Tencent Effect SDK

SDK 통합 가이드

제품 문서

목록:

SDK 통합 가이드
Tencent Effect SDK 통합하기
ios
Android
Tencent Effect를 라이브 스트리밍 SDK에 통합하기
ios
Android
Flutter
Tencent Effect를 TRTC SDK에 통합하기
ios
Android
Flutter
Tencent Effect를 UGSV SDK에 통합하기
iOS
Android
가상 Avatar 통합 가이드
iOS
Avatar 통합
Avatar SDK
Android
Avatar 통합
사용자 지정 Avatar UI
Avatar SDK
원자 기능 통합 가이드
얼굴 특징점 감지 통합 가이드
얼굴 표정
iOS
Android
신체 키포인트
iOS
Android
Audio-to-Expression
Android
iOS

SDK 통합 가이드
Tencent Effect SDK 통합하기
ios

최종 업데이트 날짜: : 2023-02-27 14:18:15

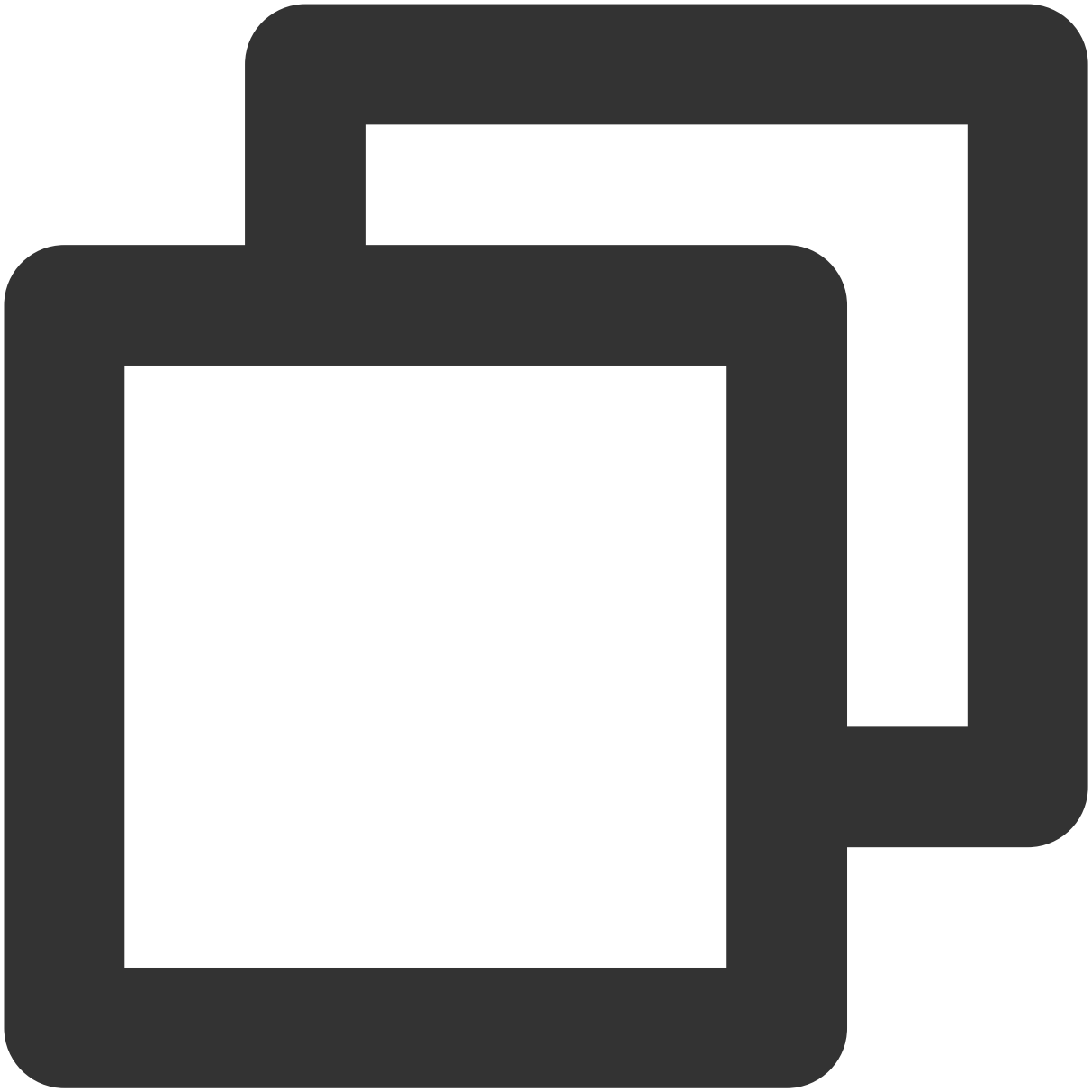
통합 준비

개발자 환경 요구 사항

Xcode 11 이상: App Store 또는 여기에서 다운로드하십시오.
권장 실행 환경:
기기 사양: iPhone 5 이상. iPhone 6 및 이전 모델은 전면 카메라에 대해 최대 720p를 지원하며 1080p은 지원되지 않습니다.
시스템 요구 사항: iOS 10.0 이상.

SDK 가져오기

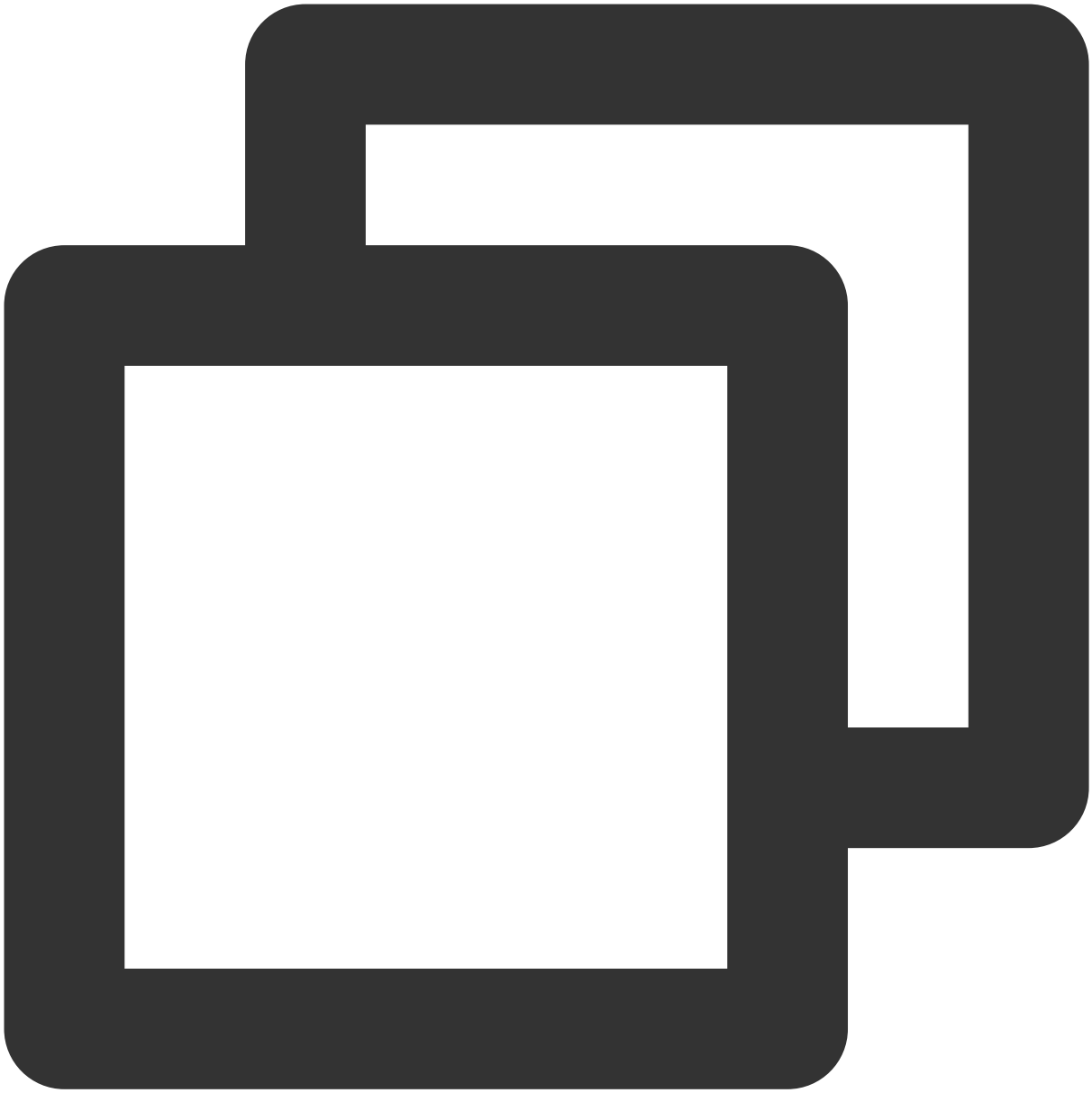
CocoaPods를 사용하거나 SDK를 수동으로 다운로드하여 프로젝트로 가져올 수 있습니다.
CocoaPods 사용
SDK를 다운로드하고 수동으로 가져오기
동적 다운로드 통합
1. **CocoaPods 설치**
터미널 창에 다음 명령어를 입력합니다(먼저 Mac에 Ruby를 설치해야 함).



```
sudo gem install cocoapods
```

2. **Podfile 생성**

프로젝트의 디렉터리로 이동하고 다음 명령을 입력하여 디렉터리에 Podfile을 생성합니다.



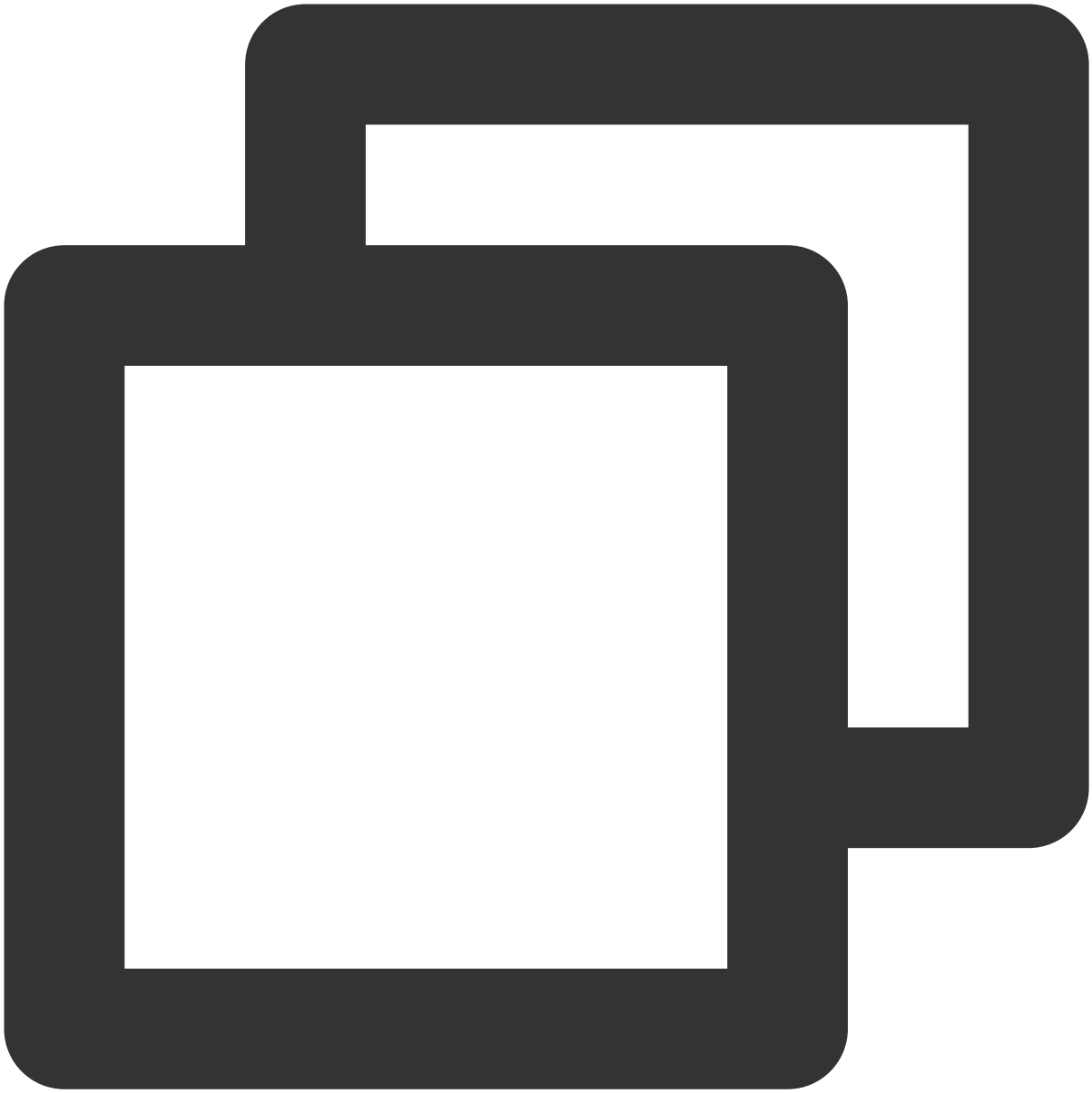
```
pod init
```

3. **Podfile 편집**

프로젝트 요구 사항에 따라 적절한 버전을 선택하고 Podfile을 편집합니다:

XMagic 스탠다드 버전

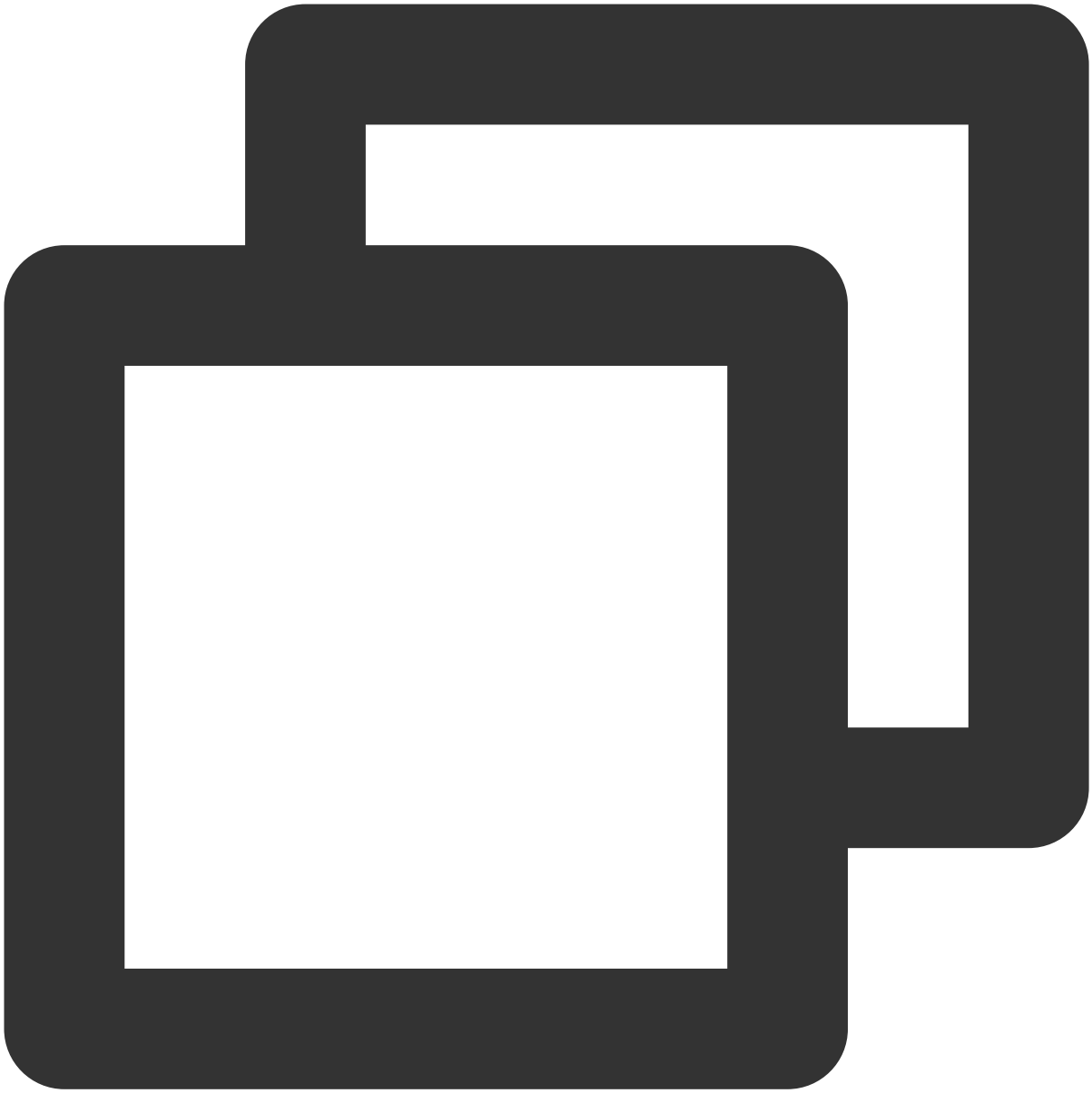
다음과 같이 Podfile을 편집합니다.



```
platform :ios, '8.0'

target 'App' do
pod 'XMagic'
end
```

XMagic 라이트 버전

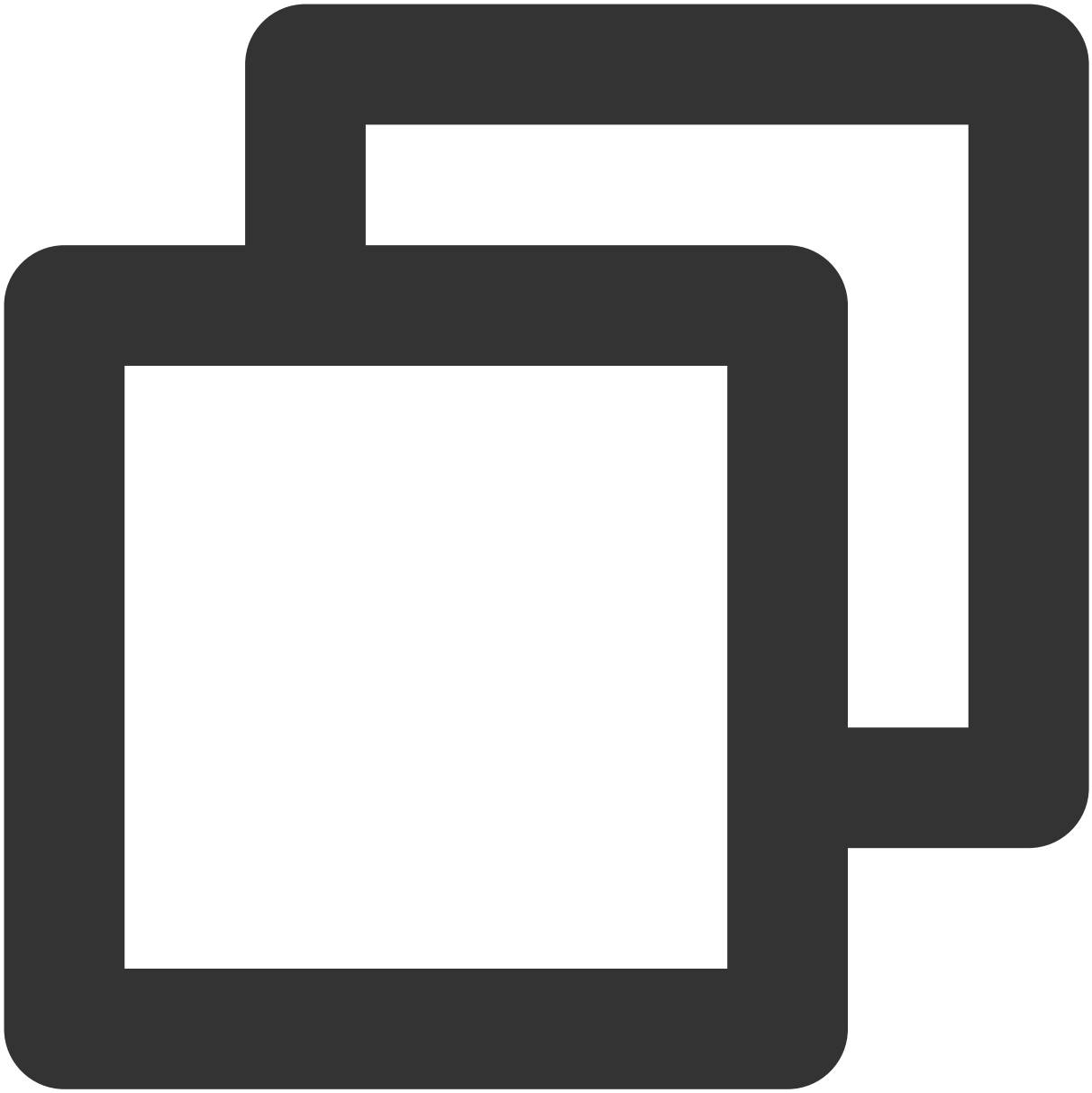
XMagic 라이트 버전의 설치 패키지는 XMagic 스탠다드 버전보다 작습니다. 기본 버전 A1-00, 기본 버전 A1-01 및 고급 버전 S1-00만 지원합니다. Podfile을 다음과 같이 편집합니다.



```
platform :ios, '8.0'

target 'App' do
pod 'XMagic_Smart'
end
```

4. **로컬 저장소 업데이트 및 SDK 설치**

터미널 창에 다음 명령어를 입력하여 로컬 리포지토리를 업데이트하고 SDK를 설치합니다.



```
pod install
```

pod 명령어 실행이 완료되면 SDK `.xcworkspace` 접미사가 통합된 프로그램 파일이 생성되며, 이를 더블클릭해 실행하면 됩니다.

5. **프로젝트에 효과 리소스 추가**

사용하는 Tencent Effect 패키지의 SDK 및 효과 리소스를 다운로드하고 파일을 압축 해제한 후 **resources** 폴더 아래의 **LightCore.bundle**, **Light3DPlugin.bundle**, **LightBodyPlugin.bundle**, **LightHandPlugin.bundle**, **LightSegmentPlugin.bundle, audio2exp.bundle**을 제외한 **모든 bundle 리소스**를 프로젝트에 추가합니다.

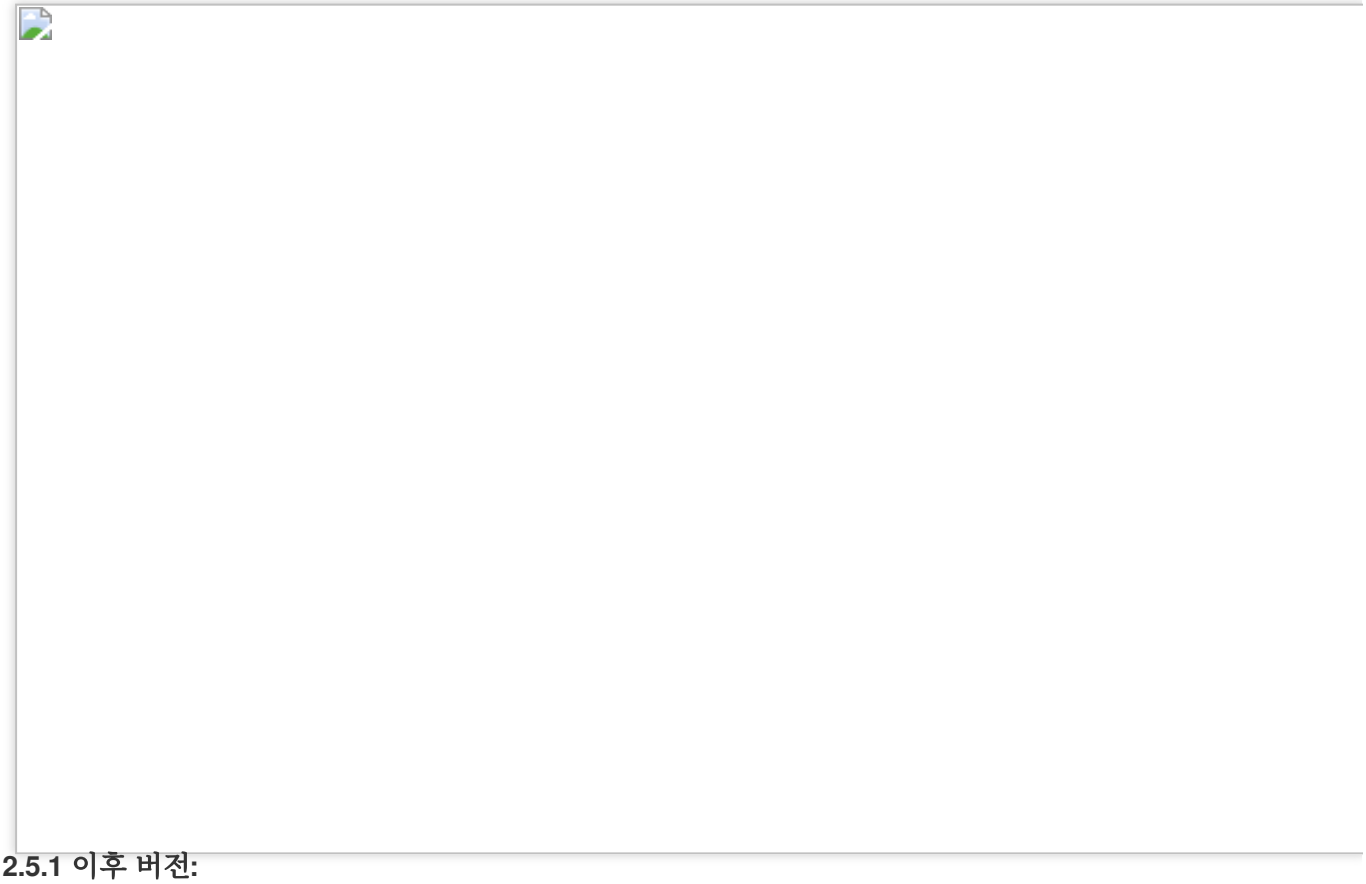
Build Settings의 Other Linker Flags에 `-ObjC` 를 추가합니다.

6. Bundle Identifier를 라이선스에 바인딩된 Bundle Identifier로 변경합니다.

1. SDK 및 효과 리소스를 다운로드하고 압축을 해제합니다. sdk는 frameworks 폴더에 있고 bundle 리소스는 resources에 있습니다.

2. **SDK 2.5.1이전 버전:**

Xcode 프로젝트를 열고 frameworks 폴더의 framework 를 실제 프로젝트에 추가하고 실행할 target을 선택한 다음 **General** 항목을 선택하고 **Frameworks,Libraries,and Embedded Content** 항목 펼치기를 클릭합니다. 하단의 '+' 아이콘을 클릭하여 종속 라이브러리를 추가합니다. 다운로드한 `XMagic.framework` , `YTCommonXMagic.framework` , `libpag.framework` 및 필요한 종속 라이브러리 `MetalPerformanceShaders.framework` , `CoreTelephony.framework` , `JavaScriptCore.framework` , `VideoToolbox.framework` , `libc++.tbd` 를 차례로 추가하고 필요에 따라 다른 툴 라이브러리 `Masonry.framework` (제어 레이아웃 라이브러리), `SSZipArchive` (파일 압축 해제 라이브러리)를 추가합니다.



SDK 2.5.1 이후 버전:

Xcode 프로젝트를 열고 frameworks 폴더의 framework 를 프로젝트에 추가합니다. 실행할 target을 선택하고, **General** 을 선택한 다음

Frameworks,Libraries,and Embedded Content를 확장하고 "+"를 클릭하여 다운로드한 프레임워크 `XMagic.framework` , `YTCommonXMagic.framework` , `libpag.framework` , `Audio2Exp.framework` , `TEFFmpeg.framework` ,

`MetalPerformanceShaders.framework` , `CoreTelephony.framework` , `JavaScriptCore.framework` , `VideoToolbox.framework` 및 `libc++.tbd` 를 추가합니다. 필요한 경우 `Masonry.framework` (컨트롤 레이아웃) 및 `SSZipArchive` (파일 압축 해제)를 추가할 수도 있습니다.

3. resources 폴더의 효과 리소스를 프로젝트에 추가합니다.
4. Build Settings의 Other Linker Flags에 `-ObjC` 를 추가합니다.
5. Bundle Identifier를 라이선스에 바인딩된 Bundle Identifier로 변경합니다.

패키지 크기를 줄이기 위해 SDK에 필요한 모델 리소스 및 애니메이션 리소스 MotionRes(일부 기본 버전 SDK 애니메이션 리소스 없음)를 인터넷으로 변경하여 다운로드할 수 있습니다. 다운로드가 완료 후 위 파일의 경로를 SDK로 설정합니다.

Demo의 다운로드 로직을 재사용하는 것이 좋습니다. 기존 다운로드 서비스를 사용할 수도 있습니다. 동적 다운로드에 대한 자세한 지침은 SDK 패키지 축소(iOS)를 참고하십시오.

권한 구성

Info.plist 파일에 해당 권한에 대한 설명을 추가하십시오. 그렇지 않으면 iOS 10 시스템에서 프로그램이 크래쉬될 수 있습니다. App이 카메라를 사용할 수 있도록 Privacy - Camera Usage Description에서 카메라 권한을 활성화하십시오.

통합 단계

1단계: 인증

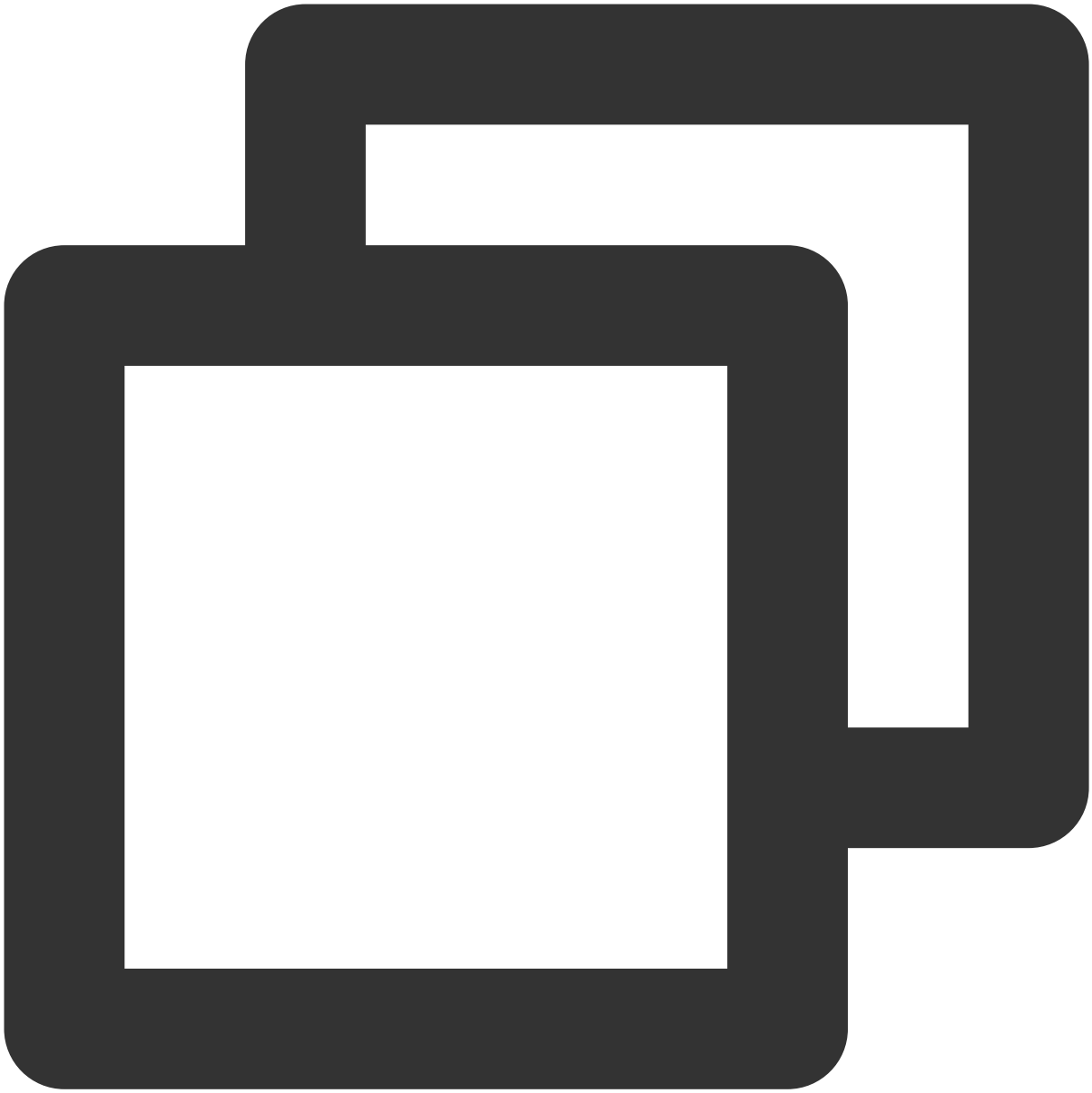
1. 라이선스를 신청하고 LicenseURL과 LicenseKEY를 받습니다.

참고

일반적인 상황에서는 App이 한 번만 성공적으로 연결되면 인증 프로세스를 완료할 수 있으므로 License 파일을 프로젝트의 프로젝트 디렉터리에 넣을 **필요가 없습니다**. 하지만 App이 인터넷에 연결되지 않은 상태에서도 SDK 관련 기능을 사용하려면 License 파일을 다운로드하여 프로젝트 디렉터리에 넣을 수 있으며, License 파일 이름은 반드시 `v_cube.license` 여야 합니다.

2. 해당 비즈니스 모듈의 초기화 코드에 URL과 KEY를 설정하여 license 다운로드를 트리거하여 사용 전에 임시로 다운로드하는 것을 피합니다. AppDelegate의 didFinishLaunchingWithOptions 메소드에서도 다운로드가 실행될 수 있습니다. 이 중 LicenseURL과 LicenseKey는 콘솔이 License에 바인딩될 때 생성되는 권한 정보입니다.

SDK 2.5.1 이전 버전에서 `TELicenseCheck.h` 는 `XMagic.framework` 에 있고, SDK 2.5.1 이후 버전에서 `TELicenseCheck.h` 는 `YTCommonXMagic.framework` 에 있습니다.



```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
 if (authresult == TELicenseCheckOk) {
     NSLog(@"인증 성공");
 } else {
     NSLog(@"인증 실패");
 }
}];
```

errorCode 인증 설명:

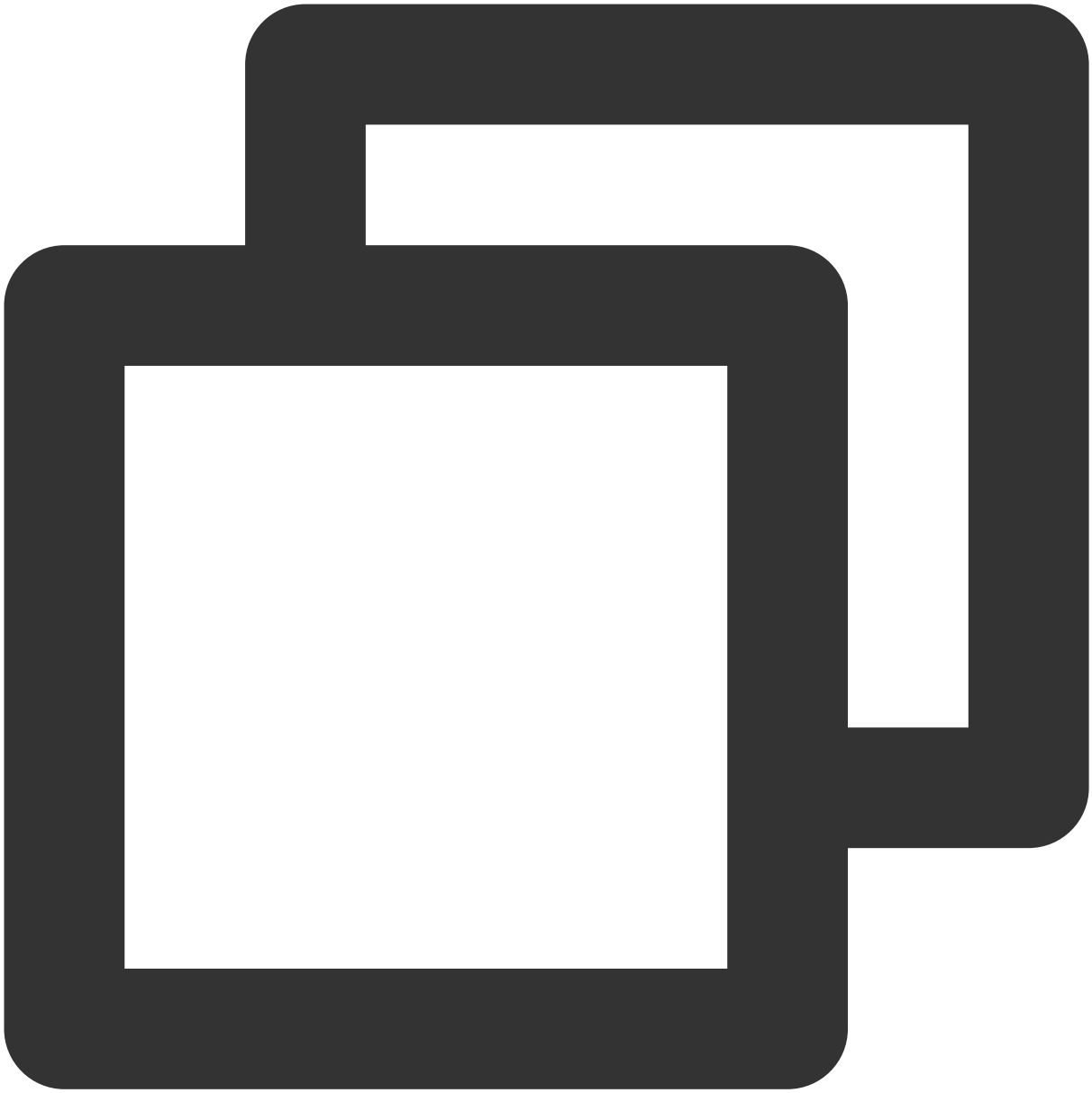
에러 코드	설명

0	성공. Success
-1	입력 매개변수가 유효하지 않습니다. 예: URL 또는 KEY가 비어 있습니다.
-3	다운로드에 실패했습니다. 네트워크 설정을 확인하십시오.
-4	로컬 시스템에서 읽은 TE SDK 인증 정보가 비어 있으며, 이는 I/O 실패로 인해 발생할 수 있습니다
-5	읽은 VCUBE TEMP License 파일의 내용이 비어 있으며 이는 I/O 실패로 인해 발생할 수 있습니다
-6	v_cube.license 파일의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-7	서명 확인에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-8	암호 해독에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-9	TELicense 필드의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-10	온라인으로 파싱된 TE SDK 인증 정보가 비어 있습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-11	TE SDK 인증 정보를 로컬 파일에 쓰지 못했습니다. I/O 실패로 인해 발생할 수 있습니다.
-12	다운로드에 실패했으며 로컬 asset을 파싱하지 못했습니다.
-13	인증 실패
기타	도움이 필요한 경우 Tencent Cloud에 문의하십시오

2단계: SDK(XMagic.framework) 로딩

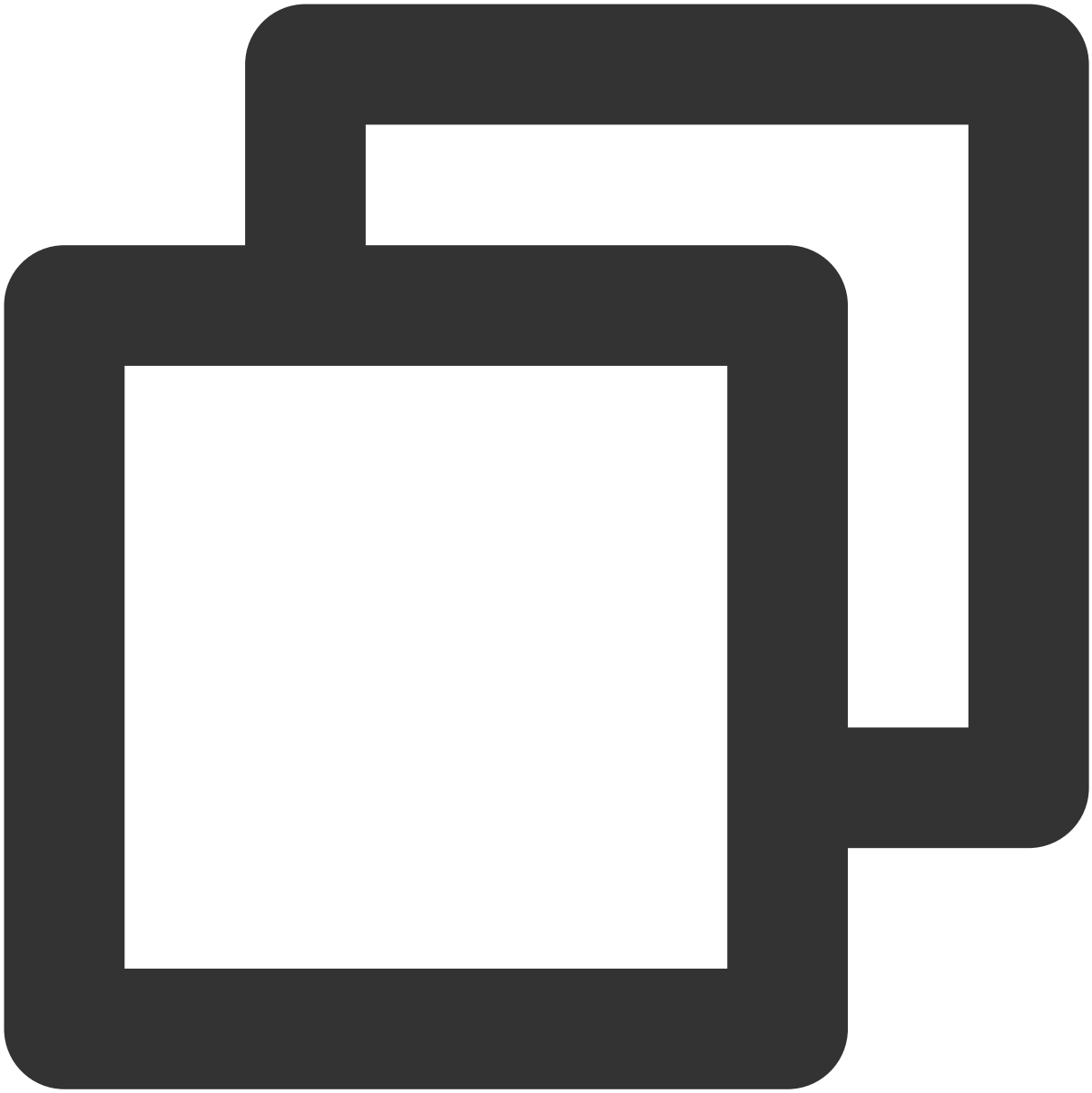
Tencent 특수 효과 SDK를 사용하는 라이프사이클은 대략 다음과 같습니다.

1. 효과 리소스를 로딩합니다.



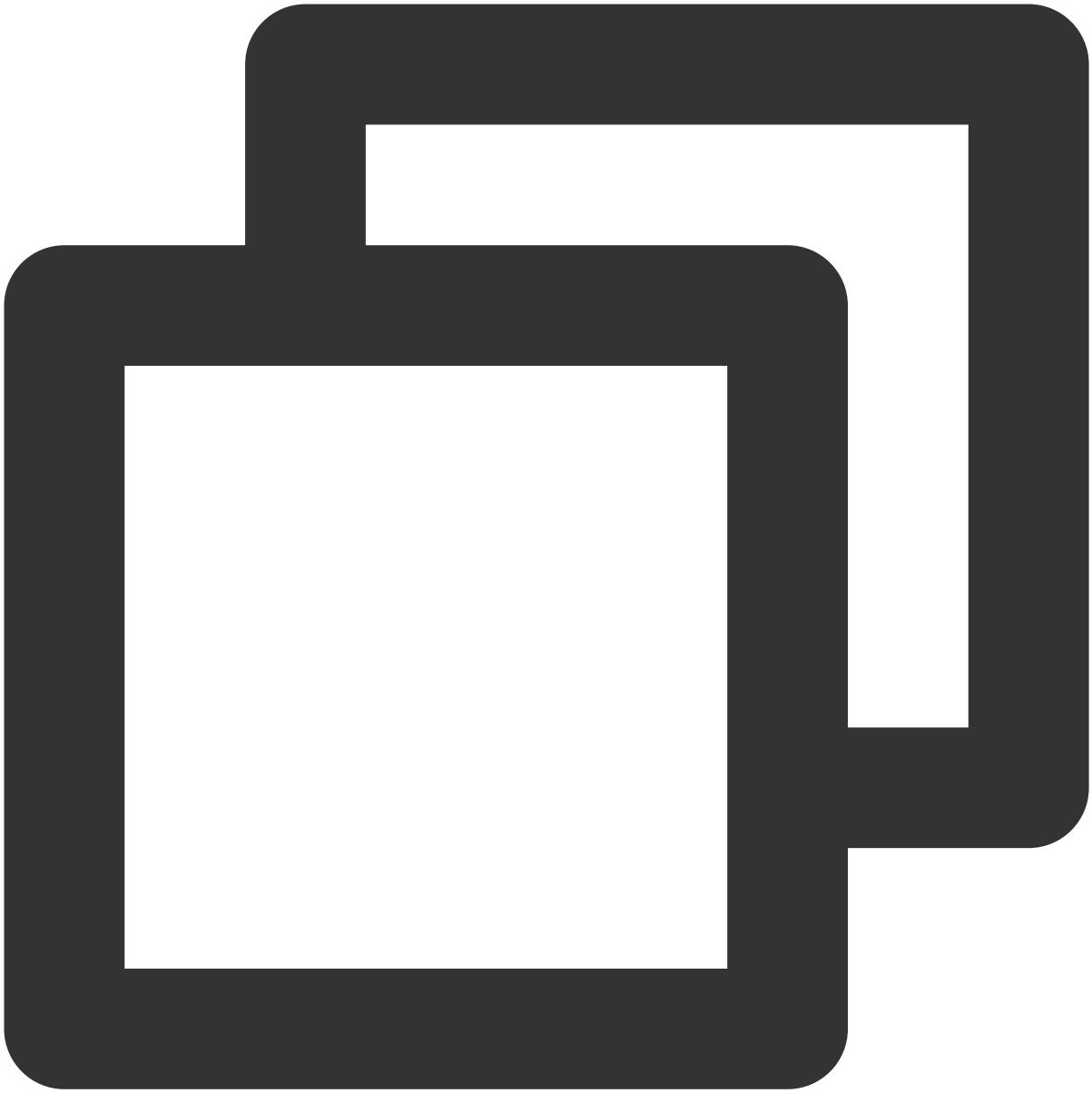
```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
 @"root_path":[[NSBundle mainBundle] bundlePath]
};
```

2. Tencent Effect SDK를 초기화합니다.

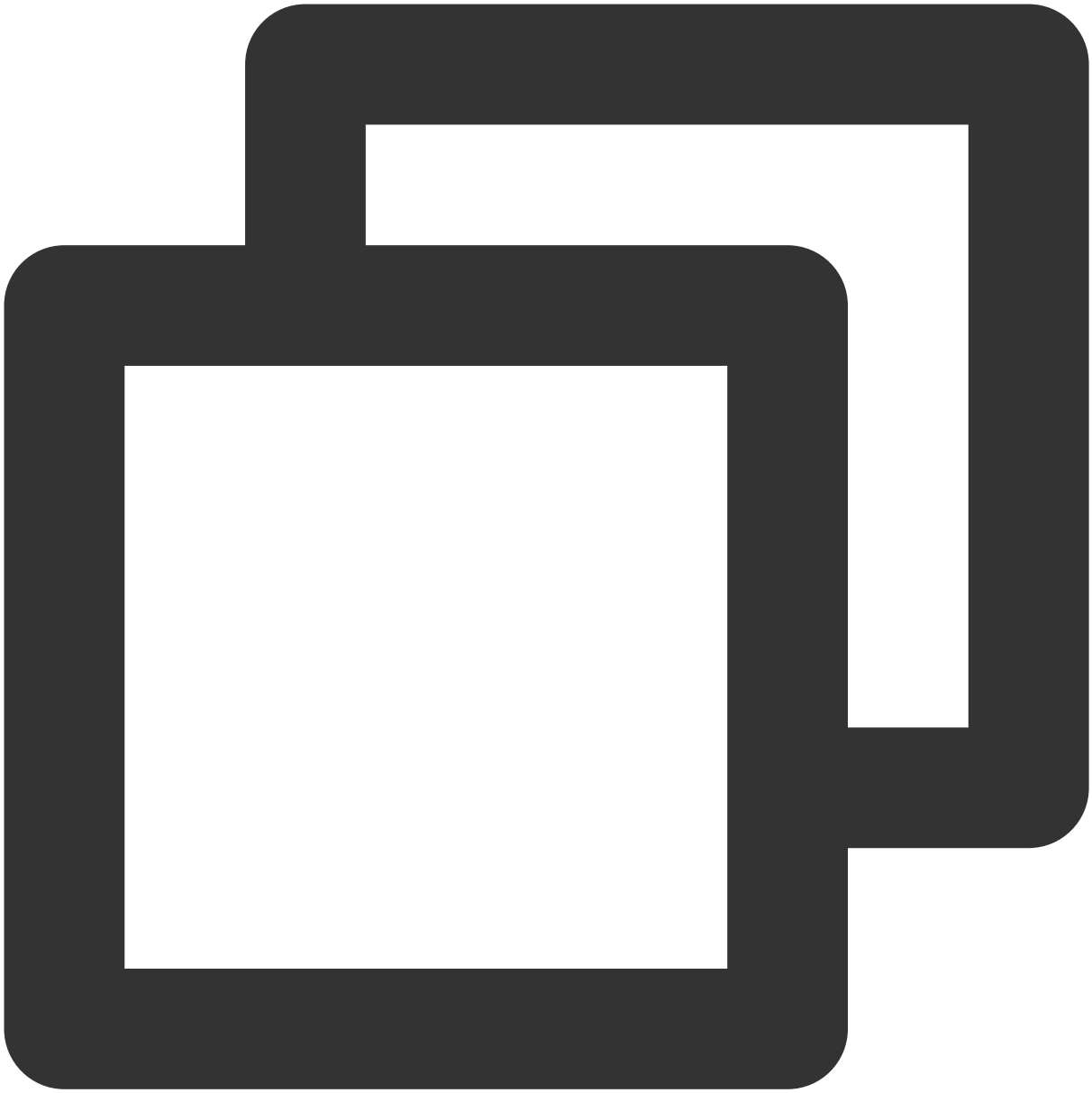


```
initWithRenderSize:assetsDict: (XMagic)
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
```

3. SDK는 데이터의 각 프레임을 처리하고 결과를 반환합니다.



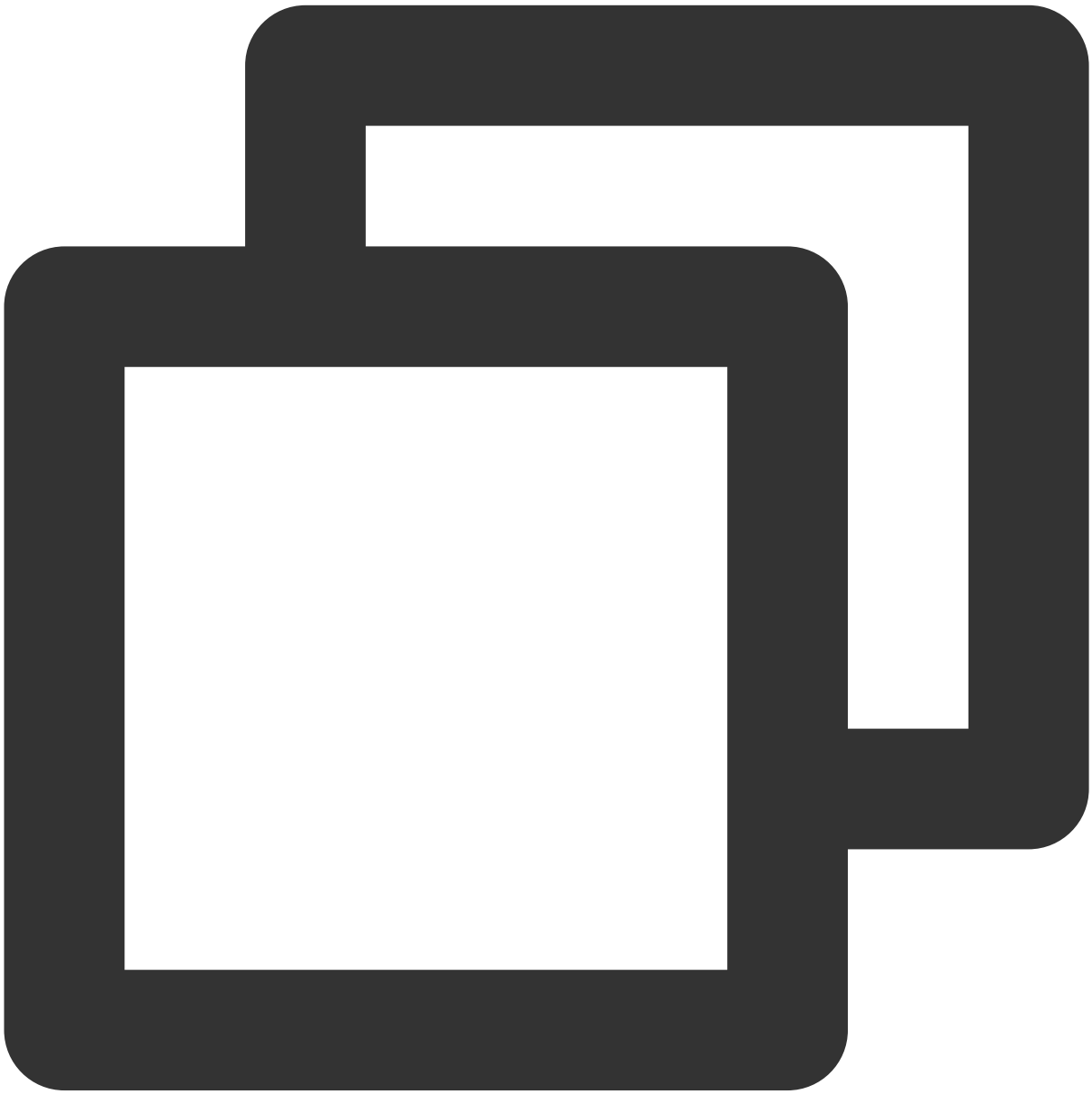
```
process: (XMagic)
```



```
// 카메라 콜백을 통해 프레임 데이터 전달
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMSam

// 원시 데이터를 가져와 각 프레임의 렌더링 정보 처리
- (void)mycaptureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMS

// CPU를 사용하여 데이터 처리
- (YTProcessOutput*)processDataWithCpuFuc:(CMSampleBufferRef)inputSampleBuffer;

// GPU를 사용하여 데이터 처리
- (YTProcessOutput*)processDataWithGpuFuc:(CMSampleBufferRef)inputSampleBuffer;
```

```
// Tencent Effect SDK의 데이터 처리 API
/// @param input 처리할 데이터 입력
/// @return 처리된 데이터 출력
- (YTProcessOutput* _Nonnull)process:(YTProcessInput * _Nonnull)input;
```

4. Tencent Effect SDK를 릴리스합니다.



```
deinit (XMagic)
// SDK의 리소스를 해제해야 할 때 이 API 호출
[self.beautyKit deinit]
```

설명

위의 단계를 완료한 후 필요에 따라 디스플레이 타이밍 및 기타 장치 환경 매개변수를 제어할 수 있습니다.

FAQ

질문1: 컴파일 오류: “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):”가 발생하는 경우 어떻게 해야 합니까?

1. **File** > **Project settings** > **Build System**으로 이동하고 **Legacy Build System**을 선택합니다.
2. Xcode 13.0++ 의 경우 **File** > **Workspace Settings**에서 **Do not show a diagnostic issue about build system deprecation**을 선택해야 합니다.

질문2: 리소스를 가져온 후 iOS 프로젝트를 컴파일할 때 “Xcode 12.X 버전 컴파일 Building for iOS Simulator, but the linked and embedded framework '.framework'...” 오류가 발생하면 어떻게 해야 합니까?

Build Settings > **Build Options**로 이동하여 **Validate Workspace**를 Yes로 변경하고 **실행**을 클릭합니다.

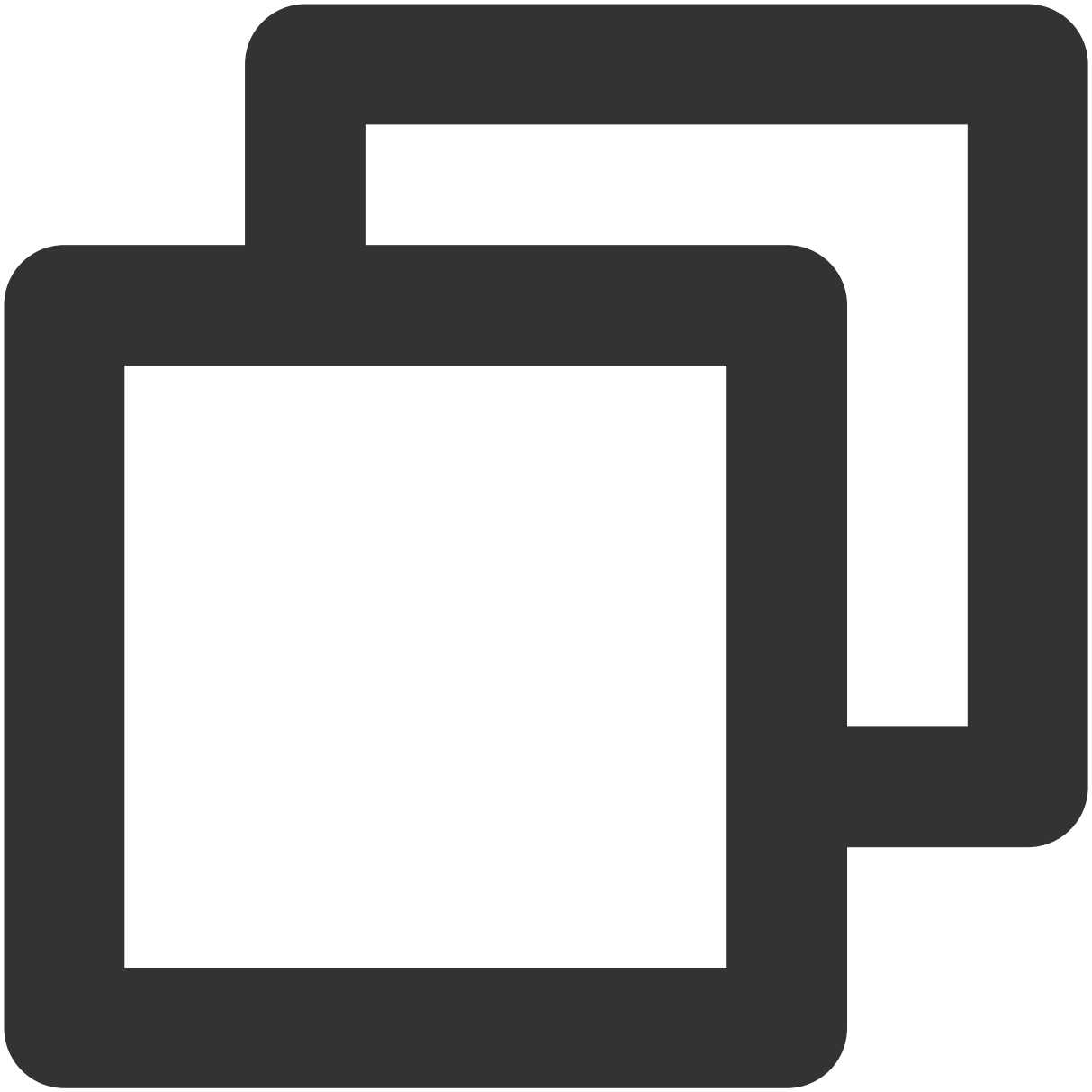
설명

Validate Workspace를 Yes로 변경한 후, 컴파일이 완료된 후 다시 No로 변경해도 프로젝트를 성공적으로 실행할 수 있습니다.

질문3: 필터 설정이 적용되지 않으면 어떻게 해야 합니까?

설정한 값을 확인하십시오(값 범위: 0-100). 값이 너무 작으면 효과가 명확하지 않을 수 있습니다.

질문4: iOS Demo 프로젝트 컴파일 시 dSYM 생성 오류가 발생하면 어떻게 해야 합니까?



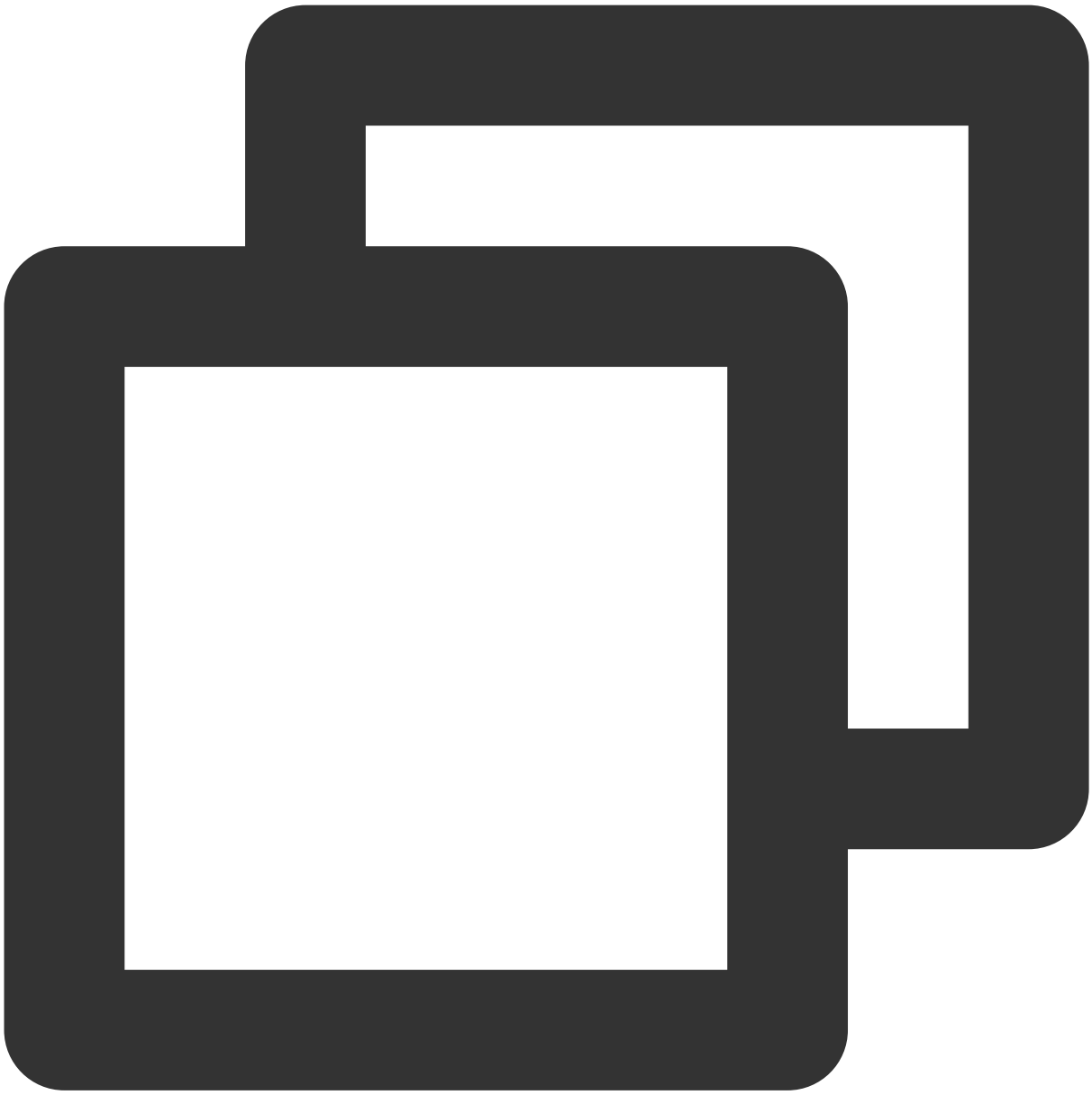
```
PhaseScriptExecution CMake\\ PostBuild\\ Rules build/XMagicDemo.build/Debug-iphoneo
    cd /Users/zhenli/Downloads/xmagic_s106
    /bin/sh -c /Users/zhenli/Downloads/xmagic_s106/build/XMagicDemo.build/Debug-iph

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

문제 원인: `libpag.framework 및 Masonary.framework` 에 다시 서명하지 못했습니다.

솔루션:

1.1 `demo/copy_framework.sh` 를 엽니다.

1.2 다음 명령을 사용하여 로컬 cmake의 절대 경로를 확인하고 `$(what cmake)` 를 cmake의 절대 경로로 변경합니다.



```
which cmake
```

1.3 모든 `Apple Development: ……` 를 자신의 서명으로 교체하십시오.

Android

최종 업데이트 날짜: : 2022-12-19 16:43:08

통합 준비

1. SDK Download 후 압축을 해제합니다.
2. **다음 파일 준비:**

파일 유형	설명
xmagic-xxx.aar	SDK, 필수
../assets/	알고리즘 모델, 소재 리소스 패키지, 필수
../jniLibs	so 라이브러리, 필수

리소스 가져오기

- 수동 통합
- Maven 통합
- 동적 다운로드 통합

통합

- 상기 파일로 준비된 모든 `.aar` 파일을 app 프로젝트의 `libs` 디렉터리에 추가합니다.
- SDK 패키지의 assets/에 있는 모든 파일을 `../src/main/assets` 디렉터리에 복사합니다. SDK 패키지의 MotionRes 폴더에 파일이 있으면 `../src/main/assets` 디렉터리에도 복사합니다.
- jniLibs 폴더를 프로젝트의 `../src/main/jniLibs` 디렉터리에 복사합니다.

가져오기 방법

app 모듈의 `build.gradle` 을 열고 종속 참조 테이블을 추가합니다.

```
android{
...
defaultConfig {
applicationId "인증된 lic 라이선스에 바인딩된 패키지 이름으로 수정"
....
}
packagingOptions {
```

```
pickFirst '**/libc++_shared.so'
}
}
dependencies{
...
compile fileTree(dir: 'libs', include: ['*.jar','*.aar'])//추가 *.aar
}
```

주의
Google의 Gson 라이브러리가 프로젝트에 통합되지 않은 경우 다음 종속성을 추가해야 합니다.

```
dependencies{
implementation 'com.google.code.gson:gson:2.8.2'
}
```

전체 프로세스

1단계: 인증

1. 라이선스를 신청하고 License URL과 License KEY를 받습니다.

주의 :
일반적인 상황에서는 App이 한 번만 성공적으로 연결되면 인증 프로세스가 완료되므로 License 파일을 프로젝트의 assets 디렉터리에 넣을 **필요가 없습니다**. 그러나 만약 App이 인터넷에 연결되지 않은 상태에서도 SDK 관련 기능을 사용하려면 License 파일을 assets 디렉터리에 다운로드하여 기본 설정으로 사용할 수 있습니다. 이 때 License 파일 이름은 `v_cube. license` 이어야 합니다.

2. 해당 비즈니스 모듈의 초기화 코드에 URL과 KEY를 설정하여 License 다운로드를 트리거하여 사용 전 일시적으로 다운로드되는 것을 방지합니다. Application의 onCreate 메소드에서 다운로드를 트리거하는 것도 가능하지만, 이 때 네트워크 권한이 없거나 네트워크 실패율이 높을 수 있기 때문에 권장하지 않습니다.

```
//license를 다운로드하거나 업데이트하기 위한 것일 뿐 인증 결과에 신경 쓰지 않는다면 네 번째 매개변수가 null로 전달됩니다.
TELicenseCheck.getInstance().setXMagicLicense(context, URL, KEY, null);
```

3. 그런 다음 실제로 뷰티 필터 기능(예시: Demo의 'LaunchActivity.java')을 사용하기 전에 인증을 수행합니다.

```
// 네트워크에서 so 라이브러리를 다운로드한 경우 TELicenseCheck.getInstance().setTELic
ense를 호출하기 전에 so 경로를 설정하십시오. 그렇지 않으면 인증이 실패합니다.
// XmagicApi.setLibPathAndLoad(validLibsDirectory);
// so가 apk 패키지에 내장되어 있으면 위의 메소드를 호출할 필요가 없습니다.
TELicenseCheck.getInstance().setTELicense(context, URL, KEY, new TELicenseCheck
Listener() {
@Override
public void onLicenseCheckFinish(int errorCode, String msg) {
//주의: 스레드 호출이 꼭 필요한 것은 아닙니다.
if (errorCode == TELicenseCheck.ERROR_OK) {
//인증 성공
} else {
//인증 실패
}
}
});
```

인증 errorCode 설명:

에러 코드	설명
0	성공. Success
-1	잘못된 입력 매개변수, 예시: 빈 URL 또는 KEY
-3	다운로드 실패, 네트워크 설정 확인
-4	로컬에서 읽은 TE 인증 정보가 비어 있음, 이는 IO 실패로 인해 발생한 것일 수 있습니다.
-5	VCUBE TEMP License 파일 읽기 내용이 비어 있습니다. 이는 IO 실패로 인한 것일 수 있습니다.
-6	`v_cube.license` 파일 JSON 필드가 잘못되었습니다. 처리를 위해 Tencent Cloud 팀에 문의하십시오.
-7	서명 인증 실패. 처리를 위해 Tencent Cloud 팀에 문의하십시오.
-8	복호화 실패. 처리를 위해 Tencent Cloud 팀에 문의하십시오.
-9	TELicense 필드의 JSON 필드가 올바르지 않습니다. 처리를 위해 Tencent Cloud 팀에 문의하십시오.
-10	네트워크에서 리졸브된 TE 인증 정보가 비어 있습니다. 처리를 위해 Tencent Cloud 팀에 문의하십시오.

에러 코드	설명
-11	IO 실패로 인해 로컬 파일에 TE 인증 정보를 쓰지 못했습니다.
-12	다운로드에 실패했으며 로컬 asset의 리졸브도 실패했습니다.
-13	인증에 실패했습니다. 패키지에 so가 있는지 확인하거나 so 경로가 올바르게 설정되었는지 확인하십시오.
3004/3005	인증이 잘못되었습니다. Tencent Cloud 팀에 문의하십시오.
3015	Bundle Id / Package Name이 일치하지 않습니다. App 에서 사용하는 Bundle Id / Package Name이 신청한 것과 동일한지 확인하고 올바른 인증 파일이 사용되었는지 확인하십시오.
3018	인증 파일이 만료되었으므로 Tencent Cloud에 연장을 신청해야 합니다.
기타	Tencent Cloud 팀에 문의하십시오.

2단계: 리소스 복사

1. 리소스 파일이 assets 디렉터리에 내장된 경우 사용하기 전에 app의 개인 디렉터리에 copy해야 합니다. 미리 copy하거나 이전 단계에서 인증 성공 콜백에서 복사 작업을 수행할 수 있습니다. 예시 코드는 Demo의 'LaunchActivity.java'에 있습니다.

```
XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").getAbsolutePath()
);
//loading
// 리소스 파일을 프라이빗 디렉터리에 copy합니다. 한 번만 수행하면 됩니다.
XmagicResParser.copyRes(getApplicationContext());
```

2. 동적 네트워크 다운로드에서 리소스 파일을 다운로드한 경우 다운로드가 성공한 후 리소스 파일 경로를 설정해야 합니다. 예시 코드는 Demo의 'LaunchActivity.java'에 있습니다.

```
XmagicResParser.setResPath(다운로드한 리소스 파일 로컬 경로);
```

3단계: SDK 초기화 및 사용 방법

Tencent 특수 효과 SDK를 사용하는 라이프사이클은 대략 다음과 같습니다.

1. 뷰티 필터 UI 데이터 구성은 Demo 프로젝트를 참고하십시오. `XmagicResParser.java,XmagicUIProperty.java,XmagicPanelDataManager.java` 코드.

2. 미리보기 레이아웃에 GLSurfaceView를 추가합니다.

```
<android.opengl.GLSurfaceView
android:id="@+id/camera_gl_surface_view"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

3. (옵션) 카메라를 빠르게 구현합니다.

Demo 프로젝트의 com.tencent.demo.camera 디렉터리를 프로젝트에 복사합니다. 카메라 기능을 빠르게 구현하려면 'PreviewMgr' 클래스를 사용하십시오. 자세한 구현 방법은 Demo 프로젝트의 'MainActivity.java'를 참고하십시오.

```
//카메라 초기화
mPreviewMgr = new PreviewMgr();
//레이아웃의 GlSurfaceView 예제를 카메라 툴 클래스에 전달
mPreviewMgr.onCreate(mGlSurfaceView,false);
//미리보기 텍스처 데이터 콜백 함수 가입
mPreviewMgr.setCustomTextureProcessor((textureId, textureWidth, textureHeight)
-> {
if (mXmagicApi == null) {
return textureId;
}
//렌더링을 위해 뷰티 필터 sdk 호출
int outTexture = mXmagicApi.process(textureId, textureWidth, textureHeight);
return outTexture;
});
//Activity의 onResume 메소드에서 카메라 실행
mPreviewMgr.onResume(this, 1280, 720);
```

4. 뷰티 필터 SDK를 초기화하고 Activity의 'onResume()' 메소드에 넣는 것을 권장합니다.

```
mXmagicApi = new XmagicApi(this, XmagicResParser.getResPath(),new XmagicApi.OnX
magicPropertyErrorListener());
```

- **매개변수**

매개변수	의미
Context context	컨텍스트
String resDir	리소스 파일 디렉터리, 자세한 내용은 2단계를 참고하십시오.

OnXmagicPropertyErrorListener errorListener	콜백 함수 구현 클래스

- **반환**
  오류 코드 의미는 API 문서를 참고하십시오.

5. 소재 프롬프트 콜백 함수를 추가합니다(메소드 콜백은 서브 스레드에서 실행될 수 있음). 고개 끄덕임, 손바닥 내밀기, 손 하트 등 일부 소재는 사용자에게 프롬프트를 표시합니다. 이 콜백은 바로 유사한 프롬프트를 보여주기 위한 것입니다.

```
mXmagicApi.setTipsListener(new XmagicTipsListener() {
final XmagicToast mToast = new XmagicToast();
@Override
public void tipsNeedShow(String tips, String tipsIcon, int type, int duration)
{
mToast.show(MainActivity.this, tips, duration);
}
@Override
public void tipsNeedHide(String tips, String tipsIcon, int type) {
mToast.dismiss();
}
});
```

6. 뷰티 필터 SDK는 데이터의 각 프레임을 처리하고 해당 처리 결과를 반환합니다.

```
int outTexture = mXmagicApi.process(textureId, textureWidth, textureHeight);
```

7. 지정된 유형의 뷰티 필터 특수 효과 값을 업데이트합니다.

```
// 사용 가능한 입력 속성은 XmagicResParser.parseRes()에서 가져오기
mXmagicApi.updateProperty(XmagicProperty<?> p);
```

8. 뷰티 필터 SDK를 Pause하려면 Activity의 'onPause()' 라이프사이클로 바인딩하는 것이 좋습니다.

```
//Activity의 onPause 시 호출되며 OpenGL 스레드에서 호출되어야 합니다.
mXmagicApi.onPause();
```

9. 뷰티 필터 SDK를 릴리스하고 Activity의 'onDestroy()' 라이프사이클과 바인딩하는 것이 좋습니다.

```
//이 메소드는 GL 스레드에서 호출되어야 합니다.
mXmagicApi.onDestroy()
```

Tencent Effect를 라이브 스트리밍 SDK에 통합하기

ios

최종 업데이트 날짜: : 2023-02-27 14:18:15

통합 준비

1. Demo 패키지를 다운로드하고 압축을 해제합니다.
2. Demo 프로젝트의 xmagic 모듈(bundle, XmagicIconRes, Xmagic 폴더)을 실제 항목의 프로젝트로 가져옵니다.
3. **사용하는 XMagic SDK 버전이 2.5.0 이전인 경우,** SDK 디렉터리에서 `libpag.framework`, `Masonry.framework`, `XMagic.framework`, `YTCommonXMagic.framework`를 가져옵니다. **사용하는 XMagic SDK 버전이 2.5.1 이상인 경우,** SDK 디렉터리에서 `libpag.framework`, `Masonry.framework`, `XMagic.framework`, `YTCommonXMagic.framework`, `Audio2Exp.framework`, `TEFFmpeg.framework`를 가져옵니다.
4. framework 서명의 경우 **General--> Masonry.framework** 및 **libpag.framework**를 **Embed & Sign**으로 설정합니다. **YTCommonXMagic.framework** 버전 2.5.1 이전은 **Do Not Embed**를 선택하고, 버전 2.5.1 이후는 **Embed & Sign**을 선택합니다.
5. 발급된 라이선스와 일치하도록 Bundle ID를 수정합니다.

개발자 환경 요구 사항

Xcode 11 이상: App Store 또는 여기에서 다운로드하십시오.
권장 실행 환경:
기기 사양: iPhone 5 이상. iPhone 6 이하의 경우 전면 카메라는 720p까지 지원하며 1080p는 지원되지 않습니다.
시스템 요구 사항: iOS 10.0 이상.

C/C++ 레이어 개발 환경

XCode는 기본적으로 C++ 환경을 사용합니다.

유형	종속성 라이브러리
시스템 종속 라이브러리	Accelerate AssetsLibrary AVFoundation CoreMedia CoreFoundation CoreML Foundation

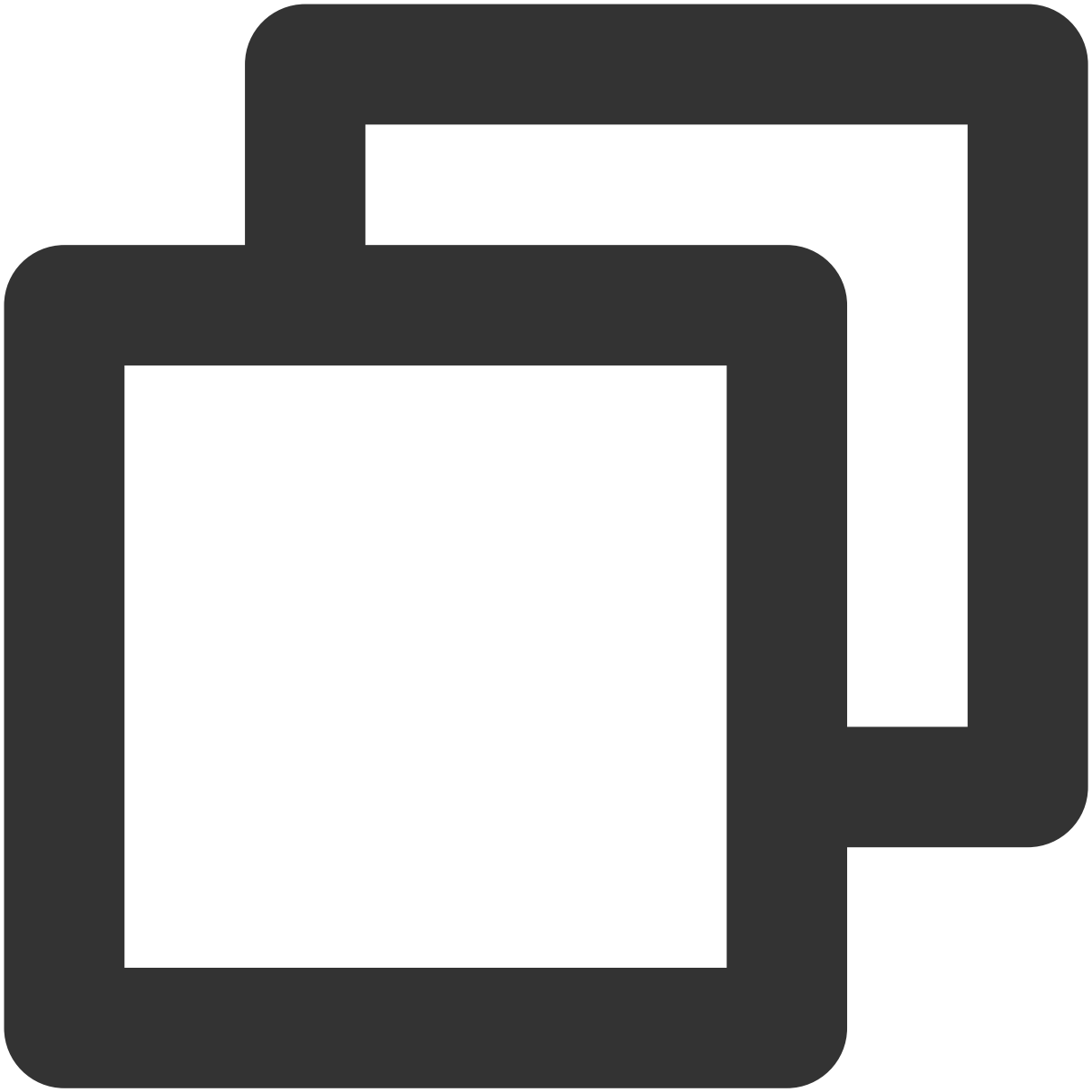
	JavaScriptCore libc++.tbd libz.b libresolv.tbd libsqlite3.0.tbd MetalPerformanceShaders MetalKit MobileCoreServices OpneAL OpneGLES ReplayKit SystemConfiguration UIKit
내장 라이브러리	YTCommon(인증 정적 라이브러리) XMagic(뷰티 필터 정적 라이브러리) libpag(비디오 디코딩 동적 라이브러리) Masonry(컨트롤 레이아웃 라이브러리) TXLiteAVSDK_Professional TXFFmpeg TXSoundTouch Audio2Exp(xmagic sdk version은 2.5.1 이후 버전에만 있음) TEFFmpeg(xmagic sdk version은 2.5.1 이후 버전에만 있음)

SDK API 통합

1단계 및 2단계는 Demo 프로젝트에 있는 `ThirdBeautyViewController` 클래스의 `viewDidLoad` 및 `buildBeautySDK` 메소드를 참고하십시오. `AppDelegate` 클래스의 `application` 메소드는 Xmagic 인증을 수행합니다.
4단계부터 7단계에 대해서는 Demo 프로젝트에서 `ThirdBeautyViewController` 및 `BeautyView` 클래스의 인스턴스 코드를 참고하십시오.

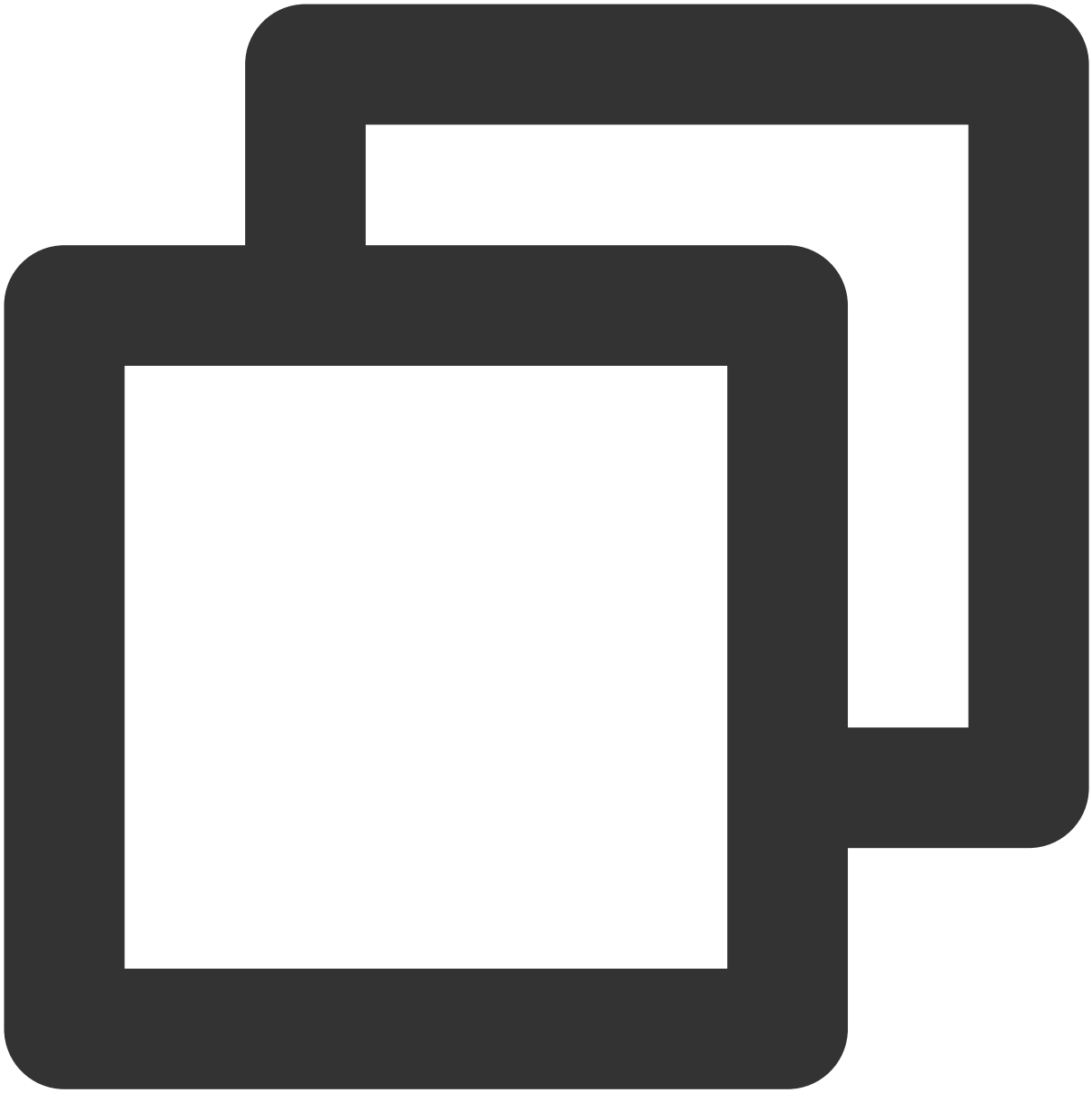
1단계: 인증 초기화

1. 프로젝트의 `AppDelegate` 의 `didFinishLaunchingWithOptions` 에 다음 인증 코드를 추가합니다. 여기서 `LicenseURL` 과 `LicenseKey` 는 Tencent Cloud 공식 웹사이트에서 얻은 인증 정보입니다.



```
[TXLiveBase setLicenceURL:LicenseURL key:LicenseKey];
```

2. Xmagic 인증: 해당 비즈니스 모듈의 초기화 코드에 URL 및 KEY를 설정하여 License 다운로드를 트리거합니다. 사용 직전에 다운로드하지 마십시오. `AppDelegate` 의 `didFinishLaunchingWithOptions` 메소드에서 다운로드를 트리거할 수도 있습니다. 여기서 `LicenseURL` 과 `LicenseKey` 는 License 바인딩 시 콘솔에서 생성되는 정보입니다. 2.5.1 이전의 SDK 버전에서 `TELicenseCheck.h` 는 `XMagic.framework` 에 있고, 2.5.1 이상의 SDK 버전에서 `TELicenseCheck.h` 는 `YTCommonXMagic.framework` 에 있습니다.



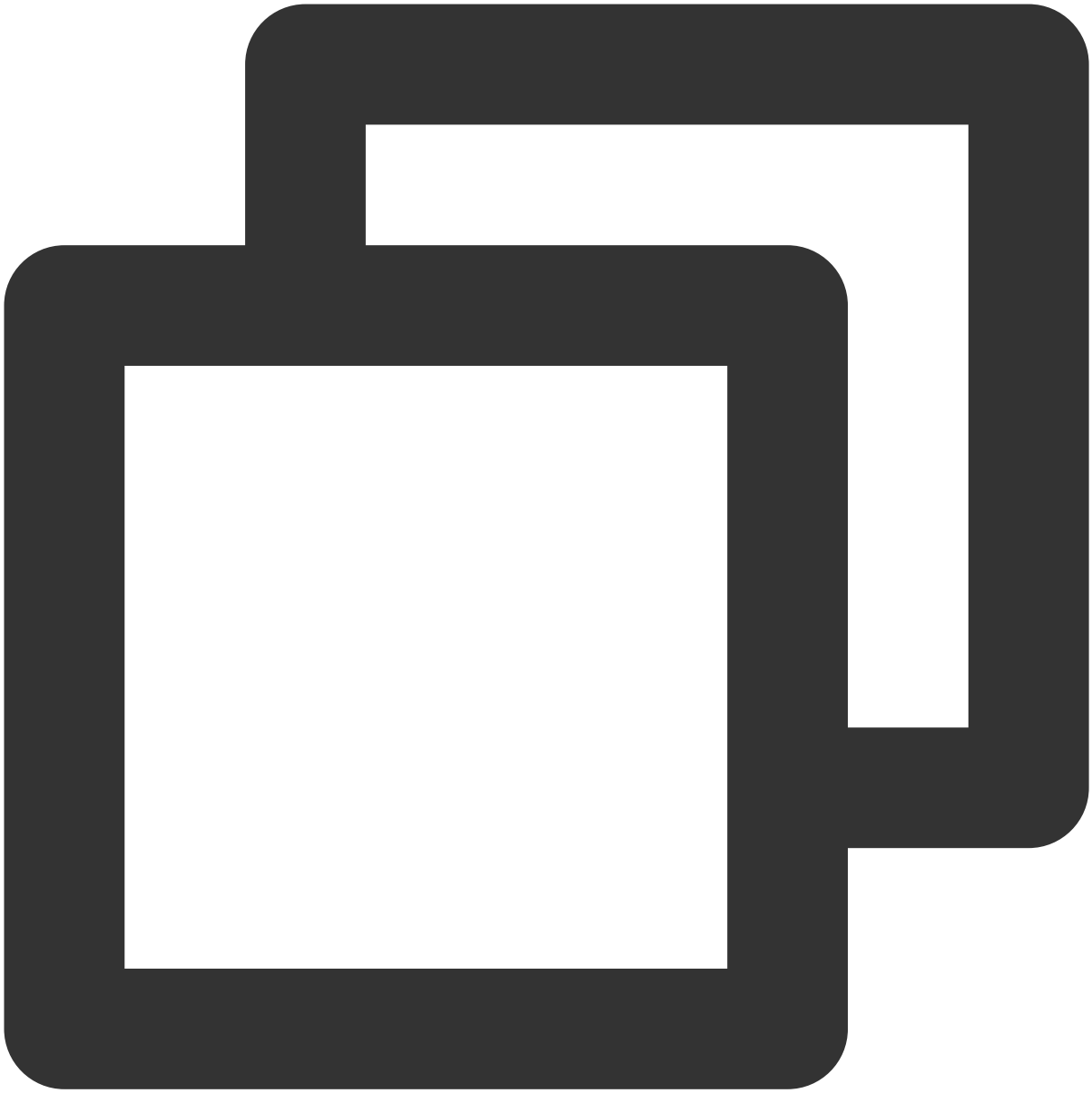
```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
    if (authresult == TELicenseCheckOk) {
        NSLog(@"인증 성공");
    } else {
        NSLog(@"인증 실패");
    }
}];
```

errorCode 인증 설명:

에러 코드	설명

0	성공. Success
-1	입력 매개변수가 유효하지 않습니다. 예: URL 또는 KEY가 비어 있습니다.
-3	다운로드에 실패했습니다. 네트워크 설정을 확인하십시오.
-4	로컬 시스템에서 읽은 TE SDK 인증 정보가 비어 있으며, 이는 I/O 실패로 인해 발생할 수 있습니다
-5	읽은 VCUBE TEMP License 파일의 내용이 비어 있으며 이는 I/O 실패로 인해 발생할 수 있습니다
-6	v_cube.license 파일의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-7	서명 확인에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-8	암호 해독에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-9	TELicense 필드의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-10	온라인으로 파싱된 TE SDK 인증 정보가 비어 있습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-11	TE SDK 인증 정보를 로컬 파일에 쓰지 못했습니다. I/O 실패로 인해 발생할 수 있습니다.
-12	다운로드에 실패했으며 로컬 asset을 파싱하지 못했습니다.
-13	인증 실패
기타	도움이 필요한 경우 Tencent Cloud에 문의하십시오

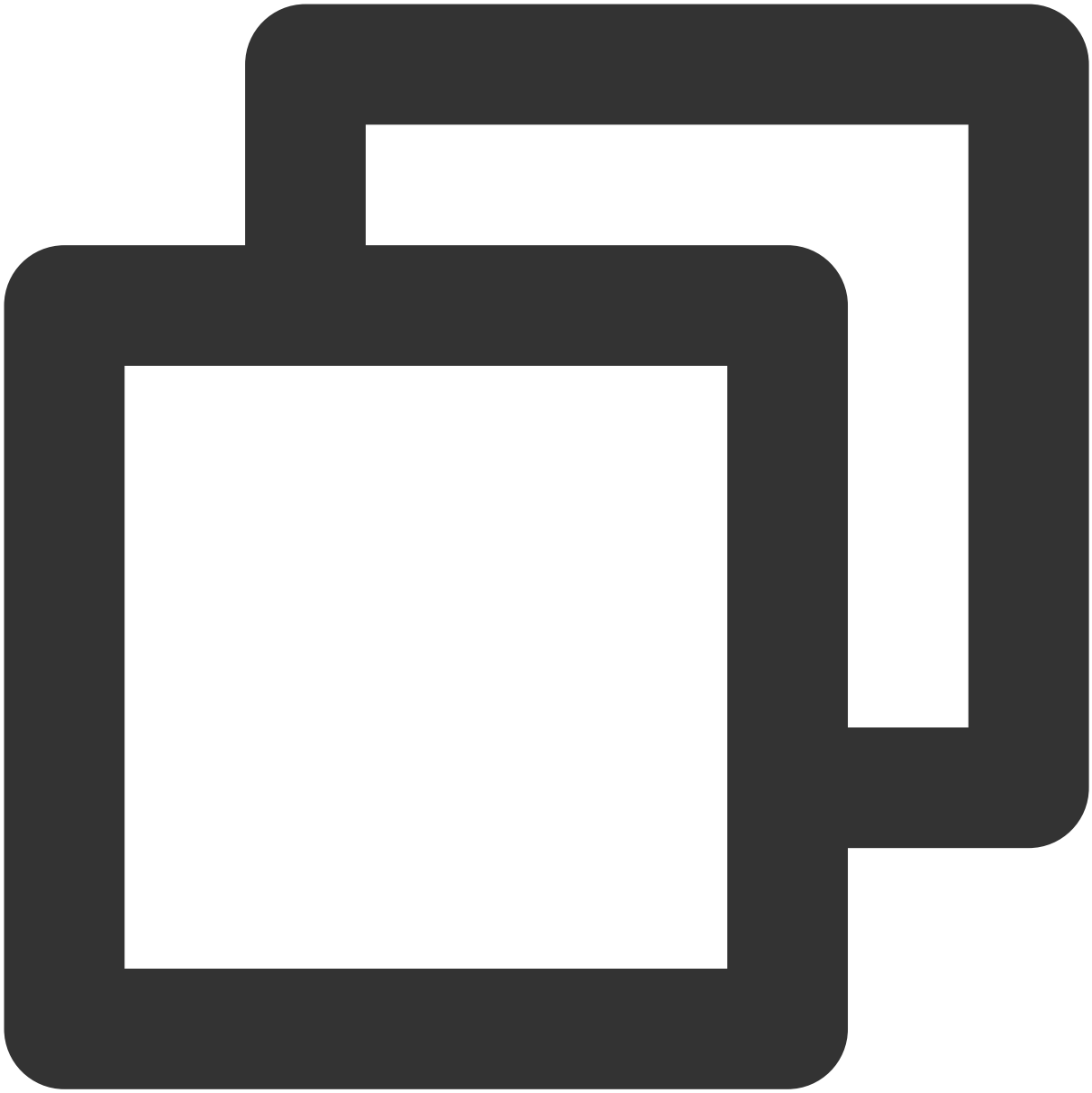
2단계: SDK 소재 리소스 경로 설정



```
CGSize previewSize = [self getPreviewSizeByResolution:self.currentPreviewResolution
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
```

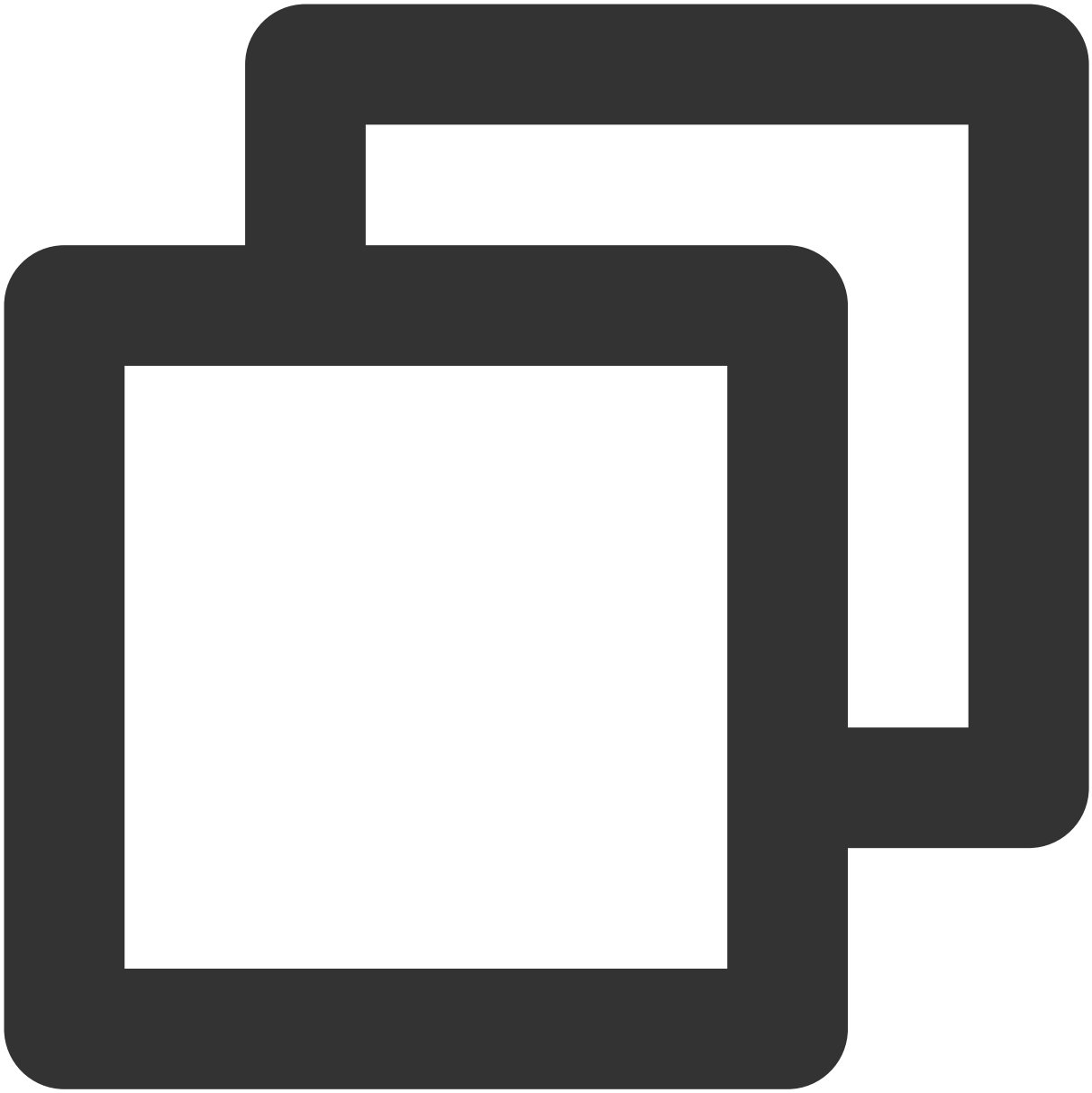
```
                                                options:NSJSONReadingMutableContainers
                                                error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Init beauty kit
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
```

3단계: 로그 및 이벤트 리스너 추가



```
// Register log
[self.beautyKit registerSDKEventListener:self];
[self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL
```

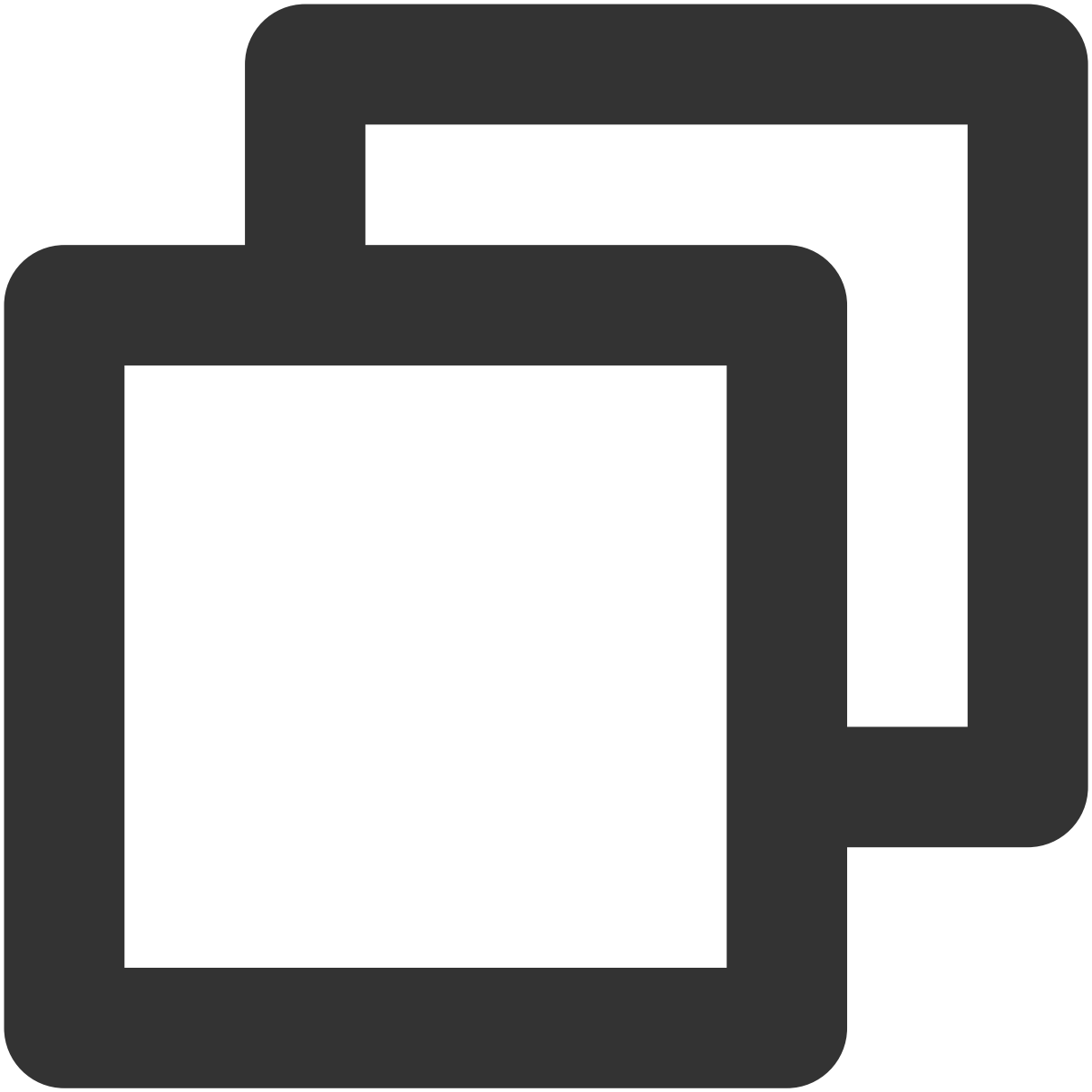
4단계: 뷰티 필터 효과 구성



```
- (int)configPropertyWithType:(NSString *_Nonnull)propertyType withName:(NSString *
```

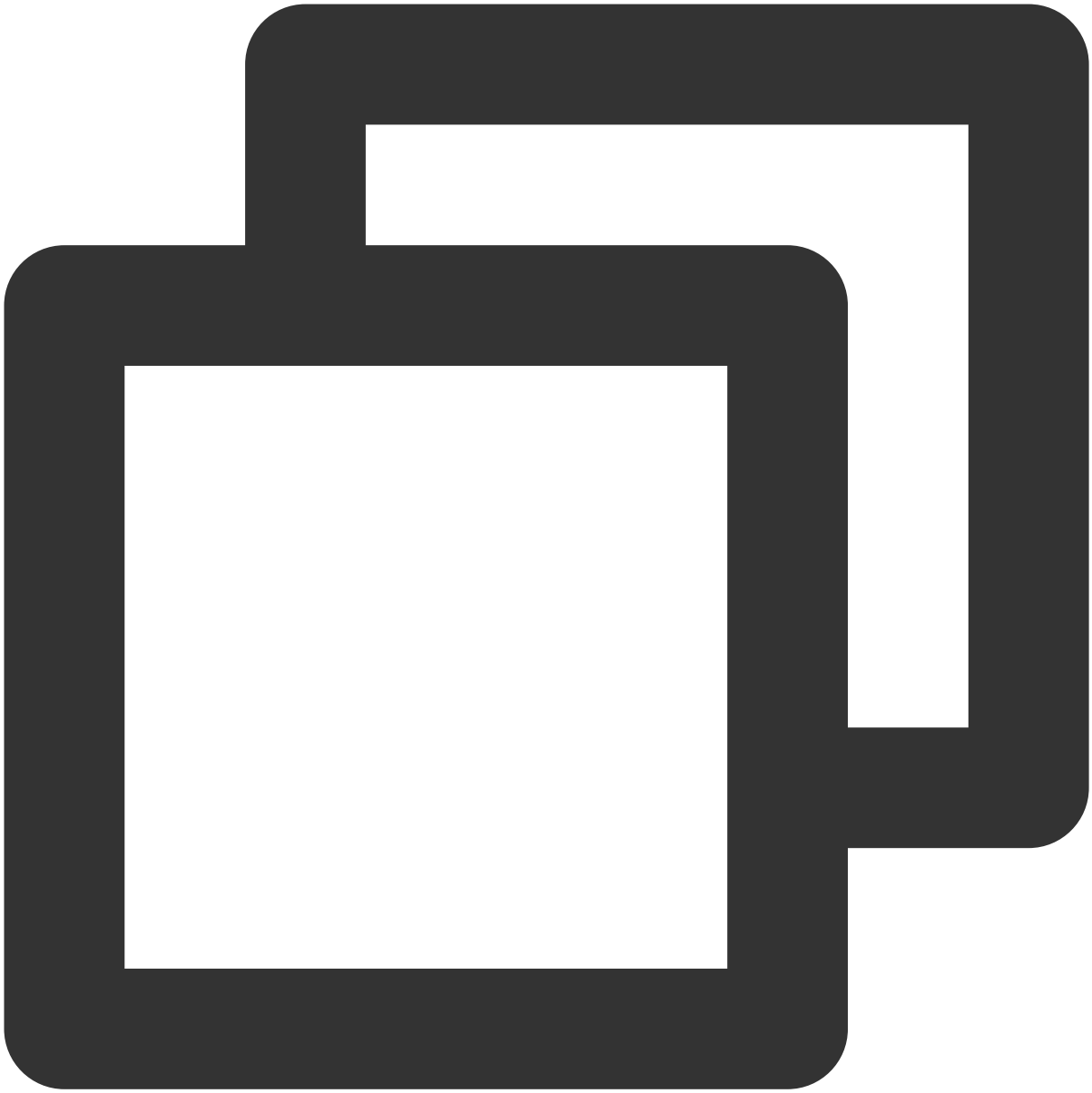
5단계: 비디오 렌더링

비디오 프레임 콜백 인터페이스에서 YTProcessInput을 구성하고 렌더링 처리를 위해 SDK에 전달합니다. Demo의 ThirdBeautyViewController를 참고하십시오.



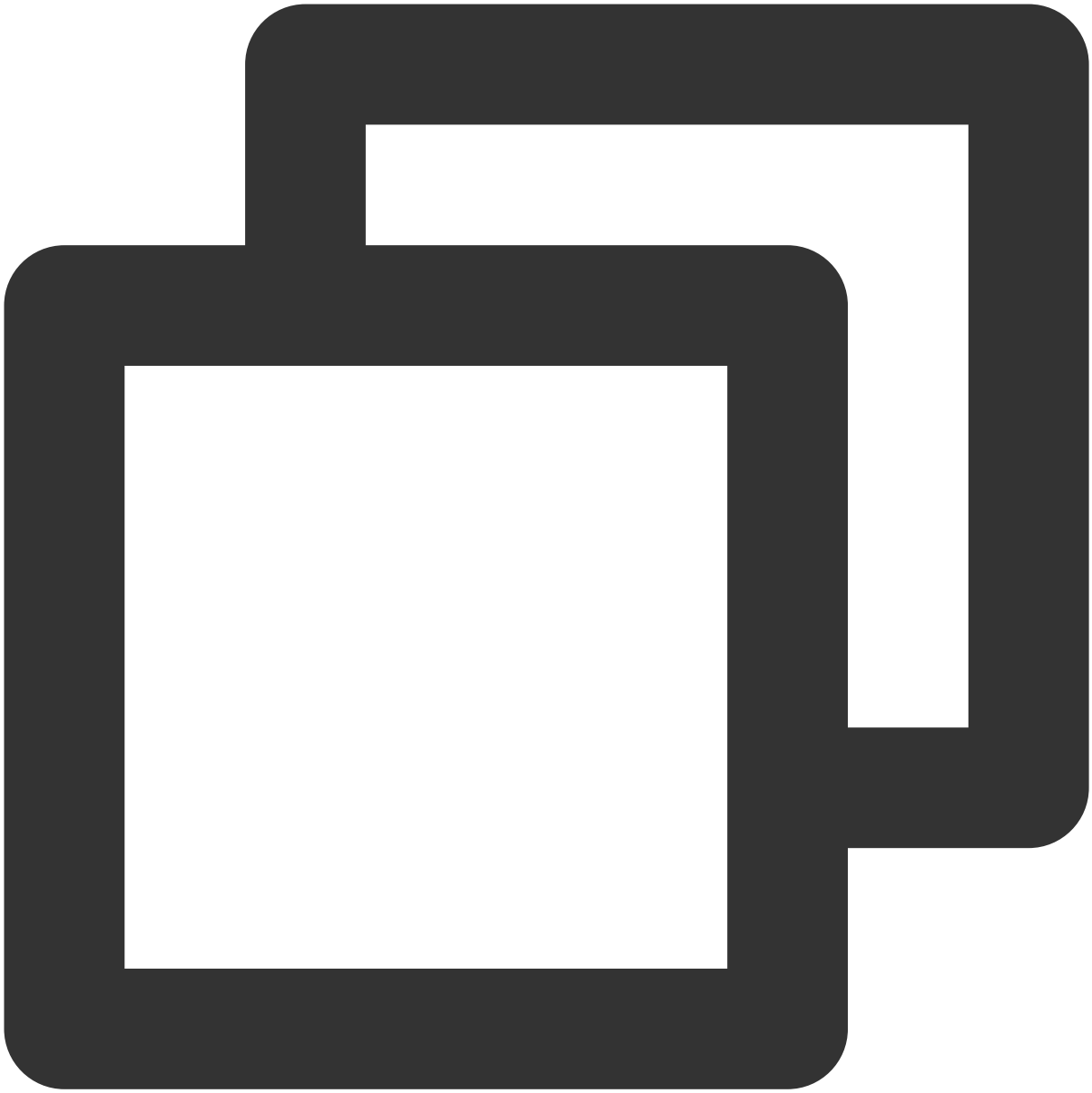
```
[self.xMagicKit process:inputCPU withOrigin:YtLightImageOriginTopLeft withOrientati
```

6단계: SDK 일시 중지/재개



```
[self.beautyKit onPause];
[self.beautyKit onResume];
```

7단계: 레이아웃에 SDK 뷰티 필터 패널 추가



```
UIEdgeInsets gSafeInset;
#if __IPHONE_11_0 && __IPHONE_OS_VERSION_MAX_ALLOWED >= __IPHONE_11_0
if(gSafeInset.bottom > 0){
}
if (@available(iOS 11.0, *)) {
    gSafeInset = [UIApplication sharedApplication].keyWindow.safeAreaInsets;
} else
#endif
    {
        gSafeInset = UIEdgeInsetsZero;
    }
```

```
dispatch_async(dispatch_get_main_queue(), ^{
    //뷰티 필터 옵션 UI
    _vBeauty = [[BeautyView alloc] init];
    [self.view addSubview:_vBeauty];
    [_vBeauty mas_makeConstraints:^(MASConstraintMaker *make) {
        make.width.mas_equalTo(self.view);
        make.centerX.mas_equalTo(self.view);
        make.height.mas_equalTo(254);
        if(gSafeInset.bottom > 0.0){  // 전체 화면으로 조정
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(0);
        } else {
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(-10);
        }
    }];
    _vBeauty.hidden = YES;
});
```

Android

최종 업데이트 날짜: : 2022-07-18 10:06:17

1단계: Demo 프로젝트 압축 해제

1. Tencent Effect(TE) SDK와 통합된 MLVB Demo를 다운로드합니다. 이 Demo는 Tencent Effect SDK S1-04 에디션을 기반으로 제작되었습니다.
2. Demo의 SDK 파일을 실제로 사용하는 SDK용 파일로 교체합니다. 구체적 작업은 다음 단계를 따르십시오.
   - xmagic 모듈의 libs 디렉터리에 있는 `.aar` 파일을 SDK의 libs에 있는 `.aar` 파일로 교체하십시오.
   - xmagic 모듈의 `../src/main/assets` 에 있는 모든 파일을 SDK의 `assets/` 에 있는 파일로 교체합니다. SDK 패키지의 MotionRes 폴더에 파일이 있는 경우 `../src/main/assets` 디렉터리에도 복사합니다.
   - xmagic 모듈의 `../src/main/jniLibs` 에 있는 모든 .so 파일을 SDK 패키지의 jniLibs에 있는 .so 파일로 교체합니다(arm64-v8a 및 armeabi-v7a에 대한 `.so` 파일을 얻으려면 jinLibs 폴더에 있는 ZIP 파일의 압축을 풀어야 합니다).
3. Demo 프로젝트의 xmagic 모듈을 실제 항목 프로젝트로 가져옵니다.

2단계: app 모듈의 build.gradle 열기

1. applicationId를 신청된 테스트 인증과 일치하는 패키지 이름으로 수정합니다.
2. gson 종속성 설정을 추가합니다.

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

3단계: SDK 인터페이스 통합

Demo 프로젝트의 ThirdBeautyActivity 클래스를 참고하십시오.

1. **인증**:

```
//인증 시 주의사항 및 오류 코드 내용은 https://www.tencentcloud.com/document/product/1143/45385#step-1.-authenticate을 참고하십시오
XMagicImpl.checkAuth((errorCode, msg) -> {
if (errorCode == TELicenseCheck.ERROR_OK) {
showLoadResourceView();
```

```
} else {
TXCLog.e(TAG, "인증 실패, 인증 url 및 key를 확인하십시오" + errorCode + " " + msg);
}
});
```

2. 소재 초기화:

```
private void showLoadResourceView() {
if (XmagicLoadAssetsView.isCopyedRes) {
XmagicResParser.parseRes(getApplicationContext());
initXMagic();
} else {
XmagicLoadAssetsView loadAssetsView = new XmagicLoadAssetsView(this);
loadAssetsView.setOnAssetsLoadFinishListener(() -> {
XmagicResParser.parseRes(getApplicationContext());
initXMagic();
});
}
}
```

3. 푸시 설정 실행:

```
String userId = String.valueOf(new Random().nextInt(10000));
String pushUrl = AddressUtils.generatePushUrl(streamId, userId, 0);
mLivePusher = new V2TXLivePusherImpl(this, V2TXLiveDef.V2TXLiveMode.TXLiveMode_
RTC);
mLivePusher.enableCustomVideoProcess(true, V2TXLivePixelFormatTexture2D, V2TXLi
veBufferTypeTexture);
mLivePusher.setObserver(new V2TXLivePusherObserver() {
@Override
public void onGLContextCreated() {
}
@Override
public int onProcessVideoFrame(V2TXLiveDef.V2TXLiveVideoFrame srcFrame, V2TXLiv
eDef.V2TXLiveVideoFrame dstFrame) {
if (mXMagic != null) {
dstFrame.texture.textureId = mXMagic.process(srcFrame.texture.textureId, srcFra
me.width, srcFrame.height);
}
return srcFrame.texture.textureId;
}
@Override
public void onGLContextDestroyed() {
if (mXMagic != null) {
```

```
mXMagic.onDestroy();
}
}
});
mLivePusher.setRenderView(mPushRenderView);
mLivePusher.startCamera(true);
int ret = mLivePusher.startPush(pushUrl);
mLivePusher.startMicrophone();
```

4. **textureId를 SDK에 불러오기 및 렌더링 처리**:

V2TXLivePusherObserver 인터페이스의 `onProcessVideoFrame(V2TXLiveDef.V2TXLiveVideoFrame srcFrame, V2TXLiveDef.V2TXLiveVideoFrame dstFrame)` 메소드에 다음 코드를 추가합니다.

```
if (mXMagic != null) {
dstFrame.texture.textureId = mXMagic.process(srcFrame.texture.textureId, srcFra
me.width,srcFrame.height);
}
return srcFrame.texture.textureId;
```

5. **SDK 일시 중지/폐기**:

`onPause()` 는 뷰티 필터 일시 중지에 사용되며, Activity/Fragment 라이프사이클 메소드에서 실행할 수 있습니다. onDestroy 메소드는 GL 스레드에서 호출해야 합니다. (onTextureDestroyed 메소드에서 XMagicImpl 객체의 `onDestroy()` 호출 가능) 자세한 사용법은 Demo를 참고하십시오.

```
mXMagic.onPause(); //일시 정지, Activity의 onPause 메소드와 연결
mXMagic.onDestroy(); //폐기, GL 스레드에서 호출
```

6. **레이아웃에 SDK 뷰티 필터 패널 추가**:

```
<RelativeLayout
android:id="@+id/livepusher_bp_beauty_pannel"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_above="@+id/ll_edit_info" />
```

7. **패널 초기화**:

```
private void initXMagic() {
if (mXMagic == null) {
mXMagic = new XMagicImpl(this, mBeautyPanelView);
```

```
}else{
mXMagic.onResume();
}
}
```

자세한 내용은 Demo 프로젝트의 `ThirdBeautyActivity.initXMagic();` 메소드를 참고하십시오.

Flutter

최종 업데이트 날짜: : 2023-04-27 16:13:55

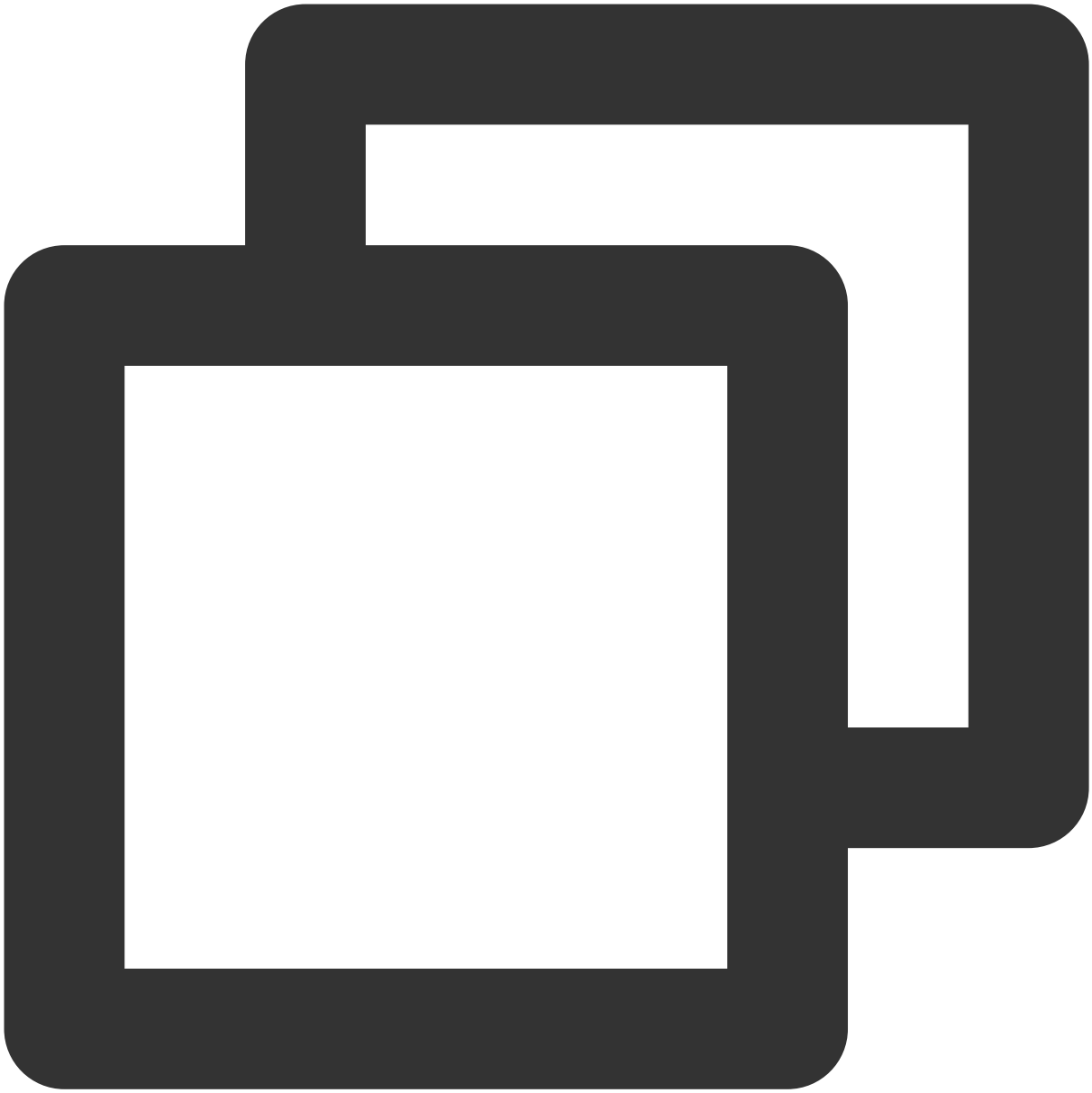
1단계: 효과 리소스 다운로드 및 통합

구매한 패키지에 해당하는 SDK를 다운로드하고 자신의 프로젝트에 리소스 파일을 추가합니다.

Android

iOS

1. app에서 build.gradle을 열고 패키지의 maven 주소를 추가합니다. 예를 들어 S1-04를 구매한 경우 다음을 추가합니다.

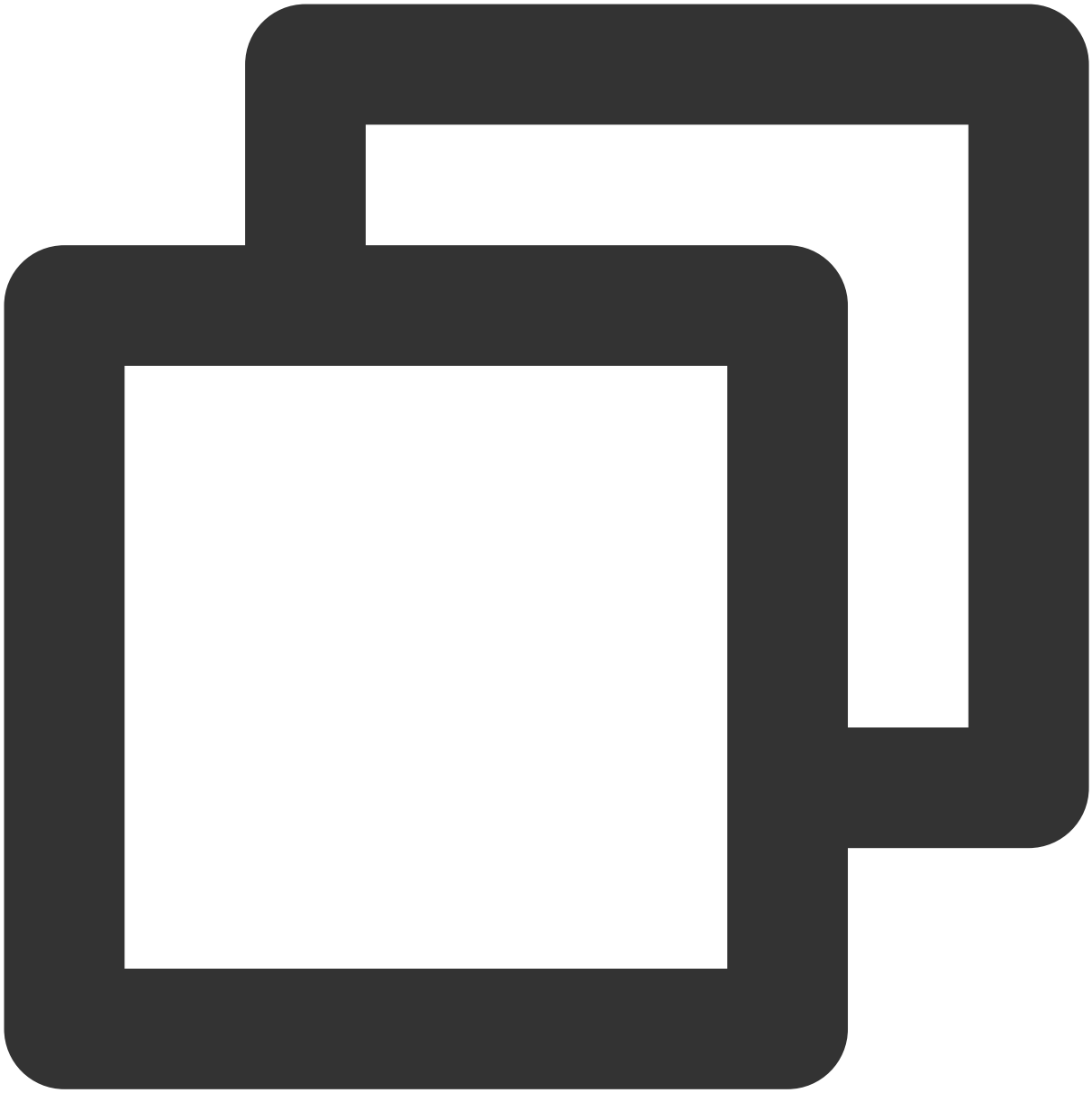


```
dependencies{
        implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
  }
```

문서에서 다양한 패키지의 maven 주소를 찾을 수 있습니다.

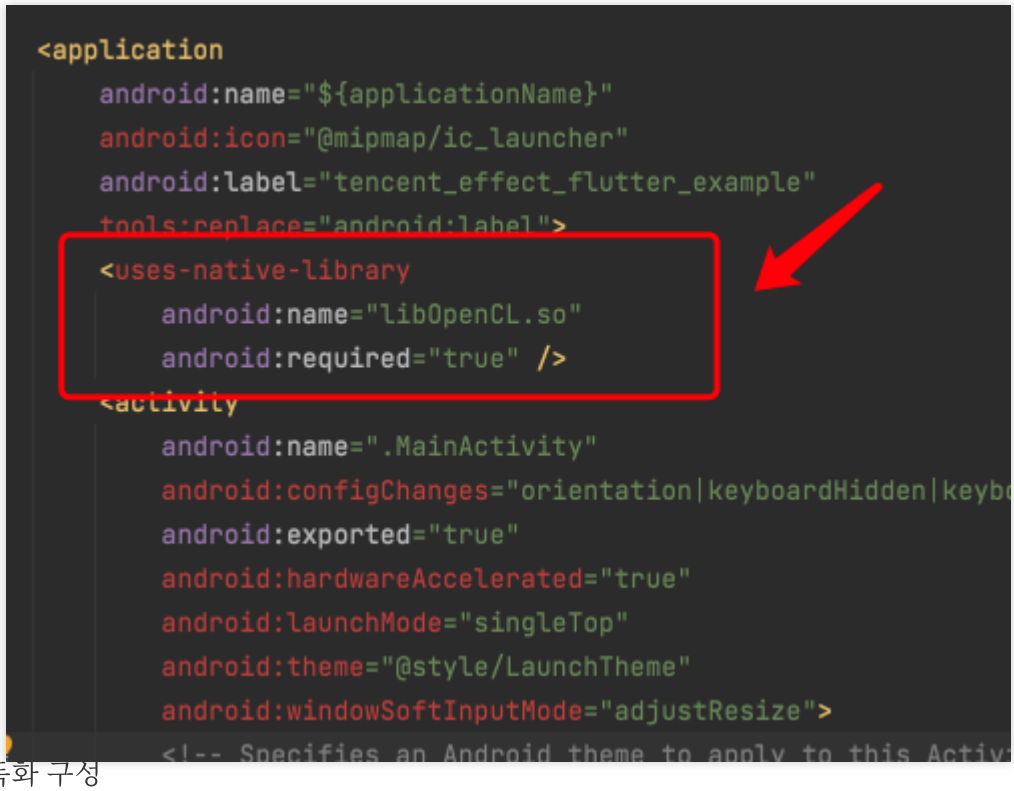
2. app에서 src/main/assets 폴더를 찾습니다(이 폴더가 없으면 새로 생성). MotionRes 폴더가 있는지 확인합니다. `../src/main/assets` 에 복사합니다.

3. app에서 AndroidManifest.xml을 열고 application에 다음 태그를 추가합니다.



```
<uses-native-library
                android:name="libOpenCL.so"
                android:required="true" />
        //여기서 true는 이 라이브러리가 없으면 애플리케이션이 제대로 작동하지 않음을 의미합니다
        //false는 애플리케이션이 이 라이브러리(있는 경우)를 사용할 수 있지만 특히 라이브러리 없
        //Android 공식 웹사이트 소개: %!s(<nil>)
```

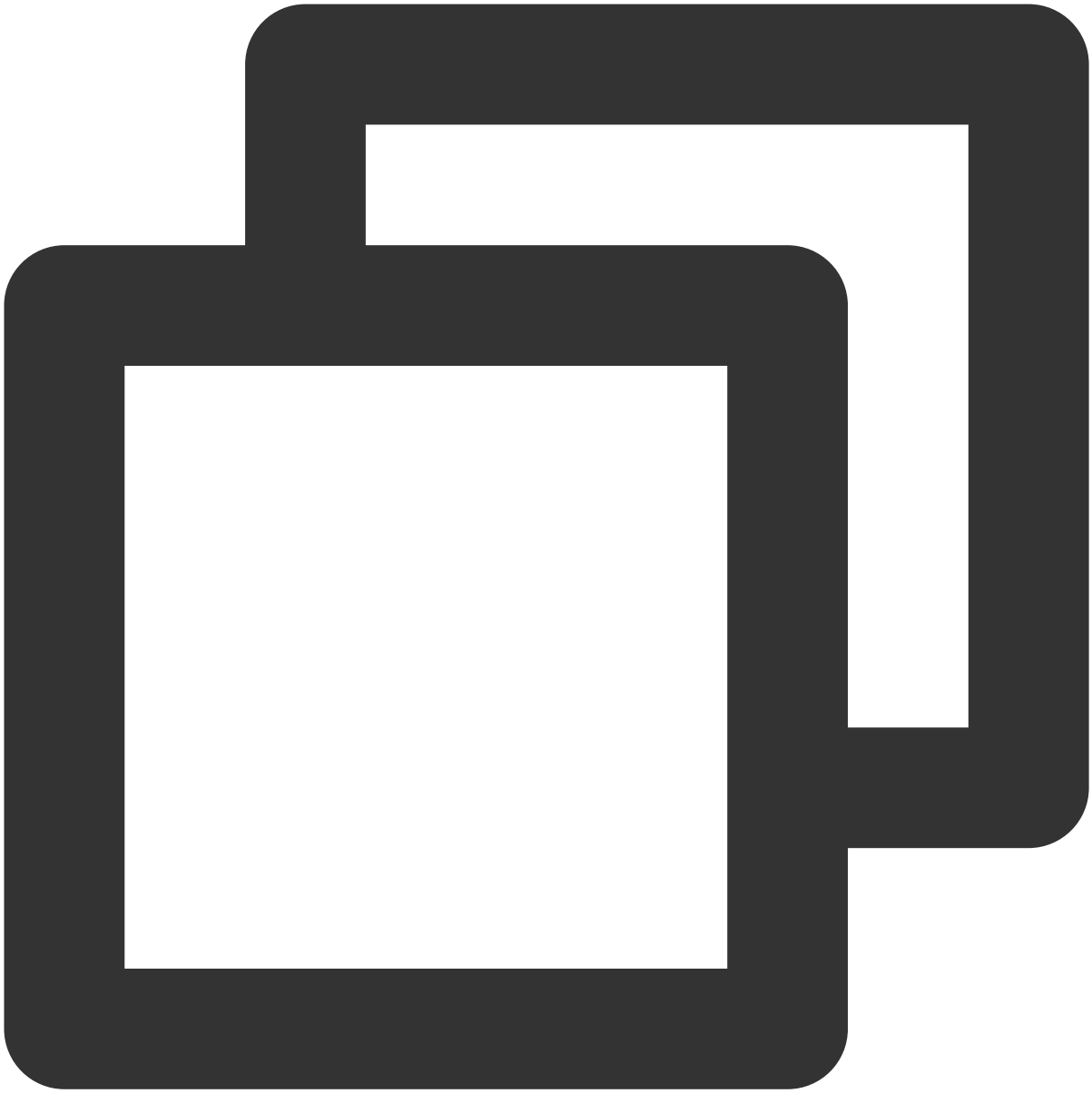
다음과 같이 표시됩니다.



4. 난독화 구성

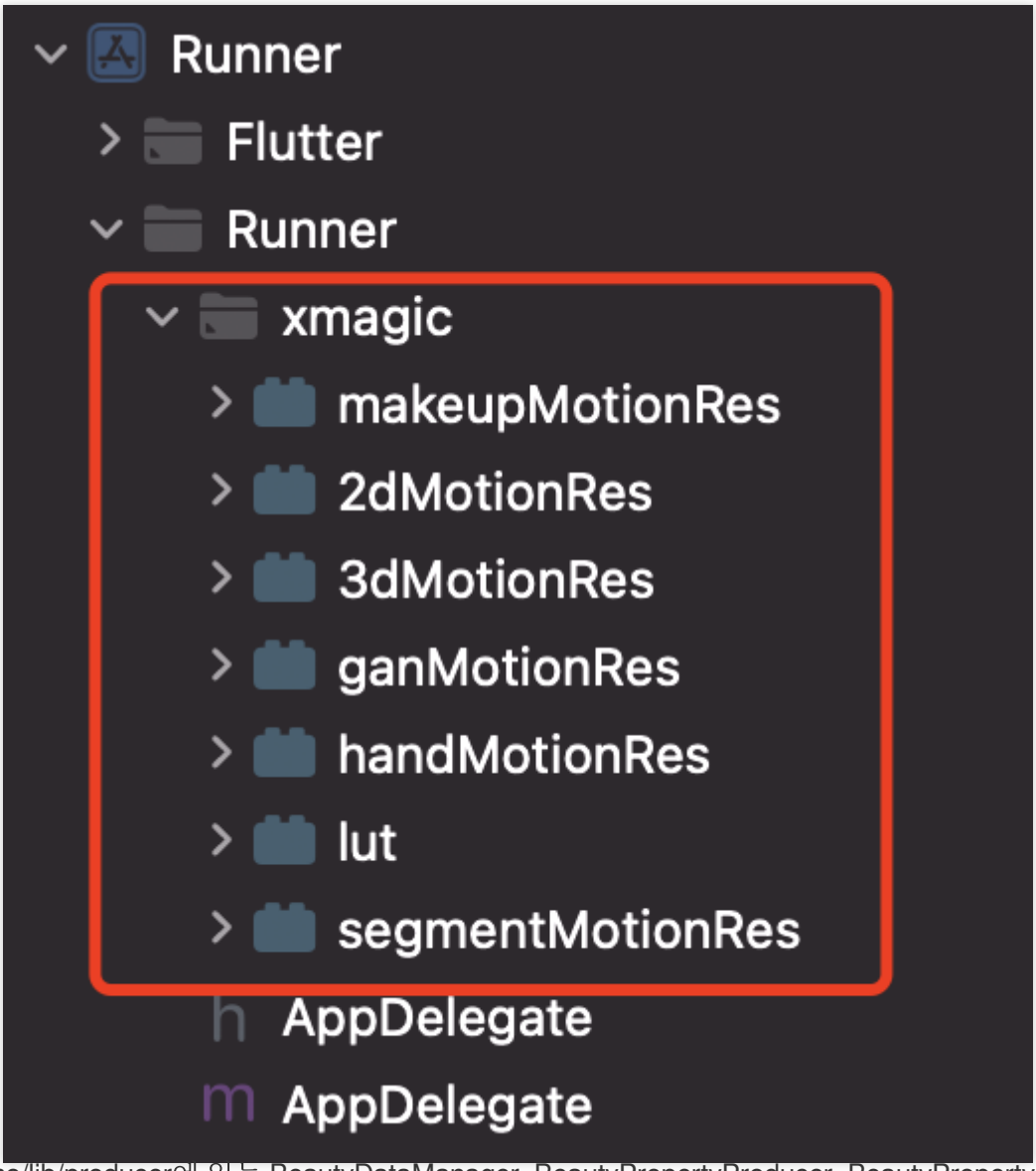
release 패키지를 인쇄할 때 컴파일 최적화(minifyEnabled를 true로 설정)를 활성화한 경우 java 레이어에서 호출되지 않은 일부 코드가 잘려서 native 레이어에서 호출되어 `no xxx method` 예외가 발생할 수 있습니다.

이러한 컴파일 최적화를 활성화한 경우 xmagic 코드가 잘리지 않도록 다음 keep 규칙을 추가합니다.



```
-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.** { *;}
```

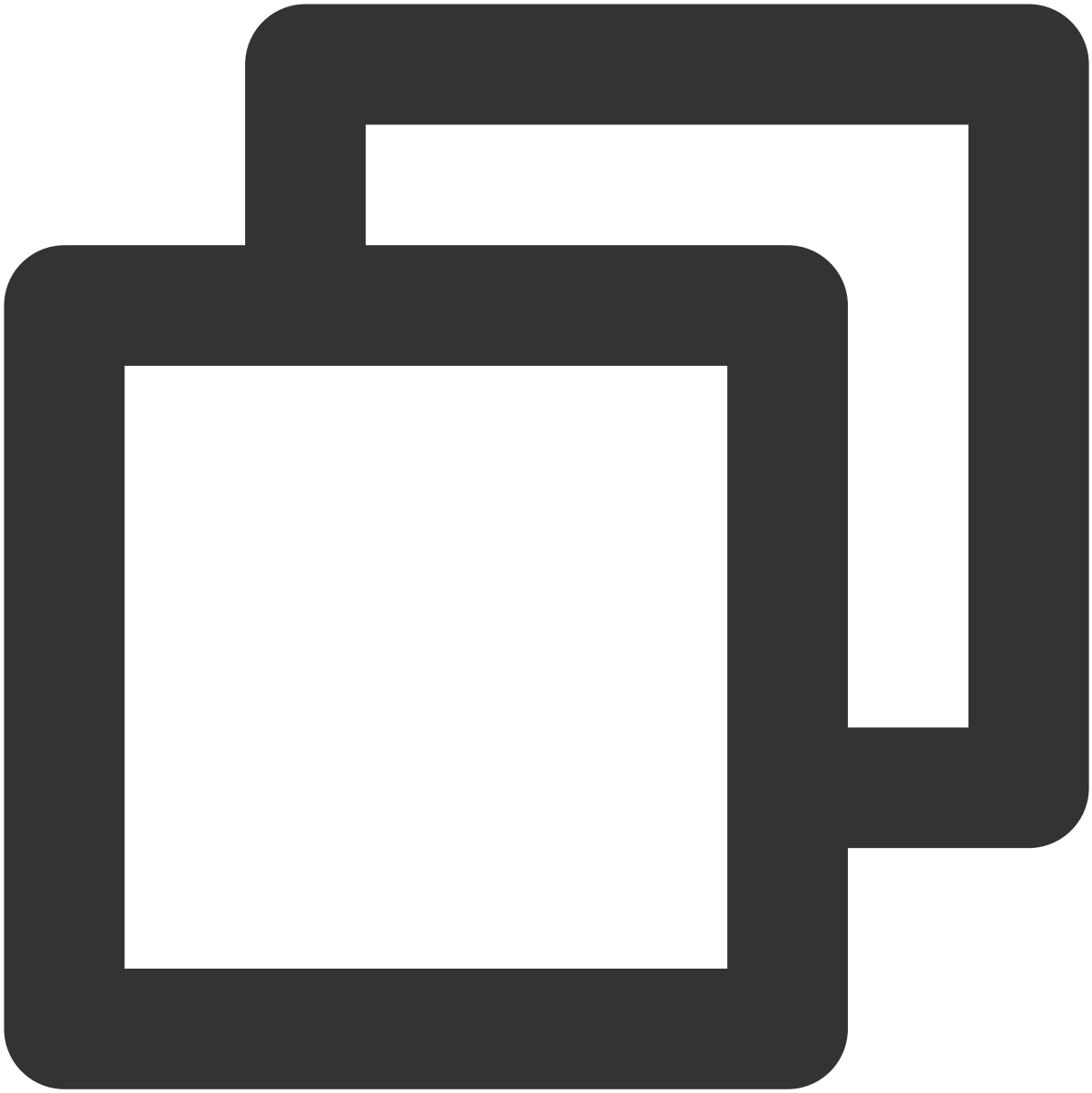
1. **프로젝트에 효과 리소스 추가**(리소스는 스크린샷과 다를 수 있음):



2. demo/lib/producer에 있는 BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid 및 BeautyPropertyProducerIOS의 네 가지 클래스를 Flutter 프로젝트에 복사합니다. 효과 패널에서 효과 리소스 및 표시 효과 옵션을 구성하는 데 사용됩니다.

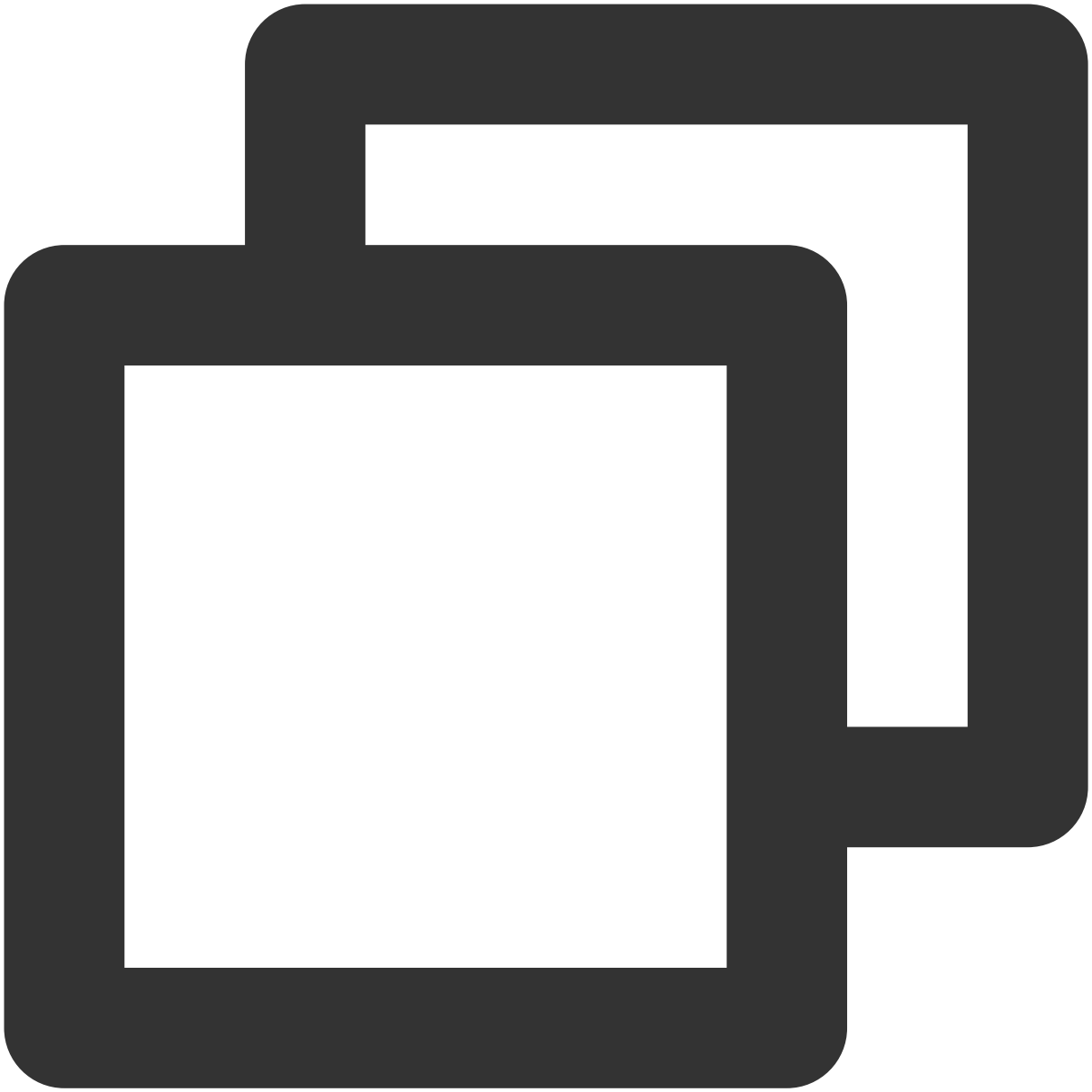
2단계: Flutter SDK 참조

프로젝트의 pubspec.yaml 파일에 다음 참조를 추가하십시오.



```
tencent_effect_flutter:
    git:
        url: https://github.com/TencentCloud/tencenteffect-sdk-flutter
```

로컬 참조: tencent_effect_flutter 최신 버전 tencent_effect_flutter를 다운로드하여 android, ios, lib 폴더와 pubspec.yaml 및 tencent_effect_flutter.iml 파일을 프로젝트 디렉터리로 복사한 다음, 프로젝트의 pubspec.yaml 파일에 다음 코드를 추가합니다. (demo 참고)



```
tencent_effect_flutter:
    path: ../
```

tencent_effect_flutter는 브릿지 역할만 합니다. 효과를 구현하는 것은 XMagic입니다. 기본적으로 최신 버전의 XMagic이 사용됩니다.

최신 버전의 뷰티필터 SDK를 사용하려면, 다음 방법을 사용하여 SDK를 업데이트할 수 있습니다.

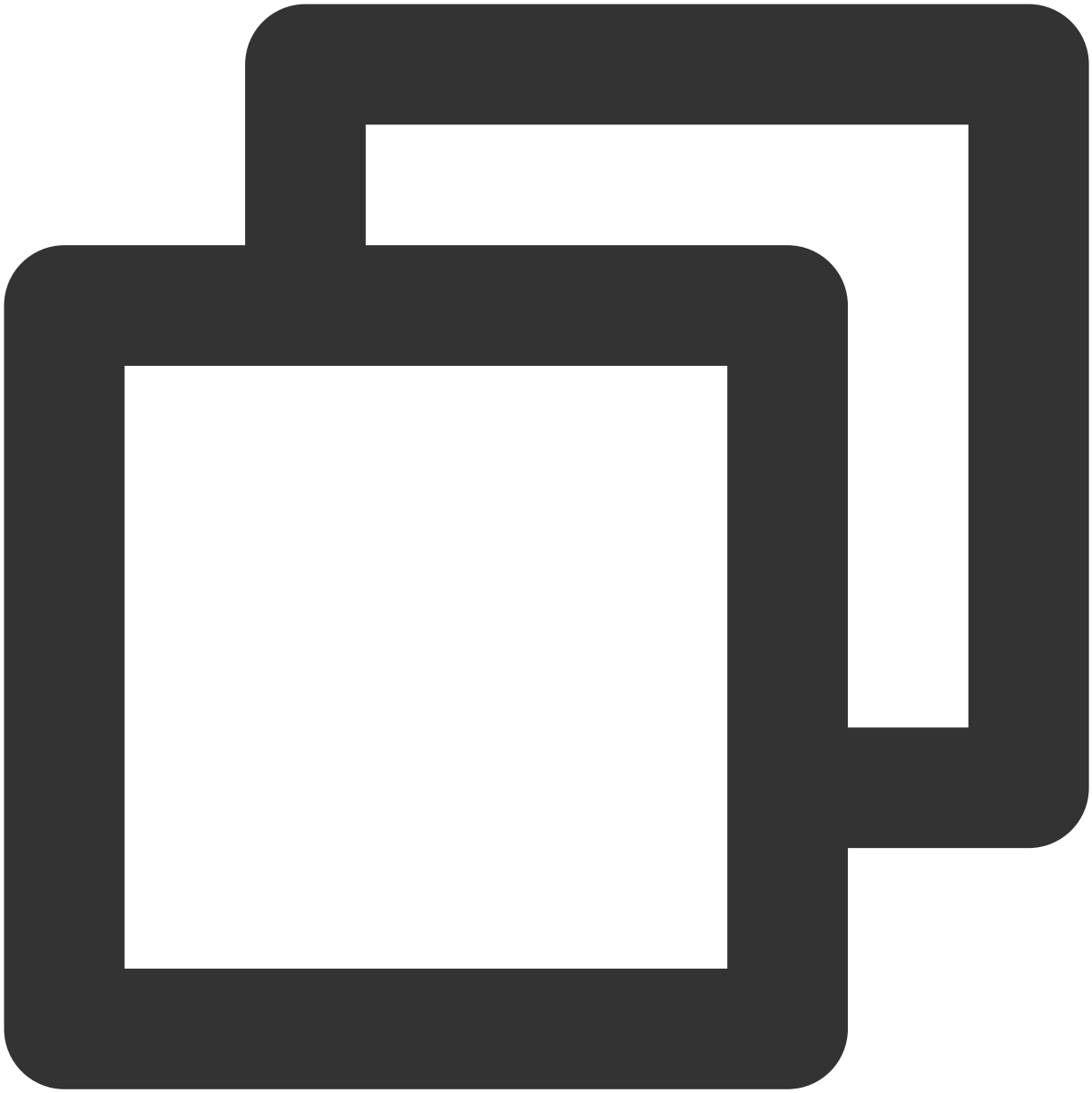
Android

iOS

프로젝트 디렉터리에서 flutter pub upgrade를 실행하거나 subspec.yaml 페이지의 오른쪽 상단에 있는 Pub upgrade를 클릭합니다.
프로젝트 디렉터리에서 flutter pub upgrade를 실행한 다음 ios 디렉터리에서 pod update를 실행합니다.

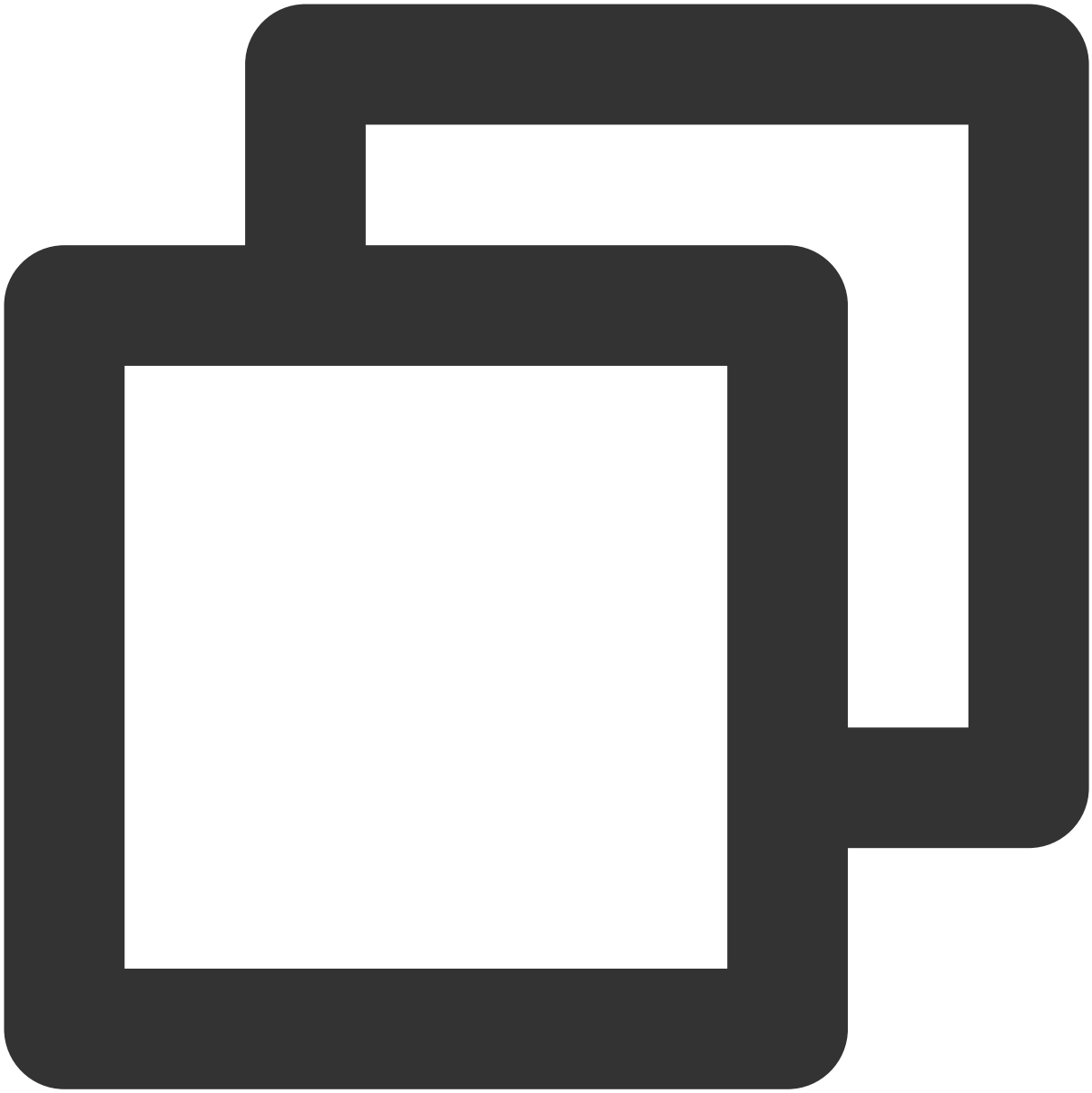
3단계: MLVB와 Tencent Effect 결합

Android
iOS
다음 코드를 application 클래스의 oncreate(또는 FlutterActivity의 onCreate)에 추가합니다.



```
TXLivePluginManager.register(new XmagicProcesserFactory());
```

AppDelegate 클래스의 didFinishLaunchingWithOptions에 다음 코드를 추가합니다.

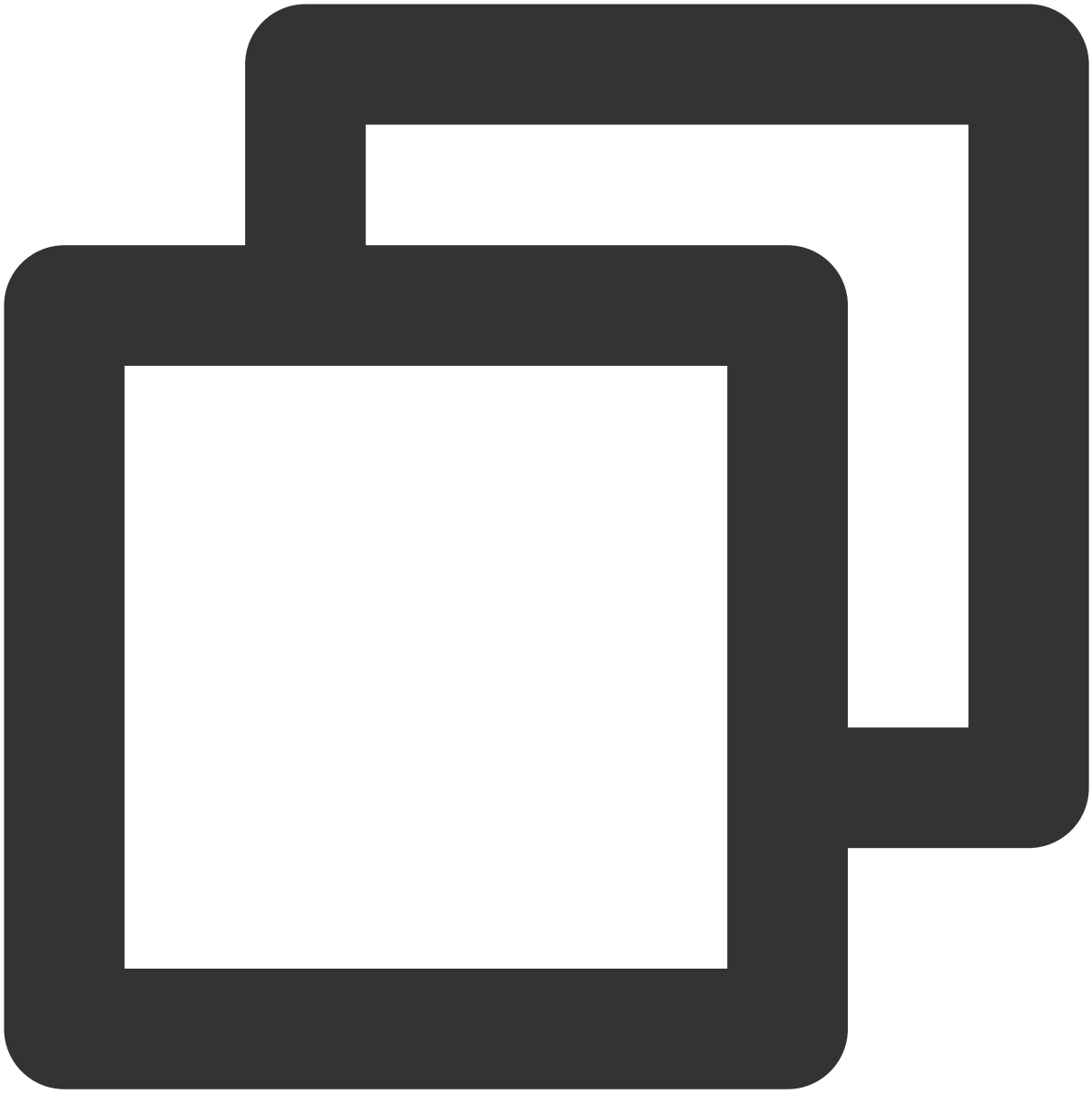


```
XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
[TXLivePluginManager registerWithCustomBeautyProcesserFactory:instance];
```

다음과 같이 표시됩니다.

```objc
#import "AppDelegate.h"
#import "GeneratedPluginRegistrant.h"
@import live_flutter_plugin;
@import tencent_effect_flutter;
@import tencent_trtc_cloud;

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
  XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
  [TXLivePluginManager registerWithCustomBeautyProcesserFactory:instance];
  [TencentTRTCCloud registerWithCustomBeautyProcesserFactory:instance];
  return [super application:application didFinishLaunchingWithOptions:launchO
}

@end
```

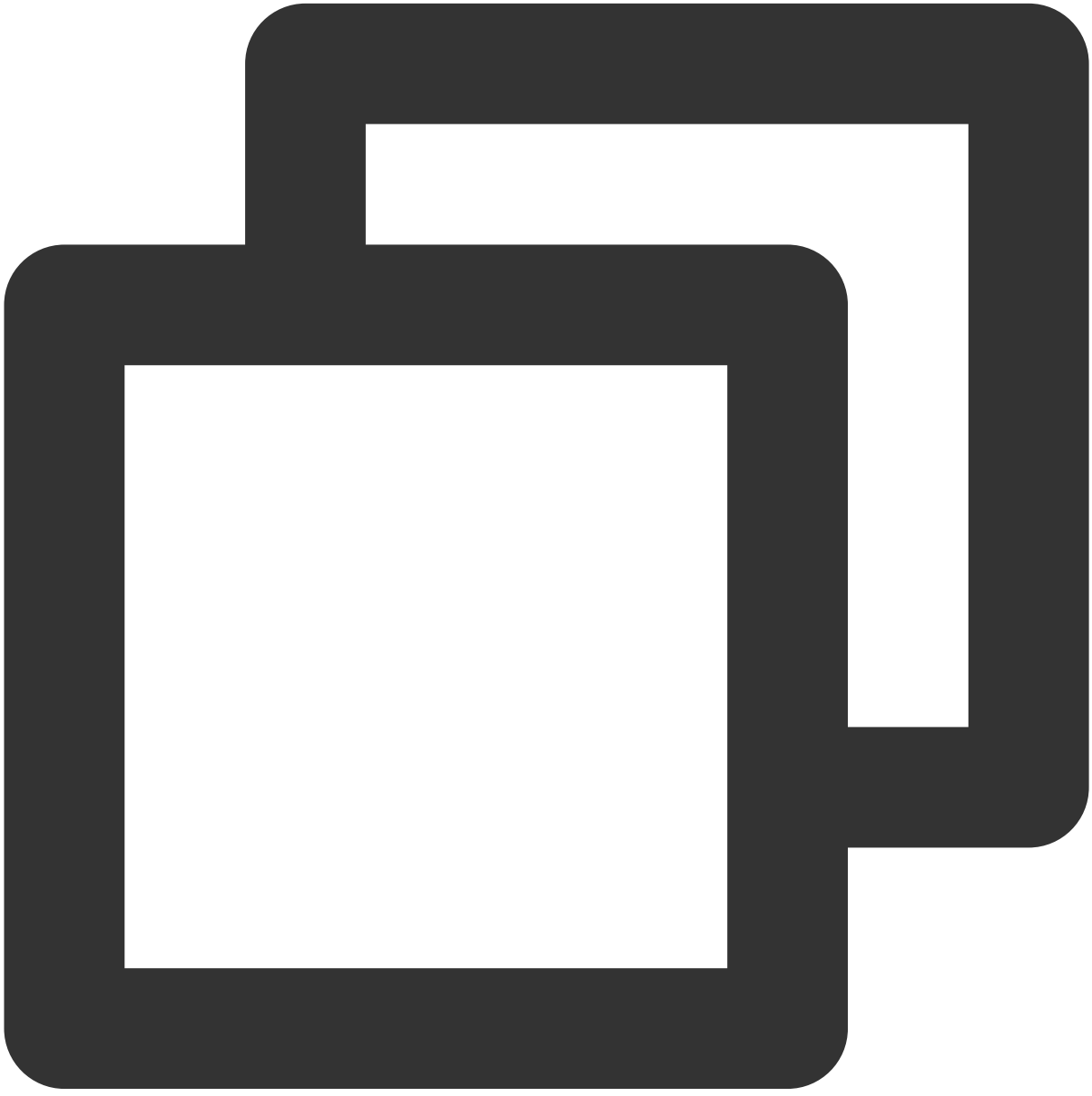
4단계: 리소스 초기화 API 호출



```
String dir =  await BeautyDataManager.getInstance().getResDir();
 TXLog.printlog('파일 경로: $dir');
 TencentEffectApi.getApi()?.initXmagic(dir,(reslut) {
     _isInitResource = reslut;
     callBack.call(reslut);
     if (!reslut) {
         Fluttertoast.showToast(msg: "리소스 초기화 실패");
     }
 }); TencentEffectApi.getApi()?.initXmagic((reslut) {
     if (!reslut) {
         Fluttertoast.showToast(msg: "리소스 초기화 실패");
```

```
        }
    });
```

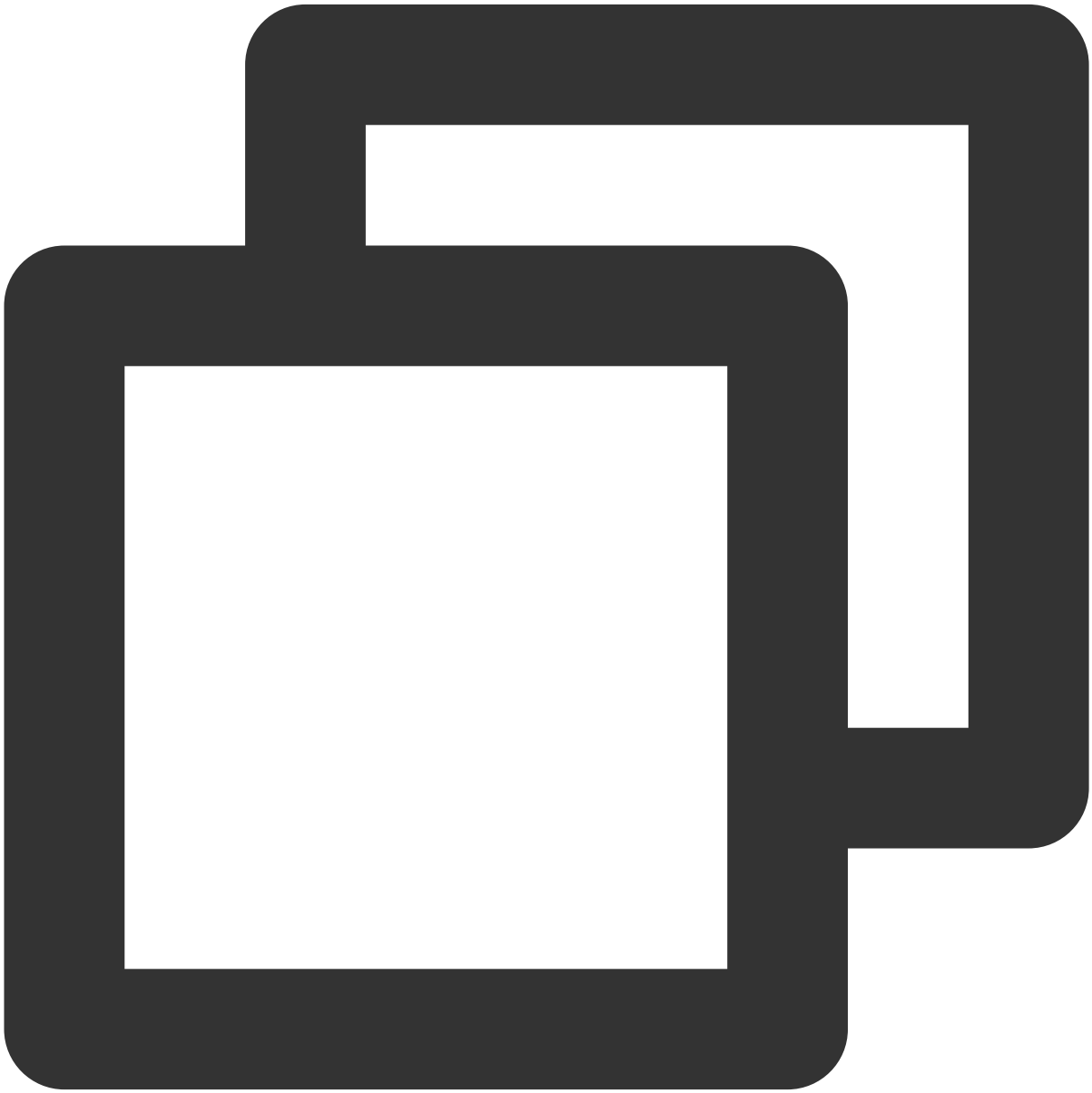
5단계: license 설정



```
TencentEffectApi.getApi()?.setLicense(licenseKey, licenseUrl,
                                      (errorCode, msg) {
                                              TXLog.printlog("인증 결과 출력 errorCode
```

```
                                        if (errorCode == 0) {
                                            //인증 성공
                                        }
    });
```

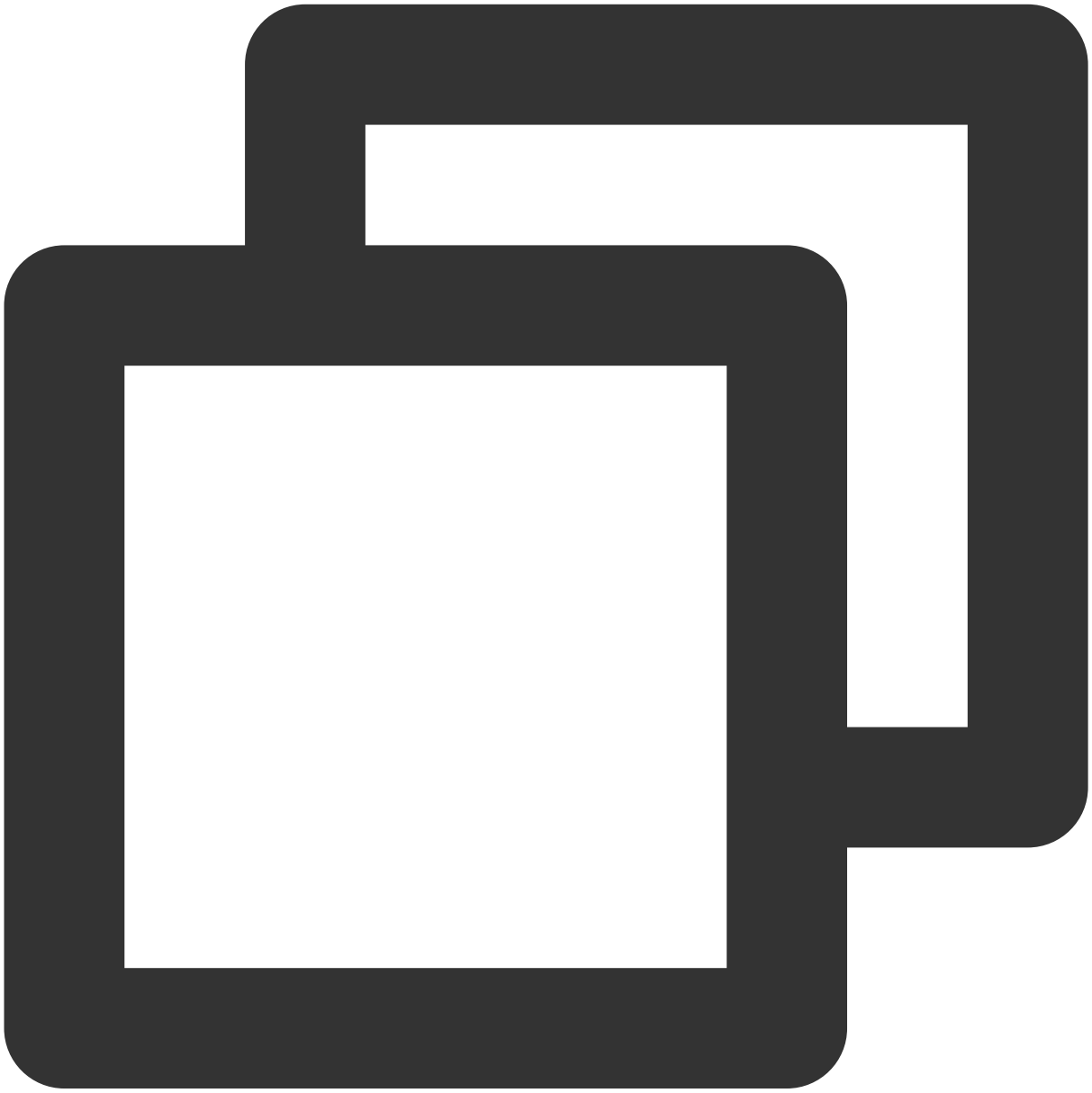
6단계: 뷰티 필터 활성화



```
///뷰티 필터 활성화
```

```
var enableCustomVideo = await _livePusher?.enableCustomVideoProcess(true);
```

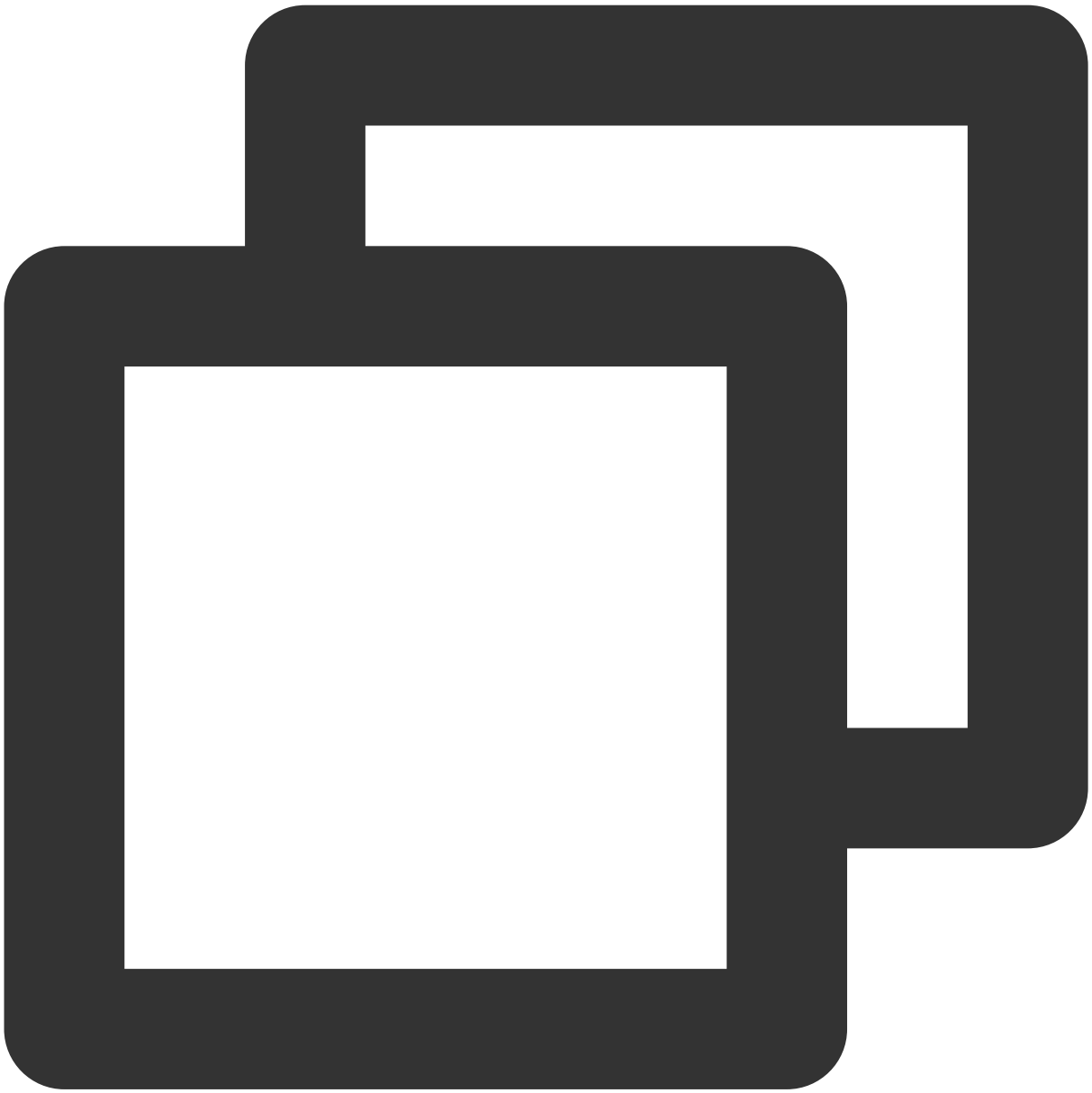
7단계: 뷰티 필터 속성 설정



```
TencentEffectApi.getApi()?.updateProperty(_xmagicProperty!);
///_xmagicProperty는 BeautyDataManager.getInstance().getAllPannelData();를 호출하여 도
```

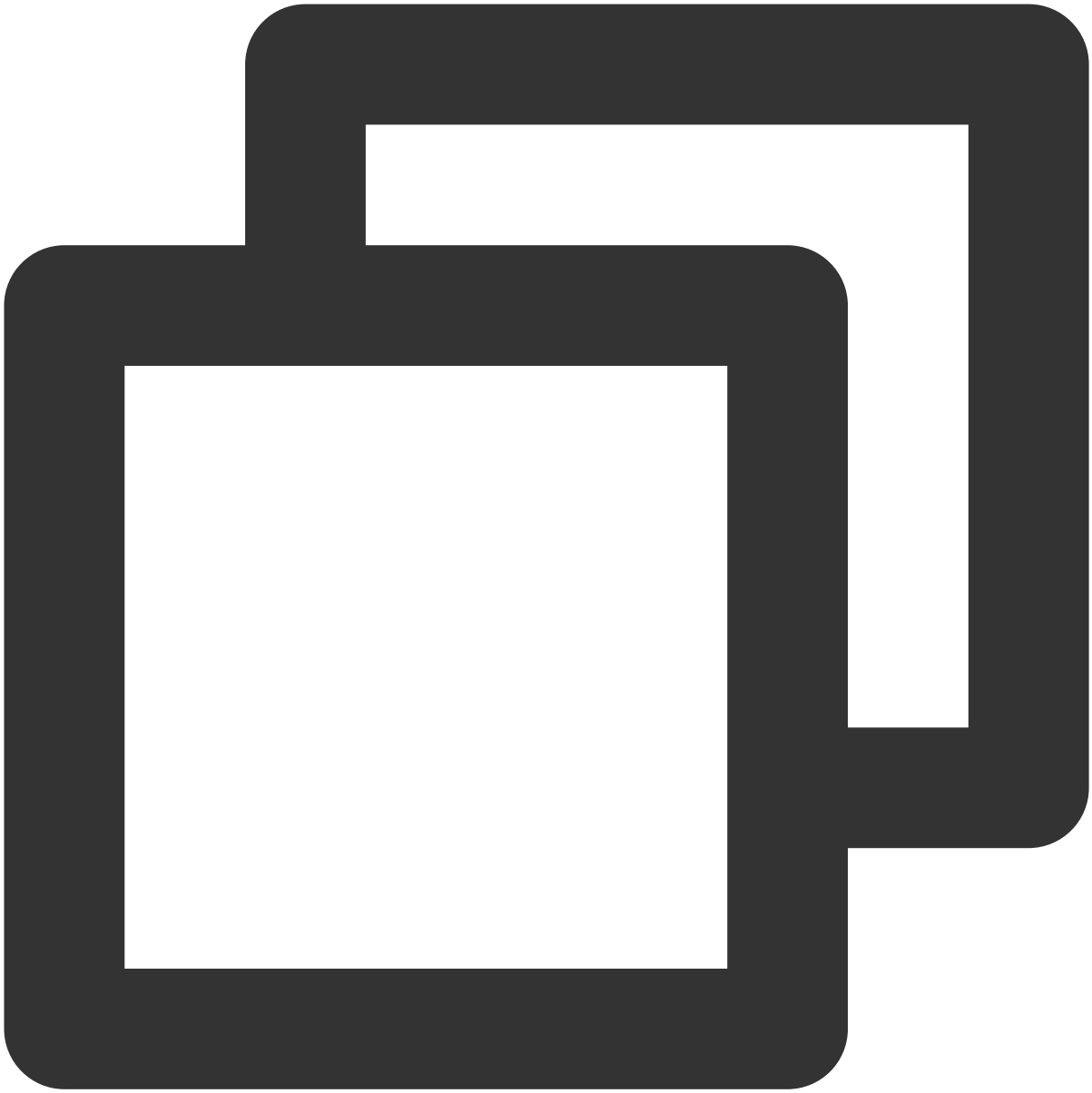
8단계: 기타 속성 설정

오디오 효과 일시 중지



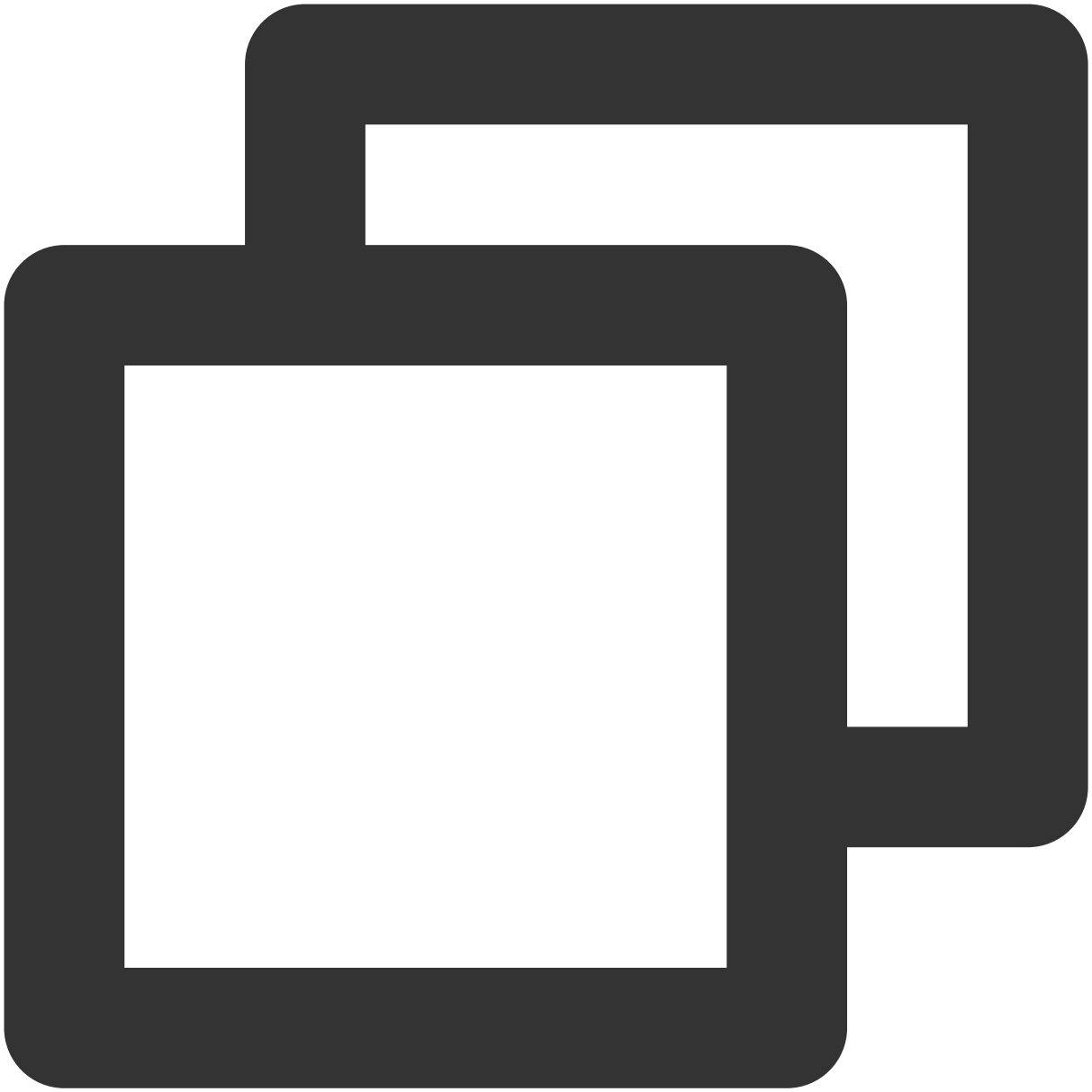
```
TencentEffectApi.getApi()?.onPause();
```

오디오 효과 재개



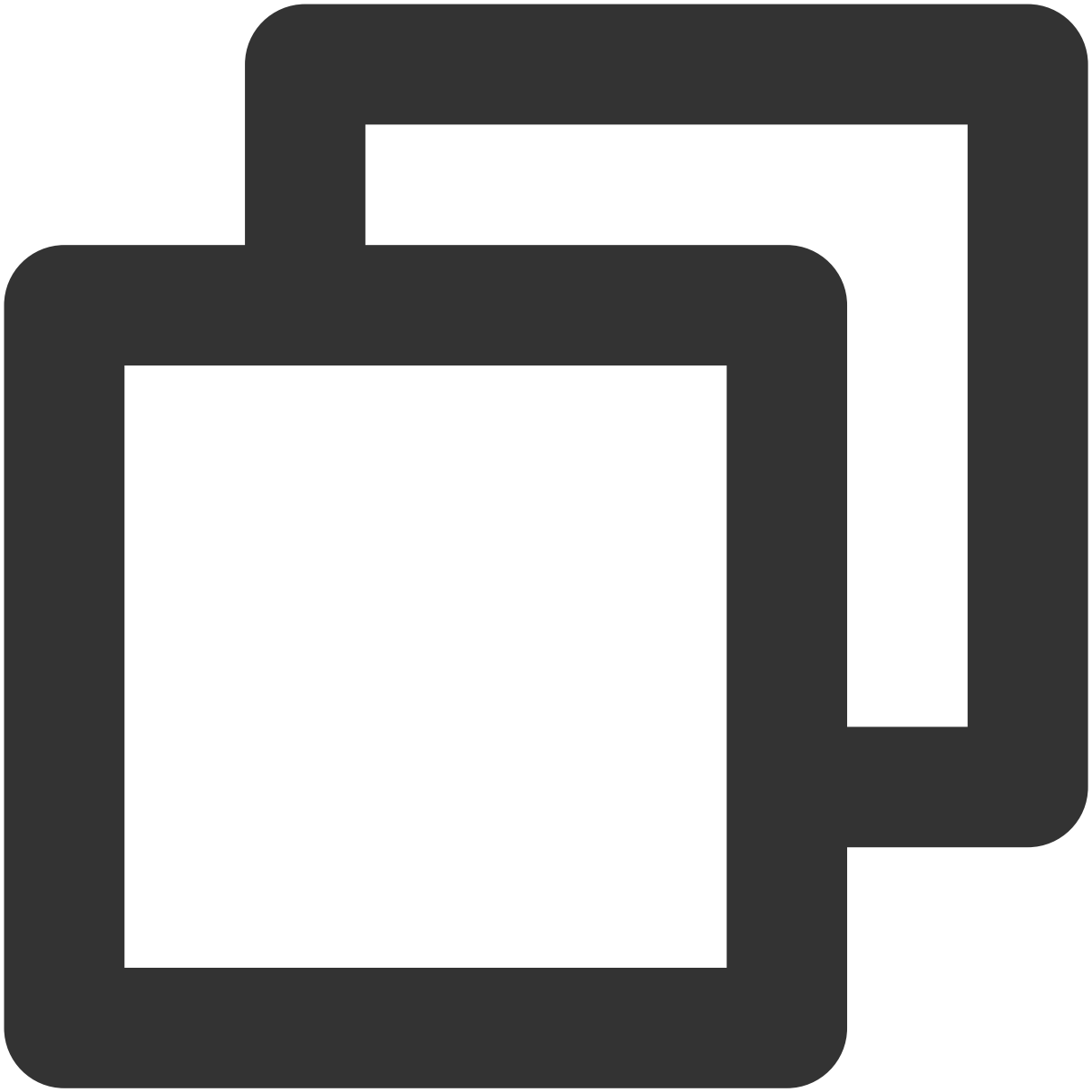
```
TencentEffectApi.getApi()?.onResume();
```

뷰티 필터 이벤트 수신



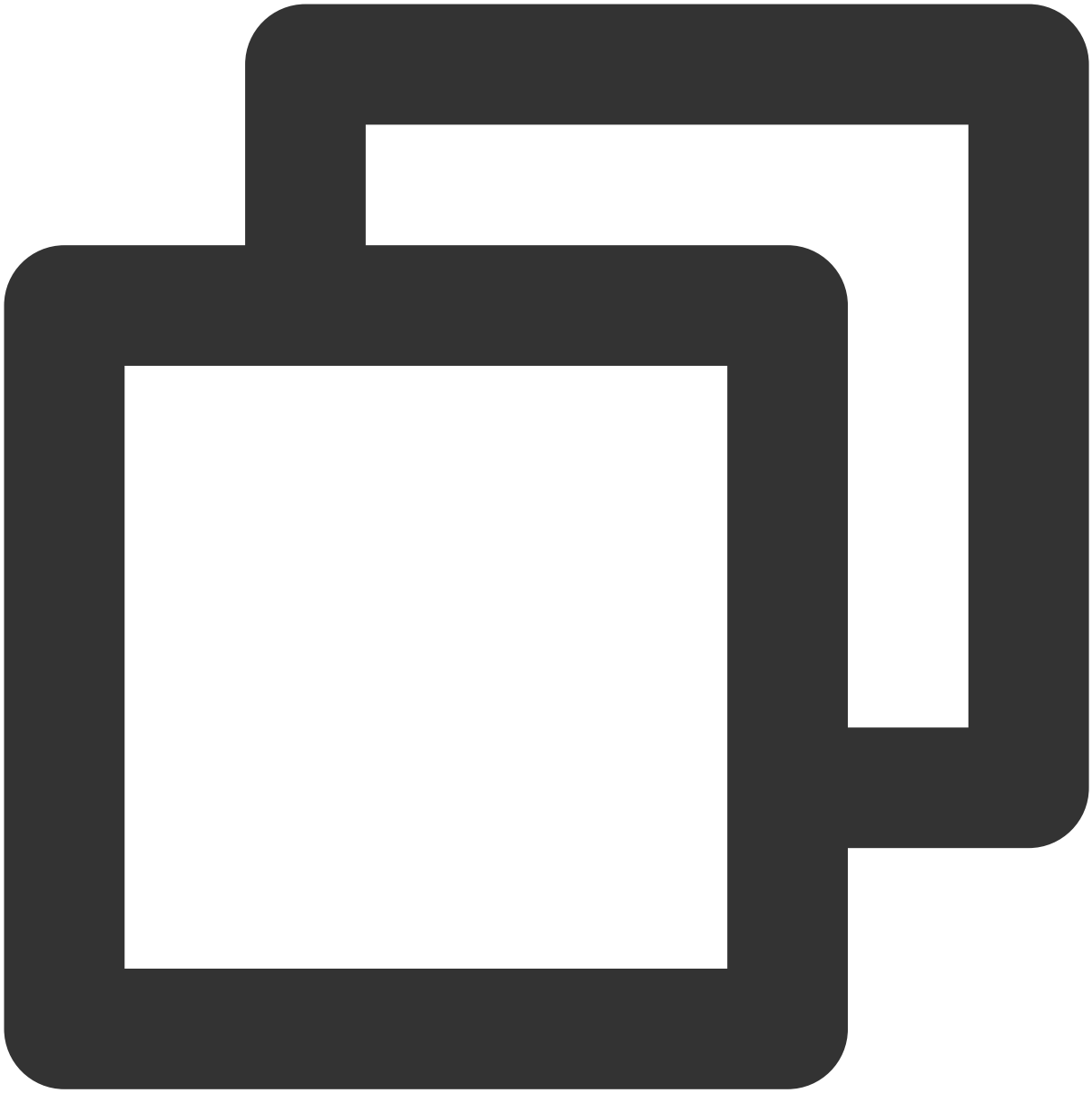
```
TencentEffectApi.getApi()
            ?.setOnCreateXmagicApiErrorListener((errorMsg, code) {
                TXLog.printlog("뷰티 필터 객체 생성 오류 errorMsg = $errorMsg , code = $
    });    ///효과 객체를 생성하기 전에 리스너를 설정해야 함
```

얼굴, 제스처, 신체 감지 결과 콜백 설정



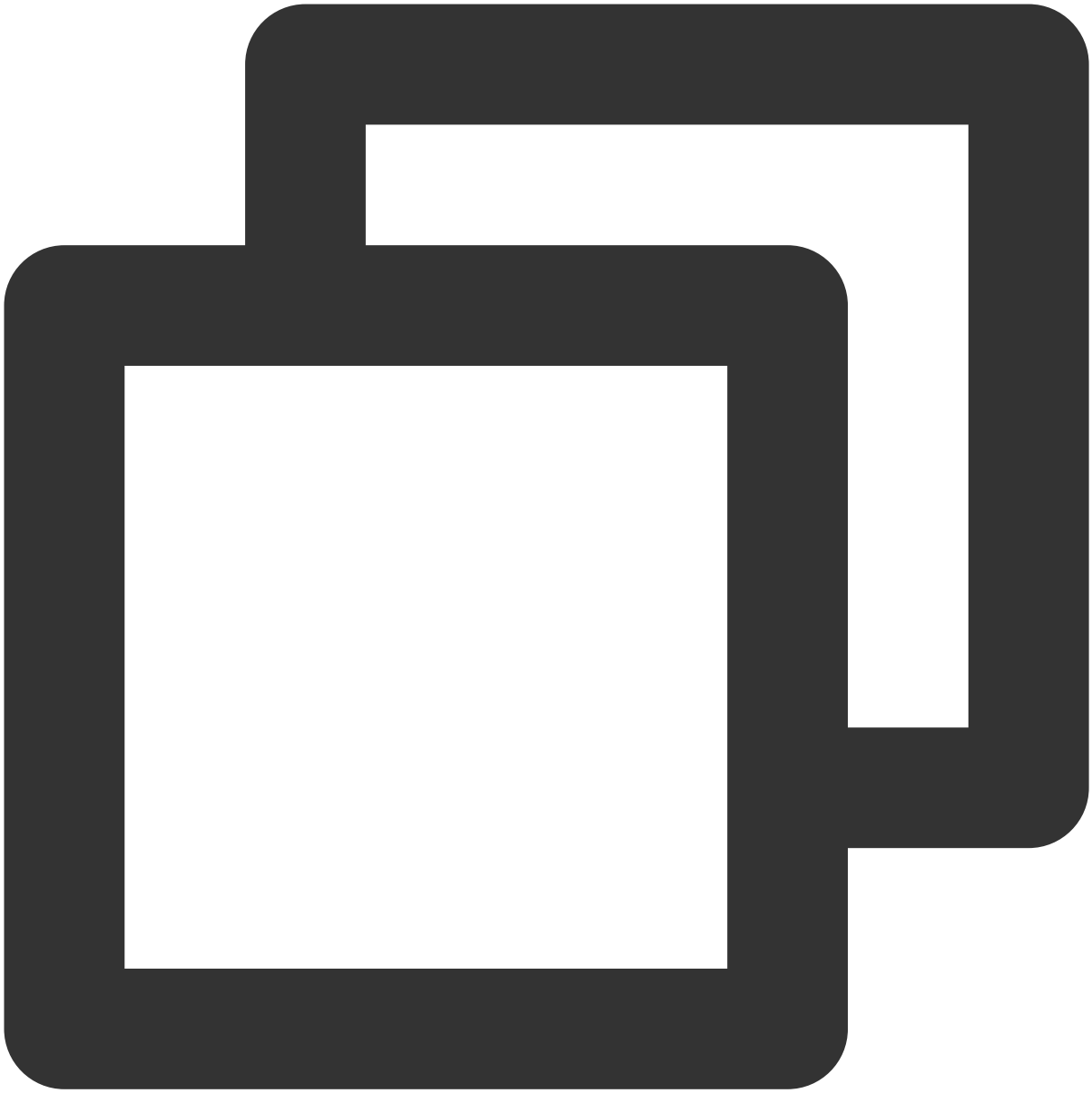
```
TencentEffectApi.getApi()?.setAIDataListener(XmagicAIDataListenerImp());
```

애니메이션 효과 팁에 대한 콜백 구성



```
TencentEffectApi.getApi()?.setTipsListener(XmagicTipsListenerImp());
```

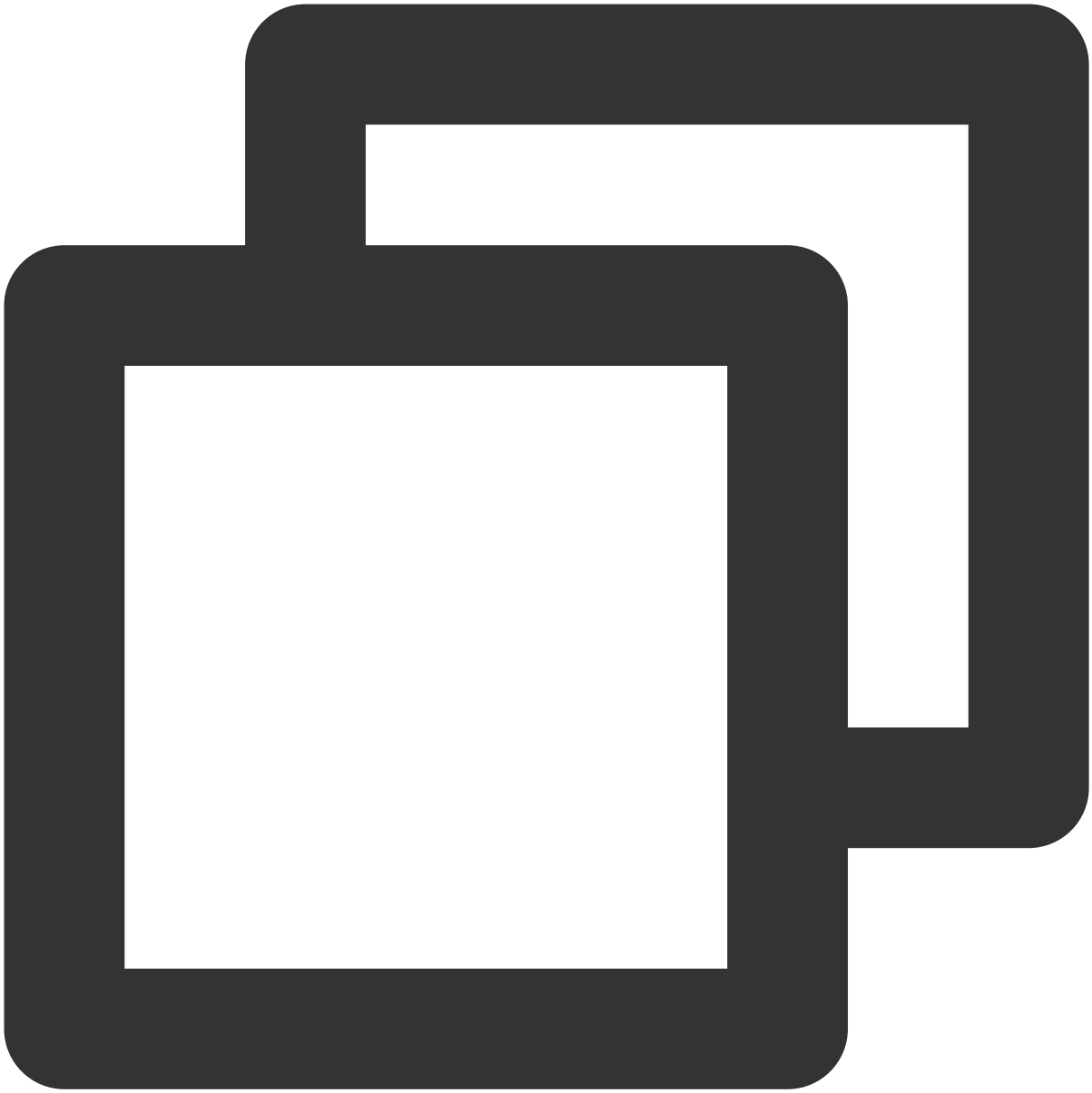
얼굴 키포인트 및 기타 데이터의 콜백 구성(S1-05 및 S1-06에서만 사용 가능)



```
TencentEffectApi.getApi()?.setYTDataListener((data) {
       TXLog.printlog("setYTDataListener  $data");
   });
```

모든 콜백 제거

페이지를 종료할 때 모든 콜백을 제거해야 합니다.



```
TencentEffectApi.getApi()?.setOnCreateXmagicApiErrorListener(null);
TencentEffectApi.getApi()?.setAIDataListener(null);
TencentEffectApi.getApi()?.setYTDataListener(null);
TencentEffectApi.getApi()?.setTipsListener(null);
```

설명

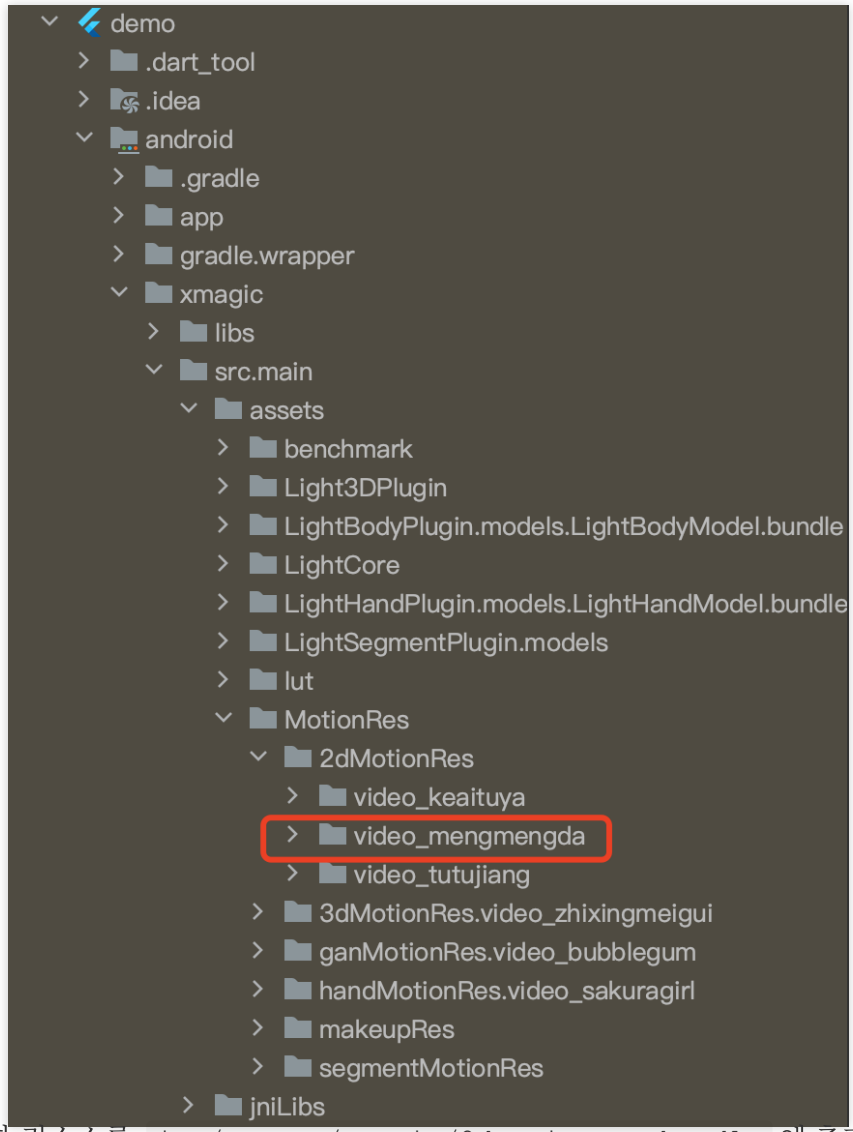
API에 대한 자세한 내용은 API 문서를 참고하십시오. 기타는 Demo 프로젝트를 참고하십시오.

9단계: 뷰티 필터 패널에 데이터 추가 및 제거

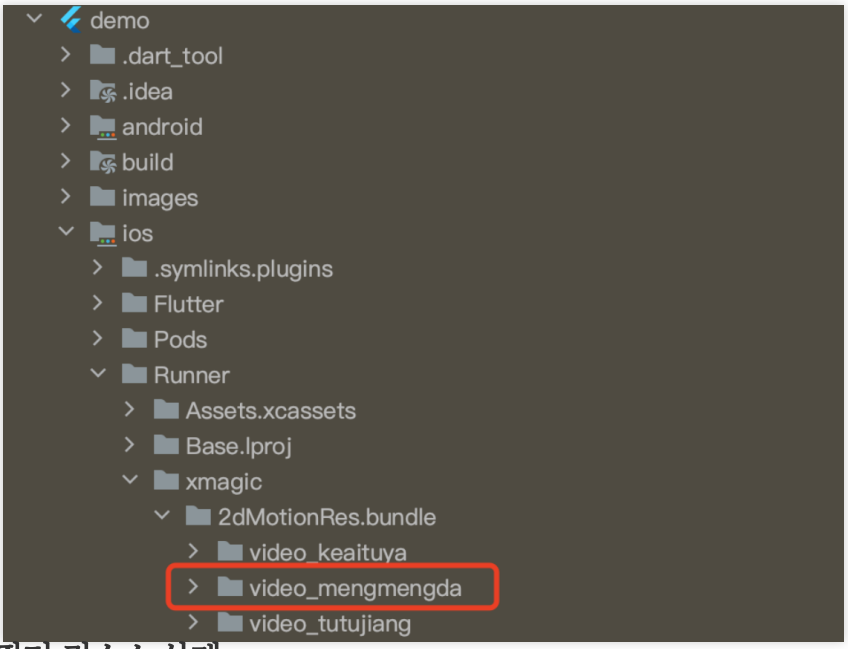
BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid 및 BeautyPropertyProducerIOS 클래스에서 뷰티 필터 패널 데이터를 사용자 지정할 수 있습니다.

뷰티 필터 리소스 추가

1. 1단계에서 설명한 대로 해당 폴더에 리소스 파일을 추가합니다. 예를 들어 2D 애니메이션 효과를 추가하려면 프로젝트의 `android/xmagic/src.mian/assets/MotionRes/2dMotionRes` 에 리소스를 넣어야 합니다.



2. 또한 리소스를 `ios/Runner/xmagic/2dMotionRes.bundle` 에 추가합니다.



뷰티 필터 리소스 삭제

귀하의 License는 패널에서 삭제할 수 있는 일부 뷰티필터 또는 신체 보정 효과를 지원하지 않을 수 있습니다. 예를 들어 립스틱 효과를 삭제하려면 다음을 수행합니다.

BeautyPropertyProducerAndroid 및 BeautyPropertyProducerIOS의 getBeautyData에서 아래 코드를 삭제합니다.

```
// Map<String, String> lipsResPathNames = {};
// lipsResPathNames["lips_fuguhong.png"] = "复古红";
// lipsResPathNames["lips_mitaose.png"] = "蜜桃色";
// lipsResPathNames["lips_shanhuju.png"] = "珊瑚橘";
// lipsResPathNames["lips_wenroufen.png"] = "温柔粉";
// lipsResPathNames["lips_huolicheng.png"] = "活力橙";
// List<XmagicUIProperty> itemLipsPropertys = [];
// String lipId = "beauty.lips.lipsMask";
// for (String ids in lipsResPathNames.keys) {
//   itemLipsPropertys.add(XmagicUIProperty(
//       uiCategory: Category.BEAUTY,
//       displayName: lipsResPathNames[ids]!,
//       id: lipId,
//       resPath: resPaths + ids,
//       thumbDrawableName: "beauty_lips",
//       effKey: BeautyConstant.BEAUTY_MOUTH_LIPSTICK,
//       effValue: XmagicPropertyValues(0, 100, 50, 0, 1),
//       rootDisplayName: "口红"));
// }
// XmagicUIProperty itemLips = XmagicUIProperty(
//     displayName: "口红",
//     thumbDrawableName: "beauty_lips",
//     uiCategory: Category.BEAUTY);
// itemLips.xmagicUIPropertyList = itemLipsPropertys;
// beautyList.add(itemLips);
```

Tencent Effect를 TRTC SDK에 통합하기

ios

최종 업데이트 날짜: : 2023-02-27 14:18:15

통합 준비

1. Demo 패키지를 다운로드하고 압축을 해제하고, Demo 프로젝트에서 실제 항목 프로젝트로 xmagic 모듈(bundle, XmagicIconRes, Xmagic 폴더)을 가져옵니다.
2. **사용하는 XMagic SDK 버전이 2.5.0 이전인 경우,** SDK 디렉터리에서 `libpag.framework` , `Masonry.framework` , `XMagic.framework` , `YTCommonXMagic.framework` 를 가져옵니다. **사용하는 XMagic SDK 버전이 2.5.1 이상인 경우,** SDK 디렉터리에서 `libpag.framework` , `Masonry.framework` , `XMagic.framework` , `YTCommonXMagic.framework` , `Audio2Exp.framework` , `TEFFmpeg.framework` 를 가져옵니다.
3. framework 서명의 경우 **General--> Masonry.framework** 및 **libpag.framework**를 **Embed & Sign**으로 설정합니다. **YTCommonXMagic.framework** 버전 2.5.1 이전은 **Do Not Embed**를 선택하고, 버전 2.5.1 이후는 **Embed & Sign**을 선택합니다.
4. Bundle ID를 신청한 테스트 인증과 동일하게 수정합니다.

개발자 환경 요구 사항

Xcode 11 이상: App Store 또는 여기에서 다운로드하십시오.
권장 실행 환경:
기기 사양: iPhone 5 이상. iPhone 6 이하의 경우 전면 카메라는 720p까지 지원하며 1080p는 지원되지 않습니다.
시스템 요구 사항: iOS 10.0 이상.

C/C++ 레이어 개발 환경

XCode는 기본적으로 C++ 환경을 사용합니다.

유형	종속성 라이브러리
시스템 종속 라이브러리	Accelerate AssetsLibrary AVFoundation CoreMedia CoreFoundation CoreML Foundation JavaScriptCore libc++.tbd

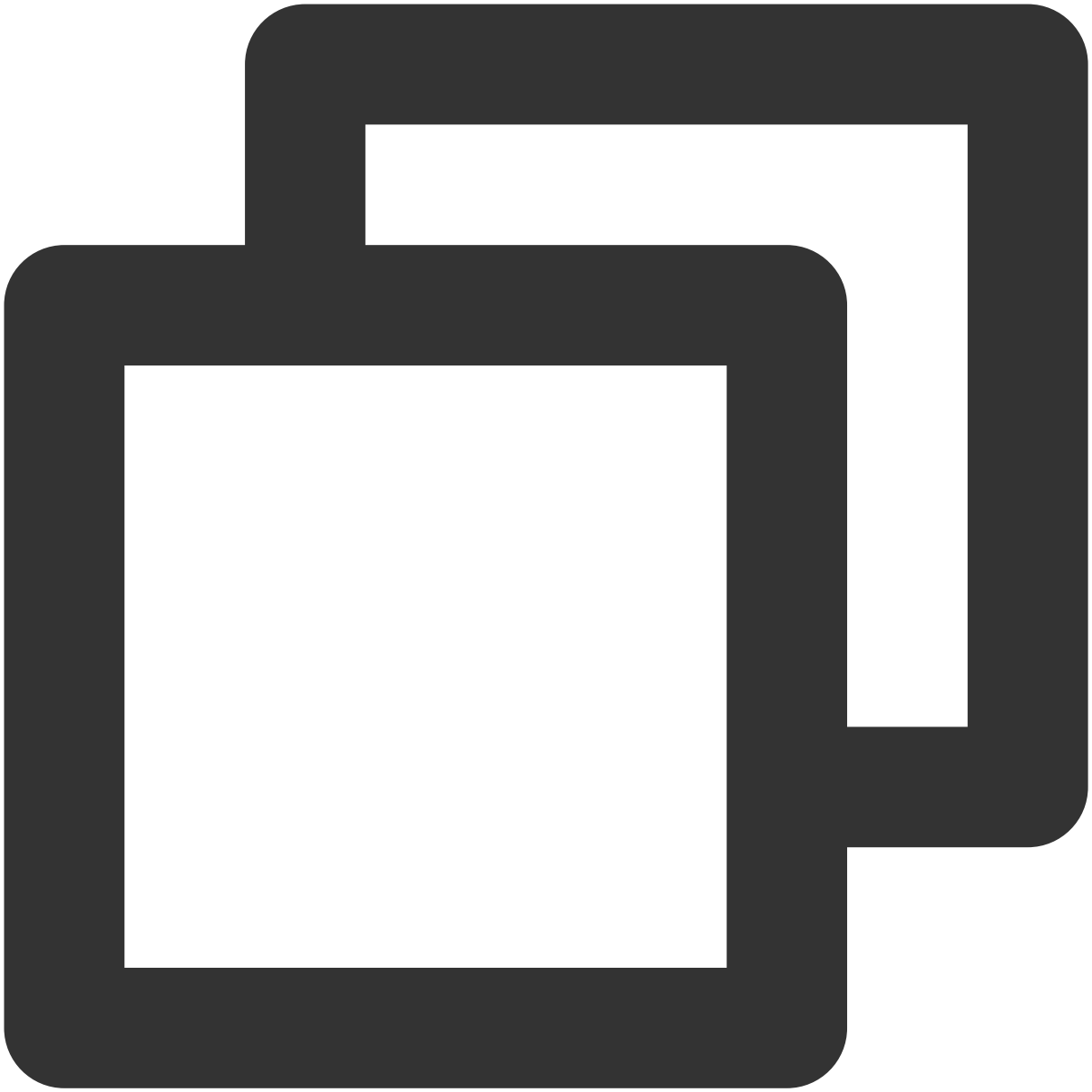
	libz.b libresolv.tbd libsqlite3.0.tbd MetalPerformanceShaders MetalKit MobileCoreServices OpneAL OpneGLES Security ReplayKit SystemConfiguration UIKit
내장 라이브러리	YTCommon(인증 정적 라이브러리) XMagic(뷰티 필터 정적 라이브러리) libpag(비디오 디코딩 동적 라이브러리) Masonry(컨트롤 레이아웃 라이브러리) TXLiteAVSDK_Professional TXFFmpeg TXSoundTouch Audio2Exp(xmagic sdk version은 2.5.1 이후 버전에만 있음) TEFFmpeg(xmagic sdk version은 2.5.1 이후 버전에만 있음)

SDK API 통합

1단계 및 2단계는 Demo 프로젝트에 있는 `ThirdBeautyViewController` 클래스의 `viewDidLoad` 및 `buildBeautySDK` 메소드를 참고하십시오. `AppDelegate` 클래스의 `application` 메소드는 Xmagic 인증을 수행합니다.
4단계부터 7단계에 대해서는 Demo 프로젝트에서 `ThirdBeautyViewController` 및 `BeautyView` 클래스의 인스턴스 코드를 참고하십시오.

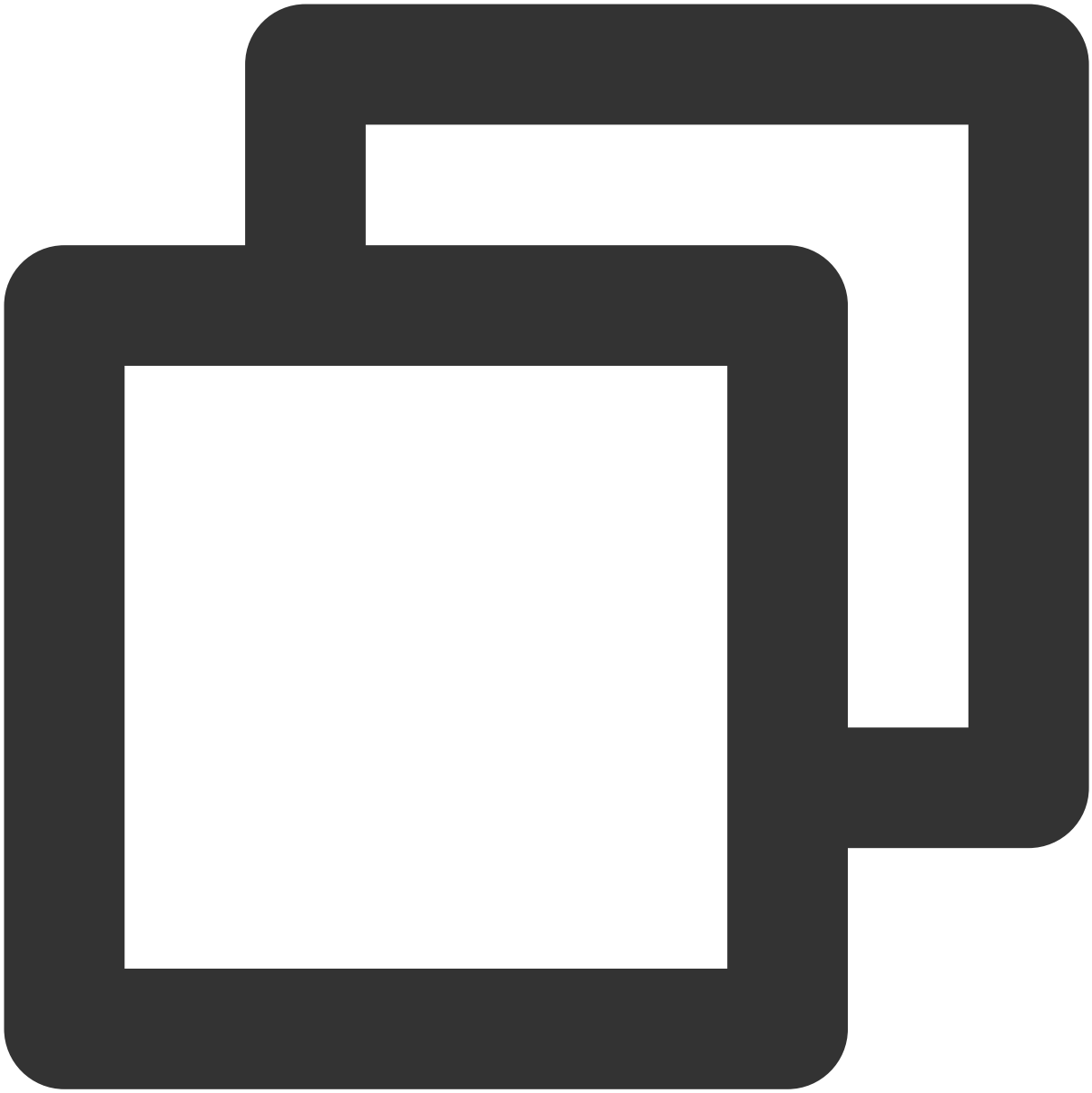
1단계: 인증 초기화

1. 프로젝트의 `AppDelegate` 의 `didFinishLaunchingWithOptions` 에 다음 인증 코드를 추가합니다. 여기서 `LicenseURL` 과 `LicenseKey` 는 Tencent Cloud 공식 웹사이트에서 얻은 인증 정보입니다.



```
[TXLiveBase setLicenceURL:LicenseURL key:LicenseKey];
```

2. Xmagic 인증: 해당 비즈니스 모듈의 초기화 코드에 URL 및 KEY를 설정하여 License 다운로드를 트리거합니다. 사용 직전에 다운로드하지 마십시오. `AppDelegate` 의 `didFinishLaunchingWithOptions` 메소드에서 다운로드를 트리거할 수도 있습니다. 여기서 `LicenseURL` 과 `LicenseKey` 는 License 바인딩 시 콘솔에서 생성되는 정보입니다. 2.5.1 이전의 SDK 버전에서 `TELicenseCheck.h` 는 `XMagic.framework` 에 있고, 2.5.1 이상의 SDK 버전에서 `TELicenseCheck.h` 는 `YTCommonXMagic.framework` 에 있습니다.



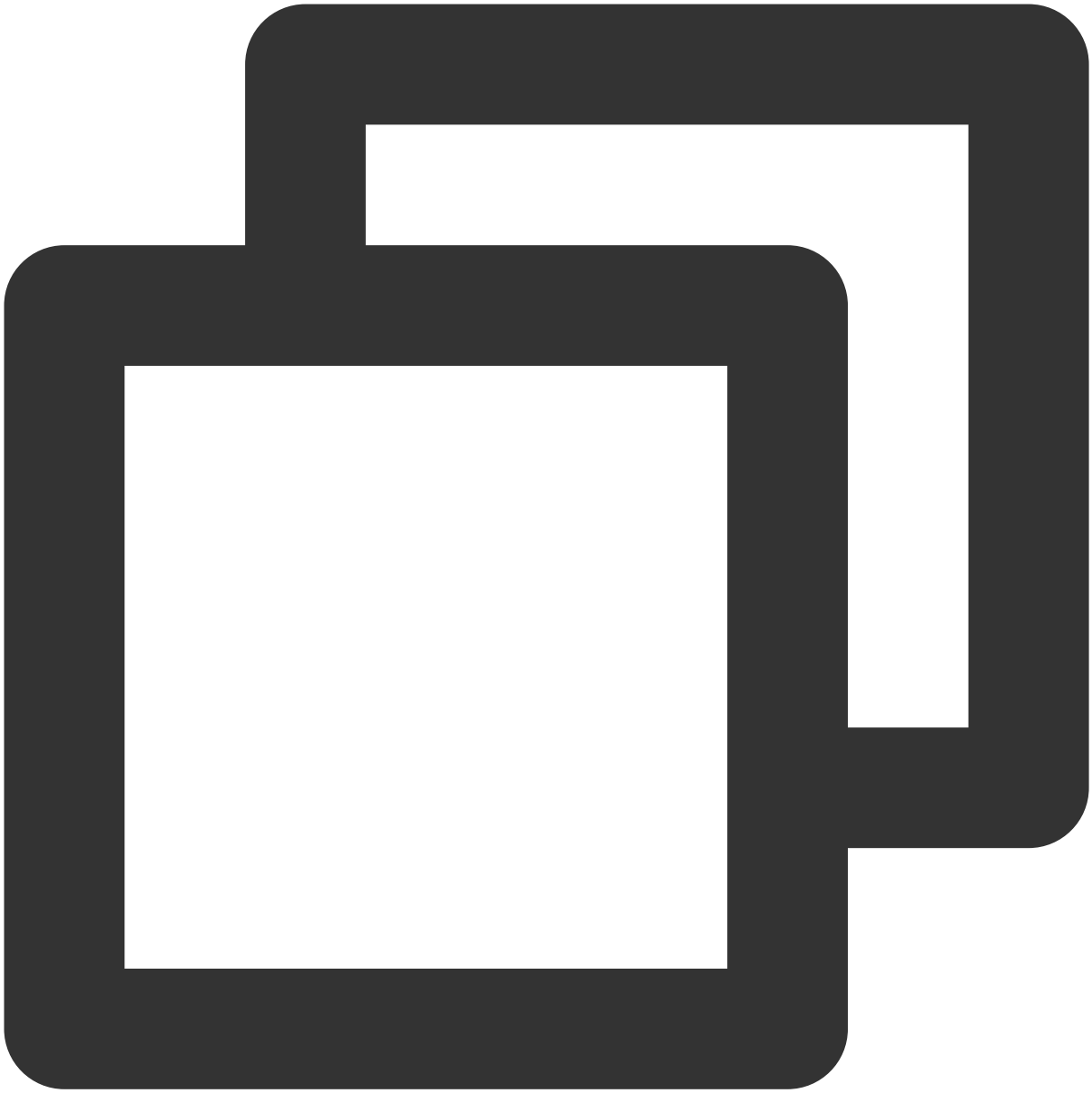
```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
    if (authresult == TELicenseCheckOk) {
        NSLog(@"인증 성공");
    } else {
        NSLog(@"인증 실패");
    }
}];
```

errorCode 인증 설명:

에러 코드	설명

0	성공. Success
-1	입력 매개변수가 유효하지 않습니다. 예: URL 또는 KEY가 비어 있습니다.
-3	다운로드에 실패했습니다. 네트워크 설정을 확인하십시오.
-4	로컬 시스템에서 읽은 TE SDK 인증 정보가 비어 있으며, 이는 I/O 실패로 인해 발생할 수 있습니다
-5	읽은 VCUBE TEMP License 파일의 내용이 비어 있으며 이는 I/O 실패로 인해 발생할 수 있습니다
-6	v_cube.license 파일의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-7	서명 확인에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-8	암호 해독에 실패했습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-9	TELicense 필드의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-10	온라인으로 파싱된 TE SDK 인증 정보가 비어 있습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-11	TE SDK 인증 정보를 로컬 파일에 쓰지 못했습니다. I/O 실패로 인해 발생할 수 있습니다.
-12	다운로드에 실패했으며 로컬 asset을 파싱하지 못했습니다.
-13	인증 실패
기타	도움이 필요한 경우 Tencent Cloud에 문의하십시오

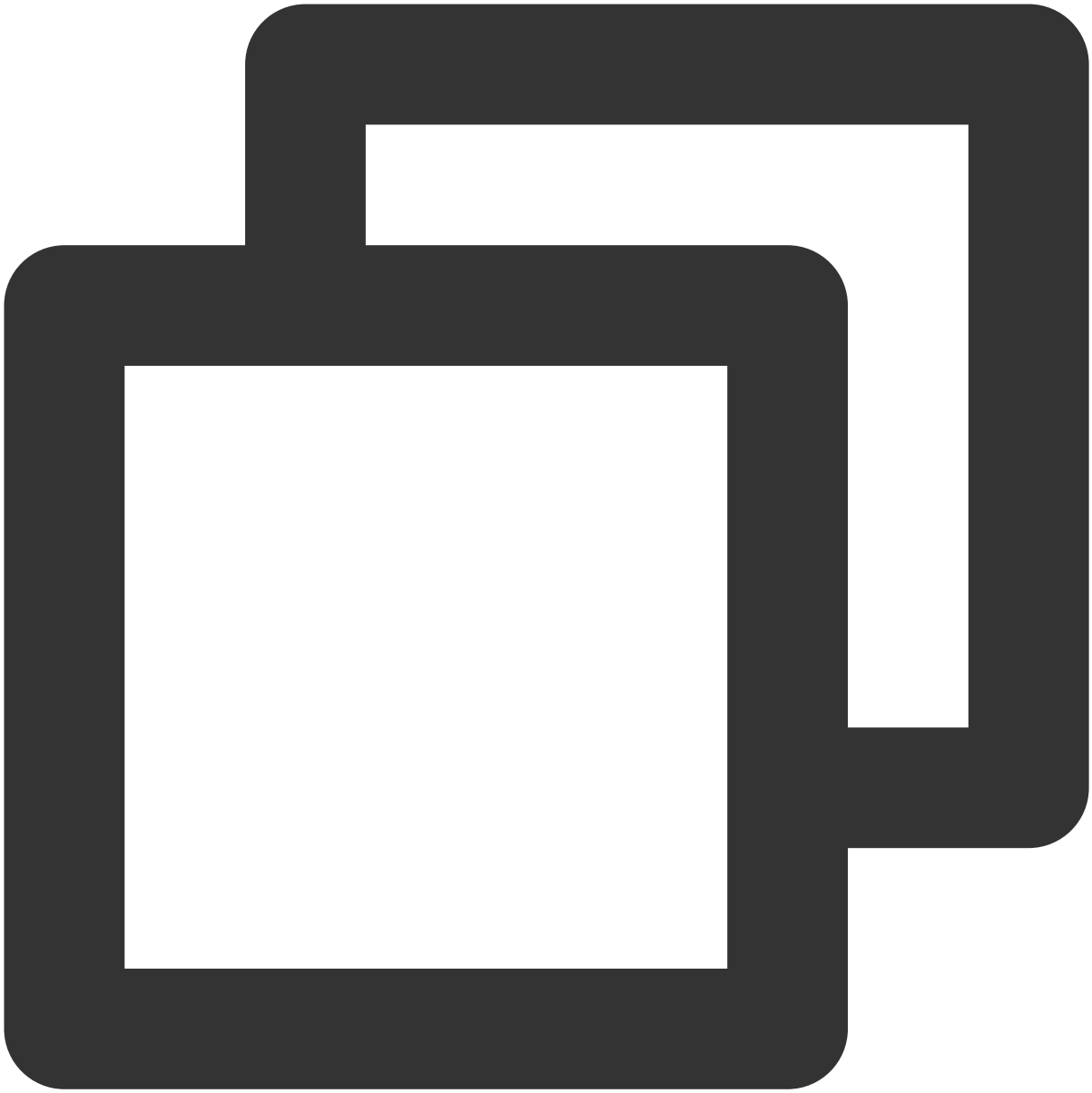
2단계: SDK 소재 리소스 경로 설정



```
CGSize previewSize = [self getPreviewSizeByResolution:self.currentPreviewResolution
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
```

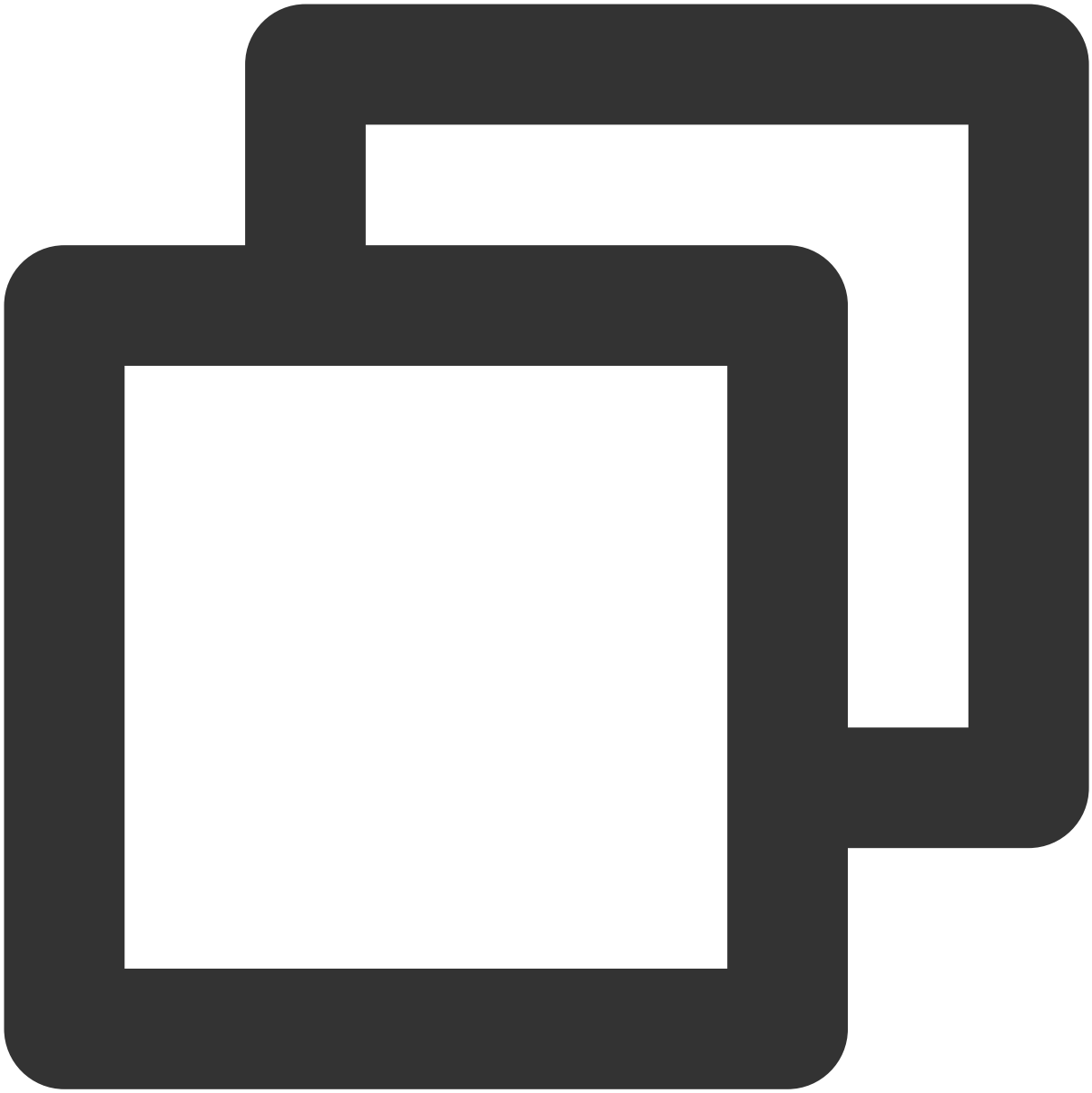
```
                                                options:NSJSONReadingMutableContainers
                                                error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Init beauty kit
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
```

3단계: 로그 및 이벤트 리스너 추가



```
 // Register log
[self.beautyKit registerSDKEventListener:self];
[self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL];
```

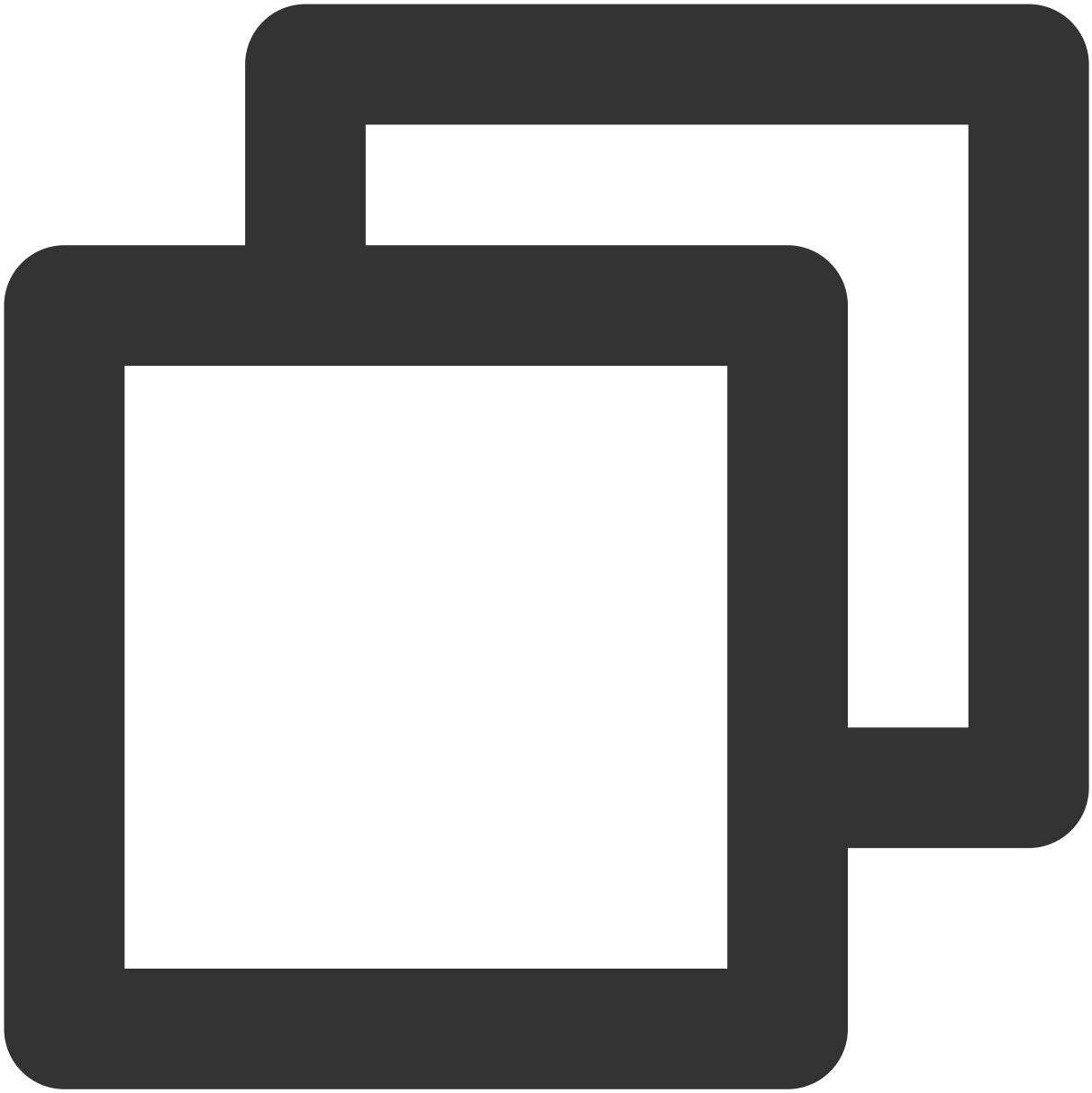
4단계: 뷰티 필터 효과 구성



```
- (int)configPropertyWithType:(NSString *_Nonnull)propertyType withName:(NSString *
```

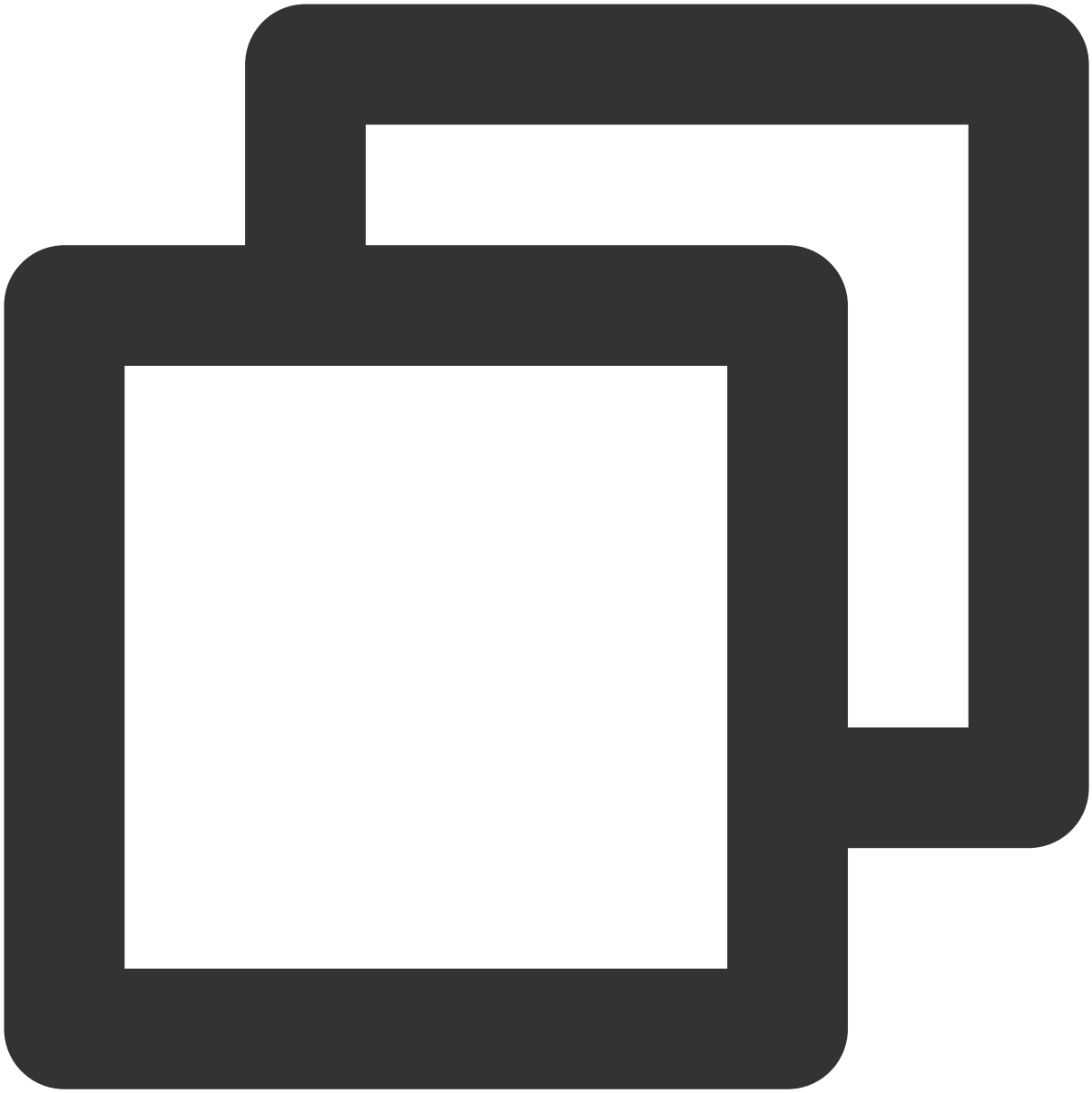
5단계: 비디오 렌더링

비디오 프레임 콜백 인터페이스에서 YTProcessInput을 구성하고 렌더링 처리를 위해 SDK에 전달합니다. Demo의 ThirdBeautyViewController를 참고하십시오.



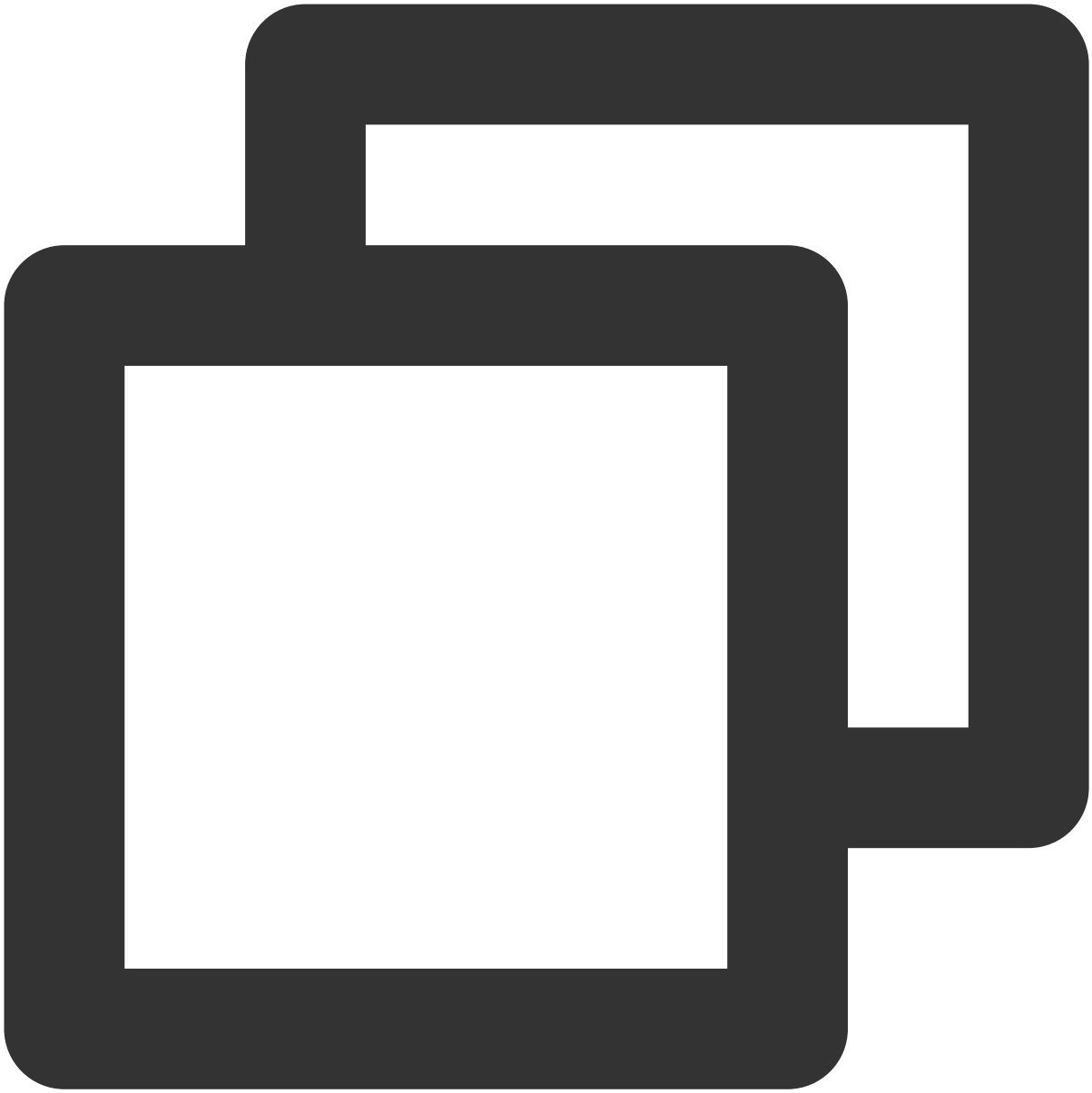
```
[self.xMagicKit process:inputCPU withOrigin:YtLightImageOriginTopLeft withOrientat
```

6단계: SDK 일시 중지/재개



```
[self.beautyKit onPause];
[self.beautyKit onResume];
```

7단계: 레이아웃에 SDK 뷰티 필터 패널 추가



```
UIEdgeInsets gSafeInset;
#if __IPHONE_11_0 && __IPHONE_OS_VERSION_MAX_ALLOWED >= __IPHONE_11_0
if(gSafeInset.bottom > 0){
}
if (@available(iOS 11.0, *)) {
    gSafeInset = [UIApplication sharedApplication].keyWindow.safeAreaInsets;
} else
#endif
    {
        gSafeInset = UIEdgeInsetsZero;
    }
```

```
dispatch_async(dispatch_get_main_queue(), ^{
    //뷰티 필터 옵션 UI
    _vBeauty = [[BeautyView alloc] init];
    [self.view addSubview:_vBeauty];
    [_vBeauty mas_makeConstraints:^(MASConstraintMaker *make) {
        make.width.mas_equalTo(self.view);
        make.centerX.mas_equalTo(self.view);
        make.height.mas_equalTo(254);
        if(gSafeInset.bottom > 0.0){  // 전체 화면으로 조정
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(0);
        } else {
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(-10);
        }
    }];
    _vBeauty.hidden = YES;
});
```

Android

최종 업데이트 날짜: : 2022-07-18 10:06:17

1단계: Demo 프로젝트 압축 해제

1. Tencent Effect(TE) SDK와 통합된 TRTC Demo를 다운로드합니다. 이 Demo는 Tencent Effect SDK S1-04 에디션을 기반으로 제작되었습니다.
2. Demo의 SDK 파일을 실제로 사용하는 SDK용 파일로 교체합니다. 구체적 작업은 다음 단계를 따르십시오.
   - xmagic 모듈의 libs 디렉터리에 있는 `.aar` 파일을 SDK의 libs에 있는 `.aar` 파일로 교체하십시오.
   - xmagic 모듈의 `../src/main/assets` 에 있는 모든 파일을 SDK의 `assets/` 에 있는 파일로 교체합니다. SDK 패키지의 MotionRes 폴더에 파일이 있는 경우 `../src/main/assets` 디렉터리에도 복사합니다.
   - xmagic 모듈의 `../src/main/jniLibs` 에 있는 모든 .so 파일을 SDK 패키지의 jniLibs에 있는 .so 파일로 교체합니다(arm64-v8a 및 armeabi-v7a에 대한 .so 파일을 얻으려면 jinLibs 폴더에 있는 ZIP 파일의 압축을 풀어야 합니다).
3. Demo 프로젝트의 xmagic 모듈을 실제 항목 프로젝트로 가져옵니다.

2단계: app 모듈의 build.gradle 열기

1. applicationId를 신청된 테스트 인증과 일치하는 패키지 이름으로 수정합니다.
2. gson 종속성 설정을 추가합니다.

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

3단계: SDK 인터페이스 통합

Demo 프로젝트의 ThirdBeautyActivity 클래스를 참고하십시오.

1. **인증:**

```
//인증 시 주의사항 및 오류 코드 내용은 https://www.tencentcloud.com/document/product/1143/45385#step-1.-authenticate를 참고하십시오
XMagicImpl.checkAuth((errorCode, msg) -> {
if (errorCode == TELicenseCheck.ERROR_OK) {
showLoadResourceView();
```

```
} else {
TXCLog.e(TAG, "인증 실패, 인증 url 및 key를 확인하십시오" + errorCode + " " + msg);
}
});
```

2. 소재 초기화:

```
private void showLoadResourceView() {
if (XmagicLoadAssetsView.isCopyedRes) {
XmagicResParser.parseRes(getApplicationContext());
initXMagic();
} else {
loadAssetsView = new XmagicLoadAssetsView(this);
loadAssetsView.setOnAssetsLoadFinishListener(() -> {
XmagicResParser.parseRes(getApplicationContext());
initXMagic();
});
}
}
```

3. 푸시 설정 활성화:

```
mTRTCCloud.setLocalVideoProcessListener(TRTCCloudDef.TRTC_VIDEO_PIXEL_FORMAT_Te
xture_2D, TRTCCloudDef.TRTC_VIDEO_BUFFER_TYPE_TEXTURE, new TRTCCloudListener.TR
TCVideoFrameListener() {
@Override
public void onGLContextCreated() {
}
@Override
public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudD
ef.TRTCVideoFrame dstFrame) {
}
@Override
public void onGLContextDestory() {
}
});
```

4. **textureId를 SDK에 불러오기 및 렌더링 처리:**

TRTCVideoFrameListener 인터페이스의 `onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudDef.TRTCVideoFrame dstFrame)` 메소드에 다음 코드를 추가합니다.

```
dstFrame.texture.textureId = mXMagic.process(srcFrame.texture.textureId, srcFra
me.width, srcFrame.height);
```

5. **SDK 일시 중지/비활성화:**
   onPause()는 뷰티 필터 일시 중지에 사용되며, Activity/Fragment 라이프사이클 메소드에서 실행할 수 있습니다. onDestroy 메소드는 GL 스레드에서 호출해야 합니다. (onTextureDestroyed 메소드에서 XMagicImpl 객체의 `onDestroy()` 호출 가능), 자세한 사용법은 Demo를 참고하십시오.

```
mXMagic.onPause(); //일시 정지, Activity의 onPause 메소드와 연결
mXMagic.onDestroy(); //폐기. GL 스레드에서 호출
```

6. **레이아웃에 SDK 뷰티 필터 패널 추가**:

```
<RelativeLayout
android:layout_above="@+id/ll_edit_info"
android:id="@+id/livepusher_bp_beauty_pannel"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
```

7. **패널 초기화:**

```
private void initXMagic() {
if (mXMagic == null) {
mXMagic = new XMagicImpl(this, mBeautyPanelView);
}else{
mXMagic.onResume();
}
}
```

자세한 내용은 Demo 프로젝트의 `ThirdBeautyActivity.initXMagic();` 메소드를 참고하십시오.

Flutter

최종 업데이트 날짜: : 2023-04-27 16:14:57

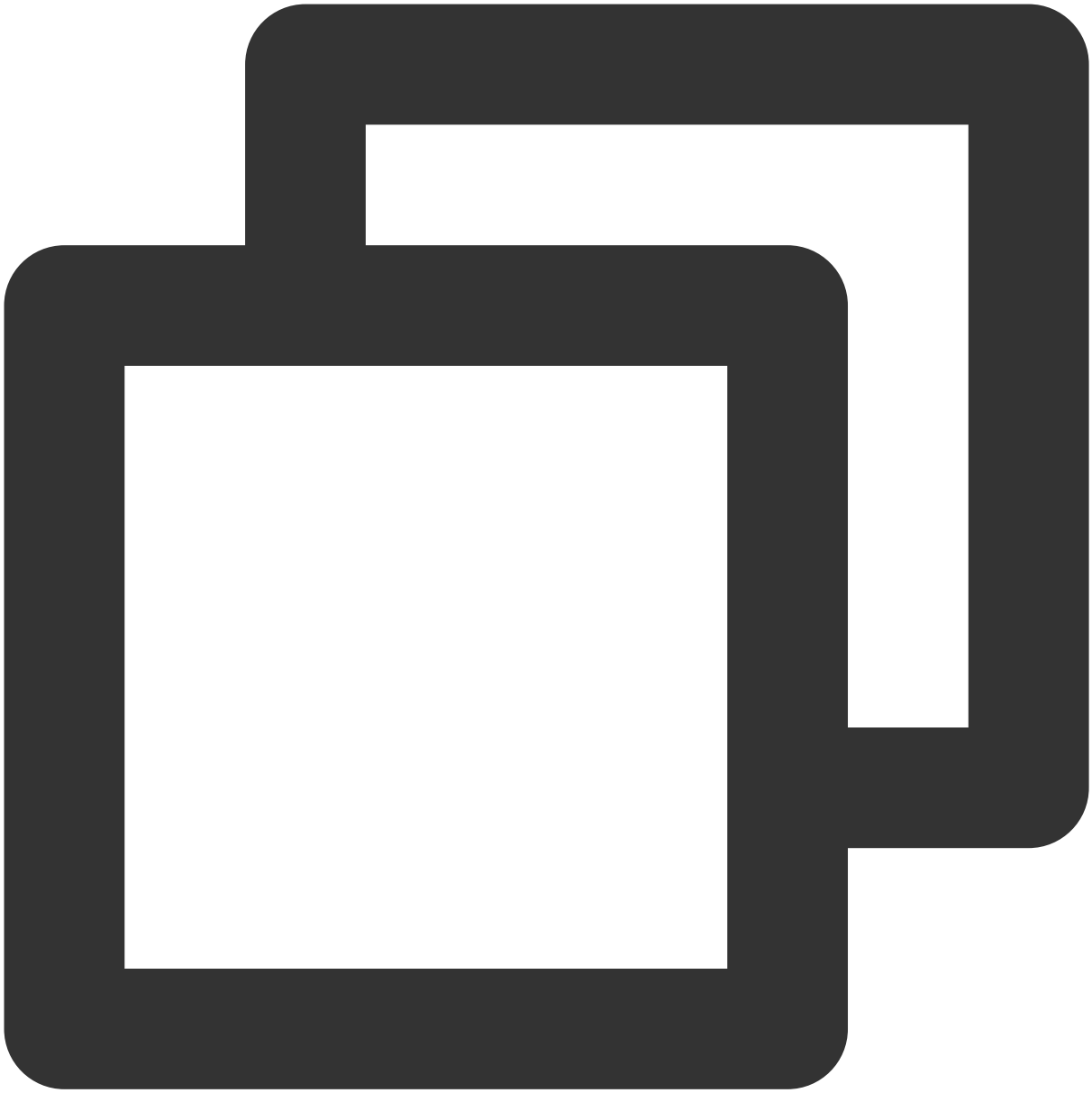
1단계: 효과 리소스 다운로드 및 통합

1. 구매한 패키지의 해당 SDK를 다운로드합니다.
2. 자신의 프로젝트에 리소스 파일을 추가합니다.

Android

iOS

1. app에서 build.gradle을 열고 패키지의 maven 주소를 추가합니다. 예를 들어 S1-04를 구매한 경우 다음을 추가합니다.

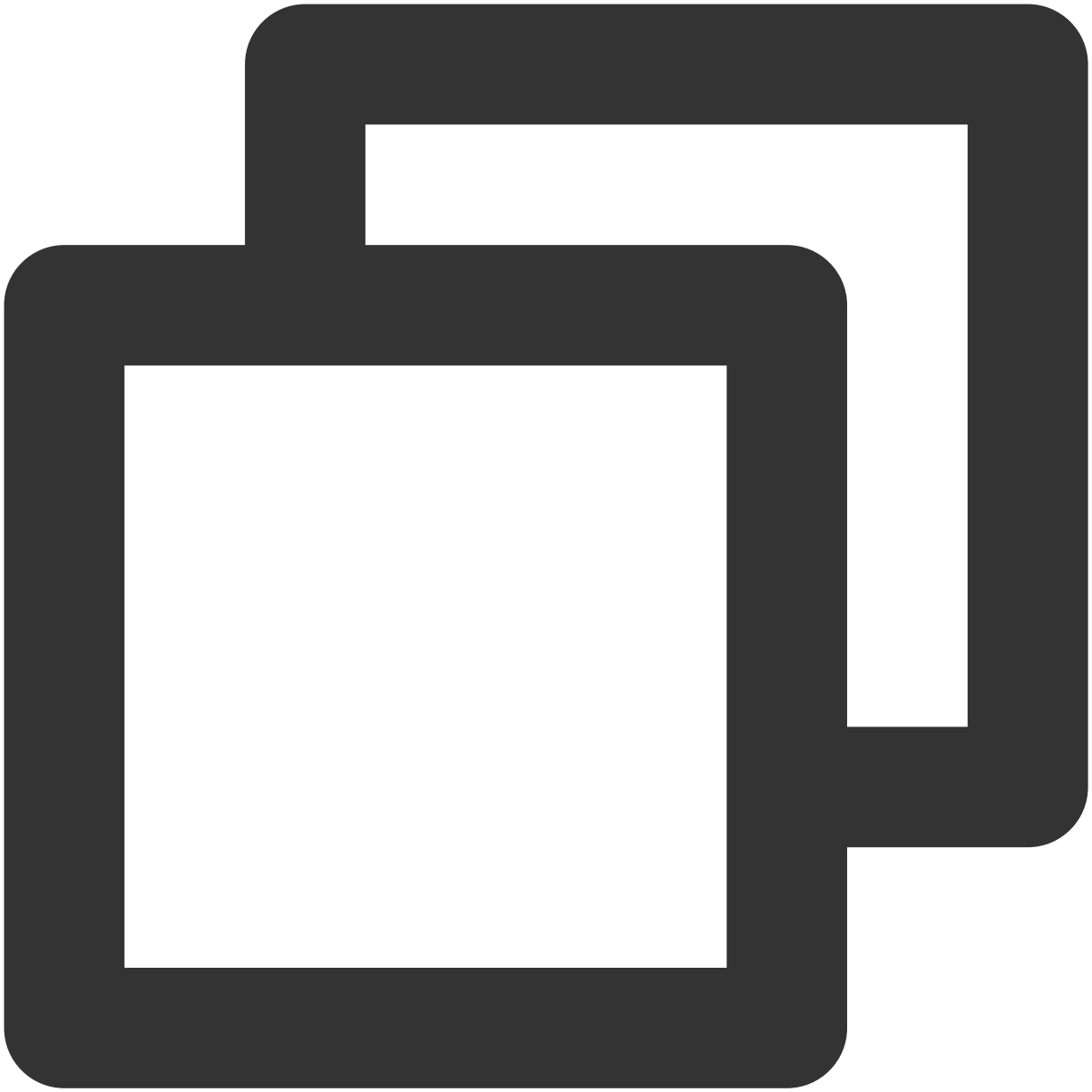


```
dependencies{
    implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
}
```

문서에서 다양한 패키지의 maven 주소를 찾을 수 있습니다.

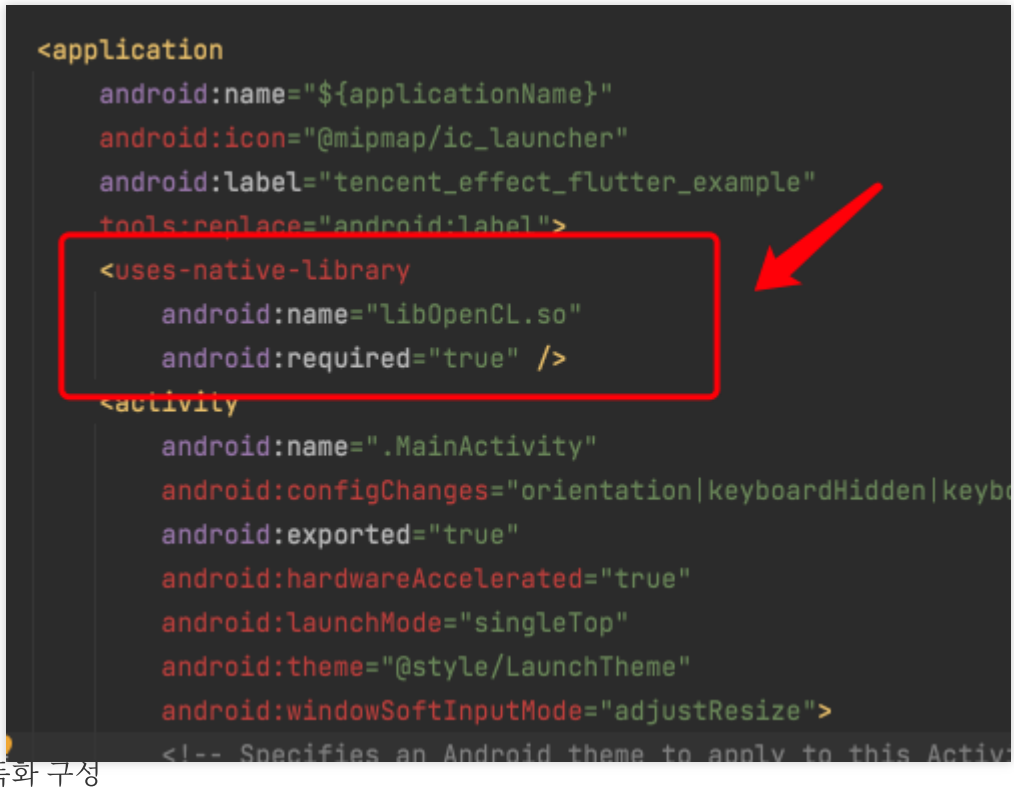
2. app에서 src/main/assets 폴더를 찾습니다(이 폴더가 없으면 새로 생성). MotionRes 폴더가 있는지 확인합니다. `../src/main/assets` 에 복사합니다.

3. app에서 AndroidManifest.xml을 열고 application에 다음 태그를 추가합니다



```
<uses-native-library
       android:name="libOpenCL.so"
       android:required="true" />
       //여기서 true는 이 라이브러리가 없으면 애플리케이션이 제대로 작동하지 않음을 의미합니다
       //false는 애플리케이션이 이 라이브러리(있는 경우)를 사용할 수 있지만 특히 라이브러리 없
       //Android 공식 웹사이트 소개: %!s(<nil>)
```

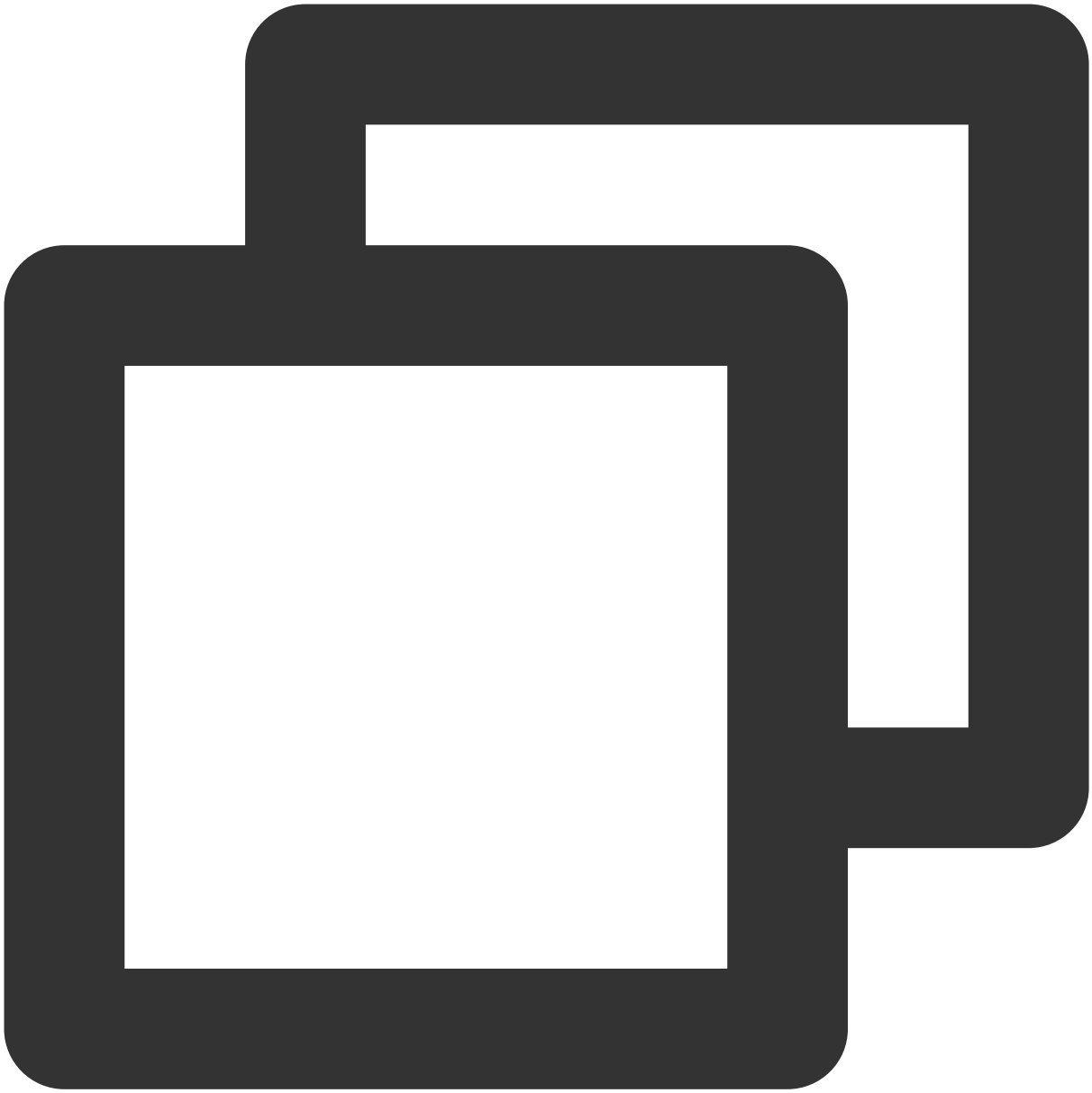
다음과 같이 표시됩니다.



4. 난독화 구성

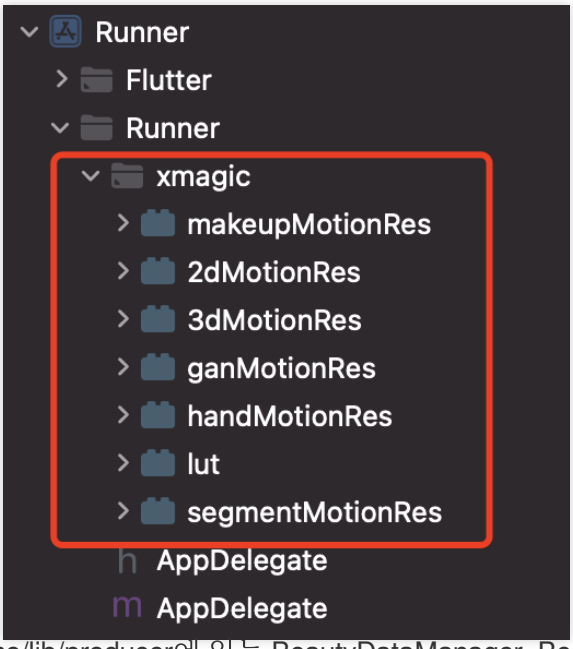
release 패키지를 인쇄할 때 컴파일 최적화(minifyEnabled를 true로 설정)를 활성화한 경우 java 레이어에서 호출되지 않은 일부 코드가 잘려서 native 레이어에서 호출되어 `no xxx method` 예외가 발생할 수 있습니다.

이러한 컴파일 최적화를 활성화한 경우 xmagic 코드가 잘리지 않도록 다음 keep 규칙을 추가합니다.



```
-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.** { *;}
```

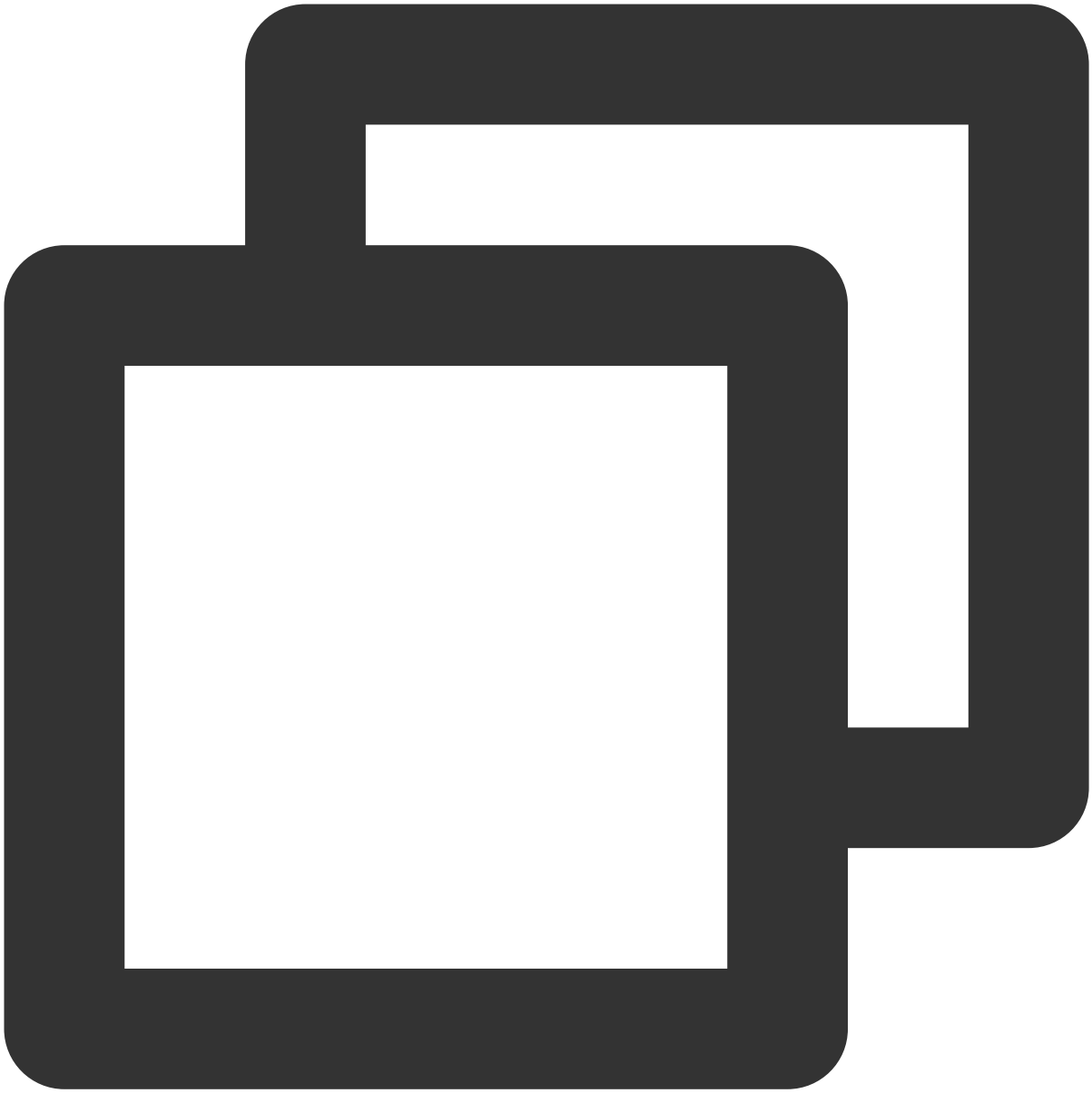
1. 프로젝트에 뷰티 필터 리소스 추가(리소스는 스크린샷과 다를 수 있음):



2. demo/lib/producer에 있는 BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid 및 BeautyPropertyProducerIOS의 네 가지 클래스를 Flutter 프로젝트에 복사합니다. 효과 패널에서 효과 리소스 및 표시 효과 옵션을 구성하는 데 사용됩니다.

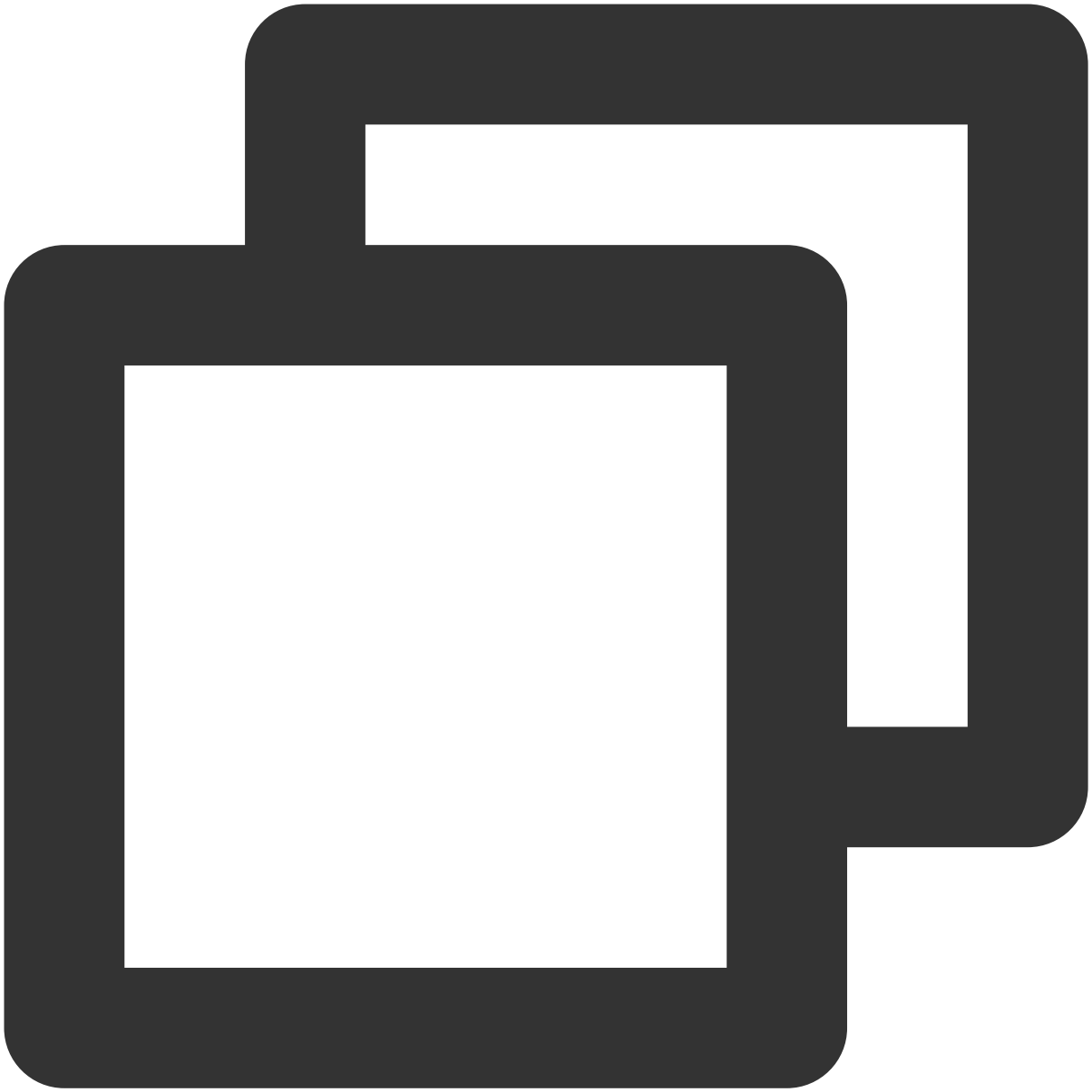
2단계: Flutter SDK 참조

GitHub 참조: 프로젝트의 pubspec.yaml 파일에 다음 참조를 추가합니다.



```
tencent_effect_flutter:
  git:
    url: https://github.com/TencentCloud/tencenteffect-sdk-flutter
```

로컬 참조: tencent_effect_flutter SDK의 최신 버전을 다운로드합니다. `android` `ios` 및 `lib` 폴더와 `pubspec.yaml` 및 `tencent_effect_flutter.iml` 파일을 프로젝트 디렉터리에 복사하고 프로젝트의 pubspec.yaml 파일에 다음 코드를 추가합니다. (demo 참고)



```
tencent_effect_flutter:
    path: ../
```

tencent_effect_flutter는 브릿지 역할만 합니다. 효과를 구현하는 것은 XMagic입니다. 기본적으로 최신 버전의 XMagic이 사용됩니다.

최신 버전의 뷰티 필터 SDK를 사용하려면 다음 방법을 사용하여 SDK를 업데이트할 수 있습니다.

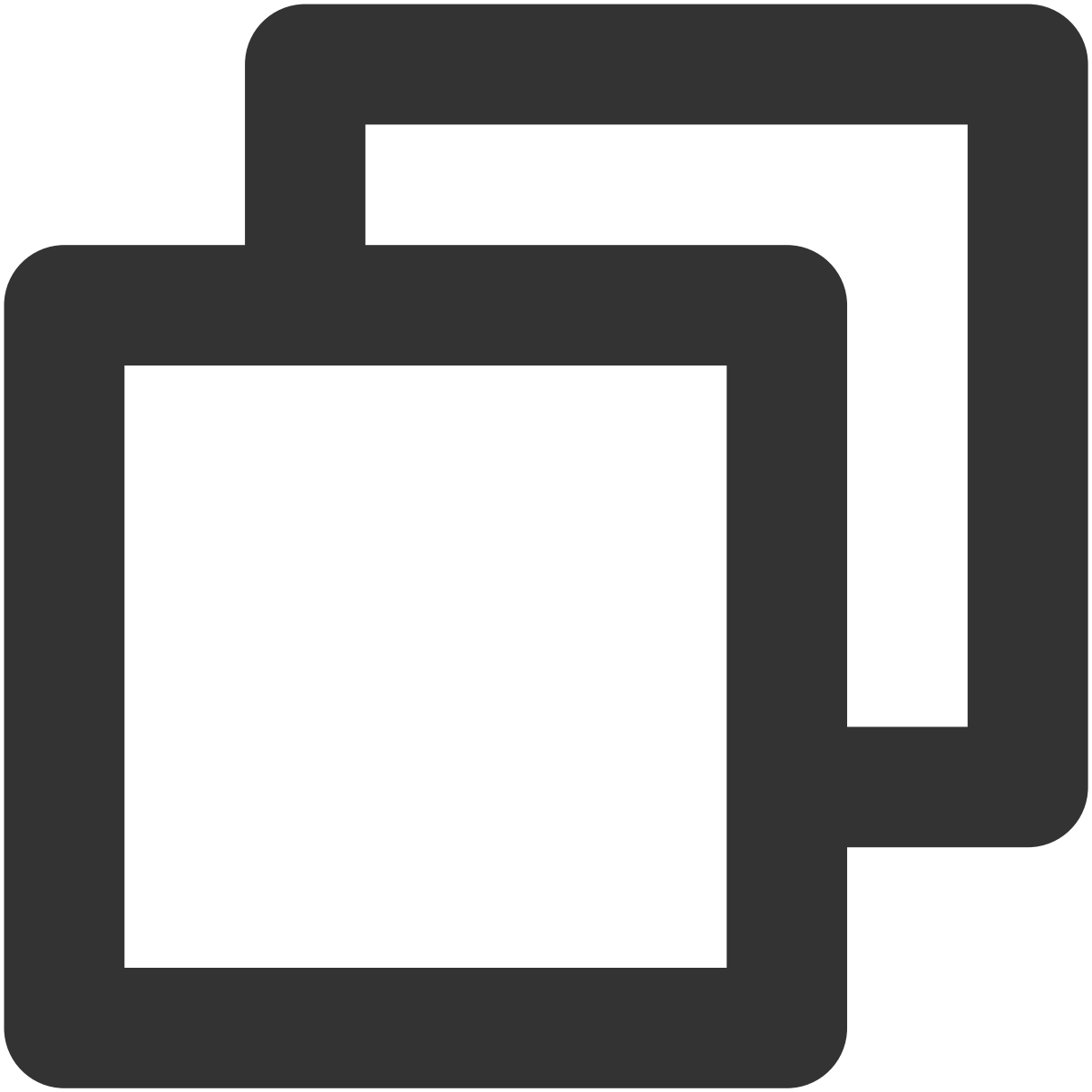
Android

iOS

프로젝트 디렉터리에서 `flutter pub upgrade` 를 실행하거나 `subspec.yaml` 페이지의 오른쪽 상단에 있는 **Pub upgrade**를 클릭합니다.
프로젝트 디렉터리에서 flutter pub upgrade를 실행한 다음 iOS 디렉터리에서 `pod update` 를 실행합니다.

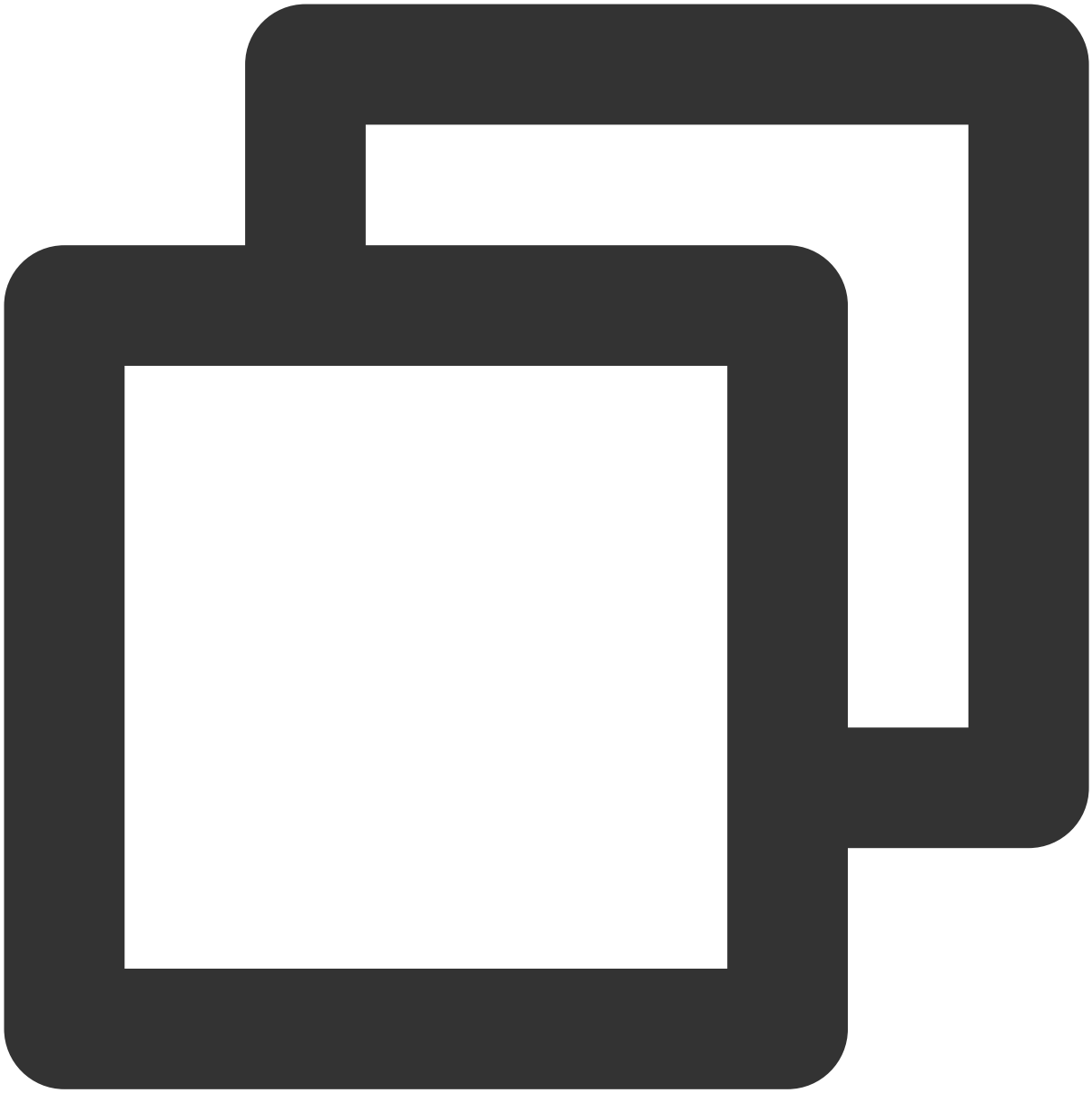
3단계: TRTC와 Tencent Effect 결합

Android
iOS
다음 코드를 application 클래스의 oncreate(또는 FlutterActivity의 onCreate)에 추가합니다.



```
TRTCCloudPlugin.register(new XmagicProcesserFactory());
```

AppDelegate 클래스의 didFinishLaunchingWithOptions에 다음 코드를 추가합니다.

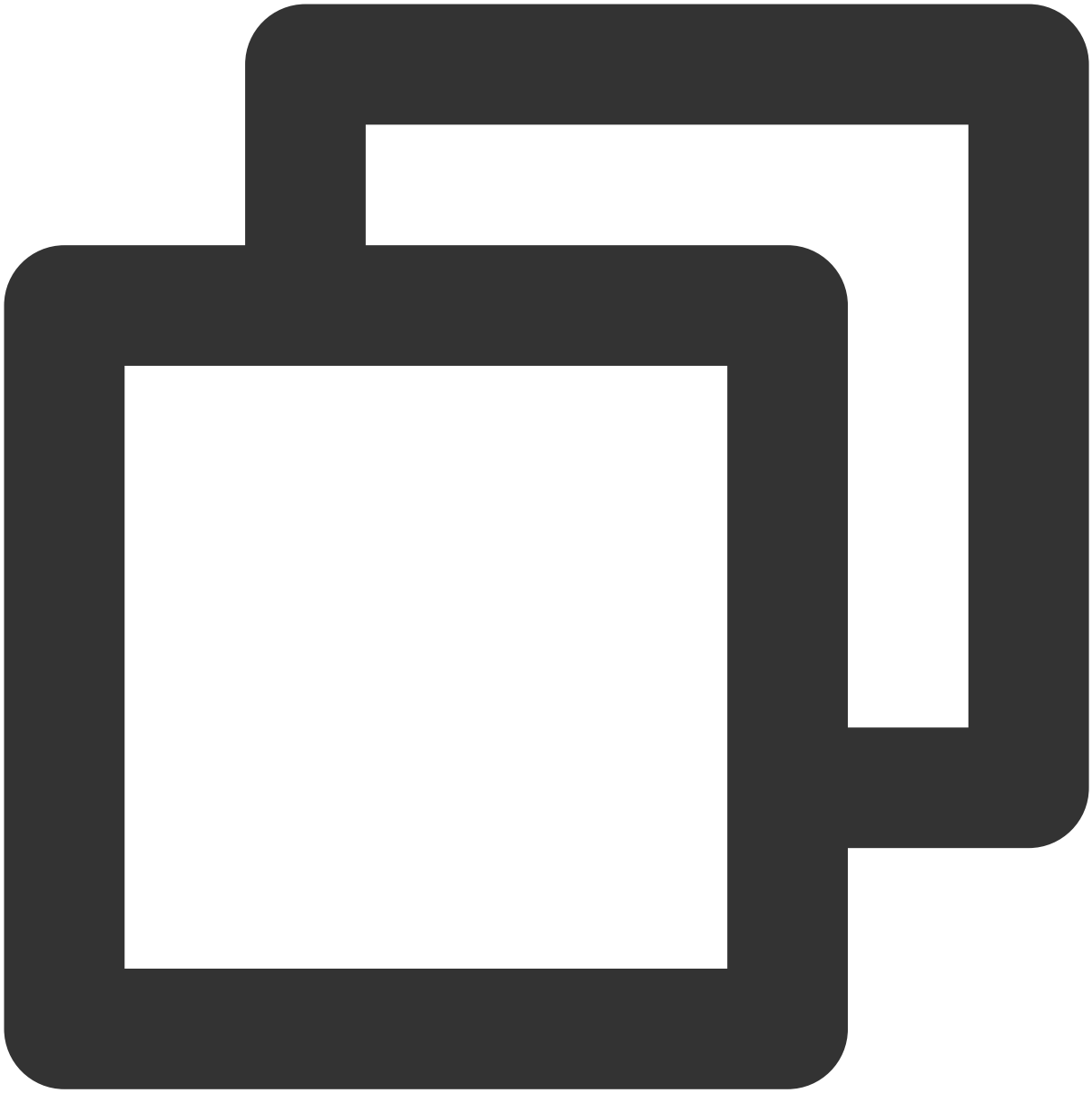


```
XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
[TencentTRTCCloud registerWithCustomBeautyProcesserFactory:instance];
```

다음과 같이 표시됩니다.

```objc
#import "AppDelegate.h"
#import "GeneratedPluginRegistrant.h"
@import live_flutter_plugin;
@import tencent_effect_flutter;
@import tencent_trtc_cloud;

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
  XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc
  [TXLivePluginManager registerWithCustomBeautyProcesserFactory:inst
  [TencentTRTCCloud registerWithCustomBeautyProcesserFactory:instan
  return [super application:application didFinishLaunchingWithOptio
}

@end
```

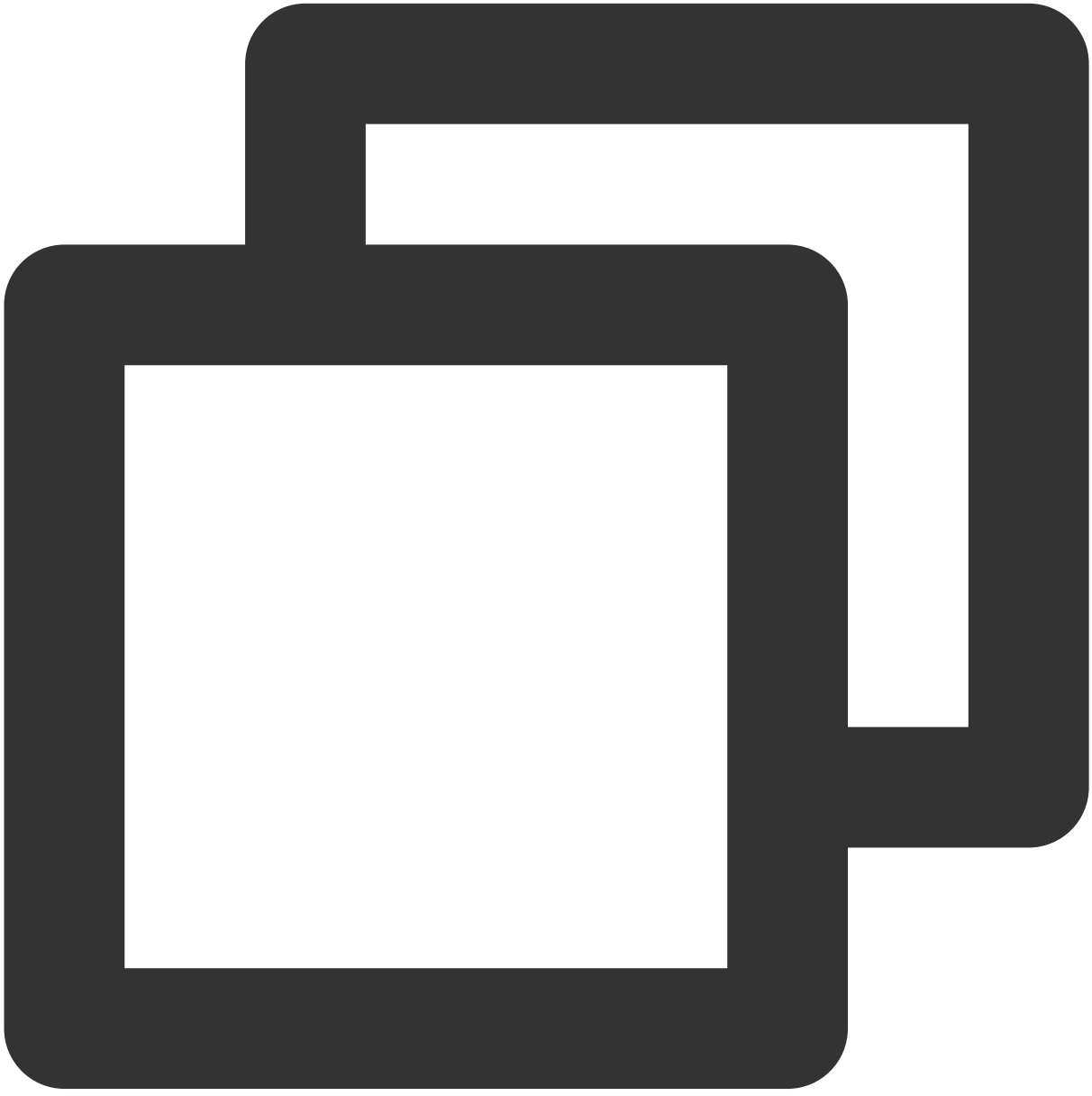
4단계: 리소스 초기화 API 호출



```
String dir =  await BeautyDataManager.getInstance().getResDir();
 TXLog.printlog('파일 경로: $dir');
 TencentEffectApi.getApi()?.initXmagic(dir,(reslut) {
   _isInitResource = reslut;
   callBack.call(reslut);
   if (!reslut) {
     Fluttertoast.showToast(msg: "리소스 초기화 실패");
   }
 }); TencentEffectApi.getApi()?.initXmagic((reslut) {
   if (!reslut) {
     Fluttertoast.showToast(msg: "리소스 초기화 실패");
```

```
        }
    });
```

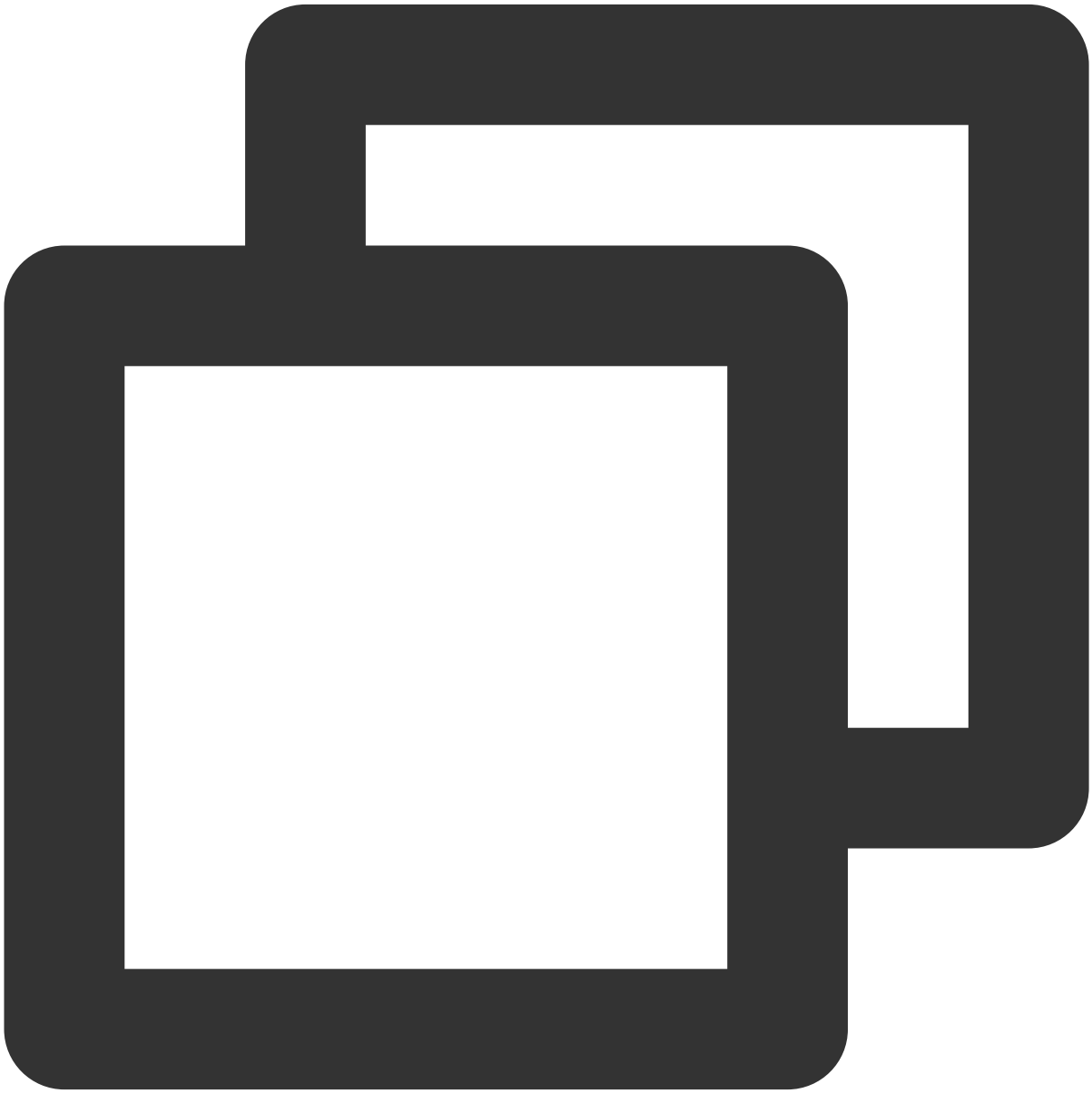
5단계: license 설정



```
TencentEffectApi.getApi()?.setLicense(licenseKey, licenseUrl,
            (errorCode, msg) {
        TXLog.printlog("인증 결과 출력 errorCode = $errorCode   msg = $msg");
```

```
        if (errorCode == 0) {
            //인증 성공
        }
    });
```

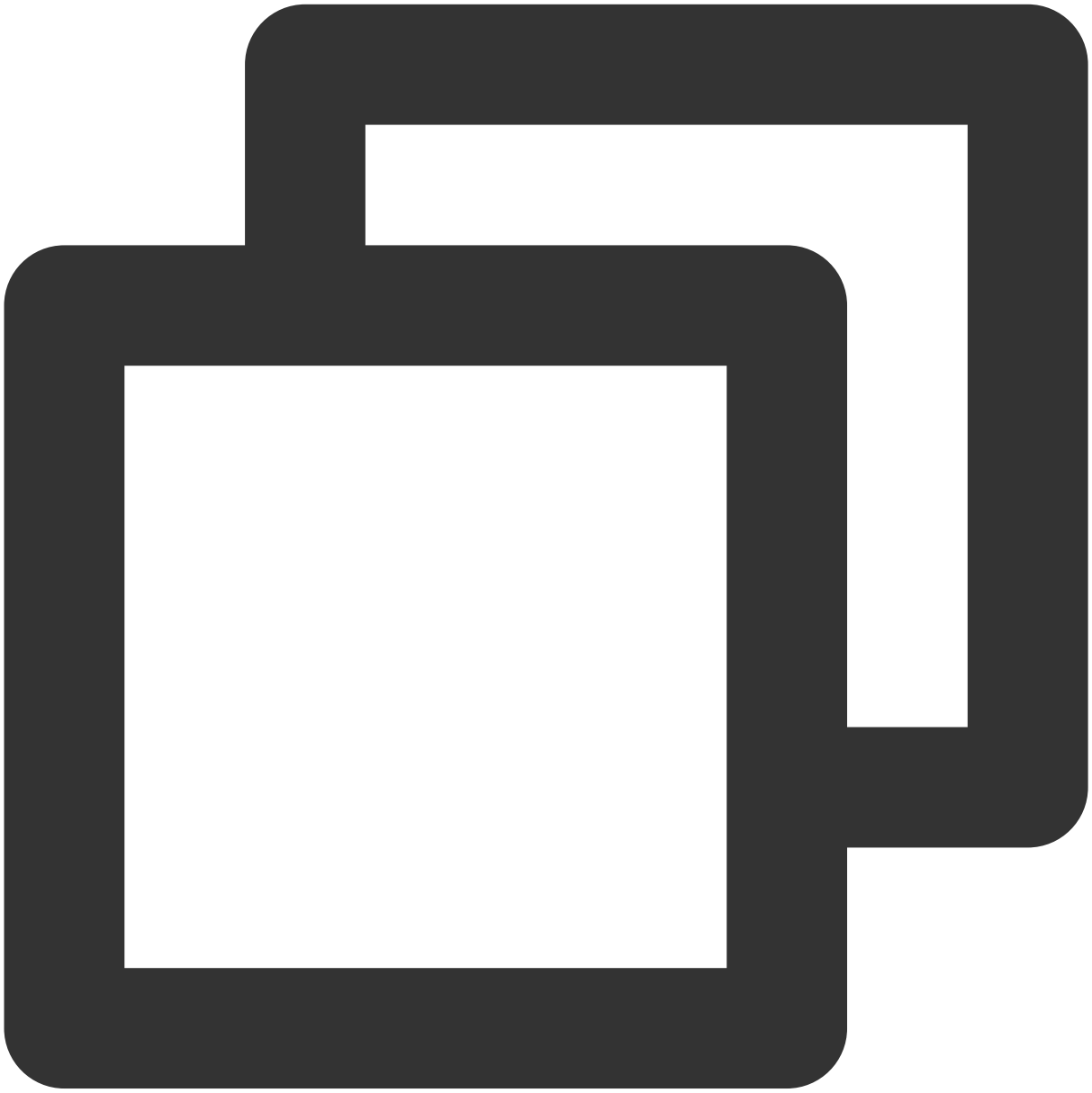
6단계: 뷰티 필터 활성화



```
///뷰티 필터 활성화
```

```
var enableCustomVideo = await trtcCloud.enableCustomVideoProcess(open);
```

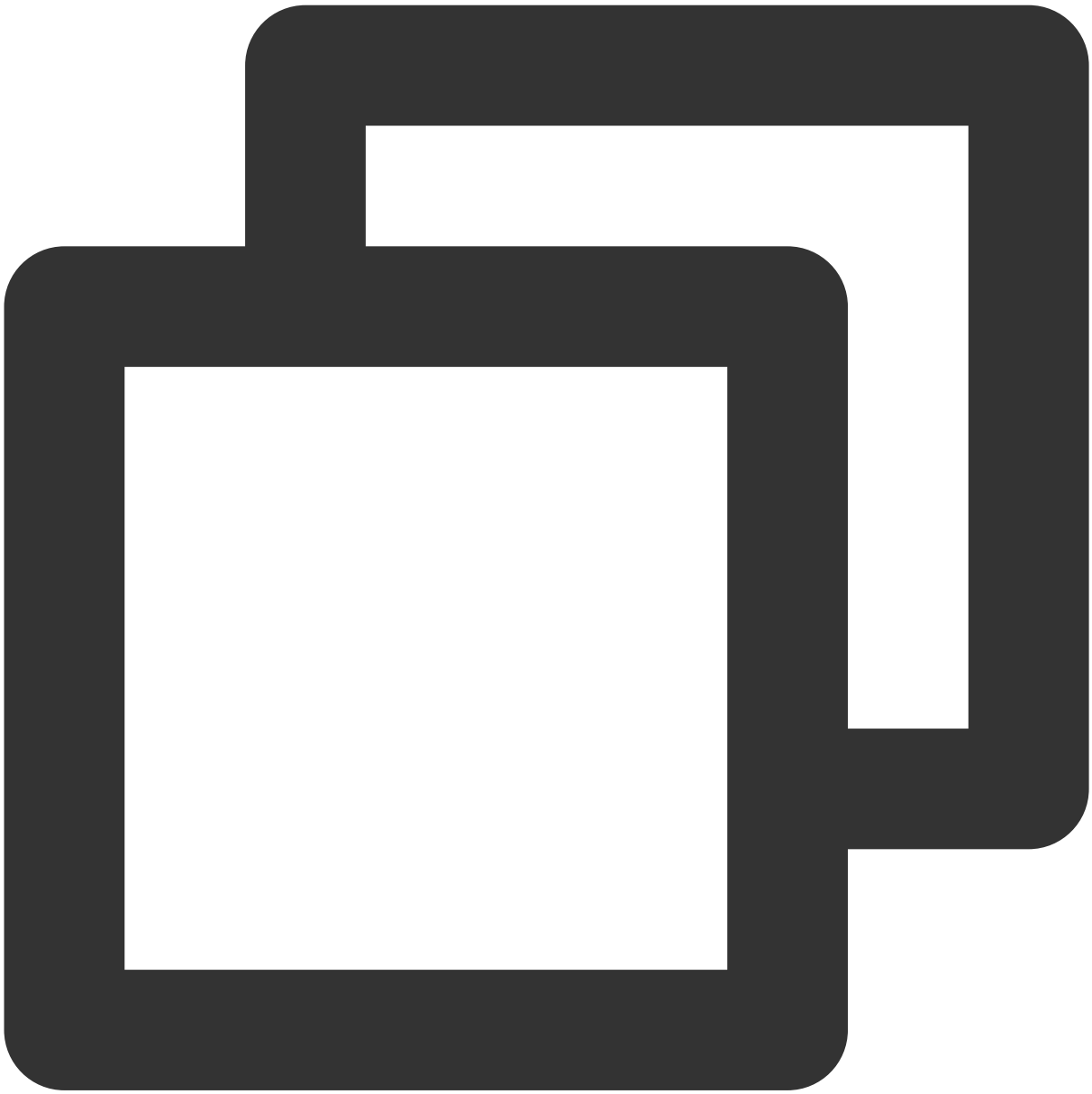
7단계: 뷰티 필터 속성 설정



```
TencentEffectApi.getApi()?.updateProperty(_xmagicProperty!);
///_xmagicProperty는 BeautyDataManager.getInstance().getAllPannelData();를 호출하여 도
```

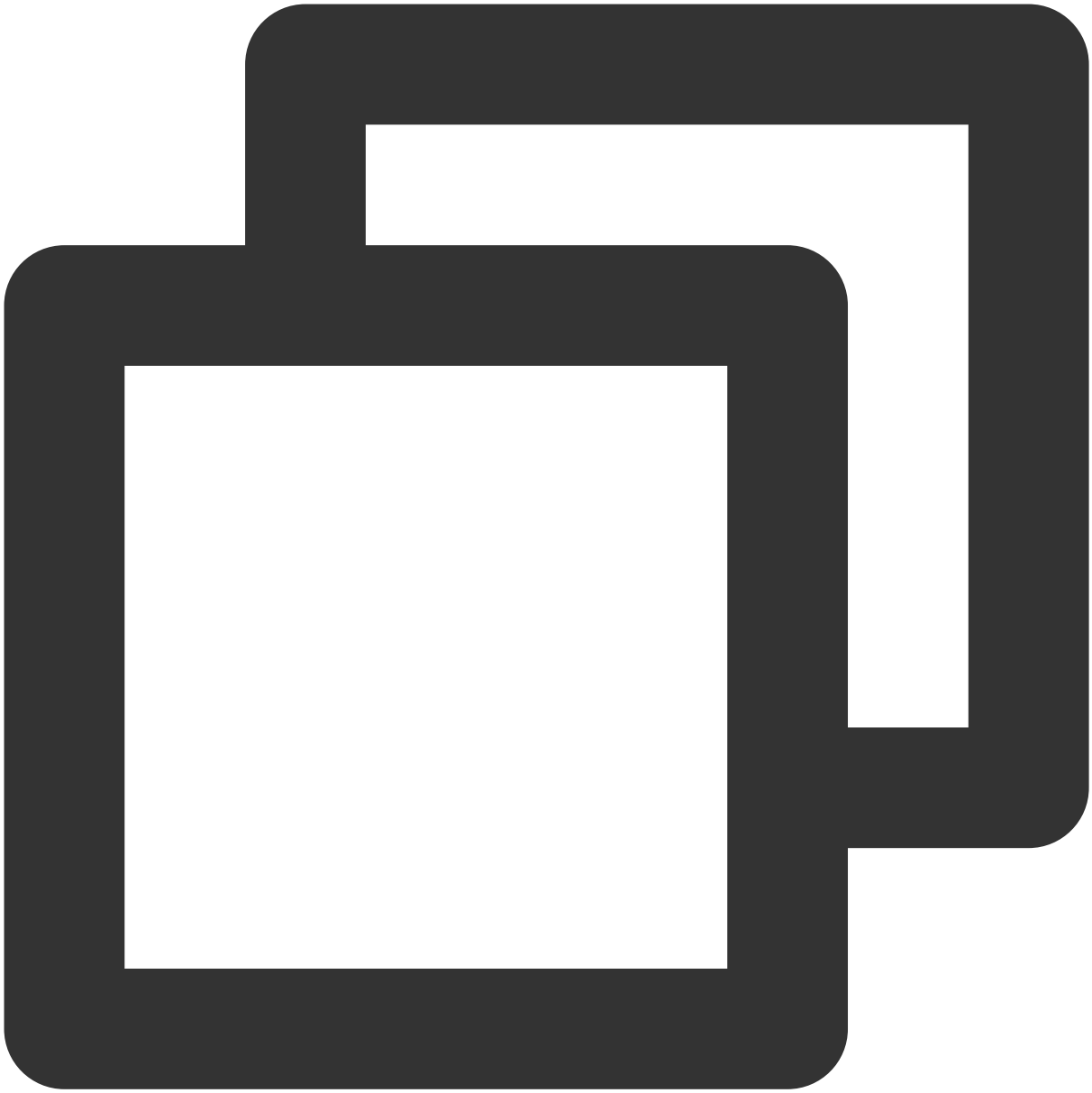
8단계: 기타 속성 설정

오디오 효과 일시 중지



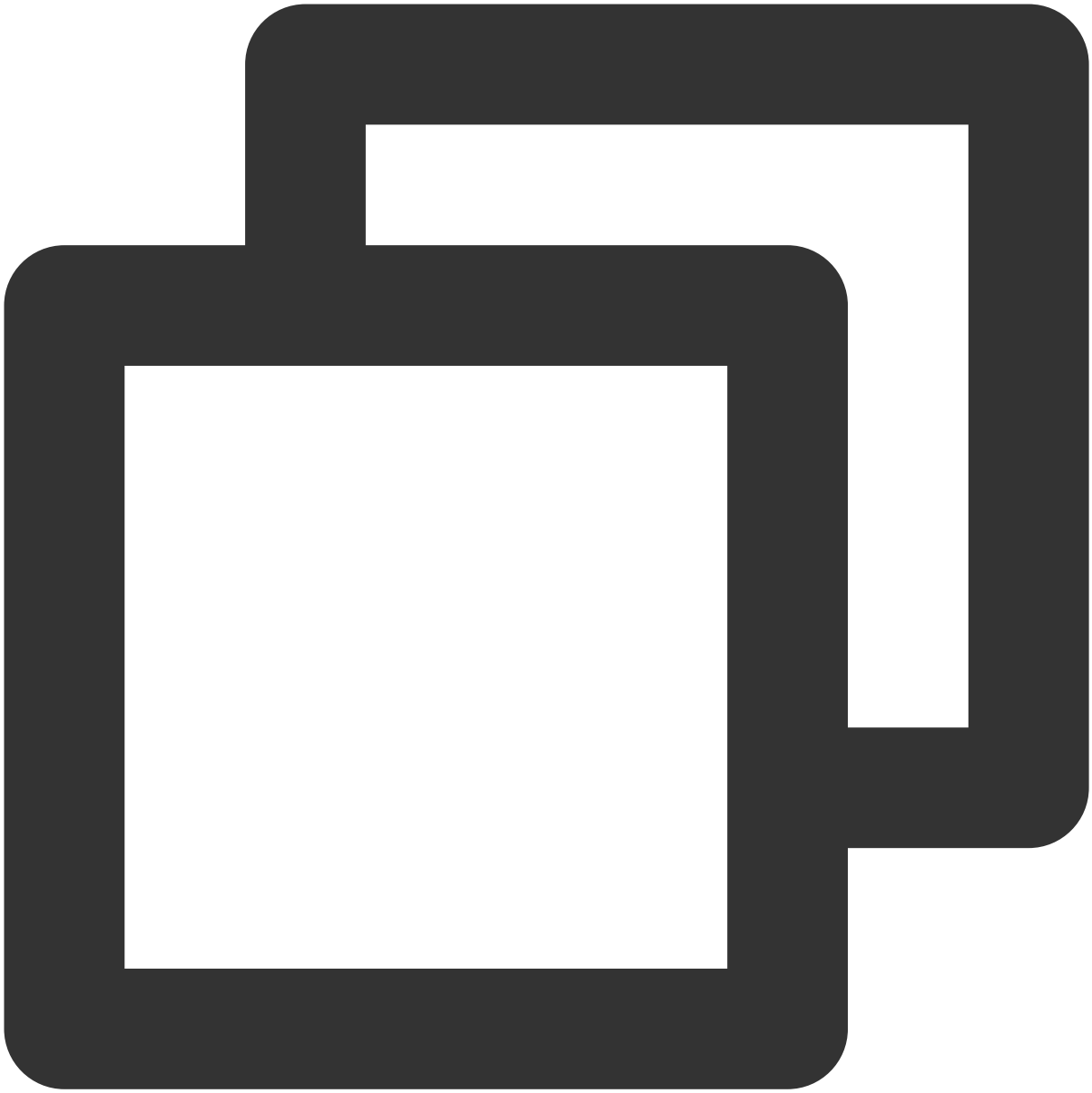
```
TencentEffectApi.getApi()?.onPause();
```

오디오 효과 재개



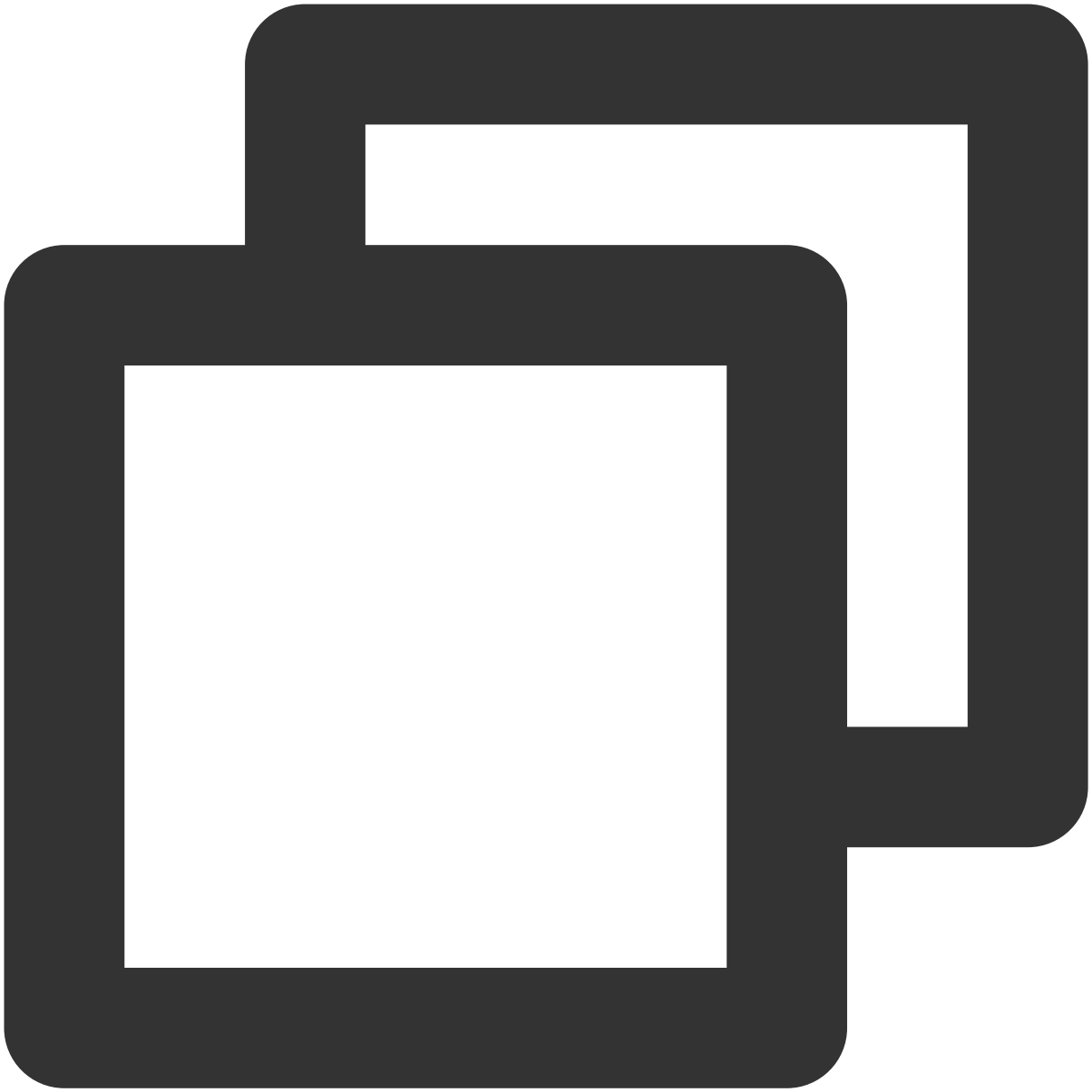
```
TencentEffectApi.getApi()?.onResume();
```

뷰티 필터 이벤트 수신



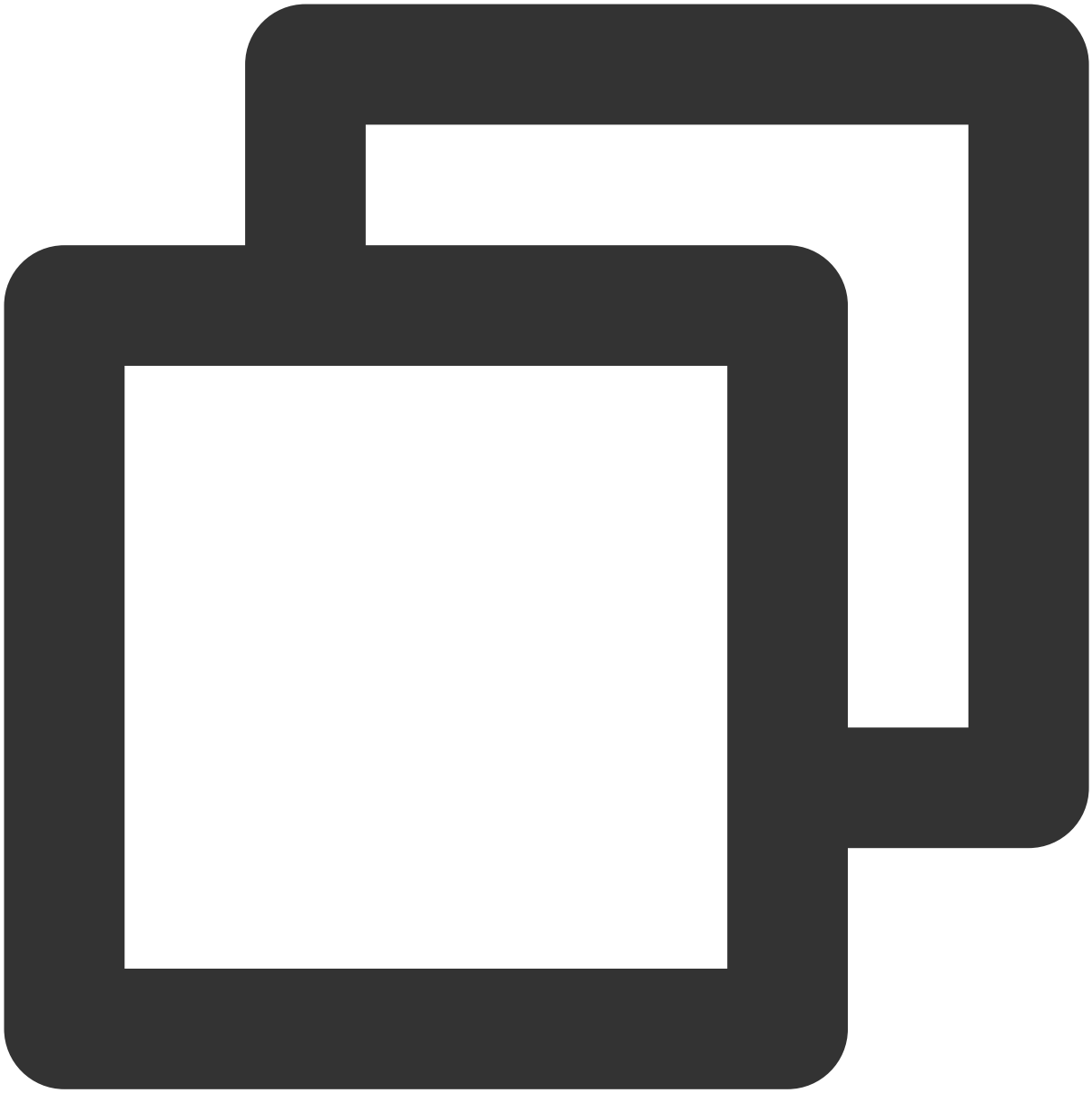
```
TencentEffectApi.getApi()
     ?.setOnCreateXmagicApiErrorListener((errorMsg, code) {
       TXLog.printlog("뷰티 필터 객체 생성 오류 errorMsg = $errorMsg , code = $code");
 });   ///효과 객체를 생성하기 전에 리스너를 설정해야 함
```

얼굴, 제스처, 신체 감지 결과 콜백 설정



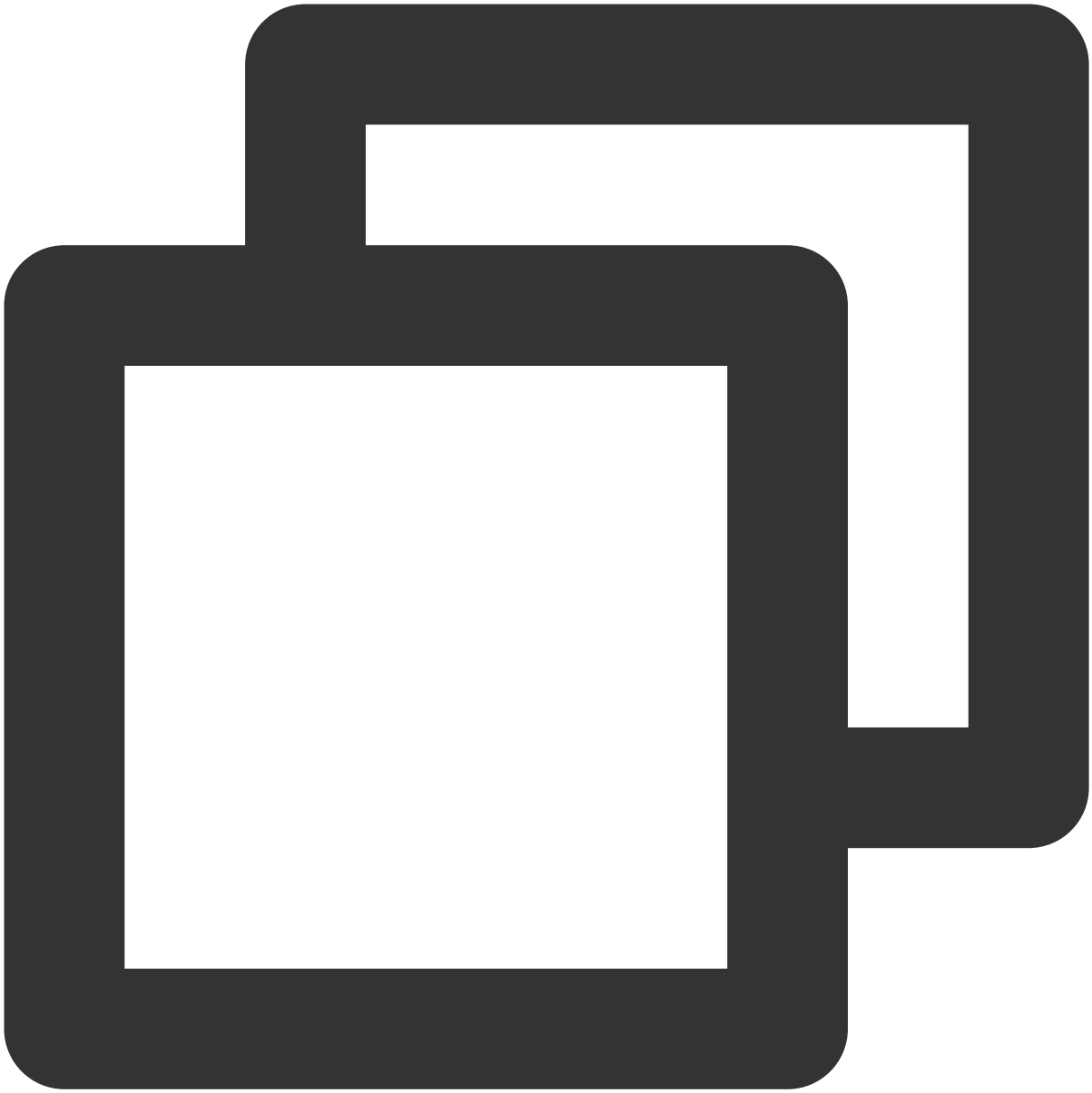
```
TencentEffectApi.getApi()?.setAIDataListener(XmagicAIDataListenerImp());
```

애니메이션 효과 팁에 대한 콜백 구성



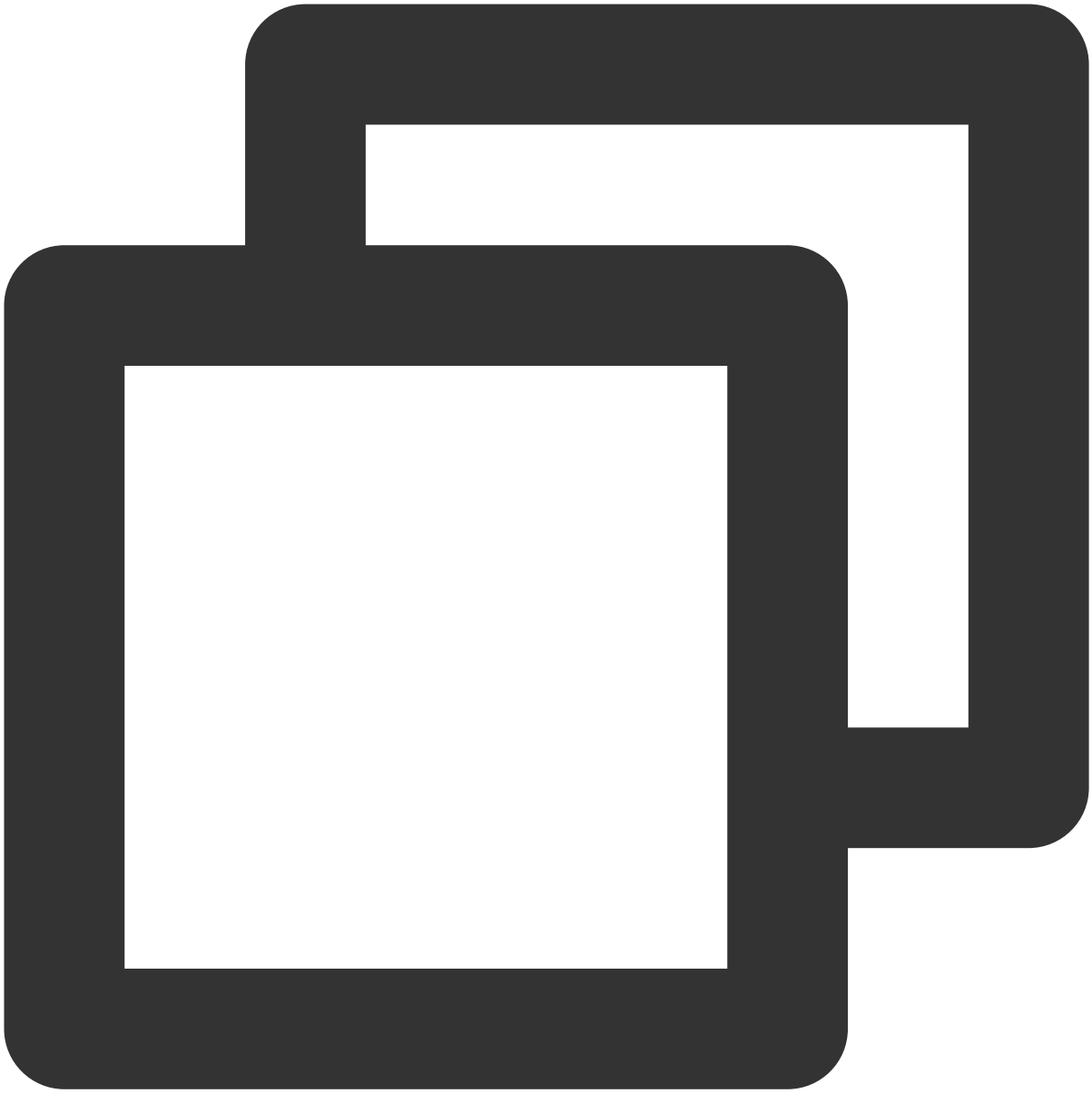
```
TencentEffectApi.getApi()?.setTipsListener(XmagicTipsListenerImp());
```

얼굴 키포인트 및 기타 데이터의 콜백 구성(S1-05 및 S1-06에서만 사용 가능)



```
TencentEffectApi.getApi()?.setYTDataListener((data) {
   TXLog.printlog("setYTDataListener  $data");
 });
```

모든 콜백을 제거합니다. 페이지를 종료할 때 모든 콜백을 제거해야 합니다.



```
TencentEffectApi.getApi()?.setOnCreateXmagicApiErrorListener(null);
TencentEffectApi.getApi()?.setAIDataListener(null);
TencentEffectApi.getApi()?.setYTDataListener(null);
TencentEffectApi.getApi()?.setTipsListener(null);
```

설명

API에 대한 자세한 내용은 API 문서를 참고하십시오. 기타는 Demo 프로젝트를 참고하십시오.

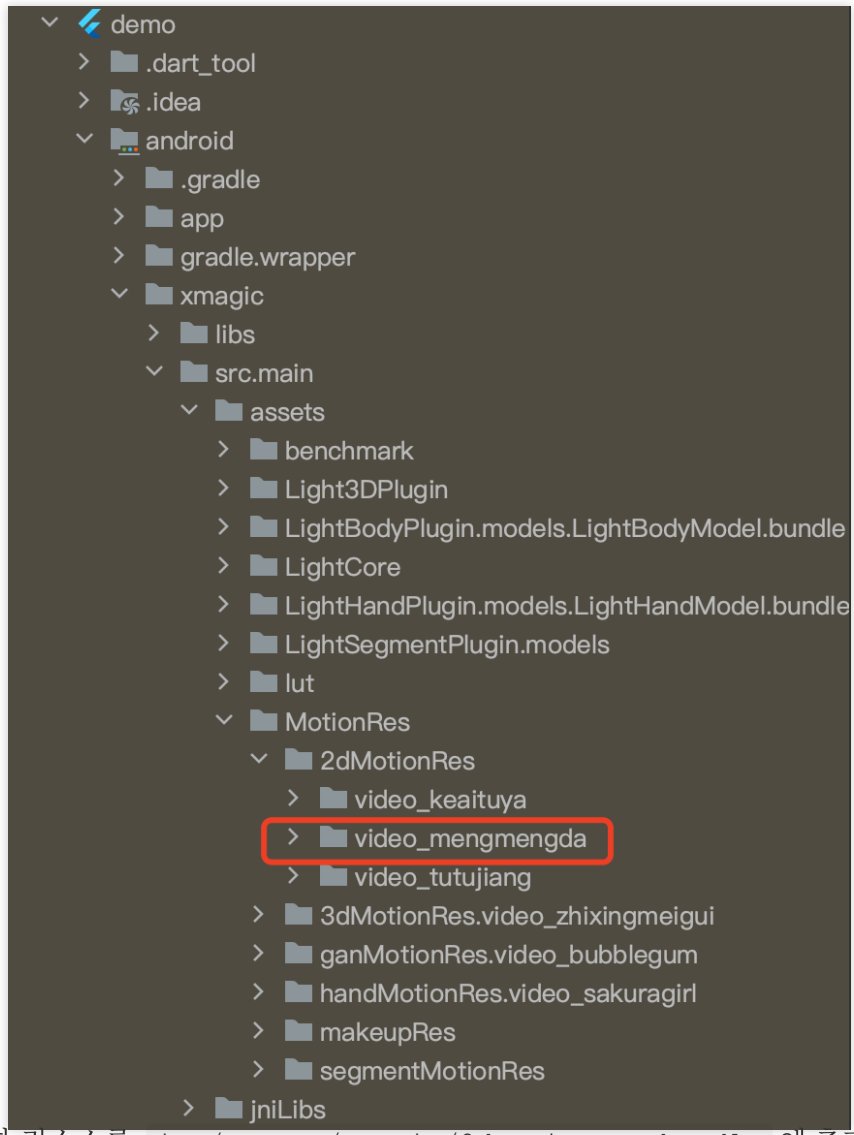
9단계: 뷰티 필터 패널에 데이터 추가 및 제거

BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid 및 BeautyPropertyProducerIOS 클래스에서 뷰티 필터 패널 데이터를 사용자 지정할 수 있습니다.

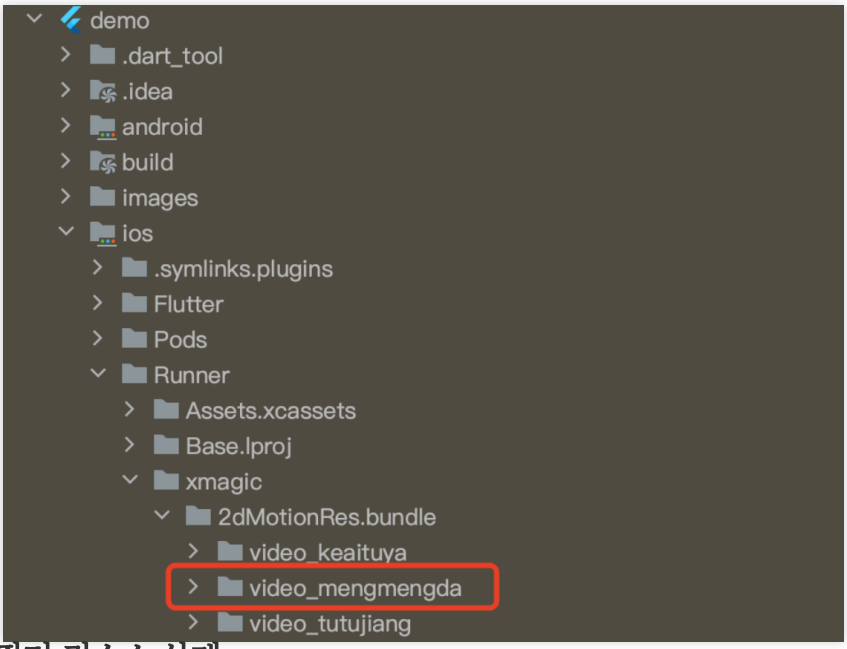
뷰티 필터 리소스 추가

1단계에서 설명한 대로 해당 폴더에 리소스 파일을 추가합니다. 예를 들어 2D 애니메이션 효과를 추가하려면 다음을 수행합니다.

1. 프로젝트의 `android/xmagic/src.mian/assets/MotionRes/2dMotionRes` 에 리소스를 넣습니다.



2. 또한 리소스를 `ios/Runner/xmagic/2dMotionRes.bundle` 에 추가합니다.



뷰티 필터 리소스 삭제

귀하의 License는 패널에서 일부 뷰티 필터 또는 신체 보정 효과를 지원하지 않을 수 있습니다. 지원되지 않는 기능은 삭제할 수 있습니다.
예를 들어 립스틱 효과를 삭제하려면 BeautyPropertyProducerAndroid와 BeautyPropertyProducerIOS의 getBeautyData에서 아래 코드를 삭제합니다.

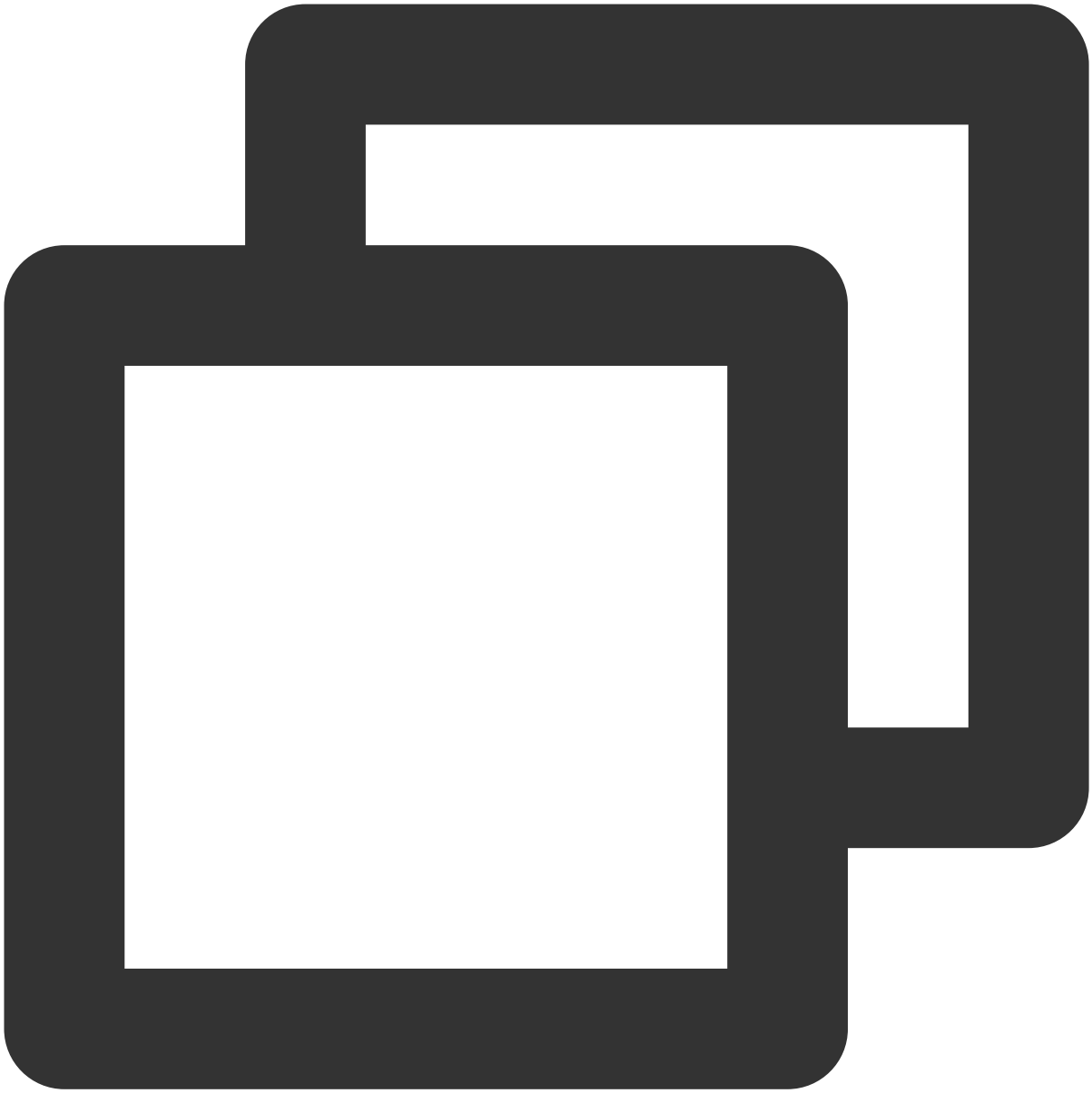
```
// Map<String, String> lipsResPathNames = {};
// lipsResPathNames["lips_fuguhong.png"] = "复古红";
// lipsResPathNames["lips_mitaose.png"] = "蜜桃色";
// lipsResPathNames["lips_shanhuju.png"] = "珊瑚橘";
// lipsResPathNames["lips_wenroufen.png"] = "温柔粉";
// lipsResPathNames["lips_huolicheng.png"] = "活力橙";
// List<XmagicUIProperty> itemLipsPropertys = [];
// String lipId = "beauty.lips.lipsMask";
// for (String ids in lipsResPathNames.keys) {
//   itemLipsPropertys.add(XmagicUIProperty(
//       uiCategory: Category.BEAUTY,
//       displayName: lipsResPathNames[ids]!,
//       id: lipId,
//       resPath: resPaths + ids,
//       thumbDrawableName: "beauty_lips",
//       effKey: BeautyConstant.BEAUTY_MOUTH_LIPSTICK,
//       effValue: XmagicPropertyValues(0, 100, 50, 0, 1),
//       rootDisplayName: "口红"));
// }
// XmagicUIProperty itemLips = XmagicUIProperty(
//     displayName: "口红",
//     thumbDrawableName: "beauty_lips",
//     uiCategory: Category.BEAUTY);
// itemLips.xmagicUIPropertyList = itemLipsPropertys;
// beautyList.add(itemLips);
```

Tencent Effect를 UGSV SDK에 통합하기
iOS

최종 업데이트 날짜: : 2023-02-27 14:27:24

통합 준비

1. Demo 패키지를 다운로드한 후 압축을 해제하고 `demo/XiaoShiPin/` 디렉터리의 xmagickit 폴더를 프로젝트의 Podfile 디렉터리에 복사합니다.
2. Podfile에 다음 종속성을 추가하고 `pod install` 을 실행하여 가져오기를 완료합니다.



```
pod 'xmagickit', :path => 'xmagickit/xmagickit.podspec'
```

3. Bundle ID를 신청한 테스트 인증과 동일하게 수정합니다.

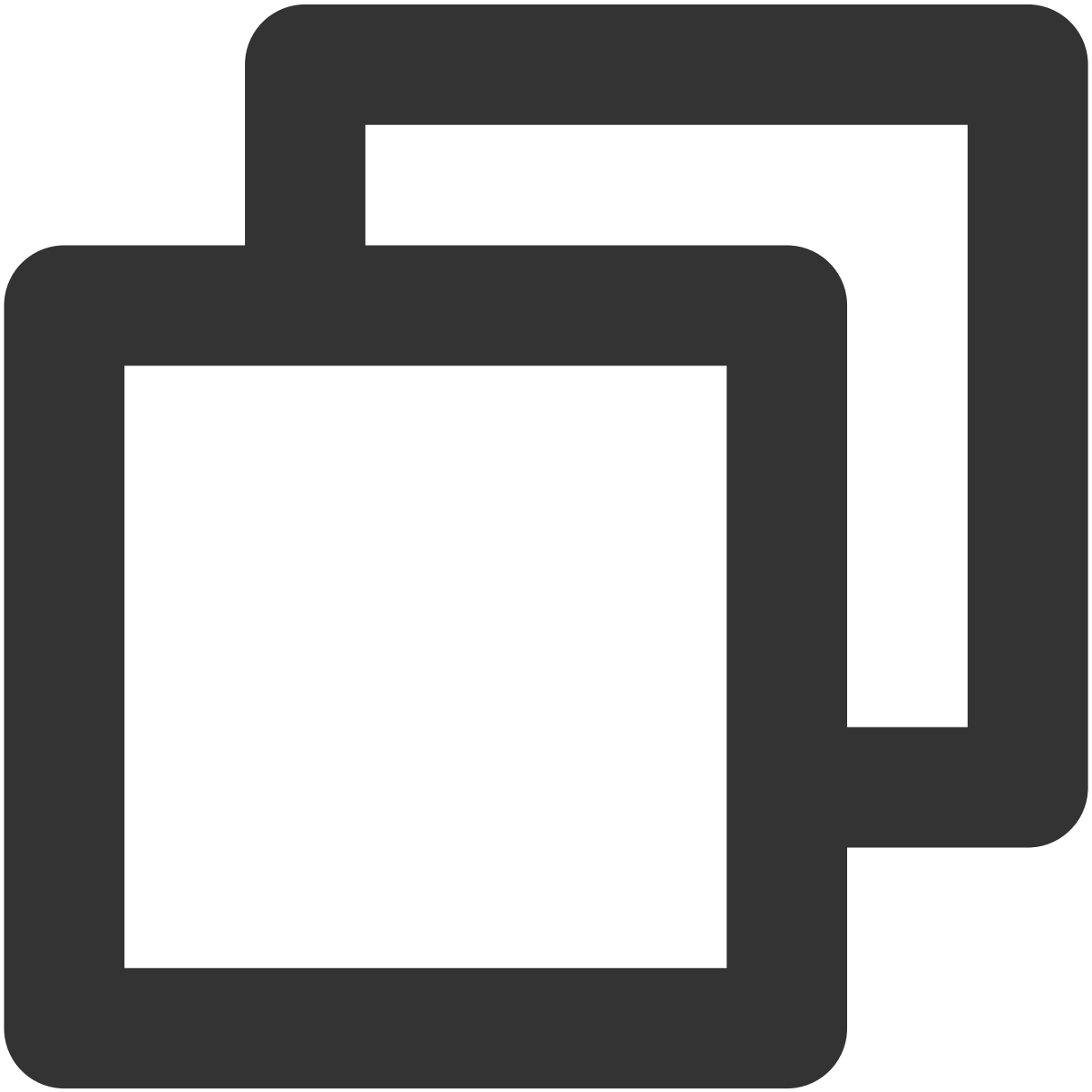
개발자 환경 요구 사항

Xcode 11 이상: App Store 또는 여기에서 다운로드하십시오.
권장 실행 환경:
기기 사양: iPhone 5 이상. iPhone 6 이하의 경우 전면 카메라는 720p까지 지원하며 1080p는 지원되지 않습니다.
시스템 요구 사항: iOS 12.0 이상.

SDK API 통합

1단계: 인증 초기화

`AppDelegate` 의 `didFinishLaunchingWithOptions` 에 다음 인증 코드를 추가합니다. Tencent Cloud 웹사이트에서 얻은 인증 정보에 따라 `LicenseURL` 및 `LicenseKey` 를 설정합니다(XMagic SDK 버전은 2.5.1 이전, `TELicenseCheck.h` 는 `XMagic.framework` 에; XMagicSDK 버전은 2.5.1 이상, `TELicenseCheck.h` 는 `YTCommonXMagic.framework` 에 있음).



```
[TXUGCBase setLicenceURL:LicenseURL key:LicenseKey];
```

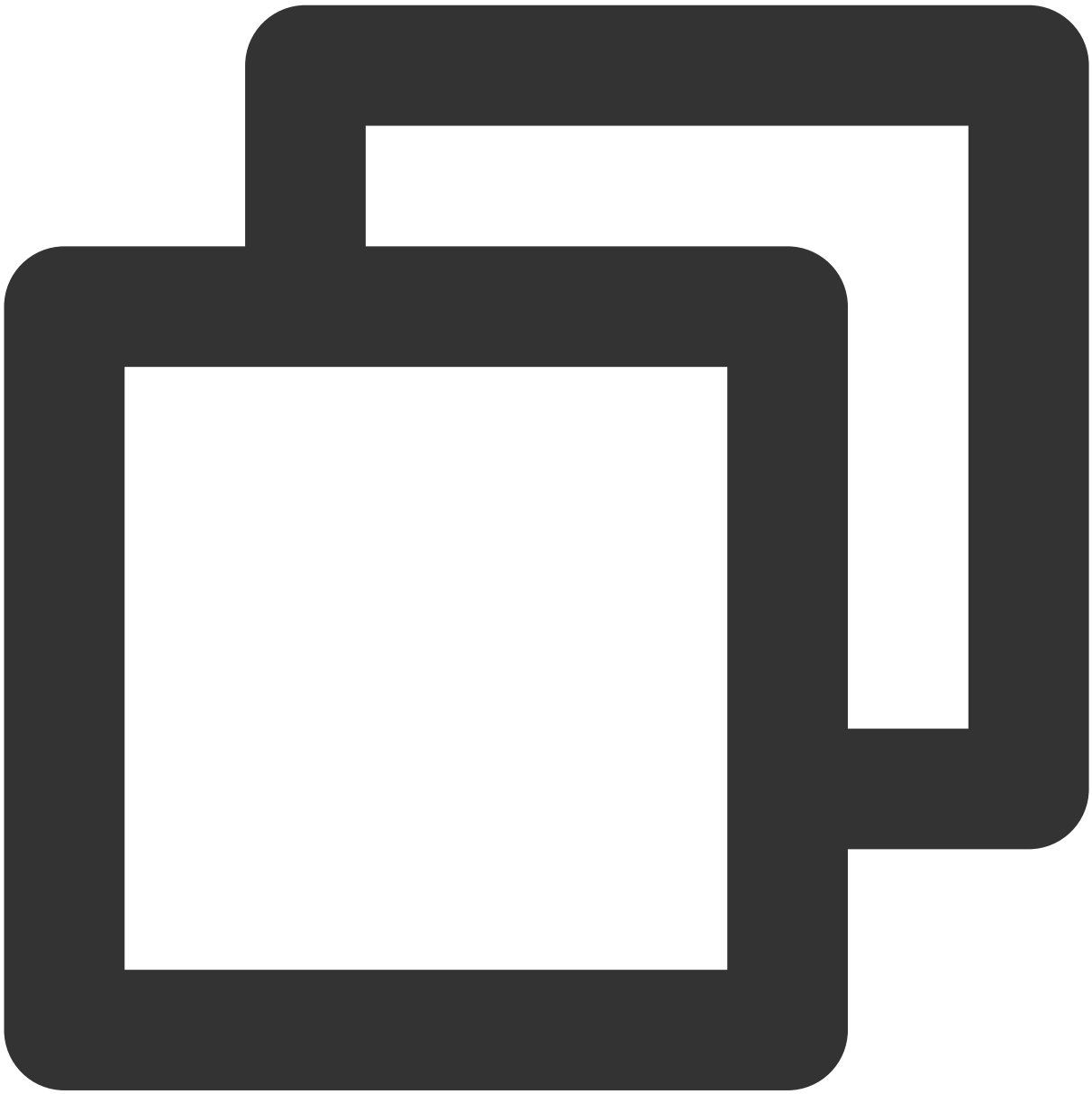
```
[TELicenseCheck setTELicense:LicenseURLkey:LicenseKey completion:^(NSInteger authre
              if (authresult == TELicenseCheckOk) {
                  NSLog(@"인증 성공");
               } else {
                  NSLog(@"인증 실패");
               }
       }];
```

errorCode 인증 설명:

에러 코드	설명
0	성공. Success
-1	입력 매개변수가 유효하지 않습니다. 예를 들어 URL 또는 KEY가 비어 있습니다.
-3	다운로드에 실패했습니다. 네트워크 설정을 확인하십시오.
-4	로컬 시스템에서 읽은 TE SDK 인증 정보가 비어 있으며 I/O 오류로 인해 발생할 수 있습니다
-5	읽은 VCUBE TEMP License 파일의 내용이 비어 있으며 I/O 오류로 인해 발생할 수 있습니다
-6	v_cube.license 파일의 JSON 필드가 올바르지 않습니다. 도움이 필요하면 Tencent Cloud에 문의하십시오.
-7	서명 인증에 실패했습니다. 도움이 필요하면 Tencent Cloud에 문의하십시오.
-8	복호화에 실패했습니다. 도움이 필요하면 Tencent Cloud에 문의하십시오.
-9	TELicense 필드의 JSON 필드가 올바르지 않습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-10	네트워크에서 리졸브된 TE 인증 정보가 비어 있습니다. 도움이 필요한 경우 Tencent Cloud에 문의하십시오.
-11	IO 실패로 인해 TE SDK 인증 정보를 로컬 파일에 쓰지 못했습니다
-12 <td>다운로드에 실패했으며 로컬 asset을 파싱하지 못했습니다	-13
인증 실패	기타

도움이 필요하면 Tencent Cloud에 문의하십시오	2단계: SDK 소재 리소스 경로 설정

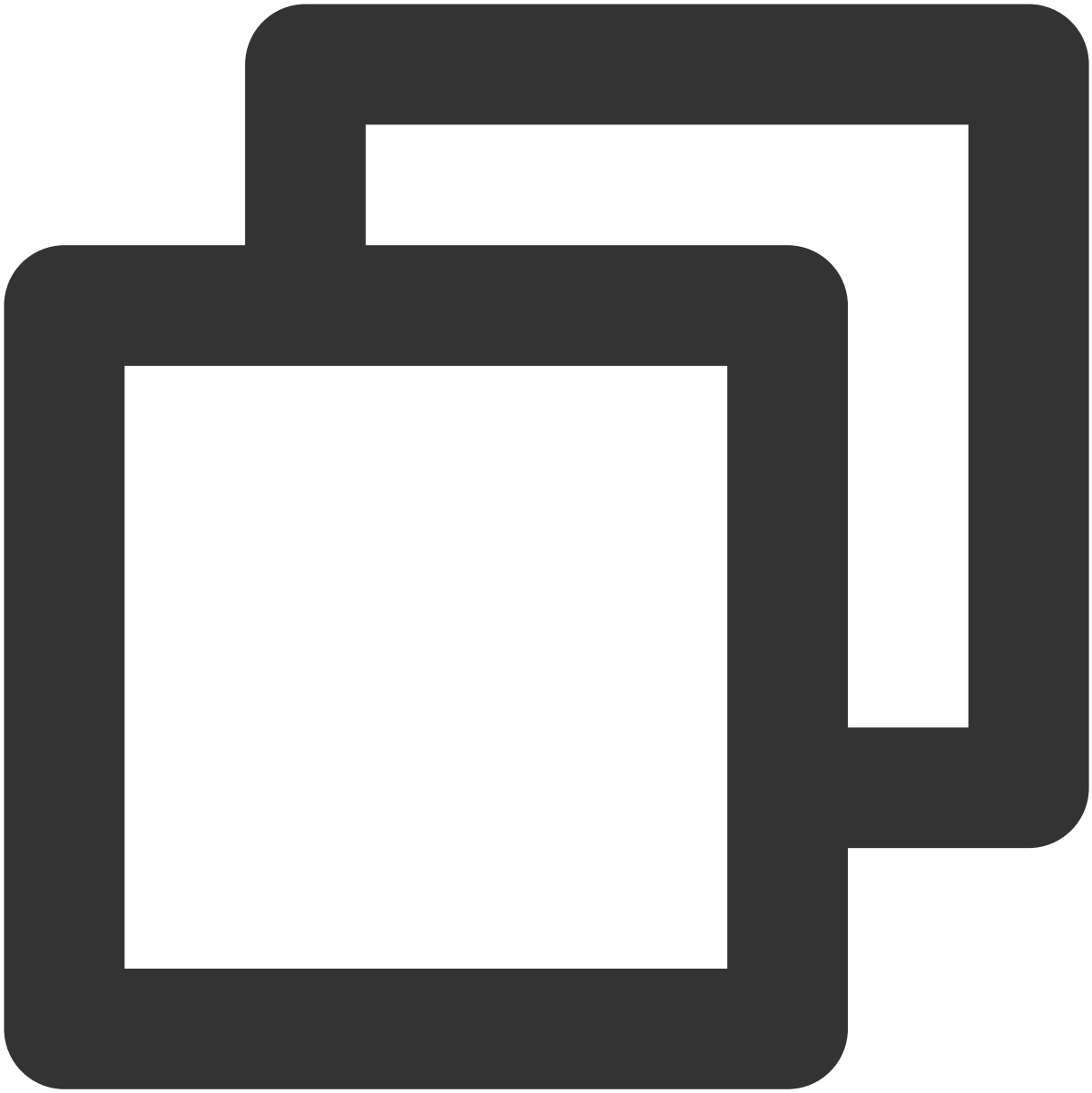
CGSize previewSize = [self getPreviewSizeByResolution:self.currentPreviewResolution];



```
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
```

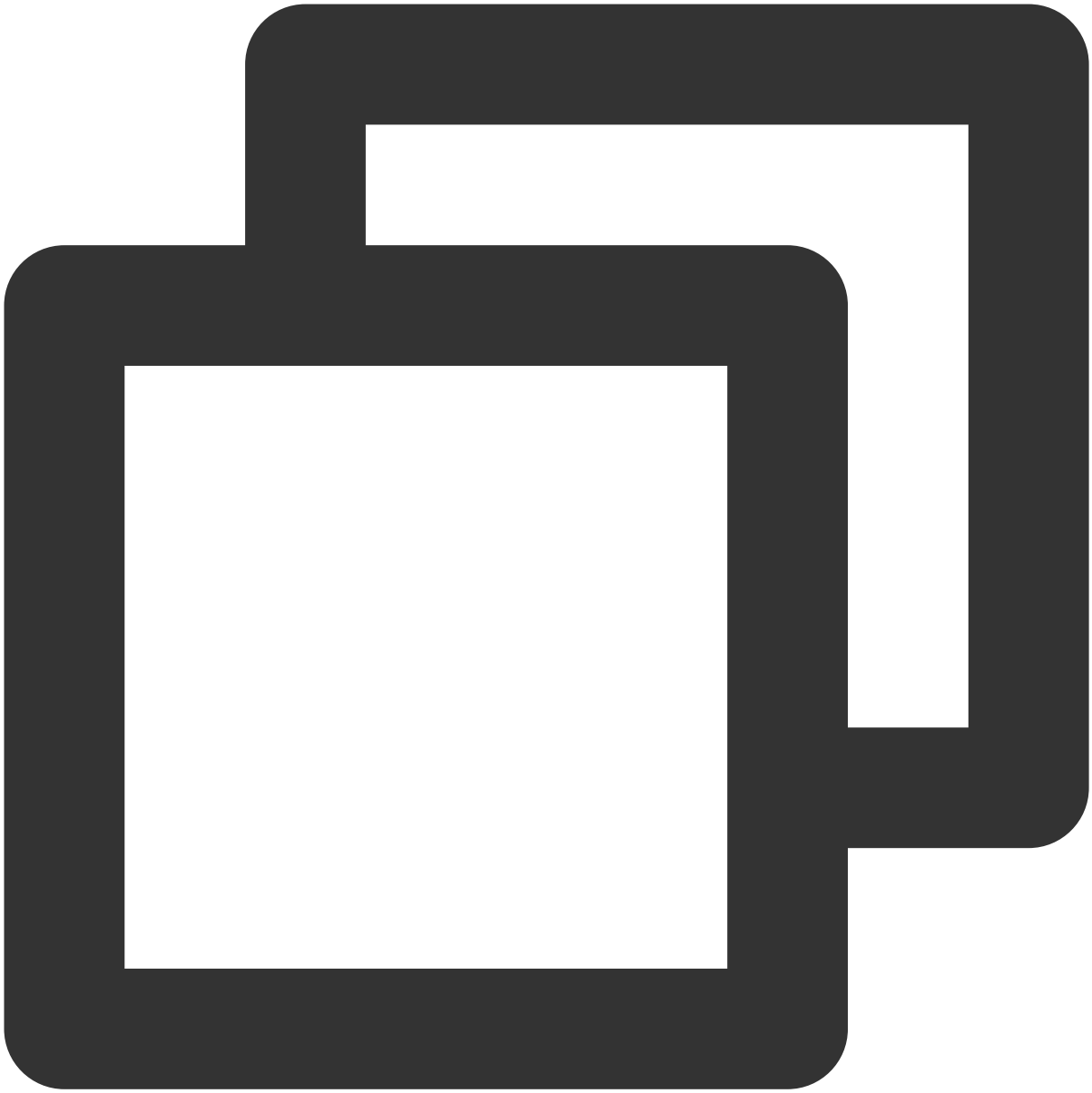
```
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                                       options:NSJSONReadingMutableContainers
                                                       error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Init beauty kit
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
### 3단계: 로그 및 이벤트 리스너 추가
```

// Register log



```
[self.beautyKit registerSDKEventListener:self];
[self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL];
### 4단계: 뷰티 필터 효과 구성
```

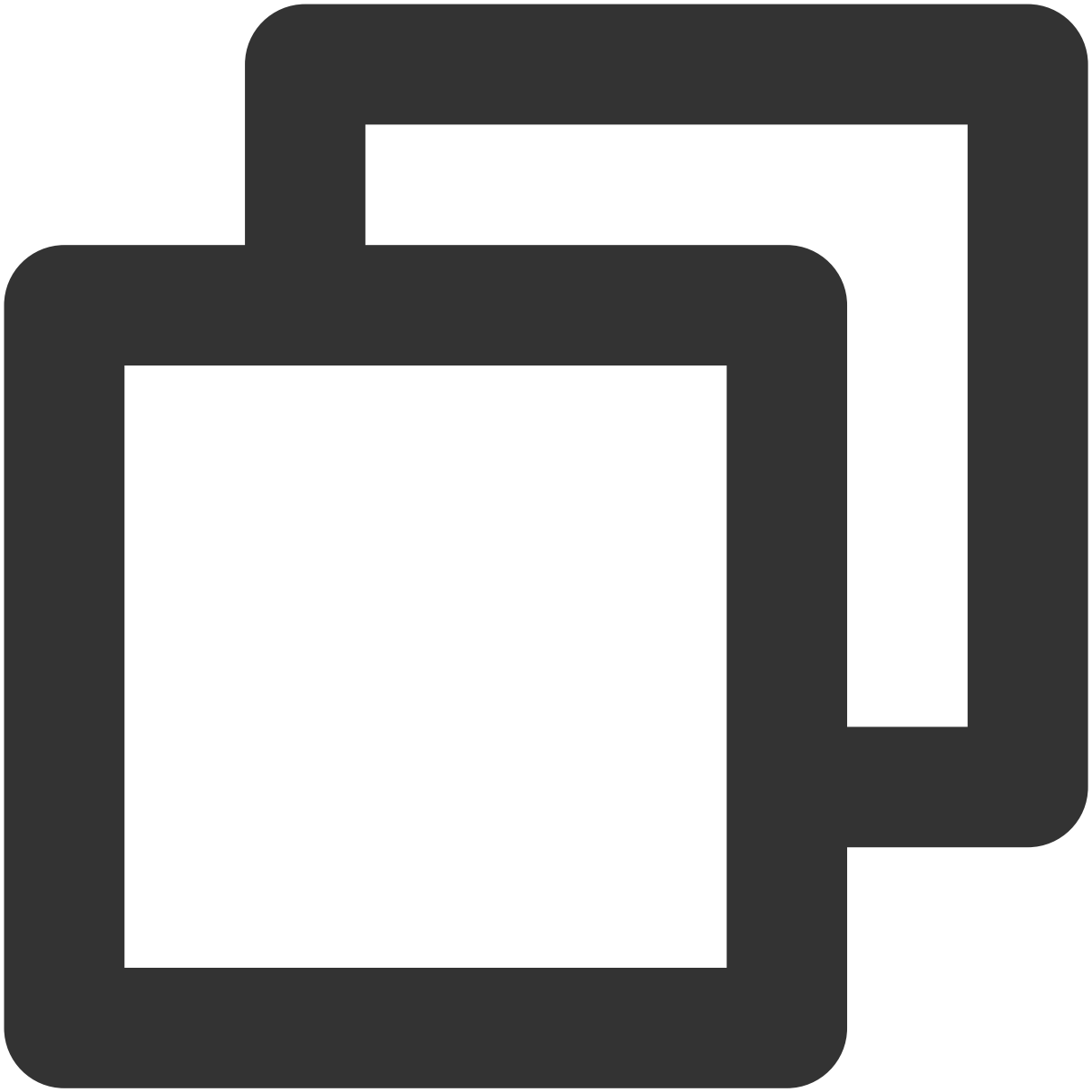
- (int)configPropertyWithType:(NSString _*Nonnull)propertyType withName:(NSString* _Nonnull)propertyName withData:(NSString*_Nonnull)propertyValue withExtraInfo:(id _Nullable)extraInfo;



```
### 5단계: 비디오 렌더링
```

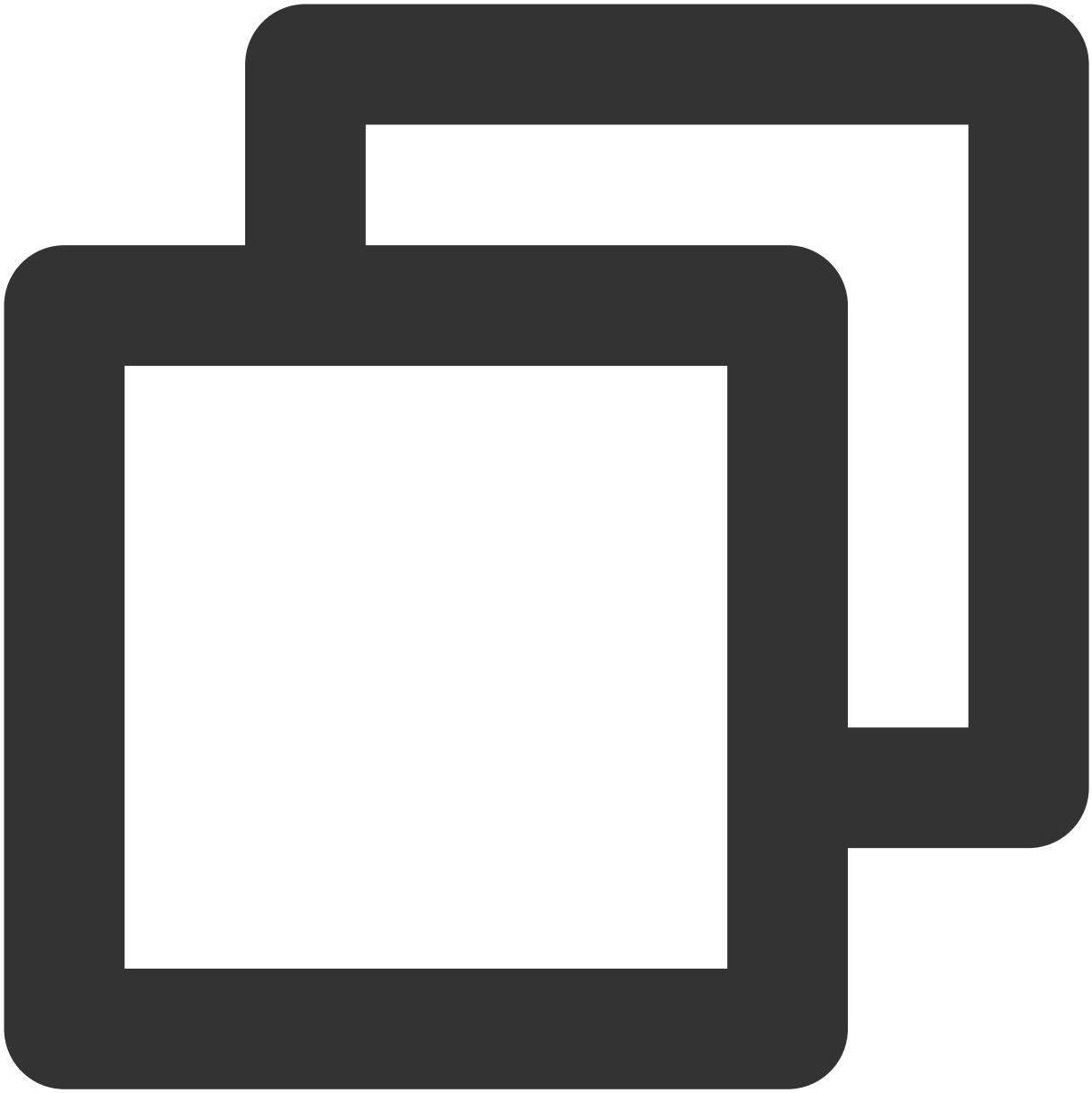
UGSV 사전 처리 프레임 콜백 인터페이스에서 YTProcessInput을 구성하고 textureId를 SDK에 전달하여 렌더링합니다.

[self.xMagicKit process:inputCPU withOrigin:YtLightImageOriginTopLeft withOrientation:YtLightCameraRotation0]



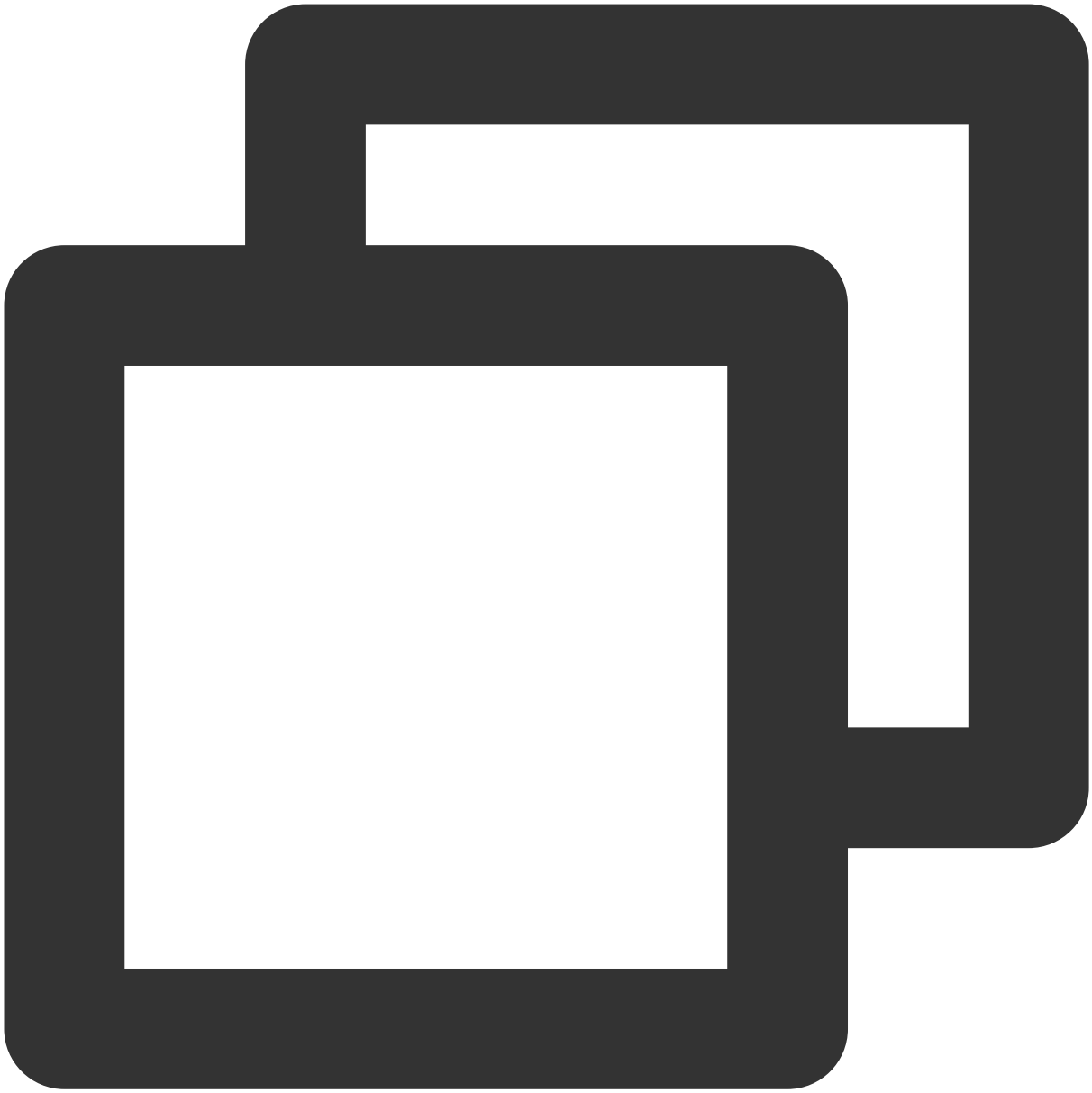
```
### 6단계: SDK 일시 중지/재개
```

[self.beautyKit onPause];



```
[self.beautyKit onResume];
### 7단계: 레이아웃에 SDK 뷰티 필터 패널 추가
```

UIEdgeInsets gSafeInset;



```
#if __IPHONE_11_0 && __IPHONE_OS_VERSION_MAX_ALLOWED >= __IPHONE_11_0
if(gSafeInset.bottom > 0){
}
if (@available(iOS 11.0, *)) {
    gSafeInset = [UIApplication sharedApplication].keyWindow.safeAreaInsets;
} else
#endif
    {
        gSafeInset = UIEdgeInsetsZero;
    }
```

```
dispatch_async(dispatch_get_main_queue(), ^{
    //뷰티 필터 옵션 UI
    _vBeauty = [[BeautyView alloc] init];
    [self.view addSubview:_vBeauty];
    [_vBeauty mas_makeConstraints:^(MASConstraintMaker *make) {
        make.width.mas_equalTo(self.view);
        make.centerX.mas_equalTo(self.view);
        make.height.mas_equalTo(254);
        if(gSafeInset.bottom > 0.0){  // 전체 화면으로 조정
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(0);
        } else {
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(-10);
        }
    }];
        _vBeauty.hidden = YES;
});
```

Android

최종 업데이트 날짜: : 2022-12-15 11:30:53

1단계: Demo 프로젝트 압축 해제

1. Tencent Effect SDK가 통합된 UGSV Demo를 다운로드합니다. 이 Demo는 Tencent Effect SDK S1-04 에디션을 기반으로 제작되었습니다.
2. Demo의 SDK 파일을 실제로 사용하는 SDK용 파일로 교체합니다. 구체적 작업은 다음 단계를 따르십시오.
   - `xmagickit module` 의 `build.gradle` 파일에서 다음을 찾습니다.

```
api 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
```

Tencent Effect SDK 통합하기(Android)에 설명된 대로 구매한 SDK 버전으로 교체합니다.

- SDK 버전에 애니메이션 효과 및 필터가 포함되어 있는 경우 해당 SDK 패키지를 다운로드하고 애니메이션 효과 및 필터에 대한 리소스를 `xmagickit module` 의 다음 디렉터리에 추가해야 합니다.
  - 애니메이션 효과: `../assets/MotionRes`
  - 필터: `../assets/lut`

3. Demo 프로젝트의 xmagickit 모듈을 실제 항목 프로젝트로 가져옵니다.

2단계: 패키지 이름 수정

app에서 build.gradle을 열고 applicationId를 평가판 라이선스에 바인딩된 패키지 이름으로 설정합니다.

3단계: SDK API 통합

Demo의 UGCKitVideoRecord 클래스를 참고하십시오.

1. **라이선스 설정:**

```
//인증 시 주의사항 및 오류 코드 내용은 다음을 참고하십시오. https://www.tencentcloud.com/document/product/1143/45385#.E6.AD.A5.E9.AA.A4.E4.B8.80.EF.BC.9A.E9.89.B4.E6.9D.83
XMagicImpl.checkAuth(new TELicenseCheck.TELicenseCheckListener() {
@Override
```

```
public void onLicenseCheckFinish(int errorCode, String msg) {
if (errorCode == TELicenseCheck.ERROR_OK) {
loadXmagicRes();
} else {
Log.e("TAG", "auth fail , please check auth url and key" + errorCode + " " + ms
g);
}
}
});
```

2. 리소스 초기화:

```
private void loadXmagicRes() {
if (XMagicImpl.isLoadedRes) {
XmagicResParser.parseRes(mActivity.getApplicationContext());
initXMagic();
return;
}
new Thread(new Runnable() {
@Override
public void run() {
XmagicResParser.copyRes(mActivity.getApplicationContext());
XmagicResParser.parseRes(mActivity.getApplicationContext());
XMagicImpl.isLoadedRes = true;
new Handler(Looper.getMainLooper()).post(new Runnable() {
@Override
public void run() {
initXMagic();
}
});
}
}).start();
}
```

3. UGSV와 뷰티 필터 바인딩:

```
private void initBeauty() {
TXUGCRecord instance = TXUGCRecord.getInstance(UGCKit.getAppContext());
instance.setVideoProcessListener(new TXUGCRecord.VideoCustomProcessListener() {
@Override
public int onTextureCustomProcess(int textureId, int width, int height) {
if (xmagicState == XMagicImpl.XmagicState.STARTED && mXMagic != null) {
return mXMagic.process(textureId, width, height);
}
```

```
return textureId;
}
@Override
public void onDetectFacePoints(float[] floats) {
}
@Override
public void onTextureDestroyed() {
if (Looper.getMainLooper() != Looper.myLooper()) { //메인 스레드가 아님
boolean stopped = xmagicState == XMagicImpl.XmagicState.STOPPED;
if (stopped || xmagicState == XMagicImpl.XmagicState.DESTROYED) {
if (mXMagic != null) {
mXMagic.onDestroy();
}
}
if (xmagicState == XMagicImpl.XmagicState.DESTROYED) {
TXUGCRecord.getInstance(UGCKit.getAppContext()).setVideoProcessListener(null);
}
}
}
});
}
```

4. **SDK 일시 중지/종료:**

onPause()는 뷰티 필터 일시 중지에 사용되며, Activity/Fragment 라이프사이클 메소드에서 구현할 수 있습니다. onDestroy 메소드는 GL 스레드에서 호출해야 합니다. (onTextureDestroyed 메소드에서 XMagicImpl 객체의 `onDestroy()` 호출 가능), 자세한 내용은 Demo의 `onTextureDestroyed` 를 참고하십시오.

```
@Override
public void onTextureDestroyed() {
if (Looper.getMainLooper() != Looper.myLooper()) { //메인 스레드가 아님
boolean stopped = xmagicState == XMagicImpl.XmagicState.STOPPED;
if (stopped || xmagicState == XMagicImpl.XmagicState.DESTROYED) {
if (mXMagic != null) {
mXMagic.onDestroy();
}
}
if (xmagicState == XMagicImpl.XmagicState.DESTROYED) {
TXUGCRecord.getInstance(UGCKit.getAppContext()).setVideoProcessListener(null);
}
}
}
```

5. **뷰티 필터 패널에 대한 레이아웃 추가:**

```
<RelativeLayout
android:id="@+id/panel_layout"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:visibility="gone"/>
```

6. **뷰티 필터 객체를 생성하고 뷰티 필터 패널을 추가합니다.**

```
private void initXMagic() {
if (mXMagic == null) {
mXMagic = new XMagicImpl(mActivity, getBeautyPanel());
} else {
mXMagic.onResume();
}
}
```

자세한 지침은 Demo의 UGCKitVideoRecord 클래스를 참고하십시오.

가상 Avatar 통합 가이드
iOS
Avatar 통합

최종 업데이트 날짜: : 2023-02-27 14:18:15

Avatar는 Tencent Effect SDK의 기능입니다. 사용하기 위해서는 먼저 SDK 통합 후 Avatar 소재를 추가해야 합니다. SDK 통합 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

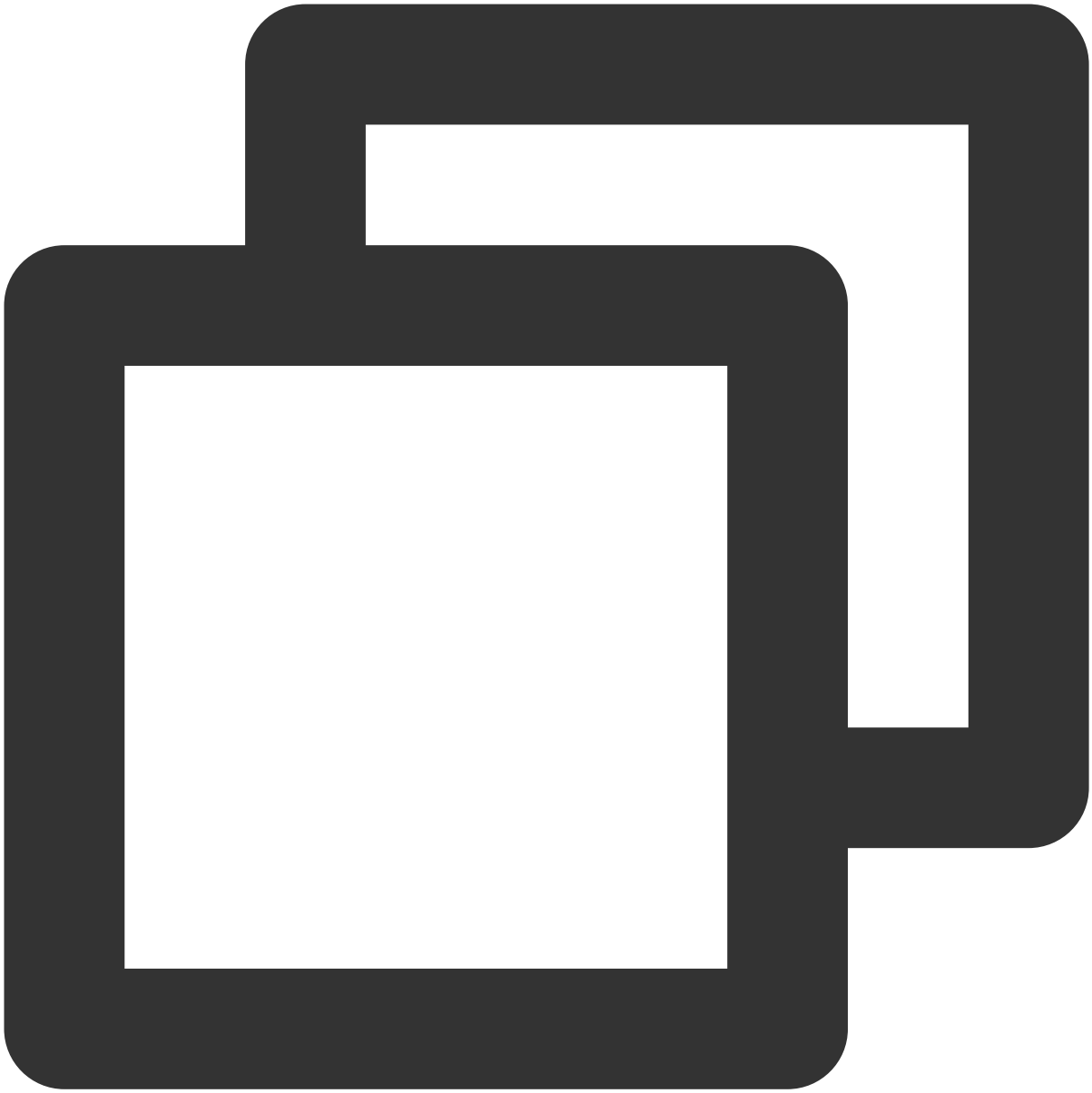
1단계: Avatar 소재 준비

1. Tencent Effect SDK를 통합합니다.
2. 웹 사이트에서 Demo 프로젝트를 다운로드하고 압축을 풉니다.
3. Demo의 `BeautyDemo/bundle/avatarMotionRes.bundle` 을 프로젝트에 복사합니다.

2단계: Demo UI 통합

통합 방법

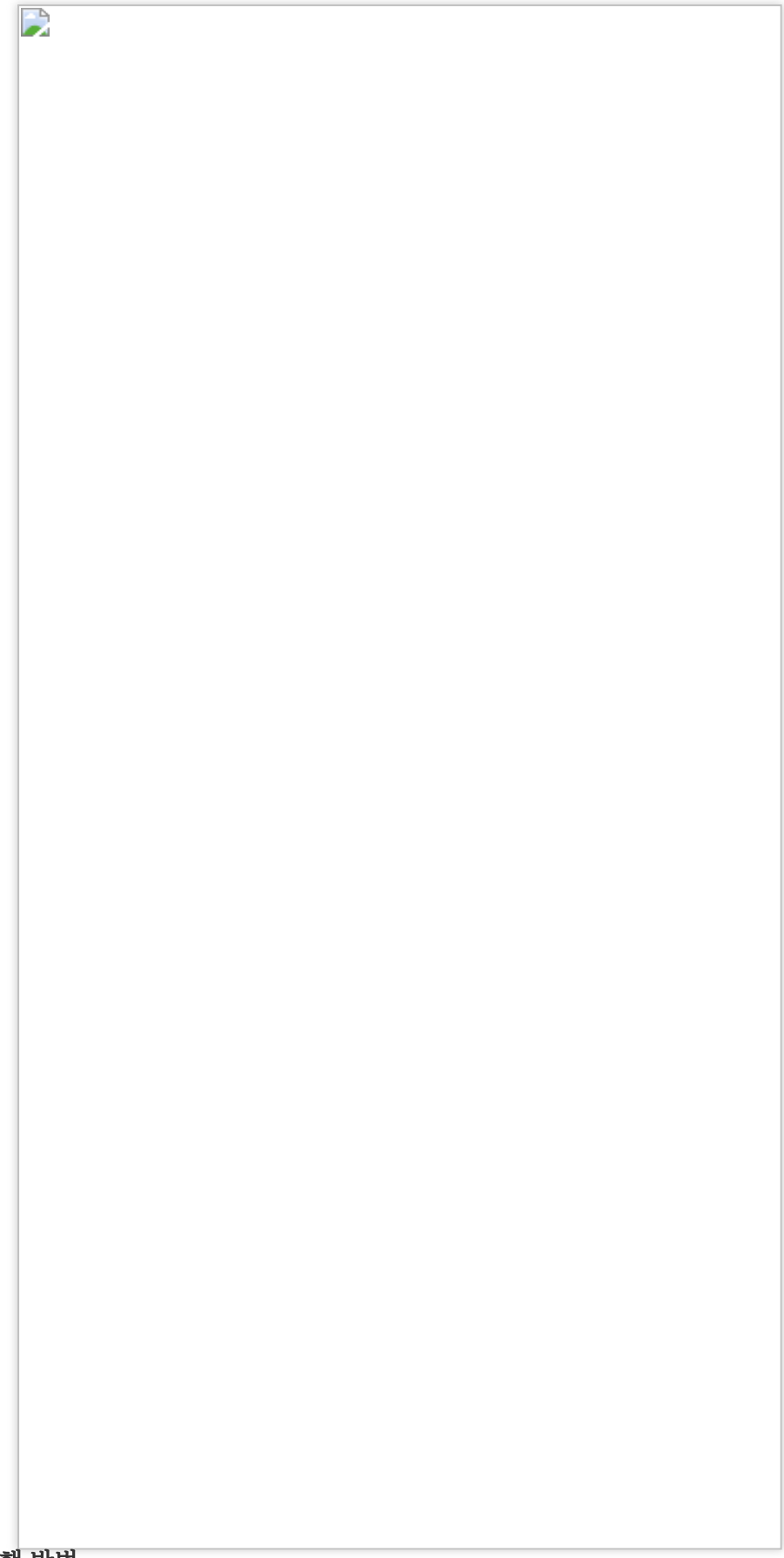
1. 프로젝트에서 BeautyDemo와 동일한 Avatar UI를 사용하려면 다음을 수행합니다.
2. Demo의 `BeautyDemo/Avatar` 폴더에 있는 모든 클래스를 프로젝트에 복사하고 다음 코드를 추가합니다.



```
AvatarViewController *avatarVC = [[AvatarViewController alloc] init];
avatarVC.modalPresentationStyle = UIModalPresentationFullScreen;
avatarVC.currentDebugProcessType = AvatarPixelData; // 이미지 또는 텍스처 Id
[self presentViewController:avatarVC animated:YES completion:nil];
```

Demo UI

1. UI



2. 실행 방법

조작 패널의 데이터는 JSON 파일을 파싱하여 얻습니다. Demo의 `BeautyDemo/Avatar/` 디렉터리에서 이 파일을 찾을 수 있습니다.

JSON 구조와 패널 요소 간의 매핑

head는 최상위 메뉴의 첫 번째 icon에 해당합니다.

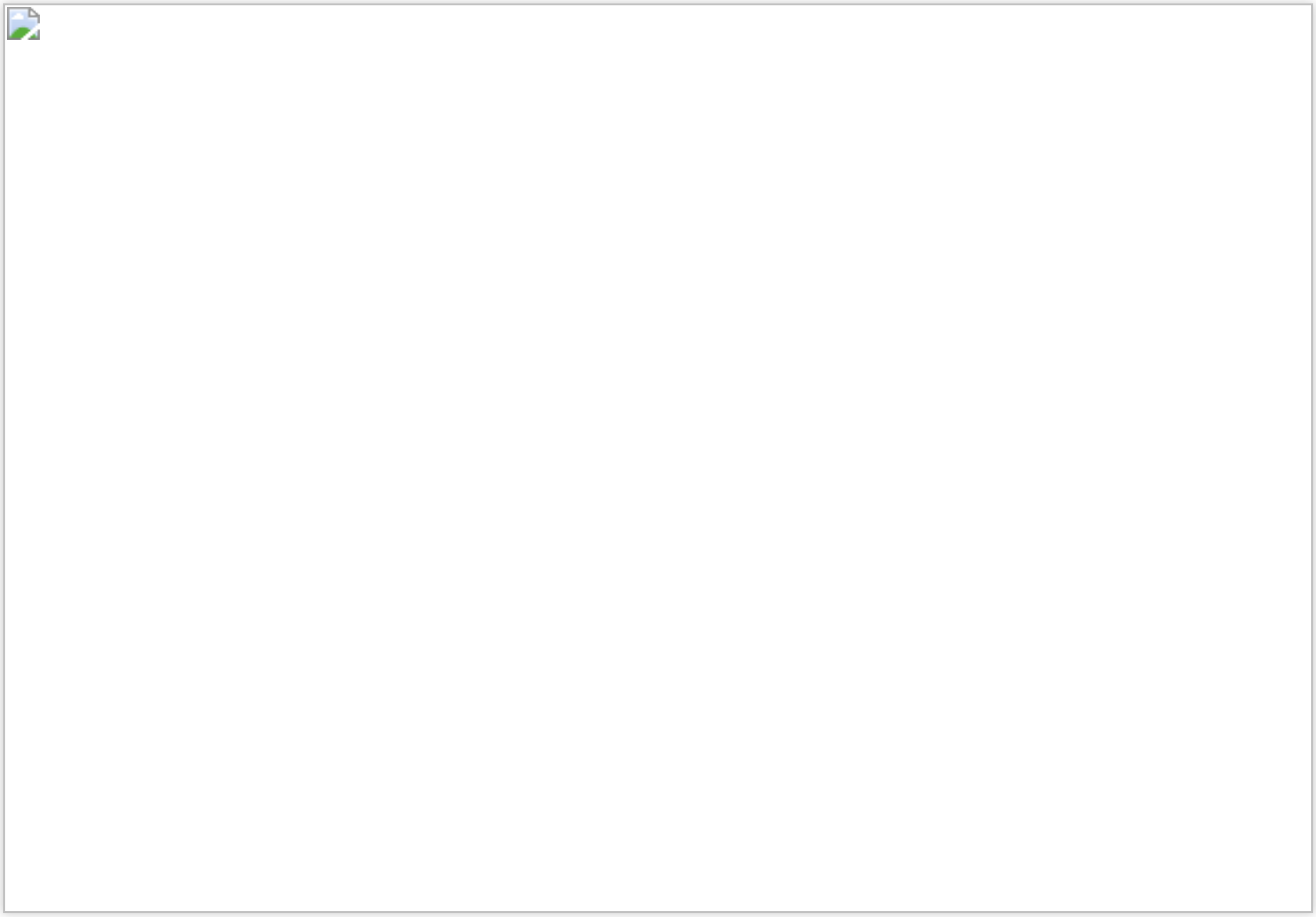
subTabs는 2단계 메뉴에 해당합니다.

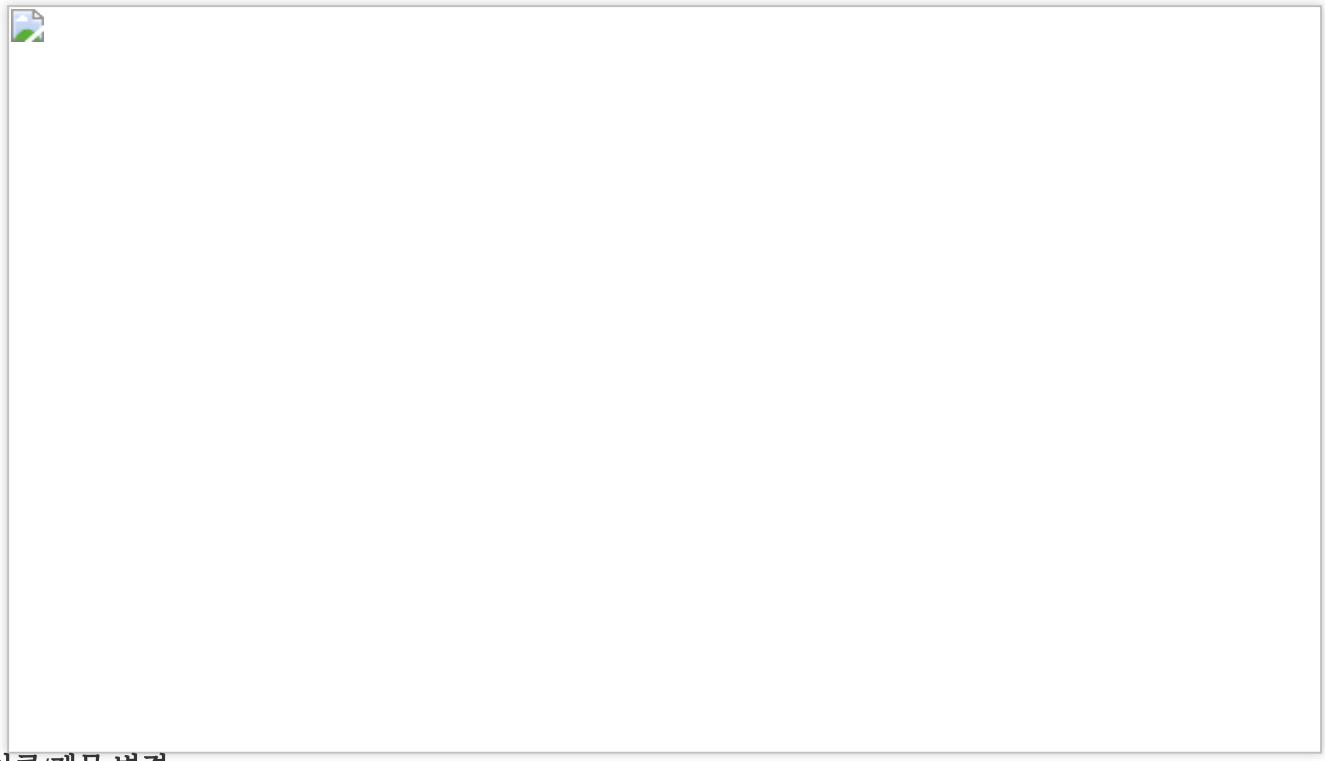
items는 3단계 메뉴에 해당합니다.

SDK API용 Avatar 객체 데이터와 패널 데이터 연결

아래의 첫 번째 스크린샷은 SDK에서 얻은 avatar 사전입니다(key는 category를 나타내고 value는 avatar 데이터 배열입니다). 두 번째 스크린샷은 패널 데이터입니다. 사용자가 패널에서 아이콘을 탭하면 2단계 category(**빨간색 상자**)에

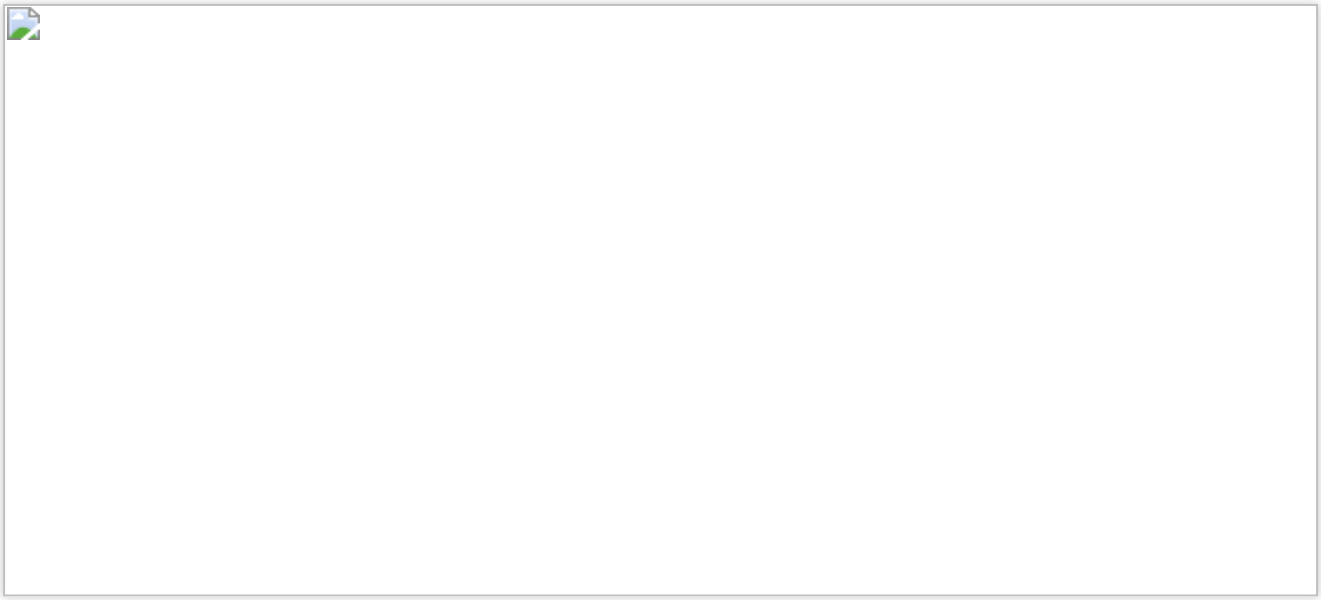
서 category를 가져오고 SDK에서 반환하는 avatar 사전에서 해당 카테고리의 avatarData 배열을 가져옵니다. 3단계 제목(**파란색 상자**)에서 ID를 가져온 다음 카테고리의 avatarData 배열에서 avatar 객체를 찾습니다. SDK의 updateAvatar API에 객체를 전달하여 avatar를 편집합니다.





3. 아이콘/제목 변경

Demo UI의 위 JSON 파일을 수정하여 UI의 icon이나 제목을 변경할 수 있습니다. 예를 들어 최상위 메뉴에서 **헤더** icon을 변경하려면 아래의 iconUrl 또는 checkedIconUrl 값을 수정하면 됩니다.



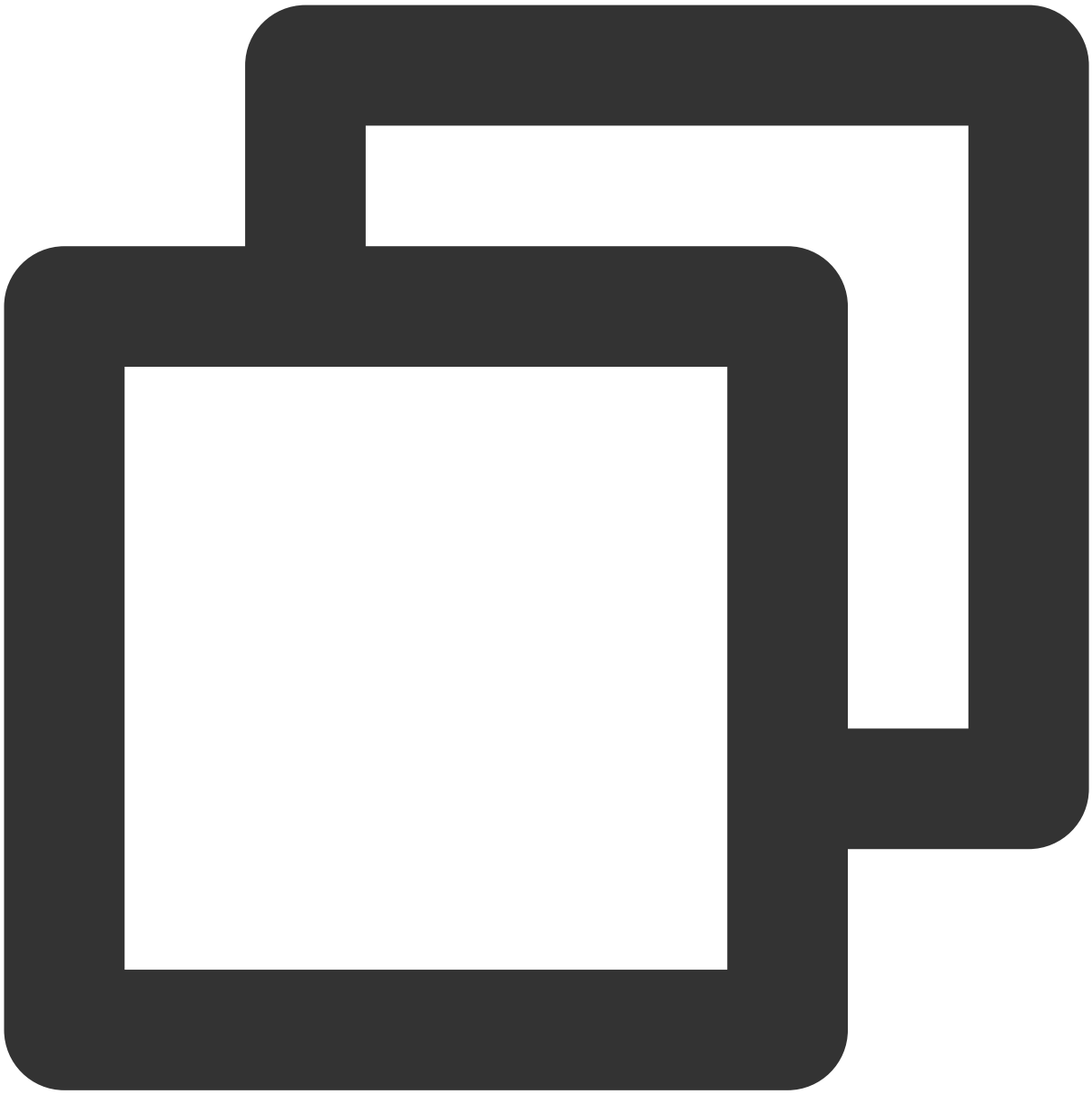
3단계: 아바타 기능 사용자 지정

`BeautyDemo/Avatar/Controller` 에서 **AvatarViewController** 코드를 참고할 수 있습니다.

설명

아바타 API에 대한 자세한 내용은 Avatar APIs를 참고하십시오.

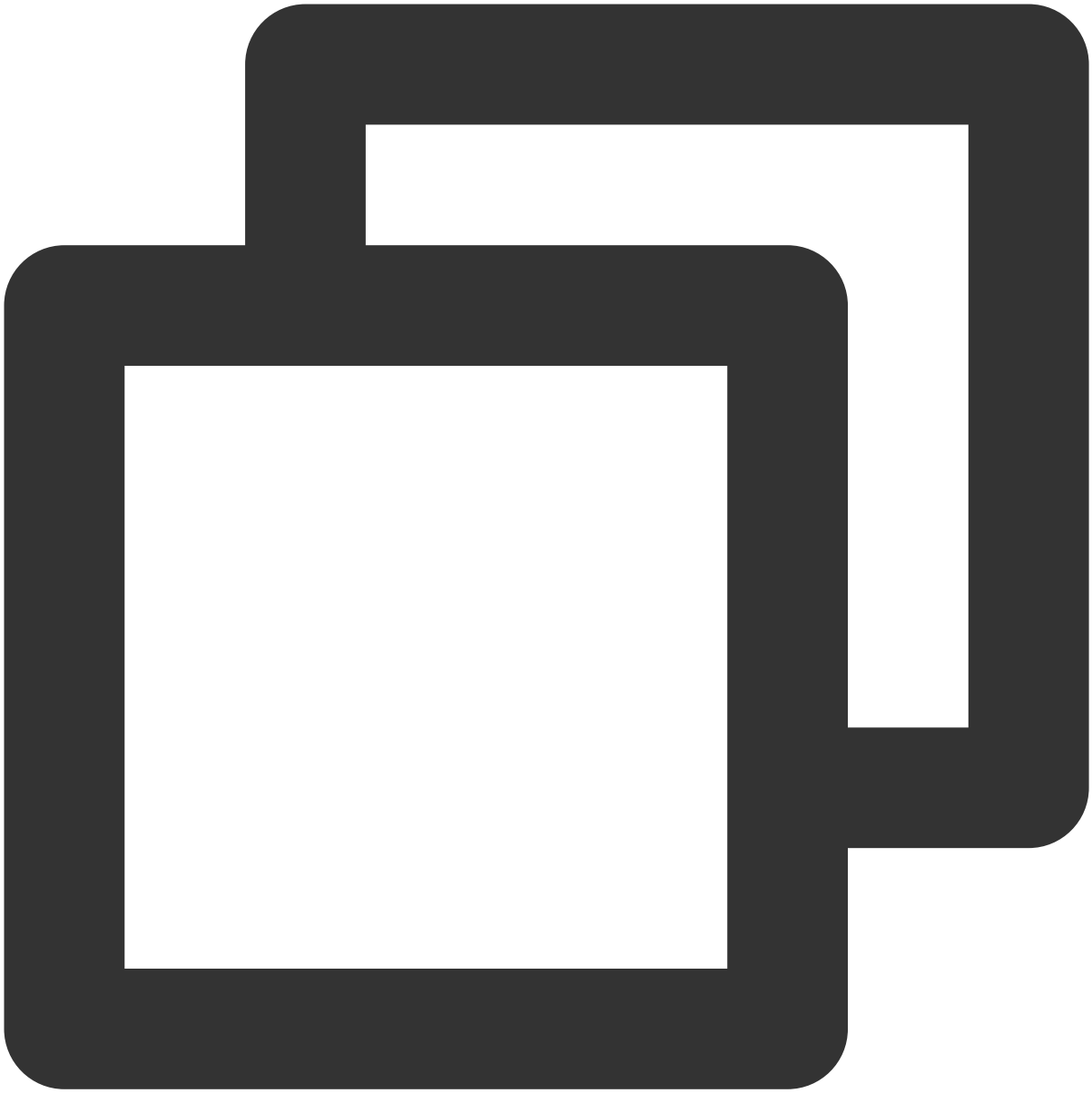
1. xmagic 객체를 생성하고 기본 Avatar 템플릿을 구성합니다.



```
- (void)buildBeautySDK {

    CGSize previewSize = CGSizeMake(kPreviewWidth, kPreviewHeight);
    NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDir
    beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_co
    NSFileManager *localFileManager=[[NSFileManager alloc] init];
    BOOL isDir = YES;
    NSDictionary * beautyConfigJson = @{};
    if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] &&
```

```
            NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beau
            NSError *jsonError;
            NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8Strin
            beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData

    }
    NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                                                                @"root_path":[[NSBundl
                                                                @"tnn_"
                                                                @"beauty_config":beaut
    };
    // Init beauty kit
    self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:asse
    // Register log
    [self.beautyKit registerSDKEventListener:self];
    [self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL

    // 아바타 소재의 경로를 전달하여 기본 아바타 로딩
    AvatarGender gender = self.genderBtn.isSelected ? AvatarGenderFemale : AvatarGe
    NSString *bundlePath = [self.resManager avatarResPath:gender];
    [self.beautyKit loadAvatar:bundlePath exportedAvatar:nil];

}
```

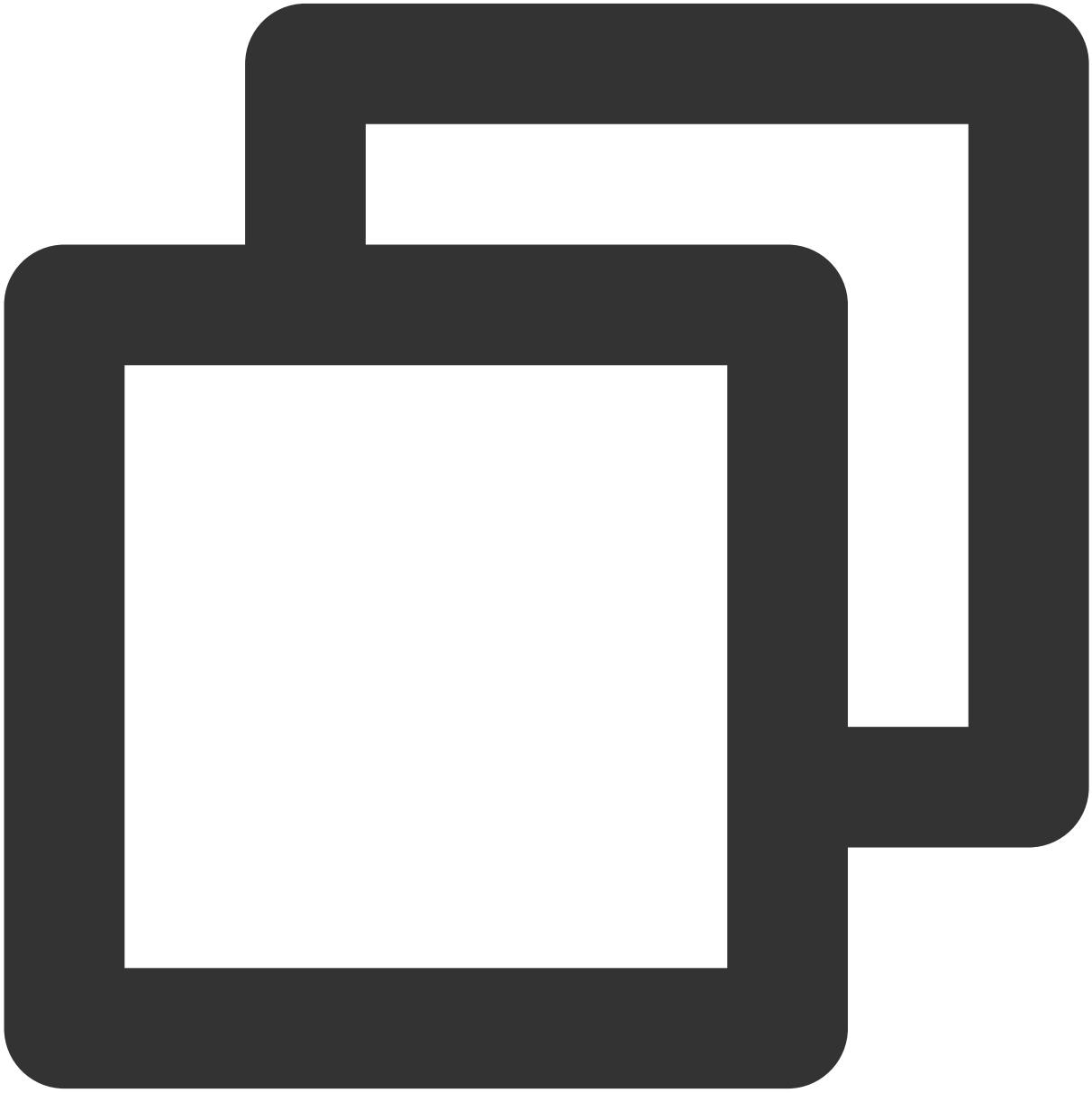
2. Avatar 소재의 소스 데이터를 가져옵니다.



```
@implementation AvatarViewController
_resManager = [[AvatarResManager alloc] init];
NSDictionary *avatarDict = self.resManager.getMaleAvatarData;
@end


@implementation AvatarResManager

- (NSDictionary *)getMaleAvatarData
{
    if (!_maleAvatarDict) {
```

```
            NSString *resDir = [self avatarResPath:AvatarGenderFemale];
            NSString *savedConfig = [self getSavedAvatarConfigs:AvatarGenderMale];
            // sdk의 API를 호출하여 소스 데이터 파싱
            _maleAvatarDict = [XMagic getAvatarConfig:resDir exportedAvatar:savedCo
    }
    return _maleAvatarDict;
}
@end
```

3. 아바타를 편집합니다.

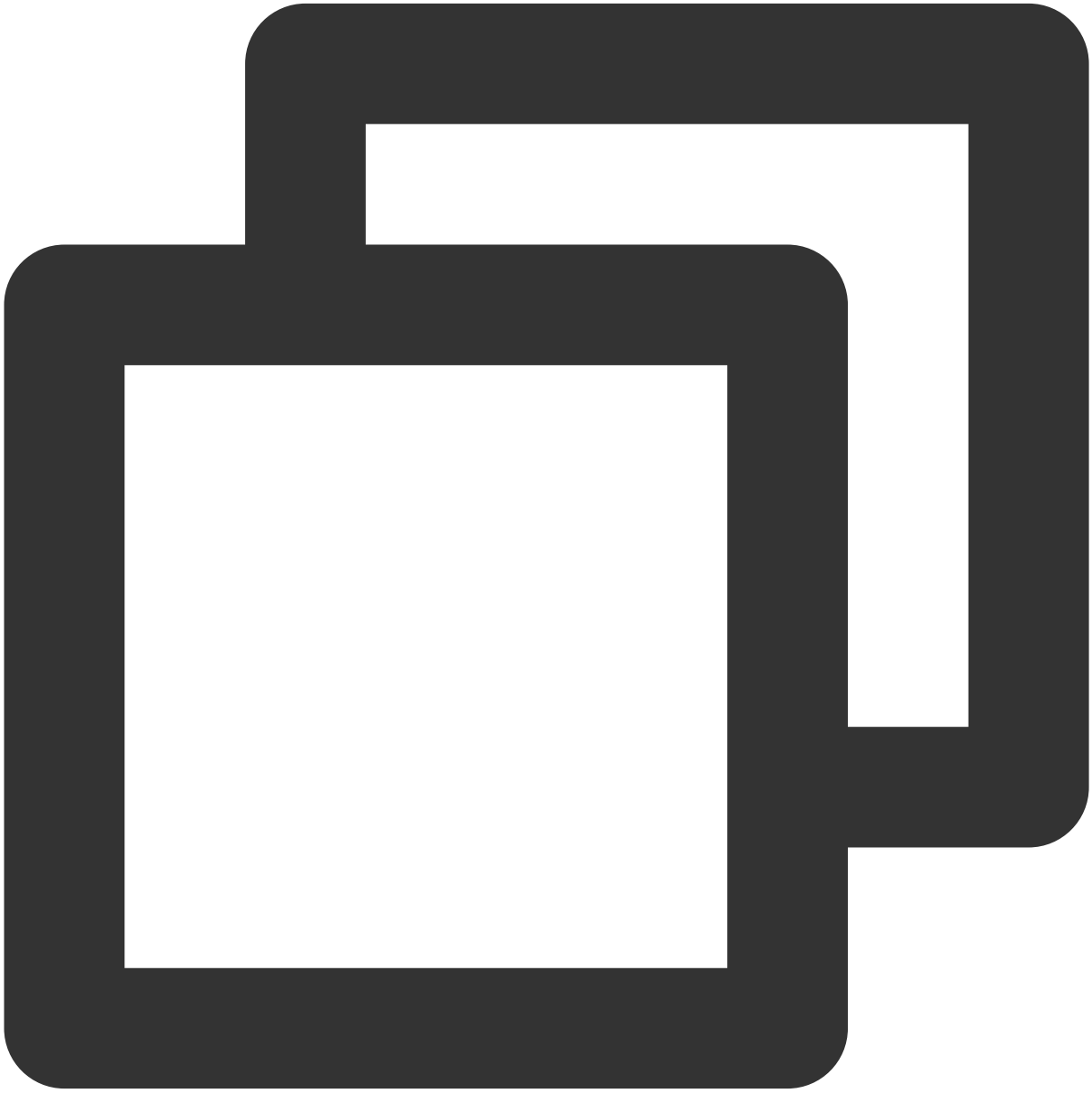


```
// sdk API로 파싱된 소재 소스 데이터에서 원하는 avatar 객체를 가져와서 sdk에 전달
```

```
NSMutableArray *avatars = [NSMutableArray array];
// avatarConfig는 getAvatarConfig:exportedAvatar: sdk API로 얻은 avatar 객체
[avatars addObject:avatarConfig];
// 아래 API를 호출하여 실시간으로 아바타 수정(얼굴 수정, 옷 입히기)
[self.beautyKit updateAvatar:avatars];
```

4. 아바타 객체를 문자열로 내보내기

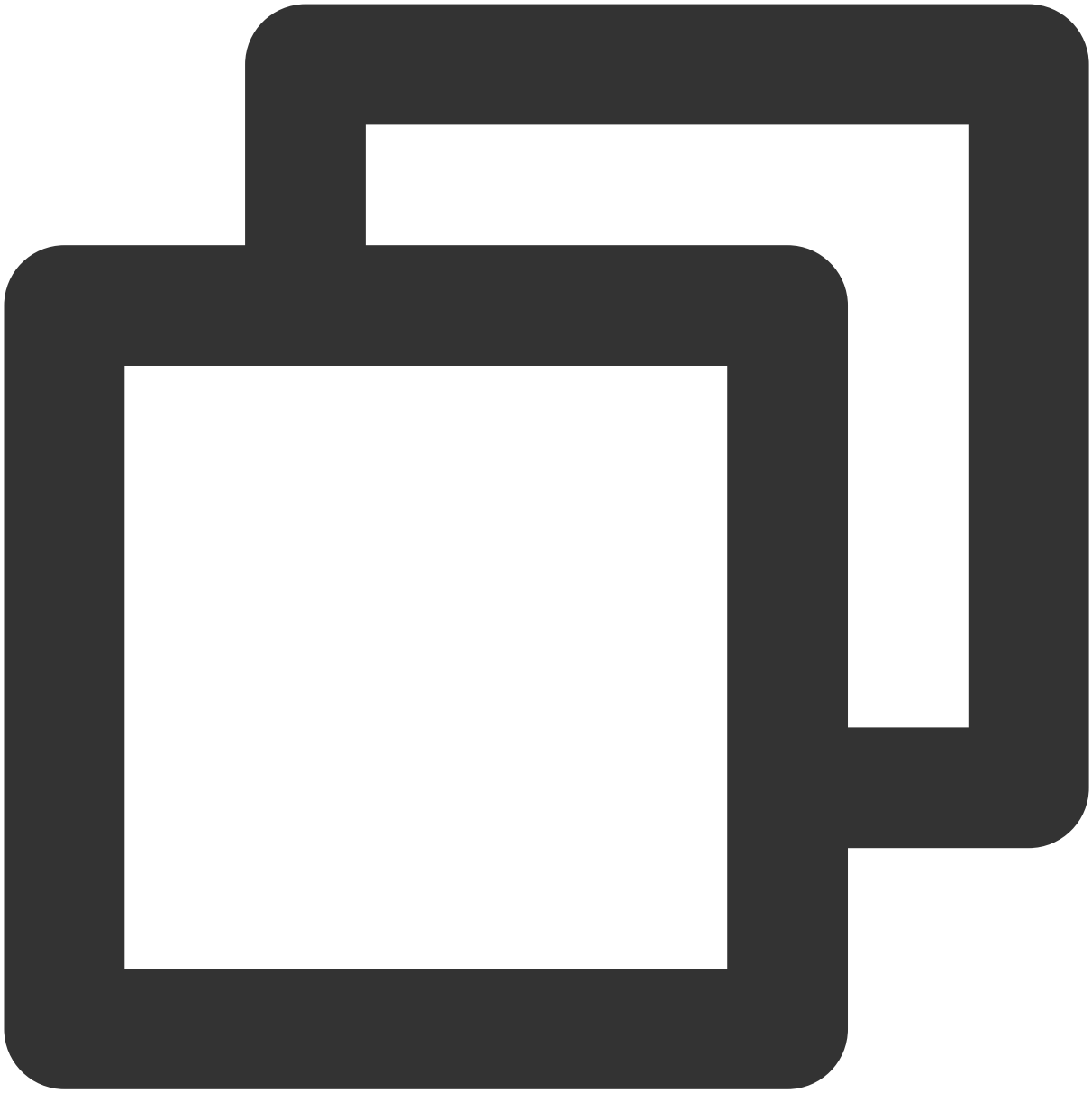
구성된 Avatar 객체를 문자열로 내보냅니다. 사용자 지정 위치에 저장할 수 있습니다.



```
- (BOOL)saveSelectedAvatarConfigs:(AvatarGender)gender
{
```

```
    NSMutableArray *avatarArr = [NSMutableArray array];
    NSDictionary *avatarDict = gender == AvatarGenderMale ? _maleAvatarDict : _fema
    // 1. 선택한 avatar 객체를 찾기 위해 순회
    for (NSArray *arr in avatarDict.allValues) {
            for (AvatarData *config in arr) {
                    if (config.type == AvatarDataTypeSelector) {
                            if (config.isSelected) {
                                    [avatarArr addObject:config];
                            }
                    } else {
                            [avatarArr addObject:config];
                    }
            }
    }
    // 2. sdk API를 호출하여 선택된 avatar 객체를 문자열로 내보내기
    NSString *savedConfig = [XMagic exportAvatar:avatarArr.copy];
    if (savedConfig.length <= 0) {
            return NO;
    }
    NSError *error;
    NSString *fileName = [self getSaveNameWithGender:gender];
    NSString *savePath = [_saveDir stringByAppendingPathComponent:fileName];
    // 디렉터리가 존재하는지 확인하고 없으면 디렉터리 생성
    BOOL isDir;
    if (![[NSFileManager defaultManager] fileExistsAtPath:_saveDir isDirectory:&isD
            [[NSFileManager defaultManager] createDirectoryAtPath:_saveDir withInte
    }
    // 3. 이후 사용을 위해 내보낸 문자열을 샌드박스에 쓰기
    [savedConfig writeToFile:savePath atomically:YES encoding:NSUTF8StringEncoding
    if (error) {
            return NO;
    }
    return YES;
}
```

5. 배경 변경



```
- (void)bgExchangeClick:(UIButton *)btn
{
    btn.selected = !btn.isSelected;
    NSDictionary *avatarDict = self.resManager.getFemaleAvatarData;
    NSArray *array = avatarDict[@"background_plane"];
    AvatarData *selConfig;
    // 배경도 avatar 객체(카테고리는 `background_plane`)이며, 배경을 바꾸는 것은 기본적으로
    for (AvatarData *config in array) {
            if ([config.Id isEqual:@"none"]) {
                    if (btn.selected) {
                            selConfig = config;
```

```
                            break;
                    }
            } else {
                    selConfig = config;
            }
    }
    [self.beautyKit updateAvatar:@[selConfig]];
}
```

Avatar SDK

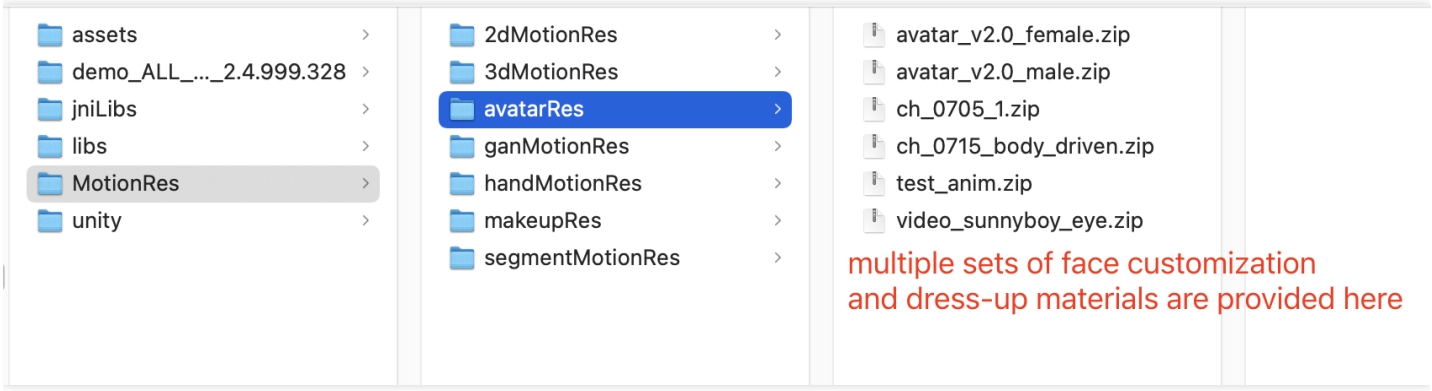
최종 업데이트 날짜: : 2022-12-22 17:44:25

SDK 통합

SDK 다운로드 및 통합 방법, 라이선스 설정 방법, Demo 프로젝트 실행 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

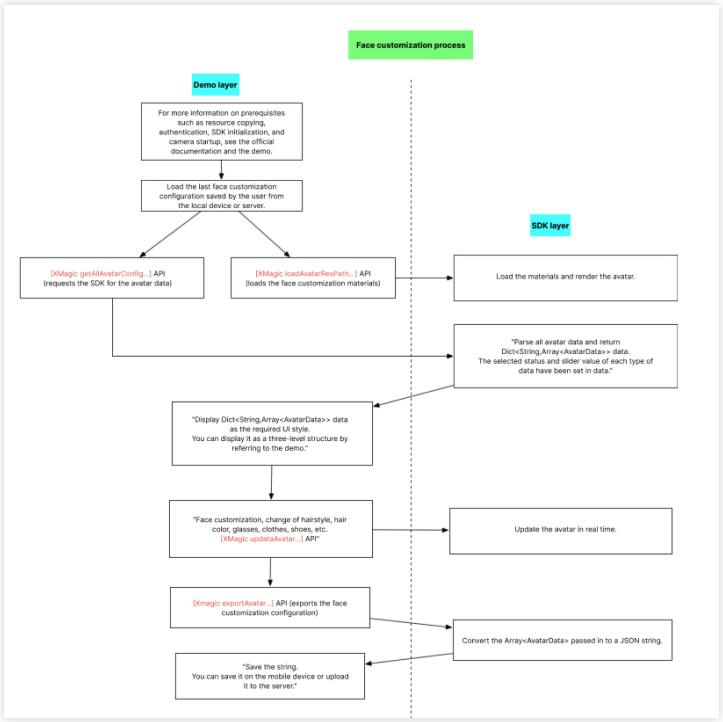
아바타 소재 준비

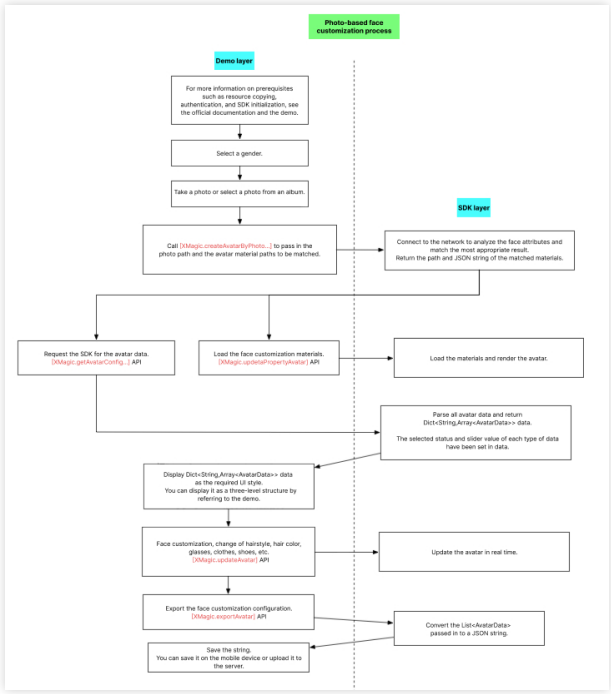
압축 해제 후 SDK 패키지의 `MotionRes/avatarRes` 디렉터리에서 찾을 수 있는 다양한 얼굴 사용자 정의 및 드레스업 자료 세트를 제공합니다. 다른 애니메이션 효과 자료와 마찬가지로 프로젝트의 assets 디렉터리에 copy해야 합니다.



아바타 DIY 및 SDK API

아바타 DIY	사진 기반 아바타 DIY





아래 섹션에서는 데이터 로딩, 아바타 DIY 및 사진 기반 아바타 DIY를 위한 API를 포함하여 `XMagicApi의 API에 대한 설명을 제공합니다.

1. Avatar 소스 데이터 가져오기(`getAvatarConfig`)

```
+ (NSDictionary <NSString *, NSArray *>* _Nullable)getAvatarConfig:(NSString * _Nullable)resPath exportedAvatar:(NSString *_Nullable)exportedAvatar;
```

- **입력 매개변수**:
  - resPath: `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` 과 같이 사용자의 모바일 장치에 있는 Avatar 소재의 절대 경로입니다.
  - exportedAvatar: 사용자가 마지막으로 아바타를 DIY했을 때부터 저장된 아바타 구성 데이터(JSON 문자열)입니다. 아바타를 처음으로 사용자 지정하거나 구성 데이터가 저장되지 않은 경우 nil을 전달하십시오.
- **출력 매개변수**:
  이 API는 NSDictionary 객체를 반환합니다. 여기서 dictionary의 key는 category(자세한 내용은 아래 TEDefine 클래스 참고)이고 value는 category의 데이터입니다. 응용 레이어가 dictionary 데이터를 가져온 후 필요에 따라 UI에 표시합니다.

2. 아바타 소재 로딩(loadAvatar)

```
- (void)loadAvatar:(NSString * _Nullable)resPath exportedAvatar:(NSString * _Nullable)exportedAvatar;
```

- **입력 매개변수**:
  - resPath: `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` 과 같이 사용자의 모바일 장치에 있는 avatar 소재의 절대 경로입니다.
  - exportedAvatar: 사용자가 마지막으로 아바타를 사용자 지정했을 때부터 저장된 아바타 구성 데이터(JSON 문자열)입니다. 아바타를 처음으로 사용자 지정하거나 구성 데이터가 저장되지 않은 경우 nil을 전달하십시오.
- **출력 매개변수**:
  이 API는 NSDictionary 객체를 반환합니다. 여기서 dictionary의 key는 category(자세한 내용은 아래 TEDefine 클래스 참고)이고 value는 category의 데이터입니다. 응용 레이어가 dictionary 데이터를 가져온 후 필요에 따라 UI에 표시합니다.

3. 얼굴 사용자 지정 및 옷 입히기(updateAvatar)

```
- (void)updateAvatar:(NSArray<AvatarData *> *_Nonnull)avatarDataList;
```

이 API가 호출되면 아바타 미리보기가 실시간으로 업데이트됩니다. 각 AvatarData 객체는 구성(예: 헤어스타일 변경)에 해당합니다. 여러 AvatarData 객체를 API 호출에 전달하여 아바타의 여러 측면을 편집할 수 있습니다(예: 헤어스타일 및 머리 색상 변경). API는 전달된 AvatarData 객체의 유효성을 검사합니다. 객체가 유효하면 SDK로 전달됩니다. 그렇지 않으면 callback이 전송됩니다.

- 예를 들어 헤어스타일을 변경하기 위해 AvatarData 객체가 전달되었지만 로컬 장치에서 헤어 모델 파일(AvatarData의 value로 지정됨)을 찾을 수 없는 경우 AvatarData 객체는 유효하지 않은 것으로 간주됩니다.
- 또한 홍채 이미지를 변경하기 위해 AvatarData 객체가 전달되었지만 로컬 장치에서 이미지 파일(AvatarData의 value로 지정됨)을 찾을 수 없는 경우 AvatarData 객체는 유효하지 않은 것으로 간주됩니다.

4. 아바타 설정 내보내기(exportAvatar)

```
+ (NSString *_Nullable)exportAvatar:(NSArray <AvatarData *>*_Nullable)avatarDataList;
```

사용자가 아바타를 편집하면 selected 값이나 AvatarData의 모양 값이 변경됩니다. 편집 후 새로운 AvatarData 목록이 생성되고 JSON 문자열로 내보낼 수 있습니다. 로컬에 저장하거나 서버에 업로드할 수 있습니다.
문자열은 다음 목적으로 내보내집니다.

- 다음 번에 XMagic의 **loadAvatar**를 호출하여 Avatar 소재를 로딩할 때 **exportedAvatar**를 JSON 문자열로 설정해야 미리 보기에서 마지막으로 아바타 설정을 기억할 수 있습니다.
- 또한 getAllAvatarData를 호출하여 selected와 Avatar 소스 데이터의 모양 값을 업데이트할 때 이 JSON 문자열을 전달해야 합니다.

5. 사진을 기반으로 아바타 DIY(createAvatarByPhoto)

이 API가 작동하려면 SDK가 인터넷에 연결되어 있어야 합니다.

```
+ (void)createAvatarByPhoto:(NSString * _Nullable)photoPath avatarResPaths:(NSArray <NSString *>* _Nullable)avatarResPaths isMale:(BOOL)isMale success:(nullable void (^)(NSString *_Nullable matchedResPath, NSString *_Nullable srcData))success failure:(nullable void (^)(NSInteger code, NSString *_Nullable msg))failure;
```

- **photoPath**: 사진 경로입니다. 얼굴이 사진 중앙에 오도록 합니다. 이상적으로는 사진에 얼굴이 하나만 포함되어야 합니다. 여러 개가 있는 경우 SDK는 임의로 하나를 선택합니다. 인식 결과를 보장하려면 사진의 짧은 면이 500px보다 길어야 합니다.

- **avatarResPaths**: 여러 세트의 Avatar 소재를 전달할 수 있으며 SDK는 사진 분석 결과에 따라 가장 적합한 세트를 선택합니다.

  > 주의 :
  > 현재 한 세트의 소재만 지원됩니다. 여러 세트가 전달되면 SDK는 첫 번째 세트를 사용합니다.

- **isMale**: 사람이 남성인지 여부입니다. 현재 이 속성은 사용되지 않습니다. 그러나 미래에 있을 수 있습니다. 올바른 값을 전달하는 것이 좋습니다.

- **success**: 성공적인 구성을 위한 콜백입니다. matchedResPath—일치하는 자료의 경로를 나타내고, srcData—일치 결과를 나타냅니다(exportAvatar의 반환 값과 동일).

- **failure**: 구성 실패 시 콜백입니다. code—오류 코드를 나타내고 msg—오류 메시지를 나타냅니다.

6. 다운로드한 구성 파일(addAvatarResource) 저장

```
+ (void)addAvatarResource:(NSString * _Nullable)rootPath category:(NSString * _Nullable)category filePath:(NSString * _Nullable)filePath completion:(nullable void (^)(NSError * _Nullable error, NSArray <AvatarData *>* _Nullable avatarList))completion;
```

Avatar 소재를 동적으로 다운로드하는 경우에 사용하는 API입니다. 예를 들어 10개의 헤어스타일을 제공하고 그 중 하나가 장치에 동적으로 다운로드된다고 가정합니다. 다운로드한 ZIP 파일의 경로를 SDK에 전달하려면 이 API를 호출해야 합니다. SDK는 파일을 파싱하여 해당 category의 폴더에 저장합니다. 다음에 getAllAvatarData를 호출하면 SDK가 새로 추가된 데이터를 반환합니다.

매개변수 설명:

- **rootPath**: `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` 과 같은 Avatar 소재의 루트 디렉터리입니다.
- **category**: 다운로드한 자료의 카테고리입니다.
- **zipFilePath**: 다운로드한 ZIP 파일의 로컬 경로입니다.
- **completion**: 결과 콜백입니다. error—오류 메시지를 나타내고 avatarList—파싱 후 얻은 avatar 데이터 배열을 나타냅니다.

7. 사용자 지정 이벤트 보내기(sendCustomEvent)

이 API는 예를 들어 얼굴이 감지되지 않을 때 유휴 상태를 표시하기 위해 사용자 지정 이벤트를 보내는 데 사용됩니다.

```
- (void)sendCustomEvent:(NSString * _Nullable)eventKey eventValue:(NSString * _Nullable)eventValue;
```

- **eventKey**: 사용자 지정 이벤트의 key입니다. 자세한 내용은 TEDefine의 AvatarCustomEventKey를 참고하십시오.
- **eventValue**: 사용자 지정 이벤트의 value로 JSON 문자열입니다. 예를 들어 `@{@"enabel" : @(YES)}`를 `json 문자열로 변환하거나 @"{\\"enable\\" : true}"` 를 직접 입력할 수 있습니다.

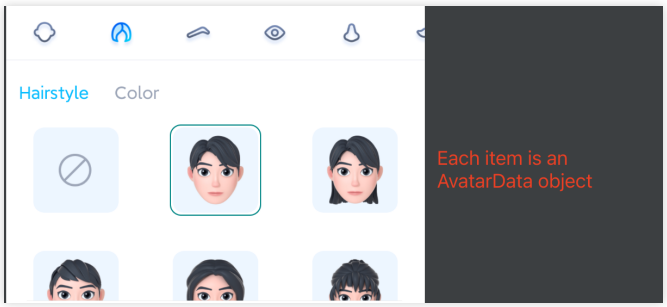
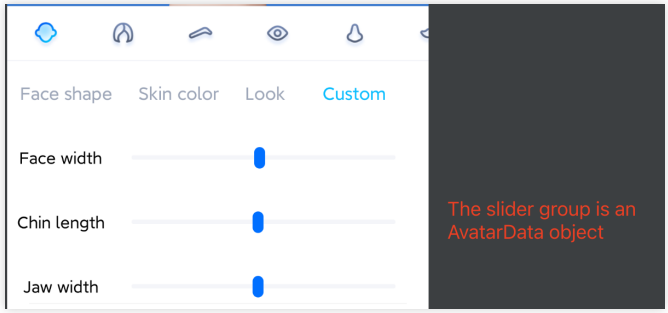
8. AvatarData 호출

AvatarData 클래스는 위 API의 핵심입니다. AvatarData에는 다음 필드가 포함됩니다.

```
/// @brief 구성 유형입니다.
@interface AvatarData : NSObject
/// 얼굴 모양 변경 또는 눈 조정 등. TEDefine에 정의된 표준 category가 요구 사항을 충족하지 못하는 경우 category를 사용자 지정할 수 있습니다(기존 category와 동일하지 않은지 확인). 이 필드는 비워 둘 수 없습니다.
@property (nonatomic, copy) NSString * _Nonnull category;
// 아바타 구성 item의 id입니다. 예를 들어 각 유형의 안경에는 고유한 id가 있습니다. 각 조정 항목도 마찬가지입니다. 이 필드는 비워 둘 수 없습니다.
@property (nonatomic, copy) NSString * _Nonnull Id;
// selector 또는 AvatarDataTypeSlider가 될 수 있는 유형입니다
@property (nonatomic, assign) AvatarDataType type;
/// 유형이 selector인 경우 이 필드는 항목이 선택되었는지 여부를 나타냅니다
@property (nonatomic, assign) BOOL isSelected;
/// 편집할 아바타의 부분. 예: 얼굴, 눈, 머리, 셔츠, 신발 등. 이를 구성하는 방법은 설명서를 참고하십시오
@property (nonatomic, copy) NSString * _Nonnull entityName;
/// entityName에서 수행할 작업(action)입니다. 자세한 내용은 설명서를 참고하십시오.
@property (nonatomic, copy) NSString * _Nonnull action;
```

```
/// entityName에서 수행할 action의 세부 정보입니다. 자세한 내용은 설명서를 참고하십시오.
@property (nonatomic, copy) NSDictionary * _Nonnull value;
@end
```

AvatarData 객체는 헤어스타일 변경이나 뺨 조정과 같은 가장 작은 구성 단위입니다.

헤어스타일 변경	뺨 조정
	

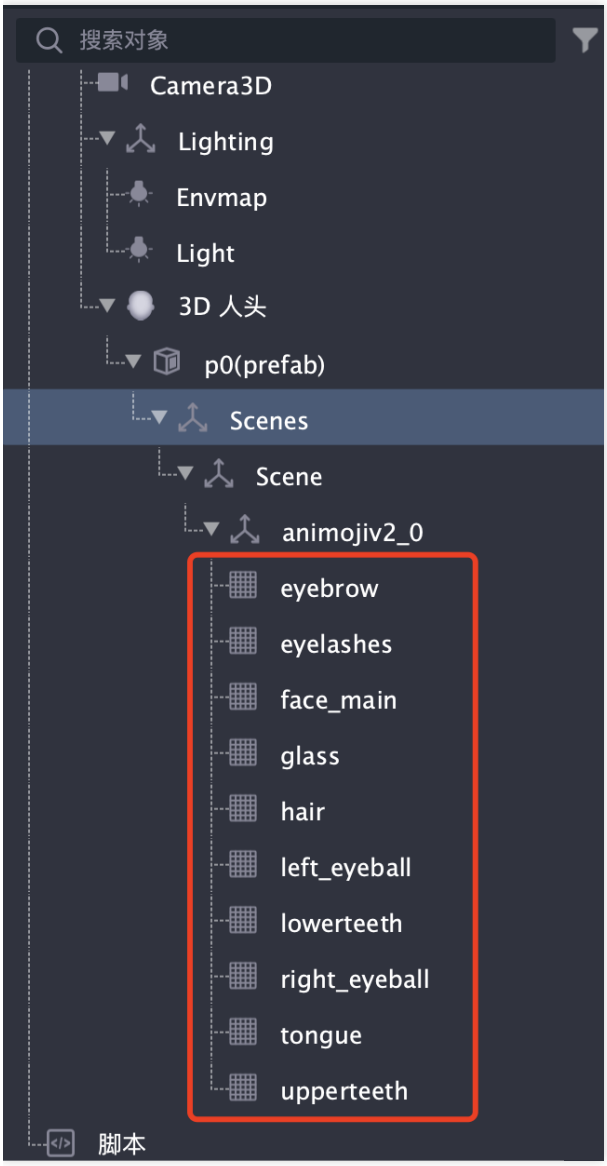
- 항목이 selector 유형인 경우 AvatarData에서 selected 값을 변경하여 구성합니다. 예를 들어 A, B, C, D의 네 종류의 안경이 있다고 가정합니다. 사용자가 A를 선택하면 A의 selected를 true로 설정하고 B, C, D의 안경을 false로 설정합니다. 사용자가 B를 선택하면 B의 selected를 true로 설정하고 A, C, D의 selected를 false로 설정합니다.
- 항목이 slider 유형인 경우 AvatarData에서 value를 변경하여 구성합니다. value 필드는 여러 key-value 쌍을 포함하는 JsonObject입니다. key-value 쌍의value를 슬라이더 값으로 설정합니다.

AvatarData에 대해 자세히 알아보기

SDK는 소재 루트 디렉터리의 custom_configs 디렉터리를 파싱하여 AvatarData를 가져와 애플리케이션 레이어으로 보냅니다. 일반적으로 AvatarData를 수동으로 구성할 필요가 없습니다.
AvatarData의 세 가지 핵심 필드는 entityName, action 및 value이며 SDK가 구성 데이터를 파싱할 때 값이 자동으로 입력됩니다. 대부분의 경우 이 세 필드의 세부 정보를 처리할 필요가 없습니다. 그러나 슬라이더 데이터의 경우 value 필드의 key-value 쌍을 파싱하고 UI에 설정된 슬라이더 값을 기반으로 구성해야 합니다.

entityName 필드

entityName은 얼굴, 눈, 머리카락, 상의 또는 신발과 같이 아바타에서 편집할 부분을 지정합니다. Demo를 예로 들면 TencentEffectStudio에서 소재 프로젝트(xxx.studio)를 열면 다음 목록이 표시됩니다.



목록의 각 항목은 아바타 부분을 나타냅니다. **항목 이름은 소재 프로젝트에서 고유해야 합니다**.

action 필드

action 필드는 entityName에서 수행할 작업(action)을 나타냅니다. 다섯 가지 action 옵션이 SDK에 정의되어 있으며 아래에 자세히 설명되어 있습니다.

action	설명	value 요구 사항
changeColor	기본 색상과 이미션(emission) 색상을 포함하는 현재 재질의 색상을 변경합니다.	value는 JsonObject여야 하며 필수 항목입니다. 자세한 내용은 아래를 참고하십시오.
changeTexture	색상 텍스처 맵, 금속/거칠기 텍스처 맵, AO(ambient occlusion) 맵, 일반 맵 및 이미션 맵을 포함하여 현재 재질의 맵을 수정합니다.	value는 JsonObject여야 하며 필수 항목입니다. 자세한 내용은 아래를 참고하십시오.

action	설명	value 요구 사항
shapeValue	blendShape 값을 수정합니다. 이 작업은 종종 얼굴 특징을 미세 조정하는 데 사용됩니다.	value는 JsonObject여야 하며(key는 도형 이름이고 value는 float임) 필수 항목입니다. 자세한 내용은 아래를 참고하십시오.
replace	예를 들어 안경, 헤어스타일 또는 옷과 같은 서브 모델을 대체합니다.	value는 3D 변환 정보, 모델 경로 및 새 서브 모델의 재질 경로를 설명하는 JsonObject여야 합니다. 서브 모델을 숨기려면 null로 설정합니다. 자세한 내용은 아래를 참고하십시오.
basicTransform	위치, 회전 및 크기 조정 설정을 조정합니다. 이 작업은 확대/축소 또는 각도를 변경하여 전체 보기와 절반 길이 보기 사이를 전환하는 데 자주 사용됩니다.	value는 JsonObject여야 하며 필수 항목입니다. 자세한 내용은 아래를 참고하십시오.

value 필드

- action은 changeColor와 동일
- action은 changeTexture와 동일
- action은 shapeValue와 동일
- action은 replace와 동일
- action은 basicTransform과 동일

설정 예시:

```
{
"key": "baseColorFactor",
"value": [43, 26, 23, 255],
"subMeshIndex":0,
"materialIndex":0
}
```

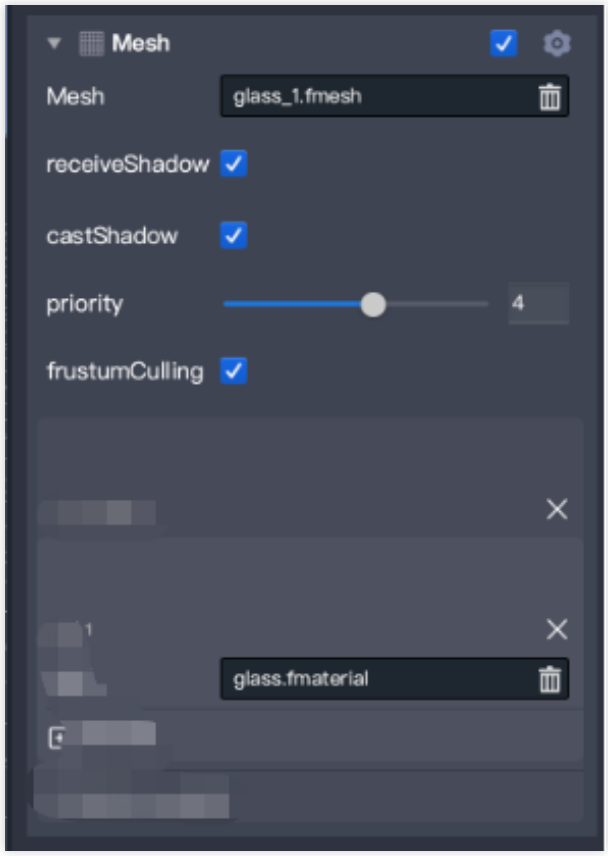
- **key**: 이 필드는 필수입니다. 현재 재질의 기본 색상, 이미션 색상 또는 사용자 지정 색상 속성을 변경할지 여부를 지정합니다.

key의 유효한 값은 현재 재질에서 사용하는 shader에 따라 다릅니다. 상기 스크린샷에서는 내장 lit_fade shader가 사용됩니다. 기본 색상의 이름은 baseColorFactor이고 이미션 색상의 이름은 emissiveFactor이므로 이 경우 baseColorFactor 및 emissiveFactor는 key의 유효한 값입니다. 재질 파일을 열어 key 값을 볼 수도 있습니다. 예를 들어 위의 예시에서 사용된 재질 파일은 `hair.fmaterial` 입니다. TEStudio 프로젝트에서 찾아 엽니다. 다음과 같이 표시됩니다.

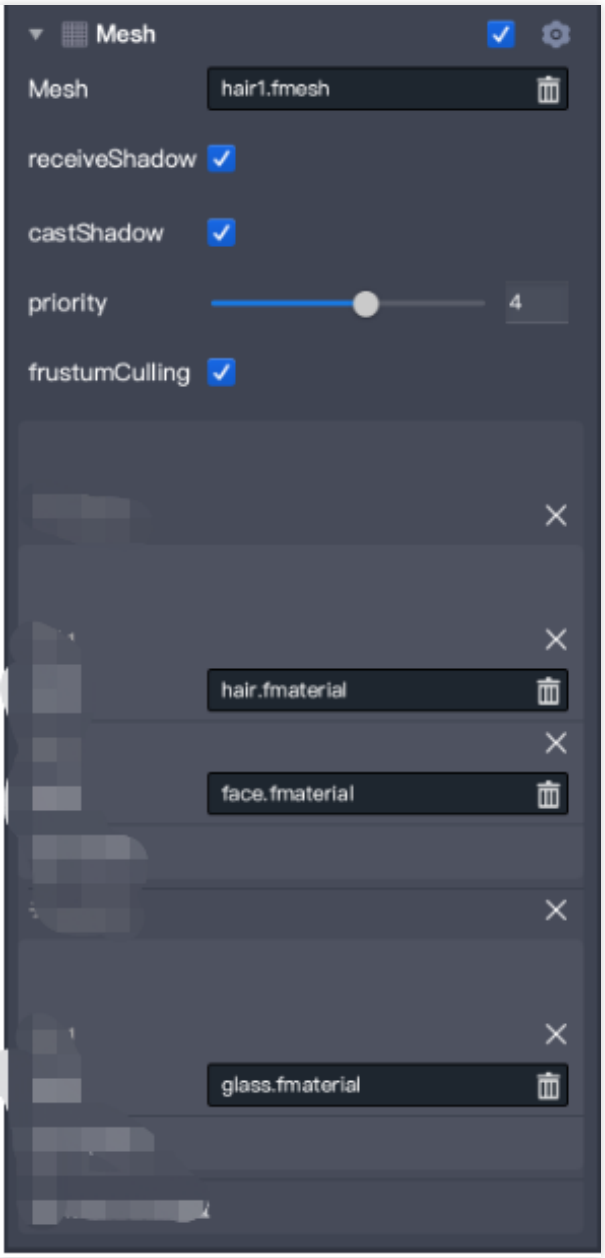
hair.fmaterial


```
{
    "renderState": {
        "colorWrite": true,
        "depthWrite": true,
        "depthCulling": true,
        "doubleSided": true,
        "cullingMode": "BACK"
    },
    "shaderResourceKey": "internal:lit_fade",
    "values": {
        "baseColorFactor": [
            0.6745098039215687,
            0.5607843137254902,
            0.36470588235294116,
            1
        ],
        "baseColorEnableTexture": false,
        "baseColorMap": "",
        "baseColorIndex": 0,
        "baseColorTexturePremultiplied": false,
        "metallicFactor": 0,
        "roughnessFactor": 0.66,
        "metallicRoughnessEnableTexture": false,
        "metallicRoughnessMap": "",
        "metallicRoughnessIndex": 0,
        "metallicSamplingChannel": 2,
        "roughnessSamplingChannel": 1,
        "aoEnableTexture": false,
        "occlusionMap": "",
        "aoIndex": 0,
        "aoSamplingChannel": 0,
        "aoStrength": 1,
        "normalEnableTexture": false,
        "normalMap": "",
        "normalIndex": 0,
        "normalScale": 1,
        "normalEnableGReverse": false,
        "emissiveFactor": [
            0,
            0,
            0,
            1
        ],
        "emissiveEnableTexture": false,
        "emissiveMap": "",
        "emissiveIndex": 0,
        "emissiveStrength": 1,
        "emissiveEnableTextureColorMultiply": true
    }
}
```

- **value**: RGBA 형식의 색상입니다. 이 필드는 필수입니다.
- **subMeshIndex 및 materialIndex**: 이 두 필드는 선택 사항입니다. 지정하지 않으면 0이 사용됩니다. TEStudio에서 Mesh(눈, 헤어스타일 또는 얼굴과 같은 서브 모델)는 일반적으로 서브 Mesh로 구성되지 않으며 하나의 재질만 있습니다.



일부 Mesh에는 서브 Mesh와 여러 재질이 있을 수도 있습니다.



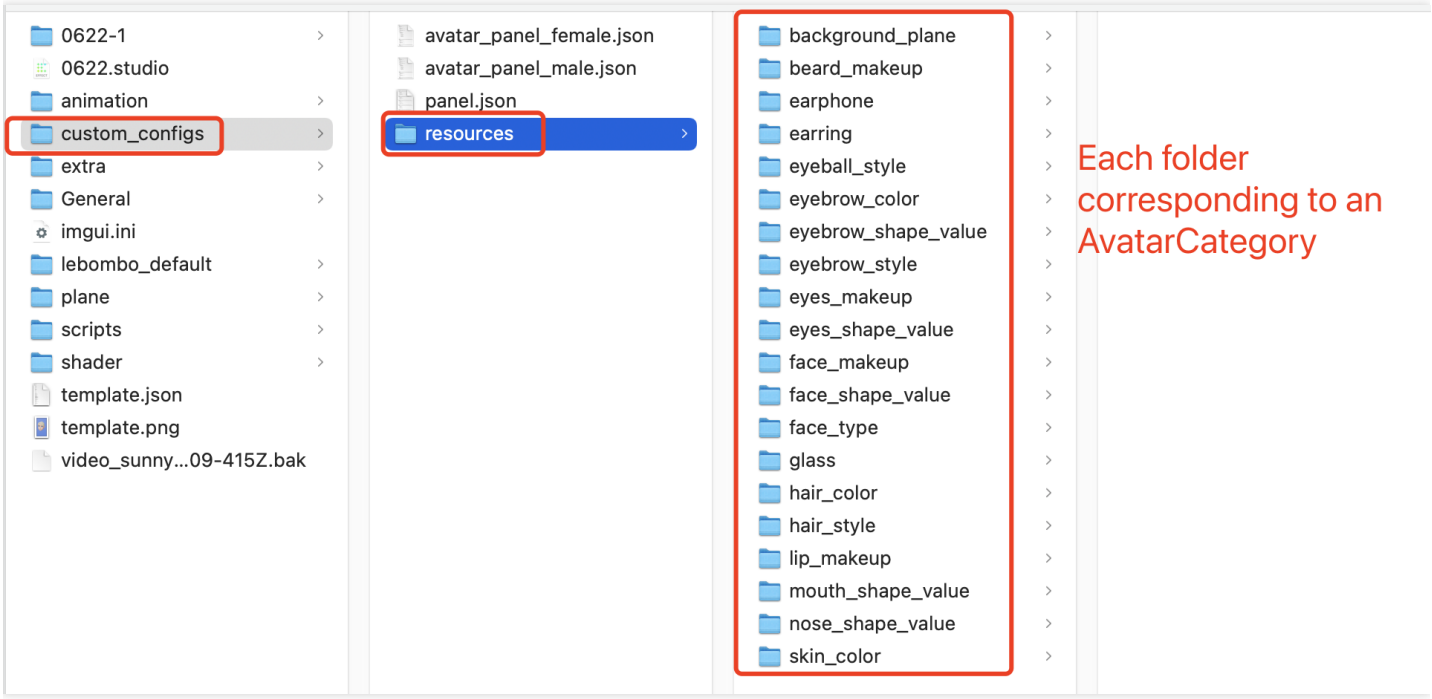
이 경우 소재 색상 변경(changeColor) 시, subMeshIndex 및 materialIndex를 사용하여 재질을 지정해야 합니다. 이 두 매개변수의 값을 결정하려면 소재 프로젝트 디렉터리에서 template.json을 열고 현재 mesh를 찾은 다음 subMeshConfigs를 찾습니다. 예를 들어 'hair'를 검색하면 다음과 같은 subMesh 설정이 표시됩니다.

```
"frustumCulling": true,
"level": 290,
"meshResourceKey": "854f65c1-75c5-4a8d-8a4b-6f3f90803b03",
"needReload": true,
"paused": false,
"priority": 4,
"receiveShadow": true,
"skinInfoResourceKey": "",
"subMeshConfigs": [
   {
      "materialResourceKeys": [
         "6224dce3-6606-4903-b5fd-3b7b7cf21cb3",
         "70fb91ad-5db9-4599-a009-b8ad530b5df5"
      ]
   },
   {
      "materialResourceKeys": [
         "64a8353e-537a-41ec-b85e-3063cdd78f3a"
      ]
   }
],
"type": "MeshRenderer3DComponentV3",
"version": 30
```


그런 다음 subMeshIndex 및 materialIndex를 지정하여 특정 재질의 색상을 변경할 수 있습니다.

Avatar 사용자 지정 데이터 구성

Avatar 구성은 resources 폴더(소재/custom_configs/resources)에 저장됩니다.



대부분의 경우 구성 파일은 자동으로 생성되며 수동으로 구성할 필요가 없습니다.
구성을 자동으로 생성하려면 TencentEffectStudio를 사용하여 재료를 디자인하고 당사에서 제공하는 resource_generator_gui app을 실행하십시오(현재 MacOS에서만 사용 가능). 자세한 내용은 Tencent Cloud 아바타 디자인 매뉴얼을 참고하십시오.

Android
Avatar 통합

최종 업데이트 날짜: : 2023-02-27 14:18:15

Avatar는 Tencent Effect SDK의 기능입니다. 사용하기 위해서는 먼저 SDK 연동 후 Avatar 소재를 추가해야 합니다. SDK를 이미 통합한 경우 첫 번째 단계를 건너뜁니다.
1. SDK를 통합합니다. 자세한 지침은 Tencent Effect SDK 통합하기를 참고하십시오.
2. 아래 순서에 따라 Avatar 소재를 로딩합니다.

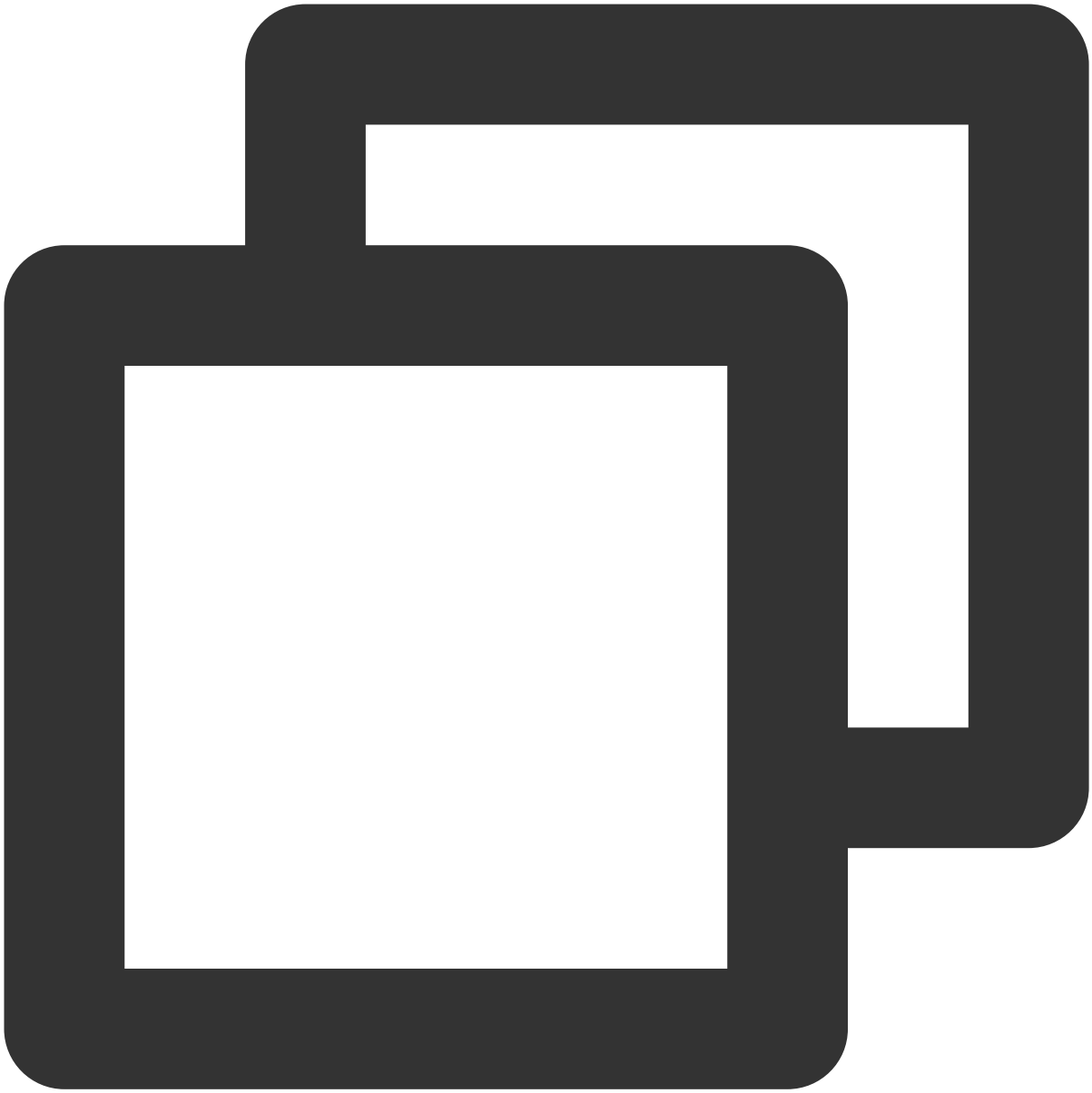
Avatar 기능 단계

1단계: Demo 파일 복사

1. 당사 웹사이트에서 demo 프로젝트를 다운로드하고 압축을 해제합니다.
2. Demo의 `demo/app/assets/MotionRes/avatarRes` 폴더를 프로젝트의 동일한 위치에 복사합니다.
3. Demo의 `com.tencent.demo.avatar` 폴더에 있는 모든 클래스를 프로젝트에 복사합니다.

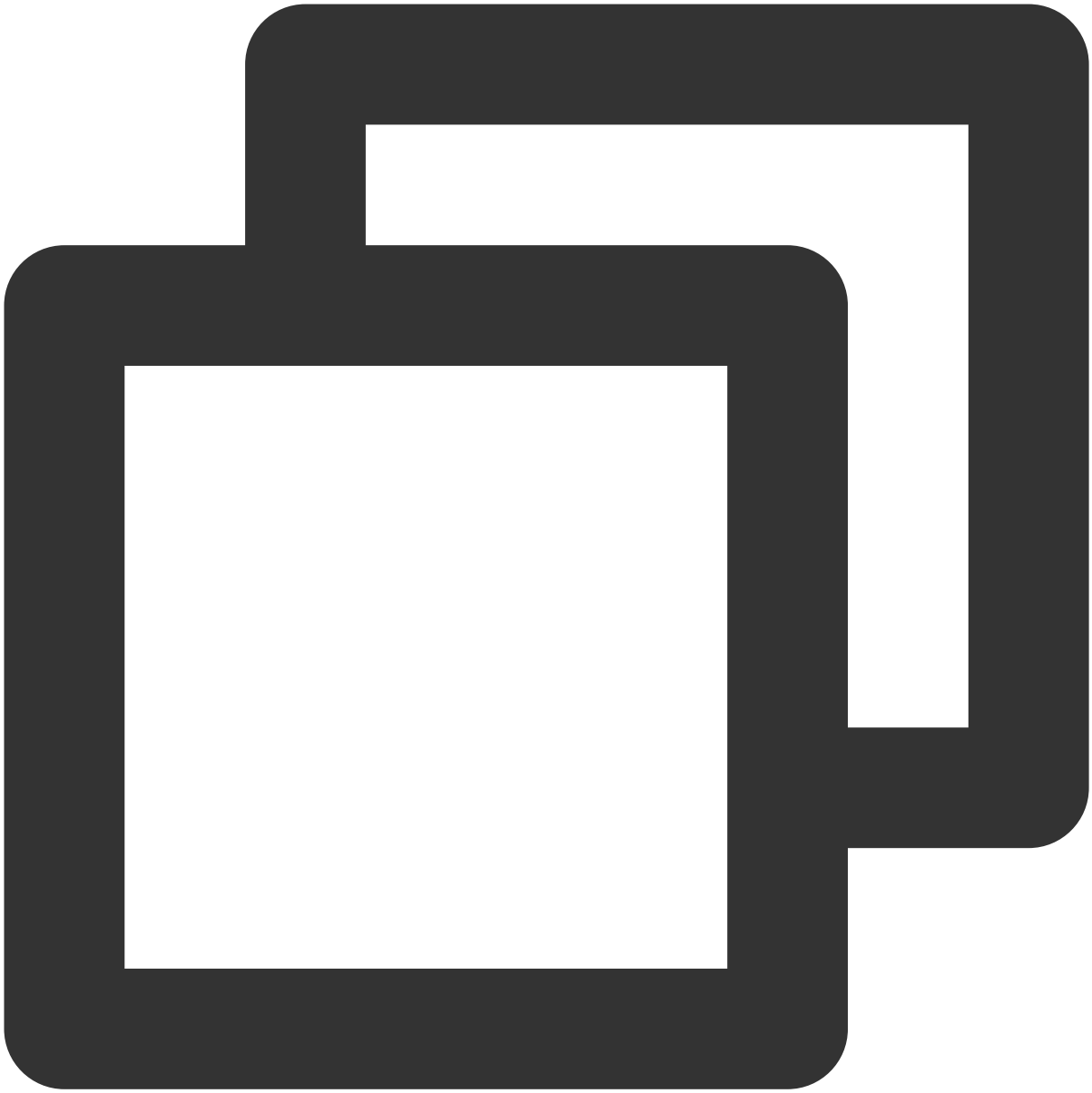
2단계: 코드 추가

다음 코드를 추가합니다(Demo의 `com.tencent.demo.avatar.AvatarActivity` 클래스 참고)
1. UI의 xml 파일에서 패널 정보를 구성합니다.



```
<com.tencent.demo.avatar.view.AvatarPanel
        android:id="@+id/avatar_panel"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        app:layout_constraintBottom_toBottomOf="parent" />
```

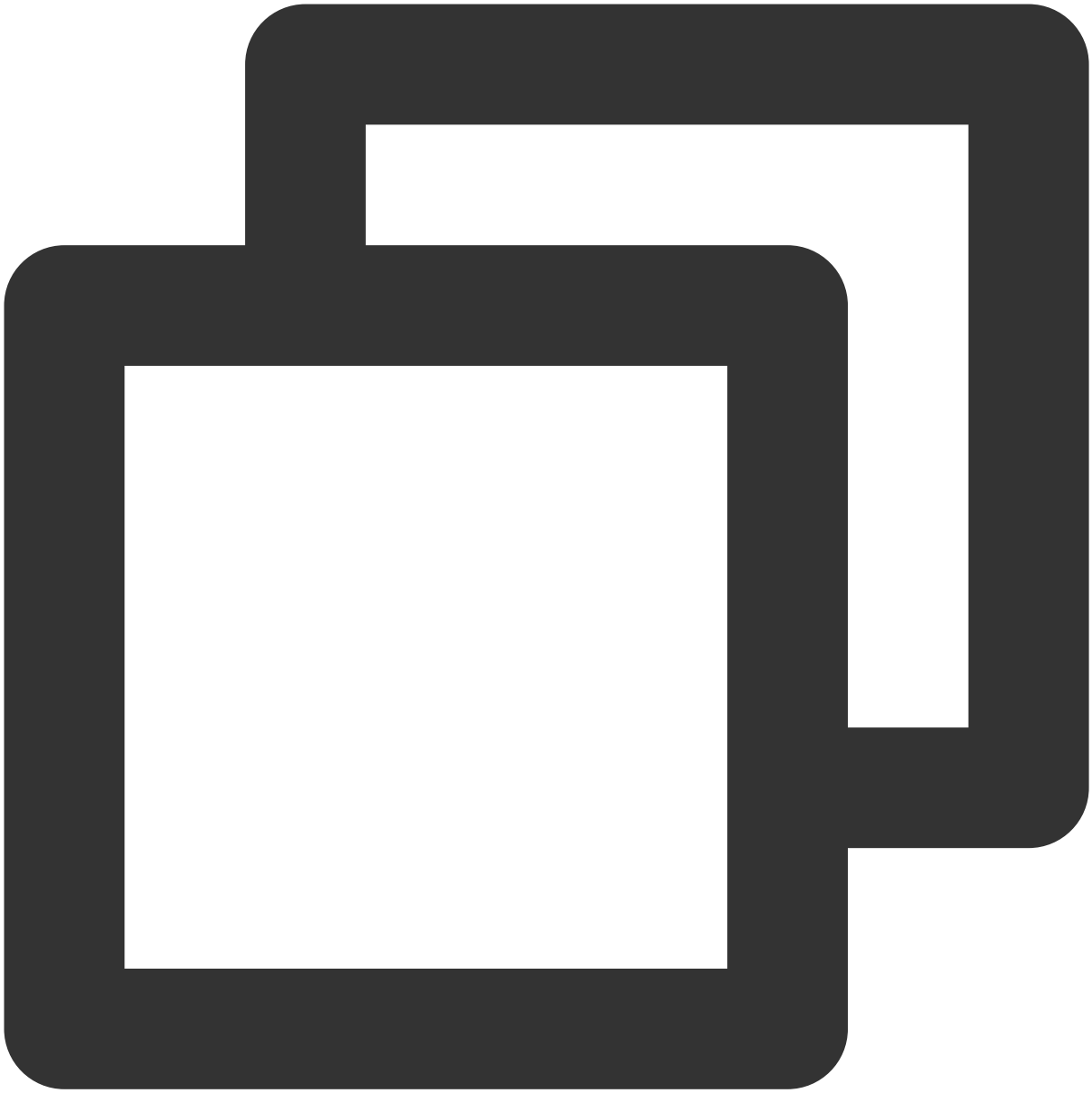
2. UI에서 패널 객체를 가져오고 콜백을 구성합니다.



```
avatarPanel.setAvatarPanelCallBack(new AvatarPanelCallBack() {
            @Override
            public void onItemChecked(MainTab mainTab, AvatarItem avatarItem) {
                if (avatarItem.avatarData == null && URLUtil.isNetworkUrl(avatarIte
                    downloadAvatarData(avatarItem, () -> updateConfig(avatarItem));
                } else {
                    updateConfig(avatarItem);
                    List<AvatarData> bindAvatarData = AvatarResManager.getAvatarDat
                    mXmagicApi.updateAvatar(bindAvatarData, AvatarActivity.this);
                }
            }
```

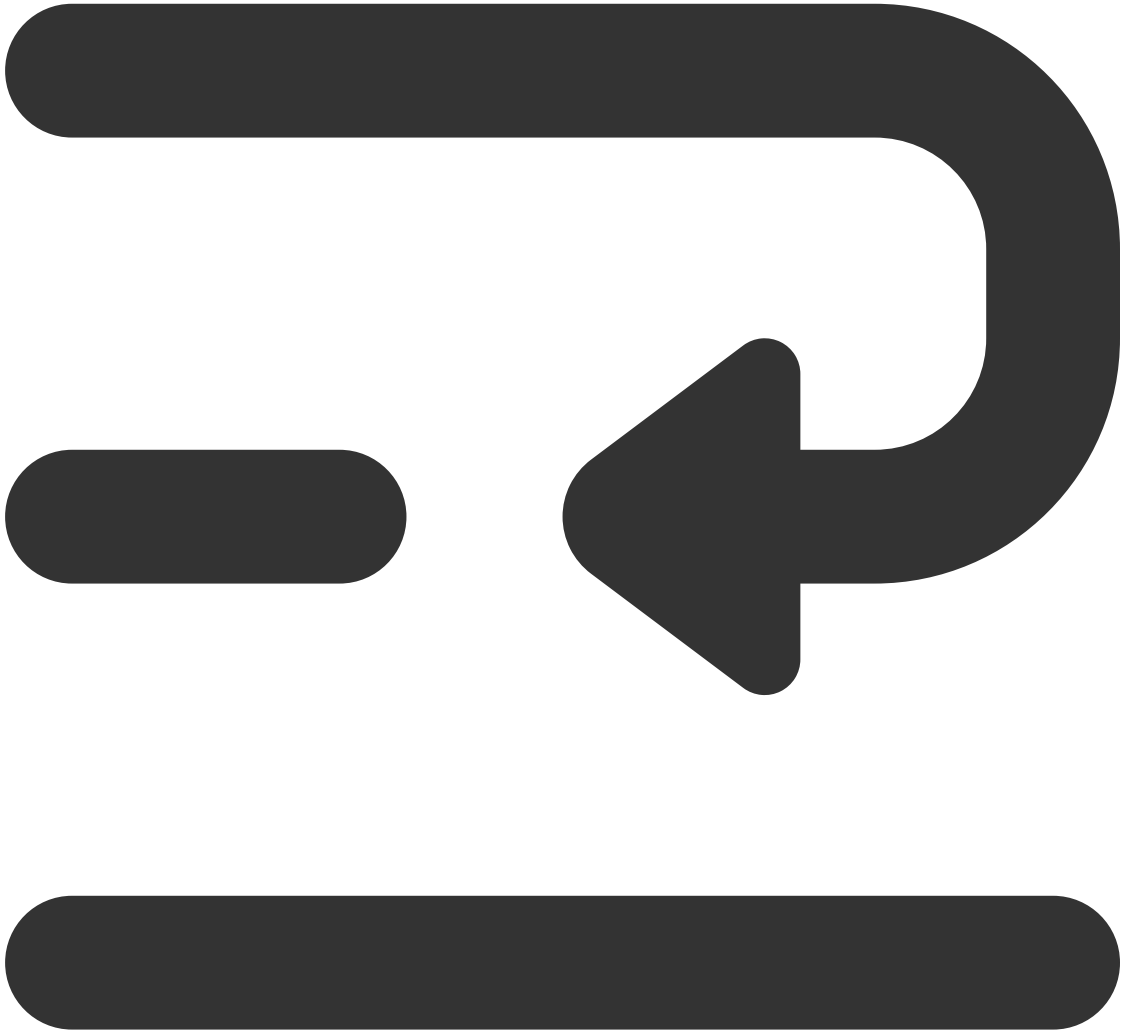
```
            @Override
            public void onItemValueChange(AvatarItem avatarItem) {
                updateConfig(avatarItem);
            }

            @Override
            public boolean onShowPage(AvatarPageInf avatarPageInf, SubTab subTab) {
                if (subTab != null && subTab.items != null && subTab.items.size() >
                    AvatarItem avatarItem = subTab.items.get(0);
                    if (avatarItem.type == AvatarData.TYPE_SLIDER && avatarItem.ava
                        downloadAvatarData(avatarItem, () -> {
                            if (avatarPageInf != null) {
                                avatarPageInf.refresh();
                            }
                        });
                        return false;
                    }
                }
                return true;
            }

            private void updateConfig(AvatarItem avatarItem) {
                if (mXmagicApi != null && avatarItem != null) {
                    List<AvatarData> avatarConfigList = new ArrayList<>();
                    avatarConfigList.add(avatarItem.avatarData);
                    mXmagicApi.updateAvatar(avatarConfigList, AvatarActivity.this);
                }
            }
        });
```

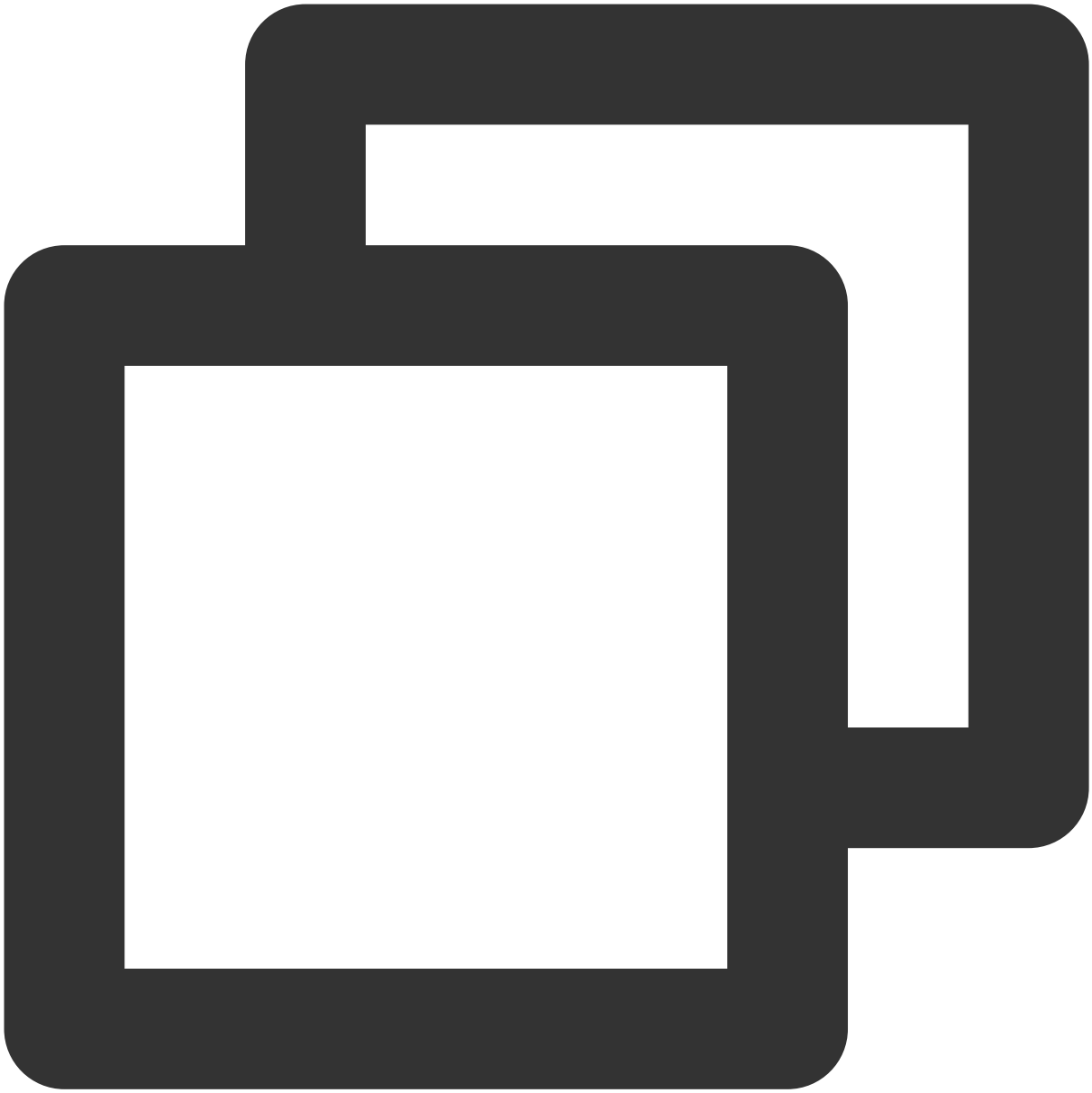
3. 패널 데이터를 가져오고 구성합니다.



```
AvatarResManager.getInstance().getAvatarData(avatarResName, getAvatarConfig(), allD
      avatarPanel.initView(allData);
 });
```

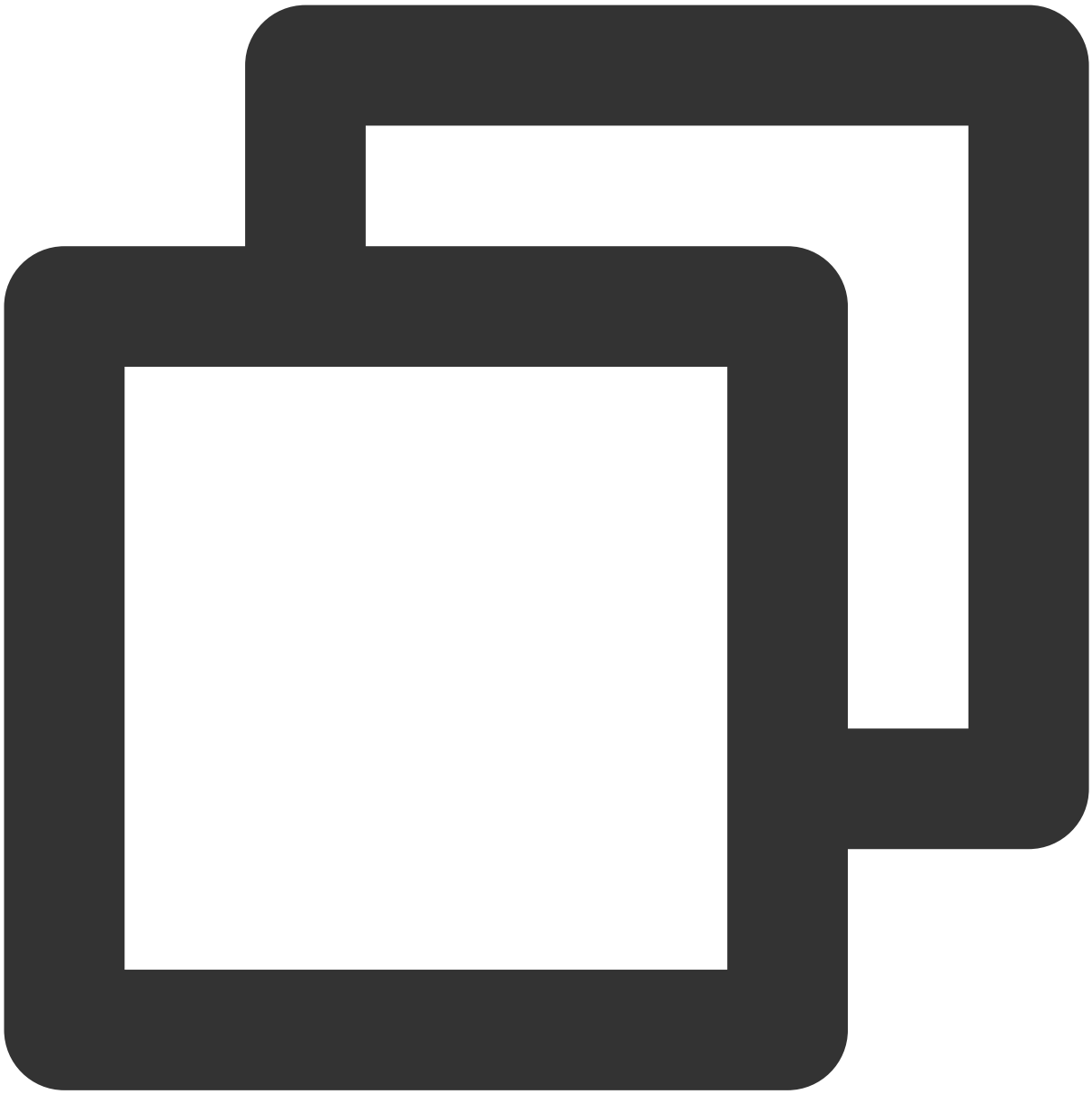
4. xmagicApi 객체를 생성하고 아바타 리소스를 로딩합니다.





```
private void initXMagic() {
    if (mXmagicApi == null) {
        mXmagicApi = XmagicApiUtils.createXmagicApi(getApplicationContext(),  n
        AvatarResManager.getInstance().loadAvatarRes(mXmagicApi, avatarResName,
        setCloseEye(true);//얼굴이 감지되지 않으면 취침/휴식 상태가 됩니다
        setAvatarPlaneType();
    } else {
        mXmagicApi.onResume();
    }
}
```

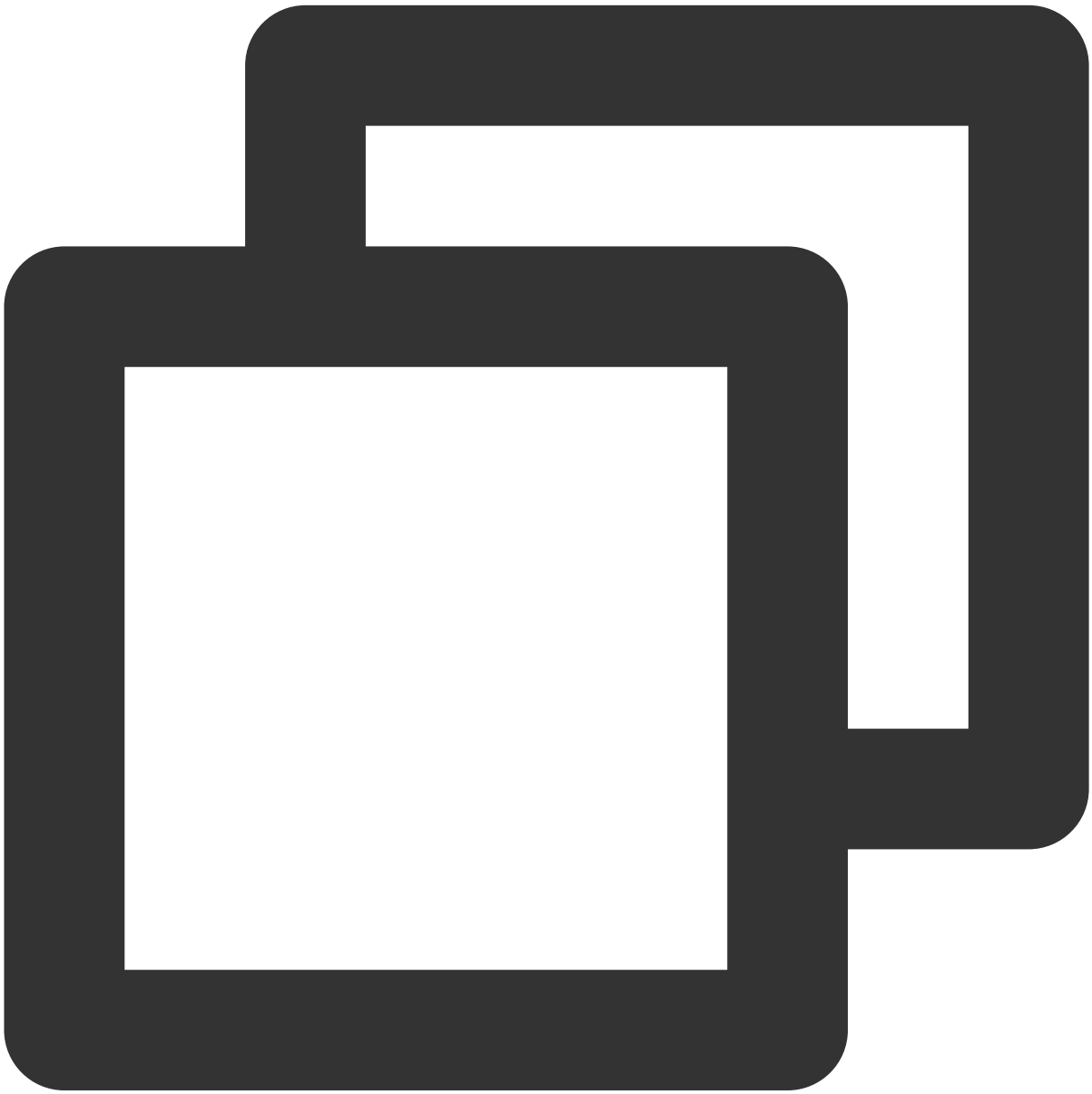
5. Avatar 속성을 저장합니다(Demo에서 saveAvatarConfigs를 참고할 수 있음)



```
/**
    * 사용자가 설정한 속성 값 또는 기본 속성 값 저장
*/
private void saveAvatarConfigs() {
        if (mXmagicApi != null && avatarPanel != null) {
                List<AvatarData> avatarDataList = AvatarResManager.getUsedAvatarDat
                new Thread(() -> {
                        String content = XmagicApi.exportAvatarConfig(avatarDataLis
                        boolean result = FileUtil.writeContentToFile(AvatarResManag
```

```
                        String tip = result ? "save avatar data successfully " : "S
                        runOnUiThread(() -> Toast.makeText(getApplicationContext(),
                }).start();
        }
}
```

6. 배경 전환(Demo에서 changeAvatarBgType 참고):



```
/**
 * 배경 전환
 */
private void changeAvatarBgType() {
```

```
        if (currentAvatarType == AvatarResManager.AvatarType.VIRTUAL_BG) {
                currentAvatarType = AvatarResManager.AvatarType.REAL_BG;
        } else {
                currentAvatarType = AvatarResManager.AvatarType.VIRTUAL_BG;
        }
        saveCurrentAvatarType();
        setAvatarPlaneType();
}

/**
 * 모델 배경 유형 설정
 */
private void setAvatarPlaneType() {
        AvatarData avatarData = AvatarResManager
                .getInstance().getAvatarPlaneTypeConfig(avatarResName, currentAvata
        if (mXmagicApi != null && avatarData != null) {
            List<AvatarData> avatarDataList = new ArrayList<>();
            avatarDataList.add(avatarData);
            mXmagicApi.updateAvatar(avatarDataList, this);
        }
 }
```

사용자 지정 Avatar UI

최종 업데이트 날짜: : 2023-02-27 14:18:15

Demo UI 설명

구현 방식

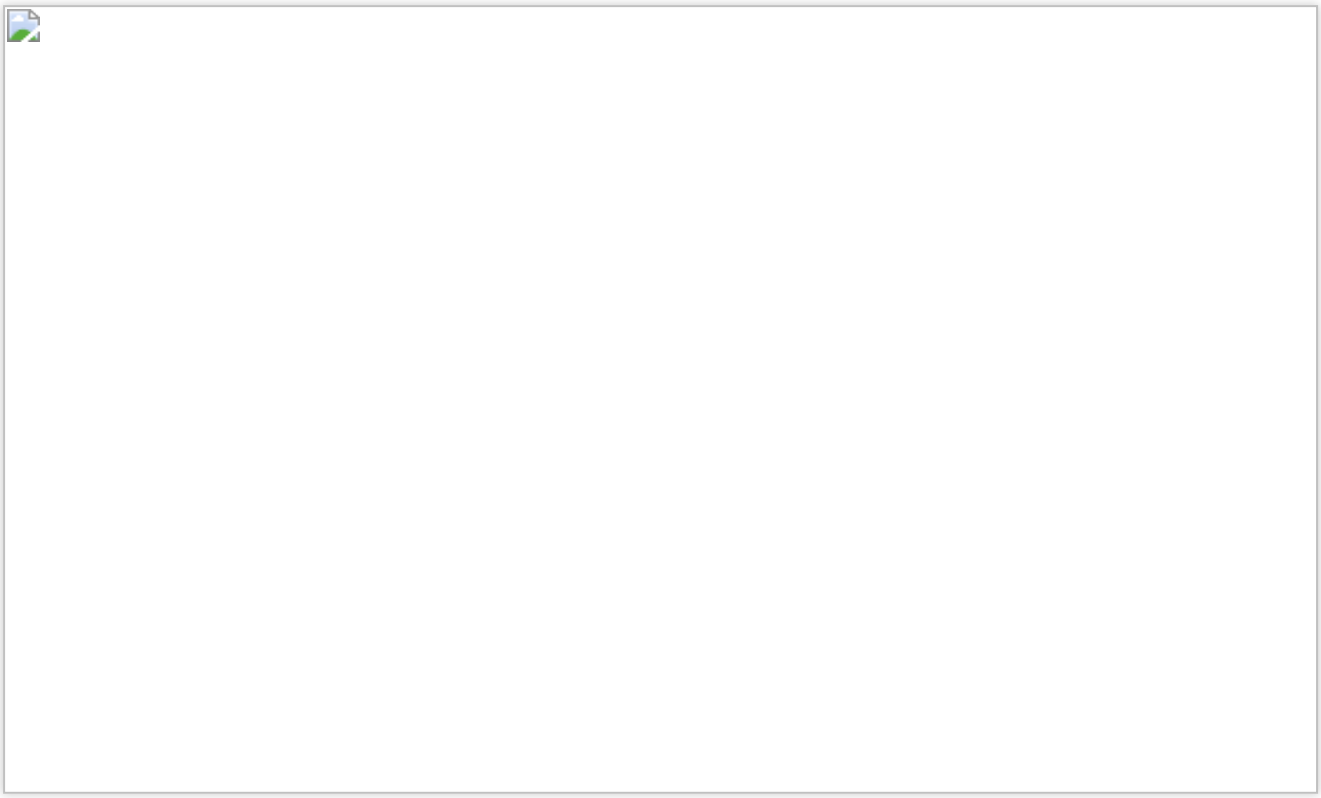
패널 구성 데이터는 어디에나 저장할 수 있습니다. Demo에서는 assets에 있습니다. 패널 파일을 처음 사용하면 설치 디렉터리에 복사됩니다.

Json 구조와 패널 요소 간의 매핑:

왼쪽 Json 파일의 item은 오른쪽의 최상위 레벨 메뉴에 해당합니다. head는 메뉴의 첫 번째 icon에 해당합니다.

왼쪽의 subTabs는 두 번째 레벨 메뉴에 해당합니다.

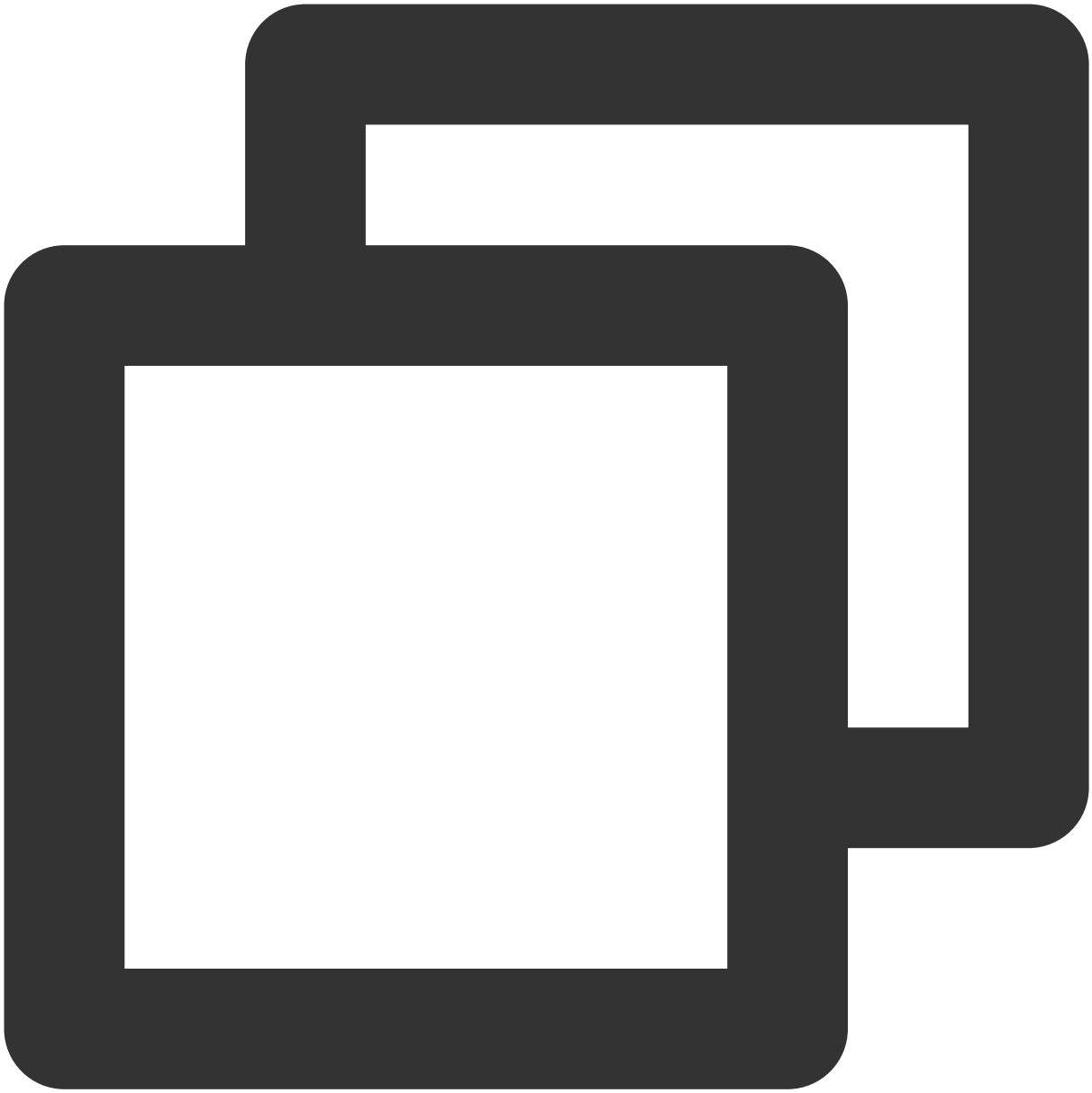
왼쪽 icon은 resources 폴더에 구성되어 있습니다. 오른쪽 파일은 패널 데이터입니다. **category 필드를 사용하여 연결**할 수 있습니다. SDK는 resources 폴더의 데이터를 파싱하고 파싱된 데이터를 map에 출력합니다. map의 key는 category 값에 해당합니다. `panel.json` 파일이 파싱된 후 SDK의 API를 사용하여 데이터를 가져오고 연결할 수 있습니다.



Demo의 주요 클래스

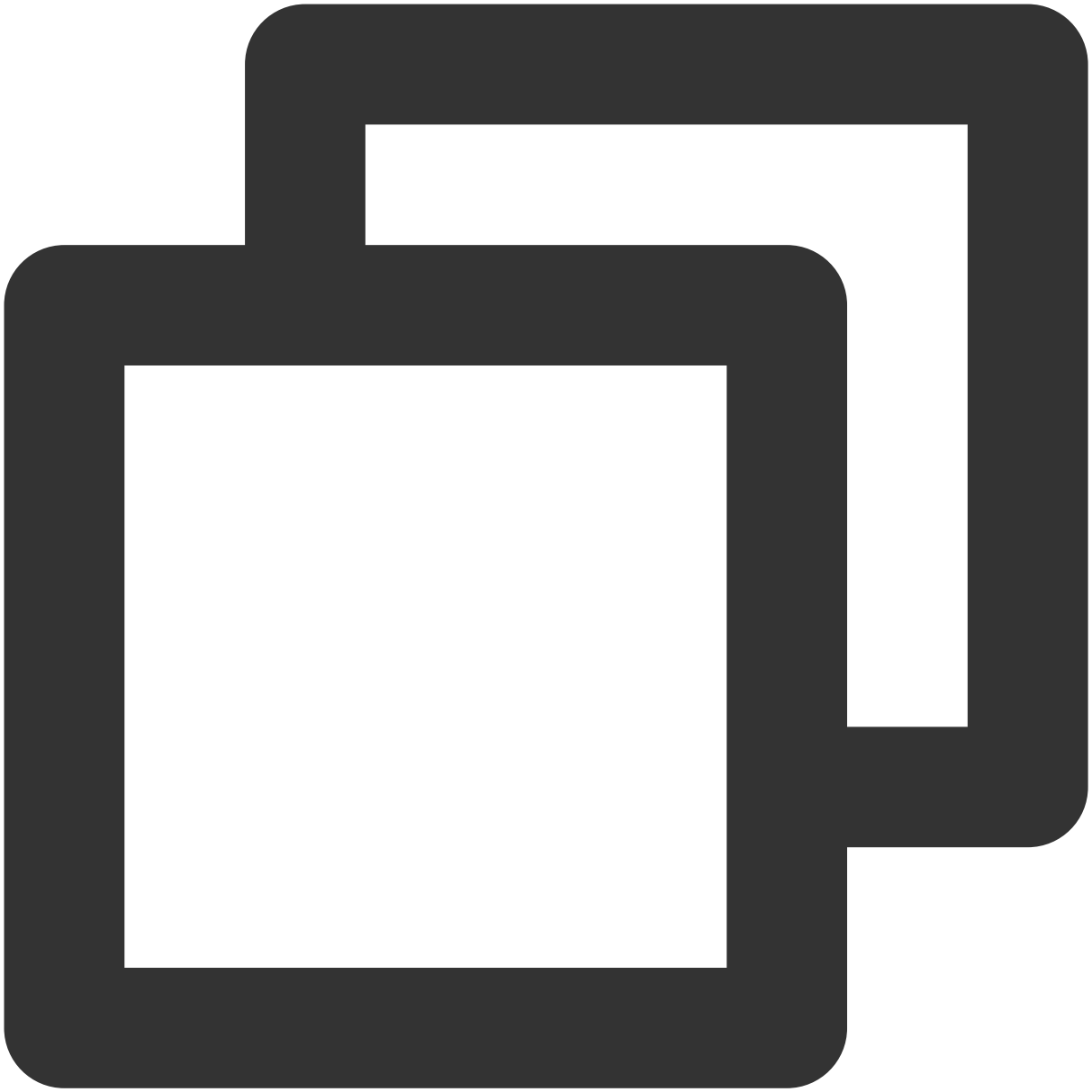
경로: `com.tencent.demo.avater.AvatarResManager.java`

1. Avatar 리소스 로딩



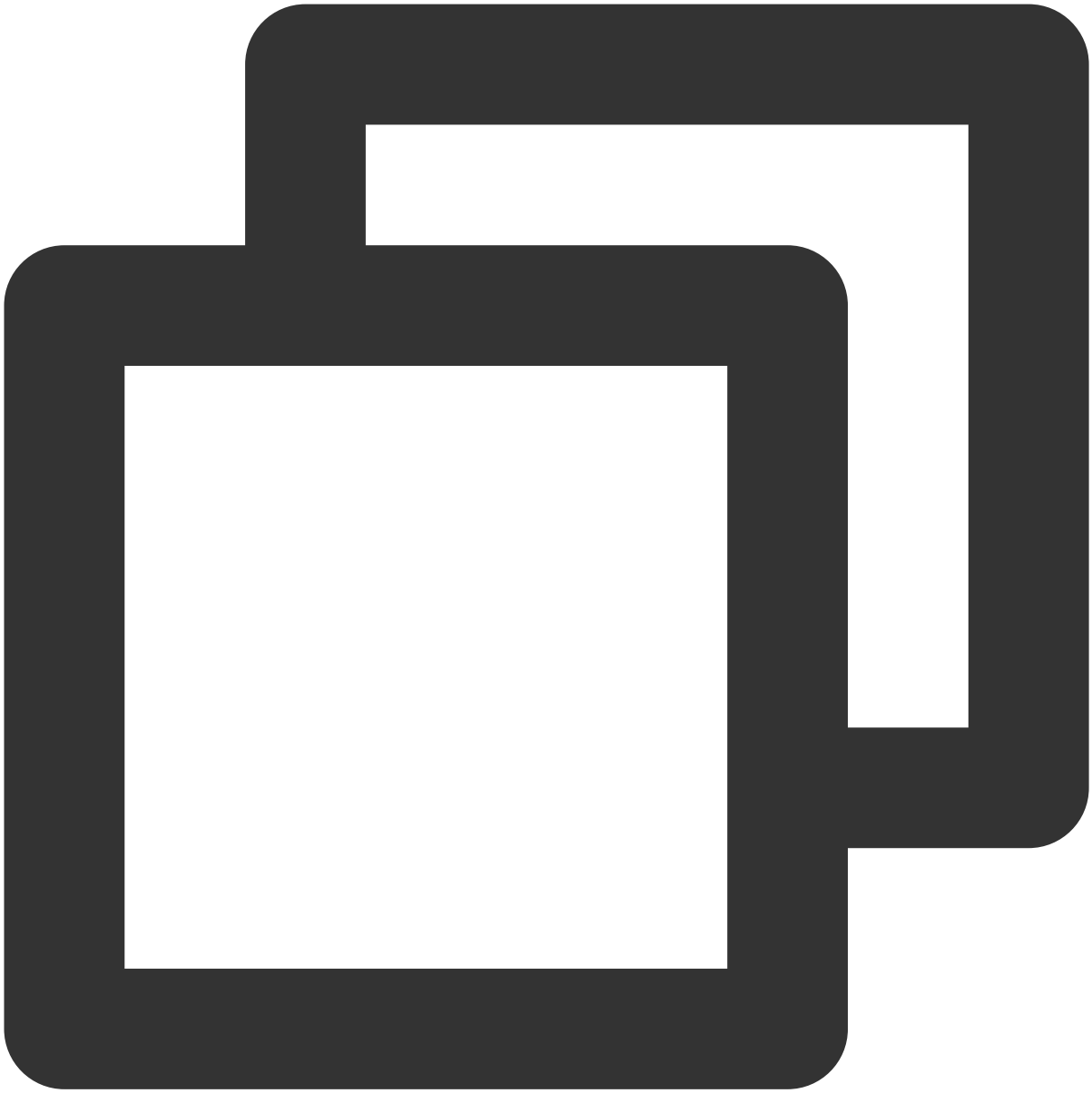
```
/**
     * Avatar 리소스를 로딩하기 위해 사용
     *
     * @param xmagicApi      XmagicApi 객체
     * @param avatarResName  이름
     * @param avatarSaveData 로딩 모델의 기본 설정, 기본 구성이 없으면 null 전달
     */
    public void loadAvatarRes(XmagicApi xmagicApi, String avatarResName, String ava
```

2. 2. 패널 데이터 가져오기



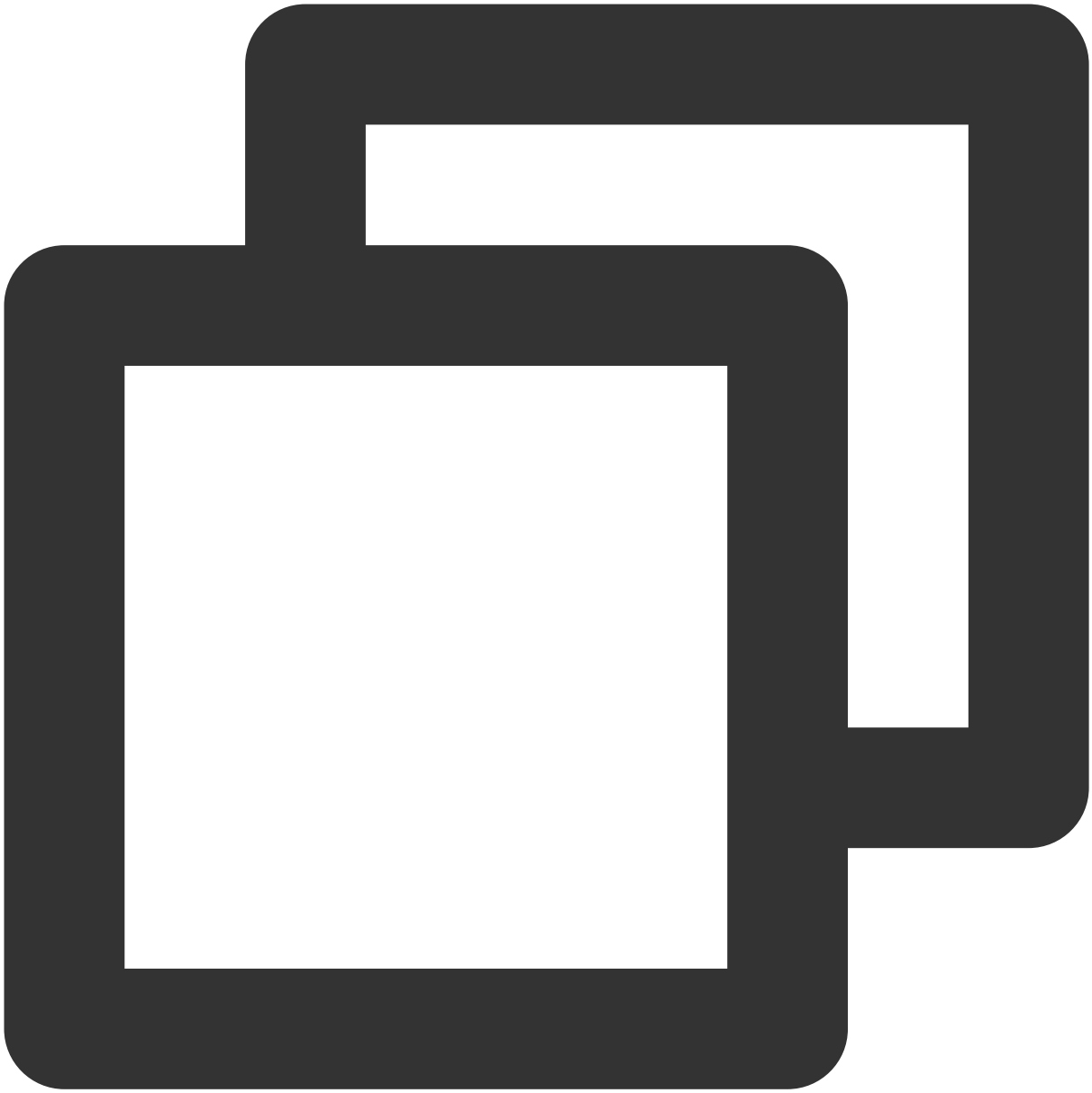
```
/**
     * avatar 패널 데이터 가져오기
     *
     * @param avatarResName      avatar 소재 이름
     * @param avatarDataCallBack API는 파일에 액세스합니다. 이 작업은 child 스레드에서 수
     *                           반환된 데이터에는 이미 resources 디렉터리의 데이터가 포함
     */
    public void getAvatarData(String avatarResName, String avatarSaveData, LoadAvat
```

3 패널 데이터에서 사용자 설정 또는 기본 설정 가져오기



```
//패널 파일에서 사용자 설정 또는 기본 설정 가져오기
public static List<AvatarData> getUsedAvatarData(List<MainTab> mainTabList)
```

4. 모델의 배경 데이터 가져오기



```
/**
     * plane Config 데이터 가져오기
     *
     * @param avatarResName 리소스 이름
     * @param avatarType    배경 유형
     * @return
     */
    public AvatarData getAvatarPlaneTypeConfig(String avatarResName, AvatarBgType a
```

부록

이 섹션에서는 MainTab.java 및 SubTab.java의 필드에 대한 설명을 제공합니다.

MainTab

필드	유형	필수 입력 여부	설명
id	String	Yes	주 메뉴 옵션의 전역적으로 고유한 식별자
label	String	No	1단계 메뉴 옵션의 이름(Demo에는 표시되지 않음
iconUrl	String	Yes	선택하지 않은 경우 아이콘의 URL
checkedIconUrl	String	Yes	선택 시 아이콘의 URL
subTabs	SubTab 목록	Yes	2단계 메뉴의 옵션

SubTab

필드	유형	필수 입력 여부	설명
label	String	Yes	2단계 메뉴 옵션의 이름
category	String	Yes	SDK의 `com.tencent.xmagic.avatar.AvatarCategory` 클래스에 정의된 2단계 메뉴 옵션의 카테고리
relatedCategory	String	No	관련 카테고리입니다. 예를 들어, 헤어 컬러는 헤어스타일 카테고리의 관련 카테고리입니다. 사용자가 헤어스타일을 변경할 때 헤어스타일 옵션의 이 필드를 현재 사용 중인 헤어 컬러(헤어 컬러 옵션의 category 값)로 설정해야 합니다. 현재 이 필드는 type이 TYPE_SELECTOR인 경우에만 적용 가능합니다.
avatarDataList	목록	Yes	AvatarIcon 목록

AvatarItem

필드	유형	필수 입력 여부	설명
id	String	Yes	SDK에서 반환된 AvatarData의 ID에 해당하는 속성 ID
icon	String	Yes	아이콘의 URL 또는 ARGB 형식의 색상 (“#FF0085CF”)

type	Int	Yes	디스플레이 유형, 유효한 값: `AvatarData.TYPE_SLIDER` (슬라이더), `AvatarData.TYPE_SELECTOR` (icon)
selected	boolean	Yes	type이 AvatarData.TYPE_SELECTOR인 경우 이 필드는 item이 선택되었는지 여부를 표시
downloadUrl	String	No	구성 파일의 동적 다운로드 주소
category	String	Yes	SubTab의 category와 동일
labels	Map<String, String>	No	type이 AvatarData.TYPE_SLIDER인 경우 이 필드는 비어 있지 않으며, 패널 왼쪽의 label
avatarData	AvatarData	Yes	SDK에 정의된 속성 연산 클래스

Avatar SDK

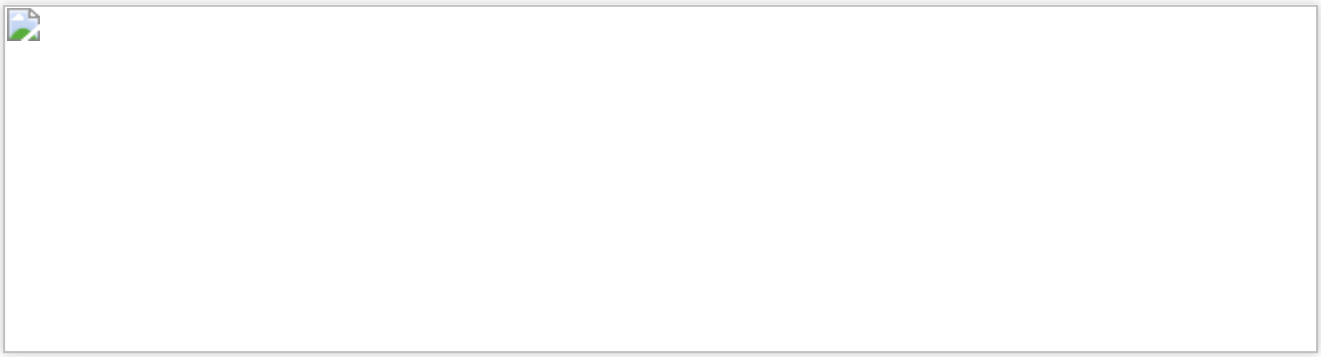
최종 업데이트 날짜: : 2023-02-27 14:18:15

SDK 통합

SDK 다운로드 및 통합 방법, 라이선스 설정 방법, Demo 프로젝트 실행 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

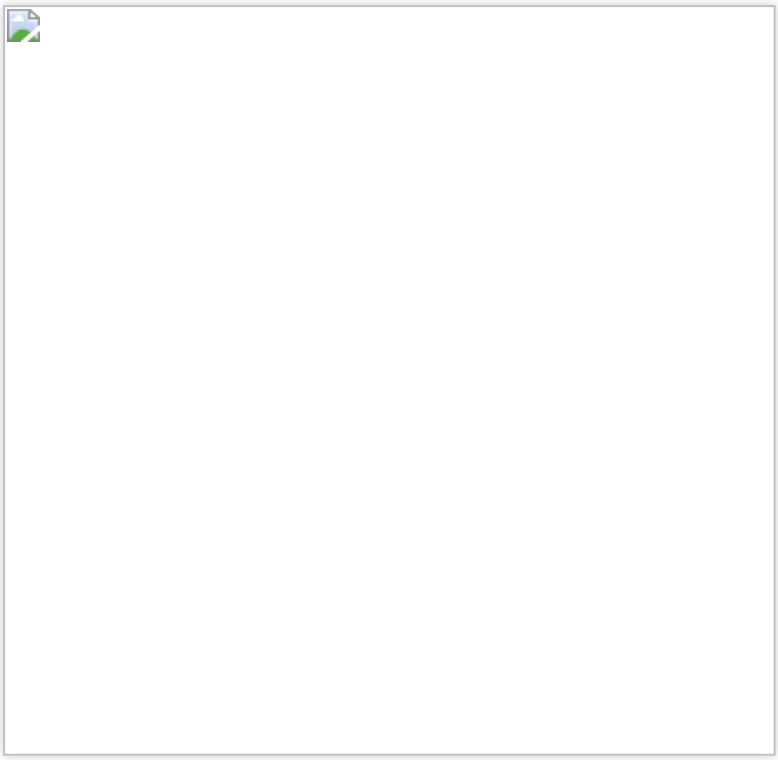
아바타 소재 복사

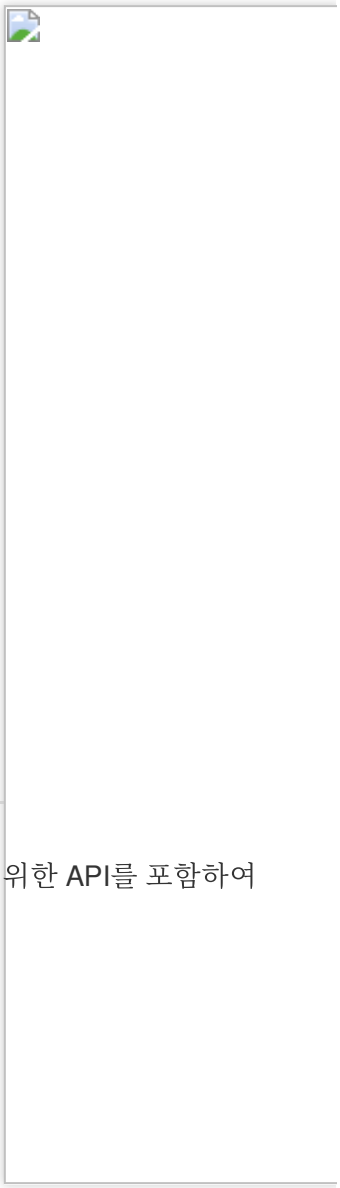
압축 해제 후 SDK 패키지의 `MotionRes/avatarRes` 디렉터리에서 찾을 수 있는 다양한 얼굴 사용자 정의 및 드레스업 자료 세트를 제공합니다. 다른 애니메이션 효과 소재와 마찬가지로 프로젝트의 assets 디렉터리에 copy해야 합니다.



아바타 커스터마이징 및 SDK API

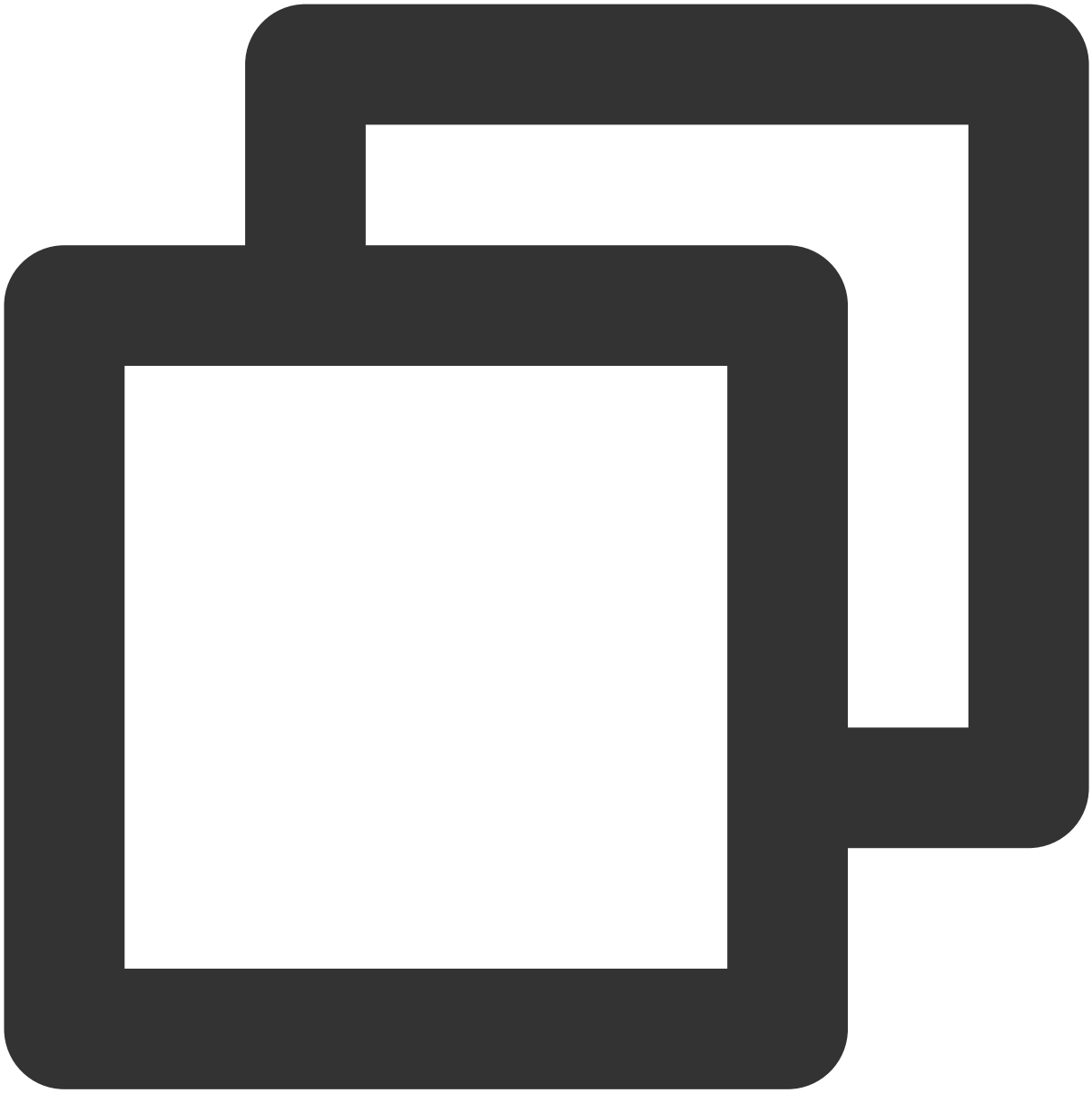
아바타 커스터마이징	사진 기반 아바타 커스터마이징





아래 섹션에서는 데이터 로딩, 아바타 커스터마이징 및 사진 기반 아바타 커스터마이징을 위한 API를 포함하여 XMagicApi의 API에 대한 설명을 제공합니다.

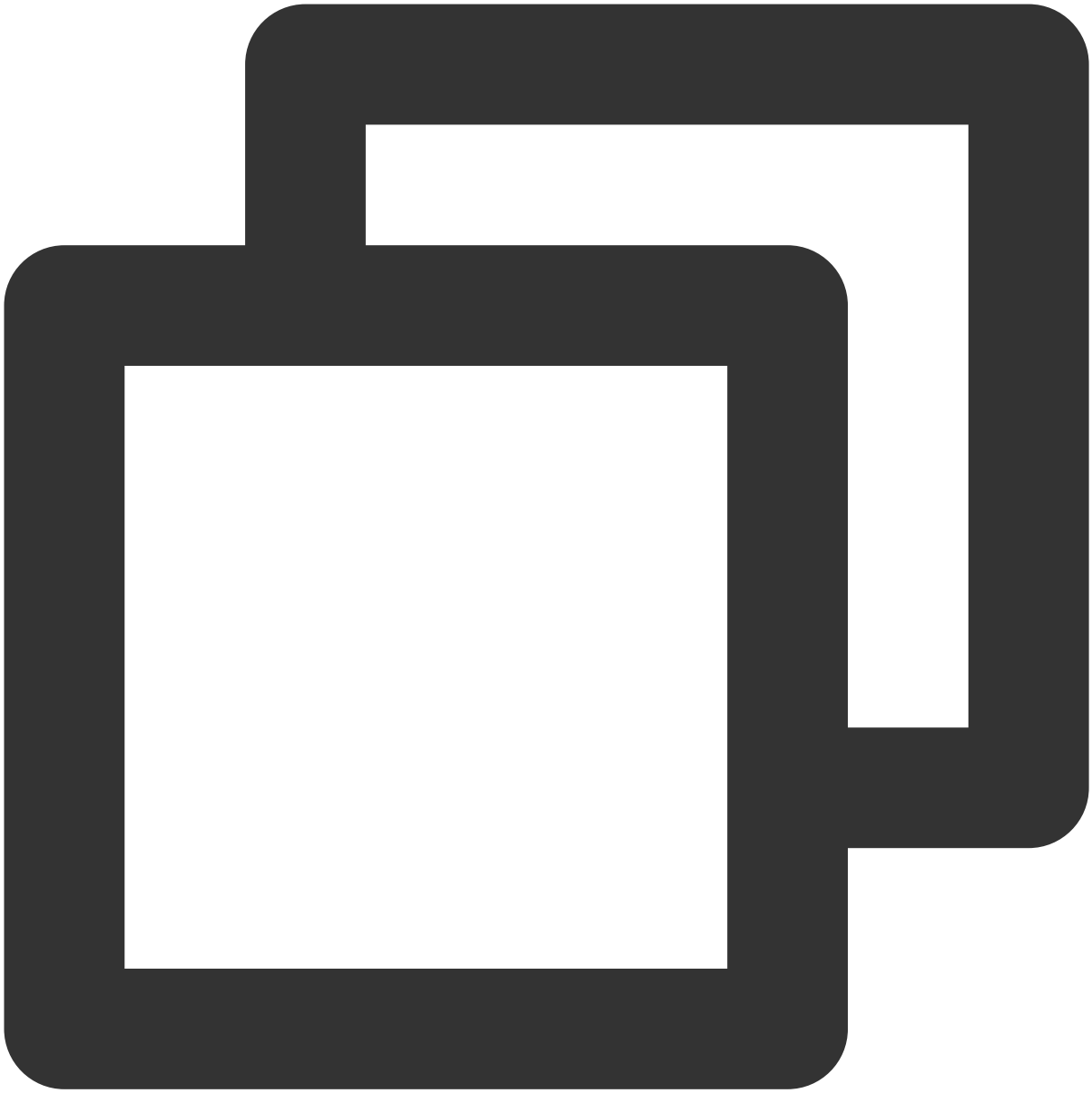
1. Avatar 소재 로딩(loadAvatar)



```
public void loadAvatar(XmagicProperty<?> property, UpdatePropertyListener updatePro
```

Avatar 소재는 다른 애니메이션 효과 소재와 같은 방식으로 로딩됩니다. loadAvatar API는 updateProperty API와 동일합니다.

2. Avatar 소스 데이터 로딩(getAvatarConfig)



```
public static Map<String,List<AvatarData>> getAvatarConfig(String avatarResPath, St
```

입력 매개변수:

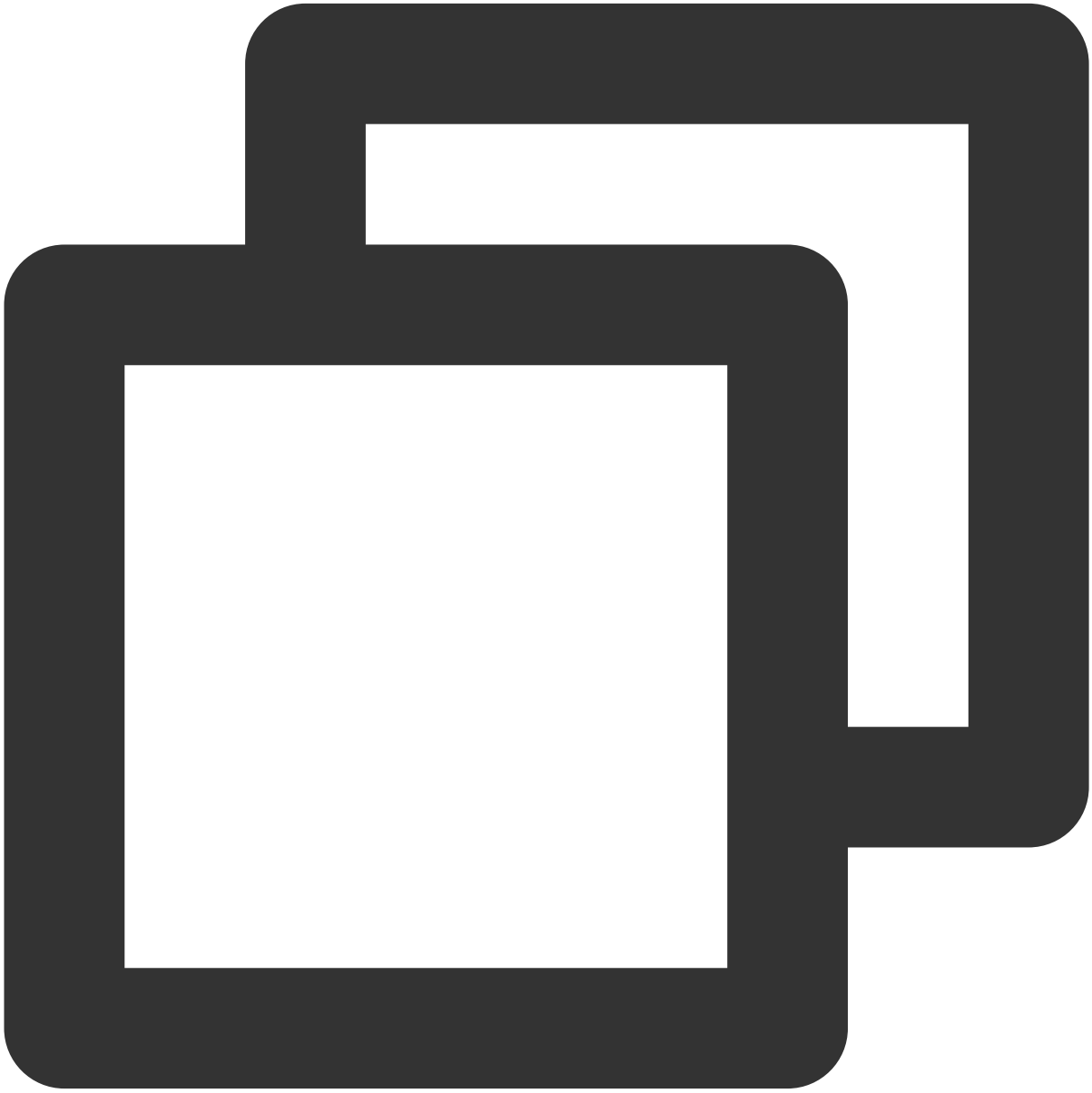
avatarResPath:

`/data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/animoji_0624` 와 같이 사용자 모바일 장치에 있는 avatar 소재의 절대 경로입니다.

savedAvatarConfigs: 사용자가 마지막으로 아바타를 커스터마이징한 시점부터 저장된 아바타 구성 데이터(JSON 문자열)입니다. 아바타를 처음으로 커스터마이징하거나 구성 데이터가 저장되지 않은 경우 null을 전달합니다.

출력 매개변수:
이 API는 key가 category에 해당하고 value가 category의 데이터에 해당하는 map을 반환합니다. 자세한 내용은 AvatarCategory 클래스를 참고하십시오. 애플리케이션 레이어가 map을 가져온 후 필요에 따라 UI에 데이터를 표시합니다.

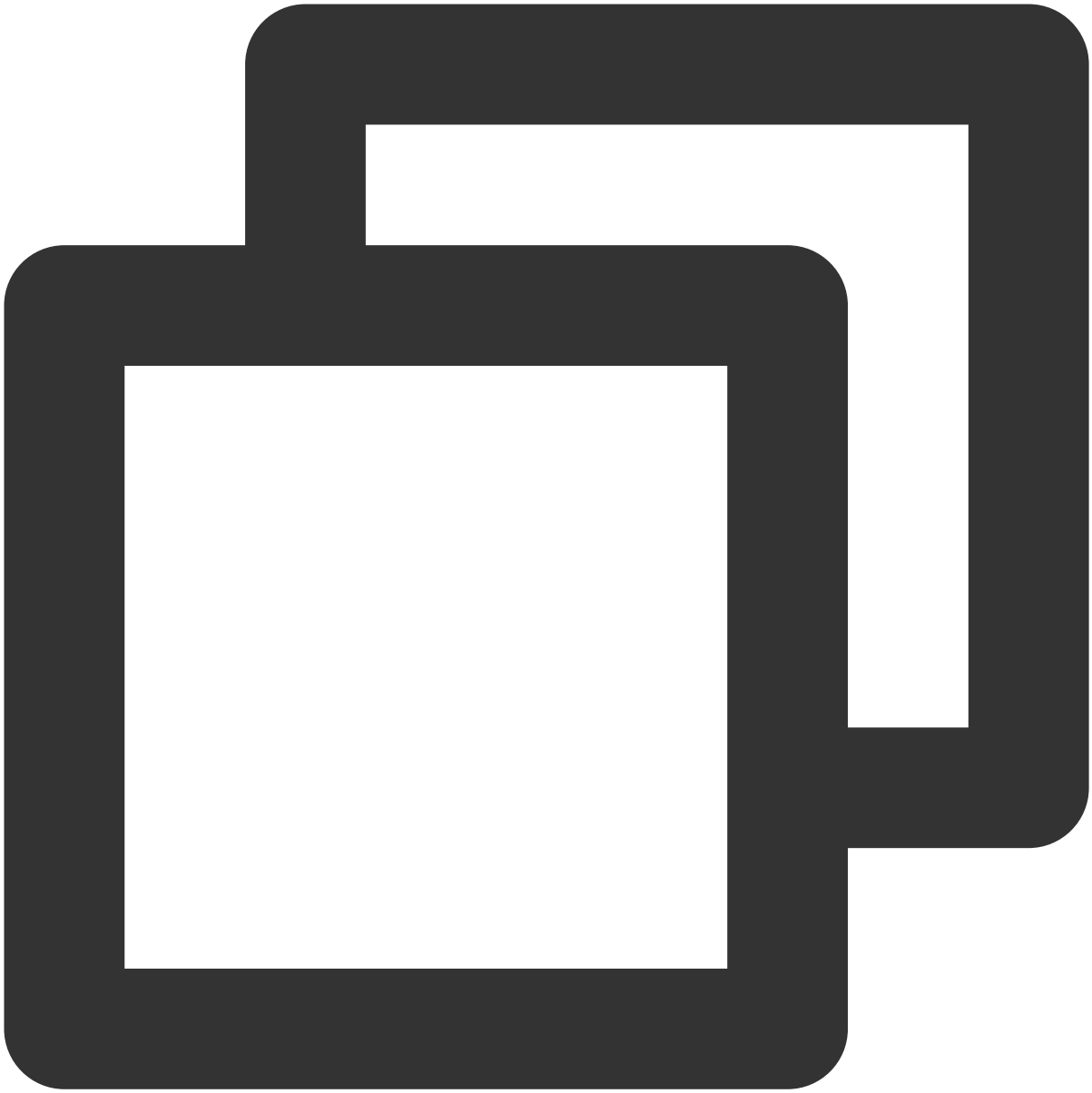
3. 얼굴 커스터마이징 및 옷 입히기(updateAvatar)



```
public void updateAvatar(List<AvatarData> avatarDataList, UpdateAvatarConfigListene
```

이 API가 호출되면 아바타 미리보기가 실시간으로 업데이트됩니다. 각 AvatarData 객체는 구성(예: 헤어스타일 변경)에 해당합니다. 여러 AvatarData 객체를 API 호출에 전달하여 아바타의 여러 측면을 편집할 수 있습니다(예: 헤어스타일 및 머리 색상 변경). API는 전달된 AvatarData 객체의 유효성을 검사합니다. 객체가 유효하면 SDK로 전달됩니다. 그렇지 않으면 callback이 전송됩니다.
예를 들어 헤어스타일을 변경하기 위해 AvatarData 객체가 전달되었지만 로컬 장치에서 헤어 모델 파일(AvatarData의 value로 지정됨)을 찾을 수 없는 경우 AvatarData 객체는 유효하지 않은 것으로 간주됩니다.
또한 홍채 이미지를 변경하기 위해 AvatarData 객체가 전달되었지만 로컬 장치에서 이미지 파일(AvatarData의 value로 지정됨)을 찾을 수 없는 경우 AvatarData 객체는 유효하지 않은 것으로 간주됩니다.

4. 아바타 설정 내보내기(exportAvatar)

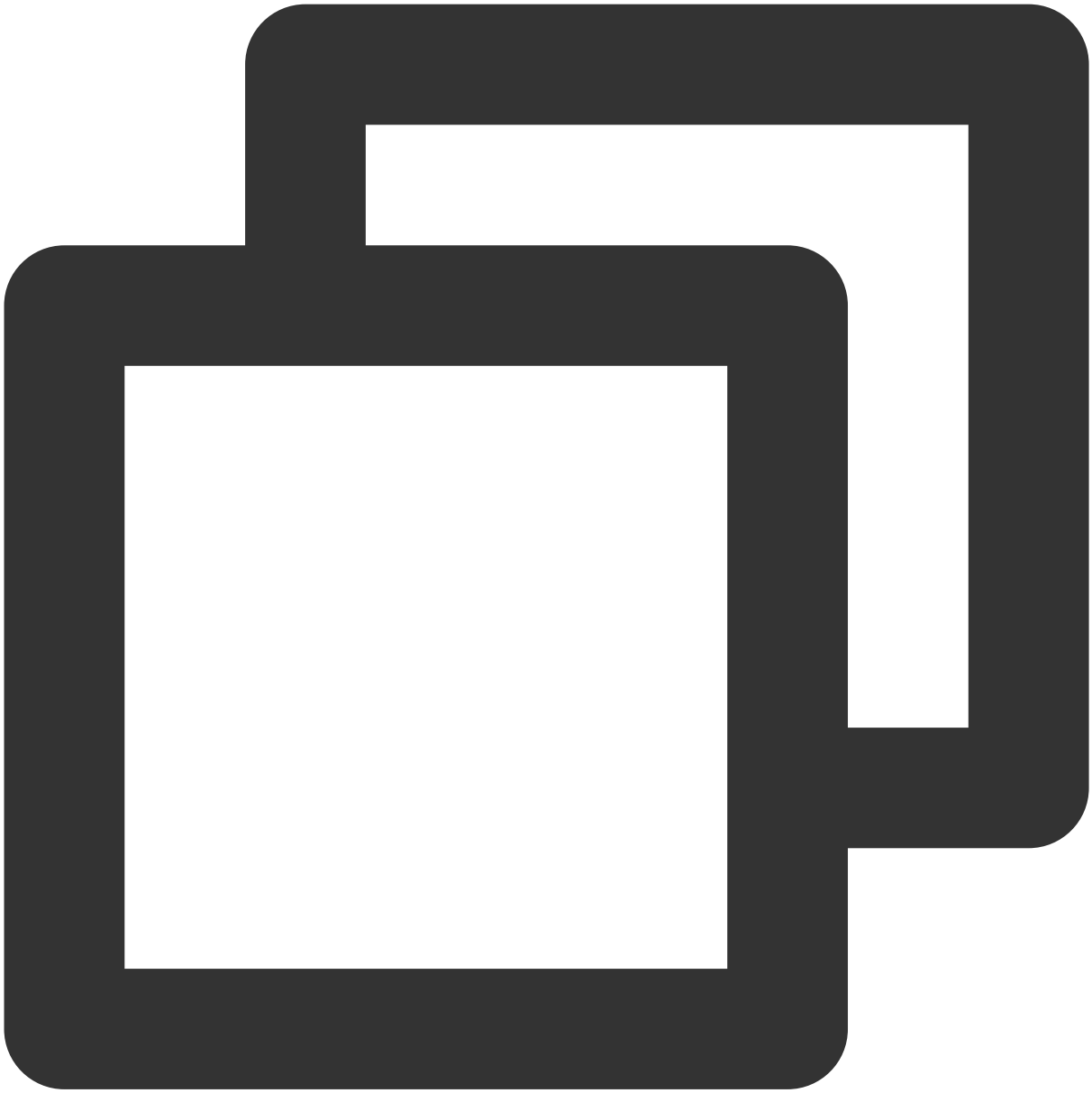


```
public static String exportAvatar(List<AvatarData> avatarDataList)
```

사용자가 아바타를 편집하면 selected 값이나 AvatarData의 모양 변경 값이 변경됩니다. 편집 후 새로운 AvatarData 목록이 생성되고 json 문자열로 내보낼 수 있습니다. 로컬에 저장하거나 서버에 업로드할 수 있습니다.
문자열은 다음 목적으로 내보내집니다.
다음 번에 XMagicApi의 loadAvatar를 호출하여 Avatar 소재를 로드할 때 XMagicProperty의 customConfigs를 JSON 문자열로 설정해야 미리보기에서 마지막으로 아바타 설정을 기억할 수 있습니다.
또한 getAllAvatarData를 호출하여 selected와 Avatar 소스 데이터의 모양 변경 값을 업데이트할 때 이 JSON 문자열을 전달해야 합니다.

5. 사진을 기반으로 아바타 커스터마이징(createAvatarByPhoto)

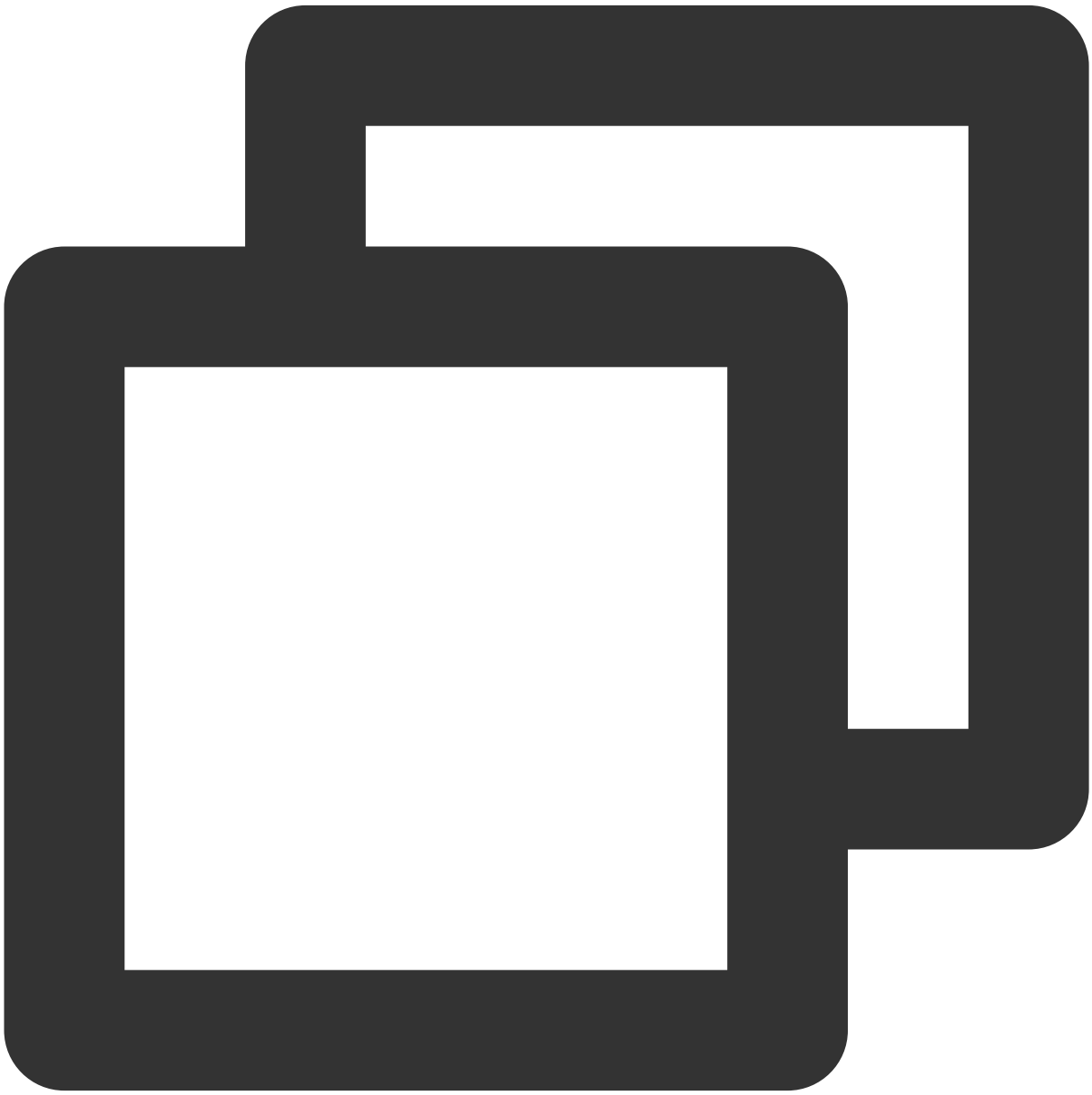
이 API가 작동하려면 SDK가 인터넷에 연결되어 있어야 합니다.



```
public void createAvatarByPhoto(String photoPath, List<String> avatarResPaths, bool
            final FaceAttributeListener faceAttributeListener)
```

photoPath: 사진 경로입니다. 얼굴이 사진 중앙에 오도록 합니다. 이상적으로는 사진에 얼굴이 하나만 포함되어야 합니다. 여러 개가 있는 경우 SDK는 임의로 하나를 선택합니다. 인식 결과를 보장하려면 사진의 짧은 면이 500px보다 길어야 합니다.

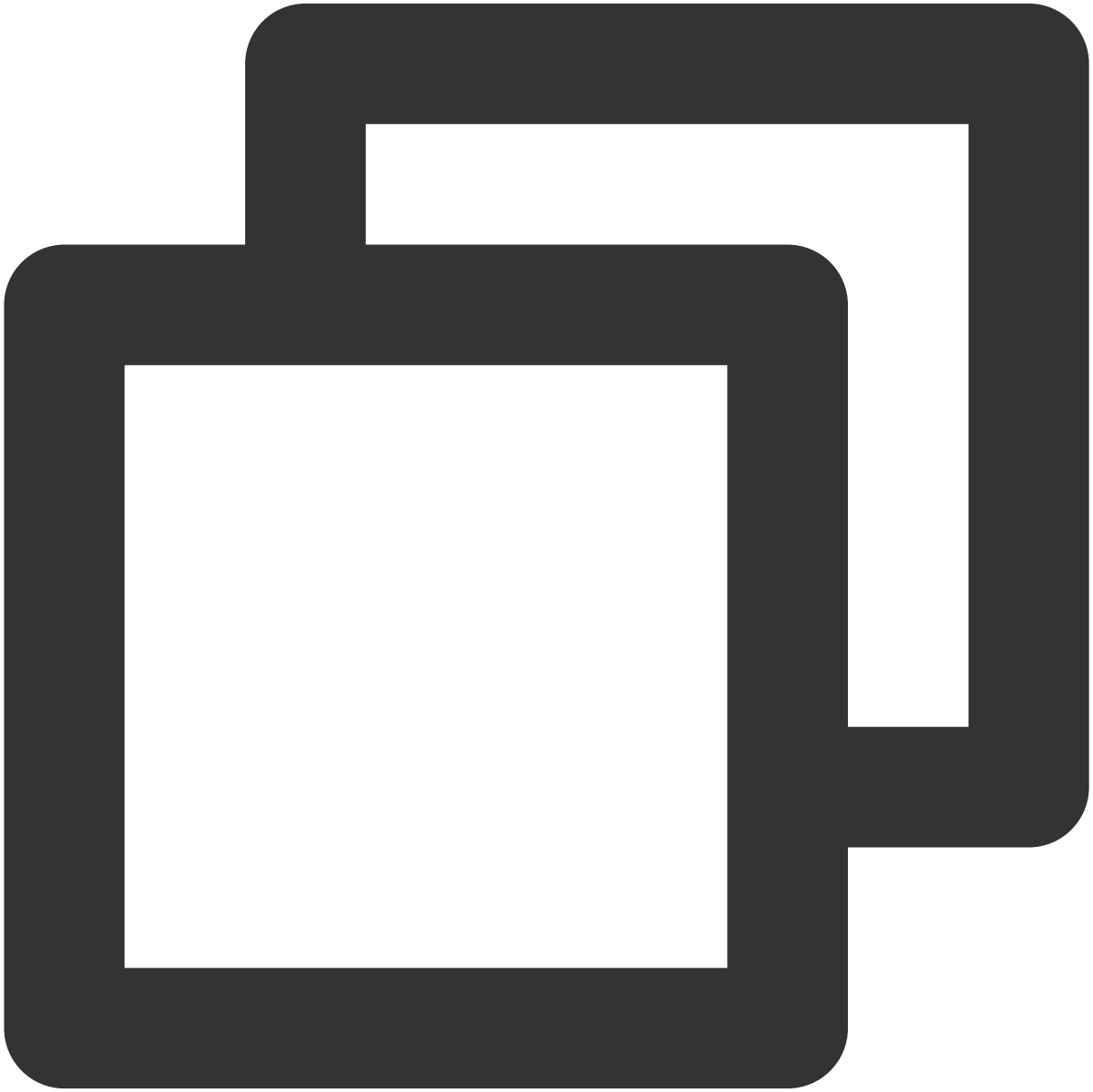
avatarResPaths: 여러 세트의 Avatar 소재를 전달할 수 있으며 SDK는 사진 분석 결과에 따라 가장 적합한 세트를 선택합니다. 참고: 현재 한 세트의 소재만 지원됩니다. 여러 세트가 전달되면 SDK는 첫 번째 세트를 사용합니다.
isMale: 사람이 남성인지 여부. 현재 이 속성은 사용되지 않습니다. 그러나 미래에 있을 수 있습니다. 올바른 값을 전달하는 것이 좋습니다.
faceAttributeListener: 사용자 지정이 실패하면 `void onError(int errCode, String msg)` 가 반환됩니다. 성공하면 `void onFinish(String matchedResPath, String srcData)` 가 반환되며 첫 번째 매개변수는 매칭된 Avatar 소재의 경로를 나타내고 두 번째 매개변수는 매칭 결과입니다(exportAvatar의 반환 값과 동일). onError의 에러 코드는 `FaceAttributeHelper.java` 에 정의되어 있습니다.



```
public static final int ERROR_NO_AUTH = 1;//권한 부족
```

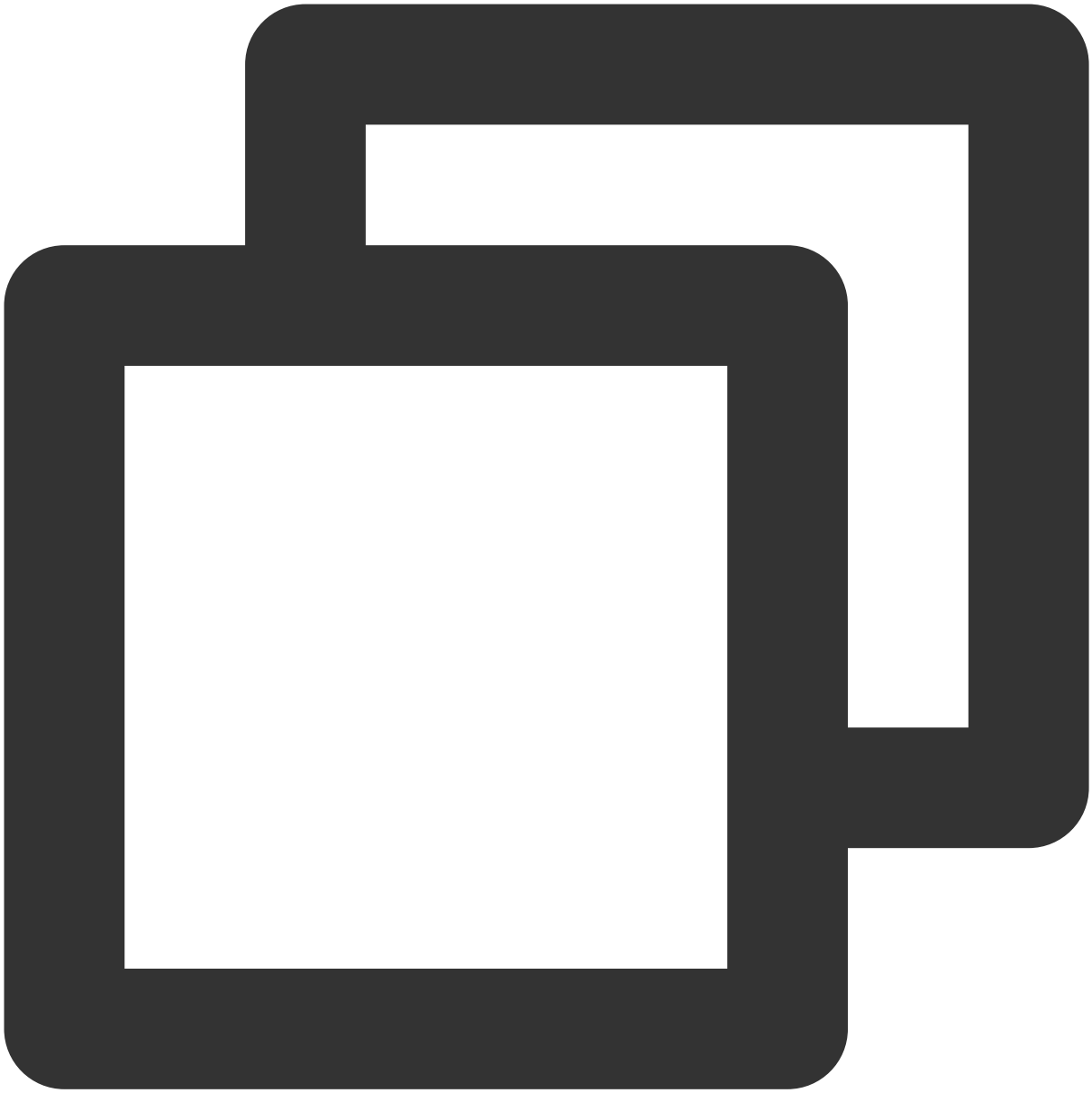
```
    public static final int ERROR_RES_INVALID = 5;//전달된 Avatar 소재 경로가 유효하지 않
    public static final int ERROR_PHOTO_INVALID = 10;//사진 읽기 실패
    public static final int ERROR_NETWORK_REQUEST_FAILED = 20;//인터넷 연결 실패
    public static final int ERROR_DATA_PARSE_FAILED = 30;//네트워크에서 반환된 데이터 파싱
    public static final int ERROR_ANALYZE_FAILED = 40;//얼굴 인식 실패
    public static final int ERROR_AVATAR_SOURCE_DATA_EMPTY = 50;//Avatar 소스 데이터 로
```

6. 다운로드한 구성 파일(addAvatarResource) 저장



```
public static Pair<Integer, List<AvatarData>> addAvatarResource(String resourceRoot
```

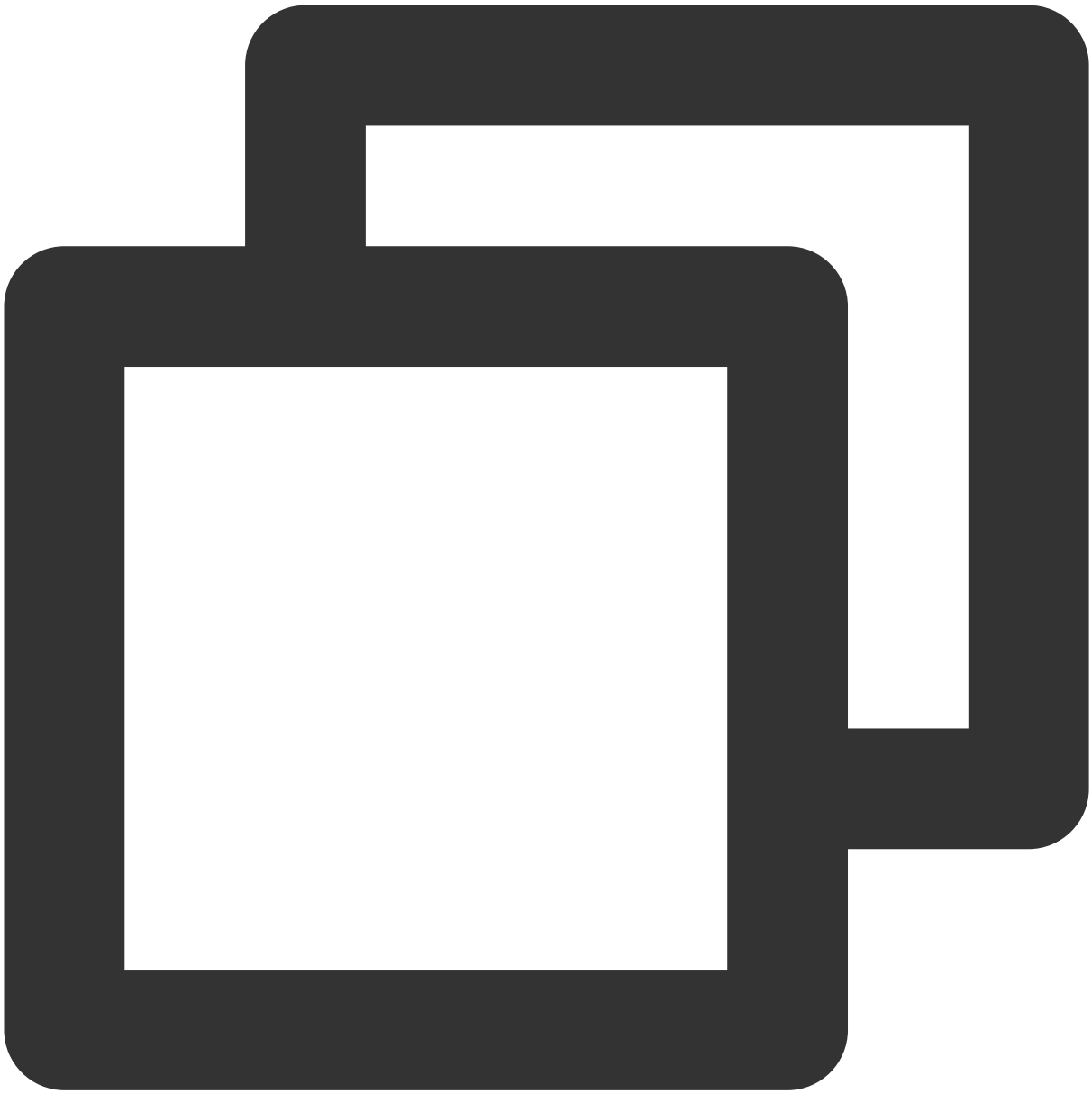
Avatar 소재를 동적으로 다운로드하는 경우에 사용하는 API입니다. 예를 들어 10개의 헤어스타일을 제공하고 그 중 하나가 장치에 동적으로 다운로드된다고 가정합니다. 다운로드한 ZIP 파일의 경로를 SDK에 전달하려면 이 API를 호출해야 합니다. SDK는 파일을 파싱하여 해당 category의 폴더에 저장합니다. 다음에 getAllAvatarData를 호출하면 SDK가 새로 추가된 데이터를 반환합니다.
매개변수 설명:
resourceRootPath: /data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/animoji_0624와 같은 Avatar 소재의 루트 디렉터리
category: 다운로드한 자료의 카테고리
zipFilePath: 다운로드한 zip 파일의 경로
API는 `Pair<Integer, List<AvatarData>>` 를 반환합니다. 여기서 pair.first는 에러 코드이고 pair.second는 새로 추가된 데이터입니다.
에러 코드는 다음과 같습니다.



```
public interface AvatarActionErrorCode {
    int OK = 0;
    int ADD_AVATAR_RES_INVALID_PARAMS = 1000;
    int ADD_AVATAR_RES_ROOT_RESOURCE_NOT_EXIST = 1001;
    int ADD_AVATAR_RES_ZIP_FILE_NOT_EXIST = 1002;
    int ADD_AVATAR_RES_UNZIP_FILE_FAILED = 1003;
    int ADD_AVATAR_RES_COPY_FILE_FAILED = 1004;
}
```

7. AvatarData 호출

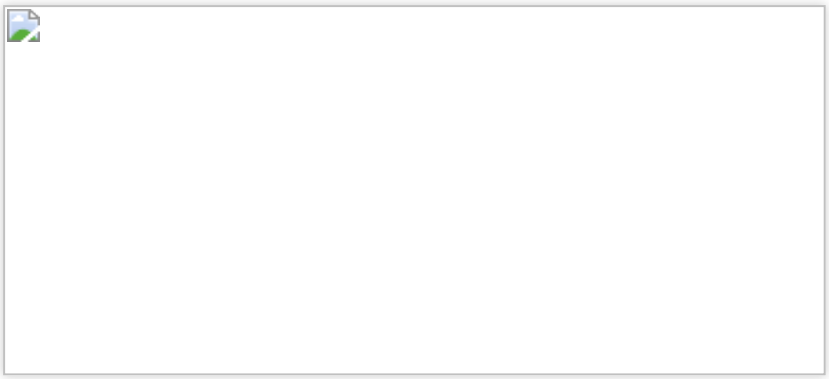
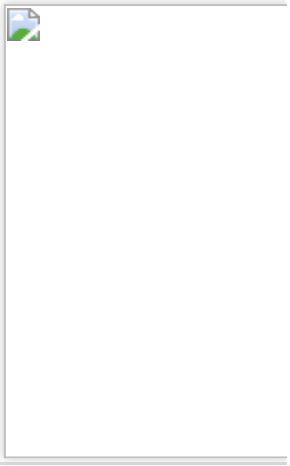
AvatarData 클래스는 위 API의 핵심입니다. AvatarData에는 다음 필드가 포함됩니다.



```
public class AvatarData {
    /**
     * 선택자(selector) 데이터입니다. 예를 들어 안경의 경우 여러 종류의 안경이 제공되며 하나만
     */
    public static final int TYPE_SELECTOR = 0;

    /**
     * 뺨 너비와 같은 슬라이더(slider) 데이터입니다.
     */
```

```
    public static final int TYPE_SLIDER = 1;

    //얼굴 모양 및 눈매 조정 등의 카테고리입니다. 표준 category는 AvatarCategory.java에 정의
    //이 필드는 비워 둘 수 없습니다.
    public String category;

    //아바타 구성 item의 ID입니다.
    //예를 들어, 각 유형의 안경에는 고유한 id가 있습니다. 각 조정 항목도 마찬가지입니다.
    //이 필드는 비워 둘 수 없습니다.
    public String id;

    //TYPE_SELECTOR 또는 TYPE_SLIDER
    public int type;

    // 유형이 selector인 경우 이 필드는 항목이 선택되었는지 여부를 나타냅니다
    public boolean selected = false;

    //각 아이콘 또는 조정 항목에는 구성 세부 정보의 세 가지 측면이 있습니다.
    public String entityName;
    public String action;
    public JsonObject value;
}
```

AvatarData 객체는 헤어스타일 변경이나 뺨 조정과 같은 가장 작은 구성 단위입니다.

헤어스타일 변경	뺨 조정
	

항목이 selector 유형인 경우 AvatarData에서 selected 값을 변경하여 구성합니다. 예를 들어 A, B, C, D의 네 종류의 안경이 있다고 가정합니다. 사용자가 A를 선택하면 A의 selected를 true로 설정하고 B, C, D의 안경을 false로 설정합니다. 사용자가 B를 선택하면 B의 selected를 true로 설정하고 A, C, D의 selected를 false로 설정합니다.
항목이 slider 유형인 경우 AvatarData에서 value를 변경하여 구성합니다. value 필드는 여러 key-value 쌍을 포함하는 JsonObject입니다. key-value 쌍의value를 슬라이더 값으로 설정합니다.

AvatarData에 대한 추가 정보

AvatarData의 세 가지 핵심 필드는 entityName, action 및 value이며 SDK가 구성 데이터를 파싱할 때 해당 값이 자동으로 입력됩니다. 대부분의 경우 이 세 필드의 세부 정보를 처리할 필요가 없습니다. 그러나 슬라이더 데이터의 경우 value 필드의 key-value 쌍을 구문 분석하고 UI에 설정된 슬라이더 값을 기반으로 구성해야 합니다.
AvatarData의 세 가지 주요 필드: entityName, action 및 value

entityName 필드

entityName은 얼굴, 눈, 머리카락, 상의 또는 신발과 같이 편집할 아바타 부분을 지정합니다.

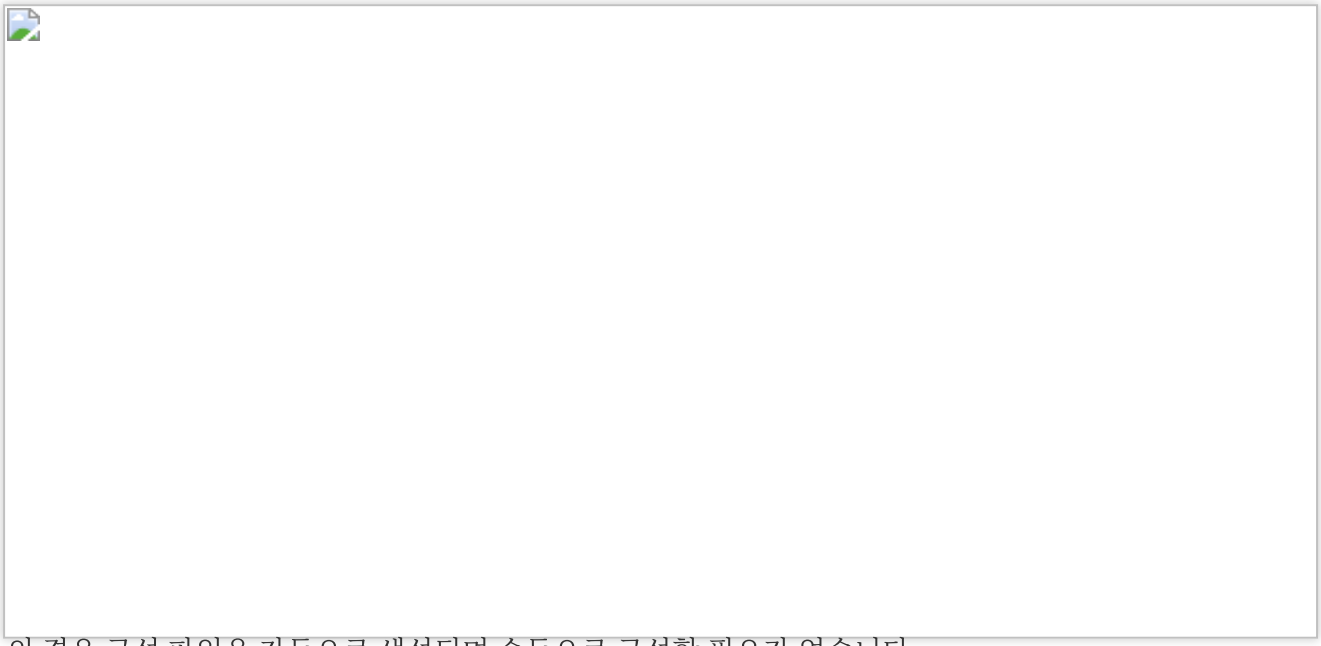
action 및 value 필드

action 필드는 entityName에서 수행할 작업(action)을 나타냅니다. 다섯 가지 action 옵션은 SDK의 AvatarAction.java에 정의되어 있으며 각 action의 의미 및 value에 대한 요구 사항은 다음과 같습니다.

action	설명	value에 대한 요구 사항
changeColor	기본 색상과 이미션(emission) 색상을 포함하는 현재 재질의 색상을 변경합니다	값은 JsonObject여야 하며 필수입니다. 소재 커스터마이징 툴에 의해 자동으로 생성됩니다.
changeTexture	색상 텍스처 맵, 금속/거칠기 텍스처 맵, AO(ambient occlusion) 맵, 노멀 맵 및 이미션 맵 등을 포함하여 현재 재질의 맵을 수정합니다	값은 JsonObject여야 하며 필수입니다. 소재 커스터마이징 툴에 의해 자동으로 생성됩니다.
shapeValue	blendShape 값을 수정합니다. 이 작업은 종종 얼굴 특징을 미세 조정하는 데 사용됩니다.	값은 JsonObject(여기서 key는 shape 이름이고 value는 float 소수점)여야 하며 필수입니다. 소재 커스터마이징 툴에 의해 자동으로 생성됩니다.
replace	예를 들어 안경, 헤어 스타일 또는 옷과 같은 서브 모델을 대체합니다	value는 3D 변환 정보, 모델 경로 및 새 서브 모델의 재질 경로를 설명하는 JsonObject여야 합니다. 소재 커스터마이징 툴에 의해 자동으로 생성됩니다. 서브 모델을 숨기려면 null로 설정합니다.
basicTransform	위치, 회전 및 크기 조정 설정을 조정합니다. 이 작업은 확대/축소 또는 각도를 변경하여 전체 보기와 절반 길이 보기 사이를 전환하는 데 자주 사용됩니다.	값은 JsonObject여야 하며 필수입니다. 소재 커스터마이징 툴에 의해 자동으로 생성됩니다.

Avatar 사용자 지정 데이터 구성

avatar 구성은 resources 폴더(소재/custom_configs/resources)에 저장됩니다.



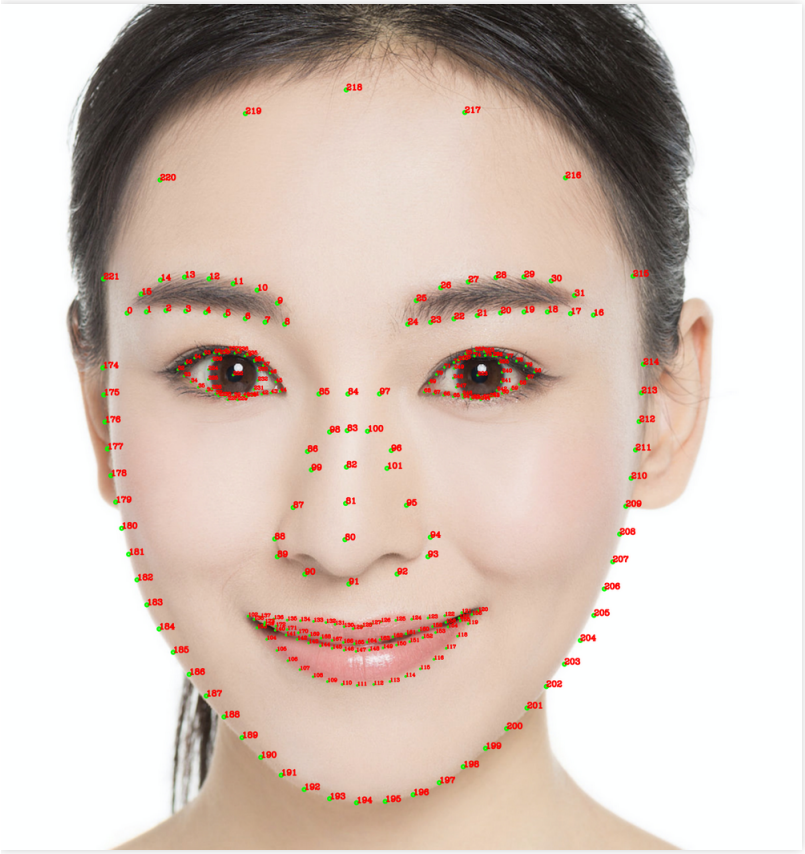
대부분의 경우 구성 파일은 자동으로 생성되며 수동으로 구성할 필요가 없습니다.
구성을 자동으로 생성하려면 TencentEffectStudio를 사용하여 당사 표준에 따라 소재를 디자인하고 당사에서 제공하는 resource_generator_gui App을 실행하십시오(현재 MacOS에서만 사용 가능). 자세한 내용은 Tencent Cloud 아바타 디자인 매뉴얼을 참고하십시오.

원자 기능 통합 가이드
얼굴 특징점 감지 통합 가이드

최종 업데이트 날짜: : 2022-12-15 11:30:53

이 기능은 얼굴이 프레임을 벗어났거나, 가려졌거나 여러 얼굴이 있는 경우를 감지합니다. 256개의 얼굴 특징점을 인식할 수 있습니다.

256 얼굴 키포인트



iOS API

iOS 통합

Tencent Effect iOS SDK 통합 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

XMagic 리스너 등록

```
/// @brief SDK 이벤트 리스너
/// @param listener: 이벤트 리스너 콜백, 주로 AI 이벤트, Tips 프롬프트 이벤트, Asset 이벤트로 구분
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;
```

YTSDKEventListener

```
#pragma mark - 콜백 API
/// @brief SDK 이벤트 콜백 API
@protocol YTSDKEventListener <NSObject>
/// @brief YTDataUpdate 콜백
/// @param event NSString*형식의 콜백
- (void)onYTDataEvent:(id _Nonnull)event;
/// @brief AI 이벤트 콜백
/// @param event dict 형식의 콜백
- (void)onAIEvent:(id _Nonnull)event;
/// @brief 팁 이벤트 콜백
/// @param event dict 형식의 콜백
- (void)onTipsEvent:(id _Nonnull)event;
/// @brief 리소스 이벤트 콜백
/// @param event string 형식의 콜백
- (void)onAssetEvent:(id _Nonnull)event;
@end
```

콜백이 성공적으로 구성된 후 SDK는 각 비디오 프레임에 대한 얼굴 데이터의 콜백을 보냅니다.

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

반환된 data는 JSON 형식이며 다음 필드를 포함합니다(256개의 얼굴 키포인트에 대한 자세한 내용은 위 그림 참고).

```
/// @note 콜백 필드
/**
| 필드 | 유형 | 값 범위 | 설명 |
| :---- | :---- |:---- | :---- |
| trace_id | int | [1,INF) | 얼굴 id, 연속 비디오 스트림에서 얻은 얼굴의 얼굴 id가 동일하면 동일한 사람으로 간주할 수 있습니다 |
| face_256_point | float | [0,screenWidth] 또는 [0,screenHeight] | 얼굴 키포인트의 위치, 256개의 얼굴 키포인트에 대해 총 512개의 값이 있습니다. (0, 0)은 화면의 왼쪽 상단 모서리입니다 |
| face_256_visible | float | [0,1] | 얼굴 256 키 포인트 가시도 |
| out_of_screen | bool | true/false | 얼굴이 프레임 밖에 있는지 여부 |
| left_eye_high_vis_ratio | float | [0,1] | 왼쪽 눈의 가시성이 높은 키포인트의 비율 |
| right_eye_high_vis_ratio | float | [0,1] | 오른쪽 눈의 가시성이 높은 키포인트의 비율 |
```

```
| left_eyebrow_high_vis_ratio | float | [0,1] | 왼쪽 눈썹의 가시성이 높은 키포인트의 비율 |
| right_eyebrow_high_vis_ratio | float | [0,1] | 오른쪽 눈썹의 가시성이 높은 키포인트의 비율 |
| mouth_high_vis_ratio | float | [0,1] | 입에 대한 가시성이 높은 키포인트의 비율 |
**/
- (void)onYTDataEvent:(id _Nonnull)event;
```

Android API

Android 통합

Tencent Effect Android SDK 통합 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

Xmagic 리스너 등록

얼굴 키포인트 및 기타 데이터의 콜백을 구성합니다.

```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
안면 인식 정보 등 데이터 콜백 설정
public interface XmagicYTDataListener {
void onYTDataUpdate(String data)
}
```

onYTDataUpdate 는 JSON string 구조를 반환하고 최대 5개의 안면 인식 정보를 반환합니다.

```
{
"face_info":[{
"trace_id":5,
"face_256_point":[
180.0,
112.2,
...
],
"face_256_visible":[
0.85,
...
],
"out_of_screen":true,
"left_eye_high_vis_ratio:1.0,
"right_eye_high_vis_ratio":1.0,
"left_eyebrow_high_vis_ratio":1.0,
"right_eyebrow_high_vis_ratio":1.0,
```

```
"mouth_high_vis_ratio":1.0
},
...
]
}
```

필드 의미

필드	유형	값범위	설명
trace_id	int	[1,INF)	얼굴 ID입니다. 연속 비디오 스트림에서 얻은 얼굴의 얼굴 ID가 동일하면 동일한 사람에 속합니다.
face_256_point	float	[0,screenWidth] 또는 [0,screenHeight]	얼굴 키포인트의 위치입니다. 256개의 얼굴 키포인트에 대해 총 512개의 값이 있습니다. (0, 0)은 화면의 왼쪽 상단 모서리입니다.
face_256_visible	float	[0,1]	256개의 얼굴 키포인트의 가시성입니다.
out_of_screen	bool	true/false	얼굴이 프레임 밖에 있는지 여부입니다.
left_eye_high_vis_ratio	float	[0,1]	왼쪽 눈의 가시성이 높은 키포인트의 비율입니다.
right_eye_high_vis_ratio	float	[0,1]	오른쪽 눈의 가시성이 높은 키포인트의 비율입니다.
left_eyebrow_high_vis_ratio	float	[0,1]	왼쪽 눈썹의 가시성이 높은 키포인트의 비율입니다.
right_eyebrow_high_vis_ratio	float	[0,1]	오른쪽 눈썹의 가시성이 높은 키포인트의 비율입니다.
mouth_high_vis_ratio	float	[0,1]	입에 대한 가시성이 높은 키포인트의 비율입니다.

매개변수

매개변수	의미
XmagicApi.XmagicYTDataListener ytDataListener	콜백 구현 클래스입니다.

얼굴 표정
iOS

최종 업데이트 날짜: : 2023-02-27 14:27:24

기능 설명

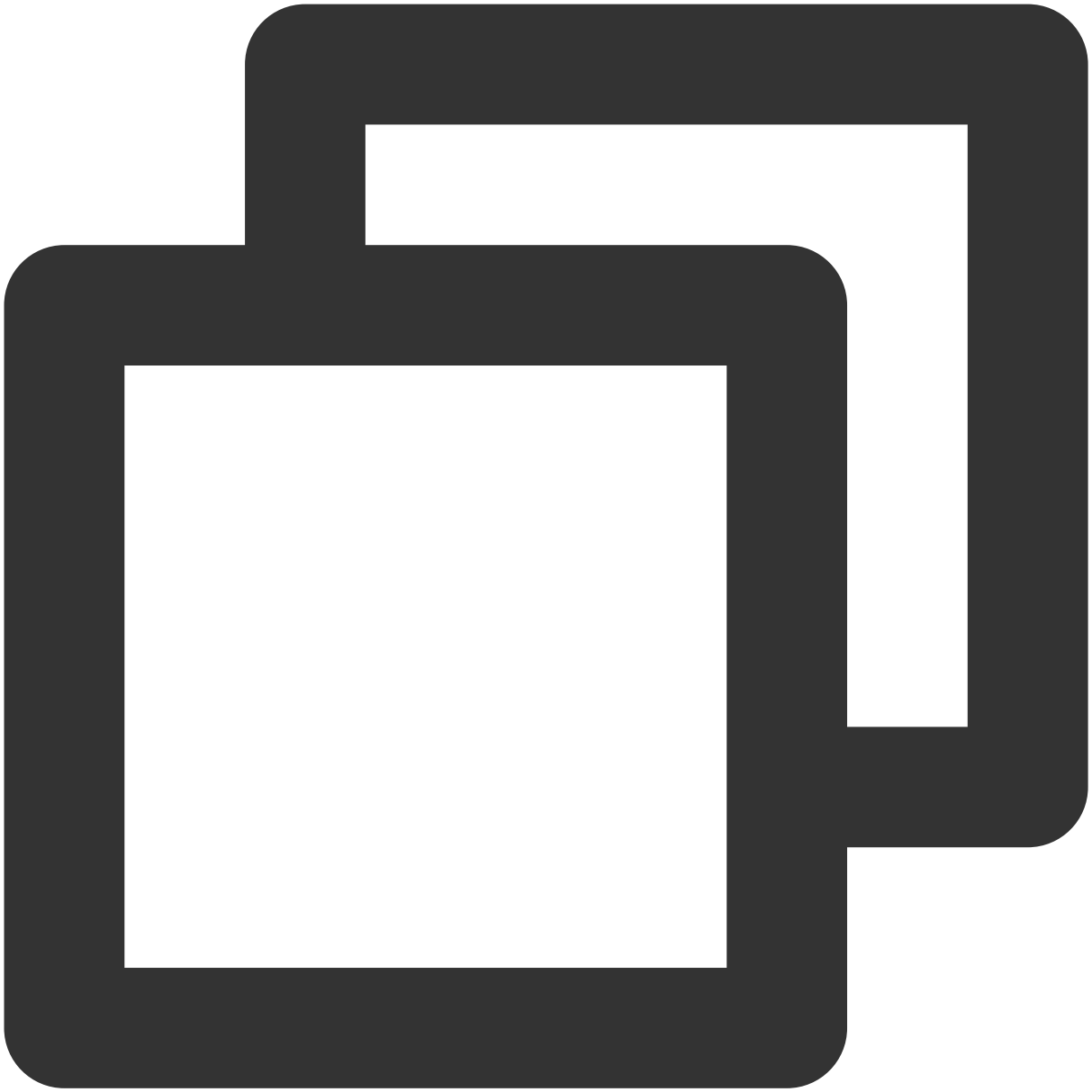
이 기능을 사용하면 카메라에서 캡처한 openGL 텍스처를 기반으로 Apple ARKit의 표준을 사용하여 52개의 BlendShape를 생성할 수 있습니다. 자세한 내용은 ARFaceAnchor를 참고하십시오. blendshape 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

iOS 통합 가이드

Tencent Effect iOS SDK의 자세한 통합 방법은 Tencent Effect SDK 통합하기를 참고하십시오.

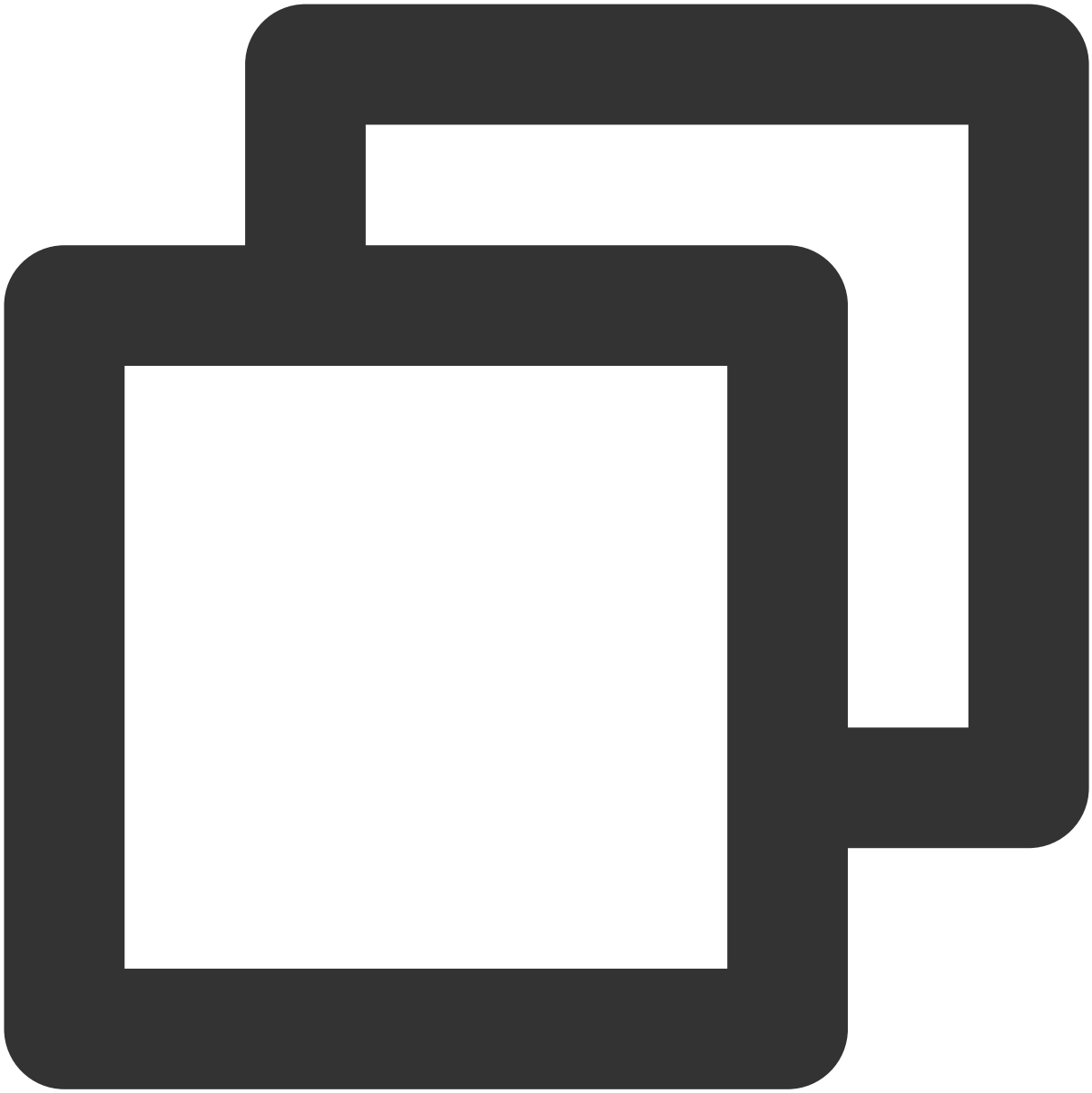
API 호출

1. 효과 활성화:



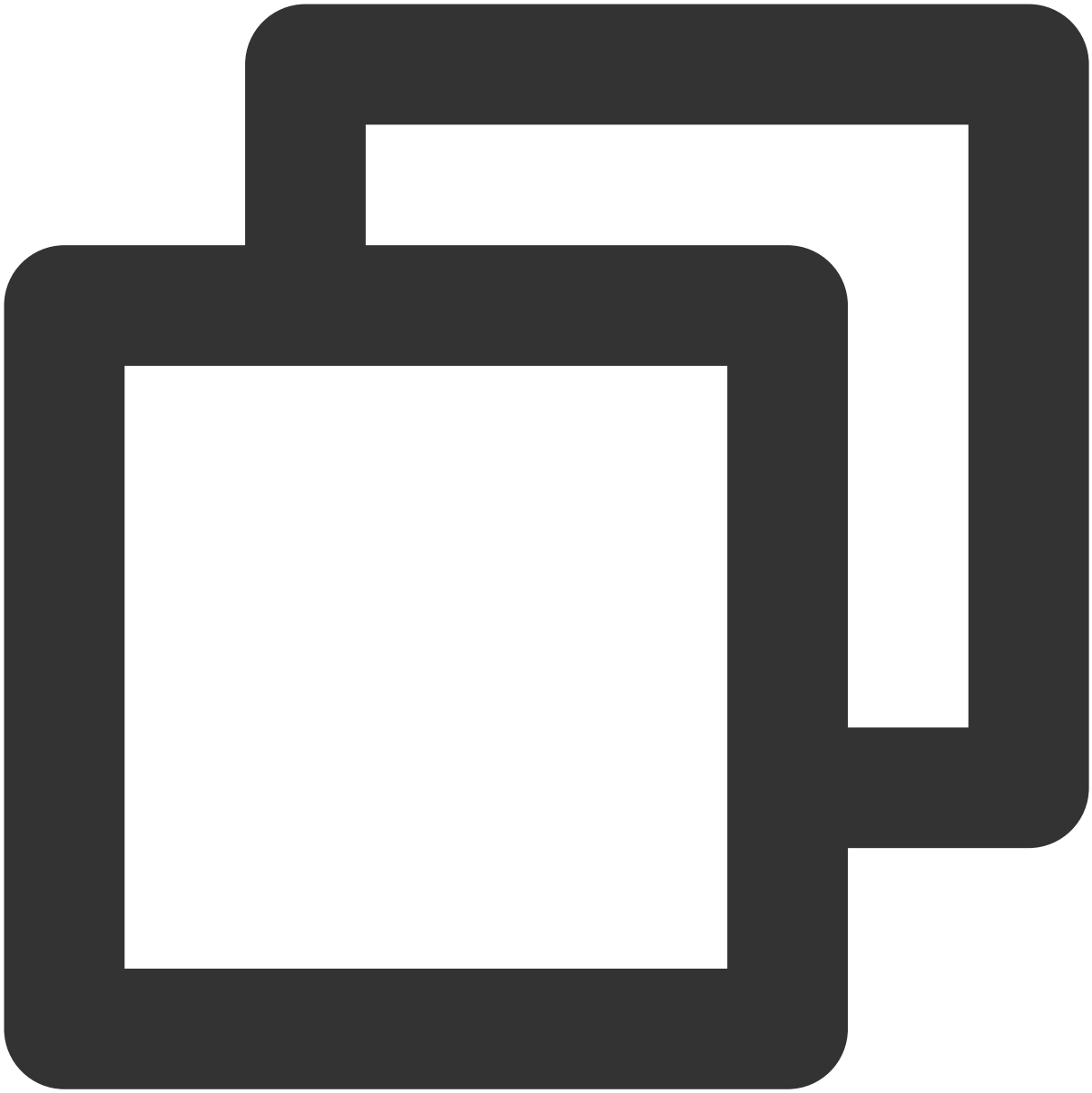
```
[self.beautyKit setFeatureEnableDisable:ANIMOJI_52_EXPRESSION enable:YES];
```

2. 얼굴 키포인트의 콜백을 구성합니다.



```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;

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

onYTDataUpdate 는 JSON string 구조를 반환하고 최대 5개의 안면 인식 정보를 반환합니다.



```
{
 "face_info":[{
  "trace_id":5,
  "face_256_point":[
 180.0,
 112.2,
 ...
  ],
  "face_256_visible":[
 0.85,
 ...
```

```
    ],
    "out_of_screen":true,
    "left_eye_high_vis_ratio:1.0,
    "right_eye_high_vis_ratio":1.0,
    "left_eyebrow_high_vis_ratio":1.0,
    "right_eyebrow_high_vis_ratio":1.0,
    "mouth_high_vis_ratio":1.0,
    "expression_weights":[
       0.12,
       -0.32
       ...
   ]
   },
   ...
   ]
  }
```

필드 의미

trace_id: 얼굴 ID입니다. 연속 비디오 스트림 과정에서 얻은 얼굴의 얼굴 ID가 동일하면 동일한 사람으로 간주할 수 있습니다.

expression_weights: 실시간 blendshape 데이터입니다. 52개 요소의 배열이며, 요소의 값 범위는 0에서 1.0입니다.

{ "eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLookOutLeft","eyeLookUpLeft","eyeSquintLeft","eyeWideLeft","eyeBlinkRight","eyeLookDownRight","eyeLookInRight","eyeLookOutRight","eyeLookUpRight","eyeSquintRight","eyeWideRight","jawForward","jawLeft","jawRight","jawOpen","mouthClose","mouthFunnel","mouthPucker","mouthRight","mouthLeft","mouthSmileLeft","mouthSmileRight","mouthFrownRight","mouthFrownLeft","mouthDimpleLeft","mouthDimpleRight","mouthStretchLeft","mouthStretchRight","mouthRollLower","mouthRollUpper","mouthShrugLower","mouthShrugUpper","mouthPressLeft","mouthPressRight","mouthLowerDownLeft","mouthLowerDownRight","mouthUpperUpLeft","mouthUpperUpRight","browDownLeft","browDownRight","browInnerUp","browOuterUpLeft","browOuterUpRight","cheekPuff","cheekSquintLeft","cheekSquintRight","noseSneerLeft","noseSneerRight","tongueOut"}

나머지 필드는 얼굴 특징 정보입니다. 반환 여부는 사용하는 License 유형에 따라 다릅니다. 얼굴 표정 데이터만 필요한 경우 해당 필드를 무시할 수 있습니다.

Android

최종 업데이트 날짜: : 2023-02-27 14:27:24

기능 설명

이 기능을 사용하면 카메라에서 캡처한 openGL 텍스처를 기반으로 Apple ARKit의 표준을 사용하여 52개의 BlendShape를 생성할 수 있습니다. 자세한 내용은 ARFaceAnchor를 참고하십시오. blendshape 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

Android 통합 가이드

Tencent Effect Android SDK를 통합하는 방법에 대한 지침은 Tencent Effect SDK 통합하기를 참고하십시오.

API 호출

1. 효과 활성화:

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

2. 얼굴 키포인트의 콜백을 구성합니다.

```
//XmagicApi.java
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
```

```
public interface XmagicYTDataListener {
void onYTDataUpdate(String data)
}
```

onYTDataUpdate 는 JSON string 구조를 반환하고 최대 5개의 안면 인식 정보를 반환합니다.

```
{
"face_info":[{
"trace_id":5,
"face_256_point":[
180.0,
112.2,
...
],
"face_256_visible":[
0.85,
...
],
"out_of_screen":true,
"left_eye_high_vis_ratio:1.0,
"right_eye_high_vis_ratio":1.0,
"left_eyebrow_high_vis_ratio":1.0,
"right_eyebrow_high_vis_ratio":1.0,
"mouth_high_vis_ratio":1.0,
"expression_weights":[
0.12,
-0.32
...
]
},
...
]
}
```

필드 의미

- **trace_id**: 얼굴 ID입니다. 연속 비디오 스트림 과정에서 얻은 얼굴의 얼굴 ID가 동일하면 동일한 사람으로 간주할 수 있습니다.
- **expression_weights**: 실시간 blendshape 데이터입니다. 52개 요소의 배열이며, 요소의 값 범위는 -1.0에서 1.0입니다.
- 나머지 필드는 얼굴 특징 정보입니다. 반환 여부는 사용하는 License 유형에 따라 다릅니다. 얼굴 표정 데이터만 필요한 경우 해당 필드를 무시할 수 있습니다.

신체 키포인트
iOS

최종 업데이트 날짜: : 2023-03-06 10:07:59

기능 설명

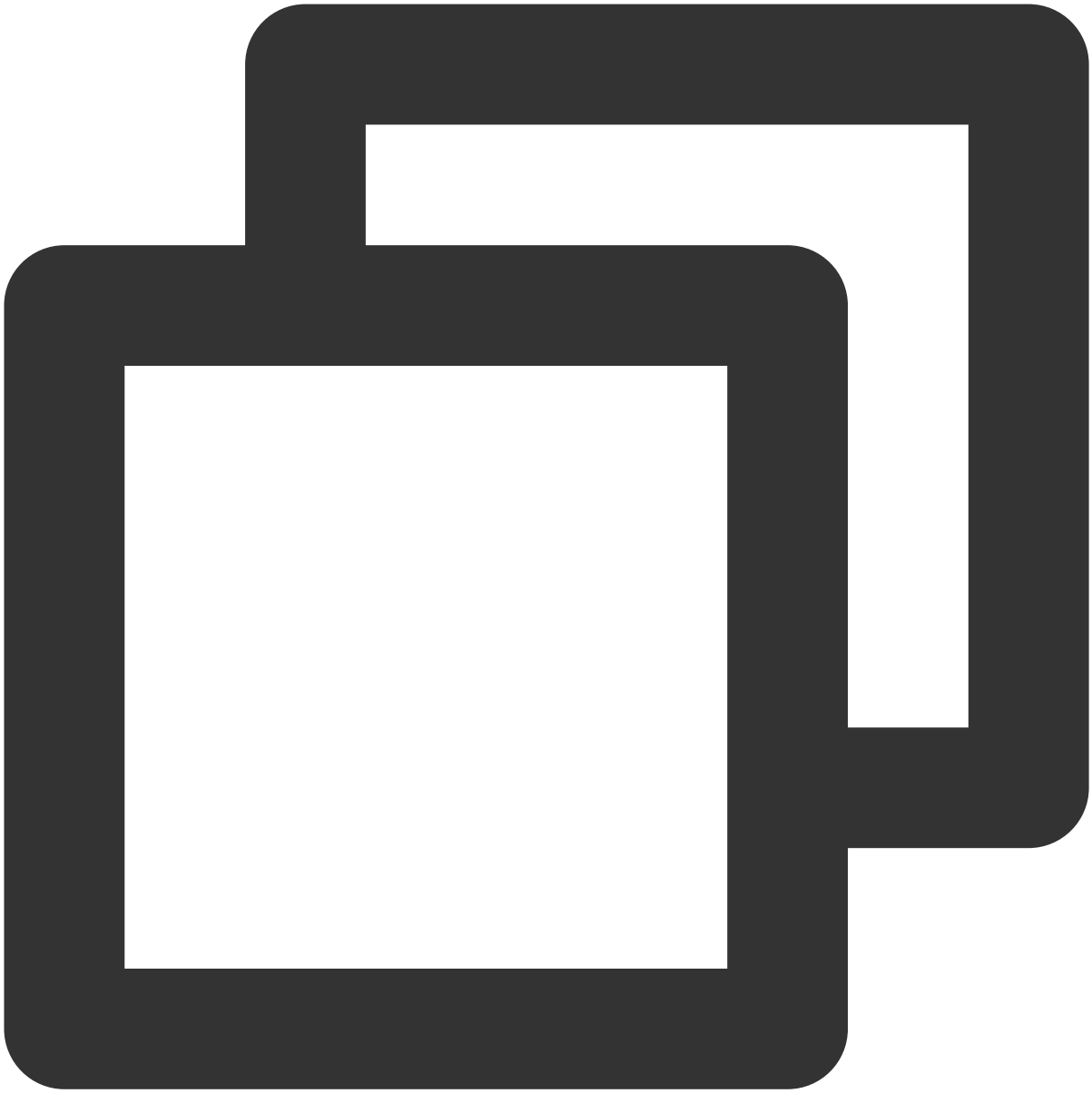
카메라에서 캡처한 openGL 텍스처를 기반으로 신체 3D 데이터를 생성할 수 있습니다. 3D 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

iOS 통합 가이드

먼저 Tencent Effect SDK를 통합해야 합니다. 자세한 내용은 Tencent Effect SDK 통합하기를 참고하십시오.

API 호출

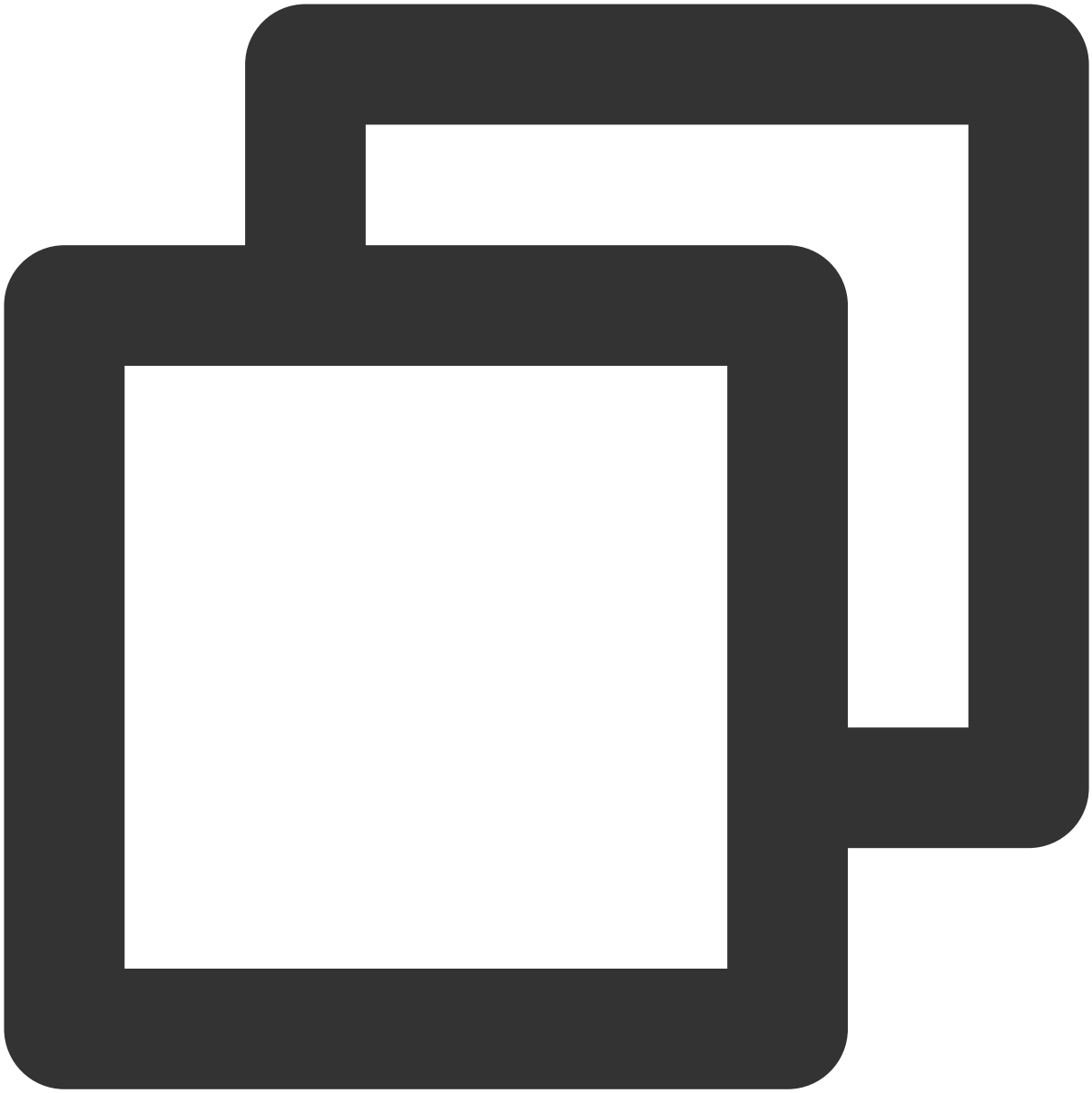
1. 효과 활성화(XMagic.h).



```
- (void)setFeatureEnableDisable:(NSString *_Nonnull)featureName enable:(BOOL)enable
```

featureName에 `XmagicConstant.FeatureName.BODY_3D_POINT` 를 입력합니다.

2. 데이터 콜백 설정(XMagic.h)



```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;

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

onYTDataEvent는 JSON 구조의 string 데이터를 반환하며, 예시는 다음과 같습니다.

"face_info"는 얼굴 관련 데이터로, 신체 3D 데이터와 관련이 없으며 무시할 수 있습니다.
"body_3d_info"의 각 필드에 대한 설명은 아래를 참고하십시오.

신체 포인트 및 포인트 데이터에 대한 설명

자세한 내용은 신체 포인트 및 데이터 설명을 참고하십시오.

Android

최종 업데이트 날짜: : 2023-02-27 14:27:24

기능 설명

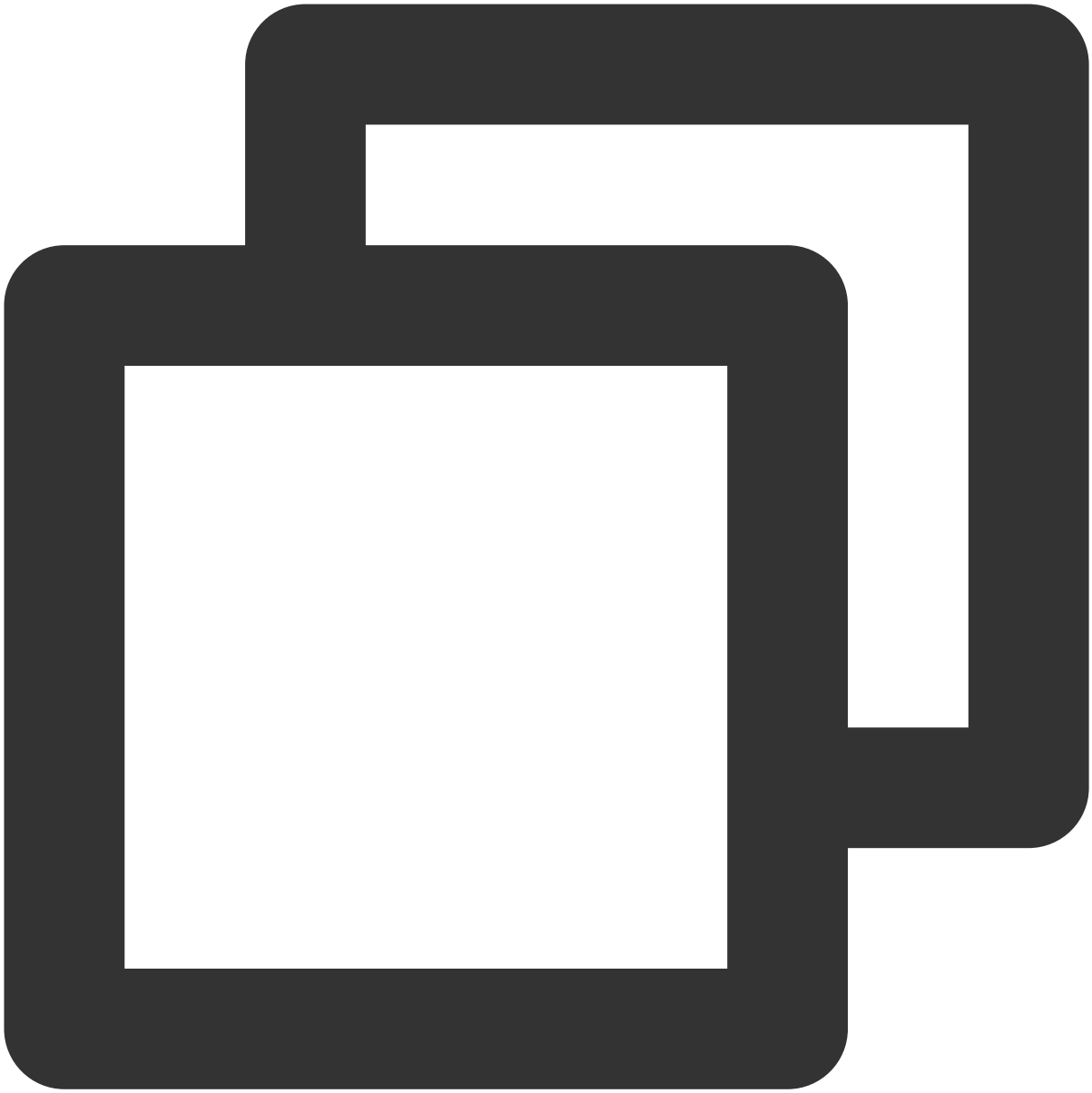
카메라에서 캡처한 openGL 텍스처를 기반으로 신체 3D 데이터를 생성할 수 있습니다. 3D 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

Android 통합

먼저 Tencent Effect SDK를 통합해야 합니다. 자세한 내용은 Tencent Effect SDK 통합하기를 참고하십시오.

API 호출

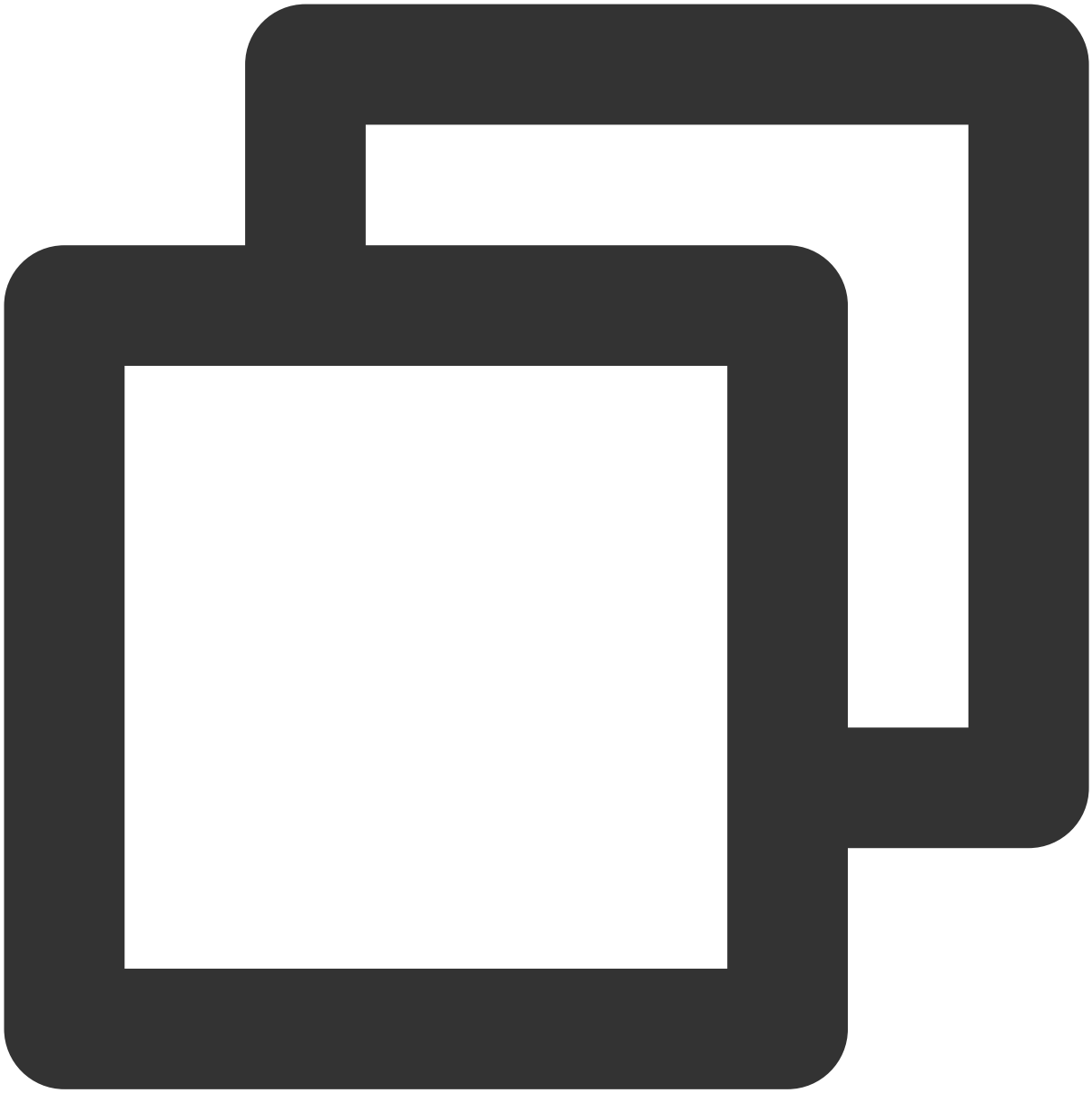
1. 효과 활성화(XmagicApi.java).



```
public void setFeatureEnableDisable(String featureName, boolean enable);
```

featureName에 `XmagicConstant.FeatureName.BODY_3D_POINT` 를 입력합니다.

2. 데이터 콜백 설정(XmagicApi.java)



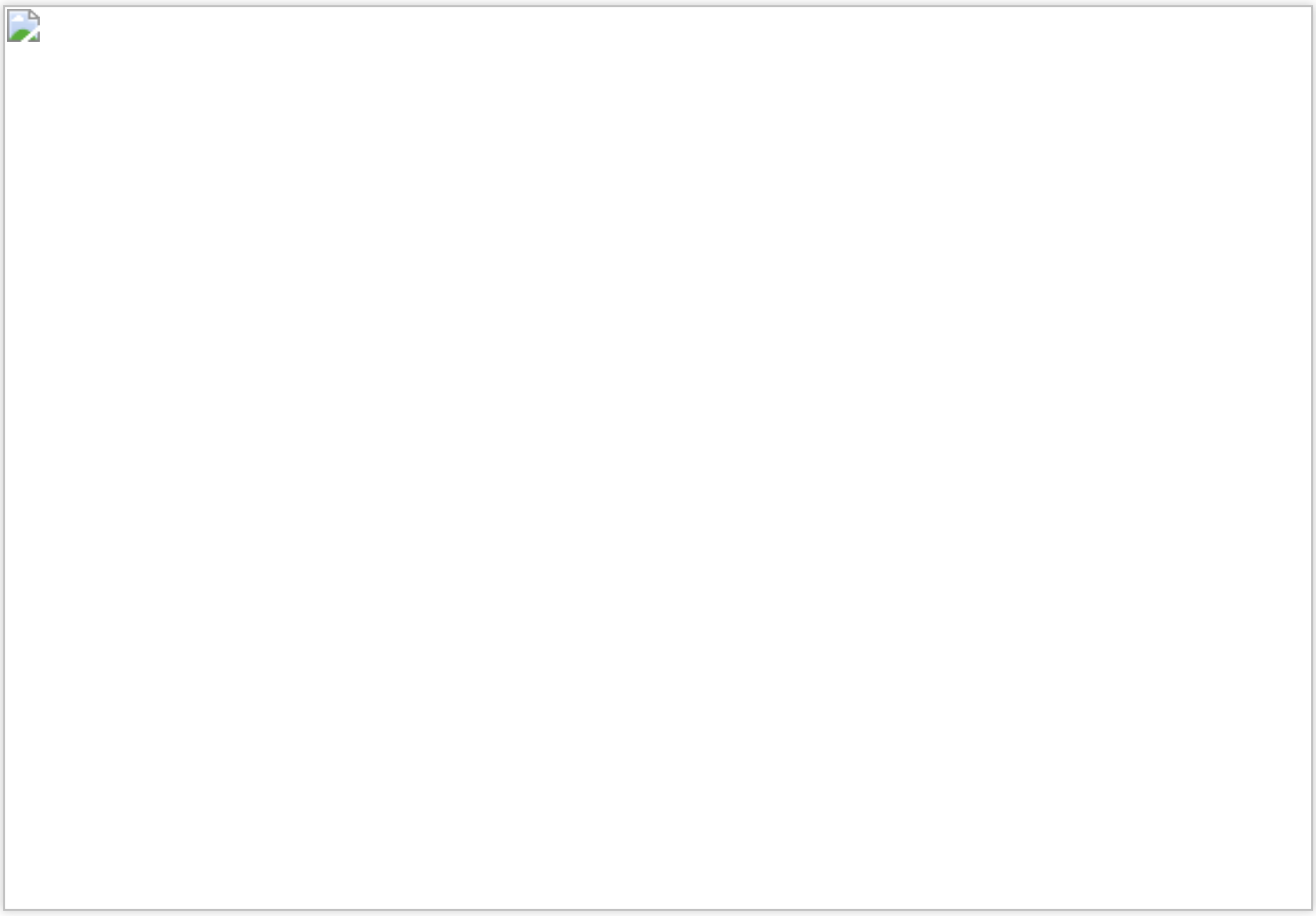
```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)

public interface XmagicYTDataListener {
 void onYTDataUpdate(String data)
}
```

onYTDataUpdate는 JSON 구조의 string 데이터를 반환합니다. 예TL는 다음과 같습니다.
"face_info"는 얼굴 관련 데이터로, 신체 3D 데이터와 관련이 없으며 무시할 수 있습니다.
"body_3d_info"의 각 필드에 대한 설명은 아래를 참고하십시오.

신체 포인트 및 포인트 데이터에 대한 설명

표준 SMPL 포인트 정의

표준 SMPLX 손 골격 포인트 정의



SDK에서 출력되는 JSON 데이터의 예시는 다음과 같습니다.

body_3d_info의 각 필드에 대한 설명은 다음과 같습니다.

imageWidth, imageHeight: SDK에 입력되는 이미지의 너비와 높이

items: 배열, 현재 하나의 요소만 있음

index: 보관 위치, 현재 무시 가능

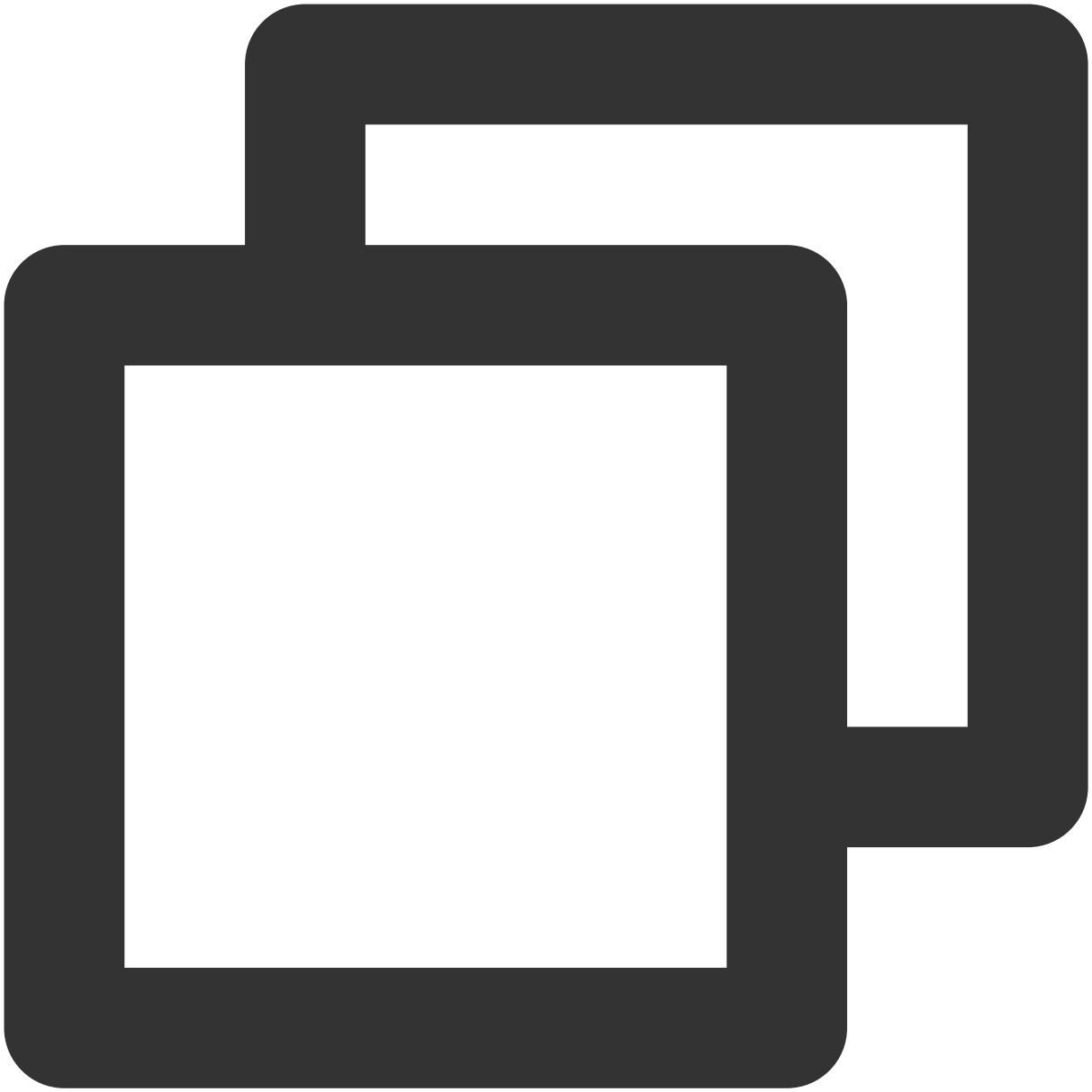
pose：

(1) [0,2]위치, 인체 위치, 카메라를 중심으로 인체 근골격의 3D 위치 xyz

(2) [3,12] 위치, 인체 형태, 10개의 float 수, 표준 SMPL의 서로 다른 mesh 10세트를 베이스로 조합하여 체형 추정

(3)[13] 위치, focal_length, 고정값은 5000

(4) [14,29] 위치, OpenGL 홀로그래피 매트릭스, focal_length를 기반으로 얻은 3D 공간에서 물체를 렌더링하는 홀로그래피 매트릭스입니다. 4X4 홀로그래피 매트릭스는 알고리즘 내에서 계산됩니다.



```
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};
}
```

(5) [30,33] 위치, 접지 데이터, 발이 지면에 닿았는지 여부, 왼쪽 뒤꿈치, 왼쪽 발가락, 오른쪽 뒤꿈치, 오른쪽 발가락 position_x,position_y,position_z：

(1) [0,23] 위치, 인체의 2D 포인트, 위의 이미지1 참고, 2D 포인트의 position_z는 모두 0

(2) [24,47] 위치, 인체의 3D 포인트, 위의 이미지1 참고

rotation

(1) [0,23] 위치, 인체 골격 회전 쿼터니언, 각 쿼터니언의 속성 순서는 wxyz

(2) [25,54] 위치, 손 골격 회전 쿼터니언, 왼손은 15개, 오른손은 15개, 각 쿼터니언의 속성 순서는 wxyz

골격에 따른 명명 방식 및 해당 관계

No.	Bone Names	Bone Names 2
0	"pelvis",	"Hips"
1	"left_hip",	"LeftUpLeg"
2	"right_hip",	"RightUpLeg"
3	"spine1",	"Spine"
4	"left_knee",	"LeftLeg"
5	"right_knee",	"RightLeg"
6	"spine2",	"Spine1"
7	"left_ankle",	"LeftFoot"
8	"right_ankle",	"RightFoot"
9	"spine3",	"Spine2"
10	"left_foot",	""
11	"right_foot",	""
12	"neck",	"Neck"
13	"left_collar",	"LeftShoulder"
14	"right_collar",	"RightShoulder"
15	"head",	"Head"
16	"left_shoulder",	"LeftArm"
17	"right_shoulder",	"RightArm"
18	"left_elbow",	"LeftForeArm"
19	"right_elbow",	"RightForeArm"
20	"left_wrist",	"LeftHand"
21	"right_wrist",	"RightHand"
22	"left_hand"	""
23	"right_hand"	""
25	"left_index1"	IndexFinger1_L
26	"left_index2"	IndexFinger2_L
27	"left_index3"	IndexFinger3_L
28	"left_middle1"	MiddleFinger1_L
29	"left_middle2"	MiddleFinger2_L
30	"left_middle3"	MiddleFinger3_L
31	"left_pinky1"	PinkyFinger1_L
32	"left_pinky2"	PinkyFinger2_L
33	"left_pinky3"	PinkyFinger3_L

34	"left_ring1"	RingFinger1_L
35	"left_ring2"	RingFinger2_L
36	"left_ring3"	RingFinger3_L
37	"left_thumb1"	ThumbFinger1_L
38	"left_thumb2"	ThumbFinger2_L
39	"left_thumb3	ThumbFinger3_L
40	"right_index1"	IndexFinger1_R
41	"right_index2"	IndexFinger2_R
42	"right_index3"	IndexFinger3_R
43	"right_middle1"	MiddleFinger1_R
44	"right_middle2"	MiddleFinger2_R
45	"right_middle3"	MiddleFinger3_R
46	"right_pinky1"	PinkyFinger1_R
47	"right_pinky2"	PinkyFinger2_R
48	"right_pinky3"	PinkyFinger3_R
49	"right_ring1"	RingFinger1_R
50	"right_ring2"	RingFinger2_R
51	"right_ring3"	RingFinger3_R
52	"right_thumb1"	ThumbFinger1_R
53	"right_thumb2"	ThumbFinger2_R
54	"right_thumb3"	ThumbFinger3_R

Audio-to-Expression
Android

최종 업데이트 날짜: : 2023-02-27 12:19:40

기능 설명

이 기능을 사용하면 오디오를 Apple ARKit의 52개 blendshape로 변환할 수 있습니다. 자세한 내용은 ARFaceAnchor를 참고하십시오. blendshape 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

통합 방식

방법1: Tencent Effect SDK 통합

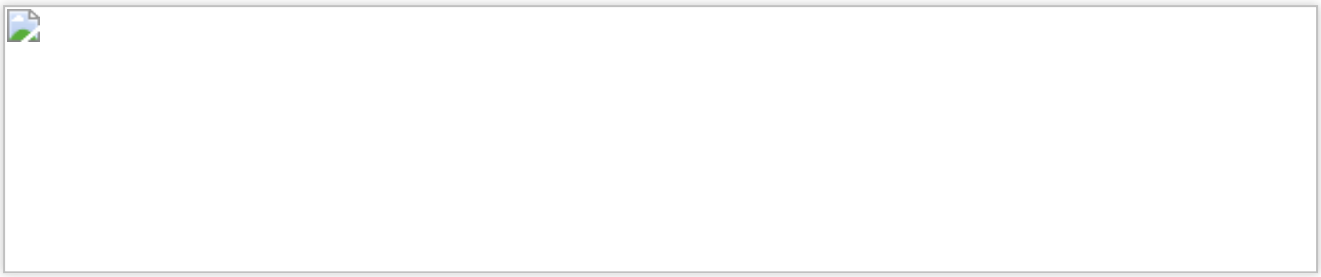
오디오를 표정으로 변환하는 기능은 Tencent Effect SDK에 내장되어 있습니다.
1. Tencent Effect SDK의 전체 버전을 다운로드합니다.
2. SDK를 통합합니다. 자세한 지침은 Tencent Effect SDK 통합하기를 참고하십시오.

방법2: Audio-to-Expression SDK 통합

다른 Tencent Effect 기능이 필요하지 않은 경우 aar 파일이 약 6MB인 Audio-to-Expression SDK를 통합할 수 있습니다. SDK를 얻으려면 저희에게 연락하십시오.

통합 단계

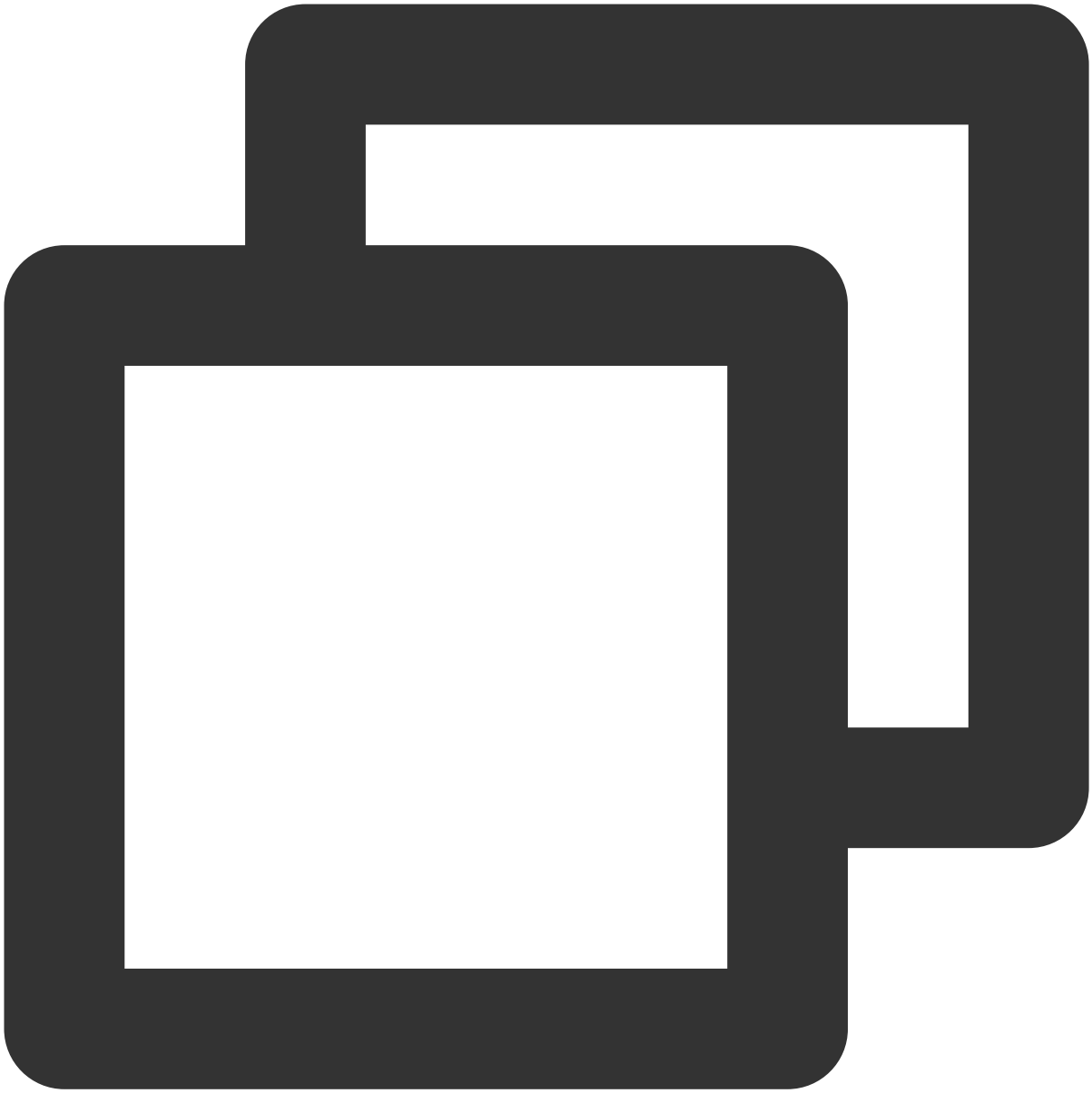
1. **License를 설정합니다**. 자세한 내용은 인증을 참고하십시오.
2. **모델 파일 구성**: assets에서 `context.getFilesDir() + "/my_models_dir/audio2exp"` 와 같은 app의 개인 디렉터리로 모델 파일을 복사합니다. 그 다음 Audio2ExpApi의 `init(String modelPath)` API를 호출하여 `context.getFilesDir() + "/my_models_dir"` 을 전달합니다.
SDK 패키지에서 모델 파일을 찾을 수 있습니다.



API 설명

API	설명
public int Audio2ExpApi.init(String modelPath);	SDK를 초기화합니다. 이 API를 호출할 때 모델 파일의 경로를 전달합니다. 0은
public float[] Audio2ExpApi.parseAudio(float[] inputData);	입력은 오디오이며 1채널이어야 하며 샘플링 레이트는 16K여야 합니다. 배열 `{"eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLo`
public int Audio2ExpApi.release();	리소스를 해제합니다. 더 이상 기능을 사용할 필요가 없을 때 이 API를 호출하

통합 코드 예시



```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.button).setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View view) {
                        TELicenseCheck.getInstance().setTELicense(MainActivity.this
                                @Override
                                public void onLicenseCheckFinish(int errorCode, Str
                                        Log.d(TAG, "onLicenseCheckFinish: errorCode
```

```
                                        if (errorCode == TELicenseCheck.ERROR_OK) {
                                                //license check success
                                                Audio2ExpApi audio2ExpApi = new Aud
                                                int err = audio2ExpApi.init(MainAct
                                                Log.d(TAG, "onLicenseCheckFinish: e
                                                //TODO start record and parse audio
                                        } else {
                                                // license check failed
                                        }
                                    }
                                });
                            }
                        });
    }
```

설명

전체 코드 샘플은 Demo를 참고하십시오.

오디오 녹음은 `com.tencent.demo.avatar.audio.AudioCapturer` 를 참고하십시오.

API 사용 방법에 대한 자세한 내용은 `com.tencent.demo.avatar.activity.Audio2ExpActivity` 및 관련 클래스를 참고하십시오.

iOS

최종 업데이트 날짜: : 2023-02-27 12:14:57

기능 설명

이 기능을 사용하면 오디오를 Apple ARKit의 52개 blendshape로 변환할 수 있습니다. 자세한 내용은 ARFaceAnchor를 참고하십시오. blendshape 데이터를 기반으로 추가 개발을 수행할 수 있습니다. 예를 들어 Unity에 데이터를 전달하여 모델을 구동할 수 있습니다.

통합 방식

방법1: Tencent Effect SDK와 함께 사용

1. 오디오를 표정으로 변환하는 기능은 Tencent Effect SDK에 내장되어 있습니다.
2. Tencent Effect SDK를 다운로드합니다.
3. SDK를 통합합니다. 자세한 지침은 Tencent Effect SDK 통합하기를 참고하십시오.
4. Tencent Effect SDK 내의 Audio2Exp.framework를 프로젝트로 가져오고, 프로젝트의 target->General->Frameworks,Libraries,and Embedded Content에서 Embed & Sign으로 설정합니다.

방법2: Audio-to-Expression SDK 통합

다른 Tencent Effect 기능이 필요하지 않은 경우 Audio2Exp.framework 파일이 약 7MB인 Audio-to-Expression SDK를 통합할 수 있습니다. 프로젝트는 Audio2Exp.framework, YTCommonXMagic.framework 두 개의 동적 라이브러리를 가져오고, 프로젝트의 target->General->Frameworks,Libraries,and Embedded Content에서 Embed & Sign으로 설정합니다.

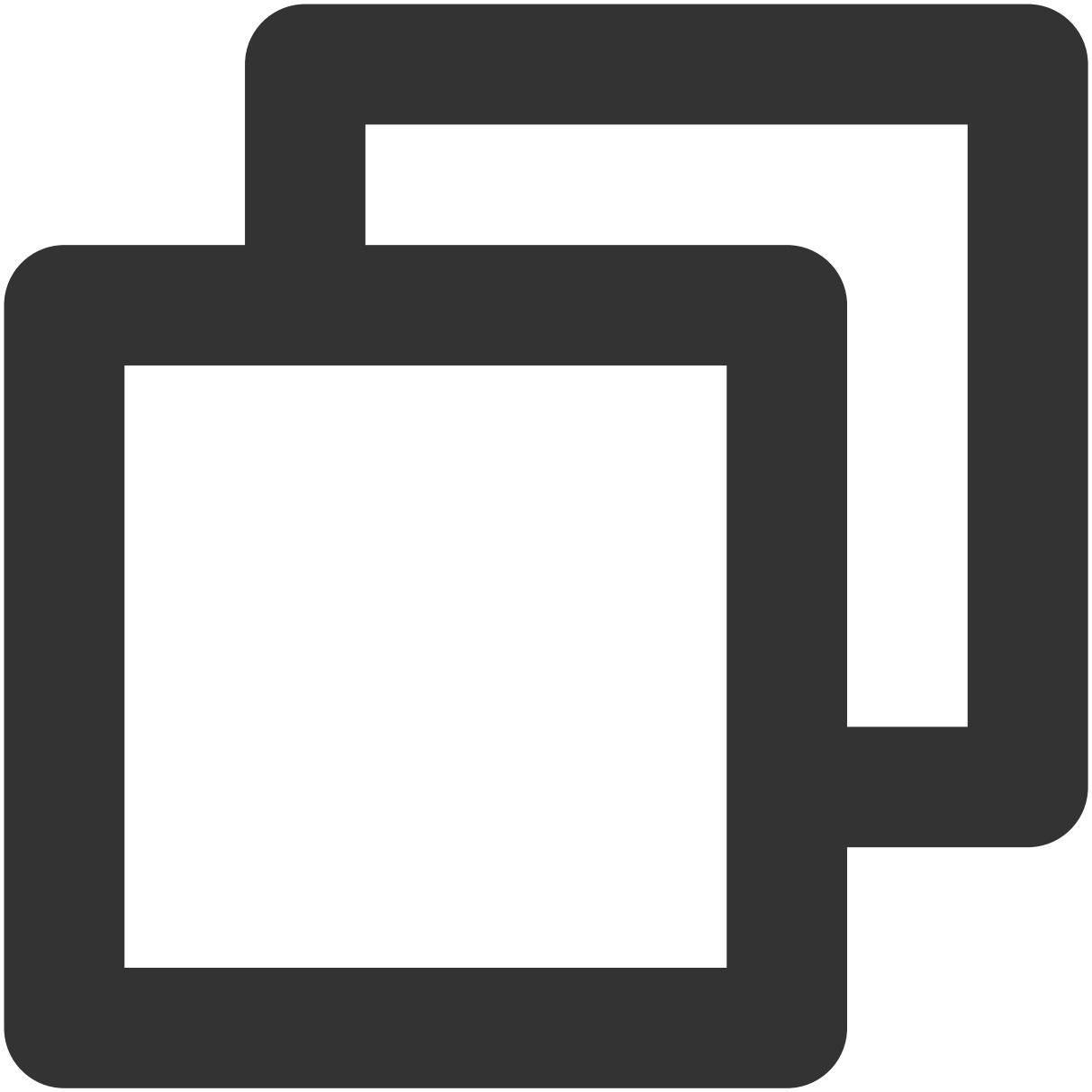
통합 단계

1. License를 설정합니다. 자세한 내용은 인증을 참고하십시오.
2. 모델 파일 구성: 모델 파일 audio2exp.bundle을 프로젝트 디렉터리로 복사합니다. 그 다음 Audio2ExpApi의 initWithModelPath: API를 호출하여 매개변수 audio2exp.bundle"모델 파일이 있는 경로를 전달합니다.

API 설명

API	설명
+ (int)initWithModelPath:(NSString*)modelPath;	SDK를 초기화합니다. 이 API를 호출할 때 모델 파일의 경로를 전달합니다. 0은 초기화
+ (NSArray *)parseAudio:(NSArray *)inputData;	입력은 오디오이며 1채널이어야 하며 샘플링 레이트는 16K여야 합니다. 배열 길이는 26 {"eyeBlinkLeft","eyeLookDownLeft","eyeLookInLeft","eyeLookOutLeft","eyeLookUpLeft",
+ (int)releaseSdk	리소스를 해제합니다. 더 이상 기능을 사용할 필요가 없을 때 이 API를 호출하십시오.

통합 코드 예시



```
// Audio-to-Expression sdk 초기화
NSString *path = [[NSBundle mainBundle] pathForResource:@"audio2exp" ofType:@"bundl
int ret = [Audio2ExpApi initWithModelPath:path];
// 음성 데이터를 52개 blendshape 데이터로 변환
NSArray *emotionArray = [Audio2ExpApi parseAudio:floatArr];
// 릴리스 sdk
[Audio2ExpApi releaseSdk];

// Tencent Effect sdk xmgaic과 결합해 사용
// 해당 리소스를 사용하여 뷰티필터 sdk 초기화
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
```

```
// avatar 소재 로딩
[self.beautyKit loadAvatar:bundlePath exportedAvatar:nil completion:nil];
// 52개의 blendshape 데이터를 뷰티필터 sdk에 전달하여 효과 확인
[self.beautyKit updateAvatarByExpression:emotionArray];
```

참고:

전체 코드 샘플은 Demo를 참고하십시오.

녹음은 `TXCAudioRecorder` 를 참고하십시오.

API 사용 방법에 대한 자세한 내용은 `VoiceViewController` 및 관련 클래스를 참고하십시오.