

腾讯特效 SDK SDK 集成指引(无 UI)







【版权声明】

©2013-2025 腾讯云版权所有

本文档(含所有文字、数据、图片等内容)完整的著作权归腾讯云计算(北京)有限责任公司单独所有,未经腾讯云事先明确书面许可,任何主体不得以任何形式 复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯,腾讯云将依法采取措施追究法律责任。

【商标声明】

🕗 腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算(北京)有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标,依法由权利人所有。未经腾讯云及有关 权利人书面许可,任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为,否则将构成对腾讯云及有关权利人商标权的侵犯,腾讯云将依 法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况,部分产品、服务的内容可能不时有所调整。 您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定,除非双方另有约定,否则,腾讯云对本文档内容不做任何明示或默示的承 诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务,及相应的技术售后服务,任何问题请联系 4009100100或95716。



文档目录

SDK 集成指引(无 UI) 通用集成腾讯特效 Android Windows Mac iOS 原子能力集成指引 手势识别 人脸点位 人像分割 人脸属性 iOS Android 人脸表情 iOS Android 身体点位 iOS Android 语音转表情 iOS Android

SDK 集成指引(无 UI) 通用集成腾讯特效 Android

最近更新时间: 2025-02-20 16:16:52

开发者环境要求

- 最低兼容 Android 4.4 (SDK API Level 19),建议使用 Android 7.0 (SDK API Level 24)及以上版本。
- Android Studio 3.5 及以上版本。

Demo 工程: TEBeauty_API_Example

从 github clone 出 <mark>Demo 工程</mark>,其中的 TEBeautyDemo 是含 UI 的 demo 工程,TEBeauty_API_Example 是不含 UI 的 demo 工程。 按照 TEBeauty_API_Example/README 文档中的指引将 TEBeauty_API_Example 运行起来,然后结合本文了解无UI 集成 SDK 的详细步骤。

集成SDK

△ 注意:

Github 上的 demo 工程采用 Maven 方式集成 SDK。

Maven 集成

腾讯特效 SDK 已经发布到 mavenCentral 库,您可以通过配置 gradle 自动下载更新。

1. 在 dependencies 中添加腾讯特效 SDK 的依赖。

```
dependencies {
    //例如: S1-04套餐如下:
    implementation 'com.tencent.mediacloud:TencentEffect_S1-04:版本号'
    //"版本号"可以在官网的"SDK下载/版本历史"页面看到,例如 3.0.0.13。"版本号"也可以使用"latest.release" ,
    //但请注意: 这会让您使用的SDK始终保持最新版,在一些变化比较大的版本上可能不符合您的预期,请慎重使用"latest.release"
}
```

2. 在 defaultConfig 中,指定 App 使用的 CPU 架构。

```
defaultConfig {
   ndk {
      abiFilters "armeabi-v7a", "arm64-v8a"
   }
}
```

 说明: 目前特效 SDK 支持 armeabi-v7a 和 arm64-v8a。

3. 单击 🅪 Sync Now,自动下载 SDK 并集成到工程里。

各套餐对应的 Maven 地址

版本	Maven 地址
A1 - 01	implementation 'com.tencent.mediacloud:TencentEffect_A1-01:版本号'
A1 - 02	implementation 'com.tencent.mediacloud:TencentEffect_A1-02:版本号'
A1 - 03	implementation 'com.tencent.mediacloud:TencentEffect_A1-03:版本号'



A1 - 04	implementation 'com.tencent.mediacloud:TencentEffect_A1-04:版本号'
A1 - 05	implementation 'com.tencent.mediacloud:TencentEffect_A1-05:版本号'
A1 - 06	implementation 'com.tencent.mediacloud:TencentEffect_A1-06:版本号'
S1 - 00	implementation 'com.tencent.mediacloud:TencentEffect_S1-00:版本号'
S1 - 01	implementation 'com.tencent.mediacloud:TencentEffect_S1-01:版本号'
S1 - 02	implementation 'com.tencent.mediacloud:TencentEffect_S1-02:版本号'
S1 - 03	implementation 'com.tencent.mediacloud:TencentEffect_S1-03:版本号'
S1 - 04	implementation 'com.tencent.mediacloud:TencentEffect_S1-04:版本号'
S1 - 05	implementation 'com.tencent.mediacloud:TencentEffect_S1-05:版本号'
S1 - 06	implementation 'com.tencent.mediacloud:TencentEffect_S1-06:版本号'
S1 - 07	implementation 'com.tencent.mediacloud:TencentEffect_S1-07:版本号'

手动集成(资源内置)

下载 SDK



集成

将 SDK 文件夹下的 xmagic-xxxx.aar 文件拷贝到您工程 libs 目录下。

导入方法

打开 app 模块的 build.gradle 添加依赖引用:

```
android{
    ...
    defaultConfig {
        applicationId "修改成与授权license绑定的包名"
        ....
    }
    packagingOptions {
        pickFirst '**/libc++_shared.so'
    }
}
dependencies{
    ...
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])//添加 *.aar
```



▲ 注意:

项目中还需添加如下依赖:

```
dependencies{
    implementation 'com.google.code.gson:gson:2.8.2'
    //版本在2.6.0到3.1.0.2 需要添加
    implementation 'androidx.exifinterface:exifinterface:1.3.3'
    //3.5.0 版本及之后需要添加
    implementation 'com.tencent.tav:libpag:4.3.33-noffavc'
}
```

如果您想使用其他版本的 pag,请点击 此处 查看。

手动集成(资源动态下载)

动态下载 assets、so、动效资源指引

- 为了减少包大小,您可以将 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。动态下载的详细指引,请参见 SDK 包体瘦身(Android)。

集成素材

如果您的套餐包含动效和滤镜功能,那么需要在 SDK 下载页面 下载对应的套餐包,解压之后,将 resource 目录下的 lut 和 MotionRes 放置到您工程的 assets 目录下:

- 动效: ..src/main/assets/MotionRes
- 滤镜: ..src/main/assets/lut

更多素材配置可参考 素材使用指南。

▲ 注意:

SDK中包含的素材是测试素材,正式素材需要您在购买套餐之后联系我们 工作人员 进行获取。

SDK 使用流程

步骤一: 鉴权

- 1. 申请授权,得到 License URL 和 License KEY,请参见 License 指引。
- 2. 在相关业务模块的初始化代码中设置 URL 和 KEY, 触发 License 下载,避免在使用前才临时去下载。例如我们的 demo 工程是在 Application 的 onCreate 方法里触发下载,但在您的项目中不建议在这里触发,因为此时可能没有网络权限或联网失败率较高,请选择更合适的时机触发 license 下载。

/**如果仅仅是为了触发下载或更新**license,而不关心鉴权结果,则第4个参数传入null。 ELicenseCheck.getInstance().setTELicense(context, URL, KEY, null);

3. 然后在真正要使用美颜功能前,再去做鉴权:

```
ELicenseCheck.getInstance().setTELicense(context, URL, KEY, new TELicenseCheckListener() {
    @Override
    public void onLicenseCheckFinish(int errorCode, String msg) {
        //注意:此回调不一定在调用线程
```



if (errorCode == TELicenseCheck.ERROR_OK) {
// 金代 100-49
//鉴权失败

鉴权 errorCode 说明:

错误码	说明
0	成功。Success
-1	输入参数无效,例如 URL 或 KEY 为空
-3	下载环节失败,请检查网络设置
-4	从本地读取的 TE 授权信息为空,可能是 IO 失败引起
-5	读取 VCUBE TEMP License文件内容为空,可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理
-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的 TE 授权信息为空。请联系腾讯云团队处理
-11	把TE授权信息写到本地文件时失败,可能是 IO 失败引起
-12	下载失败,解析本地 asset 也失败
-13	鉴权失败,请检查 so 是否在包里,或者已正确设置 so 路径
3004/3005	无效授权。请联系腾讯云团队处理
3015	Bundle Id / Package Name 不匹配。检查您的 App 使用的 Bundle Id / Package Name 和申请的是否一致,检查是否使用 了正确的授权文件
3018	授权文件已过期,需要向腾讯云申请续期
其他	请联系腾讯云团队处理

步骤二:资源拷贝

这里所指的资源文件包含两部分:

• SDK 的模型文件,位于 SDK 的 aar 包的 assets 目录。

• 滤镜和动效资源文件,位于 demo 工程的 assets 目录,命名分别是 lut 和 MotionRes。

使用美颜前需要将上述资源拷贝到 app 的私有目录。在未更新 SDK 版本的情况下,只需要拷贝一次。拷贝成功后,您可以在 App 的 SharedPreference 中 记录下来,下次就不用再拷贝了。具体可以参见 demo 工程的 TEMenuActivity.java

```
String resPath = new File(getFilesDir(),
AppConfig.getInstance().getBeautyFileDirName()).getAbsolutePath()
if (!resPath.endsWith(File.separator)) {
    resPath = resPath + File.separator;
}
AppConfig.resPathForSDK = resPath;
AppConfig.lutFilterPath = resPath + "light_material/lut";
AppConfig.motionResPath = resPath + "MotionRes";
```



new Thread(() -> {
Context context = getApplicationContext();
<pre>int addResult = XmagicApi.addAiModeFilesFromAssets(context, AppConfig.resPathForSDK);</pre>
Log.d(TAG, "copyRes, add ai model files result = " + addResult);
String lutDirNameInAsset = "lut".
booleen result = FileNtil conviseets (context lutDirNemeIndeset AppConfig lutFilterDeth):
Log.d(TAG. "copyRes. copy lut. result = " + result):
<pre>String motionResDirNameInAsset = "MotionRes";</pre>
<pre>boolean result2 = FileUtil.copyAssets(context, motionResDirNameInAsset, AppConfig.motionResPath);</pre>
Log.d(TAG, "copyRes, copy motion res, result = " + result2);
}).start();

步骤三: SDK 初始化及使用方法

1. 快速实现相机(可选)

我们假定您已经实现了相机应用,能正常启动相机,且能将相机的 SurfaceTexture 纹理信息回调到 Activity 用于美颜处理,如下所示:

@Override	
public int onCustomProcessTexture(int textureId, int textureWidth, int textureHeight) { //美颜SDK 在这里处理 textureId ,为其添加美颜和特效,并返回处理后的新的 textureID	

如果您尚未实现相机应用,可以参见 demo 工程的 TECameraBaseActivity.java ,使用 GLCameraXView 这个组件,将它添加到您的 Activity 的 layout 中,以快速实现相机预览:



2. 初始化美颜 SDK,可以在任意线程初始化,建议在子线程中进行。创建时机需要在 License 鉴权成功之后。

//AppConfig.resPathForSDK**是资源拷贝环节确定的资源路径** mXmaqicApi = new XmagicApi(this, AppConfig.resPathForSDI

参数

参数	含义
Context context	上下文
String resDir	资源文件目录,详见请参见 <mark>步骤二</mark>
OnXmagicPropertyErrorListener errorListener	可选,回调函数实现类,用于回调SDK初始化和使用过程中的一些错误码,错误码含义请参见 API 文档

3. (可选)添加素材提示语回调函数(方法回调有可能运行在子线程),部分素材会提示用户:点点头、伸出手掌、比心,这个回调就是用于展示类似的提示 语。







4. 美颜 SDK 处理每帧数据并返回相应处理结果。process 方法详细说明见 API 文档。

```
@Override
public int onCustomProcessTexture(int textureId, int textureWidth, int textureHeight) {
    return mXmagicApi.process(textureId, textureWidth, textureHeight);
}
5. 设置美颜或特效。
• 3.5.0版本及以后使用 setEffect 方法方法详细说明见 API文档。
• 3.3.0版本及之后使用此方法
    mXmagicApi.setEffect(String effectName, int effectValue, String resourcePath, Map<String, String>
    extraInfo)
    //例如设置美白,强度50
    //mXmagicApi.setEffect(XmagicConstant.EffectName.BEAUTY_WHITEN, 50, null,null);

    // 可用的入参属性可以从 xmagicResParser.parseRes() 获得
    // 3.1.0版本及之前使用此方法
    @Deprecated
    mXmagicApi.updateProperty(XmagicProperty<?> p);
6. 生命周期方法 onResume , 建议在 Activity 的 onResume() 方法中调用, 调用后会恢复特效里的声音。
```

XmagicApi.onResume();

7. 生命周期方法 onPause ,建议在 Activity 的 onPause () 方法调用,调用后会暂停特效里的声音。

NXmagicApi.onPause();

8. 释放美颜 SDK,在OpenGL环境销毁时调用,**需要在 GL 线程中调用,不能在主线程(Activity的onDestroy里)调用**,否则可能造成资源泄露,多次进 出后引起白屏、黑屏现象。

```
@Override
public void onGLContextDestroy() {
    mXmagicApi.onDestroy();
}
```

步骤四:混淆配置

- 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	org.libpag.** { *;}
-keep	org.extra.** { *;}
-keep	<pre>com.gyailib.**{ *;}</pre>
-keep	<pre>com.tencent.cloud.iai.lib.** {</pre>
-keep	<pre>com.tencent.beacon.** { *;}</pre>
-keep	<pre>com.tencent.qimei.** { *;}</pre>
-keep	<pre>androidx.exifinterface.** { *;</pre>

附件(SDK文件结构):

<u>♪ 注意</u>:

此表格列出了 SDK 用到的所有文件,可能您的套餐中没有某些文件,但并不影响该套餐功能的使用。

文件类型			说明		
	audio2exp		avatar 虚拟人语音驱动模型,如果不使用该功能,则无需该模型		
	benchmark		机型适配使用		
	Light3DPlugin		3D 贴纸使用		
	LightBodvPlu	LightBody3DModel.bu ndle	人体 3D 骨骼点位使用		
assets	gin	LightBodyModel.bundl e	美体功能使用		
	LightCore		SDK 核心模型资源		
	LightHandPlugir	n	手势贴纸、手部点位能力需要		
	LightSegmentPl	lugin	背景分割能力需要使用		
	lut		免费的滤镜资源		
demo_xxx_android_xx xx			demo 工程		
	libace_zplan.so		3D 引擎库		
	libaudio2exp.so		avatar 虚拟人语音驱动库,如果不使用该功能,则无需该库		
	libc++_shared.so		libc++_shared.so 是一个 C++ 标准库的共享库,它提供了一 组C++ 标准库函数和类,用于支持 C++ 程序的开发和运行。它 在 Android 系统中被广泛使用,是 C++ 应用程序和库的重要组 成部分。如果您的工程中已有 C++ 共享库,可以只保留一份		
jniLibs	liblight-sdk.so		light sdk 核心库		
	libpag.so		light sdk 依赖的动画文件库		
	libtecodec.so		light sdk 依赖的编解码库		
	libv8jni.so		light sdk 依赖的用于解析 JavaScript 的库		
	libYTCommonX	Magic.so	license 鉴权使用		
libs	xmagic-xxxx.aa	ar	美颜 SDK 的 aar 文件		
MotionRes	2dMotionRes		2D 贴纸		
	3dMotionRes		3D 贴纸		
	avatarRes		Avatar素材		



	ganMotionRes	童趣贴纸
	handMotionRes	手势贴纸
	makeupRes	美妆贴纸
	segmentMotionRes	背景分割贴纸
unity	aar	unity 项目需要使用的桥接 aar
unity	module	桥接 aar 的原工程



Windows

最近更新时间: 2024-03-20 17:48:51

集成准备

开发者环境要求

- •开发工具:VS2019(建议)
- 运行环境依赖: QT5.15.2 + QT VS 插件

硬件环境要求

- 显卡支持 OpenGL 4.1 或者以上
- 系统要求 windows8 或者以上
- CPU 支持 AVX 和 AVX2 指令

配置开发者环境

1. 安装 QT 插件。QT 插件安装成功后界面如下:



2. 配置 QT 插件版本,选择对应的 QT 版本。

选项								?	;	×
搜索选项(Ctrl+E)	P	Default	Version	Host	Path	Compiler				
▷ 环境 ▷ 顶日和解决方案		-	5.13.0_msvc2017_64	Windows ~	D:\Qt\5.13.0\msvc2017_64	msvc				
 ▶ J Lft (1) ▶ Lft (1) ▶ J Lft (1) ▶ Stripped and (1) ▶ Wat (1) ▶ Wat (1) ▶ Wat (1) ▶ CMake ▶ IntelliCode ▶ Invoit (1) ▶ NuGet 包管理器 ▲ Qt General ▶ Vest (1) ▶ Vest (1) ▶ Vest (1) ▶ West (1)<td></td><td></td><td><add new="" qt="" version=""></add></td><td></td><td></td><td></td><td>御定</td><td></td><td>取消</td><td></td>			<add new="" qt="" version=""></add>				御定		取消	

3. 配置工程依赖的 QT 组件。

腾讯云

QtWidgetsApplication 属性页				?	×
配置(C): 活动(Release)	→ 平台(巴):	活动(x64)	~	配置管理器(<u>0</u>)
▲配置属性 ^	Qt VS Project Format Versio	Dr. Version 3.4			^
常规	Qt Installation	5.13.0 msvc2017 64	11111		
高级	Qt Modules	core; opengl; gui; widgets			
调试	Build Config	Release			
VC++ 目录	Run Deployment Tool	否			
Qt Project Settings	Design Time Build	Run Qt Tools			
Qt Meta-Object Compiler	✓ Paths				
DQt Resource Compiler	Qt Binaries	bin			
▷ Qt User Interface Compile	Ot Library Executables	bin			
▷ C/C++	Additional Ot header searc	h paths			
▲链接器	Additional Ot library search	pat			
常规	✓ gmake				
输入	Template	MSBuild			
清单文件	Additional Command Argui	men			
调试	Additional Project Settings				
系统	✓ QML				
优化	Enable OML Debugging				
嵌入的 IDL	✓ Qt Additional Compiler O	ptions			
Windows 元数据	Additional Options	-Zc:rvalueCast -Zc:inline -Zc:strid	tStrings -Ze	c:throwingNev	N -Z
高级	Excluded Options		5	5	
所有选项	Show in Build Log				~
命令行	Ot VE Project Format Varsia				
▶清单工具	Qt vs Project Format version	I Nt Visual Studio project			
▶ YMI 立档生品器	version of the format of this C	t visual studio project.			
		TA	÷ 1		
		14月			

4. 添加头文件和 Xmagic 的 dll 库。

n.

置(C): 活动(Release)	~ 平台(P): 活动(x64)	~ 配置管理器(<u>O</u>)
配置属性 ^ 常規 高级 调试 VC++目录 Qt Project Settings ▷ Qt Meta-Object Compiler ▷ Qt Resource Compiler ▷ Qt User Interface Compiler ▷ Qt User Interface Compile ▷ C/C++ △ 链接器 常規 输入 清単文件 调试 系统 优化 嵌入的 IDL Windows 元数据 高级 所有选项 命令行 ▷ 清単工具	附加依赖项 忽略所有默认库 忽略特定默认库 模块定义文件 将模块添加到程序集 嵌入托管资源文件 强制符号引用 延迟加载的 DLL 程序集链接资源 附加依赖项 指定要添加到链接命令行的附加项。[例如 kern	el32.lib] %(AdditionalDependencies);\$(Qt_LIBS_)

接口概览

Windows 端 SDK 的集成涉及到的接口如下:

接口名

描述功能

代码示例



setTELicense	鉴权接口	示例
createXmagic	Xmagic 创建接口	示例
destroyXmagic	Xmagic 销毁接口	示例
updateProperty	设置美颜参数(如美颜、动效、化妆等)	
process	处理美颜接口,返回对应的美颜处理数据(如像素、纹理)	示例
setSegmentBg	设置自定义人像分割	
setRenderSize	重置输入数据大小	-
onPasue	暂停美颜	-
onResume	开始美颜	-

接口使用说明

鉴权接口

XMAGIC_API void setTELicense(const char* url, const char* key, int (*TELicenseCallback)(int, const char*));

参数含义:

参数	含义
url	对应的鉴权证书链接(在后台获取)
key	对应的鉴权证书 key(在后台获取)
TELicenseCallback	鉴权成功或者失败的回调

鉴权回调 code 的含义:

错误码	说明
0	成功。Success
-1	输入参数无效,例如 URL 或 KEY 为空
-3	下载环节失败,请检查网络设置
-4	从本地读取的 TE 授权信息为空,可能是 IO 失败引起
-5	读取 VCUBE TEMP License文件内容为空,可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理
-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的TE授权信息为空。请联系腾讯云团队处理
-11	把 TE 授权信息写到本地文件时失败,可能是 IO 失败引起
-12	下载失败,解析本地 asset 页失败
-13	鉴权失败

其他

请联系腾讯云团队处理

创建 Xmagic 接口

腾讯云

XMAGIC_API IXmagic* createXmagic(std::string& resDir, int width, int height);

参数含义:

参数	含义
resDir	对应的资源文件(SDK 依赖的资源路径)
width	处理数据宽
height	处理数据高

销毁 Xmagic 接口

XMAGIC_API void destroyXmagic(IXmagic** xmagic);

设置美颜参数

virtual void updateProperty(XmagicProperty* property) = 0;

参数含义:

参数	说明
category	 美颜类型,包括以下几种: 美颜:BEAUTY = 0 美体:BODY_BEAUTY 滤镜:LUT 动效:MOTION 人像分割:SEGMENTATION 化妆:MAKEUP
resPath	对应设置资源的地址(如动效、化妆、人像分割等)
effKey	美颜的 key
effValue	美颜的值
isAuth	默认 true

处理美颜(像素)

virtual void process(YTImagePixelData* srcImage, YTImagePixelData* dstImage) = 0;

参数说明:

参数	说明
srcImage	输入的美颜数据
dstImage	输出的美颜数据

YTImagePixelData 结构说明:



YTImagePixelData { /// 字段含义: 像素格式 PixelFormat pixelFormat;
/// 字段含义 uint8_t* data;
/// 字段含义: 数据的长度,单位是字节 int32_t length;
/// 字段含义: 宽度 int32_t width;
/// 字段含义: 高度 int32_t height;
YTImagePixelData() : pixelFormat(PixelFormat::PixelFormatRGBA32), data(nullptr), length(0),), height(0) {}

处理美颜纹理,返回对应的处理纹理

virtual int process(int textureId, int width, int height) = 0;

参数说明:

参数	说明
textureId	输入的纹理 ID
width	纹理宽度
height	纹理高度

设置自定义人像分割

virtual void setSegmentBg(std::string &segmentBgName ,int segmentBgType ,int timeOffset) = 0;

参数说明

参数	说明
segmentBgName	自定义人像分割路径
segmentBgType	设置背影类型(0为图片,1为视频)
timeOffset	如果为视频,设置视频播放时长

重置输入大小

virtual void setRenderSize(int width, int height) = 0;

参数说明:

参数	说明
width	重置后的宽度
height	重置后的高度



暂停美颜

virtual void onPasue() = 0;

开始美颜

virtual void onResume() = 0;

代码示例

 说明 以下仅为部分示例代码。

鉴权接口

```
//鉴权接口
auto respCallback = [](
    int ret, const char* data) -> int {
        int retCode = ret;
        const char* msg = data;
        return 0;
    };
setTELicense("url", "key", respCallback);
```

创建 Xmaigc

```
//创建Xmaigc
std::string exeFilePath = "资源位置";
IXmagic* xmagic = createXmagic(exeDirectory, 720, 1280);
```

销毁 Xmaigc

```
//销毁
if (xmagic) {
    destroyXmagic(&xmagic)
}
```

设置属性和输出

```
//设置属性和输出
    YTImagePixelData src,dst;
    src.width = 720;
    src.height = 1280;
    src.length = 4 * 720 * 1280;
    src.length = 4 * 720 * 1280;
    src.pixelFormat = PixelFormat::PixelFormatRGBA32;
    //uint8_t* rgbaBuffer = (uint8_t*)malloc(src.length);
    //int w = 0, h = 0, comp = 0;
    uint8_t* imageData;
    int w = 0, h = 0, comp = 0;
    stbi_set_flip_vertically_on_load(false); // SDK要求输入的图是倒置的,否则有问题
    imageData = stbi_load(path.c_str(), &w, &h, &comp, STBI_rgb_alpha);
    src.data = imageData;
```



dst.height = 1280; dst.height = 1280; dst.length = 4 * 720 * 1280; dst.pixelFormat = PixelFormat::PixelFormatRGBA32; uint8_t* rgbaBuffer = (uint8_t*)malloc(dst.length);//这个要释放 dst.data = rgbaBuffer; std::string bgPath = exeDirectory + "bg.jpg"; xmagic->setSegmentBg(bgPath, 0, 0); XmagicProperty property; property.category = Category::SEGMENTATION; property.category = Category : "measuretMetriceDec burdle(wideo events corrected);

for(int i =0; i < 2; i++){
 xmagic->process(&src, &dst);

版权所有:腾讯云计算(北京)有限责任公司



Mac

最近更新时间: 2024-09-25 10:32:11

集成准备

开发者环境要求

- 开发工具 XCode 11 及以上: App Store 或单击 下载 Demo。
- 建议运行环境:
 - 设备要求: macOS 系统设备。
 - 系统要求: macOS 10.15 及以上。

导入SDK

您可以先将 SDK 下载到本地,再将其手动导入到您当前的项目中。

下载SDK并手动导入

- 1. 下载并解压 SDK 和美颜资源,Libs 文件夹里面是 sdk 、 resources 文件夹里面是美颜的 bundle 资源。
- 2. 打开您的 Xcode 工程项目,把Libs文件夹里面的 Xmagic_Mac.framework、libLightSDKMac、libpag.framework 添加到实际工程中,选择要运 行的 target,选中 General 项,单击 Frameworks,Libraries,and Embedded Content 项展开,将 libLightSDKMac和libpag.framework 的 Embed 改成 Embed&Sign 。

	General Signing & Capabilities Resource Tags Info Build Settings Build Phases Build Rules
PROJECT	
🖾 BeautyDemo	✓ Deployment Info
TARGETS	Deployment Target 10.10
BeautyDemo	Main Interface Main
	✓ App Icons
	Source Appicon
	Include all app icon assets
	✓ Supported Intents
	Class Name
	Add intents eligible for in-app handling here
	 Frameworks, Libraries, and Embedded Content
	Name Embed
	Amagic_Mac.framework Do Not Embed 🗘
	✓ Development Assets

- 3. 选择对应的 target, 选中 Build Settings, 找到 Other Linker Flags 添加 "-Istdc++"。
- 4. 在项目里找到 xxx(项目名).entitlements 文件, 增加如下 key-value:

key>com.apple.security.cs.allow-dyld-environment-variables</key> <true/>



- 5. 把 resources 夹里面的美颜资源添加到实际工程中。
- 6. 将 Bundle ID 修改成与申请的测试授权一致。

动态下载集成

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。

动态下载的详细指引,请参见 SDK 包体瘦身(iOS)。

配置权限

在 Info.plist 文件中添加相应权限的说明,否则程序在 iOS 10 系统上会出现崩溃。请在 Privacy – Camera Usage Description 中开启相机权限,允许 App 使用相机。Privacy – Microphone Usage Description 开启麦克风权限,允许使用麦克风。在 target-->signing & Capabilities--> Hardened Runtime 勾选 **Audio Input、Camera**。

集成步骤

步骤一: 鉴权

1. 申请授权,得到 LicenseURL 和 LicenseKEY,请参见 License 指引。

△ 注意

```
正常情况下,只要 App 成功联网一次,就能完成鉴权流程,因此您不需要把 License 文件放到工程的工程目录里。但是如果您的 App 在从未联网的情况下也需要使用 SDK 相关功能,那么您可以把 License 文件下载下来放到工程目录,作为保底方案,此时 License 文件名必须是
v_cube.license 。
```

 在相关业务模块的初始化代码中设置 URL 和 KEY, 触发 License 下载,避免在使用前才临时去下载。也可以在 AppDelegate 的 didFinishLaunchingWithOptions 方法里触发下载。其中, LicenseURL 和 LicenseKey 是控制台绑定 License 时生成的授权信息。

```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authresult, NSString *
_Nonnull errorMsg) {
    if (authresult == TELicenseCheckOk) {
        NSLog(@"鉴权成功");
    } else {
        NSLog(@"鉴权失败");
    }
};
```

鉴权 errorCode 说明:

错误码	说明
0	成功。Success
-1	输入参数无效,例如 URL 或 KEY 为空
-3	下载环节失败,请检查网络设置
-4	从本地读取的 TE 授权信息为空,可能是 IO 失败引起
-5	读取 VCUBE TEMP License 文件内容为空,可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理
-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的TE授权信息为空。请联系腾讯云团队处理
-11	把 TE 授权信息写到本地文件时失败,可能是IO失败引起



-12	下载失败,解析本地 asset 也失败
-13	鉴权失败
其他	请联系腾讯云团队处理

步骤二:加载 SDK (XMagic.framework)

使用腾讯特效 SDK 生命周期大致如下:

1. 加载美颜相关资源。

NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle", @"root_path":[[NSBundle mainBundle] bundlePath] };

2. 初始化腾讯特效 SDK。

```
initWithRenderSize:assetsDict: (XMagic)
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDict];
```

3. 腾讯特效 SDK 处理每帧数据并返回相应处理结果。

```
process: (XMagic)
```

// demo**层**

```
// 在摄像头回调传入帧数据
```

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:
(CMSampleBufferRef)sampleBuffer fromConnection:(AVCaptureConnection *)connectior
```

```
// 获取原始数据,处理每帧的渲染信息
```

(void)mycaptureOutput: (AVCaptureOutput *)captureOutput didOutputSampleBuffer:
 (CMSampleBufferRef)inputSampleBuffer fromConnection: (AVCaptureConnection *)connection
 originImageProcess: (BOOL) originImageProcess;

```
// 使用CPU处理数据
```

- (YTProcessOutput*)processDataWithCpuFuc:(CMSampleBufferRef)inputSampleBuffer;
- // 使用GPU处理数据
- (YTProcessOutput*)processDataWithGpuFuc:(CMSampleBufferRef)inputSampleBuffer;
- // sdk**接口调用**
- // 腾讯特效 SDK处理数据接口
- /// @param input 输入处理数据信息
- /// @return 输出处理后的数据信息
- (YTProcessOutput* _Nonnull)process:(YTProcessInput * _Nonnull)input;

4. 给 SDK 设置特效。







- 5. 释放腾讯特效 SDK。
 - deinit (XMagic) // **在需要释放**SDK**资源的地方调用**
 - [self.beautyKit deinit]



iOS

最近更新时间: 2024-09-29 11:04:41

集成准备

开发者环境要求

- 开发工具 XCode 11 以上: App Store 或单击 下载地址。
- 建议运行环境:
- 设备要求: iPhone 5 及以上; iPhone 6 及以下前置摄像头最多支持到 720p,不支持 1080p。
- 系统要求: iOS 12.2 及以上。

腾讯特效SDK集成使用流程



导入 SDK

您可以选择使用 CocoaPods 方案,或者先将 SDK 下载到本地,再将其手动导入到您当前的项目中。

使用 CocoaPods

1. 安装 CocoaPods

在终端窗口中输入如下命令(需要提前在 Mac 中安装 Ruby 环境):

sudo gem install cocoapods

2. 创建 Podfile 文件

进入项目所在路径,输入以下命令行之后项目路径下会出现一个 Podfile 文件。

pod init

3. 编辑 Podfile 文件



- XMagic 版本在3.0.1以前: 根据您的项目需要选择合适的版本,并编辑 Podfile 文件:
 - XMagic 普通版

请按如下方式编辑 Podfile文件:

platform :10s, '8. target 'App' do pod 'XMagic' end

○ XMagic 精简版

安装包体积比普通版小,但仅支持基础版 A1-00、基础版 A1-01、高级版 S1-00,请按如下方式编辑 Podfile 文件:

platform :ios, '8.0'
target 'App' do
pod 'XMagic_Smart'
opd

○ XMagic 版本在3.0.1及以后:

根据您的项目套餐选择合适的版本,并编辑 Podfile 文件:

请根据您的套餐 pod install 对应的库 # 例如: 如果您的套餐是 all 类型,那么只需要 pod 'TencentEffect_All' # 例如: 如果您的套餐是 s1-04 类型,那么只需要 pod 'TencentEffect_S1-04'
<pre>pod 'TencentEffect_All'</pre>

4. 更新并安装 SDK

在终端窗口中输入如下命令以更新本地库文件,并安装 SDK:

pod install

pod 命令执行完后,会生成集成了 SDK 的 .xcworkspace 后缀的工程文件,双击打开即可。

5. 添加美颜资源到实际项目工程中

下载并解压对应套餐的 SDK 和美颜资源,将 resources/motionRes 文件夹下bundle 资源添加到实际工程中。

在 Build Settings 中的Other Linker Flags添加 - ObjC 。

6. 将 Bundle Identifier 修改成与申请的测试授权一致。



下载 SDK 并手动导入

1. 下载并解压 SDK 和美颜资源,frameworks 文件夹里面是sdk、resources 文件夹里面是美颜的bundle资源。

2. SDK版本在2.5.1以前:

打开您的 Xcode 工程项目,把 frameworks 文件夹里面的 framework 添加到实际工程中,选择要运行的 target,选中 General 项,单击 Frameworks,Libraries,and Embedded Content 项展开,单击底下的 "+"号图标去添加依赖库。依次添加下载的 XMagic.framework 、 YTCommonXMagic.framework 、 libpag.framework 及其所需依赖库 MetalPerformanceShaders.framework 、

CoreTelephony.framework 、 JavaScriptCore.framework 、 VideoToolbox.framework 、 libc++.tbd **, 根据需要添加其它工具库** Masonry.framework (控件布局库)、 SSZipArchive (文件解压库)。

	General Signi	ng & Capabilities Resource Tags Info Bui	ld Settings Build	d Phases Build Rules	
PROJECT	Supported Intents				
🔝 BeautyDemo					
TARGETS		Class Name Authentication			
< BeautyDemo		Add intents eligible	for in-app handling	a here	
		+ -			
	Frameworks Librarias and Embedded Content				
	• Frameworks, Libraries, and Embedded Content				
		Name	Filters	Embed	
		CoreTelephony.framework	Always Used	⊜∽ Do Not Embed \$	
		🚘 JavaScriptCore.framework	Always Used	⊚∽ Do Not Embed \$	
		E libc++.tbd	Always Used	(m) ~	
		🚔 libpag.framework	Always Used	'. Embed & Sign ≎	
		🚔 Masonry.framework	iOS	🗐 - Embed & Sign 🗘	
		🖴 MetalPerformanceShaders.framework	Always Used	⊜∽ Do Not Embed \$	
		SSZipArchive.framework	Always Used		
		📥 VideoToolbox.framework	Always Used		
		 VideoToolbox.framework XMagic.framework 	Always Used Always Used	 S > Do Not Embed \$ Do Not Embed \$ 	
		 VideoToolbox.framework XMagic.framework YTCommonXMagic.framework 	Always Used Always Used Always Used	 (≂) ~ Do Not Embed \$ (≂) ~ Do Not Embed \$ (≂) ~ Do Not Embed \$ 	
		VideoToolbox.framework XMagic.framework YTCommonXMagic.framework	Always Used Always Used Always Used	 (₹) ~ Do Not Embed \$ (₹) ~ Do Not Embed \$ (₹) ~ Do Not Embed \$ 	

及以后:

打开您的 Xcode 工程项目,把 frameworks 文件夹里面的 framework 添加到实际工程中,选择要运行的 target,选中 General 项,单击 Frameworks,Libraries,and Embedded Content 项展开,单击底下的"+"号图标去添加依赖库。依次添加下载的 XMagic.framework 、

YTCommonXMagic.framework 、 libpag.framework 、 Audio2Exp.framework 、

TEFFmpeg.framework(version3.0.0以后,改名为: TECodec.framework)及其所需依赖库 MetalPerformanceShaders.framework、 CoreTelephony.framework、JavaScriptCore.framework、VideoToolbox.framework、libc++.tbd,根据需要添加其它工具库 Masonry.framework(控件布局库)、SSZipArchive(文件解压库)。

General	igning & Capabilities Resource Tags Info Buil	ld Settings Buil	ld Phases Build Rules			
PROJECT						
🛃 BeautyDemo	Add intents eligible for in-app handling here					
TARGETS	± _					
BeautyDemo	Т					
Frameworks, L	ibraries, and Embedded Content					
	Name	Filters	Embed			
	🚔 AFNetworking.framework	iOS	⊜∽ Embed & Sign \$			
	🚔 Audio2Exp.framework	iOS	⊜∽ Embed & Sign \$			
	🚘 CoreTelephony.framework	Always Used				
	JavaScriptCore.framework	Always Used	⇒ Do Not Embed \$			
	🗊 libc++.tbd	Always Used	() V			
	🚘 libpag.framework	Always Used	⊜∽ Embed & Sign \$			
	🚘 Masonry.framework	iOS	⊜∽ Embed & Sign \$			
	🚘 MetalPerformanceShaders.framework	Always Used				
	🚔 SSZipArchive.framework	iOS	⊜∽ Embed & Sign 🗘			
	🚘 TEFFmpeg.framework	iOS	⊜∽ Embed & Sign \$			
	🚔 VideoToolbox.framework	Always Used				
	🚔 XMagic.framework	Always Used	⊜∽ Do Not Embed \$			
	🚘 YTCommonXMagic.framework	Always Used	⊜∽ Embed & Sign \$			
	+ -					
Development	Assets					

3. 把 resources 夹里面的美颜资源添加到实际工程中。



- 4. 在 Build Settings 中的Other Linker Flags添加 ObjC 。
- 5. 将 Bundle Identifier 修改成与申请的测试授权一致。

动态下载集成

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。

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。动态下载的详细指引,请参见 SDK 包体瘦身(iOS)。

配置权限

在 Info.plist 文件中添加相应权限的说明,否则程序在 iOS 10 系统上会出现崩溃。请在 Privacy - Camera Usage Description 中开启相机权限,允许 App 使用相机。

集成步骤

步骤一: 鉴权

- 1. 申请授权,得到 LicenseURL 和 LicenseKEY,请参见 License 指引。
- 在相关业务模块的初始化代码中设置 URL 和 KEY,触发 license 下载,避免在使用前才临时去下载。也可以在 AppDelegate 的 didFinishLaunchingWithOptions 方法里触发下载。其中,LicenseURL 和 LicenseKey 是控制台绑定 License 时生成的授权信息。
 SDK 版本在2.5.1以前, TELicenseCheck.h 在 XMagic.framework 里面; SDK版本在2.5.1及以后, TELicenseCheck.h 在 YTCommonXMagic.framework 里面。

```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authresult, NSString *
_Nonnull errorMsg) {
    if (authresult == TELicenseCheckOk) {
        NSLog(@"鉴权成功");
    } else {
        NSLog(@"鉴权失败");
    }
}];
```

鉴权 errorCode 说明:

错误码	说明
0	成功。Success
-1	输入参数无效,例如 URL 或 KEY 为空
-3	下载环节失败,请检查网络设置
-4	从本地读取的 TE 授权信息为空,可能是 IO 失败引起
-5	读取 VCUBE TEMP License文件内容为空,可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理
-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的TE授权信息为空。请联系腾讯云团队处理
-11	把 TE 授权信息写到本地文件时失败,可能是IO失败引起
-12	下载失败,解析本地 asset 也失败



-13	鉴权失败
其他	请联系腾讯云团队处理

步骤二:加载 SDK (XMagic.framework)

使用腾讯特效 SDK 生命周期大致如下:

1. 加载美颜相关资源。

```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
@"root_path":[[NSBundle mainBundle] bundlePath] // LightCore.bundle所在的目录。
};
```

2. 初始化腾讯特效 SDK。

```
/**
previewSize:视图的宽高
assetsDict: 上一步配置的LightCore.bundle及其路径
*/
self.xMagicApi = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDict];
```

3. 腾讯特效 SDK 处理每帧数据并返回相应处理结果。



ſ

美颜 process 方法及更多 API 详情见 API 文档。

4. 释放腾讯特效 SDK。

// **在需要释放**SDK**资源的地方调用** [self.xMagicApi deinit]

说明:
 完成上述步骤后,用户即可根据自己的实际需求控制展示时机以及其他设备相关环境。

常见问题

问题1:编译报错: "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):"?

1. 前往 File > Project settings > Build System 选择 Legacy Build System。

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

问题2: 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'..."?

将 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,所这里有这个问题注意下即可。
```

问题3: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。

问题4: iOS Demo 编译, 生成 dSYM 时报错?

PhaseScriptExecution CMake\ PostBuild\ Rules build/XMagicDemo.build/Debugiphoneos/XMagicDemo.build/Script-81731F743E244CF2B089C1BF.sh cd /Users/zhenli/Downloads/xmagic_s106/build/XMagicDemo.build/Debugiphoneos/XMagicDemo.build/Script-81731F743E244CF2B089C1BF.sh Command /bin/sh failed with exit code 1 • 问题原因: libpag.framework和Masonary.framework 重签名失败。 • 解决方法: 1.1 打开 demo/copy_framework.sh 1.2 用下述命令查看本机 cmake 的路径,将 \$(which cmake) 改为本地 cmake 绝对路径。

hich cmak

1.3 用自己的签名替换所有 Apple Development:



原子能力集成指引 手势识别

最近更新时间:2024-08-0316:32:21

功能说明

输入相机的 openGL 纹理,实时输出手势检测数据。您可以利用这些数据做进一步的开发。

Android 接口说明

Android 集成指引

Android 集成腾讯特效 SDK,具体请参见 独立集成腾讯特效。

Android 接口调用

1. 打开手势检测功能开关(XmagicApi.java)。

public void setFeatureEnableDisable(String featureName, boolean enable);

featureName填 XmagicConstant.FeatureName.HAND_DETECT , enable填 true 。

2. 设置数据回调(XmagicApi.java)

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)
public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
    void onHandDataUpdated(List<TEHandData> handDataList);
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);
    void onAIDataUpdated(String data);
}
```

onAIDataUpdated 返回 JSON 结构的 string 数据。

iOS 接口说明

iOS 集成指引

iOS 集成腾讯特效 SDK,具体请参见 独立集成腾讯特效。

iOS 接口调用

1. 打开手势检测功能开关(Xmagic.h)。

(void)setFeatureEnableDisable:(NSString *_Nonnull)featureName enable:(BOOL)enable;

featureName填 HAND_DETECT (可在 TEDefine.h 中引入), enable填 true。

```
2. 设置数据回调(Xmagic.h)
```





eventDict[@"ai_info"] 即为返回的 JSON 结构的 string 数据。

回调 JSON 数据说明

在回调的 JSON 数据中, "hand_info" 里是手势相关的数据,格式如下所示。

```
"hand_info": {
    "gesture": "PAPER",
    "hand_point_2d": [180.71888732910156, 569.2958984375, ... , 353.8714294433594, 836.246826171875]
}
```

hand_info 中各字段说明如下:

字段	含义
gesture	手势类型名称
hand_point_2d	捕捉到手势的数据信息

目前支持以下手势:

序号	手势	类型名称	示例图
1	手势1	ONE	
2	手势2	SCISSOR	
3	手势3	THREE	



4	手势4	FOUR	
5	手势5(open)	PAPER	
6	手势6	SIX	
7	手势8	EIGHT	
8	比心	HEART	
9	拳头	FIST	



10	我爱你	LOVE	
11	点赞	LIKE	
12	ОК	ОК	
13	摇滚手势	ROCK	
14	托	LIFT	

如果为不可识别的手势,则类型名称为 OTHER 。



人脸点位

最近更新时间: 2023-06-13 17:38:52

人脸检测(识别人脸出框、多人脸、面部遮挡),256个面部关键点位识别与输出。

人脸256点对应索引图



iOS 接口说明

iOS 集成指引

iOS 集成 SDK 指引,请参见 独立集成腾讯特效 。

Xmagic 接口回调注册

```
/// @brief SDK事件监听接口
/// @param listener 事件监听器回调,主要分为AI事件,Tips提示事件,Asset事件
```

YTSDKEventListener 回调说明

```
#pragma mark - 事件回调接口
/// @brief SDK内部事件回调接口
@protocol YTSDKEventListener <NSObject>
/// @brief YTDataUpdate事件回调
/// @param event NSString*格式的回调
- (void) onYTDataEvent: (id _Nonnull) event;
/// @brief AI事件回调
/// @param event dict格式的回调
- (void) onAIEvent: (id _Nonnull) event;
/// @brief 提示事件回调
/// @param event dict格式的回调
- (void) onTipsEvent: (id _Nonnull) event;
/// @brief 资源包事件回调
/// @param event string格式的回调
- (void) onAssetEvent: (id _Nonnull) event;
@end
```

2.6.0及之前版本 设置回调成功后,每一帧人脸事件会回调:

(void)onYTDataEvent:(id _Nonnull)event;



3.0.0版本 设置回调成功后,每一帧人脸事件会回调:

```
- (void)onAlEvent:(id _Nonnull)event;
//在onAlEvent方法中可通过下边方法可以获取到数据
NSDictionary *eventDict = (NSDictionary *)event;
if (eventDict[@"ai_info"] != nil) {
    NSLog(@"ai_info %@",eventDict[@"ai_info"]);
    }
```

回调 data 是一个 JSON 格式数据,具体含义如下(256点对应上图的位置):

- (void)onYTDataEvent:(id _Nonnull)event;

Android 接口说明

Android 集成指引

Android 集成 SDK 指引,具体请参见 独立集成腾讯特效。

Xmagic 接口回调注册

设置人脸点位信息等数据回调。 2.6.0及之前版本使用如下方法

2.0.0及之前版举使用如下方法

```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
设置人脸信息等数据回调
public interface XmagicYTDataListener {
    void onYTDataUpdate(String data)
}
3.0.0版本使用如下方法
void setALDataListener(XmagicApi.OnALDataListener aiDataListener)
```



onYTDataUpdate 和 onAlDataUpdated 返回 JSON string 结构,最多返回5个人脸信息:



字段含义

字段	类型	值域	说明
trace_id	int	[1,INF)	人脸 ID,连续取流过程中,ID 相同的可以认为是同一张人 脸。
face_256_point	floa t	[0,screenWidth] 或 [0,screenHeight]	共512个数,人脸256个关键点,屏幕左上角为(0,0)。
face_256_visible	floa t	[0,1]	人脸256关键点可见度。
out_of_screen	boo I	true/false	人脸是否出框。
left_eye_high_vis_ratio	floa t	[0,1]	左眼高可见度点位占比。
right_eye_high_vis_ratio	floa t	[0,1]	右眼高可见度点位占比。
left_eyebrow_high_vis_rati o	floa t	[0,1]	左眉高可见度点位占比。
right_eyebrow_high_vis_ra tio	floa t	[0,1]	右眉高可见度点位占比。
mouth_high_vis_ratio	floa t	[0,1]	嘴高可见度点位占比。

参数

参数	含义
XmagicApi.XmagicYTDataListener ytDataListener	回调函数实现类。



人像分割

最近更新时间:2024-08-0117:38:41

在直播、会议等场景实现虚拟背景,实时精准分割,支持自定义背景:



iOS 接口使用

iOS 集成指引

iOS 集成 SDK 指引,请参见 独立集成腾讯特效 。

虚拟背景设置

NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotionRes"
ofType:@"bundle"];//这里是segmentMotionRes文件夹的绝对路径
NSString *propertyType = @"motion"; //配置美颜的效果类型,这里以分割为例
NSString *propertyName = @"video_segmentation_blur_75"; //配置美颜的名称,这里以背景模糊-强为例
NSString *propertyValue = motionSegResPath; //配置动效的路径
NSDictionary *dic = @{@"bgName":@"BgSegmentation.bg.png", @"bgType":@0, @"timeOffset":
@0},@"icon":@"segmentation.linjian.png"};//配置预留字段
[self.beautyKit configPropertyWithType:propertyType withName:propertyName withData:propertyValue
withExtraInfo:dic];

自定义背景设置

NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotionRes" ofType:@"bundle"];//这里是segmentMotionRes文件夹的绝对路径 NSString *propertyType = @"motion"; //配置美颜的效果类型,这里以分割为例 NSString *propertyName = @"video_empty_segmentation"; //配置美颜的名称,这里以自定义背景为例 NSString *propertyValue = motionSegResPath; //配置动效的路径 NSString *imagePath = @"/var/mobile/Containers/Data/Application/06B00BBC-9060-450F-8D3A-F6028D185682/Documents/MediaFile/image.png"; //自定义背景图片的绝对路径。如果自定义背景选择的是视频,需要对视频进行压 缩转码处理,使用压缩转码处理后的绝对路径 int bgType = 0;//自定义背景的类型。 0表示图片,1表示视频 int timeOffset = 0; //时长。图片背景时,为0; 视频背景时为视频的时长



NSDictionary *dic = @{@"bgName":imagePath, @"bgType":@(bgType), @"timeOffset": @(timeOffset)},@"icon":@"segmentation.linjian.png"};//配置预留字段 [self.beautyKit configPropertyWithType:propertyType withName:propertyName withData:propertyValue withExtraInfo:dic];

Android 接口使用

Android 集成指引

Android 集成 SDK 指引,请参见 独立集成腾讯特效 。

设置属性接口

void updateProperty(XmagicProperty<?> p)

分割参数:

属性字段	说明
category	Category.SEGMENTATION
ID	 资源文件夹名称,必填示例: video_segmentation_blur_45 "无" ID 为 XmagicProperty.ID_NONE 自定义分割 ID 值必须使用: XmagicConstant.SegmentationId.CUSTOM_SEG_ID
resPath	必填,自定义分割素材的全路径
effkey	null(自定义背景除外),自定义背景的值为选择的资源路径
effValue	null

虚拟背景设置

//**初始化**XmagicProperty

- XmagicProperty xmagicProperty = new
- XmagicProperty(Category.SEGMENTATION, "video_segmentation_blur_45", resPath, null, null);

//设置属性

xmagicApi.updateProperty(xmagicProperty)

自定义背景设置

XmagicProperty xmagicProperty = new

XmagicProperty(Category.SEGMENTATION,XmagicConstant.SegmentationId.CUSTOM_SEG_ID,resPath,null,null);

xmagicApi.updateProperty(xmagicProperty)



人脸属性



最近更新时间: 2023-06-14 18:03:41

功能说明

该接口可通过上传包含人脸的照片获取人物面部特征信息,包括眼睛、眉毛、发型、肤色、性别、年龄等。该接口需要联网环境下才可通过 SDK 把照片上传到 Server 端进行解析。

集成指引

首先需要集成腾讯特效 SDK,具体请参见 独立集成腾讯特效 。

接口说明

接口所在类: XMagic.h

+ (void)getFaceFeatureFromPhoto:(UIImage * _Nullable)image success:(nullable void (^)
(FaceDetailAttributesinfo *_Nullable faceInfo))success failure:(nullable void (^)(NSInteger errorCode,
NSString *_Nullable msg))failure;

- 参数 image:请尽量让人脸位于画面中间,建议画面中只包含一个人脸。如果有多个人脸,SDK会随机选择一个。建议照片的短边大于等于500px,尺寸过 小会影响识别效果。
- 参数 success、failure: 返回识别的结果。



• failure 回调:解析失败时会回调此接口,错误码如下。



• success 回调:解析成功时回调此接口,FaceDetailAttributesInfo 说明如下。





```
眉毛长短。0:短眉毛,1:长眉毛
```



帽子佩戴状态。0:不戴帽子,1:普通帽子,2:头盔,3:保安帽子 0:自然,1:高兴,2:惊讶,3:生气,4:悲伤,5:厌恶,6:害怕 眉毛信息 帽子信息



```
胡子信息。-1:没识别,0:无胡子,1:有胡子
```



Android

最近更新时间: 2023-06-14 18:03:41

功能说明

该接口可通过上传包含人脸的照片获取人物面部特征信息,包括眼睛、眉毛、发型、肤色、性别、年龄等。该接口需要联网环境下才可通过 SDK 把照片上传到 Server 端进行解析。

集成指引

首先需要集成腾讯特效 SDK,具体请参见 独立集成腾讯特效 。

接口说明

接口所在类: XMagicApi.java

public void getFaceFeatureFromPhoto(Bitmap bitmap, FaceFeatureListener listener);

- 参数 bitmap:请尽量让人脸位于画面中间,建议画面中只包含一个人脸。如果有多个人脸,SDK 会随机选择一个。建议照片的短边大于等于500px,尺寸 过小会影响识别效果。
- 参数 FaceFeatureListener,返回识别的结果。

```
public interface FaceFeatureListener {
    void onError(int errCode, String msg);
    void onFinish(FaceDetailAttributesInfo faceInfo);
}
```

• onError 回调:解析失败时会回调此接口,错误码如下。

```
public static final int ERROR_NO_AUTH = 1;//没有权限
public static final int ERROR_RES_INVALID = 5;//传入的Avatar素材路径无效
public static final int ERROR_PHOTO_INVALID = 10;//读取照片失败
public static final int ERROR_NETWORK_REQUEST_FAILED = 20;//网络请求失败
public static final int ERROR_DATA_PARSE_FAILED = 30;//网络返回数据解析失败
public static final int ERROR_ANALYZE_FAILED = 40;//人脸分析失败
public static final int ERROR_AVATAR_SOURCE_DATA_EMPTY = 50;//加载Avatar源数据失败
```

• onFinish 回调:解析成功时回调此接口,FaceDetailAttributesInfo 说明如下。

```
public static class FaceDetailAttributesInfo {
    public int age; //[0,100]
    public int emotion; //0: 自然, 1: 高兴, 2: 惊讶, 3: 生气, 4: 悲伤, 5: 厌恶, 6: 害怕
    public int emotion; //0: 自然, 1: 高兴, 2: 惊讶, 3: 生气, 4: 悲伤, 5: 厌恶, 6: 害怕
    public Eye eye; // 眼睛信息
    public Eyebrow eyebrow; // 眉毛信息
    public int gender; // 性别信息。-1:没识别, 0: 男性, 1: 女性。
    public int gender; // 性别信息。-1:没识别, 0: 男性, 1: 女性。
    public Hair hair; // 发型信息
    public int moustache; // 描子信息
    public int moustache; // 描子信息。-1:没识别, 0:无, 1有。
    public int nose; // 鼻子信息。-1:没识别, 0: 无胡子, 1: 有胡子。
    public int nose; // 鼻子信息。-1:没识别, 0: 无胡子, 1: 有胡子。
    public int shape; // 脑型信息。-1:没识别, 0: 方脸, 1: 三角脸, 2: 鹅蛋脸, 3: 心形脸, 4: 圆脸。
    public int skin; // 肤色信息。-1:没识别, 0: 黄色皮肤, 1: 棕色皮肤, 2: 黑色皮肤, 3: 白色皮肤。
    public int smile; //微笑程度, [0,100]。
}
public static class Eye {
    public int eyelidType; // 识别是否双眼皮。-1:没识别, 0: 无, 1: 有。
    public int eyeSize; // 眼睛大小。-1:没识别, 0: 小眼睛, 1: 普通眼睛, 2: 大眼睛。
```



```
public int glass; // 识别是否佩戴跟镜。-1:没识别,0:无服镜,1:普通服镜,2:墨镜
public int eyeOpen; // 识别跟鼱的睁开、闭合状态。-1:没识别,0: 睁开,1: 闭跟
}
public static class Eyebrow {
    public int eyebrowLength; //眉毛长短。0: 短眉毛,1: 长眉毛。
    public int eyebrowDensity; //眉毛浓密。0: 淡眉,1: 浓眉。
    public int eyebrowCurve; // 眉毛弯曲。0: 不弯,1: 弯眉。
}
public static class Hair {
    public int length; //头发长度信息。0: 光头,1: 短发,2: 中发,3: 长发,4: 绑发。
    public int bang; //刘海信息。0: 无刘海,1: 有刘海。
    public int color; //头发颜色信息。0: 黑色,1: 金色,2: 棕色,3: 灰白色。
}
public static class Hat {
    public int style; //帽子佩戴状态信息。0: 不戴帽子,1: 普通帽子,2: 头盔,3: 保安帽。
    public int color; //帽子颜色.0: 不戴帽子,1: 红色系,2: 黄色系,3: 蓝色系,4: 黑色系,5: 灰白色系,6: 混色
系子。
}
```



人脸表情

```
iOS
```

最近更新时间: 2024-12-04 15:00:33

功能说明

输入相机的 openGL 纹理,实时输出人脸52表情 BlendShape 数据,遵循苹果 ARKit 规范,详情请参见 ARFaceAnchor 。您可以利用这些表情数据做一 进步的开发,例如传到 Unity 中驱动您的模型。

集成指引

iOS 集成 SDK 指引,具体请参见 独立集成腾讯特效 。

接口调用

1. 打开功能开关:

[self.beautyKit setFeatureEnableDisable:ANIMOJI_52_EXPRESSION enable:YES];

2. 设置人脸点位信息数据回调。

2.6.0及之前版本使用如下方法

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;
@implementation listener
- (void)onYTDataEvent:(id)event
{
    NSLog(@"YTData %@", event);
}
```

3.0.0版本使用如下方法

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;
- (void)onAIEvent:(id)event
{
    NSDictionary *eventDict = (NSDictionary *)event;
    if (eventDict[@"ai_info"] != nil) {
        NSLog(@"ai_info %@",eventDict[@"ai_info"]);
    }
}
```

onYTDataUpdate 返回 JSON string 结构,最多返回5个人脸信息: onAlEvent 中通过eventDict[@"ai_info"] 获取到的 JSON string 结构数据,最多返回5个人脸信息:





],

"out_of_screen":true,

字段含义

• trace_id: 人脸 ID,连续取流过程中, ID 相同的可以认为是同一张人脸。

• expression_weights: 实时表情 blendshape 数据,数组长度为52,每个数值取值范围为 0到1.0。{

"eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLookOutLeft","eyeLookUpLeft","eyeSquintLeft","eyeWideLeft","e yeBlinkRight","eyeLookDownRight","eyeLookInRight","eyeLookOutRight","eyeLookUpRight","eyeSquintRight","eyeWideRigh t","jawForward","jawLeft","jawRight","jawOpen","mouthClose","mouthFunnel","mouthPucker","mouthRight","mouthLeft"," mouthSmileLeft","mouthSmileRight","mouthFrownRight","mouthFrownLeft","mouthDimpleLeft","mouthDimpleRight","mouthSt retchLeft","mouthStretchRight","mouthRollLower","mouthRollUpper","mouthShrugLower","mouthShrugUpper","mouthPressLe ft","mouthPressRight","mouthLowerDownLeft","mouthLowerDownRight","mouthUpperUpLeft","mouthUpperUpRight","browDownL eft","browDownRight","browInnerUp","browOuterUpLeft","browOuterUpRight","cheekPuff","cheekSquintLeft","cheekSquint Right","noseSneerLeft","noseSneerRight","tongueOut"}

● 其他字段是 人脸信息,只有当您购买了相关 License 才有那些字段。如果您只想获取表情数据,请忽略那些字段。



Android

最近更新时间: 2024-12-04 15:00:33

功能说明

输入相机的 openGL 纹理,实时输出人脸52表情 BlendShape 数据,遵循苹果 ARKit 规范,详情请参见 ARFaceAnchor 。您可以利用这些表情数据做一 进步的开发,例如传到 Unity 中驱动您的模型。

集成指引

Android 集成 SDK 指引,具体请参见 独立集成腾讯特效 。

接口调用

1. 打开功能开关:

```
//XmagicApi.java
//featureName = XmagicConstant.FeatureName.ANIMOJI_52_EXPRESSION
public void setFeatureEnableDisable(String featureName, boolean enable);
```

2. 设置人脸点位信息数据回调。

2.6.0及之前版本使用如下方法

```
//XmagicApi.java
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
public interface XmagicYTDataListener {
  void onYTDataUpdate(String data)
}
```

3.0.0版本使用如下方法

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)
public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
    void onHandDataUpdated(List<TEHandData> handDataList);
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);
    void onAIDataUpdated(String data); //此方法是3.0.0版本新增方法,数据结构和之前XmagicYTDataListener接口的数
    据保持一致
}
```

onYTDataUpdate 和 onAIDataUpdated 返回 JSON string 结构,最多返回5个人脸信息:







字段含义

- trace_id: 人脸 ID,连续取流过程中, ID 相同的可以认为是同一张人脸。
- expression_weights: 实时表情blendshape数据,数组长度为52,每个数值取值范围为 0到1.0。{

"eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLookOutLeft","eyeLookUpLeft","eyeSquintLeft","eyeWideLeft","e yeBlinkRight","eyeLookDownRight","eyeLookInRight","eyeLookOutRight","eyeLookUpRight","eyeSquintRight","eyeWideRigh t","jawForward","jawLeft","jawRight","jawOpen","mouthClose","mouthFunnel","mouthPucker","mouthRight","mouthLeft"," mouthSmileLeft","mouthSmileRight","mouthFrownRight","mouthFrownLeft","mouthDimpleLeft","mouthDimpleRight","mouthSt retchLeft","mouthStretchRight","mouthRollLower","mouthRollUpper","mouthShrugLower","mouthShrugUpper","mouthPressLe ft","mouthPressRight","mouthLowerDownLeft","mouthLowerDownRight","mouthUpperUpLeft","mouthUpperUpRight","browDownL eft","browDownRight","browInnerUp","browOuterUpLeft","browOuterUpRight","cheekPuff","cheekSquintLeft","cheekSquintLeft","cheekSquintLeft","noseSneerLeft","noseSneerRight","tongueOut"}

• 其他字段是 人脸信息,只有当您购买了相关 License 才有那些字段。如果您只想获取表情数据,请忽略那些字段。



身体点位

iOS

最近更新时间: 2024-08-03 16:32:22

功能说明

输入相机的 openGL 纹理,实时输出身体3D数据。您可以利用这些3D数据做一进步的开发,例如传到 Unity 中驱动您的模型。

集成指引

首先需要集成腾讯特效SDK,具体请参见 独立集成腾讯特效 。

接口调用

1. 打开功能开关(XMagic.h)。



```
@implementation listener
- (void)onYTDataEvent:(id)event
{
     NSLog(@"YTData %@", event);
}
@end
```

onYTDataEvent 返回 JSON 结构的 string 数据,其示例如下:

- "face_info" 里是人脸相关的数据,与身体 3D 数据无关,可以忽略。
- "body_3d_info" 里各字段说明见 身体点位及数据说明。

3.0.0版本使用如下方法



eventDict[@"ai_info"] 返回 JSON 结构的 string 数据,其示例如下:

- \odot "face_info" 里是人脸相关的数据,与身体 3D 数据无关,可以忽略。
- "body_3d_info" 里各字段说明见 身体点位及数据说明。

身体点位及点位数据说明

相关说明请参见身体点位及数据说明。



Android

最近更新时间: 2023-05-18 14:58:06

功能说明

输入相机的 openGL 纹理,实时输出身体3D数据。您可以利用这些3D数据做一进步的开发,例如传到 Unity 中驱动您的模型。

集成指引

首先需要集成腾讯特效SDK,具体请参见 独立集成腾讯特效 。

接口调用

1. 打开功能开关(XmagicApi.java)。

oublic void setFeatureEnableDisable(String featureName, boolean enable);

featureName填 XmagicConstant.FeatureName.BODY_3D_POINT。

```
2. 设置数据回调(XmagicApi.java)。
```

2.6.0及之前版本使用如下方法



onYTDataUpdate 返回 JSON 结构的 string 数据,其示例如下:

- "face_info" 里是人脸相关的数据,与身体 3D 数据无关,可以忽略。
- "body_3d_info" 里各字段说明见下文。
- 3.0.0版本使用如下方法



onAlDataUpdated 返回 JSON 结构的 string 数据,其示例如下:

- "face_info" 里是人脸相关的数据,与身体 3D 数据无关,可以忽略。
- "body_3d_info" 里各字段说明见下文。

身体点位及点位数据说明

● 标准 SMPL 点位定义





● 标准 SMPLX 手部骨骼点位定义



SDK 输出的 JSON 数据示例如下:



-{
"face_info": [{ }],
<pre>"body_3d_info": {</pre>
"imageHeight": 652.
"imageWidth": 320.
"items" [{
"pose": [-0.014862474985420704, 0.72017294
"position_x": [150.62857055664062, 190.150
"position_y": [605.454833984375, 655.2125
"position_z": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
"rotation": [{
"data": [0.9990578889846802. 0.0258054
J,
"data": [0.9546145796775818, 0.263716
] }]
}]
}
}

body_3d_info 里各个字段说明如下:

- imageWidth, imageHeight: 输入给 SDK 的图像的宽高
- items: 数组,目前只有一个元素
- index:保留位,目前可以忽略它
- pose:
 - \odot [0,2]位置,人体位置,以相机为中心,人体根骨骼的3D位置 x、y、z
 - [3,12]位置,人体形态,10个float数,以标准 SMPL 的10套不同mesh为基底,组合得到人体型的估计
 - [13]位置,Focal_length,固定值为5000
 - [14,29]位置,OpenGL 投影矩阵,基于 focal_length 得到的在3D空间中渲染物体的投影矩阵。4X4投影矩阵在算法内部计算方式:

```
matrix={
    2 * focal_length / img_wid, 0, 0, 0,
    0, 2 * focal_length / img_hei, 0,0,
    0,0, (zf + zn) / (zn - zf), -1,
    0, 0, (2.0f * zf * zn) / (zn - zf), 0};
}
```

○ [30,33]位置,接地数据,脚是否踩地,左脚跟、左脚尖、右脚跟、右脚尖

- position_x, position_y, position_z:
 - [0,23]位置,人体2D点位,见上文的图1, 2D点的 position_z 都是0
 - [24,47]位置,人体3D点位,见上文的图1
- rotation
 - \odot [0,23]位置,人体骨骼旋转四元数,每个四元数的属性顺序是 w 、 x 、 y 、 z
 - [25,54]位置,手部骨骼旋转四元数,左手15个,右手15个,每个四元数的属性顺序是 w、x、y、z

骨骼的不同命名方式及对应关系

序号	Bone Names	Bone Names 2
0	"pelvis",	"Hips"
1	"left_hip",	"LeftUpLeg"
2	"right_hip",	"RightUpLeg"
3	"spine1",	"Spine"
4	"left_knee",	"LeftLeg"

🔗 腾讯云

5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	<pre>"right_knee", "spine2", "left_ankle", "right_ankle", "spine3", "left_foot", "right_foot", "neck", "left_collar", "left_collar", "left_shoulder", "left_shoulder", "left_elbow", "left_elbow", "left_elbow", "left_hand"</pre>	"RightLeg" "Spine1" "LeftFoot" "Spine2" "" "Neck" "LeftShoulder" "RightShoulder" "Head" "LeftArm" "LeftArm" "LeftArm" "LeftForeArm" "LeftForeArm" "RightForeArm" "LeftHand" "LeftHand"
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39	"left_index1" "left_index2" "left_index3" "left_middle1" "left_middle2" "left_middle3" "left_pinky1" "left_pinky2" "left_pinky3" "left_ring1" "left_ring2" "left_ring3" "left_thumb1" "left_thumb2" "left_thumb3	IndexFinger1_L IndexFinger2_L IndexFinger3_L MiddleFinger2_L MiddleFinger3_L PinkyFinger1_L PinkyFinger2_L PinkyFinger3_L RingFinger1_L RingFinger2_L RingFinger2_L ThumbFinger1_L ThumbFinger3_L
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54	"right_index1" "right_index2" "right_index3" "right_middle1" "right_middle2" "right_middle3" "right_pinky1" "right_pinky2" "right_pinky3" "right_ring1" "right_ring2" "right_ring3" "right_thumb1" "right_thumb2" "right_thumb3"	IndexFinger1_R IndexFinger2_R IndexFinger3_R MiddleFinger2_R MiddleFinger3_R PinkyFinger1_R PinkyFinger2_R PinkyFinger3_R RingFinger1_R RingFinger2_R RingFinger2_R ThumbFinger1_R ThumbFinger3_R



语音转表情

iOS

最近更新时间: 2024-07-18 11:03:22

功能说明

输入音频数据,输出苹果 ARKit 标准的52表情数据,请参见 ARFaceAnchor 。您可以利用这些表情数据做一进步的开发,例如传到 Unity 中驱动您的模 型。

接入方式

方式1: 与腾讯特效 SDK 一起使用

- 1. 语音转表情sdk结合腾讯特效 SD中,因此第一步需要按照腾讯特效文档进行接入。
- 2. 下载 腾讯特效 SDK 完整版。
- 3. 参考 独立集成腾讯特效 文档完成集成。
- 4. 将腾讯特效 SDK 完整版内的 Audio2Exp.framework 拉入项目,并且在项目的 target->General->Frameworks,Libraries,and Embedded Content 处设置为Embed & Sign。

方式2: 通过独立的语音转表情 SDK 接入

如果您只需要语音转表情,不需要用到腾讯特效 SDK 的任何能力,则可以考虑使用独立的语音转表情 SDK,Audio2Exp.framework 包约7MB左右。项目 引入 Audio2Exp.framework、YTCommonXMagic.framework 两个动态库,并且在项目的

target->General->Frameworks,Libraries,and Embedded Content 处设置为 Embed & Sign 。

接入步骤

- 1. 设置 License,请参见 鉴权。
- 2. 配置模型文件:请将必需的模型文件 audio2exp.bundle 拷贝到工程目录,然后在调用 Audio2ExpApi 的 initWithModelPath: 接口时,传入参数 "audio2exp.bundle" 模型文件所在的路径。

接口说明

接口	说明
+ (int)initWithModelPath: (NSString*)modelPath;	初始化,传入模型路径,见上文说明。返回值为0表示成功
+ (NSArray *)parseAudio: (NSArray *)inputData;	<pre>输入的是音频数据,要求单通道,16K采样率,数组长度为267(即267个采样点),输出的数据是长度为52的float 数组,表示52表情基,取值为0到1之间,顺序为 苹果标准顺序 {"eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLookOutLeft","eyeLookUpLeft","e yeSquintLeft","eyeWideLeft","eyeBlinkRight","eyeLookDownRight","eyeLookInRight","eyeL ookOutRight","eyeLookUpRight","eyeSquintRight","eyeWideRight","jawForward","jawLeft", "jawRight","jawOpen","mouthClose","mouthFunnel","mouthPucker","mouthRight","mouthLeft ","mouthSmileLeft","mouthSmileRight","mouthFrownRight","mouthFrownLeft","mouthDimpleL eft","mouthDimpleRight","mouthStretchLeft","mouthStretchRight","mouthRollLower","mout hRollUpper","mouthShrugLower","mouthShrugUpper","mouthPressLeft","mouthUpperUpRight","brow DownLeft","browDownRight","browInnerUp","browOuterUpLeft","mouteUpRight","tongueOut" }</pre>
+ (int)releaseSdk	使用完毕后调用,释放资源

集成代码示例

// 加加口语自我衣用Sax NSString *path = [[NSBundle mainBundle] pathForResource:@"audio2exp" ofType:@"bundle"];



```
int ret = [Audio2ExpApi initWithModelPath:path];
// 语音数据转52表情数据
NSArray *emotionArray = [Audio2ExpApi parseAudio:floatArr];
// 释放sdk
[Audio2ExpApi releaseSdk];
// 结合腾讯特效sdk xmgaic使用
// 使用对应的资源初始化美颜sdk
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDict];
// 加载avatar素材
[self.beautyKit loadAvatar:bundlePath exportedAvatar:nil completion:nil];
// 将52表情数据传入美颜sdk 就能看到效果
[self.beautyKit updateAvatarByExpression:emotionArray];
```

```
() 说明:
```

完整的示例代码请参考 美颜特效 SDK demo 工程。

- 录音可以参考 TXCAudioRecorder 。
- 接口使用可以参考: VoiceViewController 及其相关类。



Android

最近更新时间:2024-07-11 11:03:22

功能说明

输入音频数据,输出苹果 ARKit 标准的52表情数据,请参见 ARFaceAnchor 。您可以利用这些表情数据做一进步的开发,例如传到 Unity 中驱动您的模型。

接入方式

方式1: 通过腾讯特效 SDK 接入

语音转表情集成在腾讯特效 SDK 中,因此第一步需要按照腾讯特效文档进行接入。

1. 下载 腾讯特效 SDK 完整版。

2. 参考 独立集成腾讯特效 文档完成集成。

方式2: 通过独立的语音转表情 SDK 接入

如果您只需要语音转表情,不需要用到腾讯特效 SDK 的任何能力,则可以考虑使用独立的语音转表情 SDK,aar 包约6MB左右。请联系我们的架构师或销售获 取此 SDK。

接入步骤

- 1. 设置 License,请参见 鉴权 。
- 2. 配置模型文件:请将必需的模型文件从 assets 拷贝到 app 的私有目录,例如: context.getFilesDir() + "/my_models_dir/audio2exp",然后 在调用 Audio2ExpApi的 init(String modelPath) 按口时,传入参数 context.getFilesDir() + "/my_models_dir" 模型文件在 SDK 包里,位置如下:

assets >	<pre>download_assets ></pre>	🚞 audio2exp	>	audio2phoneme.opt.tnnmodel
demo_ALL_aid_2.4.999.328 >	download_assets.zip	📄 benchmark	>	📄 audio2phoneme.opt.tnnproto
🚞 jniLibs >		🚞 Light3DPlugin	>	🖻 kanghui.template
ibs >		🚞 LightBodyPlugin	>	phoneme2exp.opt.tnnmodel
MotionRes >		🚞 LightCore	>	📄 phoneme2exp.opt.tnnproto
i unity		🚞 LightHandPlugin	>	
		🚞 LightSegmentPlugin	>	
		🚞 lut	>	

接口说明

public int Audio2ExpApi.init(String modelPath);初始化,传入模型路径,见上文说明。返回值为0表示成功public float[] Audio2ExpApi.parseAudio(float[] inputData);输入的是音频数据,要求单通道,16K采样率,数组长度为267(即267个采样点),输出 的数据是长度为52的float数组,表示52表情基,取值为0到1之间,顺序为 苹果标准顺序	接口	说明
public float[] Audio2ExpApi.parseAudio(float[] inputData); 输入的是音频数据,要求单通道,16K采样率,数组长度为267(即267个采样点),输出 的数据是长度为52的float数组,表示52表情基,取值为0到1之间,顺序为 苹果标准顺序	public int Audio2ExpApi.init(String modelPath);	初始化,传入模型路径,见上文说明。返回值为0表示成功
	public float[] Audio2ExpApi.parseAudio(float[] inputData);	输入的是音频数据,要求单通道,16K采样率,数组长度为267(即267个采样点),输出 的数据是长度为52的float数组,表示52表情基,取值为0到1之间,顺序为 苹果标准顺序



public int Audio2ExpApi.release();	使用完毕后调用,释放资源
public int Audio2ExpApi.release();	<pre>ght", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownR: ght", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouth StretchLeft", "mouthStretchRight", "mouthRollLower", "mouthRollUppe: ", "mouthShrugLower", "mouthShrugUpper", "mouthPressLeft", "mouthPress sRight", "mouthLowerDownLeft", "mouthLowerDownRight", "mouthUpperUpp eft", "mouthUpperUpRight", "browDownLeft", "browDownRight", "browInner rUp", "browOuterUpLeft", "browOuterUpRight", "cheekPuff", "cheekSquin tLeft", "cheekSquintRight", "noseSneerLeft", "noseSneerRight", "tongr eOut"} 使用完毕后调用,释放资源</pre>
	<pre>{"eyeBilnkLeft", "eyeBookDownLeft", "eyeLookInLeft", "eyeLookOutLeft" ", "eyeLookUpLeft", "eyeSquintLeft", "eyeWideLeft", "eyeBlinkRight", " eyeLookDownRight", "eyeLookInRight", "eyeLookOutRight", "eyeLookUpR: ght", "eyeSquintRight", "eyeWideRight", "jawForward", "jawLeft", "jawI ight", "jawOpen", "mouthClose", "mouthFunnel", "mouthPucker", "mouthR: ght", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownR: ght", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouth</pre>

集成代码示例

@Override
protected void onCreate(Bundle savedInstanceState) {
<pre>super.onCreate(savedInstanceState);</pre>
<pre>setContentView(R.layout.activity_main);</pre>
<pre>findViewById(R.id.button).setOnClickListener(new OnClickListener() {</pre>
@Override
public void onClick(View view) {
TELicenseCheck.getInstance().setTELicense(MainActivity.this, licenseUrl,
licenseKey, new TELicenseCheckListener() {
@Override
<pre>public void onLicenseCheckFinish(int errorCode, String s) {</pre>
Log.d(TAG , "onLicenseCheckFinish: errorCode =
"+errorCode+",msg="+s);
if (errorCode == TELicenseCheck.ERROR OK) {
Audio2ExpApi audio2ExpApi = new Audio2ExpApi();
int err =
audio2ExpApi.init(MainActivity.this.getFilesDir() +"/models");
Log.d(TAG , "onLicenseCheckFinish: err="+err);

🕛 说明

完整的示例代码请参考 美颜特效 SDK demo 工程。

• 录音可以参考 com.tencent.demo.avatar.audio.AudioCapturer 。

• 接口使用可以参考: com.tencent.demo.avatar.activity.Audio2ExpActivity 及其相关类。