腾讯特效 SDK

功能实践

产品文档

腾讯云

文档目录

功能实践
SDK 包瘦身
iOS

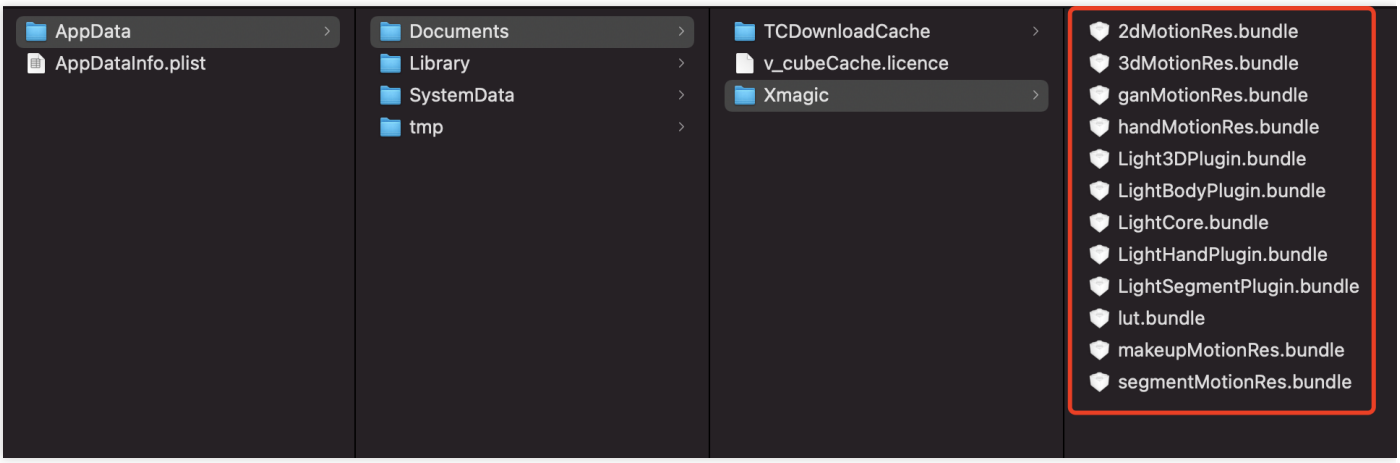
最近更新时间：2022-07-22 12:04:37

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:asset
sDict];
```

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] bu
ndlePath], model.key];
[arrayModels addObject:model];
} else {
if(_useNetResource && _filePath != nil){ //使用网络资源时
NSString *DirPath = [_filePath stringByAppendingPathComponent:@"2dMotionRes.bun
dle/"]; //获取美颜资源的绝对路径
model.icon = [NSString stringWithFormat:@"%@/%@/template.png", DirPath, model.k
ey];
}else{
model.icon = [NSString stringWithFormat:@"%@/%@/template.png", [[NSBundle mainB
undle] 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
propertyType	lut	lut
propertyName	lut.bundle/n_baixi.png	/var/mobile/Containers/Data/Application/25C7D(73F6-4F1B-AEB6- 5EE03A221D18/Documents/Xmagic/lut.bundle/n_bai
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:@"Mot
ionRes" 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:motionR
ootPath 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

字段	使用本地资源时传入的参数	使用网络资源时传
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/C 73F6-4F1B-AEB 5EE03A221D18/I`
extraInfo	nil	nil

Android

最近更新时间：2022-07-20 15:11:09

为了减少包体大小，您可将 SDK 所需的 assets 资源、so 库、以及动效资源 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 。

我们建议您复用 Demo 的下载逻辑，当然，也可以使用您已有的下载服务。

如果复用 Demo 的下载逻辑，请注意一点： Demo 中默认是开启断点续传功能的，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
```

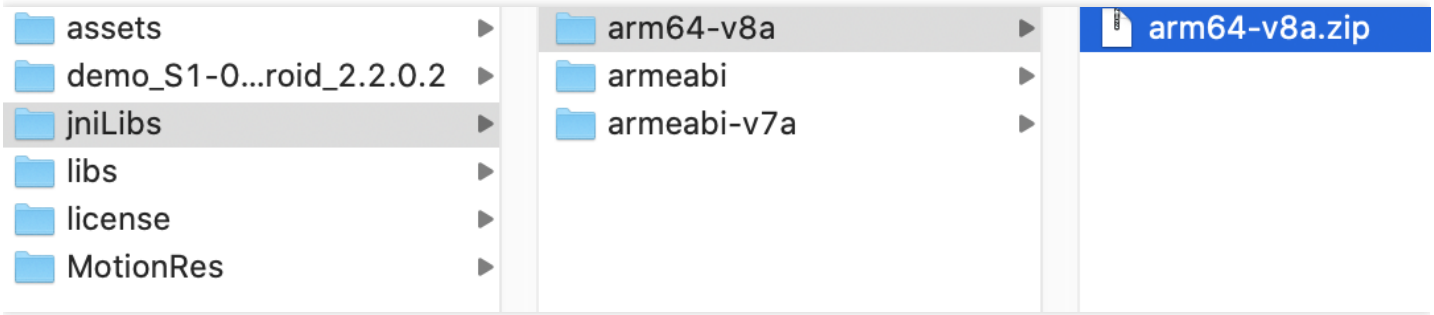
例如：

```
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。

动态下载 so

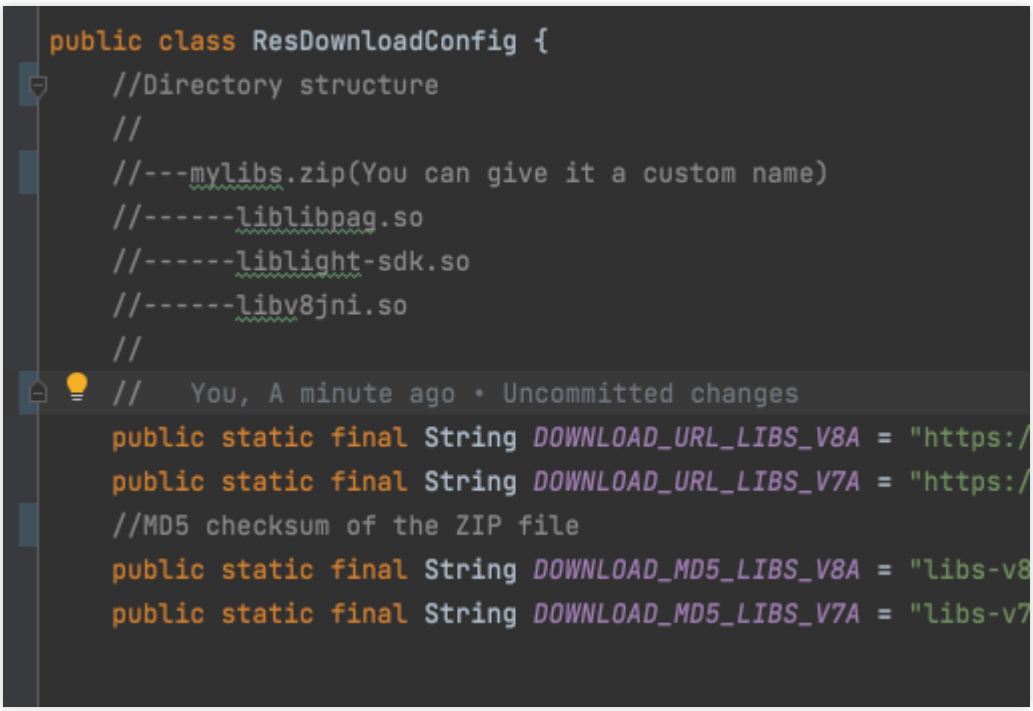
so 压缩包位于 `jniLibs/arm64-v8a` 和 `jniLibs/armeabi-v7a` ，如下图所示：



- 如果您想复用 Demo 中的下载服务
- 如果您想自己做下载服务

1. 计算出这两个 ZIP 包的 MD5 值，Mac 上可以直接在命令行使用 `md5 文件路径/文件名` 算出 MD5，或者使用其他工具软件计算出来。
2. 将压缩包上传到您的服务器，得到下载 URL。

3. 更新 Demo 工程里的 ResDownloadConfig 里的这几个常量值：



4. 调用 `ResDownloadUtil.checkOrDownloadFiles` 即可启动下载。

注意：

- 当 SDK 版本更新时，对应的 so 可能会发生变化，您需要重新下载这些 so。建议参考 Demo 中的方式，利用 MD5 进行校验。
- 无论是您自行下载 so，还是复用 Demo 中的下载服务，在调用 SDK 的 auth 接口之前，请先检查 so 是否已下载好，ResDownloadUtil 提供了如下方法进行检查。如果已下载好，则将路径设置给 SDK，如下所示：

```
String validLibsDirectory = ResDownloadUtil.getValidLibsDirectory(LaunchActivity.
this,
isCpuV8a() ? ResDownloadConfig.DOWNLOAD_MD5_LIBS_V8A : ResDownloadConfig.DOWNLOAD
_MD5_LIBS_V7A);
if (validLibsDirectory == null) {
Toast.makeText(LaunchActivity.this,"libs没有下载好，请先下载",Toast.LENGTH_LONG).sho
w();
return;
}
XmagicApi.setLibPathAndLoad(validLibsDirectory);
auth();
```

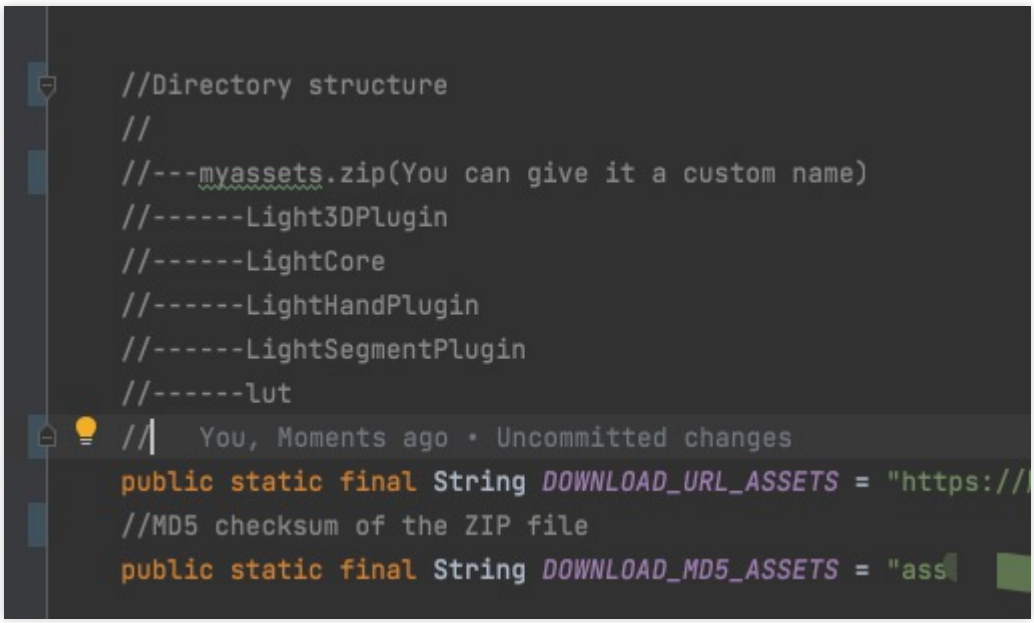
动态下载 assets 资源

如果要动态下载 assets 资源，具体操作如下：

1. 在本地工程的 assets 里进行配置：
   - **2.4.0及更高版本**：本地 assets 目录不需要保留任何文件。
   - **2.4.0以下版本**：需保留 License 文件和这4个 JSON 配置文件：`brand_name.json`、`device_config.json`、`phone_info.json`、`score_phone.json`。
2. 找到 SDK 中已经打包好的 `download_assets.zip`。

3. 与 上文 so 文件 的处理方法一样，计算这个 ZIP 包的 MD5，将它上传到服务器得到下载地址。
   - 如果您想复用 Demo 中的下载服务
   - 如果您想要自己做下载服务

i. 更新下图中的下载地址和 MD5。



ii. 调用 `ResDownloadUtil.checkOrDownloadFiles` 启动下载，调用 `ResDownloadUtil.getValidAssetsDirectory` 得到下载好的 assets 的 path，具体用法请参考 `LaunchActivity.java` 。

注意：

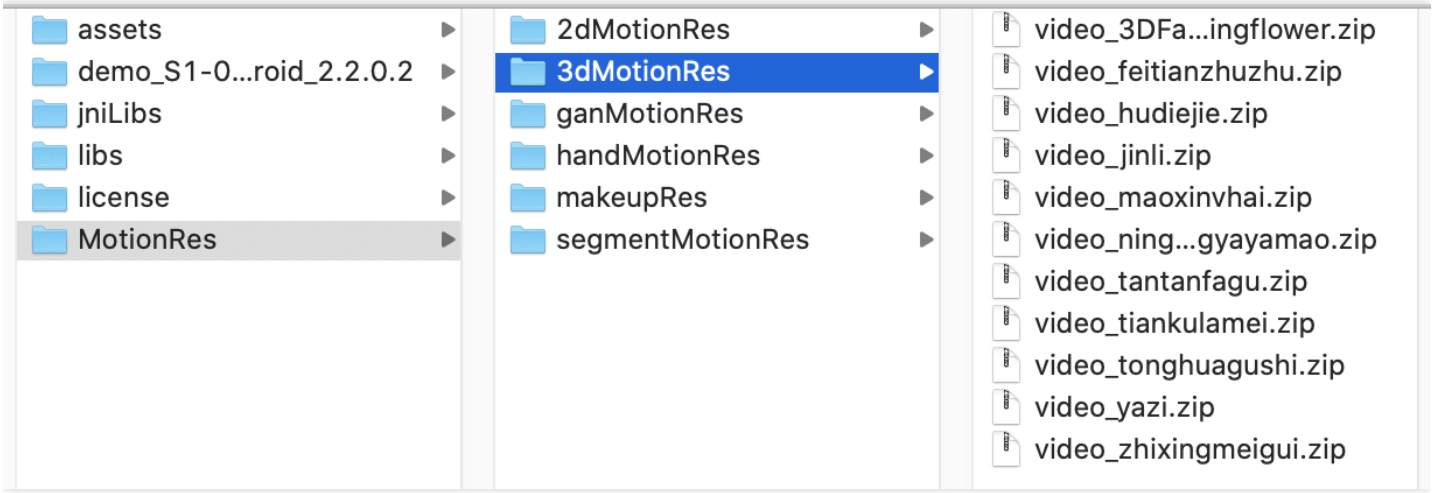
- 当 SDK 版本更新时，对应的 assets 可能会发生变化，为确保兼容性，您需要重新下载这些 assets。建议参考 Demo 中的方式，利用 MD5 进行校验 。
- 无论是您自行下载 assets，还是复用 Demo 中的下载服务，在进入拍摄之前，请先检查 assets 是否已下载好，ResDownloadUtil 提供了如下方法进行检查。如果已下载好，则将路径设置给 XmagicResParser。具体用法请参考 `LaunchActivity.java` 。

```
String validAssetsDirectory = ResDownloadUtil.getValidAssetsDirectory(LaunchActiv
ity.this,ResDownloadConfig.DOWNLOAD_MD5_ASSETS);
if (validAssetsDirectory == null) {
Toast.makeText(LaunchActivity.this,"assets没有下载好，请先下载",Toast.LENGTH_LONG).s
how();
return;
}
XmagicResParser.setResPath(validAssetsDirectory);
startActivity(intent);
```

动效资源 MotionRes 下载

部分基础套餐没有动效资源，可以忽略这一小节。

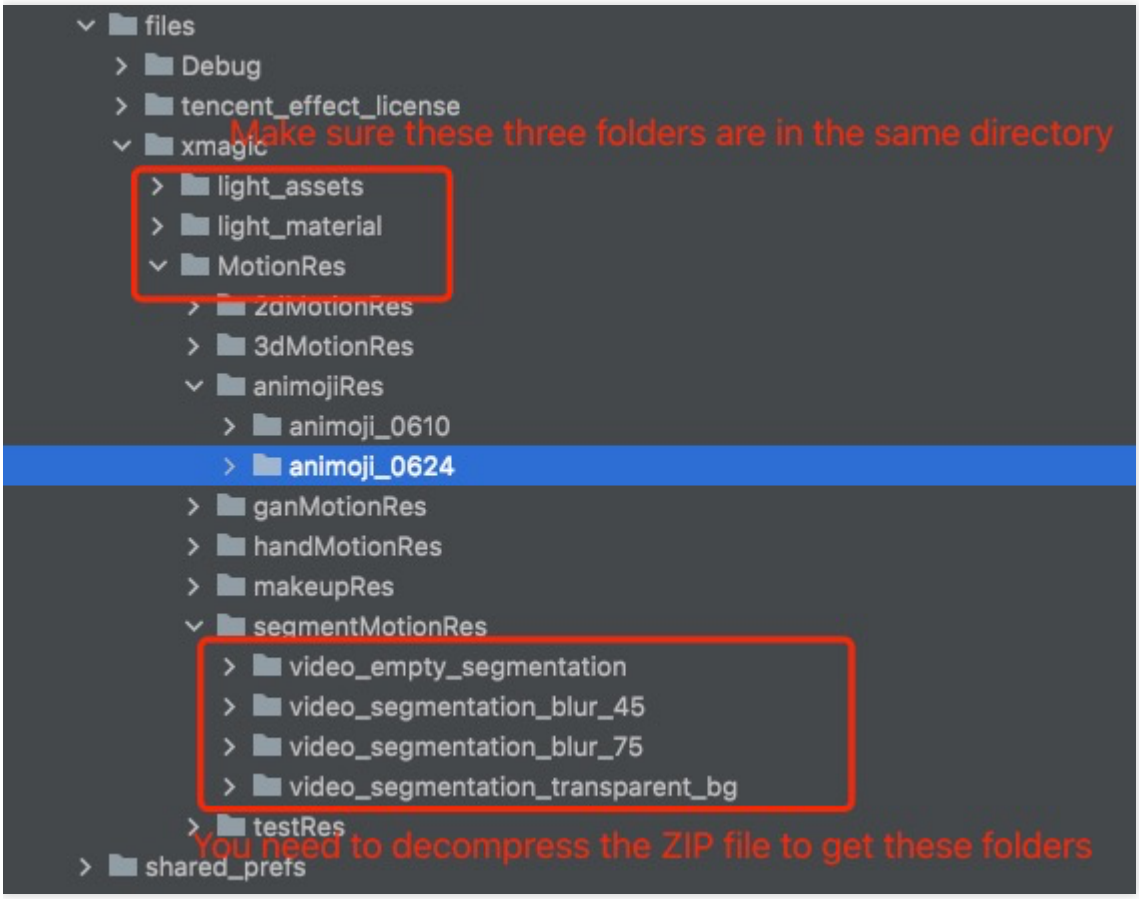
动效分为六大类，每一类中有若干个 ZIP 包，每个 ZIP 包是一种动效。根据您购买的套餐类型，这里的文件内容会有所不同。



动效资源可以按需下载，例如等用户进入到相关功能页面，或者单击了某个动效的图标之后，再下载。

您需要将这些 ZIP 包都上传到服务器，得到每一个 ZIP 包的下载地址。

> 注意：
> **动效资源下载后的目录 MotionRes 需要和上一小节的 light_assets 以及 light_material 的在同一个层级**。
> 并且每一个特效都需要解压，而不能直接放一个 ZIP 包，如下图所示：



MotionRes 的下载可参考 ResDownloadUtil.checkOrDownloadMotions 方法，建议按需一个个下载。

如果您想复用 Demo 中的下载服务，那么将 ResDownloadConfig 中的 MOTION_RES_DOWNLOAD_PREFIX 常量值替换为您自己的下载 URL 前缀即可。

美颜参数说明

Android & iOS

最近更新时间：2024-03-21 16:42:46

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

美颜、美体

		effectName	
		常量名（Android）	常量名（iOS）
美颜	美白	XmagicConstant.EffectName.BEAUTY_WHITEN	BEAUTY_WHITEN
	美白2	XmagicConstant.EffectName.BEAUTY_WHITEN_2	BEAUTY_WHITEN
	美白3	XmagicConstant.EffectName.BEAUTY_WHITEN_3	BEAUTY_WHITEN
	磨皮	XmagicConstant.EffectName.BEAUTY_SMOOTH	BEAUTY_SMOOT
	红润	XmagicConstant.EffectName.BEAUTY_ROSY	BEAUTY_ROSY
画面调整	对比度	XmagicConstant.EffectName.BEAUTY_CONTRAST	BEAUTY_CONTR
	饱和度	XmagicConstant.EffectName.BEAUTY_SATURATION	BEAUTY_SATUR
	清晰度	XmagicConstant.EffectName.BEAUTY_CLEAR	BEAUTY_CLEAR
	锐化	XmagicConstant.EffectName.BEAUTY_SHARPEN	BEAUTY_SHAPE
	弱光降噪 **(V3.6.0)**	XmagicConstant.EffectName.BEAUTY_IMAGE_DENOISE	BEAUTY_IMAGE_
	色温	XmagicConstant.EffectName.BEAUTY_IMAGE_WARMTH	BEAUTY_IMAGE_
	色调	XmagicConstant.EffectName.BEAUTY_IMAGE_TINT	BEAUTY_IMAGE_

高级美型	大眼	XmagicConstant.EffectName.BEAUTY_ENLARGE_EYE	BEAUTY_ENLAR
	亮眼	XmagicConstant.EffectName.BEAUTY_EYE_LIGHTEN	BEAUTY_EYE_LI
	眼距	XmagicConstant.EffectName.BEAUTY_EYE_DISTANCE	BEAUTY_EYE_DI
	眼角	XmagicConstant.EffectName.BEAUTY_EYE_ANGLE	BEAUTY_EYE_AN
	眼宽	XmagicConstant.EffectName.BEAUTY_EYE_WIDTH	BEAUTY_EYE_W
	眼高	XmagicConstant.EffectName.BEAUTY_EYE_HEIGHT	BEAUTY_EYE_HE
	祛眼袋	XmagicConstant.EffectName.BEAUTY_FACE_REMOVE_EYE_BAGS	BEAUTY_FACE_R
	眉毛角度	XmagicConstant.EffectName.BEAUTY_EYEBROW_ANGLE	BEAUTY_EYEBR
	眉毛距离	XmagicConstant.EffectName.BEAUTY_EYEBROW_DISTANCE	BEAUTY_EYEBR
	眉毛高度	XmagicConstant.EffectName.BEAUTY_EYEBROW_HEIGHT	BEAUTY_EYEBR
	眉毛长度	XmagicConstant.EffectName.BEAUTY_EYEBROW_LENGTH	BEAUTY_EYEBR
	眉毛粗细	XmagicConstant.EffectName.BEAUTY_EYEBROW_THICKNESS	BEAUTY_EYEBR
	眉峰	XmagicConstant.EffectName.BEAUTY_EYEBROW_RIDGE	BEAUTY_EYEBR
	瘦鼻	XmagicConstant.EffectName.BEAUTY_NOSE_THIN	BEAUTY_NOSE_T
	鼻翼	XmagicConstant.EffectName.BEAUTY_NOSE_WING	BEAUTY_NOSE_W
	鼻子位置	XmagicConstant.EffectName.BEAUTY_NOSE_HEIGHT	BEAUTY_NOSE_H
	鼻梁	XmagicConstant.EffectName.BEAUTY_NOSE_BRIDGE_WIDTH	BEAUTY_NOSE_B
	山根	XmagicConstant.EffectName.BEAUTY_NASION	BEAUTY_NASION
	白牙	XmagicConstant.EffectName.BEAUTY_TOOTH_WHITEN	BEAUTY_TOOTH
	嘴型	XmagicConstant.EffectName.BEAUTY_MOUTH_SIZE	BEAUTY_MOUTH
	嘴唇厚度	XmagicConstant.EffectName.BEAUTY_MOUTH_HEIGHT	BEAUTY_MOUTH

	嘴唇宽度	XmagicConstant.EffectName.BEAUTY_MOUTH_WIDTH	BEAUTY_MOUTH
	嘴唇位置	XmagicConstant.EffectName.BEAUTY_MOUTH_POSITION	BEAUTY_MOUTH
	微笑唇	XmagicConstant.EffectName.BEAUTY_SMILE_FACE	BEAUTY_SMILE_
	窄脸	XmagicConstant.EffectName.BEAUTY_FACE_THIN	BEAUTY_FACE_T
	瘦脸-自然	XmagicConstant.EffectName.BEAUTY_FACE_NATURE	BEAUTY_FACE_N
	瘦脸-女神	XmagicConstant.EffectName.BEAUTY_FACE_GODNESS	BEAUTY_FACE_G
	瘦脸-英俊	XmagicConstant.EffectName.BEAUTY_FACE_MALE_GOD	BEAUTY_FACE_M
	V脸	XmagicConstant.EffectName.BEAUTY_FACE_V	BEAUTY_FACE_V
	收下颌	XmagicConstant.EffectName.BEAUTY_FACE_JAW	BEAUTY_FACE_J
	短脸	XmagicConstant.EffectName.BEAUTY_FACE_SHORT	BEAUTY_FACE_S
	脸型	XmagicConstant.EffectName.BEAUTY_FACE_BASIC	BEAUTY_FACE_B
	下巴	XmagicConstant.EffectName.BEAUTY_FACE_THIN_CHIN	BEAUTY_FACE_T
	额头	XmagicConstant.EffectName.BEAUTY_FACE_FOREHEAD	BEAUTY_FACE_F
	祛皱	XmagicConstant.EffectName.BEAUTY_FACE_REMOVE_WRINKLE	BEAUTY_FACE_F
	祛法令纹	XmagicConstant.EffectName.BEAUTY_FACE_REMOVE_LAW_LINE	BEAUTY_FACE_F
	瘦颧骨	XmagicConstant.EffectName.BEAUTY_FACE_THIN_CHEEKBONE	BEAUTY_FACE_T
单点美妆	口红	XmagicConstant.EffectName.BEAUTY_MOUTH_LIPSTICK	BEAUTY_MOUTH
	腮红	XmagicConstant.EffectName.BEAUTY_FACE_RED_CHEEK	BEAUTY_FACE_R

	立体	XmagicConstant.EffectName.BEAUTY_FACE_SOFTLIGHT	BEAUTY_FACE_S
	眼影	XmagicConstant.EffectName.BEAUTY_FACE_MAKEUP_EYE_SHADOW	BEAUTY_FACE_E
	眼线	XmagicConstant.EffectName.BEAUTY_FACE_MAKEUP_EYE_LINER	BEAUTY_FACE_E
	睫毛	XmagicConstant.EffectName.BEAUTY_FACE_MAKEUP_EYELASH	BEAUTY_FACE_E
	眉毛	XmagicConstant.EffectName.BEAUTY_FACE_MAKEUP_EYEBROW	BEAUTY_FACE_E
	美瞳	XmagicConstant.EffectName.BEAUTY_FACE_MAKEUP_EYEBALL	BEAUTY_FACE_E
美体	一键瘦身	XmagicConstant.EffectName.BODY_AUTOTHIN_BODY_STRENGTH	BODY_AUTOTHIN
	长腿	XmagicConstant.EffectName.BODY_LEG_STRETCH	BODY_LEG_STRE
	瘦腿	XmagicConstant.EffectName.BODY_SLIM_LEG_STRENGTH	BODY_SLIM_LEG
	瘦腰	XmagicConstant.EffectName.BODY_WAIST_STRENGTH	BODY_WAIST_ST
	瘦肩	XmagicConstant.EffectName.BODY_THIN_SHOULDER_STRENGTH	BODY_THIN_SHO
	胸部调整	XmagicConstant.EffectName.BODY_ENLARGE_CHEST_STRENGTH	BODY_ENLARGE
	小头	XmagicConstant.EffectName.BODY_SLIM_HEAD_STRENGTH	BODY_SLIM_HEA

滤镜、美妆、动效、分割

	effectName		

		常量名(Android)	常量名(iOS)	字符串
滤镜	无	XmagicConstant.EffectName.EFFECT_LUT	EFFECT_LUT	lut
美妆	无	XmagicConstant.EffectName.EFFECT_MAKEUP	EFFECT_MAKEUP	makeu
动效	无	XmagicConstant.EffectName.EFFECT_MOTION	EFFECT_MOTION	motion
背景分割	普通	XmagicConstant.EffectName.EFFECT_SEGMENTATION	EFFECT_SEGMENTATION	segme
	绿幕分割	XmagicConstant.EffectName.EFFECT_SEGMENTATION	EFFECT_SEGMENTATION	segme
	自定义背景	XmagicConstant.EffectName.EFFECT_SEGMENTATION	EFFECT_SEGMENTATION	segme

美颜场景推荐参数

最近更新时间：2022-07-18 10:06:18

对于腾讯特效的使用场景，我们在以下几个场景给出了功能推荐参数，便于您进行参考。

方案一：效果自然

若您需要使用效果自然的美颜场景，可参考以下参数：

功能类型	参数推荐
美白	30
磨皮	45
红润	20
大眼	20
瘦脸（自然选项）	30

方案二：女神效果

若您需要使用女神效果的美颜场景，可参考以下参数：

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

方案三：英俊特效

若您需要使用男生自然的美颜场景，可参考以下参数：

功能类型	参数推荐
美白	25
磨皮	40
红润	20
瘦脸（英俊选项）	40

短视频企业版迁移指引

最近更新时间：2022-07-18 10:06:18

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 工程中的 xmagic 模块引入到实际项目工程中。

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。

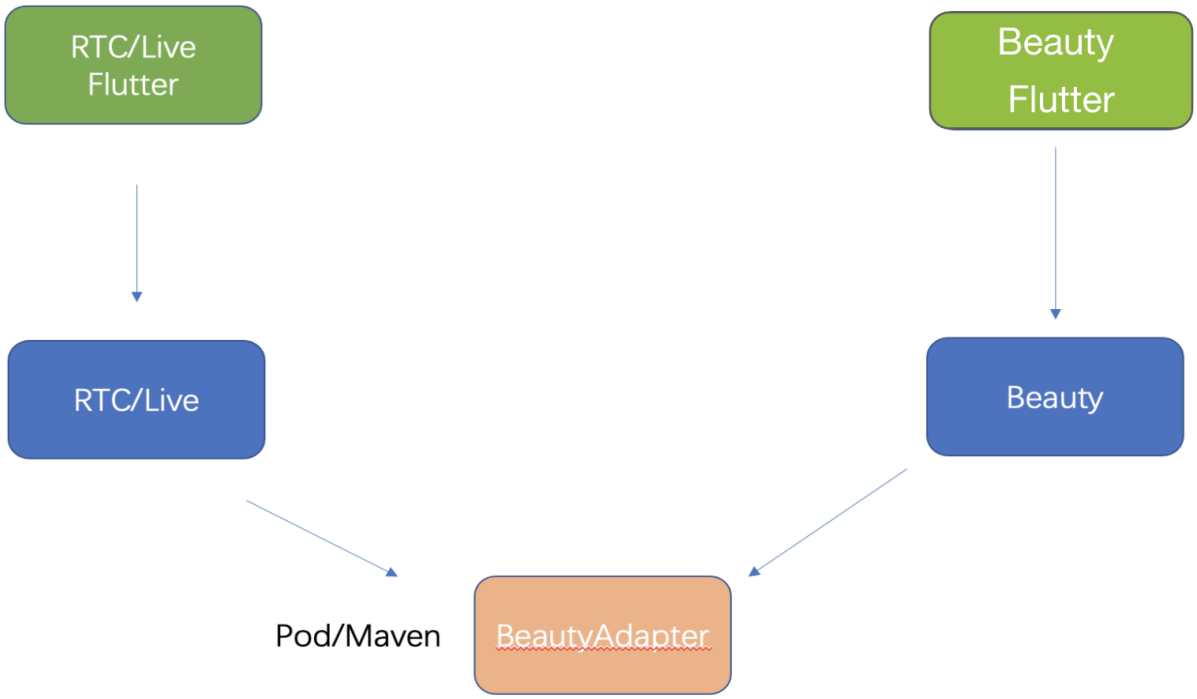
步骤六：删除对 beautysettingkit 模块的依赖

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：2022-11-30 18:02:12

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

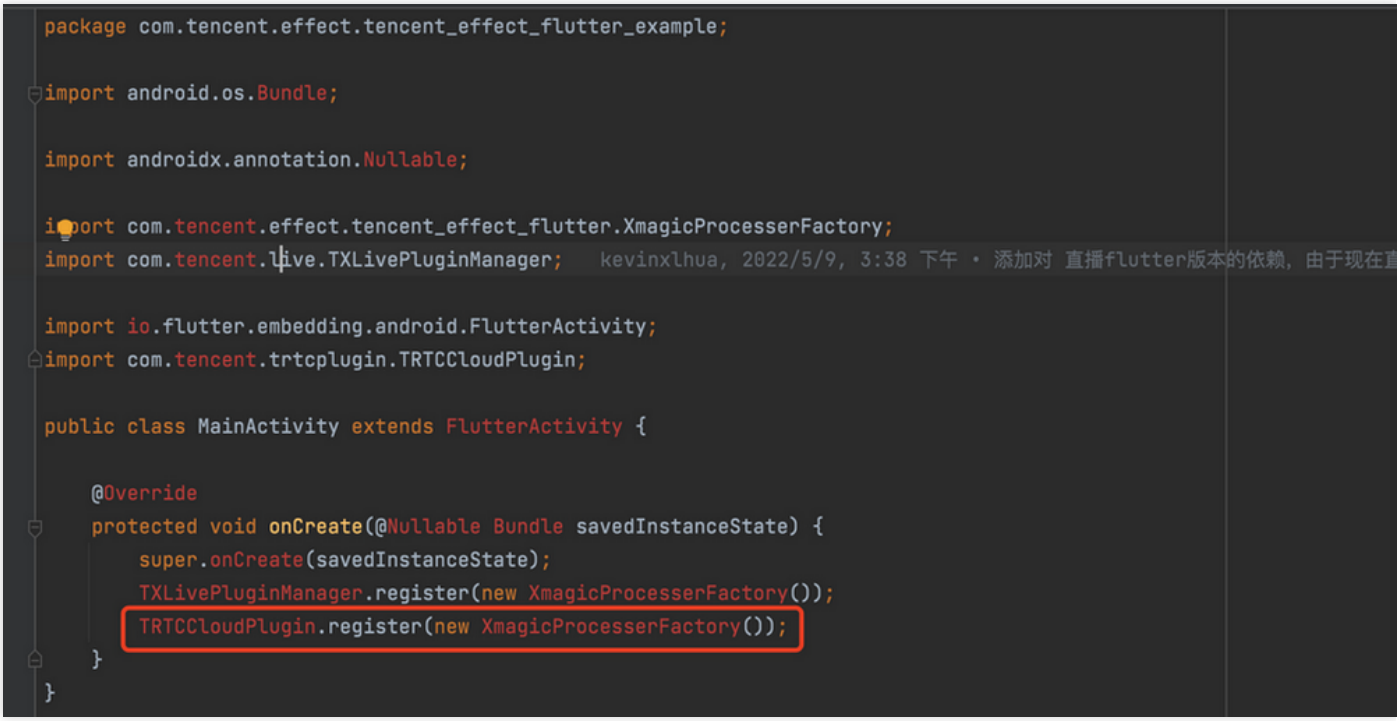
1. 美颜侧抽象一层接口，并在美颜侧实现了接口。
2. 在应用启动时将此接口注册到三方推流端，这样三方推流端就可以通过此接口进行创建、使用、销毁美颜实例。
3. 三方推流端再将创建和销毁美颜的能力暴露给自己的 Flutter 端供客户使用。
4. 美颜属性设置可通过美颜提供的 Flutter SDK 能力进行处理。

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, "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(convertTRTCPixelFormat(pixelFormat), convertTRTCBufferType(bufferType), processVideo);
        result.success(ret);
    } else {
        int ret = trtcCloud.setLocalVideoProcessListener(convertTRTCPixelFormat(pixelFormat), convertTRTCBufferType(bufferType), null);
        // processerFactory.destroyCustomBeautyProcesser();
        mCustomBeautyProcesser = null;
        result.success(ret);
    }
}
```

When set to null, the onGlContextDestroy method of the last set processVideo will be called back

```java
.dart  ProcessVideoFrame.java
package com.tencent.trtcplugin.listener;
import com.tencent.live.beauty.custom.TXCustomBeautyDef;
import com.tencent.trtc.TRTCCloudDef;
import com.tencent.trtc.TRTCCloudListener;

import com.tencent.live.beauty.custom.ITXCustomBeautyProcesserFactory;
import com.tencent.live.beauty.custom.ITXCustomBeautyProcesser;
import com.tencent.trtcplugin.TRTCCloudPlugin;

import static com.tencent.live.beauty.custom.TXCustomBeautyDef.TXCustomBeautyBufferType;
import static com.tencent.live.beauty.custom.TXCustomBeautyDef.TXCustomBeautyPixelFormat;
import static com.tencent.live.beauty.custom.TXCustomBeautyDef.TXCustomBeautyVideoFrame;

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

    public ProcessVideoFrame(ITXCustomBeautyProcesser processer) {
        mCustomBeautyProcesser = processer;
    }


    private static TXCustomBeautyVideoFrame createCustomBeautyVideoFrame(TRTCCloudDef.TRTCVideoFrame frame) {...}


    public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame  srcFrame,
                                   TRTCCloudDef.TRTCVideoFrame  dstFrame) {...}


    public void onGLContextCreated() {}

    public void onGLContextDestory() {...}
}
```

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：2023-04-11 16:08:06

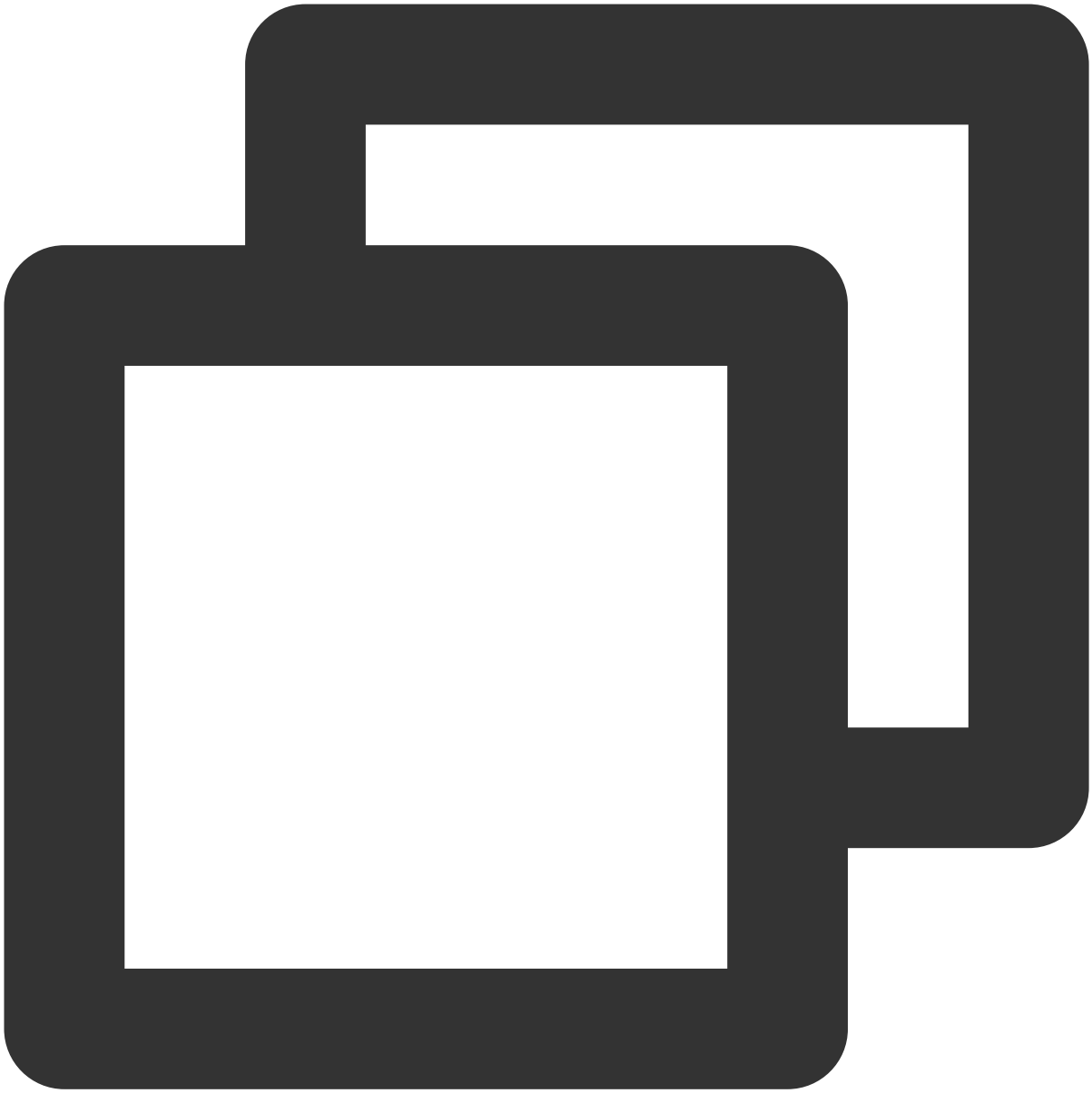
准备工作

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

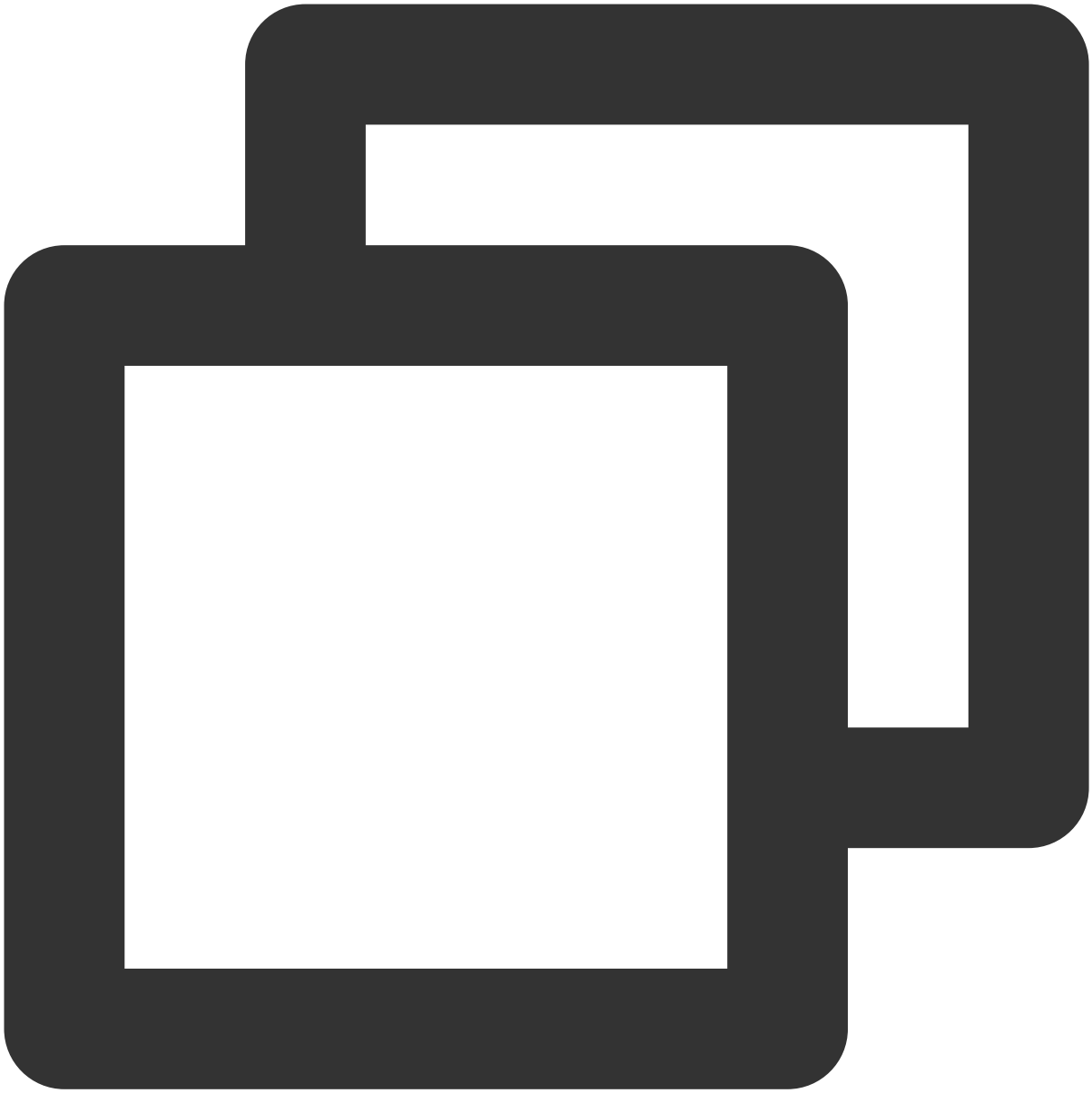
步骤1：小程序后台配置域名白名单

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：

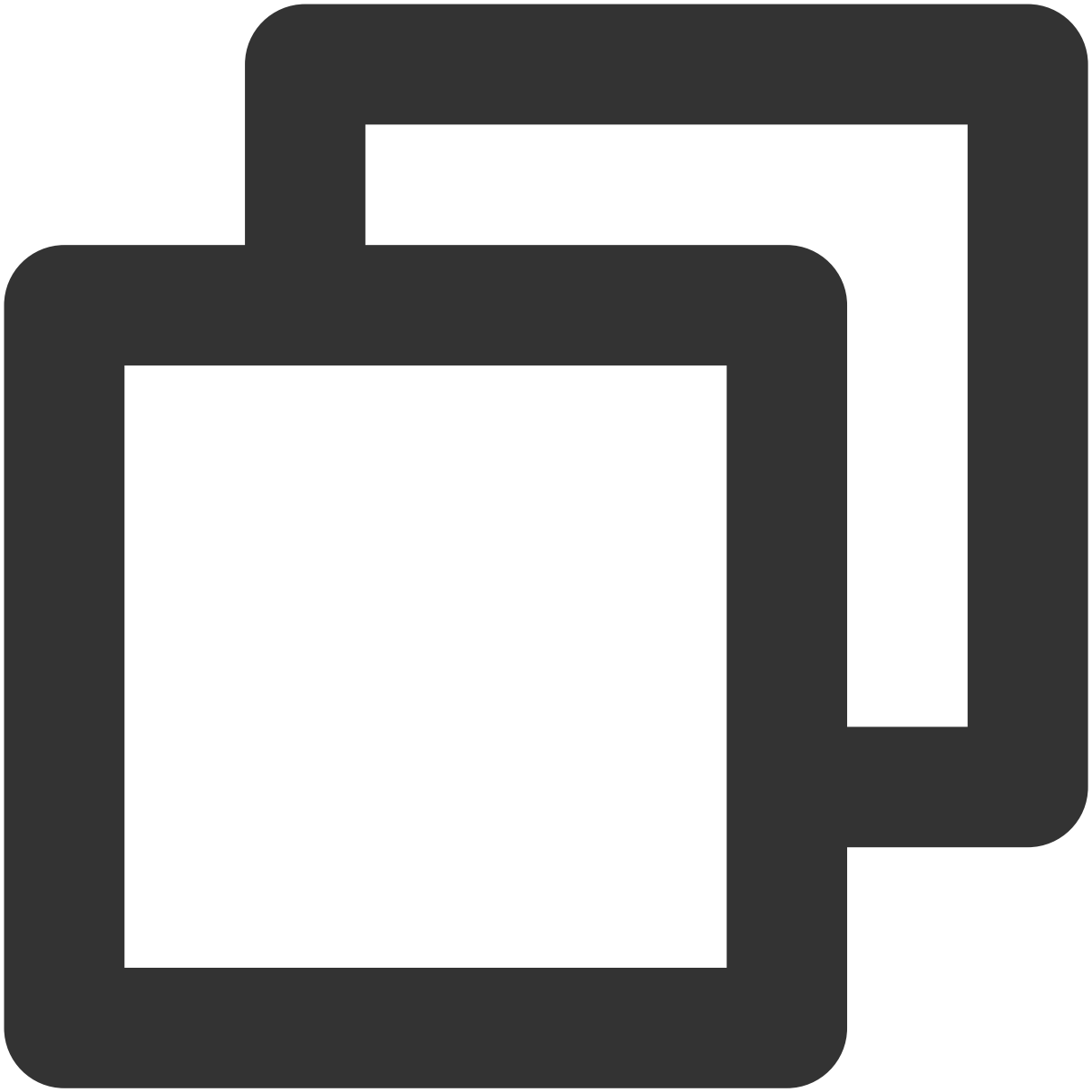


```
https://webar-static.tencent-cloud.com
```

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：

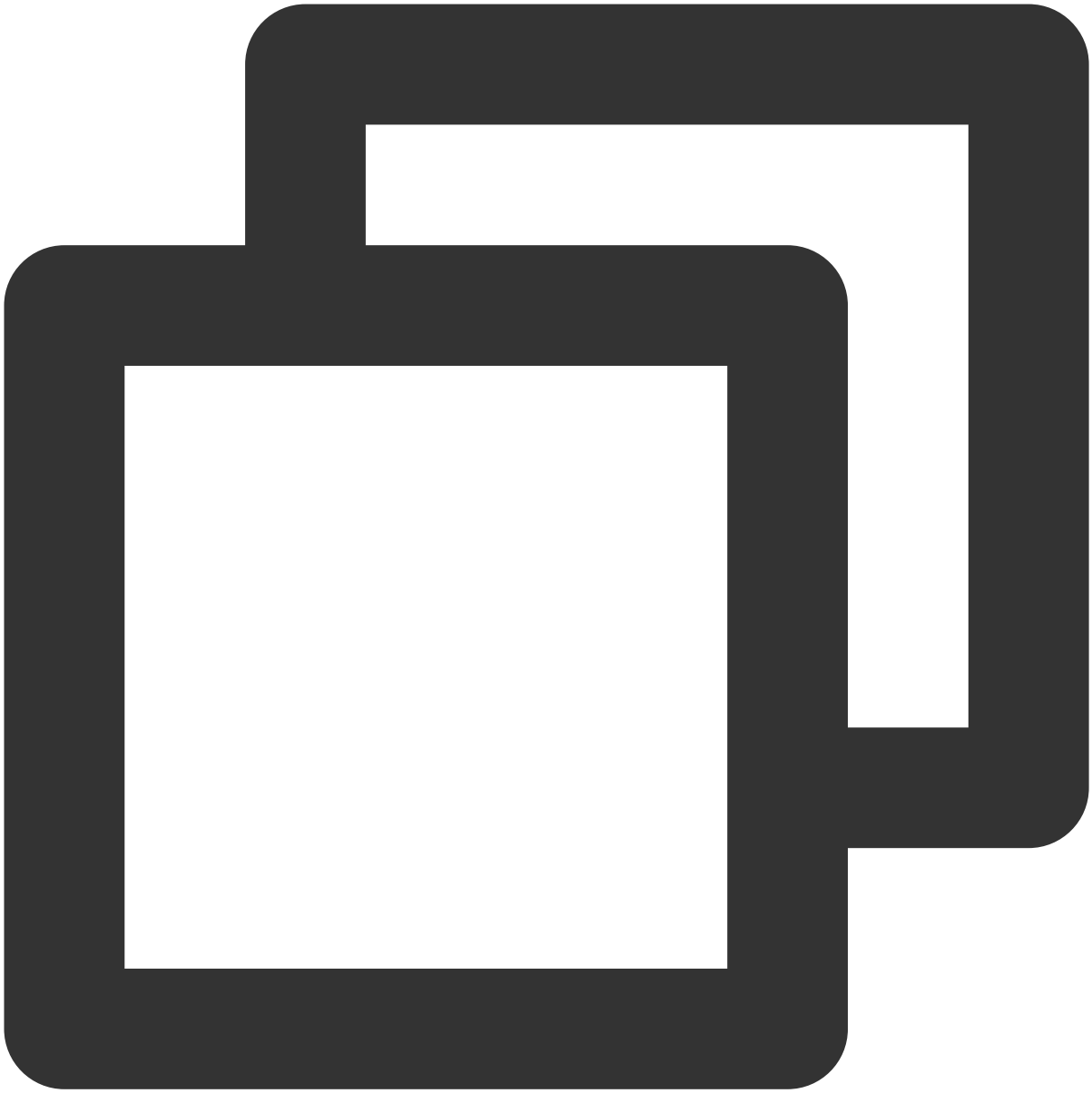


```
npm install tencentcloud-webar
```

2. 构建：

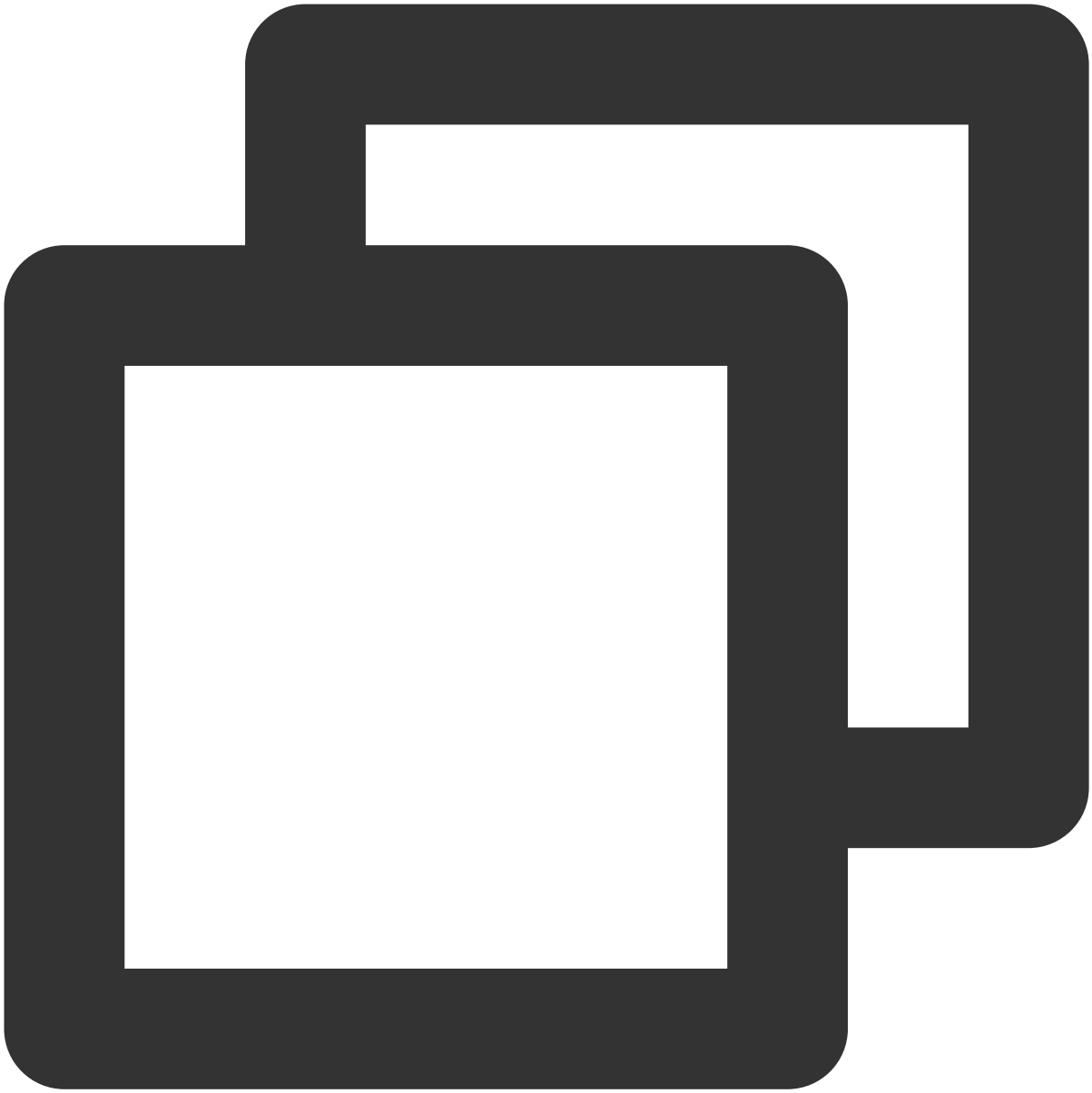
打开小程序开发者工具，顶部菜单选择 **工具 > 构建 npm**。

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

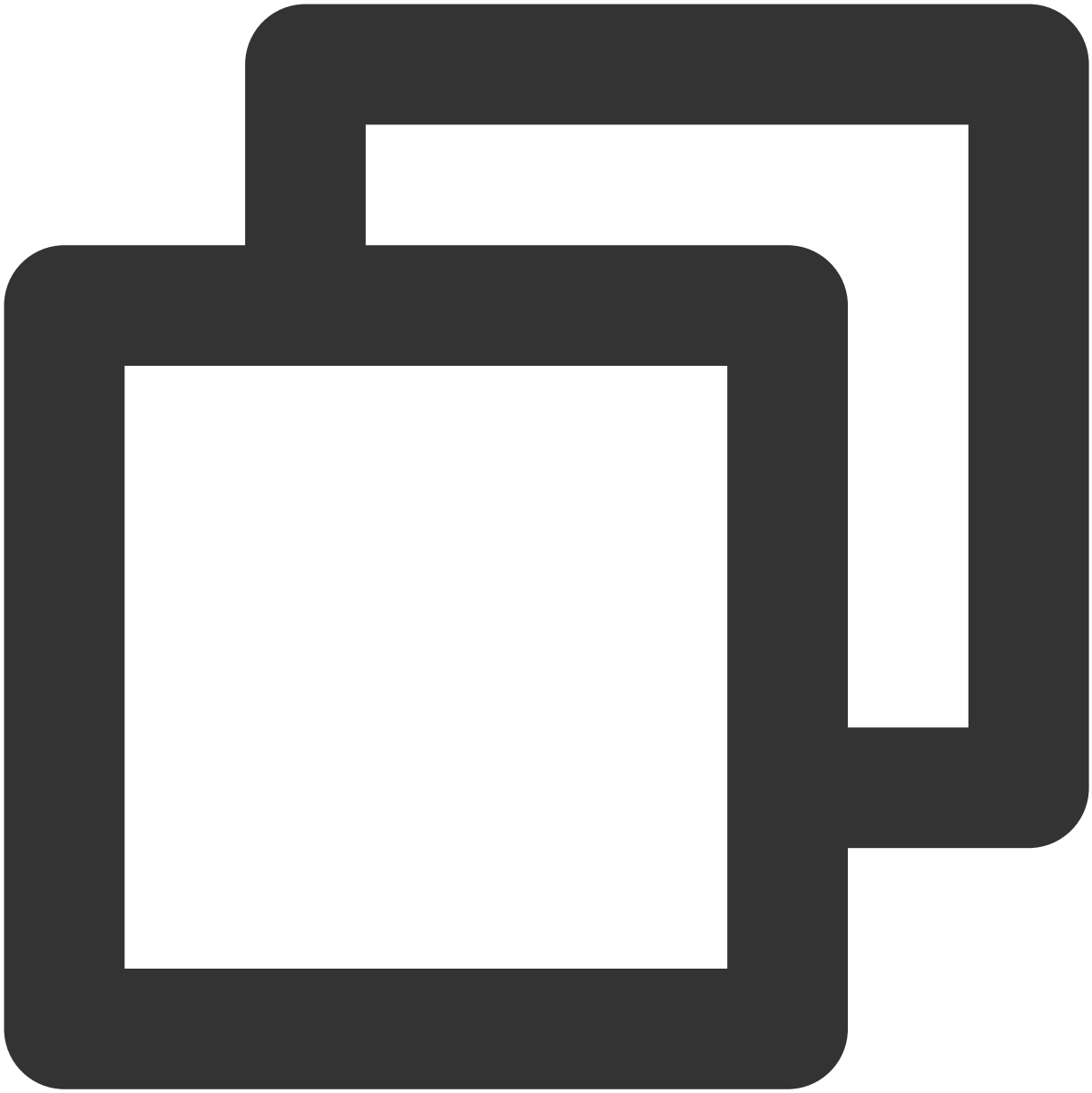
注意

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
 */
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
 * [签名方法](https://cloud.tencent.com/document/product/616/71370#.E7.AD.BE.E5.90.8I
 */
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).toUpper
        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')>
                const stickerList = list.filter(item=>item.label.indexOf('贴纸')
                // 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);
        }
    }
})
```

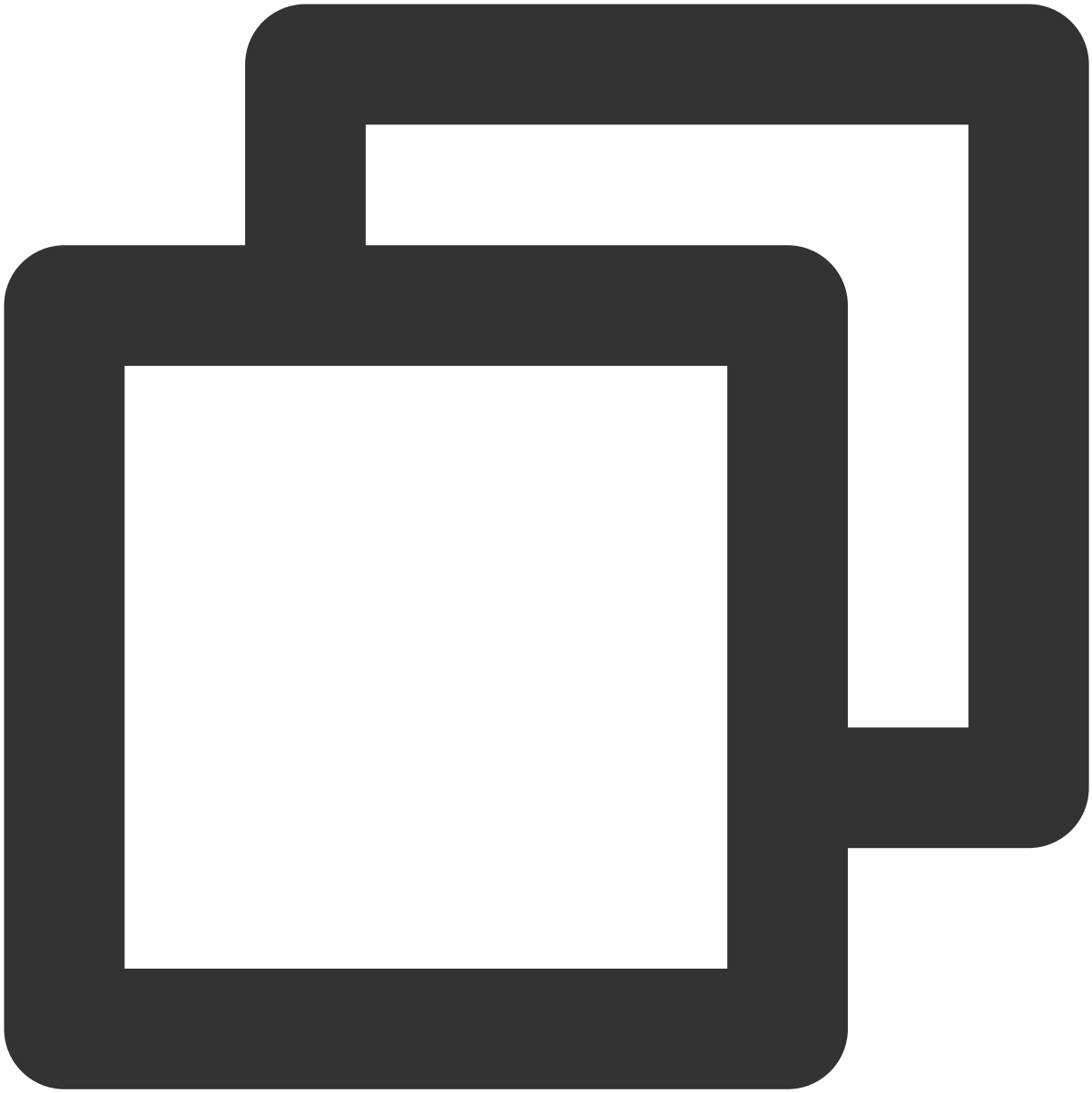
步骤5：拍照和录像功能实现

示例代码如下：

拍照

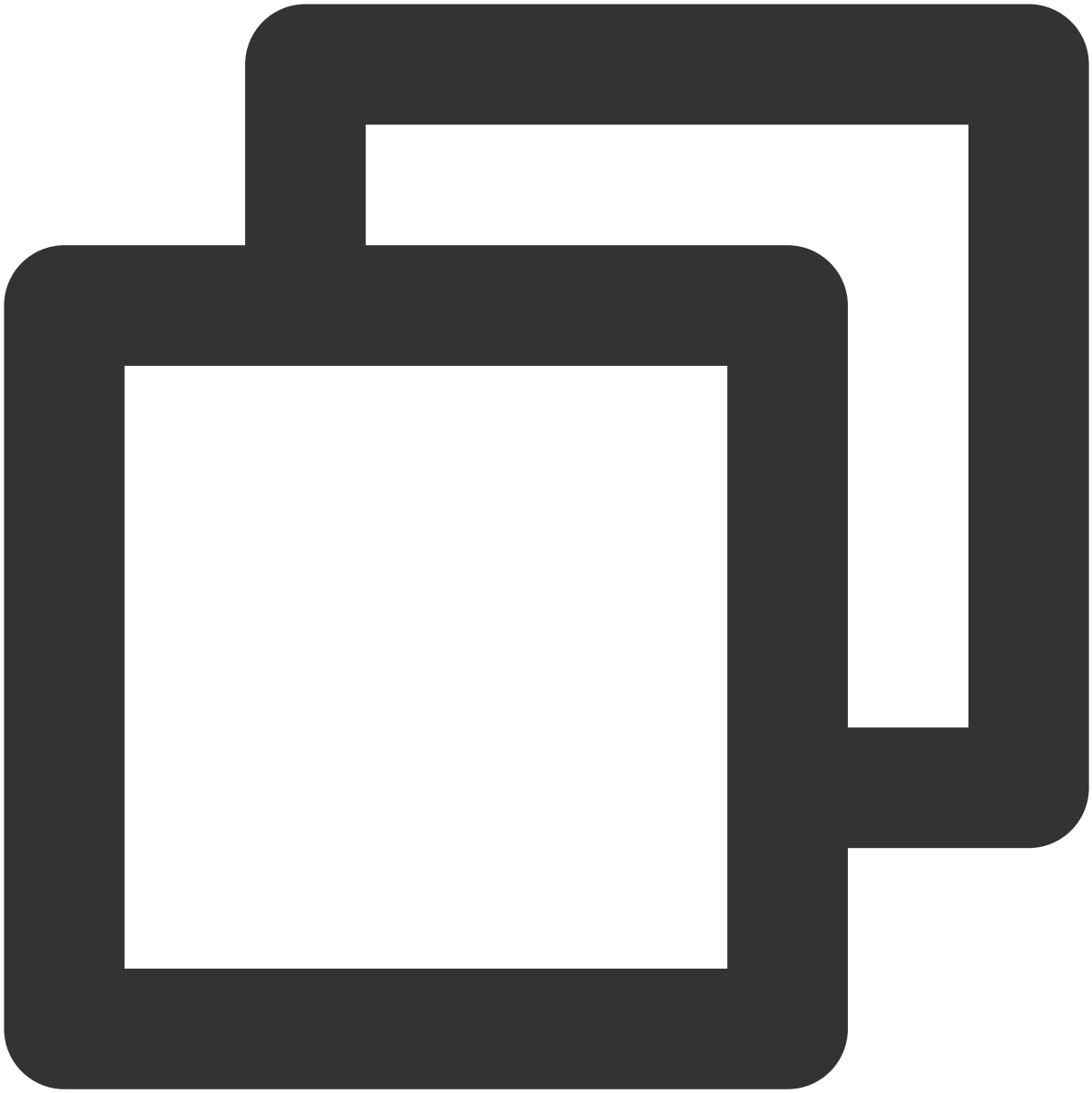
录像

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
    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
```

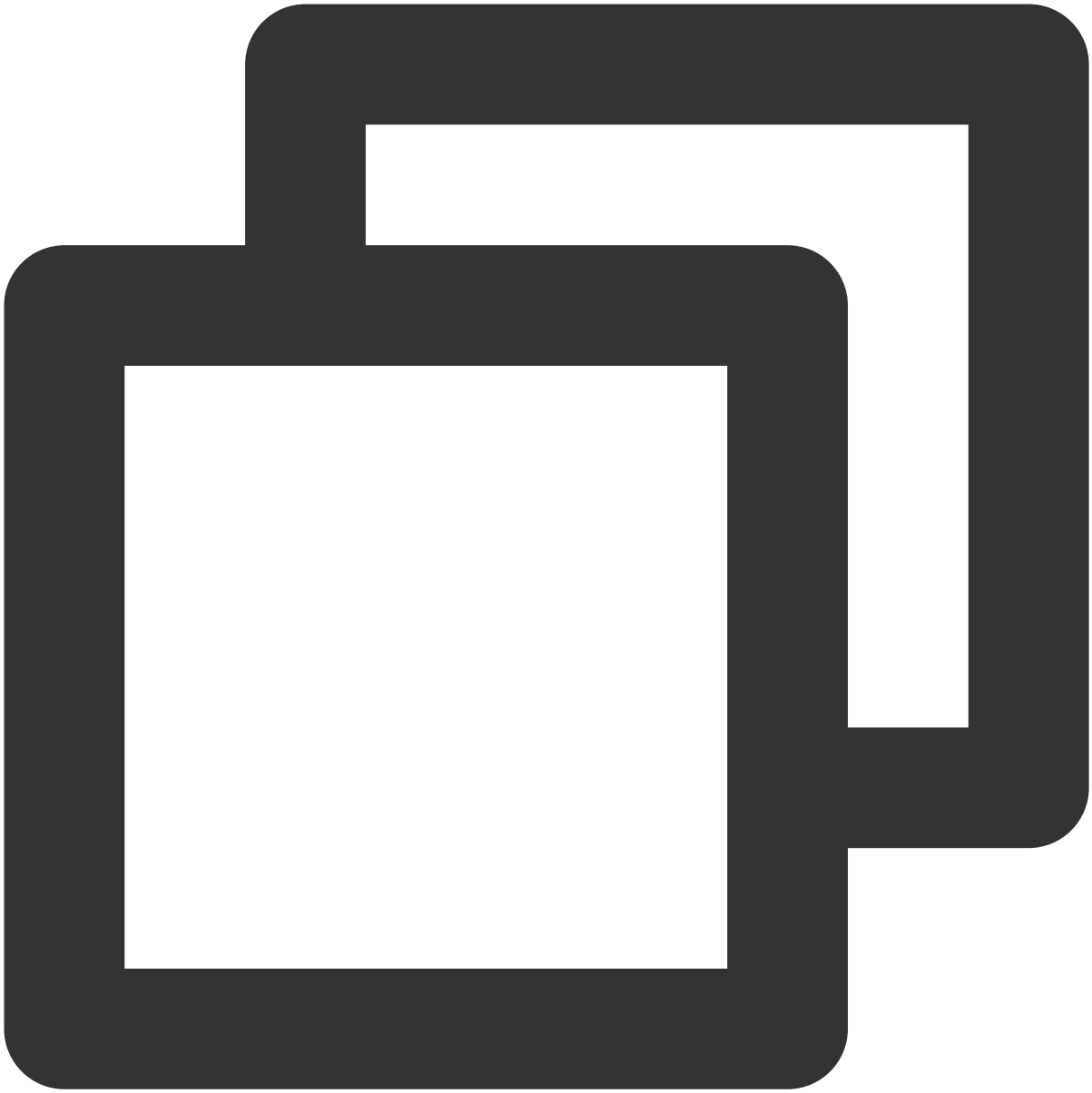
```
    wx.canvasToTempFilePath({
        canvas: photoCanvasNode,
        x: 0,
        y: 0,
        width: width,
        height: height,
        destWidth: width,
        destHeight: height,
        success: (res) => {
            // 保存照片到本地
            wx.saveImageToPhotosAlbum({
                filePath: res.tempFilePath
            });
        }
    })
}
```



```
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。

素材制作工具使用
素材制作工具介绍

最近更新时间：2024-03-19 15:28:54

产品介绍

素材制作工具（Tencent Effect Studio）支持客户自定义 2D、3D 贴纸，进行个性化的美妆素材制作，制作完成后导入到 SDK 中即可使用。

工具特点

Ai 配置：AI 能力组件化编辑。
3D 编辑：进行 3D 模型素材设计。
流程控制：丰富的流程控制选项，制作各类复杂动态效果。
动态效果导入：支持导入本地动态素材查看效果。

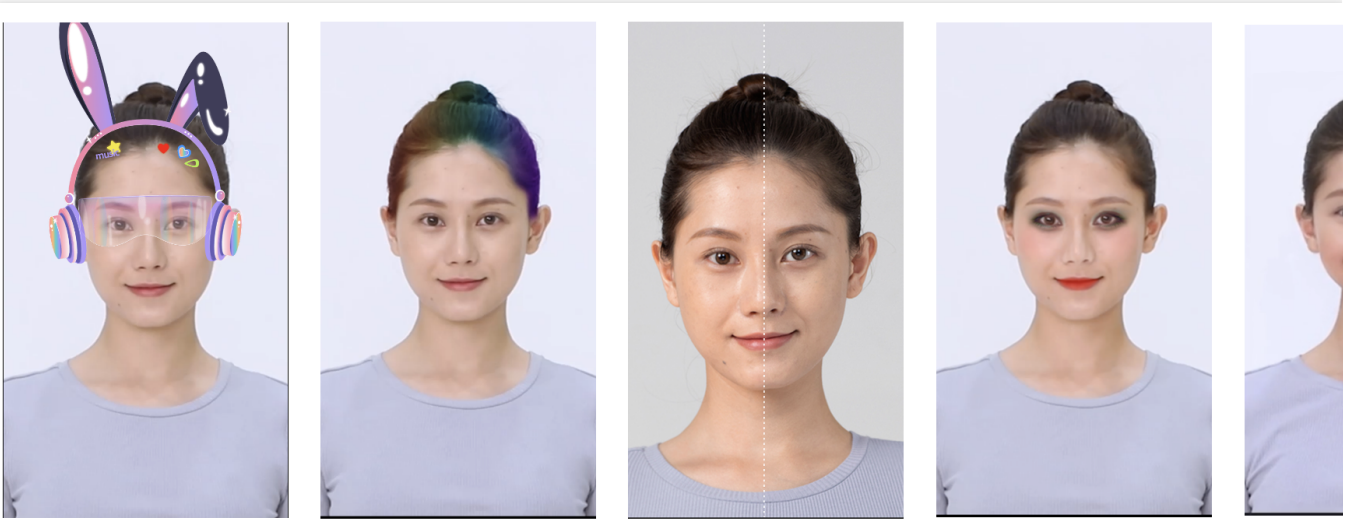
工具下载

Windows	Mac
下载	下载

产品能力介绍

1、2D 贴纸&美妆特效设计

代码环境面板可视化，设计小白也可以轻松上手。



2、3D 互动特效

支持各类材质的渲染，实现丰富的特效制作效果。

3、素材制作能力一览

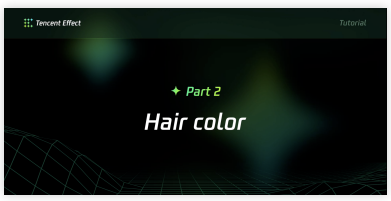
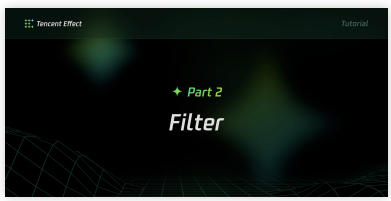
分类	细项	分类	细项
通用能力	增加滤镜 增加音乐	AI	分割 抠五官

	增加流程管理 增加脚本		扣头 AI 变脸 人脸融合 表情迁移
贴纸	前景贴纸 人脸跟随贴纸 手势跟随贴纸 身体跟随贴纸	特效	定格一帧 截帧 捏脸 液化 后处理 avatar2D
美妆	脸妆 眉妆 眼妆 美瞳 唇彩	3D	摄像头 光源 3D人头 ……
美颜	脸部美型 美体 染发	3D 模型	方体 球体 圆柱体 平面 ……

素材制作工具视频教程

最近更新时间：2024-03-12 14:32:18

我们对如何使用素材制作工具提供了**视频教程**以及相关资源，您可单击下载查看。

项目	下载查看		
界面介绍	界面介绍 		
2D效果	面部处理 	妆容 	面部跟随 
	发色 	滤镜 	音乐设置 
	背景设置		

			
3D 效果	3D 效果 		
相关资源	教程体验过程中可能用到的相关资源		