**虹图人像人体特效产品使用指南**

1. **关于我们**

虹图+ · 人像人体可以实现人像美颜、AR贴纸道具、人像抠图、手势识别、人脸关键点追踪等功能，可单独使用其中功能模块，包体更小，适用于各种直播、音视频社交、短视频、特效相机等场景。

1. **产品介绍**

产品功能

1）美肤功能：美白、磨皮、红润、清晰、亮度、去黑眼圈、去法令纹；

2）美型功能：包括瘦脸、大眼、V脸、窄脸、瘦颧骨、瘦下颌骨、丰太阳穴、小头、小脸、短脸、下巴、缩人中、发际线、眼间距、开眼角、长鼻、瘦鼻、嘴型、微笑嘴唇等；

3）滤镜功能：提供不少于30款滤镜；

4）风格推荐：经典、网红、女神、复古、日杂等；

5）AR贴纸道具：贴纸+水印；

6）手势识别；

7）人像抠图：绿幕抠图 + AI抠图。

购买指南

1）注册账号：通过虹图+官网注册虹图账号并登录。如果已有账号请直接登录。

2）购买付费版证书：在确认产品符合功能需求和业务场景后，您可以根据具体需求场景，在控制台购买页选择并购买对应SDK版本的正式版授权 APP ID 。购买成功之后，您可以继续以下步骤，在控制台完成SDK授权和应用的绑定。 注意：您可以先创建应用，绑定信息，申请测试版证书，在 SDK下载 文档中，下载虹图·人像人体SDK进行集成试用，付费购买后无需替换证书 APP ID 直接升级为付费版。

3）创建应用：您可以进入虹图用户中心控制台创建应用并绑定包名、应用名信息，自动生成 APP ID。具体流程请参见 控制台指南。创建一个免费或试用版应用并绑定包名、应用名信息后，需要通过官方审批，请填写完绑定信息后尽快提交申请，通常申请会在1-3个工作日完成审批。

控制台指南

为了让用户更加快速地掌握并使用控制台，我们将一些常用服务根据不同用户的不同使用需求进行了划分整理，目前控制台主要分为总揽，项目管理，设置三个模块用于不同的作用导航。

1. **快速入门**

Demo体验

您可以直接进入官网下载我们的Demo进行功能体验，包括演示APP X 、Demo工程以及集成了云服务推拉流、音视频SDK的Demo工程，您可以根据当前项目需要进行针对性的调试。

跑通集成示例

虹图·人像人体在Gitee上提供一个开源的示例项目 HTEffectDemo 。本文介绍如何快速跑通该示例项目，体验虹图·人像人体效果。您也可以直接观看我们的视频教程。 注意：本文中UI可能和最新版有出入，以最新代码工程为准。

操作步骤

1）创建应用

登录到用户中心控制台，进入左侧项目管理>人像人体SDK，单击创建，在弹出的对话框中输入您的应用名称和包名，并单击提交。

2）下载 HTEffectDemo 源码

在刚刚创建出的应用中，单击配置进入项目配置页面，单击右侧文档中心里的下载即可进入下载页面。

3）配置 HTEffectDemo 工程文件

Android

下载完成后，进入 HTEffectDemo 工程，打开 app 模块的 AndroidManifest 文件，将 android:label 的值替换为您刚刚创建的应用名称。

打开 app 模块中的 build.gradle 文件，将 applicationId 的值替换为您刚刚创建的包名。

全局搜索 initHTEffect 方法，将参数 YOUR\_APP\_ID 修改为创建应用时生成的APP ID。

iOS

下载完成后，打开 HTEffectDemo 工程，将 Display Name 和 Bundle Identifier 分别替换为您刚刚创建的应用名称和包名并回车确认。

全局搜索 initHTEffect 方法，将参数 YOUR\_APP\_ID 修改为创建应用时生成的APP ID。

4）编译运行

Android

使用 AndroidStudio （4.1及以上的版本）编译运行 HTEffectDemo 工程即可。

iOS

使用 Xcode（11.0及以上的版本）打开源码目录下的 HTEffectDemo.xcworkspace 工程，编译并运行 HTEffectDemo 工程即可。

1. **SDK实践**

iOS平台

1. 导入工程  
   （1）将下载好的 HTEffect.framework 库文件和 HTEffect.bundle 资源包放到您的项目文件夹下。  
   （2）在 Xcode > General 中[添加动态库](https://help.apple.com/xcode/mac/current/" \l "/dev51a648b07)，确保 **Embed** 属性设置为 **Embed & Sign**。  
   （3）在 Xcode > Build Settings 中搜索 bitcode ，将 **Enable Bitcode** 设置为 **No**。  
   （4）在 Xcode > Info 中添加 **App Transport Security Settings** > **Allow Arbitrary Loads** 并设置为 **YES**。
2. 使用 HTUI （可选）

HTUI 是开源的，可根据项目需求选用。 使用 HTUI 方法如下:  
将 HTUI 文件夹添加到您的项目文件夹中，导入 HTUIManager.h 并设置 HTUIManagerDelegate。

[[HTUIManager shareManager] loadToWindowDelegate:self];

1. 集成开发

3.1 初始化  
HTEffect 的初始化函数调用一次即可生效。但由于 app 在第一次安装的时候需要获取网络权限，此时调用初始化是无法成功的，所以推荐您在 AppDelegate 中调用 initHTEffect 并设置 HTEffectDelegate 回调，待进入视频页面时判断初始化状态，若失败可再执行初始化方法，避免出现第一次安装时初始化失败。

// 在线鉴权初始化方法

[[HTEffect shareInstance] initHTEffect:@"YOUR\_APP\_ID" withDelegate:self];

// 离线鉴权初始化方法

[[HTEffect shareInstance] initHTEffect:@"YOUR\_APP\_ID"];

注：项目运行后在控制台搜索 InitInfo，可查看具体的版本信息和初始化状态。

3.2 添加 HTUI （可选）  
如果需要使用 HTUI，您可以在 viewDidLoad 中添加以下方法：

[[HTUIManager shareManager] loadToWindowDelegate:self];

[self.view addSubview:[HTUIManager shareManager].defaultButton];

3.3 渲染步骤  
定义一个 **BOOL**变量 isRenderInit ，用来标志渲染器的初始化状态。根据获取到的视频格式，采用对应的方法进行渲染：

1）Texture 纹理  
首先调用 initTextureRenderer 初始化纹理渲染器，再调用渲染方法 processTexture。

/\*\*

\* 初始化纹理渲染器

\*

\* @param width 图像宽度

\* @param height 图像高度

\* @param rotation 图像是否需要旋转，不需旋转为CLOCKWISE\_0

\* @param isMirror 图像是否存在镜像

\* @param maxFaces 人脸检测数目上限设置，推荐取值范围为1~5

\*

\* @return 返回初始化结果，成功返回true，失败返回false

\*/

- (BOOL)initTextureRenderer:(int)width height:(int)height rotation:(HTRotationEnum)rotation isMirror:(BOOL)isMirror maxFaces:(int)maxFaces;

/\*\*

\* 处理纹理数据输入

\*

\* @param textureId 纹理ID

\*

\* @return 返回处理后的纹理数据

\*/

- (GLuint)processTexture:(GLuint)textureId;

/\*\*

\* 使用示例

\*/

if (!\_isRenderInit) {

[[HTEffect shareInstance] releaseTextureRenderer];

\_isRenderInit = [[HTEffect shareInstance] initTextureRenderer:width height:height rotation:rotation isMirror:isMirror maxFaces:maxFaces];

}

[[HTEffect shareInstance] processTexture:textureId];

2）CVPixelBufferRef 视频帧  
首先调用 initBufferRenderer 初始化buffer渲染器，再调用渲染方法 processBuffer。

/\*\*

\* 初始化buffer渲染器

\*

\* @param format 图像格式

\* @param width 图像宽度

\* @param height 图像高度

\* @param rotation 图像是否需要旋转，不需旋转为CLOCKWISE\_0

\* @param isMirror 图像是否存在镜像

\* @param maxFaces 人脸检测数目上限设置，推荐取值范围为1~5

\*

\* @return 返回初始化结果，成功返回true，失败返回false

\*/

- (BOOL)initBufferRenderer:(HTFormatEnum)format width:(int)width height:(int)height rotation:(HTRotationEnum)rotation isMirror:(BOOL)isMirror maxFaces:(int)maxFaces;

/\*\*

\* 处理buffer数据输入

\*

\* @param buffer 视频帧数据

\*/

- (void)processBuffer:(unsigned char \*)buffer;

/\*\*

\* 使用示例

\*/

CVPixelBufferLockBaseAddress(pixelBuffer, 0);

unsigned char \*buffer = (unsigned char \*) CVPixelBufferGetBaseAddressOfPlane(pixelBuffer, 0);

if (!\_isRenderInit) {

[[HTEffect shareInstance] releaseBufferRenderer];

\_isRenderInit = [[HTEffect shareInstance] initBufferRenderer:format width:width height:height rotation:rotation isMirror:isMirror maxFaces:maxFaces];

}

[[HTEffect shareInstance] processBuffer:buffer];

CVPixelBufferUnlockBaseAddress(pixelBuffer, 0);

3.4 销毁

结束渲染时，需根据视频格式，调用对应的释放方法，通常写在 dealloc 方法里:

/\*\*

\* 销毁纹理渲染资源

\*/

[[HTEffect shareInstance] releaseTextureRenderer];

/\*\*

\* 销毁buffer渲染资源

\*/

[[HTEffect shareInstance] releaseBufferRenderer];

3.5 资源文件  
HTEffect 所有资源均支持放在用户自己的服务器，可使用以下方法进行设置：

/\*\*

\* 设置素材网络路径

\* 将素材保存在自定义的网络存储中的情况下，设置网络路径

\*

\* @param url 素材网络路径

\*/

[[HTEffect shareInstance] setResourceUrl:url];

3.6 人脸检测功能  
HTEffect 支持人脸检测功能，可通过以下方法返回人脸数量：

int faceNumber = [[HTEffect shareInstance] isTracking];

Android平台

1. 导入工程  
   （1）将 HTEffect.aar 文件拷贝到 app 模块中的 libs 文件夹下，并在 app 模块的 build.gradle 文件的 dependencies 中，增加如下依赖：

dependencies {

implementation files('libs/HTEffect.aar')

}

1. 将 jniLibs 文件夹中，各个 ABI 对应的 libHTEffect.so 文件，拷贝到对应目录中。  
   （3）将 assets 资源文件拷贝到项目的对应目录中。

2. 使用 HTUI （可选）  
HTUI 是开源的，可根据项目需求选用。 使用 HTUI 方法如下:  
（1）依赖我们的 htui 工程，使用我们提供的开源 UI 库，将 htui 文件夹拷贝到工程根目录下，在工程根目录的 settings.gradle 文件中，增加如下代码：

include(":htui")

（2）在 app 模块中的 build.gradle 文件的 dependencies 中，增加如下代码：

implementation project(':tiui')

1. 集成开发

3.1 初始化  
HTEffect 初始化函数程序中调用一次即可生效，建议您在 Application 创建的时候调用;如果渲染功能使用不频繁，也可以在使用的时候调用，接口如下:

// 在线鉴权初始化方法

HTEffect.shareInstance().initHTEffect(context, "YOUR\_APP\_ID", new InitCallback() {

@Override public void onInitSuccess() {}

@Override public void onInitFailure() {}

});

// 离线鉴权初始化方法

HTEffect.shareInstance().initHTEffect(context,"YOUR\_APP\_ID");

注：项目运行后在控制台搜索 InitInfo，可查看具体的版本信息和初始化状态。

3.2 添加 **HTUI** （可选）  
（1）设置使用 htui 的 Activity 继承或间接继承 FragmentActivity，例如：

public class CameraActivity extends FragmentActivity {

//...

}

（2）如果需要使用 htui，请调用 addcontentView 实现UI的添加，代码如下：

addContentView(

new HTPanelLayout(this).init(getSupportFragmentManager()),

new FrameLayout.LayoutParams(

ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.MATCH\_PARENT));

3.3 渲染步骤  
定义布尔变量 isRenderInit ，用来标志渲染方法是否初始化完成，然后根据得到的视频帧格式的不同，使用对应的方法进行渲染：  
（1）GL\_TEXTURE\_EXTERNAL\_OES 纹理格式，首先调用渲染初始化方法 initTextureOESRenderer，当返回 true 时初始化完成；然后调用渲染方法 processTextureOES，返回值为 GL\_TEXTURE\_EXTERNAL\_OES 类型的纹理：

/\*\*

\* 纹理渲染初始化

\*

\* @param width 纹理宽度

\* @param height 纹理高度

\* @param rotation 纹理是否需要旋转，不需旋转为CLOCKWISE\_0

\* @param isMirror 纹理是否存在镜像

\* @param maxFaces 人脸检测数目上限设置，推荐取值范围为1~5

\*/

public boolean initTextureOESRenderer(int width, int height, HTRotationEnum rotation, boolean isMirror, int maxFaces) ;

/\*\*

\* 渲染

\*

\* @param textureOES 纹理id

\*/

public int processTextureOES(int textureOES);

/\*\*

\* 使用方法

\*/

if (!isRenderInit) {

isRenderInit = HTEffect.shareInstance().initTextureOESRenderer(width, height, rotation, isMirror, maxFaces);

}

int textureId = HTEffect.shareInstance().processTextureOES(textureOES);

（2）GL\_TEXTURE\_2D 纹理格式，首先调用渲染初始化方法 initTextureRenderer，当返回 true 时初始化完成；然后调用渲染方法 processTexture，返回值为 GL\_TEXTURE\_2D 类型的纹理：

/\*\*

\* 纹理渲染初始化

\*

\* @param width 纹理宽度

\* @param height 纹理高度

\* @param rotation 纹理是否需要旋转，不需旋转为CLOCKWISE\_0

\* @param isMirror 纹理是否存在镜像

\* @param maxFaces 人脸检测数目上限设置，推荐取值范围为1~5

\*/

public boolean initTextureRenderer(int width, int height, HTRotationEnum rotation, boolean isMirror, int maxFaces) ;

/\*\*

\* 渲染

\*

\* @param texture2D 纹理id

\*/

public int processTexture(int texture2D);

/\*\*

\* 使用方法

\*/

if (!isRenderInit) {

isRenderInit = HTEffect.shareInstance().initTextureRenderer(width, height, rotation, isMirror, maxFaces);

}

int textureId = HTEffect.shareInstance().processTexture(texture2D);

（3）byte[] 视频帧，首先调用渲染初始化方法 initBufferRenderer ，当返回 true 时初始化完成；然后调用渲染方法 processBuffer ：

/\*\*

\* 初始化buffer渲染器

\*

\* @param format 图像格式

\* @param width 图像宽度

\* @param height 图像高度

\* @param rotation 图像是否需要旋转，不需旋转为CLOCKWISE\_0

\* @param isMirror 图像是否存在镜像

\* @param maxFaces 人脸检测数目上限设置，推荐取值范围为1~5

\*

\* @return 返回初始化结果，成功返回true，失败返回false

\*/

public boolean initBufferRenderer(HTFormatEnum format, int width, int height, HTRotationEnum rotation, boolean isMirror, int maxFaces);

/\*\*

\* 处理buffer数据输入

\*

\* @param buffer 视频帧数据

\*/

public void processBuffer(byte[] buffer);

/\*\*

\* 使用方法

\*/

if (!isRenderInit) {

isRenderInit = HTEffect.shareInstance().initBufferRenderer(format,width, height, rotation, isMirror, maxFaces);

}

HTEffect.shareInstance().processBuffer(buffer);

3.4 销毁  
结束渲染时，为防止内存泄漏的发生，需根据视频帧格式的不同，调用对应的 destroy 方法释放掉资源，调用位置通常在 视频帧回调接口 的销毁处，或者是 Activity ， Fragment 的生命周期结束处，同时将定义的布尔变量 isRenderInit 置为 false:

/\*\*

\* 使用其中一个

\*/

HTEffect.shareInstance().releaseTextureOESRenderer();

HTEffect.shareInstance().releaseTextureRenderer();

HTEffect.shareInstance().releaseBufferRenderer();

/\*

\* 将 bool 置为 false

\*/

isRenderInit = false;

3.5 资源文件  
HTEffect 所有资源均支持放在用户自己的服务器,使用以下方法进行设置：

HTEffect.shareInstance().setResourceUrl(String url);

3.6 人脸检测功能  
HTEffect支持人脸检测功能，可通过以下方法返回人脸数量

int faceNumber = HTEffect.shareInstance().isTracking();