软件绿色联盟
Software Green Alliance

Alibaba Group 阿里巴巴集团

HUAWEI

软件绿色联盟技术沙龙第28期·走进阿里

华为MateX适配指导及当前实践

2019 / 8 / 3 杭州 · 阿里巴巴

闫鸿飞

MateX适配项目经理

从事项目管理5年，负责Nexus 6P，荣耀8等项目软件项目交付

从事第三方应用适配项目3年。

当前负责MateX项目应用适配。

CONTENTS

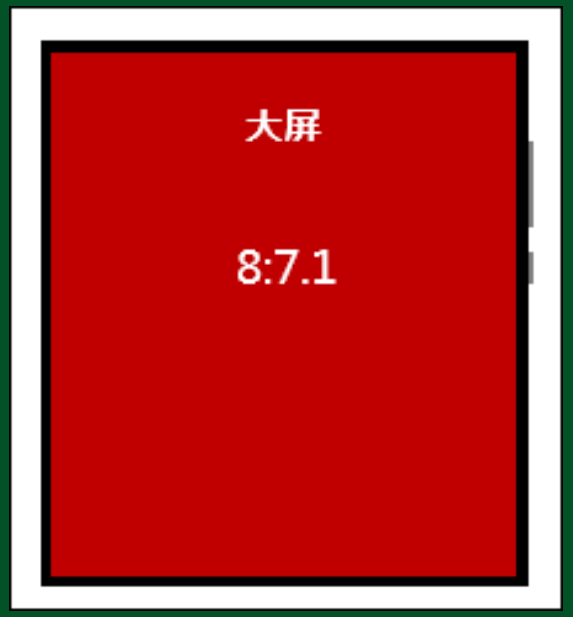
软件绿色联盟
Software Green Alliance

Alibaba Group 阿里巴巴集团

HUAWEI

01 MateX规格和接口介绍

MateX规格简介



展开大屏：
尺寸：8英寸
分辨率：2480x2200像素
屏幕比例：8:7.1
屏幕密度（densityDpi）：480
density:3.0

折叠主屏：
尺寸：6.6英寸
分辨率：2480×1148像素
屏幕比例：19.5:9
屏幕密度（densityDpi）：480
density:3.0

折叠副屏：
尺寸：6.38英寸
分辨率：2480×892像素
屏幕比例：25:9
屏幕密度（densityDpi）：480
density:3.0

MateX适配接口介绍

接口1：判断是否MateX手机

方法1：通过Build.MANUFACTURER+Build.DEVICE判断

方法2：通过Build.MANUFACTURER+Build.MODEL判断

```
private boolean isMateX() {
    if ("HUAWEI".equalsIgnoreCase(Build.MANUFACTURER) &&
("unknownRLI".equalsIgnoreCase(Build.DEVICE) ||
"HWTAH".equalsIgnoreCase(Build.DEVICE))) {
        return true;
    }
    return false;
}

private boolean isMateX2() {
    if ("HUAWEI".equalsIgnoreCase(Build.MANUFACTURER) && ("RLI-
AN00".equalsIgnoreCase(Build.MODEL) || "RLI-
N29".equalsIgnoreCase(Build.MODEL)
            || "TAH-AN00".equalsIgnoreCase(Build.MODEL) || "TAH-
N29".equalsIgnoreCase(Build.MODEL))) {
        return true;
    }
    return false;
}
```

接口2：判断当前屏幕是不是展开态大屏显示

```
if ((getResources().getConfiguration().screenLayout &
Configuration.SCREENLAYOUT_SIZE_MASK) ==
Configuration.SCREENLAYOUT_SIZE_LARGE) {
    // 华为Matex展开大屏
    Log.e(TAG, "is matex large screen");
} else {
    //华为Matex折叠态
    Log.e(TAG, "is matex small screen");
}
```

02 基础适配

适配指导（基础要求）

1.基本要求：

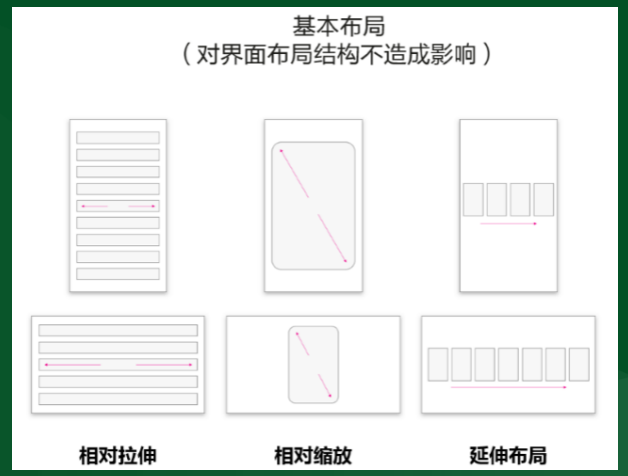
- 应用的所有页面在折叠态和展开大屏态下都能全屏显示，UI显示正常
- 应用的所有页面在折叠态和展开大屏态之间切换都能保证用户业务不中断，UI显示正常

2.适配步骤指导：

- 配置展开态全屏显示：

方式1：支持resizeable
方式2：设置支持的最大和最小屏幕比例

- 适配展开态大屏布局



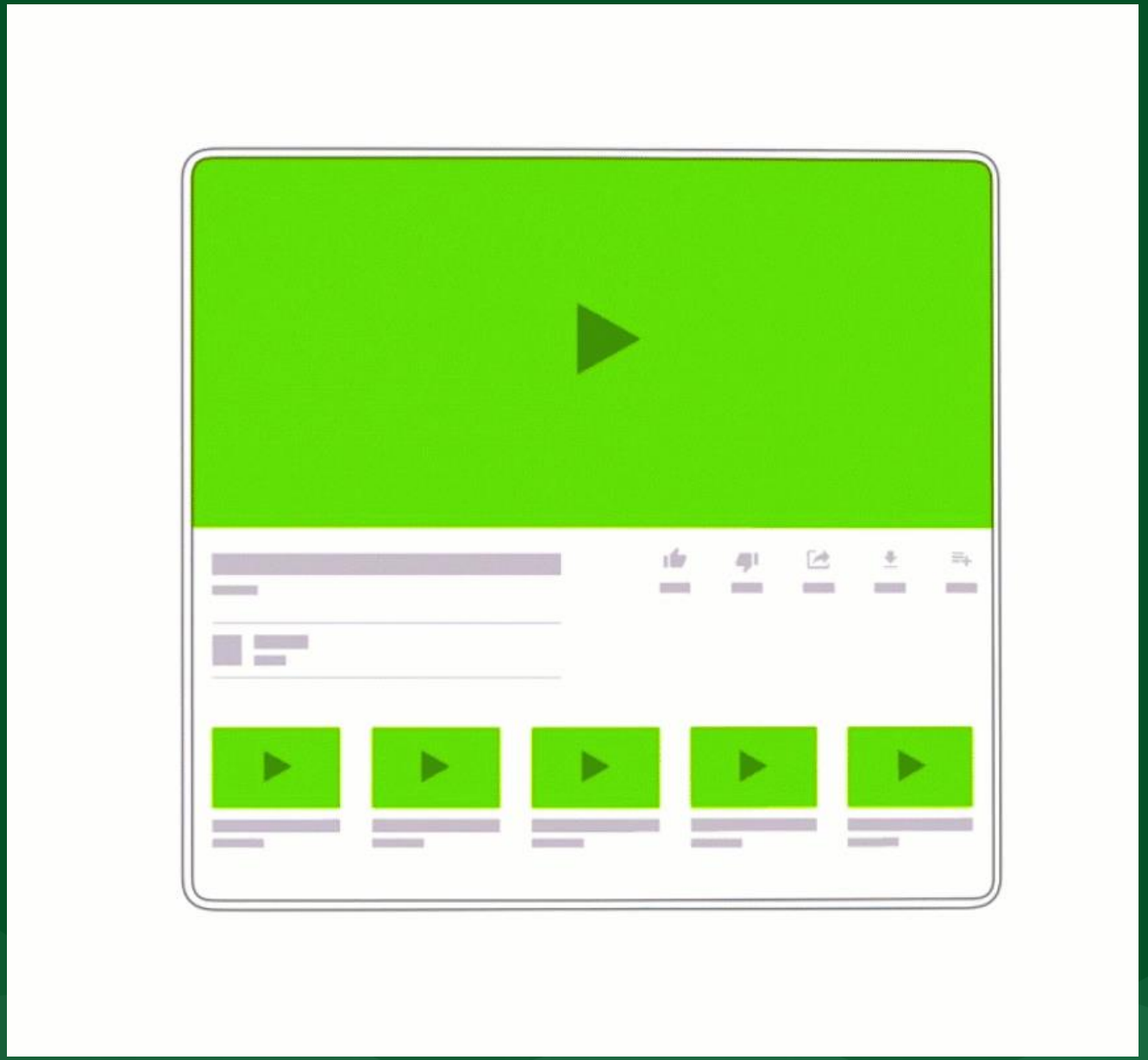
- 屏幕折叠展开时保证业务的连续性和UI调整

方式1：通过不重启方式
方式2：通过重启，重启前保存页面和用户数据的方式

```
<activity
   android:name=".MainActivity"
android:configChanges="screenSize|smallestScreenSize|screenLayout"/>
 activity>
```

✓ 不重启：需要调整的显示内容不多，建议选择。在onConfigurationChanged方法中通过代码动态调整UI来适配；重新初始化View，把View和数据重新绑定即可
✓ 重启： UI复杂，需要动态调整的地方多，建议针对大屏单独写一套layout资源放在（layout-sw600dp）目录下面。在销毁activity之前，通过onSaveInstancesState()存储状态，在onCreate()或者是onRestoreInstanceState()进行状态的恢复

适配指导（DEMO）



软件绿色联盟
Software Green Alliance

Alibaba Group 阿里巴巴集团

HUAWEI

03 典型案例分享

1.页面无法全屏显示，有黑边

问题分析：应用不支持自适应，默认无法全屏显示，默认应用必须支持的最小比例是4:3

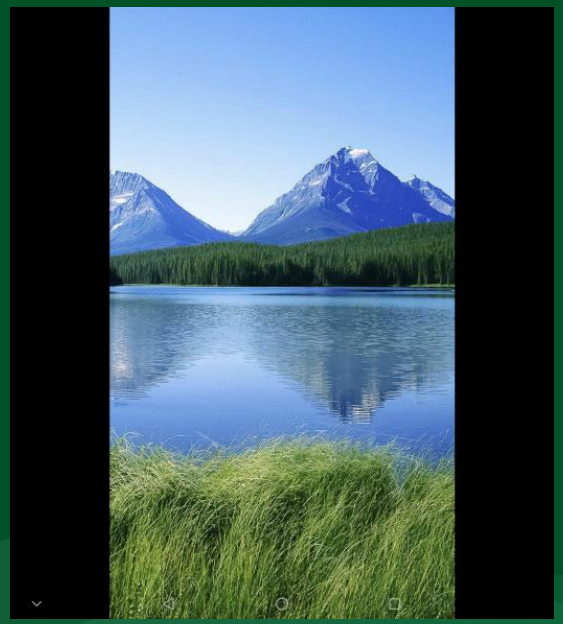
- 应用的TargetSdkVersion>=24，应用的页面主动申明不支持resizeable：android:resizeableActivity=“false"
- 应用的TargetSdkVersion<24，应用的页面没有主动申明支持resizeable：android:resizeableActivity="true“

修改建议：

- 推荐方式：应用支持resizeable，TargetSdkVersion>=24的应用去掉设置的android:resizeableActivity=“false”；TargetSdkVersion<24的应用增加android:resizeableActivity="true“
- 方式2：设置最大和最小比例

```
//设置应用支持的最大和最小屏幕比例方式1:
<application
    android:allowBackup= “true”
    android:maxAspectRatio= “2.4 “//只能在26或者26以上的手机生效
    android:minAspectRatio=” 1.0 “ //只能在29或者29以上的手机生效
```

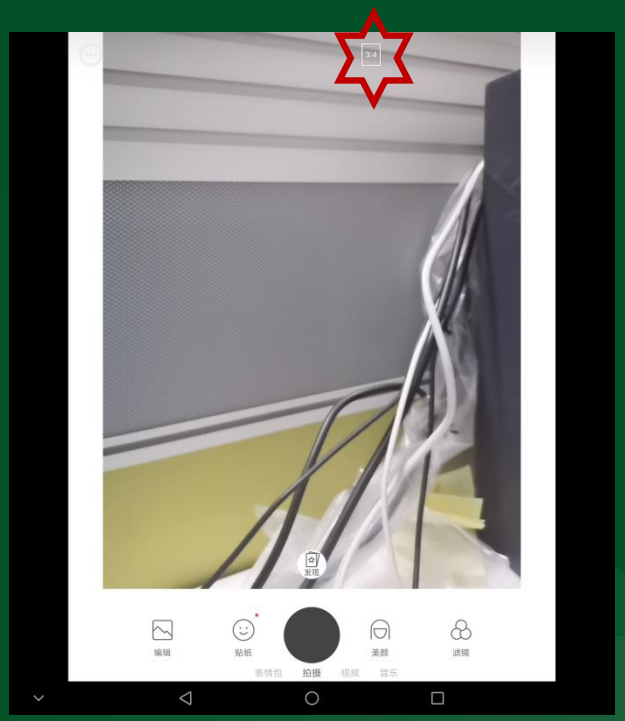
```
//设置应用支持的大和最小屏幕比例方式2:
<meta-data android:name="android.max_aspect" android:value="2.4" />
<meta-data android:name="android.min_aspect" android:value="1.0" />
```

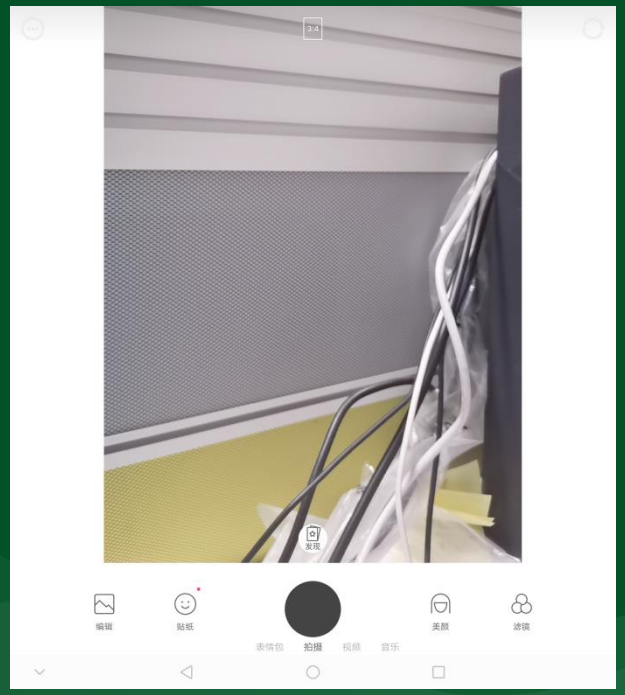


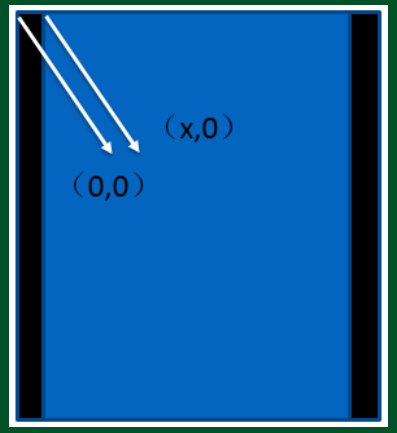
2.大屏控件缺失和布局截断问题

问题分析：坐标系不统一的问题，在折叠态应用是全屏显示，应用的window坐标系和手机屏幕的坐标是一致的，所以通过屏幕坐标系或者应用window坐标系计算控件显示的位置都是一致的，坐标计算不会有问题；但是在展开态，因为应用未适配，默认只能按照4:3显示，导致应用的window坐标系和手机屏幕的坐标系不统一，通过屏幕坐标系计算的控件显示位置和通过屏幕坐标系计算的控件显示位置发生了错位，显示在了应用的window之外的区域，出现了该问题。

修改建议：通过支持分屏或者是设置最小比例为1.0让应用在展开态能全屏显示，全屏显示屏幕坐标系和应用窗口坐标系就一致了。不过还是建议应用在计算和设置控件显示位置的时候需要统一坐标系，都使用屏幕坐标系，或者都使用应用窗口坐标系。







3.图片和视频显示截断问题

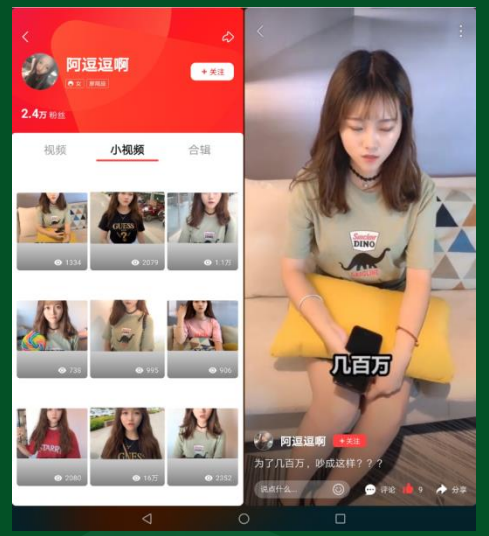
问题分析：因为图片或者视频资源分辨率和手机屏幕宽高比不一致导致，在保证全屏显示的效果时，必然出现截断的问题。

以 16:9 (1920*1080)图片为例。放大到2200的宽度分分辨率为(3911*2200)，显示在大屏(2480*2200)时，显示内容裁剪掉 37%

适配建议：大屏显示和视频全屏时保证图片和视频内容完整显示。

通过分栏布局：列表+视频播放解决该问题（fragment）

其他创新方案，保证图片不裁剪的同时充分利用大屏界面。



4.折叠态和展开态动态切换应用闪退和重启问题

问题分析：折叠态和展开态切换，应用如果没有设置不重启，默认系统会触发应用的Activity重走生命周期：

- 如果应用重启之前没有在Activity的onSaveInstanceState()方法中保存应用页面的状态数据，就会导致应用出现重启的问题；
- 如果应用在onDestroy()中调用finish()或其他自行终止进程方法。这将导致应用程序在设备折叠或展开时出现关闭、闪退等问题

修改建议：

- 不重启适配方式：

在AndroidManifest文件中增加设置：

```
<activity
    android:name=".MainActivity"
    android:configChanges="screenSize|smallestScreenSize|screenLayout">
```

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    initView();//动态调整布局
}
```

重写onConfigurationChanged() 方法，通过该方法的Configuration参数获得屏幕的分辨率等信息，就可以针对不同比例屏幕下的应用界面布局做相应调整，如切换布局、调整控件位置和间距等，**如果不动态调整就会出现后面的问题6**

- 重启方式：需要进行状态的保存和恢复。

在销毁activity之前，通过onSaveInstancesState()存储状态， 在onCreate()或者是onRestoreInstanceState()进行状态的恢复。

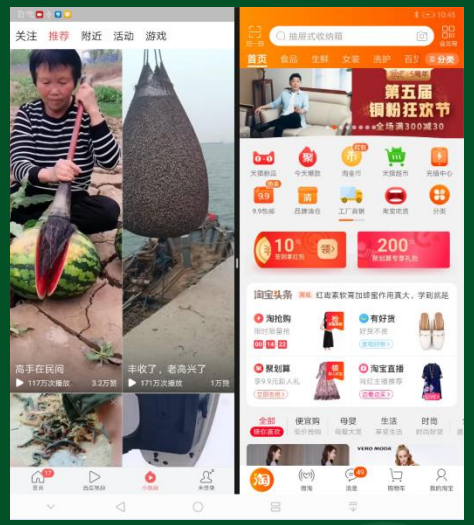
Note:不要在OnDestroy()中调用finish()或其他自行终止进程。这将导致应用程序在设备折叠或展开时关闭、闪退等问题。

5.折叠态和展开态动态切换或者应用进入分屏模式应用布局显示异常问题

问题分析：就是应用设置了页面不重启，但是没有在onConfigurationChanged方法中动态调整布局导致的问题，应用的窗口宽度发生变化，一些控件的宽度是和应用窗口宽度动态计算出来的，发生变化之后需要重新计算。

修改建议：

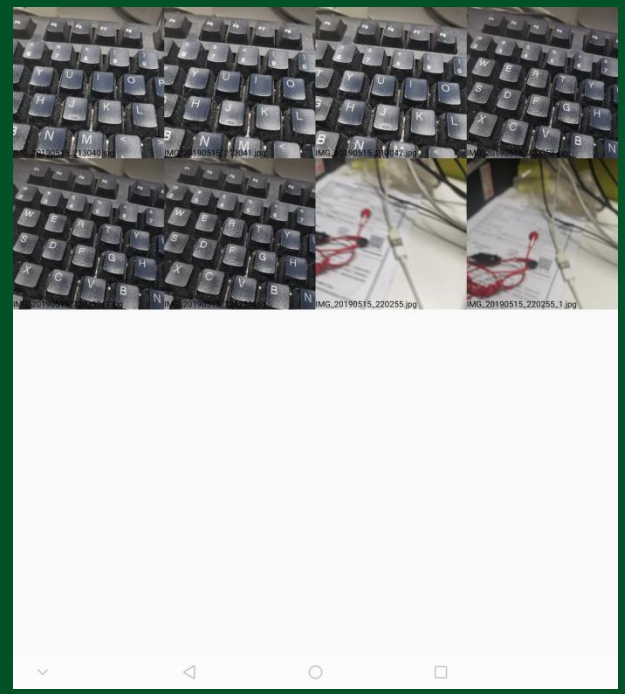
对于控件的宽度和高度是通过屏幕尺寸计算之后代码设置的，需要在onConfigurationChanged中把这部分逻辑代码重新执行一遍即可



```java
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    initView();//屏幕宽度变化，需要动态调整控件显示的宽度
}
```

```java
/**
 * 通过屏幕宽度计算图片显示的宽度:
 * 1.大屏每行显示4个图片，图片的宽度等于应用窗口宽度的1/4
 * 2.小屏每行显示2个图片，图片的宽度等于应用窗口宽度的1/2
 */
private void initView() {
    int w_screen2 = DensityUtil.dip2px(this,
getResources().getConfiguration().screenWidthDp);
    if ((getResources().getConfiguration().screenLayout &
Configuration.SCREENLAYOUT_SIZE_MASK) == Configuration.SCREENLAYOUT_SIZE_LARGE) {
        // 华为Matex展开大屏
        Log.e(TAG, "is matex large screen");
        mRecyclerView.setLayoutManager(new GridLayoutManager(this, 4));
        mRecyclerAdapter.setWidth(w_screen2 / 4);
        mRecyclerAdapter.setHeight(w_screen2 / 4);
    } else {
        //华为Matex折叠态
        Log.e(TAG, "is matex small screen");
        mRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));
        mRecyclerAdapter.setWidth(w_screen2 / 2);
        mRecyclerAdapter.setHeight(w_screen2 / 2);
    }
}
```

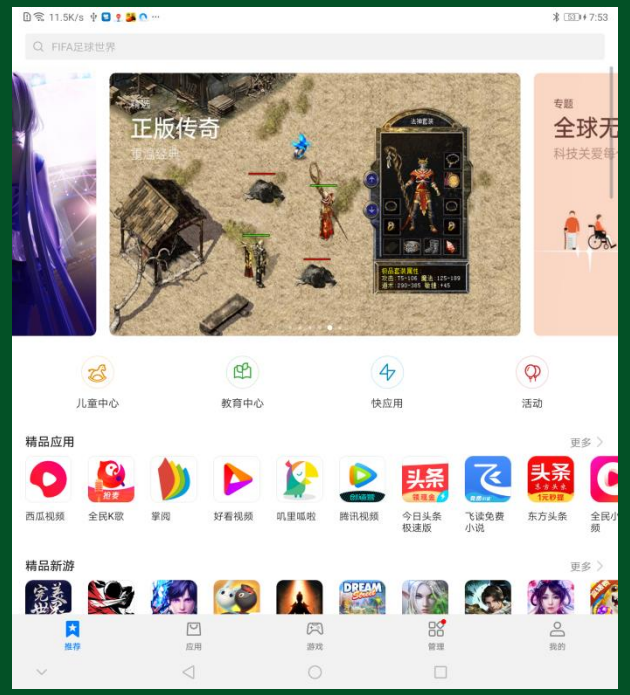
```java
@Override
public void onBindViewHolder(final MyViewHolder holder, final int position) {
    final Item item = itemList.get(position);
    holder.title.setText(item.getTitle());
    holder.image.setImageBitmap(item.getBitmap());
    //动态修改图片的宽度和高度
    ViewGroup.LayoutParams lp = holder.image.getLayoutParams();
    lp.height = height;
    lp.width = width;
    holder.image.setLayoutParams(lp);
```



6.大屏应用布局放大，显示比例失调问题

问题分析：应用布局自适应规则问题，控件的宽度和高度的比例固定，高度根据宽度计算，宽度根据屏幕宽度变大而变大，导致应用的布局整体等比例放大，控件的高度过大，导致应用布局显示比例失调，显示内容反而变少

修改建议：使用延伸布局，参考华为应用市场的实现，保持控件的宽度和高度不变，通过增加每行显示的控件的个数优化



7. 视频切全屏体验优化

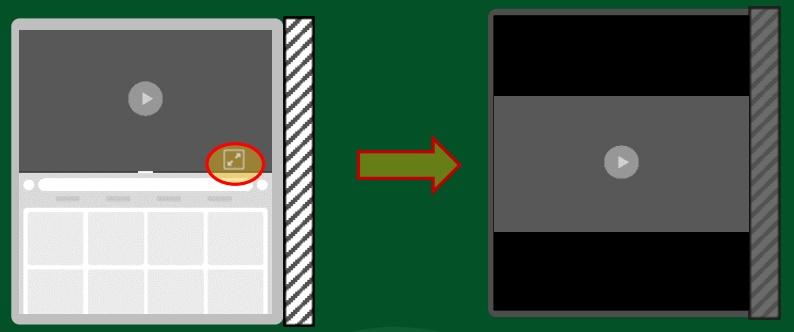
问题说明：类正方形屏幕视频全屏播放竖屏播放和横屏播放体验差异小。点击全屏按钮之后不做屏幕旋转，直接在竖屏的状态下全屏播放体验更佳。

类正方形定义：宽高比例介于[3/4 ~ 4/3]之间

示例代码：

根据屏幕比例进行差异化处理：

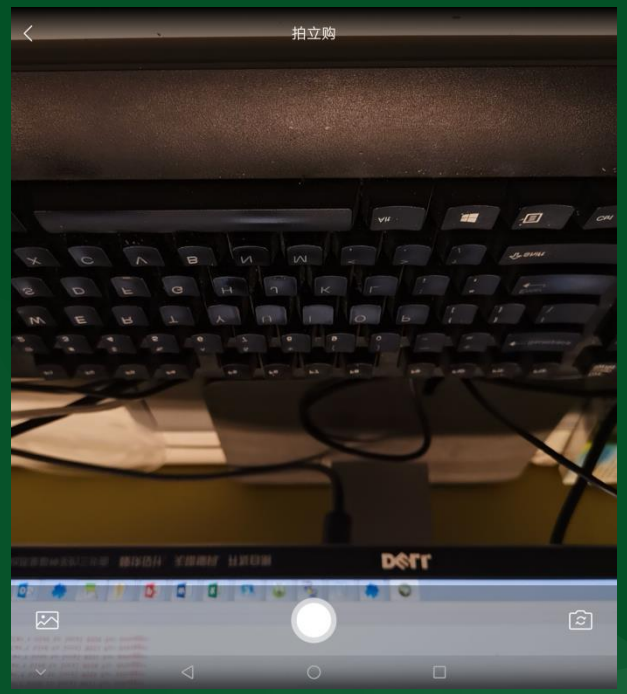
```
float ratio = screenWidthDp screenHeightDp;
if(ratio >= 3.0f/4.0f && ratio <= 4.0f/3.0f){
        //只做全屏、不做转屏
}else{
        //原有处理逻辑
}
```



8.前置相机屏幕角度问题

问题原因：应用使用camera api中的setDisplayOrientation(90)设置了固定角度。

修改建议：使用Android推荐方法获取和设置旋转角度




```
/**
 * If you want to make the camera image show in the same orientation as the display,
you can use the following code.
 */
public static void setCameraDisplayOrientation(Activity activity,
                                                int cameraId, android.hardware.Camera
camera) {
    android.hardware.Camera.CameraInfo info =
            new android.hardware.Camera.CameraInfo();
    android.hardware.Camera.getCameraInfo(cameraId, info);
    int rotation = activity.getWindowManager().getDefaultDisplay()
            .getRotation();
    int degrees = 0;
    switch (rotation) {
        case Surface.ROTATION_0: degrees = 0; break;
        case Surface.ROTATION_90: degrees = 90; break;
        case Surface.ROTATION_180: degrees = 180; break;
        case Surface.ROTATION_270: degrees = 270; break;
    }
    int result;
    if (info.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
        result = (info.orientation + degrees) % 360;
        result = (360 - result) % 360;  // compensate the mirror
    } else {  // back-facing
        result = (info.orientation - degrees + 360) % 360;
    }
    camera.setDisplayOrientation(result);
}
```

软件绿色联盟
Software Green Alliance

Alibaba Group 阿里巴巴集团

HUAWEI

04 进阶适配

适配指导（进阶要求）

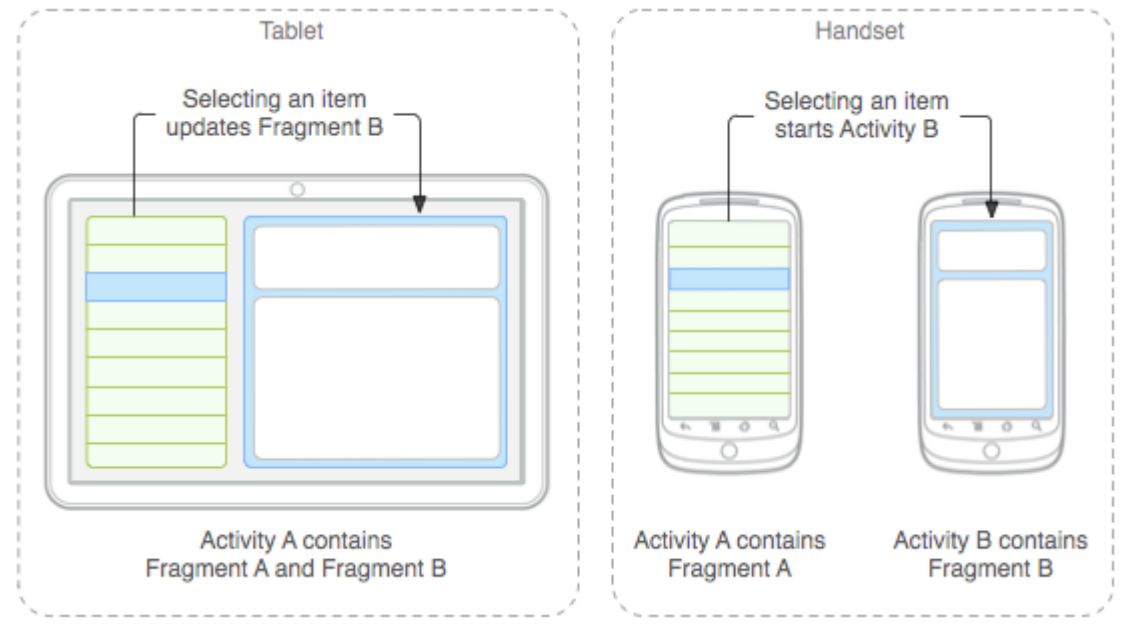
分栏布局

1.谷歌提供的fragment指导：

https://developer.android.com/guide/components/fragments?hl=zh-CN

背景：Android 在 Android 3.0（API 级别 11）中引入了片段，主要是为了给大屏幕（如平板电脑）上更加动态和灵活的 UI 设计提供支持。

理解：fragment是组件；Activity可以有一个或者多个fragment组合拼接而成



大屏Actvity布局包含两个fragment（TitlesFragment +DetaisFragment）：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"
            android:id="@+id/titles" android:layout_weight="1"
            android:layout_width="0px" android:layout_height="match_parent" />

    <FrameLayout android:id="@+id/details" android:layout_weight="1"
            android:layout_width="0px" android:layout_height="match_parent"
            android:background="?android:attr/detailsElementBackground" />

</LinearLayout>
```

小屏Actvity布局只包含一个fragment（TitlesFragment）：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"
            android:id="@+id/titles"
            android:layout_width="match_parent" android:layout_height="match_parent" />
</FrameLayout>
```

实现两个fragment：

```
public static class TitlesFragment extends ListFragment {
...
}
```

```
public static class DetailsFragment extends Fragment {
...
}
```

多窗口

Android的多窗口开发指导：
https://developer.android.com/guide/topics/ui/multi-window.html?hl=zh-cn

要素一：正确处理应用的横竖屏

- 其实“竖屏”真正的意义仅仅是高度大于宽度，“横屏”的意义是宽度大于高度。
- 所以从这个定义来考虑，在应用调整大小时，可能会从一个朝向转到另一个朝向就说得通了。在多窗口模式下在运行时锁定朝向的
  setRequestedOrientation()
  android:screenOrientation

方法都是无效的。

要素二：正确处理Activity的Pause状态

- Q版本之前，手机系统中只有一个resumed activity，一个focus activity，用户仍可以看到处于pause状态的应用，应用在暂停状态下可能仍需要继续操作。例如处于pause状态但可见的视频应用应继续播放（可以通过Activity.isInMultiWindowMode()方法确认 Activity 是否处于多窗口模式）
- Q版本后，Android允许multi resumed activity出现，并不是仅有一个resumed activity。Q版本应该注意一些互斥资源的访问，比如：相机、麦克风。

```
protected void onTopResumedActivityChanged(boolean
topResumed) {
    if (topResumed) {
        // Top resumed activity
        // Can be a signal to re-acquire exclusive resources
    } else {
        // No longer the top resumed activity
    }
}
```

要素三：应用支持分屏需要考虑应用的窗口宽度变化对应用布局的影响

可以通过下面的方法获取当前应用窗口的宽和高：

```
getResources().getConfiguration().screenWidthDp
```

```
getResources().getConfiguration().screenHeightDp
```

获取高度和宽度之后需要动态调整布局

要素四：悬浮窗

应用的TargetSdkVersion>=26，使用 SYSTEM_ALERT_WINDOW 权限的应用无法再使用以下窗口类型来在其他应用和系统窗口上方显示提醒窗口：TYPE_PHONE、TYPE_PRIORITY_PHONE、TYPE_SYSTEM_ALERT、TYPE_SYSTEM_OVERLAY、TYPE_SYSTEM_ERROR

相反，应用必须使用名为 TYPE_APPLICATION_OVERLAY 的新窗口类型

画中画（Picture-in-picture）支持

Android的画中画开发指导： https://developer.android.google.cn/guide/topics/ui/picture-in-picture#handling_ui

1.申明支持画中画

默认不支持画中画，需要应用在AndroidManifest文件视频播放Activity通过设置android:supportsPictureInPicture和android:resizeableActivity为true支持：

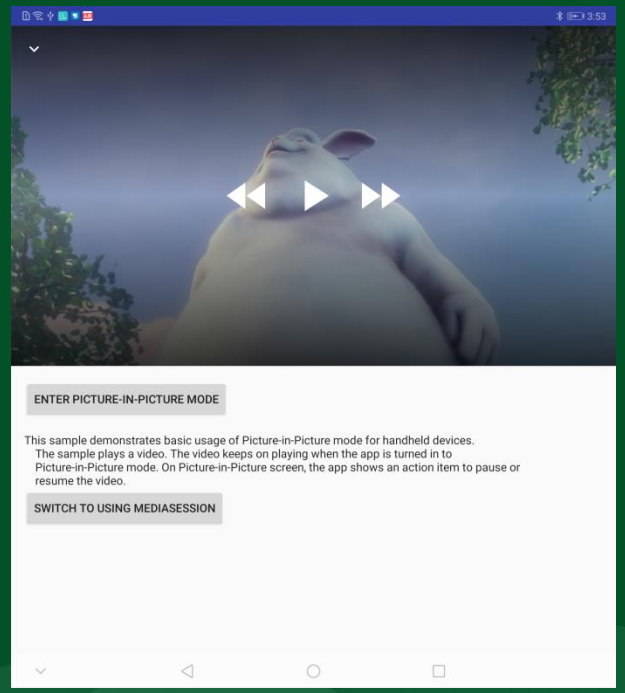
```
<activity android:name="VideoActivity"
    android:resizeableActivity="true"
    android:supportsPictureInPicture="true"
    android:configChanges=
        "screenSize|smallestScreenSize|screenLayout|orientation"
    ...
```

2.将应用的 Activity 切换到画中画模式：

```
Rational aspectRatio = new Rational(mMovieView.getWidth(),
mMovieView.getHeight());
mPictureInPictureParamsBuilder.setAspectRatio(aspectRatio).build();
enterPictureInPictureMode(mPictureInPictureParamsBuilder.build());
```

3.处理画中画模式中的 UI：

```
@Override
public void onPictureInPictureModeChanged(boolean isInPictureInPictureMode) {
    if (isInPictureInPictureMode) {
        // Hide the controls in picture-in-picture mode.
        ...
    } else {
        // Restore the playback UI based on the playback status.
        ...
    }
}
```

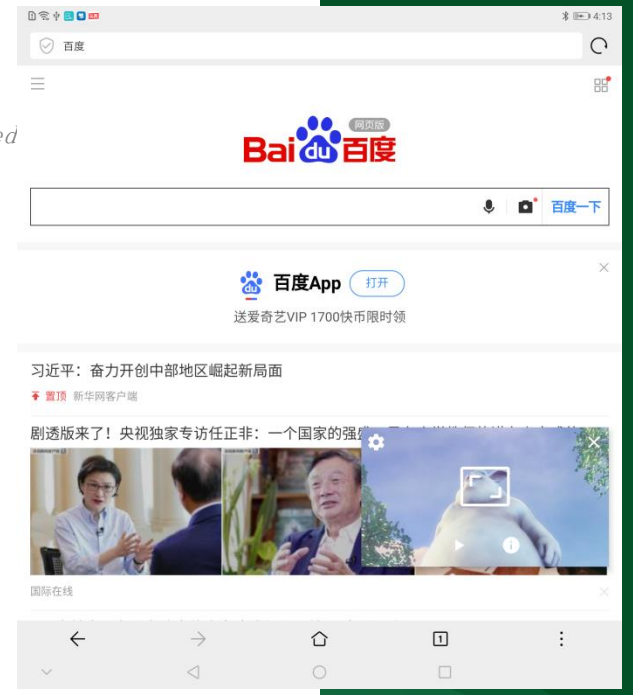


画中画（Picture-in-picture）支持

4.在画中画模式中增加自定义的菜单和响应菜单点击

```
final ArrayList<RemoteAction> actions = new ArrayList<>();
final PendingIntent intent =
        PendingIntent.getBroadcast(
                MainActivity.this,
                requestCode,
                new
Intent(ACTION_MEDIA_CONTROL).putExtra(EXTRA_CONTROL_TYPE, controlType),
                0);
final Icon icon = Icon.createWithResource(MainActivity.this, iconId);
actions.add(new RemoteAction(icon, title, title, intent));
// Another action item. This is a fixed action.
actions.add(
        new RemoteAction(
                Icon.createWithResource(MainActivity.this,
R.drawable.ic_info_24dp),
                getString(R.string.info),
                getString(R.string.info_description),
                PendingIntent.getActivity(
                        MainActivity.this,
                        REQUEST_INFO,
                        new Intent(
                                Intent.ACTION_VIEW,
                                Uri.parse(getString(R.string.info_uri))),
                        0)));
mPictureInPictureParamsBuilder.setActions(actions);
setPictureInPictureParams(mPictureInPictureParamsBuilder.build());
```

```
@Override
public void onPictureInPictureModeChanged(
        boolean isInPictureInPictureMode, Configuration configuration) {
    super.onPictureInPictureModeChanged(isInPictureInPictureMode,
configuration);
    if (isInPictureInPictureMode) {
        // Starts receiving events from action items in PiP mode.
        mReceiver =
                new BroadcastReceiver() {
                    @Override
                    public void onReceive(Context context, Intent intent) {
                        if (intent == null
                                ||
!ACTION_MEDIA_CONTROL.equals(intent.getAction())) {
                            return;
                        }
                        // This is where we are called
in-Picture action
                        // items.
                        final int controlType =
intent.getIntExtra(EXTRA_CONTROL_TYPE, 0);
                        switch (controlType) {
                            case CONTROL_TYPE_PLAY:
                                mMovieView.play();
                                break;
                            case CONTROL_TYPE_PAUSE:
                                mMovieView.pause();
                                break;
                        }
                    }
                };
        registerReceiver(mReceiver, new
IntentFilter(ACTION_MEDIA_CONTROL));
    }
```



画中画（Picture-in-picture）支持

5.在画中画模式中继续视频播放

```
@Override
protected void onStop() {
    // On entering Picture-in-Picture mode, onPause is called, but not onStop.
    // For this reason, this is the place where we should pause the video
playback.
    mMovieView.pause();
    super.onStop();
}
```

```
@Override
public void onPause() {
   // If called while in PIP mode, do not pause playback
   if (isInPictureInPictureMode()) {
      // Continue playback
      ...
   } else {
      // Use existing playback logic for paused Activity behavior.
      ...
   }
}
```

6.播放新的视频

```
<activity android:name="VideoActivity"
   ...
   android:supportsPictureInPicture="true"
   android:launchMode="singleTask"
   ...
```

7.判断手机是否支持画中画

```
hasSystemFeature(PackageManager.FEATURE_PICTURE_IN_PICTURE)
```

支持拖拽

1.谷歌提供的开发指导：

https://developer.android.com/guide/topics/ui/drag-drop?hl=zh-CN

一种更快捷的文件和数据共享方式：借助 Android 拖放框架，可允许您的用户使用图形化拖放手势，将数据从当前布局中的一个视图移到另一个视图。 该框架包括拖拽事件类、拖拽侦听器以及帮助程序方法和类。

适配支持的大致流程：

- 定义响应拖拽的操作方式（一般是长按）

```
holder.image.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View view) {
        startDrag(view, item);
        return true;
    }
});
```

- 准备拖拽的共享数据，支持文本和文件的content uri

```
ClipData.Item item = new ClipData.Item(imageItem.getTitle(), null, null,
imageItem.getUri());
ClipData dragData = new ClipData((CharSequence) v.getTag(),
MIMETYPES_TEXT_PLAIN, item);
```

- 准备拖拽阴影

```
View.DragShadowBuilder myShadow = new MyDragShadowBuilder(v);
```

- 开始拖拽

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
    v.startDragAndDrop(dragData,  // the data to be dragged
            myShadow,  // the drag shadow builder
            null,      // no need to use local data
            View.DRAG_FLAG_GLOBAL | View.DRAG_FLAG_GLOBAL_URI_READ
    );
} else {
    v.startDrag(dragData,  // the data to be dragged
            myShadow,  // the drag shadow builder
            null,      // no need to use local data
            View.DRAG_FLAG_GLOBAL | View.DRAG_FLAG_GLOBAL_URI_READ
    );
}
```

- 数据接收方，监听拖拽事件：

```
mDragListen = new MyDragEventListener();
mDropImageView.setOnDragListener(mDragListen);
```

- 数据接收方，响应拖拽事件：

```
case DragEvent.ACTION_DROP:
    ClipData.Item item = dragEvent.getClipData().getItemAt(0);
    Uri uri = item.getUri();
    // Gets the text data from the item.
    dragData = item.getText();
    try {
        ((ImageView) view).setImageBitmap(getBitmapFromUri(uri));
    } catch (IOException e) {
        e.printStackTrace();
    }
    // Displays a message containing the dragged data.
    Toast.makeText(MateXAdaptationDragActivity.this, "Dragged data is " +
dragData, Toast.LENGTH_LONG).show();
    return true;
```

软件绿色联盟
Software Green Alliance

Alibaba Group 阿里巴巴集团

HUAWEI

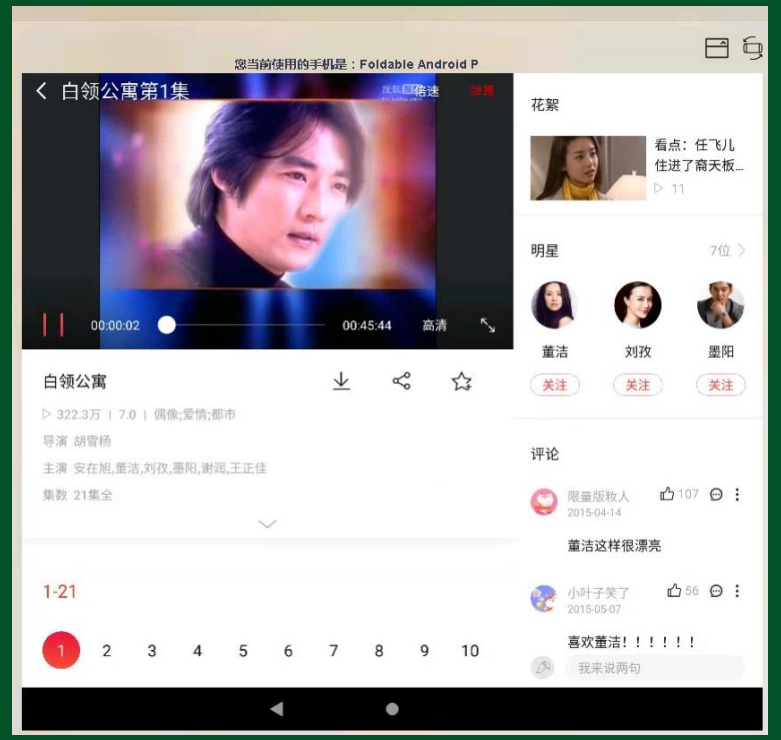
05 开发与调试

远程真机

网址：http://deveco.Huawei.com/

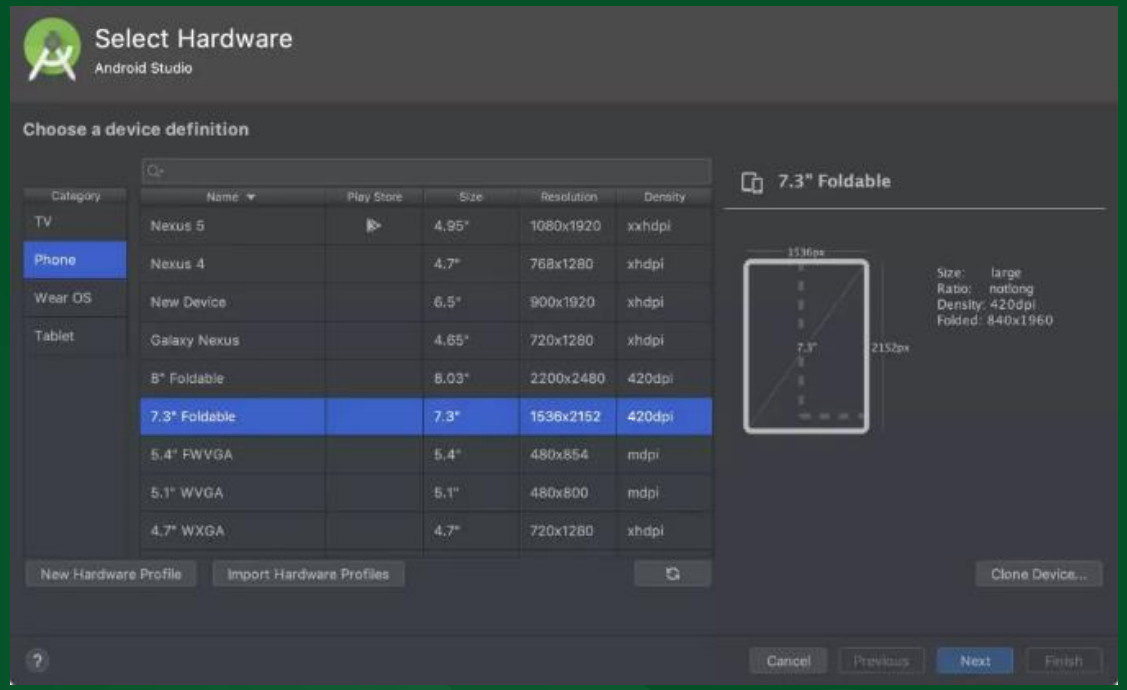
使用指导文档：https://mp.weixin.qq.com/s/86aVzyRkbLREefNkSMEb8w

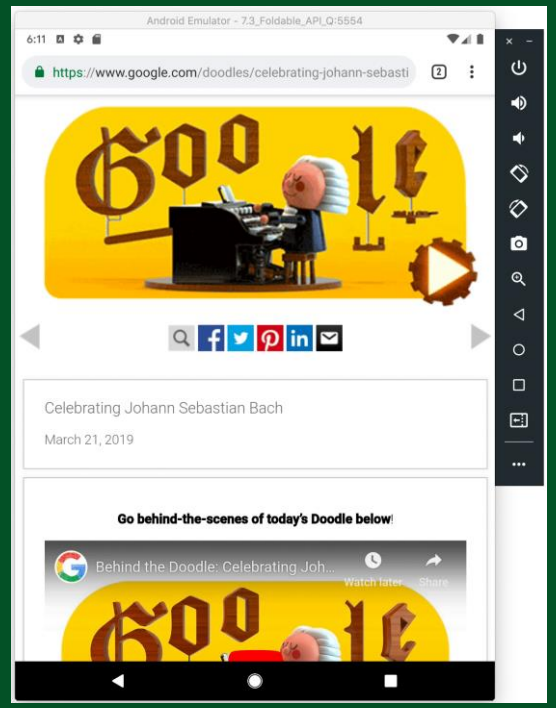
- 注册和登录
- 选择“测试服务-远程真机”
- 选择“廊坊仿真机”



谷歌模拟器

从Android Studio 3.5 Canary版本开始，开发者可创建一个运行Q Beta 2版本的折叠屏虚拟设备，其支持的硬件配置分别为7.3英寸(折叠后为4.6英寸)，以及8英寸(折叠后为6.6英寸)。建议开发者**选择8英寸的设备**来模拟适配华为的MateX折叠屏手机（备注：**Android Studio 3.5 Canary版本下载地址：https://developer.android.google.cn/studio/preview**）





命令模拟调试

开发者也可以在非折叠屏手机上面通过命令修改手机的屏幕分辨率来进行模拟调试：

1.折叠切展开模拟方法：

- 预先将手机设置主屏分辨率：

adb shell wm size 1148x2480

- 通过修改手机分辨率为全屏分辨率模拟状态切换：

adb shell wm size 2200x2480

2.展开切折叠模拟方法：

- 预先将手机设置全屏分辨率：

adb shell wm size 2200x2480

- 通过修改手机分辨率为主屏分辨率模拟状态切换：

adb shell wm size 1148x2480

3.分辨率恢复方法：**adb shell wm size reset**