系统在 resize 之后给应用发送 onConfigurationChanged 的回调，应用在该回调中根据 screensize 设置不同的 layout

4. ~~配置是否需要 resize 的方式是给 Activty 配置 meta-data，如果不配置，系统默认会 resize~~

```
<activity android:name=".TestActivity"
        android:configChanges="screenSize|smallestScreenSize|screenLayout|orientation"
        >
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <meta-data
            android:name="resize_display"
            android:value="false"/>
    </activity>
```

5. 为了平滑显示动画效果，不出现切换过程中的黑屏，在分屏的时候会先 show 小屏然后 resize；在二屏合一的时候先 resize，然后 dismiss 小屏；这个过程可以保证小屏的 show/dismiss 是平滑和流畅的，但是屏幕左侧的显示，由于应用会 relayout，如果这个过程比较快或者 relayout 前后该区域显示没有变化，那么看上去比较流畅，否则可能会出现视觉 gap，例如在图片 gallery 界面，会不会出现图片 gallery 的一步步加载，这个系统无法控制，需要应用优化。

6. 下图所示的 relayout_ok.mp4, 显示了切换时候的动画，系统对 TestActivity 执行了 resize, 由于 TestActivity 布局简单，所以很快就呈现了 relayout（布局文件更改为 test_half.xml ）后的界面，看上去比较流畅，，日志中可以看到 onConfigurationChanged 被触发。


relayout_ok.mp4

TestActivity onConfigurationChanged：

```
01-
01 13:14:11.517  8203  8203 D TestActivityTest: configuration changed:{1.0 ?mcc?mnc [zh_CN] ldltr sw1080dp w1174dp h1024dp 160dpi xlrg land car finger qwerty/v/v dpad/v appBounds=Rect(0, 0 - 1174, 1024) s.5};for:com.demo.launcher.TestActivity@5f8a998

```

7. 为了防止 relayout_ok.mp4 误导大家，让人以为布局没有做任何变化,这里再提供一个在 onConfigurationChanged 里面不改变布局（布局文件是 test.xml）的对比视频：


relayut_nochange
.mp4

Relayout_nochange.mp4 和 relayout_ok.mp4 的对比可以让大家意识到在 onConfigrationChanged 中对布局进行更新对视觉效果的影响。

8. 另外再提供一个布局对视觉产生影响的视频来说明应用的 UI 布局对视觉效果的影响：

relayout_gap.mp4

这个是以原生的 gallery 应用演示的分屏效果，从全屏展示到半屏展示发生了一定的偏移，这是因为应用计算布局的时候没有考虑到左边视图尽量保持不变的原则，这个是我们的应用需要避免的，是做 UI 设计的时候需要考虑的问题。

9. 对于应用关心的分屏之后 **relayout 时获取的屏幕大小**问题，我测试了几个函数都没有问题，都可以获取到期望的 screen size（物理分辨率的一半），MATCH_PARENT 也没有问题，参考如下信息：

```
@Override
    public void onConfigurationChanged(Configuration newConfig)
    {

        ///01-01 13:14:11.517  8203  8203 D TestActivityTest: configuration changed:{1.0 ?mcc?mnc [zh_CN] ldltr sw1080dp w1174dp h1024dp 160dpi xlrg land car finger qwerty/v/v dpad/v appBounds=Rect(0, 0 - 1174, 1024) s.5}
        Log.d(TAG, "configuration changed:" + newConfig + ";for:" + this);
        super.onConfigurationChanged(newConfig);


        ////01-01 13:14:11.517  8203  8203 D TestActivityTest: screen width: 1174;screnheight:1024
        Log.d(TAG, "screen width: " +newConfig.screenWidthDp + ";screnheight:" + newConfig.screenHeightDp);
        if(newConfig.screenWidthDp == 1174)
            setContentView(R.layout.test_half);
        else
            setContentView(R.layout.test);



        /// 01-01 13:14:11.522  8203  8203 D TestActivityTest: left = 0,top = 0,right = 1174,bottom = 1024

        Rect outSize = new Rect();
        getWindowManager().getDefaultDisplay().getRectSize(outSize);
        int left = outSize.left;
        int top = outSize.top;
        int right = outSize.right;
        int bottom = outSize.bottom;
        Log.d(TAG, "left = " + left + ",top = " + top + ",right = " + right + ",bottom = " + bottom);


        //////01-01 13:14:11.523  8203  8203 I TestActivityTest: x = 1174,y = 1024

        Display defaultDisplay = getWindowManager().getDefaultDisplay();
```

```
33.         Point point = new Point();
34.         defaultDisplay.getSize(point);
35.         int x = point.x;
36.         int y = point.y;
37.         Log.i(TAG, "x = " + x + ",y = " + y);
38.
39.
40.         /////01-
    01 13:14:11.524  8203  8203 I TestActivityTest: widthPixels = 1174,heightPixels = 1024
41.
42.         DisplayMetrics outMetrics = new DisplayMetrics();
43.         getWindowManager().getDefaultDisplay().getMetrics(outMetrics);
44.         int widthPixels = outMetrics.widthPixels;
45.         int heightPixels = outMetrics.heightPixels;
46.         Log.i(TAG, "widthPixels = " + widthPixels + ",heightPixels = " + heightPixels);
47.
48.
49.         /////01-
    01 13:14:11.525  8203  8203 W TestActivityTest: widthPixel = 2348,heightPixel = 1080
50.         DisplayMetrics out1Metrics = new DisplayMetrics();
51.         getWindowManager().getDefaultDisplay().getRealMetrics(out1Metrics);
52.         int widthPixel = out1Metrics.widthPixels;
53.         int heightPixel = out1Metrics.heightPixels;
54.         Log.w(TAG, "widthPixel = " + widthPixel + ",heightPixel = " + heightPixel);
55.
56.     }
```

10. TestActivity 的简单布局文件：

test_layout.zip

多屏切换动画的设计

在前面多屏方案中遗留了一个待解答的问题，就是分屏时候的动画实现，这里的动画分为三个部分，一个是小屏自己的窗口动画，一个背景应用和前景应用的动画。我们把被小屏挡住的应用称为背景应用（BG app），把在小屏上出现的应用称为前景应用（FG app）。所以切换动画由如下几部分构成：

- BG app animation
- Right display animation
- FG app animation
- Navigation bar animation

根据动画设计效果图，BG/FG app animation 看上去存在操作内部 view 的动画，例如卡片依

次出现，这样的动画称为 view 动画，view 动画需要 app 来实现（系统无法拿到应用的 view 实例），所以系统采用下面的方式实现三者动画的同步：

在切换过程中，系统会启动一个 500ms 的动画来对 right display 进行动画处理（translate/alpha/scale），同时在动画 start/end 的时候通知 BG/FG app 进行动画播放和结束，应用会收到这样的四种通知（Broadcast）：

- Right display enter animation start（start_display_enter）：
  BG app 按需播放右侧 view exit 动画，FG app 播放 enter 动画
- Right display enter animation end（end_display_enter）：
  BG app 按需结束播放，FG app 结束播放
- Right display exit animation start（start_display_exit）：
  BG app 按需播放右侧 view enter 动画，FG app 播放 exit 动画
- Right display exit animation end（end_display_exit）：
  BG/FG app 按需停止播放

应用收到通知后执行相应的动画（满足设计效果的那种卡片式动画），如上说明。

三、分屏模式下应用操作说明

1. 怎么保证 Activity 显示到 Display A 或者 Display B 上
   Android O 中具备应用显示到不同的 display 上的 API（launchDisplayId/ launchStackId）；所以一般是在 launcher 里面启动 app 的时候指定好 display id，这样就能保证 app 启动到对应的 display，或者在 startActivity 里面指定 launchDisplayId， 参考如下代码：

在启动 Activity 的时候可以指定需要启动的 displayid 参数以便将 activity 启动到对应的 display 上，sample code：

```
1. btnQNX.setOnClickListener(new View.OnClickListener() {
2.         @Override
3.         public void onClick(View view) {
4.             Log.d(TAG, "start action on QNX display");
5.             Intent intent = new Intent();
6.             intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
7.             intent.addFlags(Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED| Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT);
```

```
8.          intent.setAction("display.demo.QNXDisplay");
9.          Bundle options = new Bundle();
10.         options.putInt("android.activity.launchDisplayId", DisplayHelper.getInstance(getApplicationC
ontext()).getQnxDisplayId());
11.         MainActivity.this.startActivity(intent, options);
12.       }
13.   });
```

AndroidManifest.xml

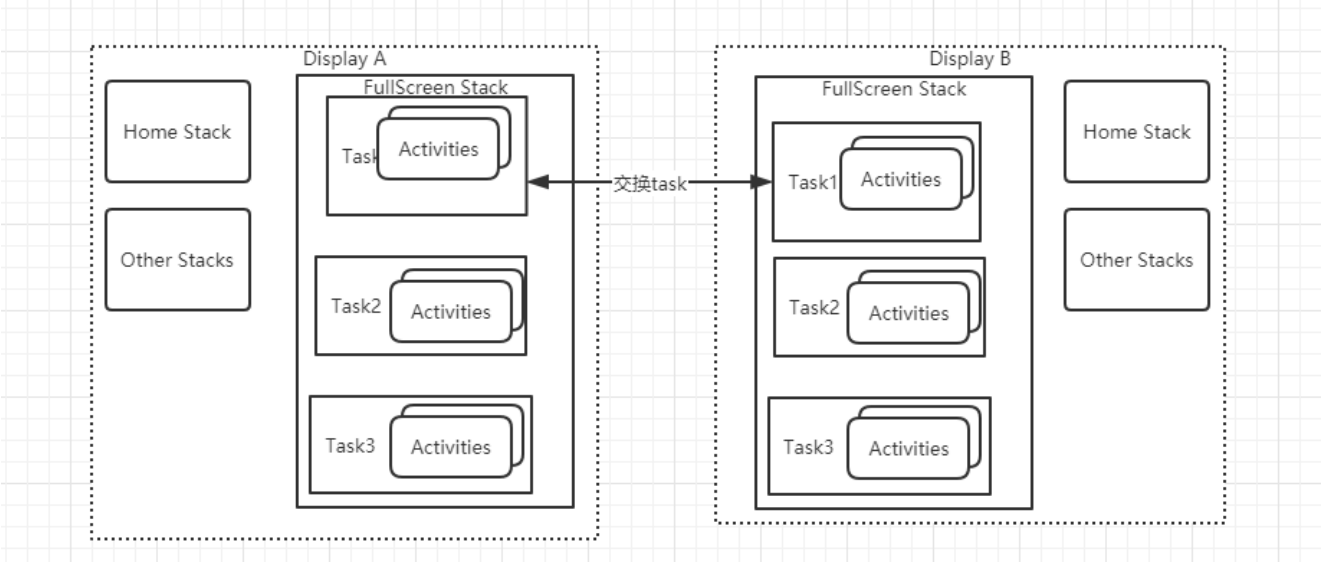
```
1.  <activity android:name=".QNXActivity"
2.          android:taskAffinity="demo.qnx"
3.          >
4.          <intent-filter>
5.              <action android:name="android.intent.action.MAIN" />
6.              <action android:name="display.demo.QNXDisplay"/>
7.              <category android:name="android.intent.category.DEFAULT" />
8.          </intent-filter>
9.      </activity>
```

说明:

- ✓ 在 launcher options 中设置"android.activity.launchDisplayId"，指定 displayID
- ✓ 在 launch intent 中设置 Intent.FLAG_ACTIVITY_NEW_TASK
- ✓ 为了保证 activity 在已经启动的情况下能够启动到其他的 display 上，intent 中需要设置 Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT;
- ✓ 在 AndroidManifest.xml 中为该 activity 指定不同的 taskAffinity; taskAffinity 为 activity 设置新的 taskRecord， 否则 AMS 会将该 activity 启动到和 starter activity 同一个 task 中，就无法实现启动到新的 display 上的目的，这个适用于同一个 APK 的不同 Activity 启动到不同的 display 上。对于 Launcher 启动的 APP 不需要设置这个属性（只要 App activty 和 launcher 的 activity 不属于同一个 task)。
- ✓ 对于多屏应用启动的方式，是按照 android 原生多屏的方式去做，可以参考/查询 Android API 说明。例如 activity 当前所处的 display 可以通过 getDisplay 来获取。当这个 Activity 需要启动其他的 activity 到自己所在的 display 时，通过 getDisplay 获取到 display 信息，然后设置待启动应用的 launchdDisplayId 参数。
- ✓ 如果需要将整个 TaskRecord（该 TaskRecord 包括 Activity A->Activity B->Activity C 等）移动到 Display A 或 Display B 上的话，可以直接如上指定 action 的启动方式，并加上必须的参数和执行的 displayId 即可；如果想要将该 TaskRecord 中的某个 Activity 移动到 Display A 或 Display B 上的话，可以指定该需要启动 Activity 的 componentName 并加上必须的参数和指定的 display 即可;
- ✓ 如果一个应用中包含了多个 TaskRecord 以及 TaskRecord 中包含了多个 Activity 的话，被不同 display 的应用启动的时候，不同 display 上的启动其他应用的那个应用中仍然需要加上以上相应的 flags，这样系统匹配到被启动的那个应用的时候会将被启动的应用中的所有 taskRecord 都移动到相应的 display 上

2. 主副驾屏上的应用可以交换

我们这边默认的实现方式是将顶层应用的 task 进行交换，根据 UE 的定义，这应该是系统的行为，应用不需要执行交换操作，但是应用可能会收到 onConfigurationChanged(display change) 的消息，应用通过 WindowManager.getDefaultDisplay 可以获取到当前所在的 display。



上面的切换逻辑保证了同一个 task 内的所有 activity 都会切换到另一个 display。所以对于 a1->a2->a3， 如果这三个 activity 都属于同一个 task，那么就同时切换到另一个 display；如果 a1 属于一个 task， a2/a3 属于另一个 task，那么只有 a2/a3 被切换。

3、主副驾屏上的窗口如何交换

Framework 提供了一个接口 changeViewDisplay(View view)，主副驾切换时（切换时会发送广播，action 值为"android.intent.action.swap）app 判断窗口 view 显示则需要调用 windowManager 的 changeViewDisplay 方法，示例：

```
WindowManager wm = (WindowManager) getSystemService(Context.WINDOW_SERVICE);
wm.changeViewDisplay(view);
```

3 双开的方案

这个听说你们以前沟通过，launcher 是有两个不同的 launcher，是主屏/副屏独立的 APP。

随心听的双开听说也是启动不同 Activity，最好不要启动同一个 activty 的多个实例，在切换的过程中不知道是否会引起其他的问题。

如果不想开发多个 APP，那可以考虑基于应用的双开方案，就是更新 APK 的包名，或者你们有其他的基于应用的双开方案推荐，我们可以共同研究一下。

四、 如何启动多屏模式

目前多屏服务已经集成在系统中，但不会自启动，需要根据如下两种方式之一：
1 am start –a TestActivity
这个是启动测试 activity，该 activity 会拉起多屏服务以及相关测试 window
2 am broadcast –a TestBootService -f 0x1000000
这个直接启动 service，并提供一个 testwindow

TestWindow 上有 5 个 buttong：
Start resize：显示小屏， 小屏 name 为”right_display”, display id 固定为 3
Stop resize：隐藏小屏
Show display/Hide display：只做 show/hide，不做 resize，这个后面会去掉
Swap Display：切换 display 的 task

当前 27 寸的版本中已经将分屏服务集成进去，无需手动启动

第二屏现在默认启动的 Home 这个 APK，如果要替换为自己的 APK，需要为自己的 Home Activity 设置如下属性：

```
1.  <intent-filter>
2.          <action android:name="android.intent.action.MAIN" />
3.          <category android:name="android.intent.category.SECOND_HOME"/>
4.          <category android:name="android.intent.category.DEFAULT" />
5.  </intent-filter>
```

如果设备里面没有 Home.apk，需要把这个 apk 推到车机里面，否则第二块屏上没有显示内容(显示黑屏)：

Home.apk

adb root
adb push Home.apk /system/app/Home/
adb sync
然后重启设备

五、多屏下 SystemUI 的设计

在多屏模式下 SystemUI 需要考虑如下模块的变更：

- Status bar
  监听 virtual display add/remove 消息，添加/消除 2rd statusbar；WMS 也会监听 display add/remove 消息，从而创建 display 相关的对象，例如 DisplayContent 等，systemui 必须在 DisplayContent 创建完毕之后才能去 addWindow，所以注意和 WMS 的竞态，以免系统 crash。

- Navigation bar
  监听 virtual display add/remove 消息，添加/消除 2rd navigation bar
- Quick panel
  需要添加分屏入口，并注意分屏的时候先隐藏 quick panel
- Multi-Notification
  ？？
- Multi-RecentApps
  需要为不同的 display 创建 recent-app list