**Mstar 4K图片播放流程**

1. **LMP业务部分**

1、入口：Activity：com.helios.activity.LocalPhotoPlayActivity

onCreate:

LocalPhotoHelper.getInstances().init(LocalPhotoPlayActivity.this)

New PlayPhotoManager，执行initData setPhotoPath

2、4k、2k管理PlayPhotoManager ：com.helios.module.photo.PlayPhotoManager

4k callback：mViewFourkSoftDecode

2k callback：mNormalPictrueCallBack

photoInfo：mPhotoBottomControlCallBack

viewLoadCallBack：运行2k：TimerTask2K、运行4k：TimerTask4K

a、initData中通过setPhotoPath调用viewLoadCallBack运行TimerTask2K和TimerTask4K

mPhotoView.loadFourPic

show4KPhoto

mImg4K.isPreparing()

startSoftDecodePhoto

m4kRunnable

image4kView.start

b、onResume：

mPhotoView.refreshFourPic();

startSoftDecodePhoto

mView4kSoftDecode.callBackSoftDecode

m4kRunnable

image4kView.start

3、com.helios.baseView.LocalPhotoHelper

判断是否4K图片

得到非4k图片的bitmap

得到图片的数据信息

得到缩略图等

4、com.helios.viewModule.photo.PhotoBottomView

显示图片信息

5、com.helios.viewModule.photo.PhotoView

设置监听对HarewareImageView

mImg4K = (HardwareImageView) findViewById(R.id.photo\_4k\_display);

isSupport4KPhoto = mImg4K.isSupport();

if(isSupport4KPhoto){

mImg4K.setOnIHarewareImageListener(mFourKListener);

}

rotate4K: mImg4K.rotate((mRotateNum+1)\*-90, 1.0f)

setPhotoPath showLowQualityImage

6、com.helios.viewModule.photo.PhotoItemView

显示gif、2k

showLowQualityImage

sendMainThread 启动ViewRefreshCallBack refresh

启动toRefresh

调用mViewLoadCallBack.callBackLoadPic()

调用ViewLoadCallBack callBackLoadPic

调用TimerTask4K

调用loadFourPic

调用show4KPhoto

mImg4K.isPreparing()

startSoftDecodePhoto

m4kRunnable

image4kView.start

7、涉及到4k类HardwareImageView的调用：

PlayPhotoManager：

image4kView.start(image4kPath);

PhotoView：

mImg4K.isSupport()

mImg4K.rotate()

mImg4K.setOnIHarewareImageListener(mFourKListener);

1. **LMP 4K图片播放类，封装了framework层的4K图片播放类**

从第一部分分析看出，LMP实际上对HardwareImageView的调用只有isSupport、start和rotate接口

调用逻辑：如果不是prepareing(prepareing表明正在start)状态则执行start，通过PhotoView HarewareImageListener监听是否prepared，是则真正显示出图片

import tv.whaley.vendor.HardwareImagePlayer 系统侧java类，通过jni接口调用到ImagePlayer

HardwareImageView继承SurfaceView，内部底层解码播放器使用的是HardwareImagePlayer

isSupport：判断底层是否支持4K播放（通过HardwareImagePlayer接口isAvailable）

surfaceChanged、 surfaceCreated、surfaceDestroyed，与SurfaceView相关，不懂？？？

Reset:执行HardwareImagePlayer stop reset

Release：执行HardwareImagePlayer stop reset release

mListener，LMP传下来的监听接口

start：

{

nextPrepared

setImagePath

openImagePlayer

}

openImagePlayer：

执行HardwareImagePlayer setOnErrorListener、setDataSource、setOnInfoListener（MEDIA\_INFO\_VIDEO\_RENDERING\_START显示完成）、setOnPreparedListener（MEDIA\_PLAYER\_STATE\_PREPARED prepare完成）、prepareAsync。

onInfo：HARDWARE\_IMAGE\_LOADING\_END

start：HARDWARE\_IMAGE\_LOADING\_START

prepare完成后执行startImagePlayer

startImagePlayer:HardwareImagePlayer setDisplay、start

prepareNextPhoto:下一张4K图片提前解码准备

setDstImageSize：配置图片显示宽高

Rotate：setDstImageSize HardwareImagePlayer ImageRotateAndScale

Scale : setDstImageSize HardwareImagePlayer ImageRotateAndScale

moveDirection：HardwareImagePlayer ImageCropRect

prepareNextPhoto：HardwareImagePlayer ImageDecodeNext

showNextPhoto：HardwareImagePlayer ImageShowNext

1. **系统framework java层4K图片类- HardwareImagePlayer**

device/helios/common/WtvSDK/WtvApi/src/tv/whaley/vendor/HardwareImagePlayer.java。

从第二部分看出app 4K类对该类的调用接口有：

isAvailable、stop、reset、release、setOnErrorListener、setDataSource、setOnInfoListener、setOnPreparedListener、prepareAsync、setDisplay、start、ImageRotateAndScale、ImageCropRect、ImageDecodeNext、ImageShowNext。

HardwareImagePlayer是暴露给app的类，只是为了封装了私有成员HardwareImagePlayerImp的所有接口。

不同的平台只实现类HardwareImagePlayerImp即可(对接口类IHardwareImagePlayer的实现)。对app来说接口HardwareImagePlayer保持不变。但是封装HardwareImagePlayer用法的app类HardwareImageView的行为可能需要改变，如果每个vendor的调用流程差异较大。也可以同时修改HardwareImagePlayer和HardwareImageView更好的兼容到所有vendor平台。

目标：尽量不修改HardwareImageView和HardwareImagePlayer类。只实现HardwareImagePlayerImp来满足所有平台的4K图片播放功能。

Mstar平台HardwareImagePlayerImp实现：封装com.mstar.android.media.MMediaPlayer。

1. **mstar平台提供的4K图片播放**

device/mstar/common/libraries/media/java/com/mstar/android/media/MMediaPlayer.java

jni接口：

device/mstar/common/libraries/media/jni/android\_media\_MMediaPlayer.cpp

走mediaplayer打分机制，选择imgplayer播放器，app调用prepareAsync、setdatasource和start后，imgplayer解码并把bitmap画在video层。

Imgplayer：

device/mstar/common/libraries/media/mmplayer/imgplayer.cpp

imgplayer start后去解码时，会调用skia库，通过SkImageDecoder去解码，解码后的rgb数据需要convert成YUV（420、422）格式，再通过surface、nativewindow直接显示在video层。