杰理图像转换库
使用说明

版本历史

注意: 版本倒序编写，最新版本置顶

版本	日期	修改者	修改记录
1.6.0	2025/5/23	钟卓成	1. 增加707N手表SDK图像转换指定输出编码格式 2. 增加转换参数
1.5.0	2025/2/24	钟卓成	1. 增加707N手表SDK图像资源压缩算法
1.4.0	2024/8/16	钟卓成	1. 支持707N手表SDK图像资源算法 2. 统一算法的名称
1.2.0	2023/7/18	钟卓成	1. 支持695N/701N手表SDK图像资源不打包算法
1.1.0	2023/4/26	钟卓成	1. 支持701N手表SDK图像资源转换算法
1.0.0	2022/12/16	陈森华	1. 初始版本 2. 支持695N手表SDK图像资源转换算法

一、概述

> 杰理图像转换库，主要是用于杰理手表/手环类带屏SDK的图像资源生成。通过杰理自有的算法，转换常见的图像格式(PNG，JPEG，BMP)为杰理图像资源，适用于杰理UI框架，或者第三方UI框架。

二、开发说明

1. 库导入

BmpConvert_Vxxx.aar：图像转换算法库， 其中 `xxx` 表示版本号。

把上述库，导入到工程的libs文件夹。

build.gradle配置如下:

```
dependencies {

    implementation fileTree(include: ['*.aar'], dir: 'libs')
}
```

2. 图像转换功能

```
String inPath = "输入文件路径(例如: /in.png, /in.jpg)";
String outPath = "输出文件路径(例如:out.res, out)";
//1.初始化图片转换对象
BmpConvert bmpConvert = new BmpConvert();
//2.开始图像转换
int flag = BmpConvert.TYPE_701N_RGB;        //701N图像转换算法 - RGB
//flag = BmpConvert.TYPE_695N_RBG;            //695N图像转换算法 - RGB
//V1.1.0+ 支持
//flag = BmpConvert.TYPE_701N_ARGB;           //701N图像转换算法 - ARGB
//V1.2.0+ 支持
//flag = BmpConvert.TYPE_701N_RGB_NO_PACK;    //701N图像转换算法 - RGB & 不打包封装
//flag = BmpConvert.TYPE_701N_ARGB_NO_PACK;   //701N图像转换算法 - ARGB & 不打包封装
//V1.4.0+ 支持
//flag = BmpConvert.TYPE_707N_RGB;            //707N图像转换算法 - RGB
//flag = BmpConvert.TYPE_707N_ARGB;           //707N图像转换算法 - ARGB
//flag = BmpConvert.TYPE_707N_RGB_NO_PACK;    //707N图像转换算法 - RGB & 不打包封装
//flag = BmpConvert.TYPE_707N_ARGB_NO_PACK;   //707N图像转换算法 - ARGB & 不打包封装

//V1.6.0+ 支持
//指定输出编码格式
int format = ConvertParam.FORMAT_AUTO;  //自动选取压缩后最小的格式，ARGB8565或者
ARGB8888
//        format = ConvertParam.FORMAT_ARGB_8565; //指定输出ARGB8565格式。 若算法是
RGB算法，会变成指定输出RGB565格式
//        format = ConvertParam.FORMAT_ARGB_8888; //指定输出ARGB8888格式。 若算法是
RGB算法，会变成指定输出RGB888格式
//构造转换参数
ConvertParam param = new ConvertParam().setFormat(format);

bmpConvert.bitmapConvert(flag, inPath, outPath, param, new OnConvertListener() {
```

```
    //回调转换开始
    //path: 输入文件路径
    @Override
    public void onStart(String path) {

    }

    //回调转换结束
    //result:  转换结果
    //output:  输出文件路径
    @Deprecated
    @Override
    public void onStop(boolean result, String output) {
        //3.不需要使用图片转换功能时，需要释放图片转换对象
        //bmpConvert.release();
    }

    //回调转换结束
    //result:  转换结果
    //output:  输出文件路径
    @Override
    public void onStop(ConvertResult result, String outFilePath) {
        //result.isConvertSuccess(); //转码成功
        //result.getAlgorithm();     //转换算法
        //result.getBufSize();       //转换数据大小
        //result.getPixelFormat();   //转换图像格式
        //result.getCompressMode();  //压缩模式

        //3.不需要使用图片转换功能时，需要释放图片转换对象
        //bmpConvert.release();
    }
});
```

ConvertParam

转换参数

```
public class ConvertParam {

    /**
     * 自动选取压缩后最小的格式，ARGB8565或者ARGB8888
     */
    public static final int FORMAT_AUTO = 0;
    /**
     * 指定输出ARGB8565格式
     * <p>
     *      若算法是RGB算法，会变成指定输出RGB565格式
     * </p>
     */
    public static final int FORMAT_ARGB_8565 = 1;
    /**
     * 指定输出ARGB8888格式
     * <p>
```

```
     *      若算法是RGB算法，会变成指定输出RGB888格式
     * </p>
     */
    public static final int FORMAT_ARGB_8888 = 2;

    /**
     * 编码格式
     */
    private int format = FORMAT_AUTO;
}
```

ConvertResult

转换结果

```
public class ConvertResult implements Parcelable {
    /**
     * 结果码
     * <p>
     * 说明:<br/>
     * 1. 小于等于0，视为转码失败，负数为错误码<br/>
     * 2. 大于0，意味着转码成功
     * </p>
     */
    private int result;
    /**
     * 算法类型
     */
    private int algorithm;
    /**
     * 数据大小
     */
    private int bufSize;
    /**
     * 图像格式
     * <p>
     * 说明: 从707N算法后，增加该字段，用于【不打包算法】的处理
     * </p>
     */
    private int pixelFormat;
    /**
     * 压缩模式
     * <p>
     * 说明: 从707N算法后，增加该字段，用于【不打包算法】的处理
     * </p>
     */
    private int compressMode;
}
```

OnConvertListener

```
public interface OnConvertListener {

    /**
     * 回调转换开始
     *
     * @param inFilePath String 文件路径
     */
    void onStart(String inFilePath);

    /**
     * 回调转换结束
     *
     * @param isOk        boolean 转换结果
     * @param outFilePath String 输出路径
     */
    @Deprecated
    void onStop(boolean isOk, String outFilePath);

    /**
     * 回调转换结束
     *
     * @param result      ConvertResult 转换结果
     * @param outFilePath String 输出文件路径
     */
    void onStop(ConvertResult result, String outFilePath);
}
```

2.1 同步接口

- 旧接口

方法	**bitmapConvertBlock**
作用	bitmap图像转换(阻塞)
参数	type : 编码算法 inFilePath: 输入文件路径(例如：in.png ,in.jpg) outFilePath: 输出文件路径(例如:out.res, out)
结果	result： 结果码 结果码大于0视为成功，其他值为错误码
备注	耗时流程，请在【子线程】调用

- 新接口

方法	bitmapConvertAndCompressBlock
作用	bitmap图像转换并压缩(阻塞)
参数	type : 编码算法 inFilePath: 输入文件路径(例如：in.png ,in.jpg) outFilePath: 输出文件路径(例如:out.res, out) convertParam: 转换参数
结果	result： 转码结果
备注	耗时流程，请在【子线程】调用

2.2 注意事项

1. 转换图像必须小于等于屏幕尺寸，不建议质量过大
2. 转换库仅支持单线程操作，不能并发执行。
3. 目前仅AC707N支持压缩算法

3. 算法名称对照表

芯片系列	旧算法名称	新算法名称	算法说明
AC695N	TYPE_BR_23	TYPE_695N_RBG	AC695N-WATCH-SDK 图像转换算法 - RGB
JL701N	TYPE_BR_28	TYPE_701N_RGB	JL701N-WATCH-SDK 图像转换算法 - RGB
	TYPE_BR_28_ALPHA	TYPE_701N_ARGB	JL701N-WATCH-SDK 图像转换算法 - ARGB
	TYPE_BR_28_RAW	TYPE_701N_RGB_NO_PACK	JL701N-WATCH-SDK 图像转换算法 - RGB & 不打包封装
	TYPE_BR_28_ALPHA_RAW	TYPE_701N_ARGB_NO_PACK	JL701N-WATCH-SDK 图像转换算法 - ARGB & 不打包封装
AC707N		TYPE_707N_RGB	AC707N-WATCH-SDK 图像转换算法 - RGB

芯片系列	旧算法名称	新算法名称	算法说明
		TYPE_707N_ARGB	AC707N-WATCH-SDK 图像转换算法 - ARGB
		TYPE_707N_RGB_NO_PACK	AC707N-WATCH-SDK 图像转换算法 - RGB & 不打包封装
		TYPE_707N_ARGB_NO_PACK	AC707N-WATCH-SDK 图像转换算法 - ARGB & 不打包封装

4. 算法的选择规则

1. 优先根据【芯片类型】选择
2. 然后根据【是否打包】选择
3. 最后根据【资源格式】选择

比如: JL701N系列芯片，手表SDK，推送 a.png图片， 打包格式。

JL701N系列芯片 ---> TYPE_701N_RGB

TYPE_701N_ARGB

TYPE_701N_RGB_NO_PACK

TYPE_701N_ARGB_NO_PACK

打包格式 ---> TYPE_701N_RGB

TYPE_701N_ARGB

a.png图片 ---> 因为PNG图片带alpha值，所以选择 【TYPE_701N_ARGB】

当然，如果想资源小一点，可以选择【TYPE_701N_RGB】