腾讯特效 SDK

功能实践

腾讯云

文档目录

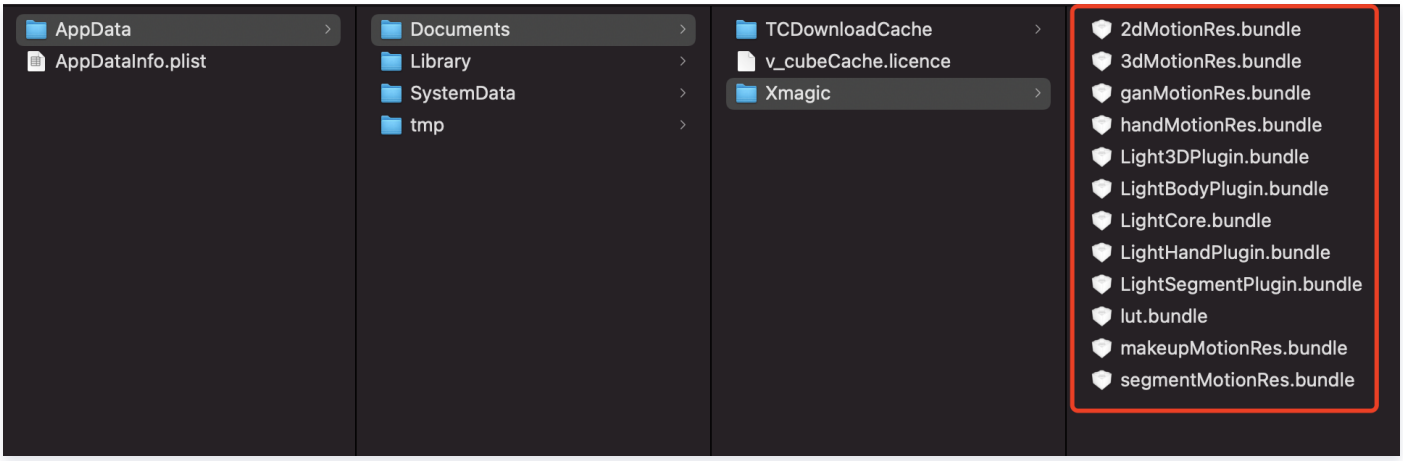
功能实践
SDK 包瘦身
iOS

最近更新时间：2024-05-31 14:41:22

资源动态下载

为了减少包大小，您可以将 SDK 所需的模型资源和动效资源 MotionRes（部分基础版 SDK 无动效资源）改为联网下载。在下载成功后，将上述文件的路径设置给 SDK。

1. 把美颜资源的 ZIP 包上传至云端，生成下载 URL。例如：`https://服务器地址/LightCore.bundle.zip`。
2. 在工程里面使用生成的 URL 下载文件并解压到沙盒（例如：沙盒路径 `Document/Xmagic`）。此时 Document/Xmagic 文件夹里面有 SDK 需要的资源。



3. SDK 初始化时，在 root_path 字段传入上一步的沙盒路径。

```
 NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                              @"root_path":_filePath ,//_filePath为美颜资源下载到本地后的父目
录:Ducument/Xmagic,
                              @"tnn_"
                              @"beauty_config":beautyConfigJson
 };
 // Init beauty kit                                  @"root_path":Ducument/Xmagic,
 self.beautyKit = [[XMagic alloc] initWithRenderSize:_inputSize assetsDict:assetsDict];
```

4. 设置美颜面板各个美颜效果的 icon，在下载的资源文件里面获取对应的 image。

```
NSMutableArray *arrayModels = [NSMutableArray array];
for (NSDictionary* dict in motionArray) {
 BeautyCellModel* model = [BeautyCellModel beautyWithDict:dict];
 // Load default mainbundle path of motionres
 if ([model.title isEqualToString:NSLocalizedString(@"item_none_label",nil)]) {
     model.icon = [NSString stringWithFormat:@"%@/%@.png", [[NSBundle mainBundle] bundlePath],
model.key];
     [arrayModels addObject:model];
 } else {
     if(_useNetResource && _filePath != nil){ //使用网络资源时
         NSString *DirPath = [_filePath stringByAppendingPathComponent:@"2dMotionRes.bundle/"]; //获取美
颜资源的绝对路径
         model.icon = [NSString stringWithFormat:@"%@/%@/template.png", DirPath, model.key];
     }else{
```

```
            model.icon = [NSString stringWithFormat:@"%@/%@/template.png", [[NSBundle mainBundle]
pathForResource:@"2dMotionRes" ofType:@"bundle"], model.key];
        }
        if ([fileManager fileExistsAtPath:model.icon]) {
            [arrayModels addObject:model];
        }
    }
}
```

5. 设置美颜效果的参数传递（参数的具体设置请参见 API 文档）：

```
/// @brief 配置美颜各种效果
/// @param propertyType 效果类型 字符串: beauty, lut, motion
/// @param propertyName 效果名称
/// @param propertyValue 效果数值
/// @param extraInfo 预留扩展，附加额外配置dict
/// @return 成功返回0，失败返回其他
/// @note 具体说明
/**
| 效果类型 | 效果名称 | 效果值 | 说明 | 备注 |
| :---- | :---- |:---- | :---- | :---- |
| beauty | 美颜id名称 | 美颜效果强度数值 |美颜类型配置接口 | 无 |
| lut | 滤镜路径+滤镜名称 | 滤镜强度数值 | 滤镜类型配置接口 | 无 |
| motion | 动效路径名称 | 动效路径 | 动效类型配置接口| **注意**: 如果资源中有zip，请确保传入动效路径为可写路径，否则跟app包走需要手动unzip才可以使用 |
**/
- (int)configPropertyWithType:(NSString *_Nonnull)propertyType withName:(NSString
*_Nonnull)propertyName withData:(NSString*_Nonnull)propertyValue withExtraInfo:(id _Nullable)extraInfo;
```

示例

设置美颜效果

“美颜”和“美体”的特效，不需要做处理，在 SDK 内部会自动使用下载的资源文件。以使用美颜中的美白效果为例，SDK 传参示例：

```
[self.beautyKitRef configPropertyWithType:@"beauty" withName:@"beauty.whiten" withData:@"30"
withExtraInfo:nil];
```

此时，传入到 SDK 的各参数的值分别是：

字段	值
propertyType	beauty
propertyName	beauty.whiten
propertyValue	30
extraInfo	nil

设置滤镜效果

需要对 key 做处理，可以使用内置的本地美颜资源或者网络下载到本地以后的美颜资源：

```
NSString *key = [_model.lutIDs[index] path];
if (key != nil) {
    key = [@"lut.bundle/" stringByAppendingPathComponent:key];//滤镜效果图片的相对路径
}
if(_useNetResource && _filePath != nil){ //如果使用下载的美颜资源
    key = [_filePath stringByAppendingPathComponent:key];//生成效果图片的绝对路径
```

```
}
[self.beautyKitRef configPropertyWithType:@"lut" withName:key withData:[NSString
stringWithFormat:@"%f",value] withExtraInfo:nil];
```

设置滤镜中的白皙效果

使用本地资源和网络资源的传参示例：

字段	使用本地资源时传入的参数	使用网络资源时传入的参数	备注
propertyType	lut	lut	-
propertyName	lut.bundle/n_baixi.png	/var/mobile/Containers/Data/Application/25C7D01A-73F6-4F1B-AEB6-5EE03A221D18/Documents/Xmagic/lut.bundle/n_baixi.png	文件路径
propertyValue	60.000000	60.000000	-
extraInfo	null	null	-

设置动效、美妆、分割效果

需要对 propertyValue 字段做处理，可以使用内置的本地美颜资源或者网络下载到本地以后的美颜资源。

```
NSString *key = [_model.motionIDs[index] key];
        NSString *path = [_model.motionIDs[index] path];
        NSString *motionRootPath = path==nil?[[NSBundle mainBundle] pathForResource:@"MotionRes"
ofType:@"bundle"]:path;
        if(_useNetResource && _filePath != nil){//如果使用下载的美颜资源
            motionRootPath = [_filePath stringByAppendingPathComponent:@"2dMotionRes.bundle"];//生成
2dMotionRes的绝对路径
        }
        [self.beautyKitRef configPropertyWithType:@"motion" withName:key withData:motionRootPath
withExtraInfo:nil];
```

设置 2D 动效—可爱涂鸦的效果

使用本地资源和网络资源的传参示例：

字段	使用本地资源时传入的参数	使用网络资源时传入的参数	备注
propertyType	motion	motion	-
propertyName	video_keaituya	video_keaituya	-
propertyValue	/private/var/containers/Bundle/Application/FD2D7912-E58E-4584-B7E4-8715B8D2338F/BeautyDemo.app/2dMotionRes.bundle	/var/mobile/Containers/Data/Application/25C7D01A-73F6-4F1B-AEB6-5EE03A221D18/Documents/Xmagic/2dMotionRes.bundle	文件路径
extraInfo	nil	nil	-

Android

最近更新时间：2024-05-31 14:41:22

为了减少包体大小，您可将 SDK 所需的 so 库、模型资源改为联网下载，只需要在 SDK 初始化之前下载好这些文件即可。对于滤镜和动效资源，建议在用户点击使用时，点击一项下载一项。

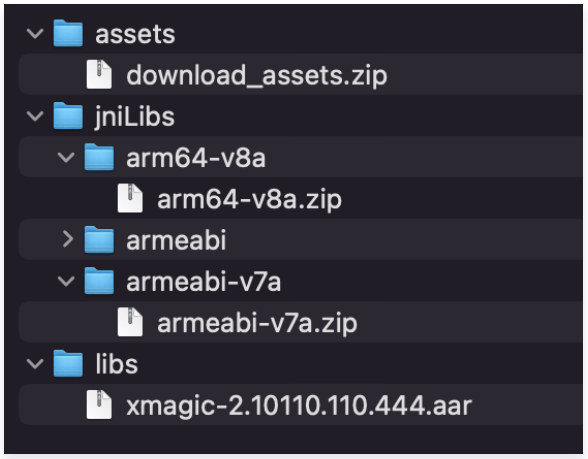
Demo工程： TEBeauty_Download_Example

从 github clone 出 demo工程，根据 TEBeauty_Download_Example/readme 文档配置和运行TEBeauty_Download_Example，以了解动态下载的整体流程。

动态下载 so 库和模型资源

如果您复用 Demo 中的下载代码

1. 将 demo 工程 `com.tencent.demo.download` 目录下的代码拷贝到您的工程。
2. 下载SDK，解压，然后从"SDK"目录找到 `.zip` 格式的压缩包，再次解压，您将看到如下文件：



3. 将 download_assets.zip, arm64-v8a.zip, armeabi-v7a.zip 上传到您的服务器，得到下载地址。计算出这3个 zip 文件的 MD5。将这3个下载地址和 MD5 填在 `ResDownloadConfig.java` 里对应的常量上。
4. 参见 `TEMenuActivity.java` 里的代码，通过 `ResDownloadUtil.getValidLibsDirectory` 检查 so 库是否已经下载好，如果没下载好，则调用 `ResDownloadUtil.checkOrDownloadFiles` 启动下载，下载成功后得到so库的路径sdkLibraryDirectory，然后调用 `XmagicApi.setLibPathAndLoad(sdkLibraryDirectory)` 加载so库。

```
String validLibsDirectory = ResDownloadUtil.getValidLibsDirectory(this, libraryMD5);
if (validLibsDirectory == null) {
    ResDownloadUtil.checkOrDownloadFiles(this, ResDownloadUtil.FILE_TYPE_LIBS, libraryURL,
libraryMD5,
        new TEDownloadListener() {
            @Override
            public void onDownloadSuccess(String directory) {
              sdkLibraryDirectory = directory;
            }

            @Override
            public void onDownloading(int progress) {
            }

            @Override
            public void onDownloadFailed(int errorCode) {
            }
        });
} else {
```

```
    sdkLibraryDirectory = validLibsDirectory;
}
```

5. 参见 `TEMenuActivity.java` 里的代码，通过 `ResDownloadUtil.getValidAssetsDirectory` 检查模型资源是否已经下载好，如果没下载好，则调用 `ResDownloadUtil.checkOrDownloadFiles` 启动下载，模块内部会把这些资源下载、整理、拷贝到 `AppConfig.resPathForSDK` 目录，在 new XmagicApi 时传给 SDK。

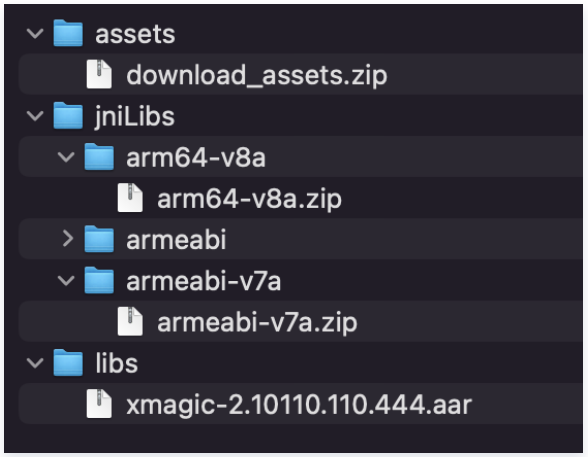
```
String validAssetsDirectory = ResDownloadUtil.getValidAssetsDirectory(this,
ResDownloadConfig.DOWNLOAD_MD5_ASSETS);
if (TextUtils.isEmpty(validAssetsDirectory)) {
    ResDownloadUtil.checkOrDownloadFiles(this, ResDownloadUtil.FILE_TYPE_ASSETS,
            ResDownloadConfig.DOWNLOAD_URL_ASSETS,
            ResDownloadConfig.DOWNLOAD_MD5_ASSETS, new TEDownloadListener() {
                @Override
                public void onDownloadSuccess(String directory) {
                }

                @Override
                public void onDownloading(int progress) {
                }

                @Override
                public void onDownloadFailed(int errorCode) {
                }
            });
} else {

}
```

6. Demo 中默认是开启断点续传功能的（`ResDownloadUtil.java` 的 `ENABLE_RESUME_FROM_BREAKPOINT` 属性为 `true`），可以确保在下载异常中断后，下次继续从中断点接着下载。如果您也想开启断点续传，请确保您的下载服务器支持断点续传能力。检测方法：

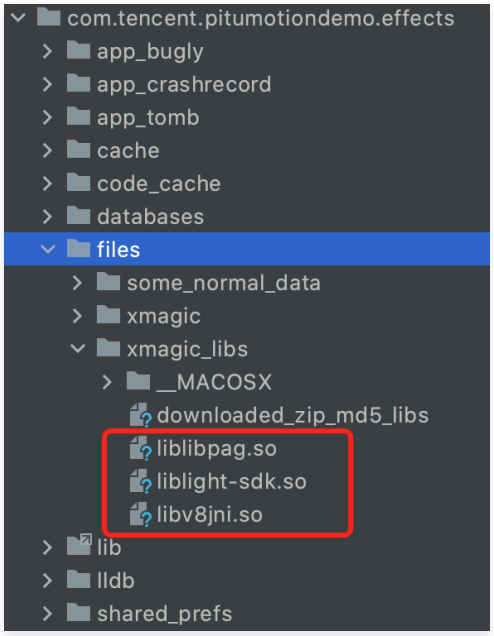
```
判断服务器是否支持断点续传，看Web服务器是否支持Range请求即可。测试方法是在命令行中执行curl命令：
curl -i --range 0-9 https://您的服务器地址/待下载的文件名
例如：
curl -i --range 0-9 https://mediacloud-76607.gzc.vod.tencent-
cloud.com/TencentEffect/Android/2.4.1.119/xmagic_S1-04_android_2.4.1.119.zip
如果返回的内容有Content-Range 字段，则表示服务器支持断点续传。
```

如果您自己做下载

1. 下载SDK，解压，然后从"SDK"目录找到 .zip 格式的压缩包，再次解压，您将看到如下文件。assets 里的模型文件和 jniLibs 里的 so 文件可以动态下载。libs 里的 aar 则需要内置到包里。



2. 下载完 so 文件并解压后，调用 `XmagicApi.setLibPathAndLoad(/path/to/so/files)` 加载 so 库。



> **注意：**
> 强烈建议您将 so 下载到 App 的私有目录，而不是外部存储，以防 so 被清理软件误删。同时，建议您根据用户手机的 CPU 类型按需下载 v8a 或 v7a 的 so，以加快下载速度，这里可以参考 Demo 工程的 TEMenuActivity 的做法。

3. 对于 download_assets.zip 包里的文件，下载完成后，解压，然后调用下面的代码让 SDK 把文件拷贝到正确的目录（`AppConfig.resPathForSDK` 所指向的目录），代码中的 `downloadedDirectory` 是您解压后的文件所在目录。
`addAiModeFiles` 返回的错误码-2表示文件拷贝过程中失败了，可能是手机空间不足或 IO 异常，可尝试重新拷贝或重新下载。

```
  private static boolean organizeAssetsDirectory(String downloadedDirectory) {
    for (String path : XmagicResourceUtil.AI_MODE_DIR_NAMES) {
        if (XmagicApi.addAiModeFiles(downloadedDirectory + File.separator + path,
AppConfig.resPathForSDK) == -2) {
            return false;
        }
    }
    return true;
}
```

> **注意：**

> 当 SDK 版本更新时，对应的 so 和 assets 可能会发生变化，为确保兼容性，您需要重新下载这些文件。建议参见 Demo 中的方式，利用 MD5 进行校验 。

滤镜和动效资源下载

- 每个滤镜都是一张 png 格式的图片，每个动效都是一个文件夹，对于滤镜和动效资源，建议在用户点击使用时，点击一项下载一项。下载成功后，调用 SDK 的 XmagicApi.setEffect 接口，将滤镜路径或动效文件夹的路径设置给 SDK 即可。
- 滤镜和动效资源可以保存在手机任意目录，我们建议您保存在 app 私有目录，防止被误清理。

SDK 集成问题排查
Android

最近更新时间：2024-10-31 14:46:51

1. Android release 包报错找不到某些方法，如何解决?

- 如果您在打 release 包时，启用了编译优化（把 minifyEnabled 设置为 true），会裁掉一些未在 java 层调用的代码，而这些代码有可能会被 native 层调用，从而引起 `no xxx method` 的异常。
- 如果您启用了这样的编译优化，那就要添加这些 keep 规则，防止 xmagic 的代码被裁掉：

```
-keep class com.tencent.xmagic.** { *;}
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class com.tencent.cloud.iai.lib.** { *;}
-keep class com.tencent.beacon.** { *;}
-keep class com.tencent.qimei.** { *;}
-keep class androidx.exifinterface.** { *;}
```

2. Android SDK 集成到宿主工程报 gson 库冲突，如何解决?

在宿主工程 `build.gradle` 文件中添加如下代码：

```
Android{
  configurations {
    all*.exclude group: 'com.google.code.gson'
  }
}
```

3. Android targetSdkVersion 为31或更高时，so 库没有加载成功? 或者无法使用 GAN 类型特效（例如童话脸、童年泡泡糖等）?

Android targetSdkVersion 为31或更高版本时需要在 app 模块下找到 `AndroidManifest.xml` 文件，在 application 标签内加入如下标签：

```
  <uses-native-library
   android:name="libOpenCL.so"
   android:required="false" />
        //true 表示libOpenCL是当前app必需的。如果没有此库，系统将不允许app安装。不建议设置为true，否则可能导致用户无法安装app。
        //false 表示libOpenCL不是当前app必需的。无论有没有此库，都可以正常安装app。如果设备有此库，腾讯特效SDK里的GAN类型特效能正常生效（例如童话脸、国漫脸）。如果设备没有此库，GAN类型不会生效，但也不影响SDK内其他功能的使用。
        //关于uses-native-library的说明，请参考Android 官网介绍：https://developer.android.com/guide/topics/manifest/uses-native-library-element
```

具体请参见 开发指引 。

4. 使用美颜时传递的纹理是横向纹理，如何解决?

可以使用 demo 中工具类 `TextureConverter.java` 的`convert` 方法对纹理进行旋转，转换为竖屏，然后再传递给美颜 SDK。

```
    /**
     * 此方法用于对rgba纹理进行旋转和镜像处理。处理过程为：先顺时针旋转rotation度（可取值0，90，180，270），再进行左右翻转（flipHorizontal）和 上下翻转（flipVertical）。
     * 使用场景：某些推流SDK返回的纹理是横屏纹理或者画面中人物朝向不对，而腾讯特效SDK要求纹理中的人物是正向的，所以可以通过此方法对纹理进行转换。
```

```
     *
     * @param srcID     rgba纹理
     * @param width     纹理宽度
     * @param height    纹理高度
     * @param rotation 需要进行旋转的角度。
     * @return 旋转后的纹理，注意：如果旋转90或者270度，那么宽度需要进行交换。
     */
    public int convert(int srcID, int width, int height, @RotationDegreesValue int rotation, boolean
flipVertical, boolean flipHorizontal)
```

5. 使用美颜时传递的纹理是 oes 纹理，如何解决?

可以使用 demo 中工具类 `TextureConverter.java` 的`oes2Rgba` 方法对纹理进行转换，转换为 RGBA 纹理，然后再传递给美颜 SDK。

```
   /**
     * 此方法用于将oes纹理转换为rgba纹理
     *
     * @param srcID  oes 纹理
     * @param width  纹理宽度
     * @param height 纹理高度
     * @return rgba纹理ID
     */
    public int oes2Rgba(int srcID, int width, int height)
```

6. 如果想使用别的版本的 pag 如何解决? V3.5.0及以上支持

客户集成美颜 SDK 时:

如果是通过 Maven 集成，通过 implementation TencentEffect 就能引入 pag。如果您不想用 TencentEffect 依赖的 pag，可以通过 exclude 排除，然后在自己 app 的 build.gradle 中引入您需要的 pag 版本:

```
implementation ('com.tencent.mediacloud:TencentEffect_S1-04:版本号'){
       exclude group:       "com.tencent.tav", module: "libpag"
}
```

如果是下载美颜 SDK 的 aar 手动集成，在项目中依赖 TencentEffect.aar，这个 aar 是不带 pag 的，您还需要在 app 的 build.gradle 加一句 implementation pag 引入 pag 才能用:

```
implementation 'com.tencent.tav:libpag:4.3.33-noffavc'
```

如果您想动态下载 pag 的 so，请从 pag 官网 找到您需要的版本，下载 aar，将 .aar 重命名为 .zip，解压，剔除其中的 so，再把剩余文件压缩为 .zip，然后重命名为 .aar，最后引入这个不包含 so 的 pag aar，pag 的 so 则联网动态下载。

iOS

最近更新时间：2024-05-31 14:41:22

1. iOS 导入资源运行后报错?

Xcode 12.X 版本编译提示：Building for iOS Simulator, but the linked and embedded framework '.framework'...
在 Build Settings > Build Options > Validate Workspace 改为 Yes，再单击运行。

> **说明：**
> Validate Workspace 改为 Yes 之后编译完成，再改回 No，也可以正常运行，所以这里有这个问题注意下即可。

2. 滤镜设置没反应?

检查下设置的值是否正确，范围为 0~100，可能值太小了效果不明显。

3. iOS Demo 编译，生成 dSYM 时报错?

- 报错信息：

```
PhaseScriptExecution CMake\ PostBuild\ Rules build/XMagicDemo.build/Debug-
iphoneos/XMagicDemo.build/Script-81731F743E244CF2B089C1BF.sh
  cd /Users/zhenli/Downloads/xmagic_s106
  /bin/sh -c /Users/zhenli/Downloads/xmagic_s106/build/XMagicDemo.build/Debug-
iphoneos/XMagicDemo.build/Script-81731F743E244CF2B089C1BF.sh

Command /bin/sh failed with exit code 1
```

- 问题解析：原因是 `libpag.framework` 和 `Masonary.framework` 重签名失败。
- 解决方法：

  1.1 打开 demo/copy_framework.sh 。

  1.2 `$(which cmake)` 改为本地 cmake 绝对路径。

  1.3 签名 `Apple Development: ......` 改为自己的账号。

4. iOS Demo，进入主页显示授权错误?

查看日志中打印的授权失败错误码。如果使用的是本地 License 文件，检查文件是否添加进工程。

5. iOS Demo 编译报错?

- 报错信息：

```
unexpected service error: build aborted due to an internal error: unable to write manifest to-xxxx-
manifest.xcbuild': mkdir(/data, S_IRWXU | S_IRWXG | S_IRWXO): Read-only file system (30):
```

- 解决方法：

  1.1 在 File > Project settings > Build System 选择 Legacy Build System。

  1.2 Xcode 13.0++ 需要在 File > Workspace Settings 勾选 Do not show a diagnostic issue about build system deprecation。

性能调优
低端机性能优化实践教程

最近更新时间：2024-12-13 11:23:12

美颜特效涉及 AI 检测、图像处理、2D 和 3D 图形渲染、动画特效等操作，会占用一定的 CPU 和 GPU 资源。如果在直播、拍摄场景时，系统本身负载已经很高，再叠加上美颜特效，在性能较差的设备上可能出现卡顿、掉帧现象。因此我们整理了低端机性能优化实践教程，尽可能减少美颜特效 SDK 在低端机上的性能开销，确保用户有良好的使用体验。

低端机的定义

SDK 提供了 `getDeviceLevel` 接口获取设备等级（接口说明见：Android ，iOS ），等级取值为1 ~ 5，1为最低端机，5为最高端机。我们建议将等级小于等于3的设备视为低端机。
您也可以根据自身产品数据以及自身App的性能消耗情况，自行判断当前设备的等级。
通过判断不同的设备等级，结合以下措施减少低端机的性能消耗：

措施一：使用 SDK 的 Normal 模式

从 SDK V3.9.0 开始，创建 SDK 时必须指定 EffectMode，它有两个取值： EffectMode_Normal 和 EffectMode_Pro。

- EffectMode_Normal 等价于旧版本 SDK 的“高性能模式”。
- EffectMode_Pro 等价于旧版本 SDK 的默认模式。

建议在低端机上使用 EffectMode_Normal。更多详细说明见：EffectMode（高性能模式）使用指引 。

措施二：关闭 SDK 的某些高级能力

通过 setFeatureEnableDisable 接口关闭某些高级能力：

- `FeatureName.WHITEN_ONLY_SKIN_AREA`
  美白仅对皮肤生效。默认未开启。开启此功能会触发开启“皮肤分割能力”。低端机上不建议开启。
- `FeatureName.SEGMENTATION_SKIN`
  皮肤分割能力，开启后可使磨皮和美白区域更精准，减少对周围环境的影响。SDK在设备等级大于等于4时会默认开启。低端机上不建议开启。
- `FeatureName.SEGMENTATION_FACE_BLOCK`
  人脸遮挡检测能力，开启后可避免妆容画到遮挡物上。SDK在设备等级大于等于5时会默认开启。低端机上不建议开启。
- `FeatureName.SMART_BEAUTY`
  智能美颜(为男性、宝宝减淡美颜美妆效果)。默认未开启。低端机上不建议开启。

此外，“美黑”能力也会触发开启 FeatureName.SEGMENTATION_SKIN 能力。低端机上不建议使用美黑能力。

措施三：使用轻美妆代替风格整妆

轻美妆是腾讯特效 SDK 在V3.9.0版本推出的新功能，与之前的“风格整妆特效”相比，轻美妆性能更好，且能跟其他特效很好地叠加。
更多说明见：轻美妆使用说明 。

措施四：使用性能更好的特效素材

我们提供丰富的特效供客户选择。有些特效比较简单，在低端机上能流畅展示。但有些特效需要消耗较多的 CPU 和 GPU 资源，在低端机上不建议使用，例如 3D 特效，GAN 特效（例如变娃娃脸、变漫画脸），背景分割特效等等。
我们提供了低端机专区，供客户自行选择，详情见美颜特效 Demo。

其他优化措施

除了以上美颜特效相关的优化措施，也可以关注外部影响性能/流畅度的因素：

1、选择合适的分辨率

分辨率越高，SDK 需要处理的像素就越多。在低端机上直播或拍摄时，建议不要超过540P分辨率。

2、设置合适的日志开关

SDK 提供了 setXmagicLogLevel 接口（ Android ， iOS ）用于设置日志等级，默认等级为 WARN 或 INFO。您可以将它进一步提升到 ERROR 级别，以减少日志输出。切记不能设置为 DEBUG 级别，否则大量的日志会影响性能。

3、检查推流帧率

检查是否设置的比较低，建议调整到24fps以上。当您的应用在没有设置美颜的时候画面也不太流畅时，需要检查一下采集模块的相机帧率，可以适当提高相机帧率从而达到画面流畅的效果。如果您使用的是 TRTC，那么可以参见 此文档 调整帧率。

4、检查美颜特效之外的模块的性能

如果在使用美颜特效之前，您的应用就已经很卡了，或者 CPU 占用率已经很高，说明 APP 性能已经出现了问题，这种情况下再使用美颜特效，只能让情况变得更差。所以建议先优化美颜特效之外的模块的性能。

EffectMode（高性能模式）使用指引

最近更新时间：2024-11-20 12:00:32

EffectMode

从 SDK V3.9.0 开始，创建 SDK 时必须指定 EffectMode，它有两个取值： EffectMode_Normal 和 EffectMode_Pro。

- EffectMode_Normal 等价于旧版本 SDK 的“高性能模式”。
- EffectMode_Pro 等价于旧版本 SDK 的默认模式。

二者区别如下：

模式 / 差异	EffectMode_Normal (旧版的高性能模式)	EffectMode_Pro（旧版的默认模式）
性能差异	占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象。适合低端机/中端机/高端机。	性能良好，在中、高端机上能流畅使用。
功能差异	以下项目不可用： 1. 眼部：眼宽、眼高、眼睛位置、祛眼袋、亮眼。 2. 眉毛：角度、距离、高度、长度、粗细、眉峰 3. 嘴部：微笑唇。 4. 面部：收下颌，祛皱、祛法令纹。 5. 鼻子：鼻梁、山根。 6. 其他：美黑、染发、弱光降噪。 另外，磨皮效果与 Pro 模式有差异，具体可在 Demo 中体验。	SDK 全功能可用。

高性能模式

- “高性能模式”是 SDK V3.9.0之前的概念，当时 SDK 有两种模式：高性能模式和默认模式。
- 从V3.9.0开始，高性能模式变成了 EffectMode_Normal，默认模式变成了 EffectMode_Pro。
- 高性能模式与默认模式的区别请参考上文中 EffectMode_Normal 和 EffectMode_Pro 的区别。

V3.9.0及之后如何设置 EffectMode

Android

方式一

如果您是直接使用的 `XmagicApi` 对象，那么请在创建 `XmagicApi` 对象时，在构造方法中指定 EffectMode：

```
public XmagicApi(Context context, EffectMode effectMode, String resDir)

public XmagicApi(Context context, EffectMode effectMode, String resDir, OnXmagicPropertyErrorListener
xmagicPropertyErrorListener)
```

方式二

如果您是使用的 TEBeautyKit 对象，可以调用如下方法开启高性能模式。

```
public TEBeautyKit(Context context, EffectMode effectMode)

public static void create(@NonNull Context context, EffectMode effectMode, @NonNull OnInitListener
initListener)
```

EffectMode 定义如下：

```
public enum EffectMode{
    NORMAL(0),
```

```
    PRO(1);

    private final int value;

    EffectMode(int value) {
        this.value = value;
    }

    public int getValue() {
        return value;
    }
}
```

iOS

方式一

如果您是直接使用的 `XMagic` 对象，那么需要在初始化 `XMagic` 的时候指定 EffectMode，如下代码所示：

```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"effect_mode":@(effectMode)
};
self.xmagic = [[XMagic alloc] initWithRenderSize:CGSizeMake(720, 1280) assetsDict:assetsDict];
```

方式二

如果您是使用的 TEBeautyKit 对象，请在调用 createXMagic 方法时传入 EffectMode 参数。

```
+ (void)createXMagic:(EffectMode)effectMode onInitListener:(OnInitListener _Nullable )onInitListener;
```

EffectMode 的定义如下：

```
typedef NS_ENUM(NSInteger, EffectMode) {
    EFFECT_MODE_NORMAL = 0,
    EFFECT_MODE_PRO = 1,
};
```

Flutter

可以通过调用 `TencentEffectApi` 的 `setDowngradePerformance` 方法开启。

> ⚠ **注意：**
> **此方法需要在开启美颜之前调用，也就是 `TRTC或者Live` 中的 `enableCustomVideoProcess` 方法之前调用。**

uniapp

可以通过调用 `XmagicApi` 的 `setDowngradePerformance` 方法开启。

> ⚠ **注意：**
> **此方法需要在开启美颜之前调用，也就是在 `enableCustomVideoProcess` 方法之前调用。**

V3.9.0之前如何开启高性能模式

Android

方式一

如果您是直接使用的 `XmagicApi` 对象，那么**请在创建 `XmagicApi` 对象之后立即调用**以下接口开启高性能模式：

- SDK 3.7.0及以后：调用 `enableHighPerformance` 方法。
- SDK 3.7.0以前：调用 `setDowngradePerformance` 方法。

方式二

如果您是使用的 TEBeautyKit 对象，可以调用如下方法开启高性能模式。

```
  /**
 * @param context                应用上下文
 * @param isEnableHighPerformance 是否开启高性能模式
 */
public TEBeautyKit(Context context, boolean isEnableHighPerformance)

  /**
 *
 * 异步创建TEBeautyKit对象
 * @param context Android应用上下文
 * @param isEnableHighPerformance 是否开启增强模式
 * @param initListener 初始化回调接口
 */
public static void create(@NonNull Context context, boolean isEnableHighPerformance, @NonNull
OnInitListener initListener)
```

iOS

方式一

如果您是直接使用的 `XMagic` 对象，那么可以在初始化 `XMagic` 的时候开启：

- SDK 3.7.0及以后：请在assetsDict字典中将 `enableHighPerformance` 设置为YES。
- SDK 3.7.0以前：请在assetsDict字典中将 `setDowngradePerformance` 设置为YES。

```
NSDictionary *assetsDict = @{
@"core_name":@"LightCore.bundle",
@"root_path":[NSBundle mainBundle] bundlePath],
@"setDowngradePerformance":@(YES)//YES:开启高性能模式，NO：不开启高性能模式。默认不开启高性能模式。
};
self.xmagic = [[XMagic alloc] initWithRenderSize:CGSizeMake(720, 1280) assetsDict:assetsDict];
```

方式二

如果您是使用的 TEBeautyKit 对象，可以调用如下方法开启高性能模式。

```
 /**
 *
 * 创建TEBeautyKit对象
 * @param isEnableHighPerformance 是否开启高性能模式。YES：开启高性能模式；NO：不开启高性能模式
 * @param initListener 初始化回调接口
 */
+ (void)create:(BOOL)isEnableHighPerformance onInitListener:(OnInitListener _Nullable )onInitListener;
```

Flutter

可以通过调用 `TencentEffectApi` 的 `setDowngradePerformance` 方法开启。

> **注意：**
> 此方法需要在开启美颜之前调用，也就是 `TRTC`或者`Live` 中的 `enableCustomVideoProcess` 方法之前调用。

uniapp

可以通过调用 `XmagicApi` 的 `setDowngradePerformance` 方法开启。

> **注意：**
> 此方法需要在开启美颜之前调用，也就是在 `enableCustomVideoProcess` 方法之前调用。

性能问题排查

最近更新时间：2024-05-31 14:41:22

如果您的应用使用美颜时发现美颜处理过程耗时较长，可通过如下方法进行排查。

第一步：检查传入美颜画面的分辨率

- 原因：分辨率是指图像或视频的像素数量，通常以宽度和高度来表示。美颜处理涉及对图像进行复杂的算法计算和处理，例如磨皮、美白、去瑕疵等。因此，分辨率的大小会直接影响美颜处理的时长。
- 较高的分辨率意味着图像中有更多的像素，需要更多的计算和处理。这会导致美颜处理所需的时间更长。相比之下，较低的分辨率意味着图像中的像素较少，处理所需的计算量也较小，因此美颜处理的时长会相对较短。
- 此外，美颜处理通常涉及对图像的多个区域进行处理，例如人脸检测和人脸特征点定位。在较高分辨率的图像中，需要处理更多的像素和更复杂的图像细节，这可能需要更多的时间来完成。
- 因此，需要权衡分辨率和美颜效果之间的关系，以获得满意的处理速度和图像质量。

第二步：检查日志开关

当日志设置为 `Log.DEBUG` 时，美颜在处理过程中会打印大量的日志信息，从而影响性能，所以应用 release 包时设置为 `LOG.WARN` 。

第三步：3D/Gan 贴纸比较耗性能，在低端机上表现可能存在卡顿问题，可以根据实际情况是否开启使用。

第四步：画面卡顿问题，检查推流帧率是否设置的比较低，建议调整到24fps以上。

- 当您的应用在没有设置美颜的时候画面也不太流畅时，需要检查一下RTC模块的相机帧率，可以适当提高相机帧率从而达到画面流畅的效果。
- 如果您使用的是 TRTC，那么可以参见 此文档 调整帧率。

效果调优
增强模式使用指引

最近更新时间：2024-06-03 14:56:21

增强模式是什么?

SDK 建议设置的各项美颜参数范围是0 ~ 100或-100 ~ 100（ 见 美颜参数说明 ），在此范围内调整数值，通常都能达到令人满意的美颜效果。如果将强度调整到最大值或最小值之后仍然无法满足需求，则可以考虑使用增强模式，增强模式可以让美颜效果更明显，例如磨皮更明显、瘦脸瘦得更多等。

如何使用增强模式

在 SDK 3.5.0版本之后，我们优化了增强模式的使用方式，您**只需要设置更大的数值给 SDK 即可**，例如建议的数值范围是-100 ~ 100，那您可以设置-120 ~ 120给 SDK。

Android

1. **如果您使用了我们的 UI 组件 TEBeautyKit:**
   请调用 `TEBeautyKit` 的 `enableEnhancedMode` 方法，调用后，TEBeautyKit就会将面板上显示的数值乘以合适的倍数再设置给SDK。例如在UI面板上设置的瘦脸数值是80，TEBeautyKit会将它乘以 1.2 变成 96 再设置给 SDK。
2. **如果您没有使用 TEBeautyKit 而是直接使用 XmagicApi:**
   调用 XmagicApi 的 setEffect 方法时，将 value 数值乘以合适的倍数即可。

iOS

1. **如果您使用了我们的 UI 组件 TEBeautyKit:**
   使用 `TEPanelView` ，调用 `setEnhancedMode` 方法，调用后，TEBeautyKit 就会将面板上显示的数值乘以合适的倍数再设置给 SDK。例如在 UI 面板上设置的瘦脸数值是80，TEBeautyKit 会将它乘以1.2变成96再设置给 SDK。

```
/**
*
* 开启增强模式
* @param enhancedMode 是否开启增强模式。YES：开启增强模式；NO：不开启增强模式。默认不开启增强模式。
*/
[self.tePanelView setEnhancedMode:YES];
```

2. **如果您没有使用 TEBeautyKit而是直接使用 `XMagic` 对象:**
   调用 setEffect 方法时，将 value 数值乘以合适的倍数即可。

Flutter

1. 调用 `TencentEffectApi` 的 `enableEnhancedMode` 方法开启增强模式。
2. 使用 `setEffect` 方法设置美颜参数时， `effectValue` 的最大值可以为下表推荐的最大值。

```
void setEffect(String effectName,int effectValue,String? resourcePath,Map<String,String>?
extraInfo);
```

uniapp

1. 调用 `XmagicApi` 的 `enableEnhancedMode` 方法开启增强模式。
2. 使用 `setEffect` 方法设置美颜参数时， `effectValue` 的最大值可以为下表推荐的最大值。

```
/**
 * 更新美颜对象
 * @param effect 对象结构如下
 * {
 *      effectName:"",不为空的字符串，参考美颜参数表
 *      effectValue: 数值，一般为-100---100的值，可参考官网的美颜参数表
 *      resourcePath: 资源文件的路径，请参考美颜参数表
https://cloud.tencent.com/document/product/616/103616
 *      extraInfo:一个map集合，具体数值请参考美颜参数表
 * }
 */
static setEffect(effect)
```

增强模式推荐的增强倍数

我们提供了一份增强倍数的参考值，不建议超出我们的推荐值，否则美颜效果可能变差。参考值见下：

美颜项名称	建议最大增强倍数
美白，短脸，V脸，眼距，鼻子位置，祛法令纹，口红，立体	1.3倍
亮眼	1.5倍
腮红	1.8倍
其他	1.2倍

TEBeautyKit 在 `DefaultEnhancingStrategy.java` 中设置了上述增强倍数，您可以按需修改。如果是直接使用 Android 的 XmagicApi 或 iOS 的 XMagic ，那么在 setEffect 时，将 value 数值乘以合适的倍数即可。

效果问题排查

最近更新时间：2024-09-11 21:11:52

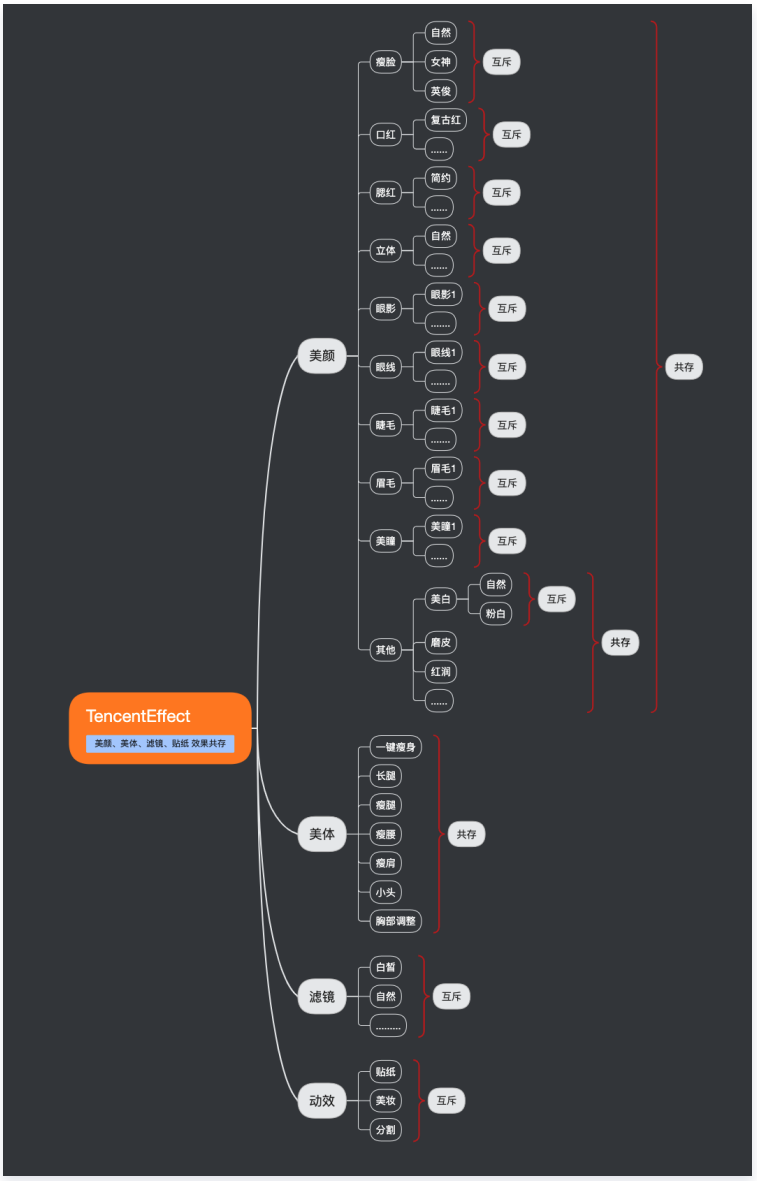
1. 画面出现噪点怎么办？

如果您在灯光弱的环境下，画面中出现了噪点，可以开启降噪属性。

2. 分割效果不太好怎么办？

在使用背景分割效果时，建议背景不要太复杂，背景颜色和衣服颜色不能太相似，否则分割效果会降低。

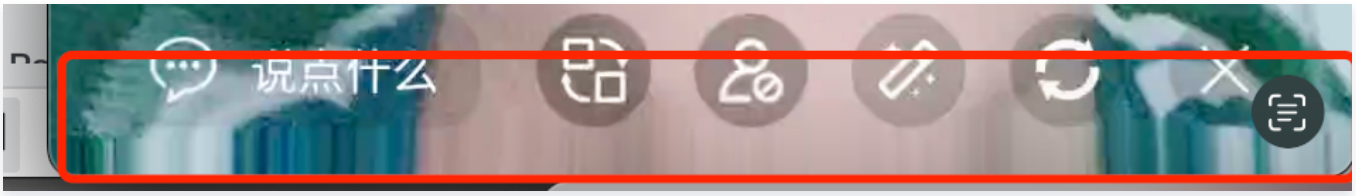
3. 美妆素材中的美颜跟美颜关系是什么？



4. 使用某一项美颜没有效果怎么办？

这里可能是license权限问题，可能是参数问题（例如滤镜和动效的路径问题），建议检查属性参数。

5. 美颜之后画面边缘模糊问题怎么办？



这种情况是因为开了瘦脸特效（瘦脸特效会导致拉伸脸部周围像素），如果脸比较靠近屏幕边缘，边缘的拉伸就更多。可通过裁剪画面边缘的方式进行处理，裁剪方法可以参考 demo。

6. 横屏时人脸没有效果怎么办?

检查画面中人脸方向，设置对应的偏移角度。

Android

1. Android 中可以使用 `readTexture` 方法获取当前画面，查看画面中人脸的方向，根据下图设置对应的角度。

```
  public static Bitmap readTexture(int texture, int width, int height) {
    int[] frame = new int[1];
    GLES20.glGenFramebuffers(1, frame, 0);
    GLES20.glBindFramebuffer(GLES20.GL_FRAMEBUFFER, frame[0]);
    GLES20.glFramebufferTexture2D(GLES20.GL_FRAMEBUFFER, GLES20.GL_COLOR_ATTACHMENT0,
GLES20.GL_TEXTURE_2D, texture, 0);
    byte[] data = new byte[width * height * 4];
    ByteBuffer buffer = ByteBuffer.wrap(data);
    GLES20.glPixelStorei(GLES20.GL_PACK_ALIGNMENT, GLES20.GL_TRUE);
    GLES20.glReadPixels(0, 0, width, height, GLES20.GL_RGBA, GLES20.GL_UNSIGNED_BYTE, buffer);
    Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
    bitmap.copyPixelsFromBuffer(buffer);
    GLES20.glBindFramebuffer(GLES20.GL_FRAMEBUFFER, 0);
    GLES20.glDeleteFramebuffers(1, frame, 0);
    return bitmap;
}
```

2. Android 中调用 `setImageOrientation` 方法。

iOS

1. iOS中，可以使用 `readTexture` 方法获取当前画面，查看画面中人脸的方向，根据下图设置对应的角度。

```
  #import <OpenGLES/ES2/gl.h>
-(void)readTexture:(int)textureId width:(int)width height:(int)height{
    glBindTexture(GL_TEXTURE_2D, textureId);
    GLuint framebuffer;
    glGenFramebuffers(1, &framebuffer);
    glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);
    glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, textureId, 0);
```

```
    GLenum status = glCheckFramebufferStatus(GL_FRAMEBUFFER);
    if (status != GL_FRAMEBUFFER_COMPLETE) {
        NSLog(@"Framebuffer is not complete.");
    }
    GLubyte *pixels = (GLubyte *)malloc(width * height * 4 * sizeof(GLubyte));
    glReadPixels(0, 0, width, height, GL_RGBA, GL_UNSIGNED_BYTE, pixels);
    glBindFramebuffer(GL_FRAMEBUFFER, 0);
    glDeleteFramebuffers(1, &framebuffer);
    CVPixelBufferRef pixelBuffer = NULL;
    CVPixelBufferCreateWithBytes(NULL, width, height, kCVPixelFormatType_32BGRA, pixels, width * 4,
NULL, NULL, NULL, &pixelBuffer);
    free(pixels);
    CVPixelBufferRelease(pixelBuffer);
}
```

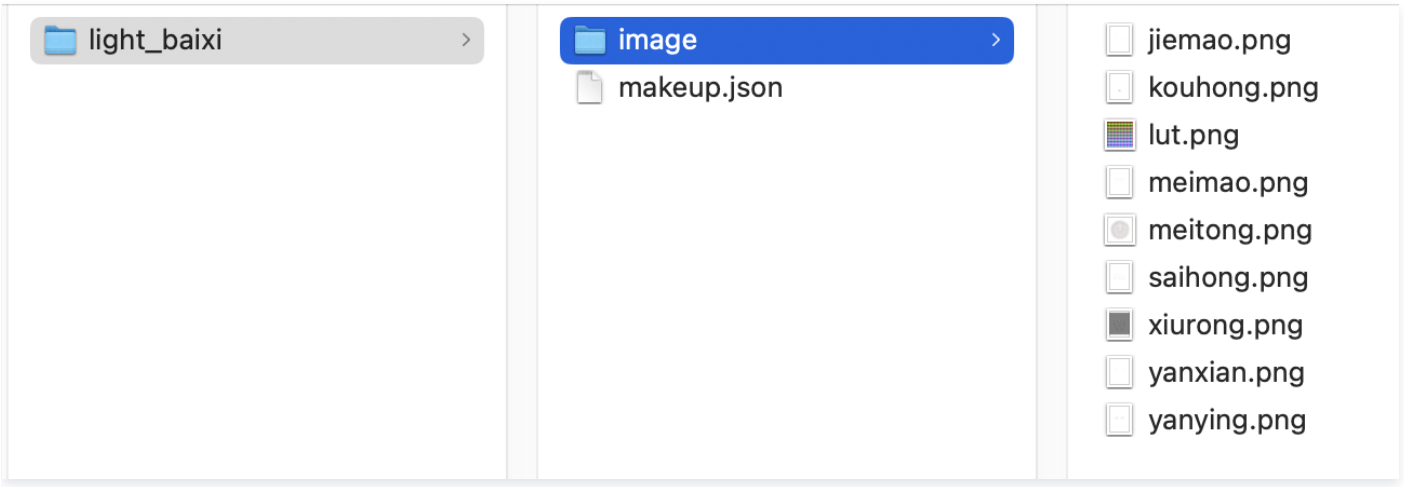
2. iOS中调用 `setImageOrientation` 方法。

轻美妆使用说明

最近更新时间：2025-02-20 16:16:52

什么是轻美妆

轻美妆是腾讯特效 SDK 在V3.9.0版本推出的新功能。一套轻美妆里最多可包含这些美妆项目：滤镜、口红、腮红、立体、眼影、眼线、睫毛、眉毛、美瞳、双眼皮、卧蚕。轻美妆本质上跟SDK已有的“单点美妆”是同一个能力，可以理解为把多个单点美妆搭配组合在了一起。
与之前的“整妆特效”相比，**轻美妆性能更好，且能跟其他特效很好地叠加**。
一套轻美妆素材包含若干张美妆图片和一个 json 配置文件，例如“light_baixi”这套轻美妆的配置如下：



如何使用轻美妆

请调用 SDK 的 setEffect 接口使用轻美妆：

- effectName 为 `EFFECT_LIGHT_MAKEUP`。
- effectValue 为妆容强度，取值0 ~ 100。
- resourcePath 为轻美妆素材路径，即： path/to/your_light_makeup。
- extraInfo 是可选的，如果您想只修改轻美妆里的滤镜强度而不修改妆容强度，则在 extraInfo 里添加一对 key-value，key 为"makeupLutStrength"，value 为滤镜强度，取值为0 ~ 100，注意 value 也是字符串格式的。

注意事项

1、轻美妆与单点美妆的关系

轻美妆本质上是单点美妆的集合，因此后设置的会覆盖先设置的，具体如下：

- 场景1：如果先设置了若干单点美妆，再设置了一套轻美妆，则轻美妆效果会覆盖单点美妆的效果。
- 场景2：如果先设置了一套轻美妆（假设里面配置了口红、眼影、眉毛等），再设置单点美妆（例如口红），则最终效果是：新设置的这个口红 + 轻美妆里的眼影 + 轻美妆里的眉毛。

  对于场景2，我们 Demo 中的处理方式是：设置单点美妆时，清空轻美妆。您可以根据您的产品实际情况选择是否清空轻美妆。

2、轻美妆与其他特效的叠加关系

轻美妆可以和其他任意特效叠加，包括：美颜、美型、美体、贴纸、虚拟背景、运镜特效等。

3、轻美妆与贴纸特效里的“风格整妆”的关系

在V3.9.0之前，我们的整妆都是“风格整妆”，它本质上是一种特效，无法跟其他特效叠加，或叠加后不符合预期。而轻美妆可以跟其他任意特效叠加，也可以与风格整妆叠加（但不建议这么做）。在V3.9.0及之后的 Demo 中，我们把“风格整妆”的体验入口与 2D 贴纸和 3D 贴纸的入口放在了一起。

素材使用
素材集成指引
Android

最近更新时间：2024-05-31 14:41:22

滤镜

每个滤镜都是一张 png 格式的图片，使用时，您需要将图片路径传给 SDK。具体做法如下：

场景一：如果您使用了 TEBeautyKit

TEBeautyKit 是腾讯特效的 UI 面板库，用于客户快速方便的使用和管理美颜功能。
操作步骤见下：

1. 参见文档 接入 TEBeautyKit。
2. 添加滤镜资源

   将新增加的滤镜图片放到您工程的 `assets/lut` 目录，然后，修改面板配置文件 `assets/beauty_panel/lut.json`，参见json中已有的内容新增加一项。APP 运行时，调用 `TEBeautyKit` 的 `copyRes` 方法，会把滤镜图片从 assets 目录 copy 到 `lut.json` 里配置的 downloadPath 目录。
3. 配置滤镜图标

   `lut.json` 的 icon 字段是该滤镜的图标，请把图标放在 `assets/beauty_panel/panel_icon/lut_icon` 目录。icon 字段的值也可以是图标的 URL，以 `http` 或 `https` 开头，TEBeautyKit 会从网络拉取这个图标。
4. 配置滤镜资源

   `lut.json` 的 resourceUri 字段是滤镜图片在 app 私有目录的保存路径，请参见json中已有的项目进行配置并把 resourceUri 的后缀"xxx.png"改为新增加的这个滤镜文件名，确保不会跟 `lut.json` 里的已有的滤镜冲突。resourceUri 字段也可以是滤镜图片的 URL，以 `http` 或 `https` 开头，点击后会联网下载，并保存在 `lut.json` 里配置的 downloadPath 目录。

场景二：如果您未使用 TEBeautyKit，而是直接集成腾讯特效 SDK

1. 请将新增加的滤镜图片放到您工程的 assets 的任意目录，然后在 APP 初始化时，将它 copy 到 app 私有目录或 SD 卡，得到图片的路径，记为 `/path/to/your/lut_xxx.png`。为简化操作，建议您把图片放到 assets/lut 目录，然后从 demo 工程中把 TEBeautyKit 的 copyRes 代码 copy 过来使用。
2. 使用滤镜时，调用 SDK 的 setEffect 方法，将滤镜图片路径传给 SDK。

动效贴纸

每个动效都是一个文件夹，使用时，您需要将该文件夹的路径传给 SDK。具体做法如下：

场景一：如果您使用了 TEBeautyKit

TEBeautyKit 是腾讯特效的 UI 面板库，用于客户快速方便的使用和管理美颜功能。
操作步骤见下：

1. 参见文档 接入 TEBeautyKit。
2. 添加动效素材

   将新增加的动效文件夹放到您工程的 `assets/MotionRes` 目录，然后，修改面板配置文件 `assets/beauty_panel/motions.json`，参见已有的内容新增加一项。APP 运行时，调用 `TEBeautyKit` 的 `copyRes` 方法，会把动效文件夹从 assets 目录 copy 到 `motions.json` 里配置的 downloadPath 目录。
3. 配置动效图标

   `motions.json` 的 icon 字段是该动效的图标，请把图标放在 `assets/beauty_panel/panel_icon/motions_icon` 目录。icon 字段的值也可以是图标的 URL，以 `http` 或 `https` 开头，TEBeautyKit 会从网络拉取这个图标。
4. 配置动效素材

   `motions.json` 的 resourceUri 字段是动效在 app 私有目录的保存路径，请参见已有的项目进行配置，并确保不会跟 `motions.json` 里的已有的动效冲突。resourceUri 字段也可以是动效压缩包的 URL，以 `http` 或 `https` 开头，点击后会联网下载，并保存在 `motions.json` 里配置的 downloadPath 目录。

场景二：如果您未使用 TEBeautyKit，而是直接集成腾讯特效 SDK

请将新增加的动效文件夹放到您工程的 assets 的任意目录，然后在 APP 初始化时，将它 copy 到 app 私有目录或 SD 卡，得到动效的路径，记为 `/path/to/your/motion` 。使用动效时，调用 SDK 的 setEffect 方法，将该路径传给 SDK。

美妆、背景分割动效

与上文中的动效贴纸用法是一样的，二者对应的 json 文件分别是 `makeup.json` 和 `segmentation.json` 。

iOS

最近更新时间：2024-06-03 14:56:21

滤镜

每个滤镜都是一张 png 格式的图片，使用时，您需要将图片路径传给 SDK。

场景一：如果您使用了 TEBeautyKit

`TEBeautyKit` 是腾讯特效的 UI 面板库，用于客户快速方便的使用和管理美颜功能。接入步骤见下：

1. 参见文档 接入 TEBeautyKit 。
2. 添加滤镜素材

   将新增加的滤镜图片放到您工程的 `lut.bundle` 目录，然后，修改面板配置文件 `TEBeautyKit/Assets/json/lut.json` ，参见json中已有的内容新增加一项。
3. 配置滤镜图标

   `lut.json` 的 icon 字段是该滤镜的图标，请把图标放在 `TEBeautyKit/Assets/BeautyRes` 目录。icon 字段的值也可以是图标的 URL，以 http 或 https 开头， `TEBeautyKit` 会从网络拉取这个图标。
4. 配置滤镜资源

   `lut.json` 的 `resourceUri` 字段是滤镜图片在 app 私有目录的保存路径，请参见json中已有的项目进行配置并把 `resourceUri` 的后缀 "xxx.png" 改为新增加的这个滤镜文件名，确保不会跟 lut.json 里的已有的滤镜冲突。 `resourceUri` 字段也可以是滤镜图片的 URL，以 http 或 https 开头，点击后会联网下载，并保存在 `lut.json` 里配置的 `downloadPath` 目录。

场景二：直接集成腾讯特效 SDK

1. 请将新增加的滤镜图片放到您工程的 `lut.bundle` 目录。如果采用动态下载的方案，把滤镜图片下载到沙盒中，记录滤镜图片的路径。
2. 使用滤镜时，调用 SDK 的 `setEffect` 方法，将滤镜图片路径传给 SDK。操作方法参见 美颜参数说明 。

动效贴纸

每个动效都是一个文件夹，使用时，您需要将该文件夹的路径传给SDK。具体做法如下：

场景一：如果您使用了TEBeautyKit

TEBeautyKit 是腾讯特效的 UI 面板库，用于客户快速方便的使用和管理美颜功能。

1. 参见文档 接入 TEBeautyKit 。
2. 添加动效资源

   请将新增加的动效文件夹放到您工程对应的 `resource bundle` 目录： `2dMotionRes.bundle`中 是2D动效， `3dMotionRes.bundle` 中是3D动效， `ganMotionRes.bundle` 中是趣味动效， `handMotionRes.bundle` 中是手势动效， 然后，修改面板配置文件 `TEBeautyKit/Assets/json/motions.json` ，参考已有的内容新增加一项。
3. 配置动效icon

   `motions.json` 的 `icon` 字段是该动效的图标，请把图标放在 `TEBeautyKit/Assets/BeautyRes` 目录。icon字段的值也可以是图标的URL，以 `http`或`https` 开头， `TEBeautyKit` 会从网络拉取这个图标。
4. 配置动效资源

   `motions.json` 的 `resourceUri` 字段是动效在app私有目录的保存路径，请参考已有的项目进行配置，并确保不会跟 `motions.json` 里的已有的动效冲突。 `resourceUri` 字段也可以是动效压缩包的URL，以 `http`或`https` 开头，点击后会联网下载，并保存在 `motions.json` 里配置的 `downloadPath` 目录，动效压缩包需要解压以后才能使用。

场景二：如果您未使用TEBeautyKit，而是直接集成腾讯特效SDK

请将新增加的动效文件夹放到您工程对应的 `resource bundle` 目录： `2dMotionRes.bundle` 中是2D动效， `3dMotionRes.bundle` 中是3D动效， `ganMotionRes.bundle` 中是趣味动效， `handMotionRes.bundle` 中是手势动效，在腾讯特效SDK version 3.6.0及以前的版本，如果是加密的动效文件，需要把动效文件拷贝到沙箱，记录这个动效文件的路径。如果采用动态下载的方案，把动效文件下载到沙盒中并解压，记录解压后的动效文件夹的路径。使用动效时，调用SDK的setEffect方法，将该路径传给SDK，详见： 美颜参数表 。

美妆、背景分割动效

与上文中的动效贴纸用法是一样的，二者对应的json文件分别是 `makeup.json` 和 `segmentation.json` 。

素材叠加指引

最近更新时间：2024-05-31 14:41:22

动效素材叠加是指多个动效素材可以同时生效。

素材叠加注意事项：

1. 客户需要自行管理素材之间是否适合叠加。举两个例子：
   例1：特效 A 是变成贵妃脸，特效 B 是变成童话脸，这两个特效叠加后可能会导致画面非常别扭。
   例2：特效 A 是个兔耳朵，特效 B 是猪耳朵，两个叠加后，就有两种耳朵。
   例1和例2这两种情况不适合叠加。如果特效 A 是兔耳朵，特效 B 是送一个飞吻，这两个特效不会冲突，就适合叠加。
2. 只支持简单素材之间的叠加。简单素材是指只有单动效能力、或者单美妆效果、或者单抠背等，复杂素材是指包含了多种效果。简单素材和复杂素材没有明确的界定，建议客户充分测试后，自行管理哪些素材之间可以叠加，哪些不能叠加。
3. 叠加时，有动作触发的特效（例如伸出手触发某个特效、微笑触发某个特效等）属于复杂特效，需要放在前面，简单特效放在后面叠加在它之上。
4. 使用示例：主播使用了特效 A，然后观众送礼物特效 B，B 要叠加在 A 之上，一段时间后 B 消失，恢复成特效 A。那么设置步骤如下：
   4.1 设置特效 A，mergeWithCurrentMotion 设置为 false。
   4.2 设置特效 B，mergeWithCurrentMotion 设置为 true。
   4.3 一小段时间后，再设置 A，mergeWithCurrentMotion 设置为 false。

如何配置同时生效?

V3.5.0及以上

1. 如果您使用 `setEffect` 方法来更新美颜属性，要实现素材叠加功能，可以在 `extrainfo` 中添加 `mergeWithCurrentMotion` 字段设置为 `"true"`
2. 如果使用的是 `updateProperty` 方法，那么可参见 V3.0.1 中列举的方法。

V3.0.1及以上

Android:

如果想要某个动效/美妆/分割素材叠加在当前素材上，则将该素材 XmagicProperty 对象的 mergeWithCurrentMotion 设置为 true。
XMagicProperty 对象的其他属性设置见 美颜参数设置 。

```
XmagicProperty xmagicProperty = new
XmagicProperty(XmagicProperty.Category.MOTION,"video_keaituya",mResPath+"xmagic/MotionRes/2dMotionRes/
video_keaituya",null,null);
xmagicProperty.mergeWithCurrentMotion = true;
mXMagicApi.updateProperty(xmagicProperty);
```

iOS:

如果想要某个动效/美妆/分割素材叠加在当前素材上，则设置该素材时，在 withExtraInfo 的字典中设置 mergeWithCurrentMotion 为 true，示例如下:

```
NSString *key = _xmagicUIProperty.property.Id;
NSString *value = [[NSBundle mainBundle] pathForResource:@"makeupMotionRes" ofType:@"bundle"];
NSDictionary* extraInfo = @{@"mergeWithCurrentMotion":@(true)};
 [self.beautyKitRef configPropertyWithType:@"motion" withName:key withData:[NSString
stringWithFormat:@"%@",value] withExtraInfo:extraInfo];
```

美颜参数说明
Android & iOS

最近更新时间：2025-02-13 11:32:22

当您使用 `setEffect` 函数更新美颜效果时，可参考如下参数表。参数表中的 `effectName` 常量定义在Android中位于 `XmagicConstant.java` 文件，iOS 位于 `XmagicConstant.h` 文件。

注意： 如果您使用的 SDK 版本是 `V3.3.0` 及之前版本，请参见 Android 旧版美颜参数表 ， iOS 旧版美颜参数表 。

美颜、美体

		effectName		effectValue	resourcePath
		常量名	常量值	效果强度	资源路径
美颜	美白-靓白 (V3.9.0)	BEAUTY_WHITEN0	beauty.lutFoundationAlpha0	0 ~ 100	V3.9.0以前：无 V3.9.0及之后：【可选】自定义美白滤镜路径
	美白-自然	BEAUTY_WHITEN	beauty.lutFoundationAlpha	0 ~ 100	V3.9.0以前：无 V3.9.0及之后：【可选】自定义美白滤镜路径
	美白-粉白	BEAUTY_WHITEN2	beauty.lutFoundationAlpha2	0 ~ 100	V3.9.0以前：无 V3.9.0及之后：【可选】自定义美白滤镜路径
	美白-冷白	BEAUTY_WHITEN3	beauty.lutFoundationAlpha3	0 ~ 100	V3.9.0以前：无 V3.9.0及之后：【可选】自定义美白滤镜路径
	美黑 (V3.7.0)	BEAUTY_BLACK_1	beauty.lutBlackAlpha1	0 ~ 100	无
	小麦色 (V3.7.0)	BEAUTY_BLACK_2	beauty.lutBlackAlpha2	0 ~ 100	无
	磨皮	BEAUTY_SMOOTH	smooth.smooth	0 ~ 100	无
	红润	BEAUTY_ROSY	smooth.rosy	0 ~ 100	无
画面调整	对比度	BEAUTY_CONTRAST	beauty.imageContrastAlpha	-100 ~ 100	无
	饱和度	BEAUTY_SATURATION	smooth.saturation	-100 ~ 100	无
	清晰度	BEAUTY_CLEAR	beauty.lutClearAlpha	0 ~ 100	无
	锐化	BEAUTY_SHARPEN	smooth.sharpen	0 ~ 100	无
	亮度 (V3.8.0)	BEAUTY_IMAGE_BRIGHTNESS	beauty.imageBrightness	-100 ~ 100	无
	弱光降噪 (V3.6.0)	BEAUTY_IMAGE_DENOISE	postEffect.denoise	0 ~ 100	无
	色温	BEAUTY_IMAGE_WARMTH	beauty.imageWarmth	-100 ~ 100	无
	色调	BEAUTY_IMAGE_TINT	beauty.imageTint	-100 ~ 100	无
高级美型	大眼	BEAUTY_ENLARGE_EYE	basicV7.enlargeEye	0 ~ 100	无
	亮眼	BEAUTY_EYE_LIGHTEN	beauty.eyeLighten	0 ~ 100	无
	眼距	BEAUTY_EYE_DISTANCE	basicV7.eyeDistance	-100 ~ 100	无
	眼角	BEAUTY_EYE_ANGLE	basicV7.eyeAngle	-100 ~ 100	无

眼宽	BEAUTY_EYE_WIDTH	basicV7.eyeWidth	-100 ~ 100	无
眼高	BEAUTY_EYE_HEIGHT	basicV7.eyeHeight	-100 ~ 100	无
眼睛位置 (V3.8.0)	BEAUTY_EYE_POSITION	basicV7.eyePosition	-100 ~ 100	无
外眼角 (V3.9.0)	BEAUTY_EYE_OUT_CORNER	basicV7.eyeOutCorner	-100 ~ 100	无
祛眼袋	BEAUTY_FACE_REMOVE_EYE_BAGS	beauty.removeEyeBags	0 ~ 100	无
眉毛角度	BEAUTY_EYEBROW_ANGLE	basicV7.eyebrowAngle	-100 ~ 100	无
眉毛距离	BEAUTY_EYEBROW_DISTANCE	basicV7.eyebrowDistance	-100 ~ 100	无
眉毛高度	BEAUTY_EYEBROW_HEIGHT	basicV7.eyebrowHeight	-100 ~ 100	无
眉毛长度	BEAUTY_EYEBROW_LENGTH	basicV7.eyebrowLength	-100 ~ 100	无
眉毛粗细	BEAUTY_EYEBROW_THICKNESS	basicV7.eyebrowThickness	-100 ~ 100	无
眉峰	BEAUTY_EYEBROW_RIDGE	basicV7.eyebrowRidge	-100 ~ 100	无
瘦鼻	BEAUTY_NOSE_THIN	basicV7.thinNose	0 ~ 100	无
鼻翼	BEAUTY_NOSE_WING	basicV7.noseWing	-100 ~ 100	无
鼻子位置	BEAUTY_NOSE_HEIGHT	basicV7.noseHeight	-100 ~ 100	无
鼻梁	BEAUTY_NOSE_BRIDGE_WIDTH	basicV7.noseBridgeWidth	-100 ~ 100	无
山根	BEAUTY_NASION	basicV7.nasion	-100 ~ 100	无
白牙	BEAUTY_TOOTH_WHITEN	beauty.toothWhiten	0 ~ 100	无
嘴型	BEAUTY_MOUTH_SIZE	basicV7.mouthSize	-100 ~ 100	无
嘴唇厚度	BEAUTY_MOUTH_HEIGHT	basicV7.mouthHeight	-100 ~ 100	无
嘴唇宽度	BEAUTY_MOUTH_WIDTH	basicV7.mouthWidth	-100 ~ 100	无
嘴唇位置	BEAUTY_MOUTH_POSITION	basicV7.mouthPosition	-100 ~ 100	无
微笑唇	BEAUTY_SMILE_FACE	basicV7.smileFace	-100 ~ 100	无
窄脸	BEAUTY_FACE_THIN	basicV7.thinFace	0 ~ 100	无
瘦脸-自然	BEAUTY_FACE_NATURE	basicV7.natureFace	0 ~ 100	无
瘦脸-女神	BEAUTY_FACE_GODNESS	basicV7.godnessFace	0 ~ 100	无
瘦脸-英俊	BEAUTY_FACE_MALE_GOD	basicV7.maleGodFace	0 ~ 100	无
V脸	BEAUTY_FACE_V	basicV7.vFace	0 ~ 100	无
收下颌	BEAUTY_FACE_JAW	basicV7.faceJaw	0 ~ 100	无
短脸	BEAUTY_FACE_SHORT	basicV7.shortFace	0 ~ 100	无
脸型	BEAUTY_FACE_BASIC	liquefaction.basic3	0 ~ 100	无
下巴	BEAUTY_FACE_THIN_CHIN	basicV7.chin	-100 ~ 100	无

	额头	BEAUTY_FACE_FOREHEAD	basicV7.forehead	-100 ~ 100	无
	祛皱	BEAUTY_FACE_REMOVE_WRINKLE	beauty.removeWrinkle	0 ~ 100	无
	祛法令纹	BEAUTY_FACE_REMOVE_LAW_LINE	beauty.removeLawLine	0 ~ 100	无
	瘦颧骨	BEAUTY_FACE_THIN_CHEEKBONE	basicV7.cheekboneThin	0 ~ 100	无
单点美妆	口红	BEAUTY_MOUTH_LIPSTICK	beauty.faceFeatureLipsLut	0 ~ 100	口红图片在手机上的绝对路径 或者 相对于美颜模型文件目录的相对路径 示例： `/images/beauty/lips_fuguhong.png`
	腮红	BEAUTY_FACE_RED_CHEEK	beauty.faceFeatureRedCheek	0 ~ 100	示例： `/images/beauty/saihong_jianyue.png`
	立体	BEAUTY_FACE_SOFTLIGHT	beauty.faceFeatureSoftlight	0 ~ 100	示例： `/images/beauty/liti_ziran.png`
	染发 (V3.7.0)	BEAUTY_HAIR_COLOR_LUT	beauty.hairColorLut	0 ~ 100	示例： `/images/hair_color/red.png`
	眼影	BEAUTY_FACE_EYE_SHADOW	beauty.faceFeatureEyesMakeup.eyeShadow	0 ~ 100	示例： `/images/beauty/eyes_makeup_eye_shadow_0-albatross.png`
	眼线	BEAUTY_FACE_EYE_LINER	beauty.faceFeatureEyesMakeup.eyeLiner	0 ~ 100	示例： `/images/beauty/eyes_makeup_eye_liner_0.png`
	睫毛	BEAUTY_FACE_EYELASH	beauty.faceFeatureEyesMakeup.eyelash	0 ~ 100	示例： `/images/beauty/eyes_makeup_eyelash_0.png`
	眉毛	BEAUTY_FACE_EYEBROW	beauty.faceFeatureEyesMakeup.eyebrow	0 ~ 100	示例： `/images/beauty/eyes_makeup_eyebrow_0.png`
	美瞳	BEAUTY_FACE_EYEBALL	beauty.faceFeatureEyesMakeup.eyeball	0 ~ 100	示例： `/images/beauty/eyes_makeup_eyeball_0.png`
	双眼皮 (V3.8.0)	BEAUTY_FACE_MAKEUP_EYELIDS	beauty.faceFeatureEyesMakeup.eyelids	0 ~ 100	示例： `/images/beauty/eyes_makeup_eyelids_kaishan.png`

	卧蚕(V3.8.0)	BEAUTY_FACE_MAKEUP_EYEWOCAN	beauty.faceFeatureEyesMakeup.eyewocan	0 ~ 100	示例：`/images/beauty/eyes_makeup_eye_wocan_keai.png`
美体	一键瘦身	BODY_AUTOTHIN_BODY_STRENGTH	body.autothinBodyStrength	0 ~ 100	无
	长腿	BODY_LEG_STRETCH	body.legStretch	0 ~ 100	无
	瘦腿	BODY_SLIM_LEG_STRENGTH	body.slimLegStrength	0 ~ 100	无
	瘦腰	BODY_WAIST_STRENGTH	body.waistStrength	0 ~ 100	无
	瘦肩	BODY_THIN_SHOULDER_STRENGTH	body.thinShoulderStrength	0 ~ 100	无
	胸部调整	BODY_ENLARGE_CHEST_STRENGTH	body.enlargeChestStrength	-100 ~ 100	无
	小头	BODY_SLIM_HEAD_STRENGTH	body.slimHeadStrength	0 ~ 100	无
	瘦胳膊	BODY_SLIM_ARM_STRENGTH	body.slimArmStrength	-100 ~ 100	无

滤镜、美妆、动效、分割

	effectName		effectValue	resourcePath	extraInfo
	常量名	常量值	效果强度	资源路径	附加参数（键值对类型）
滤镜	EFFECT_LUT	lut	0~100	滤镜图片在手机上的绝对路径，示例：`/data/user/0/xxxxxx/files/xmagic/light_material/lut/aiqing_lf.png` 如果要取消滤镜，这里填null	无
轻美妆（V3.9.0）	EFFECT_LIGHT_MAKEUP	light.makeup	0~100	轻美妆素材在手机上的绝对路径。 如果要取消轻美妆，这里填null	[可选] makeupLutStrength：轻美妆素材中的滤镜强度，取值"0"到"100"
美妆	EFFECT_MAKEUP	makeup	0~100	美妆素材在手机上的绝对路径。 如果要取消美妆，这里填null	• [可选] `makeupLutStrength`：美妆素材中的滤镜强度，取值"0"到"100" • [可选] `mergeWithCurrentMotion`："true"或"false"，表示是否叠加在当前动效上。如果不填写此字段，则认为是 false
动效	EFFECT_MOTION	motion	无	动效素材在手机上的绝对路径，示例：`/data/user/0/xxxxxx/files/xmagic/light_material/motion/video_keaituya` 如果要取消动效，这里填null	• [可选] `mergeWithCurrentMotion`："true"或"false"，表示是否叠加在当前动效上。如果不填写此字段，则认为是 false
背景分割（普通）	EFFECT_SEGMENTATI	segmentation	无	背景分割素材在手机上的绝对路径 如果要取消分割，这里填null	• [可选] `mergeWithCurrentMotion`："true"或"false"，表示是否叠加在当前动效上。如果不填写此字段，则认为是 false

	ON				
背景分割（绿幕）	EFFECT_SEGMENTATION	segmentation	无	背景分割素材在手机上的绝对路径 如果要取消分割，这里填null	• **[必要]** `segType`：**"green_background"** • **[必要]** `bgType`：**自定义背景类型，"0"表示图片或pag，"1"表示视频** • **[可选]** `bgPath`：**自定义背景图片或视频路径** • **[可选]** `keyColor`：**绿幕颜色RGB，格式如"#00ff00"** • **[可选]** `mergeWithCurrentMotion`：**"true"或"false"，表示是否叠加在当前动效上。如果不填写此字段，则认为是 false** **注：** `bgPath` **和** `keyColor` **必须设置一项。**
背景分割（自定义）	EFFECT_SEGMENTATION	segmentation	无	背景分割素材在手机上的绝对路径 如果要取消分割，这里填null	• **[必要]** `segType`：**"custom_background"** • **[必要]** `bgType`：**自定义背景类型，"0"表示图片或者pag，"1"表示视频** • **[必要]** `bgPath`：**自定义背景图片或视频路径** • **[可选]** `mergeWithCurrentMotion`：**"true"或"false"，表示是否叠加在当前动效上。如果不填写此字段，则认为是 false**

美颜场景推荐参数

最近更新时间：2024-06-03 14:56:21

如下是项目 demo 中的一键美颜效果参数表，如果您想在应用中实现一键美颜效果，可以根据如下参数配置对应的美颜效果。

demo 默认效果：

功能类型	参数推荐
美白/自然	40
磨皮	40
清晰度	80
锐化	30
窄脸	5
瘦脸/自然	30
V脸	20
祛法令纹	30
大眼	20
亮眼	40
祛眼袋	50
瘦鼻	20
白牙	40

自然-1

功能类型	参数推荐
美白/自然	40
磨皮	40
清晰度	80
锐化	30
大眼	20
亮眼	40
祛眼袋	50
瘦鼻	20
白牙	40
窄脸	5
瘦脸/自然	30
祛法令纹	30
V脸	20

自然-2

功能类型	参数推荐
美白/自然	30
磨皮	30
对比度	-30
饱和度	-70
清晰度	15
锐化	25
大眼	20
亮眼	50
祛眼袋	5
瘦鼻	10
白牙	10
窄脸	10
瘦脸/自然	20
祛法令纹	30
V脸	10
口红/复古红	30
立体/自然	50

自然-3

功能类型	参数推荐
美白/粉白	40
磨皮	40
清晰度	70
锐化	30
大眼	10
亮眼	40
祛眼袋	50
瘦鼻	20
白牙	40
窄脸	20
祛法令纹	80
瘦颧骨	10
瘦脸/自然	30
V脸	20

口红/温柔粉	20
立体/自然	40

自然-4

功能类型	参数推荐
美白/自然	50
磨皮	20
饱和度	-60
大眼	20
瘦脸/自然	20
口红/蜜桃色	30
立体/自然	30

自然-5

功能类型	参数推荐
美白/自然	30
磨皮	40
红润	20
大眼	20
瘦脸/自然	30

女神-1

功能类型	参数推荐
美白/粉白	50
磨皮	50
清晰度	70
锐化	30
大眼	20
亮眼	40
祛眼袋	80
瘦鼻	20
白牙	40
窄脸	30
祛法令纹	80
瘦颧骨	10
瘦脸/自然	40
V脸	30

收下颌	10
口红/温柔粉	20
立体/自然	50

女神-2

功能类型	参数推荐
美白/粉白	60
磨皮	80
清晰度	70
锐化	30
大眼	35
亮眼	40
祛眼袋	100
瘦鼻	40
白牙	40
窄脸	40
祛法令纹	100
瘦颧骨	10
瘦脸/自然	60
V脸	40
收下颌	10
口红/温柔粉	20
立体/自然	50

女神-3

功能类型	参数推荐
美白/自然	60
磨皮	70
红润	35
大眼	40
瘦脸/自然	40
立体/自然	50
亮眼	30
瘦鼻	10

英俊

功能类型	参数推荐

美白/自然	20
磨皮	30
红润	5
瘦脸/英俊	60
脸型	20
额头	-20
立体/俊朗	60
瘦鼻	60

元气-1

功能类型	参数推荐
美白/自然	30
磨皮	80
红润	5
对比度	-40
饱和度	-30
清晰度	80
锐化	30
大眼	20
亮眼	40
祛眼袋	50
瘦鼻	20
白牙	40
窄脸	5
瘦脸/自然	30
祛法令纹	30
V脸	10

元气-2

功能类型	参数推荐
美白/自然	60
磨皮	65
红润	30
对比度	28
大眼	50
亮眼	60

瘦鼻	35
瘦脸/女神	50
窄脸	25
V脸	40
口红/复古红	30

短视频企业版迁移指引

最近更新时间：2024-07-11 16:19:42

目前，短视频企业版已经下线，其中美颜模块解耦升级成为腾讯特效 SDK。腾讯特效 SDK 美颜效果更加自然，产品功能更加强大，集成方式更加灵活。本文是短视频企业版升级为腾讯特效（美颜特效）的迁移指引。

注意事项

1. 修改 xmagic 模块中的 glide 库的版本号，与实际使用保持一致。
2. 修改 xmagic 模块中的最低版本号，与实际使用保持一致。

集成步骤

步骤一：解压 Demo 工程

1. 下载集成了腾讯特效 TE 的 UGSV Demo 工程。本 Demo 基于腾讯特效 SDK S1-04 套餐构建。
2. 替换资源。由于本 Demo 工程使用的 SDK 套餐未必与您实际的套餐一致，因此要将本 Demo 中的相关 SDK 文件替换为您实际使用的套餐的 SDK 文件。具体操作如下：
   - 删除 xmagic 模块中 libs 目录下的 `.aar` 文件，将 SDK 中 libs 目录下的 `.aar` 文件拷贝进 xmagic 模块中 libs 目录下。
   - 删除 xmagic 模块中 assets 目录下的所有文件，将 SDK 中的 `assets/` 目录下的全部资源拷贝到 xmagic 模块 `../src/main/assets` 目录下，如果SDK 包中的 MotionRes 文件夹内有资源，将此文件夹也拷贝到 `../src/main/assets` 目录下 。
   - 删除 xmagic 模块中jniLibs目录下的所有 .so 文件，在 SDK 包内的 jniLibs 中找到对应的 .so 文件（由于 SDK 中 jinLibs 文件夹下的 arm64-v8a 和 armeabi-v7a 的 .so 文件在压缩包中，所以需要先解压），拷贝到 xmagic 模块中的 `../src/main/jniLibs` 目录下。
3. 将 Demo 程中的 xmagic 模块引 到实际项 程中。

步骤二：SDK 版本升级

将 SDK 从 Enterprise 版本升级为 Professional 版本。

- **替换前：** `implementation 'com.tencent.liteav:LiteAVSDK_Enterprise:latest.release'`
- **替换后：** `implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'`

步骤三：设置美颜 License

1. 在项目中的 application 的 oncreate 方法中调用如下方法：

```
XMagicImpl.init(this);
XMagicImpl.checkAuth(null);
```

2. 在 XMagicImpl 类中替换成您申请的腾讯特效 License URL 和 Key。

步骤四：代码实现

以小视频录制界面（TCVideoRecordActivity.java）为例。

1. 在 `TCVideoRecordActivity.java` 类中添加如下变量代码：

```
private XMagicImpl mXMagic;
private int isPause = 0;//0 非暂停，1暂停，2暂停中 3.表示要销毁
```

2. 在 `TCVideoRecordActivity.java` 类 onCreate 方法后边添加如下代码：

```
TXUGCRecord instance = TXUGCRecord.getInstance(this);
instance.setVideoProcessListener(new TXUGCRecord.VideoCustomProcessListener() {
  @Override
  public int onTextureCustomProcess(int textureId, int width, int height) {
          if (isPause == 0 && mXMagic != null) {
                  return mXMagic.process(textureId, width, height);
          }
          return 0;
```

```
    }

    @Override
    public void onDetectFacePoints(float[] floats) {
    }

    @Override
    public void onTextureDestroyed() {
            if (Looper.getMainLooper() != Looper.myLooper()) {  //非主线程
                    if (isPause == 1) {
                            isPause = 2;
                            if (mXMagic != null) {
                                    mXMagic.onDestroy();
                            }
                            initXMagic();
                            isPause = 0;
                    } else if (isPause == 3) {
                            if (mXMagic != null) {
                                    mXMagic.onDestroy();
                            }
                    }
            }
    }
});
XMagicImpl.checkAuth((errorCode, msg) -> {
    if (errorCode == TELicenseCheck.ERROR_OK) {
            loadXmagicRes();
    } else {
            TXCLog.e("TAG", "鉴权失败，请检查鉴权url和key" + errorCode + " " + msg);
    }
});
```

3. 在 onStop 方法中添加如下代码：

```
isPause = 1;
if (mXMagic != null) {
    mXMagic.onPause();
}
```

4. 在 onDestroy 方法中添加如下代码：

```
isPause = 3;
XmagicPanelDataManager.getInstance().clearData();
```

5. 在 onActivityResult 方法最前边添加如下代码：

```
if (mXMagic != null) {
    mXMagic.onActivityResult(requestCode, resultCode, data);
}
```

6. 在此类的最后添加如下两个方法：

```
private void loadXmagicRes() {
    if (XMagicImpl.isLoadedRes) {
            XmagicResParser.parseRes(getApplicationContext());
            initXMagic();
            return;
    }
```

```
        new Thread(() -> {
                XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").getAbsolutePath());
                XmagicResParser.copyRes(getApplicationContext());
                XmagicResParser.parseRes(getApplicationContext());
                XMagicImpl.isLoadedRes = true;
                new Handler(Looper.getMainLooper()).post(() -> {
                        initXMagic();
                });
        }).start();

}
/**
* 初始化美颜SDK
*/
private void initXMagic() {
        if (mXMagic == null) {
                mXMagic = new XMagicImpl(this, mUGCKitVideoRecord.getBeautyPanel());
        }else {
                mXMagic.onResume();
        }
}
```

步骤五：对其他类的修改

1. 将 AbsVideoRecordUI 类的 mBeautyPanel 类型修改为 RelativeLayout 类型，getBeautyPanel() 方法返回类型也修改为 RelativeLayout，同时修改对应 XML 中的配置，注掉报错的代码。
2. 注释掉 UGCKitVideoRecord 类中报错的代码。
3. 修改 ScrollFilterView 类中的代码，删除 mBeautyPanel 变量，注释掉报错的代码。

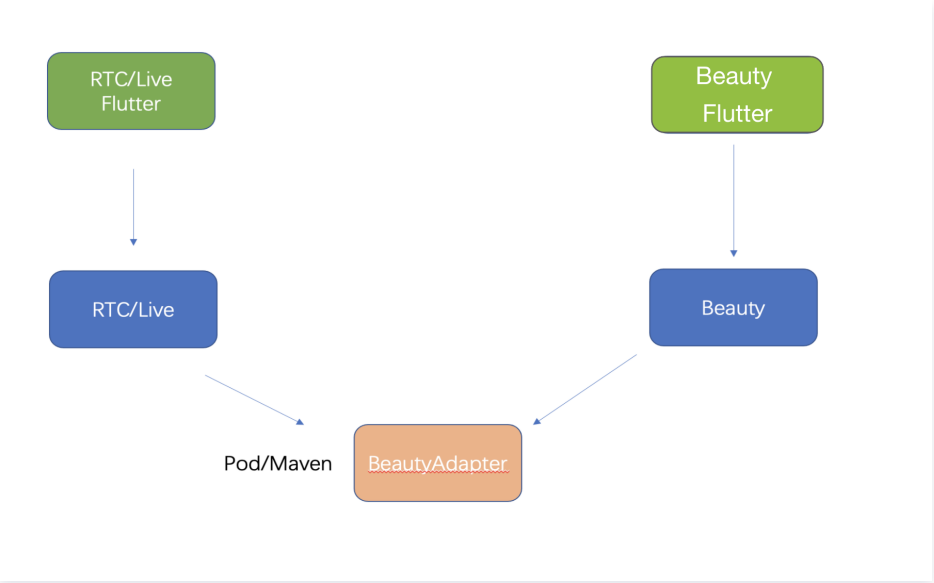
步骤六：删除对 beautysettingkit 模块的依赖

在 ugckit 模块的 build.gradle 文件中删除对 beautysettingkit 模块的依赖，编译项目将报错的代码注释掉即可。

第三方推流接入美颜（Flutter）

最近更新时间：2024-08-06 17:53:41

由于 Flutter 端的 GL 环境与原生端环境进行了隔离，所以 Flutter 中接入美颜时无法直接建立绑定关系，需要在原生端进行关系的绑定，如下图所示：



实现方式总体流程

1. 美颜侧抽象一层接口，并在美颜侧实现了接口。
2. 在应用启动时将此接口注册到三方推流端，这样三方推流端就可以通过此接口进行创建、使用、销毁美颜实例。
3. 第三方推流端再将创建和销毁美颜的能力暴露给自己的 Flutter 端供客户使用。
4. 美颜属性设置可通过美颜提供的 Flutter SDK 能力进行处理。

以 TRTC 为例

美颜侧定义的接口：

```
public interface ITXCustomBeautyProcesserFactory {

    /**
     * 创建美颜实例
     * @return
     */
    ITXCustomBeautyProcesser createCustomBeautyProcesser();

    /**
     * 销毁美颜实例（需要在GL线程调用）
     */
    void destroyCustomBeautyProcesser();
}
public interface ITXCustomBeautyProcesser {

    //获取美颜支持的视频帧的像素格式。美颜支持的是：OpenGL 2D 纹理。
    TXCustomBeautyPixelFormat getSupportedPixelFormat();
    //获取美颜支持的视频数据包装格式。美颜支持的是：V2TXLiveBufferTypeTexture  直接操作纹理 ID，性能最好，画质损失最少。
    TXCustomBeautyBufferType getSupportedBufferType();
    //在GL线程调用（srcFrame中需要包含RGBA纹理，以及width，height），美颜处理之后会将处理后的纹理对象放置在dstFrame中的texture.textureId中。
    void onProcessVideoFrame(TXCustomBeautyVideoFrame srcFrame, TXCustomBeautyVideoFrame dstFrame);
```

```
}
```

1. TRTC提供一个注册的方法，在应用启动时，需将美颜侧 ITXCustomBeautyProcesserFactory 接口的实现类 `com.tencent.effect.tencent_effect_flutter.XmagicProcesserFactory` 注册进 TRTC 中（在原生端进行）。

```
package com.tencent.effect.tencent_effect_flutter_example;

import android.os.Bundle;

import androidx.annotation.Nullable;

import com.tencent.effect.tencent_effect_flutter.XmagicProcesserFactory;
import com.tencent.live.TXLivePluginManager;   kevinxlhua, 2022/5/9, 3:38 下午 • 添加对 直播flutter版本的依赖，由于现在直

import io.flutter.embedding.android.FlutterActivity;
import com.tencent.trtcplugin.TRTCCloudPlugin;

public class MainActivity extends FlutterActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TXLivePluginManager.register(new XmagicProcesserFactory());
        TRTCCloudPlugin.register(new XmagicProcesserFactory());
    }
}
```

2. 在 `Flutter` 层，提供 `Future<V2TXLiveCode> enableCustomVideoProcess(bool enable)` 接口进行开启或关闭自定义美颜接口。
3. TRTC原生端实现开关美颜方法。

```
 */
public void enableCustomVideoProcess(MethodCall call, MethodChannel.Result result) {
    boolean enable = CommonUtil.getParam(call, result,  param: "enable");
    ITXCustomBeautyProcesserFactory processerFactory = TRTCCloudPlugin.getBeautyProcesserFactory();
    mCustomBeautyProcesser = processerFactory.createCustomBeautyProcesser();
    TXCustomBeautyBufferType bufferType = mCustomBeautyProcesser.getSupportedBufferType();
    TXCustomBeautyPixelFormat pixelFormat = mCustomBeautyProcesser.getSupportedPixelFormat();
    if(enable) {
        ProcessVideoFrame processVideo =  new ProcessVideoFrame(mCustomBeautyProcesser);
        int ret = trtcCloud.setLocalVideoProcessListener(pixelFormat.convertTRTCPixelFormat(), bufferType.convertTRTCBufferType(), processVideo;
        result.success(ret);
    } else {
        int ret = trtcCloud.setLocalVideoProcessListener(pixelFormat.convertTRTCPixelFormat(), bufferType.convertTRTCBufferType(),  trtcVideoFrameListener: null);
        mCustomBeautyProcesser = null;   clavie, 2022/5/23, 7:49 下午 • 美颜预处理
        result.success(ret);
    }
}
```

当设置为 null 是则会回调上次设置的 processVideo 的 onGlContextDestroy 方法

```java
package com.tencent.trtcplugin.listener;
import ...

public class ProcessVideoFrame implements TRTCCloudListener.TRTCVideoFrameListener {
    private ITXCustomBeautyProcesser    mCustomBeautyProcesser;

    public ProcessVideoFrame(ITXCustomBeautyProcesser processer) {   clavie, 2022/5/23, 7:49 下午 • 美颜预处理
        mCustomBeautyProcesser = processer;
    }
    /** 自定义视频处理回调 ...*/
    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame  srcFrame,
                                   TRTCCloudDef.TRTCVideoFrame  dstFrame) {
        TXCustomBeautyVideoFrame srcThirdFrame = new TXCustomBeautyVideoFrame(srcFrame);
        TXCustomBeautyVideoFrame dstThirdFrame = new TXCustomBeautyVideoFrame(dstFrame);
        mCustomBeautyProcesser.onProcessVideoFrame(srcThirdFrame, dstThirdFrame);
        if (dstThirdFrame.texture != null) {
            dstFrame.texture.textureId = dstThirdFrame.texture.textureId;
        }
        dstFrame.data = dstThirdFrame.data;
        dstFrame.buffer = dstThirdFrame.buffer;
        dstFrame.width = dstThirdFrame.width;
        dstFrame.height = dstThirdFrame.height;
        dstFrame.rotation = dstThirdFrame.rotation;
        return 0;
    }

    /** SDK 内部的 OpenGL 环境的创建通知 */
    public void onGLContextCreated() {}

    /**
     * SDK 内部的 OpenGL 环境的销毁通知
     */
    public void onGLContextDestory() {
        ITXCustomBeautyProcesserFactory processerFactory = TRTCCloudPlugin.getBeautyProcesserFactory();
        if (processerFactory != null) {
            processerFactory.destroyCustomBeautyProcesser();
        }
        mCustomBeautyProcesser = null;
    }
}
```

附录

美颜提供的抽象层依赖

```
///
implementation 'com.tencent.liteav:custom-video-processor:latest.release'
```

小程序美颜特效实践

最近更新时间：2024-08-20 14:07:41

准备工作

- 小程序开发入门请参见 微信小程序文档 。
- 请阅读 Web 美颜特效 SDK 接入指南，熟悉 SDK 基本用法。

开始使用

步骤1：小程序后台配置域名白名单

SDK 内部会请求后台进行鉴权和资源加载，因此小程序创建完后，需要在小程序后台配置域名白名单。

1. 打开 小程序后台，进入**开发 > 开发管理 > 开发设置 > 服务器域名**。
2. 单击**修改**，配置以下域名并保存。
   - 请求域名：

```
https://webar.qcloud.com;
https://webar-static.tencent-cloud.com;
https://aegis.qq.com;
以及鉴权签名接口（get-ar-sign）的地址
```

   - downloadFile 域名：

```
https://webar-static.tencent-cloud.com
```

步骤2：安装并构建 npm

小程序 npm 相关请参见 小程序使用 npm 。

1. 安装：

```
npm install tencentcloud-webar
```

2. 构建：
   打开小程序开发者工具，顶部菜单选择**工具 > 构建 npm**。
3. 在 app.json 中配置 workers 路径：

```
"workers": "miniprogram_npm/tencentcloud-webar/worker"
```

步骤3：引入文件

```
// 0.3.0之前版本引用方式（1个文件）
// import "../../miniprogram_npm/tencentcloud-webar/lib.js";
// 0.3.0及之后版本引用方式（2个文件 + 按需初始化3d模块）
import '../../miniprogram_npm/tencentcloud-webar/lib.js';
import '../../miniprogram_npm/tencentcloud-webar/core.js';
// 按需初始化3d插件，如果不需要3d则可以不引用
import '../../miniprogram_npm/tencentcloud-webar/lib-3d.js';
import { plugin3d } from '../../miniprogram_npm/tencentcloud-webar/plugin-3d'
// 导入 ArSdk
import { ArSdk } from "../../miniprogram_npm/tencentcloud-webar/index.js";
```

> ⚠ **注意：**
> - 小程序有单文件不超过 500kb 的限制，因此 SDK 分为两个 js 文件提供。

- 0.3.0版本之后，对 SDK 进行了进一步的拆分，新增3D支持，针对3D模块提供按需加载方式，导入前请确认当前使用的 SDK 版本信息，选择对应的导入方式。

步骤4：初始化 SDK

⚠ 注意：

- 小程序初始化 SDK 前须在控制台配置小程序 APPID，请参见 快速上手 。
- 需在页面中插入 camera 标签来打开相机，然后设置 camera 参数，参数配置详情请参见 接入指南 。
- 小程序不支持 getOutput，需要自行传入一个在屏的 webgl canvas，SDK 直接输出画面到此 canvas 上。

示例代码如下：

```
// wxml
//打开相机，通过position使相机不展示
<camera
  device-position="{{'front'}}"
  frame-size="large" flash="off" resolution="medium"
  style="width: 750rpx; height: 134rpx;position:absolute;top:-9999px;"
/>
//sdk 将处理完的画面实时输出到此 canvas 上
<canvas
  type="webgl"
  canvas-id="main-canvas"
  id="main-canvas"
  style="width: 750rpx; height: 1334rpx;">
</canvas>
//拍照将 ImageData 对象绘制到此 canvas 上
<canvas
  type="2d"
  canvas-id="photo-canvas"
  id="photo-canvas"
  style="position:absolute;width:720px;height:1280px;top:-9999px;left:-9999px;">
</canvas>
// js
/** ----- 鉴权配置 ----- */

/**
 * 腾讯云账号 APPID
 *
 * 进入[腾讯云账号中心](https://console.cloud.tencent.com/developer) 即可查看 APPID
 */
const APPID = ''; // 此处请填写您自己的参数

/**
 * Web LicenseKey
 *
 * 登录音视频终端 SDK 控制台的[Web License 管理](https://console.cloud.tencent.com/vcube/web)，创建项目即可获得
LicenseKey
 */
const LICENSE_KEY = ''; // 此处请填写您自己的参数

/**
 * 计算签名用的密钥 Token
 *
 * 注意：此处仅用于 DEMO 调试，正式环境中请将 Token 保管在服务端，签名方法迁移到服务端实现，通过接口提供，前端调用拉取签名，
参考
 * [签名方法](https://cloud.tencent.com/document/product/616/71370#.E7.AD.BE.E5.90.8D.E6.96.B9.E6.B3.95)
 */
```

```
const token = ''; // 此处请填写您自己的参数

Component({
    data: {
        makeupList: [],
        stickerList: [],
        filterList: [],
        recording: false
    },
    methods: {
        async getCanvasNode(id) {
            return new Promise((resolve) => {
                this.createSelectorQuery()
                    .select(`#${id}`)
                    .node()
                    .exec((res) => {
                        const canvasNode = res[0].node;
                        resolve(canvasNode);
                    });
            });
        },
        getSignature() {
            const timestamp = Math.round(new Date().getTime() / 1000);
            const signature = sha256(timestamp + token + APPID + timestamp).toUpperCase();
            return { signature, timestamp };
        },
        // 初始化相机类型
        async initSdkCamera() {
            // 获取在屏的 canvas，sdk 会将处理完的画面实时输出到 canvas 上
            const outputCanvas = await this.getCanvasNode("main-canvas");
            // 获取鉴权参数
            const auth = {
                licenseKey: LICENSE_KEY,
                appId: APP_ID,
                authFunc: this.getSignature
            };
            // 构造SDK初始化参数
            const config = {
                auth,
                camera: {
                    width:720,
                    height:1280,
                },
                output: outputCanvas,
                // 初始美颜效果（可选）
                beautify: {
                    whiten: 0.1, // 美白 0-1
                    dermabrasion: 0.3, // 磨皮 0-1
                    lift: 0, // 瘦脸 0-1
                    shave: 0, // 削脸 0-1
                    eye: 0.2, // 大眼 0-1
                    chin: 0, // 下巴 0-1
                }
            };
            const ar = new ArSdk(config);
            // created回调里可以开始获取内置特效与滤镜列表
            ar.on('created', () => {
                // 获取内置美妆、贴纸列表
                ar.getEffectList({
                    Type: 'Preset'
                }).then((res) => {
```

```
                const list = res.map(item => ({
                    name: item.Name,
                    id: item.EffectId,
                    cover: item.CoverUrl,
                    url: item.Url,
                    label: item.Label,
                    type: item.PresetType,
                }));
                const makeupList = list.filter(item=>item.label.indexOf('美妆')>=0)
                const stickerList = list.filter(item=>item.label.indexOf('贴纸')>=0)
                // 渲染特效列表
                this.setData({
                    makeupList,
                    stickerList
                });
            }).catch((e) => {
                console.log(e);
            });
            // 内置滤镜
            ar.getCommonFilter().then((res) => {
                const list = res.map(item => ({
                    name: item.Name,
                    id: item.EffectId,
                    cover: item.CoverUrl,
                    url: item.Url,
                    label: item.Label,
                    type: item.PresetType,
                }));
                // 渲染滤镜列表
                this.setData({
                    filterList: list
                });
            }).catch((e) => {
                console.log(e);
            });
        });
        // ready 回调中可以设置美颜、滤镜、特效效果
        ar.on('ready', (e) => {
            this._sdkReady = true
        });

        ar.on('error', (e) => {
            console.log(e);
        });

        this.ar = ar
    },
    // 改变美颜参数，需要确保sdk ready
    onChangeBeauty(val){
        if(!this._sdkReady) return
        // 可以通过setBeautify设置美颜效果，支持6种属性，详见SDK接入指南
        this.ar.setBeautify({
            dermabrasion: val.dermabrasion, // 磨皮 0-1
        });
    },
    // 改变美妆，需要确保sdk ready
    onChangeMakeup(id, intensity){
        if(!this._sdkReady) return
        // 使用setEffect设置特效，setEffect的输入参数支持三种格式，详见SDK接入指南
        this.ar.setEffect([{id, intensity}]);
    },
```

```
        // 改变贴纸，需要确保sdk ready
        onChangeSticker(id, intensity){
            if(!this._sdkReady) return
            // 使用setEffect设置特效，setEffect的输入参数支持三种格式，详见SDK接入指南
            this.ar.setEffect([{id, intensity}]);
        },
        // 改变滤镜，需要确保sdk ready
        onChangeFilter(id, intensity){
            if(!this._sdkReady) return
            // 使用setFilter设置滤镜，第二个参数表示滤镜强度（范围0-1）
            this.ar.setFilter(id, 1);
        }
    }
})
```

步骤5：拍照和录像功能实现

示例代码如下：

拍照

SDK 会返回包含宽高和 buffer 数据的对象，用户可以通过自己页面内预设的 2d canvas（上述代码中id为photo-canvas）绘制此数据并导出为图片文件。

```
async takePhoto() {
    const {uint8ArrayData, width, height} = this.ar.takePhoto(); // takePhoto 方法返回当前画面的 buffer
数据
    const photoCanvasNode = await this.getCanvasNode('photo-canvas');
    photoCanvasNode.width = parseInt(width);
    photoCanvasNode.height = parseInt(height);
    const ctx = photoCanvasNode.getContext('2d');
    // 用 sdk 返回的数据创建 ImageData 对象
    const imageData = photoCanvasNode.createImageData(uint8ArrayData, width, height);
    // 将 ImageData 对象绘制到 canvas 上
    ctx.putImageData(imageData,0,0,0,0,width,height);
    // 将 canvas 保存为本地图片
    wx.canvasToTempFilePath({
        canvas: photoCanvasNode,
        x: 0,
        y: 0,
        width: width,
        height: height,
        destWidth: width,
        destHeight: height,
        success: (res) => {
            // 保存照片到本地
            wx.saveImageToPhotosAlbum({
                filePath: res.tempFilePath
            });
        }
    })
}
```

录像

```
Component({
```

```
    methods: {
        // 开始录像
        startRecord() {
            this.setData({
                recording: true
            });
            this.ar.startRecord()
        }
        // 结束录像
        async stopRecord() {
            const res = await this.ar.stopRecord();
            // 保存录像到本地
            wx.saveVideoToPhotosAlbum({
                filePath: res.tempFilePath
            });
            this.setData({
                recording: false
            });
        }
    }
})
```

当小程序切换后台或检测到手机锁屏时，需要调用 stopRecord 停止录像，再次回到页面时重新开启SDK即可。

```
onShow() {
    this.ar && this.ar.start();
},
onHide() {
    this.ar && this.ar.stop();
},
async onUnload() {
    try {
        this.ar && this.ar.stop();
        if (this.data.recording) {
            await this.ar.stopRecord({
                destroy: true,
            });
        }
    } catch (e) {
    }
    this.ar && this.ar.destroy();
}
```

代码示例

您可以下载 示例代码 解压后查看 `ar-miniprogram` 代码目录。