## wm自带的功能

***wm size:*** 设置输出到设备上的像素数（分辨率）。正常1920x1080设备，若设置成1920x2160则会纵轴下面一半是延展的部分。应用会刷新配适新的分辨率。

***wm density:*** 设置像素密度，暂时发现设置之后无效果。

***wm overscan:*** 设置overscan区域，即黑边区域大小。直接从原图中扣除。特别需要注意的是：overscan x0,y0,x1,y1 四个坐标 x0,x1分别是左右黑边宽度。y0,y1分别是上下黑边宽度。但特别注意的是，显示内容纵轴是从0到height-y0-y1；横轴是x0到width-x1。

## 2．wm的实现

wm是一个指令，代码实现在framework/base/cmds/wm相关目录下。

Wm.java 继承BaseCommand 并使用app\_process(zygote)启动main函数代码块。

***wm size :*** 会保存两个Point参数 initialSize,baseSize 从WMS中获取。

initialSize 来自getInitialDisplaySize，获取默认输出设备的DisplayContent，获取分辨率。

baseSize 来自getBaseDisplaySize，同上获取mBaseDisplay属性。

当根据参数，设置好需要修改的w,h之后，调用setForcedDisplaySize(Display.DEFAULT\_DISPLAY, w, h)，DEFUALT\_DISPLAY是默认ID 即0，而不是默认显示设备则**不能修改**。

需要注意的是，setForcedDisplaySize中w,h最小200，最大是原本分辨率的2倍。

并将其用Settings.Global.putString写入表中。

接下来setForcedDisplaySizeLocked设置w,h值，然后reconfigureDisplayLocked刷新。

///////////分析如何刷新////////////////////////

***wm density :*** 会保存两个Point参数 initialDensity,baseDensity

两个属性同上，也是从默认的DisplayContent中获取

## 3. 尝试修改

模仿overscan机制，在wm指令中添加onehand模式，将输入的rect作为可以输出的区域。现在已经尝试在DisplayContent.DisplayInfo中的logicalWidth、logicalHeight修改值。但发现reconfigureDisplayLocked会重设这个值。而且就算不重设，在performXXX过程中，这个值不被使用到，w,h作为SystemRect的值，并没有读取logical的值。

所以尝试失败，接下来需要在performXXX中尝试修改SystemRect看是否可以直接达到我们想要的效果。