实时音视频

语音聊天室（含 UI）

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

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

文档目录

语音聊天室（含 UI）
- 组件介绍（TUILiveKit）
- 开通服务（TUILiveKit）
- 快速接入（TUILiveKit）
  - iOS
  - Android
- 互动弹幕（TUILiveKit）
  - iOS
  - Android
- 互动礼物（TUILiveKit）
  - iOS
  - Android
- 客户端API（TUILiveKit）
  - iOS
    - UIKit API
    - Engine API
      - API 概览
  - Android
    - UIKit API
    - Engine API
      - API 概览
- 错误码（TUILiveKit）
- 发布日志（TUILiveKit）
  - iOS
  - Android
- 常见问题（TUILiveKit）
  - iOS
  - Android

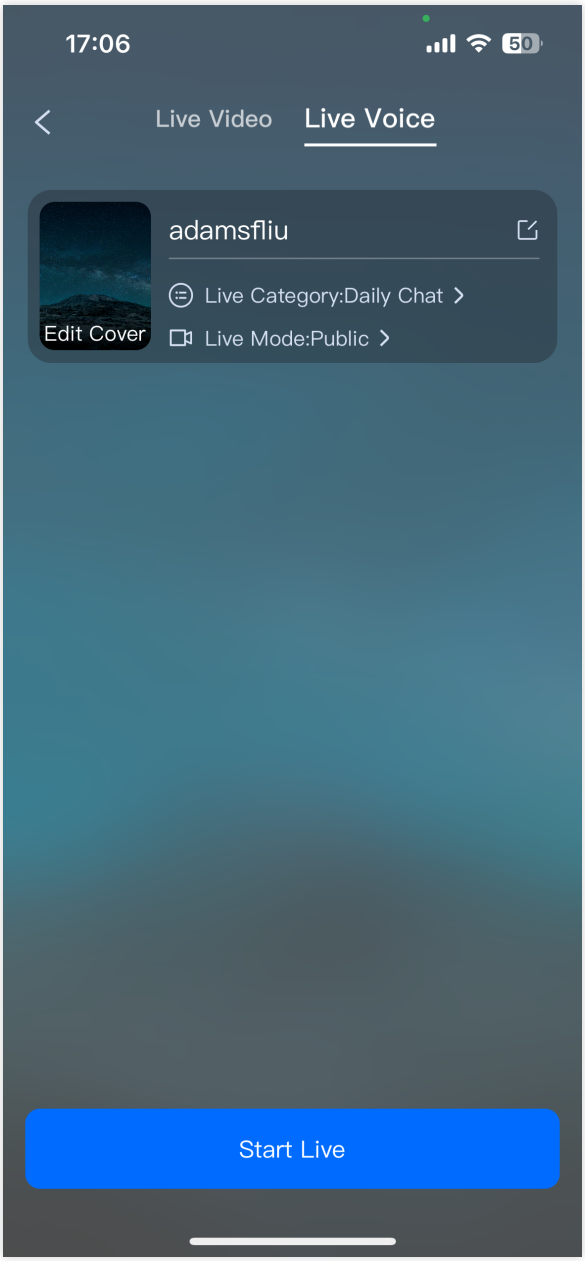
语音聊天室（含 UI）
组件介绍（TUILiveKit）

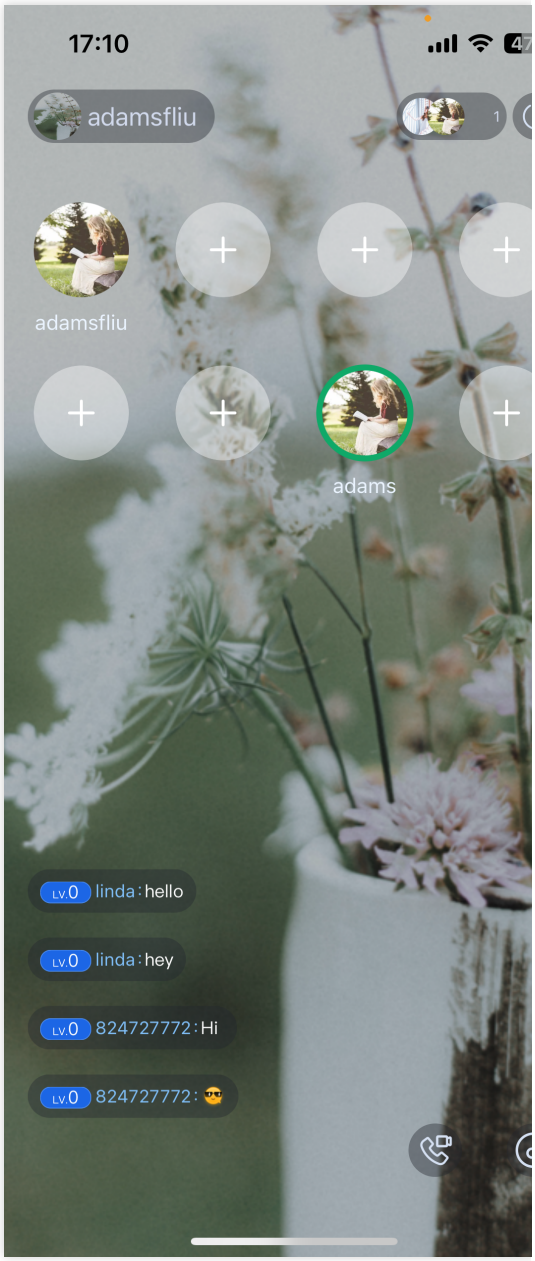
最近更新时间：2024-05-17 11:20:40

组件介绍

TUILiveKit 是一款适用于社交娱乐、游戏互动、实时合唱，语音电台等视频互动直播和语音聊天场景的产品，通过集成该产品，仅需三步，30分钟内就可以为您的应用添加语聊房间、互动连麦、送礼、房间管理，麦位管理等功能，快速上线业务。基本功能展示如下图：

语音直播预览页	语音直播主播页





支持平台

平台	Android	iOS
是否支持		
支持语言/框架	Java Kotlin	Swift Objective-C

功能介绍

基本功能	高级功能	功能优势

语音聊天室 直播观看 观众连麦 麦位管理 查看观众列表 成员管理：拉黑禁言	聊天弹幕 心动点赞 互动礼物（全屏礼物） 音效&变声	**完善的 UI 交互** **专业级直播 3A 算法，更好的音质** **丰富的 REST API**

在线体验

平台	Android	iOS
demo 集成	Github: Andorid	Github: iOS

交流与反馈

如果您有任何需求或反馈，可以联系：info_rtc@tencent.com。

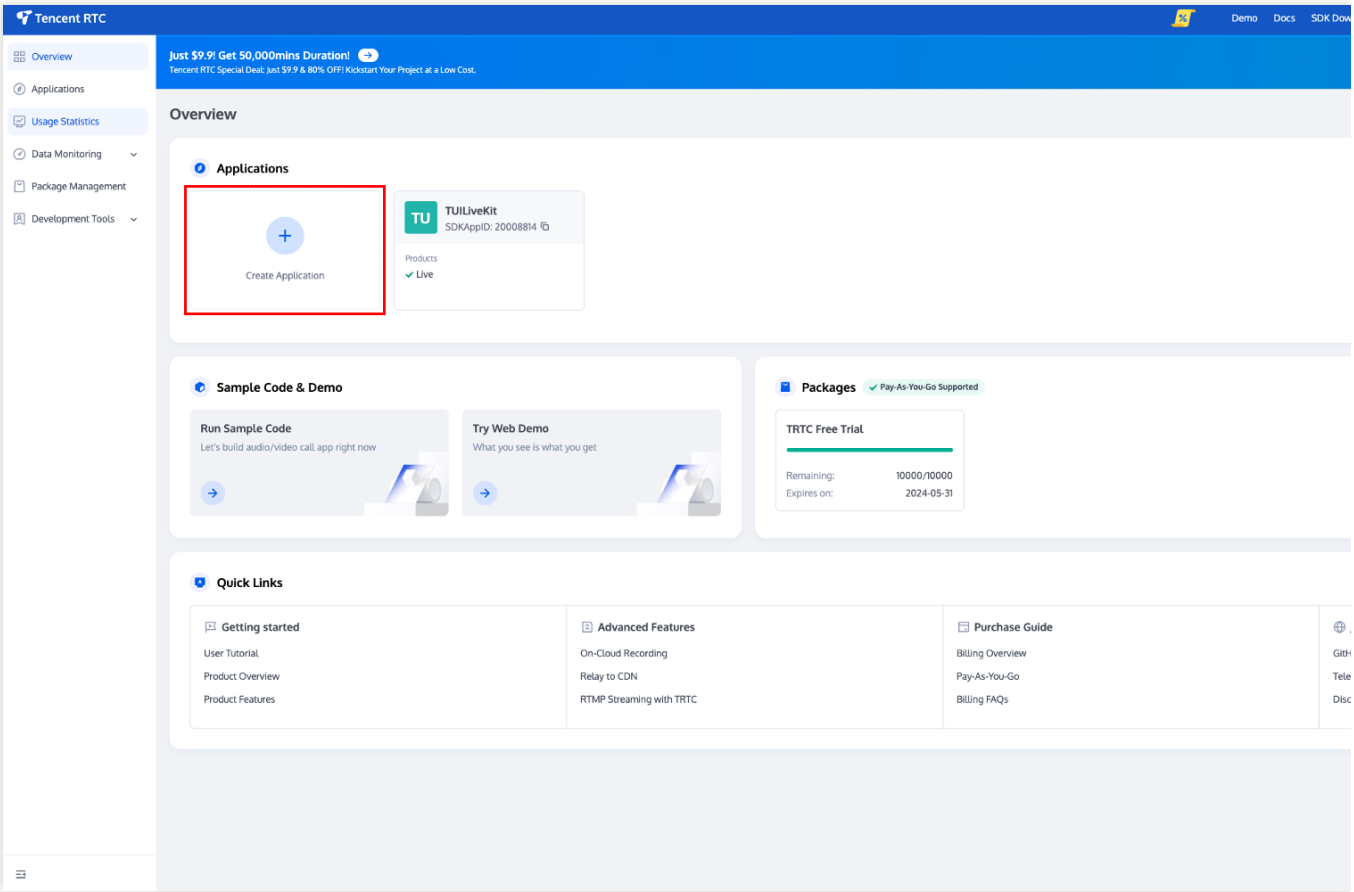
开通服务（TUILiveKit）

最近更新时间：2024-07-03 17:01:13

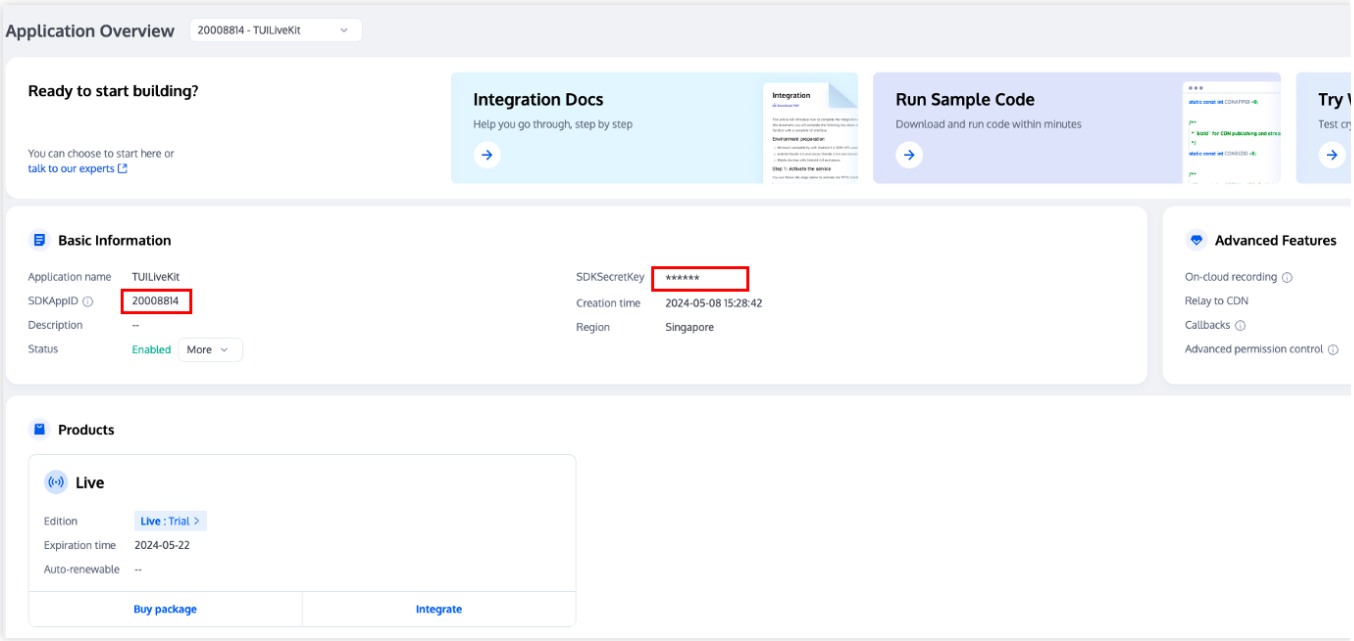
开通体验服务

为了您能更好地体验互动直播 Live 的功能，我们免费为每个 SDKAppID 提供了14天体验版（体验版不额外赠送通话时长），每个 SDKAppID 可免费体验2次，每次有效期均为14天，同时一个账号下所有 SDKAppID 的体验总次数为10次。

1. 登录到 实时音视频 TRTC 控制台，点击 **Create Application**。



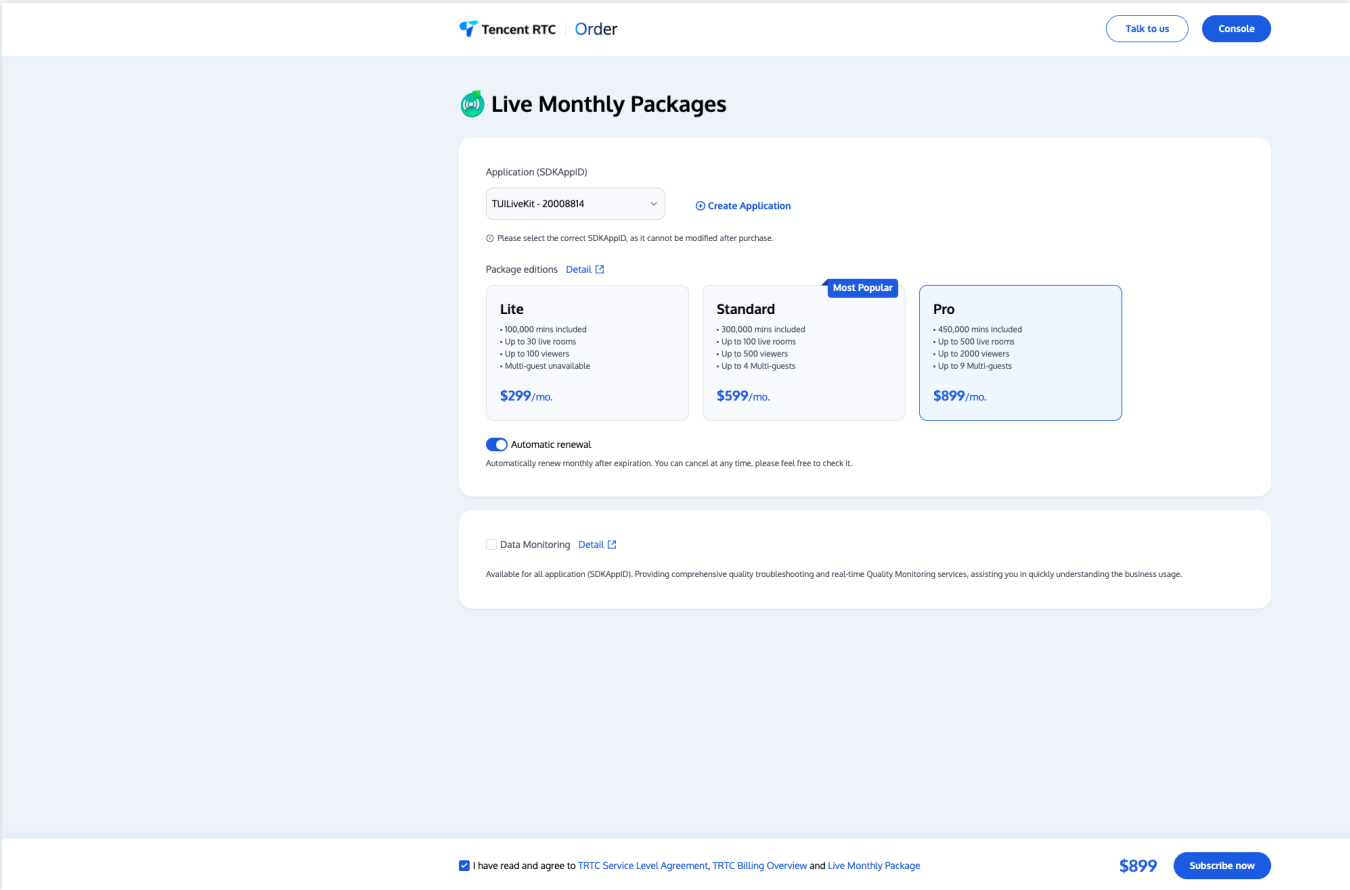
2. 在创建弹窗中输入**应用名称**，选择 **Live**， 并选择合适的 **Region**，点击 **Create**，即可完成体验版的创建。
3. 创建完成后，您就可以在当前页面查看应用的详细信息，比如应用的 `SDKAppID` 、 `SDKSecretKey` 等等信息，可前往 TRTC 控制台 查看应用版本信息，并可参见 快速接入（TUILiveKit） 进行集成。



购买正式版

在正式上线前，您需要购买正式版的 Live 包月套餐，Live 包月套餐的价格和功能对比详情可参见 Live 包月套餐计费说明，套餐包购买流程如下：

1. 访问 Live 购买页面，选择需要购买的应用（SDKAppID）及版本，**同时建议您开启自动续期以避免影响业务使用**。确认购买信息后，勾选协议内容并点击 **Subscribe now**。



说明：

针对语音聊天室业务，建议您购买 Pro 版本，支持的麦位更多，玩法更灵活！

2. 前往订单确认页确认商品信息，并完成支付即可。

快速接入（TUILiveKit）
iOS

最近更新时间：2024-08-09 22:25:01

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频或语音直播功能。

环境准备

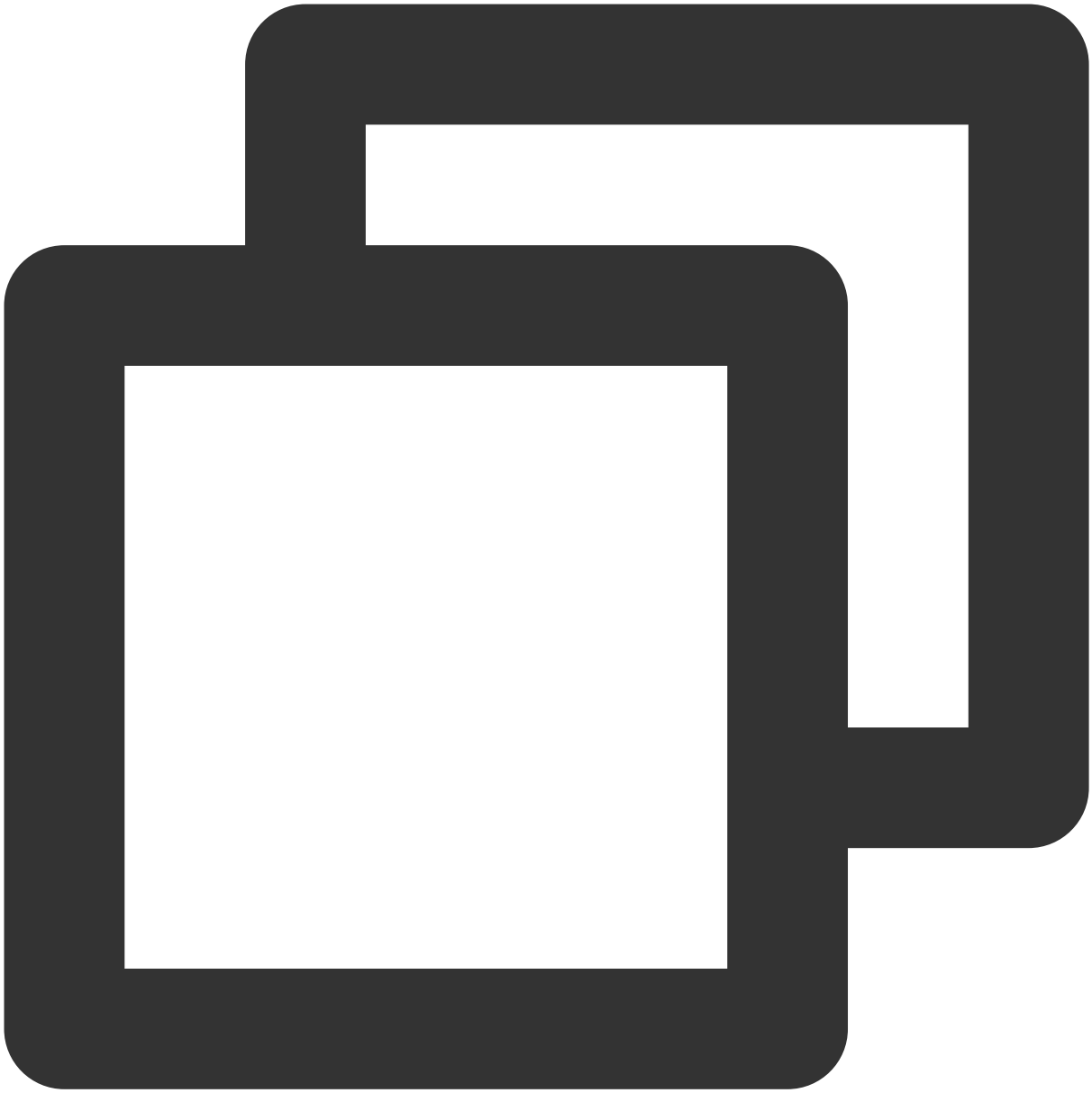
Xcode 15 及以上
iOS 13.0 及以上
CocoaPods环境安装，点击查看。
如果您的接入和使用中遇到问题，请参见 常见问题。

步骤一：开通服务

在使用腾讯云提供的音视频服务前，您需要前往控制台，为应用开通音视频服务。具体步骤请参见 开通服务（TUILiveKit）。

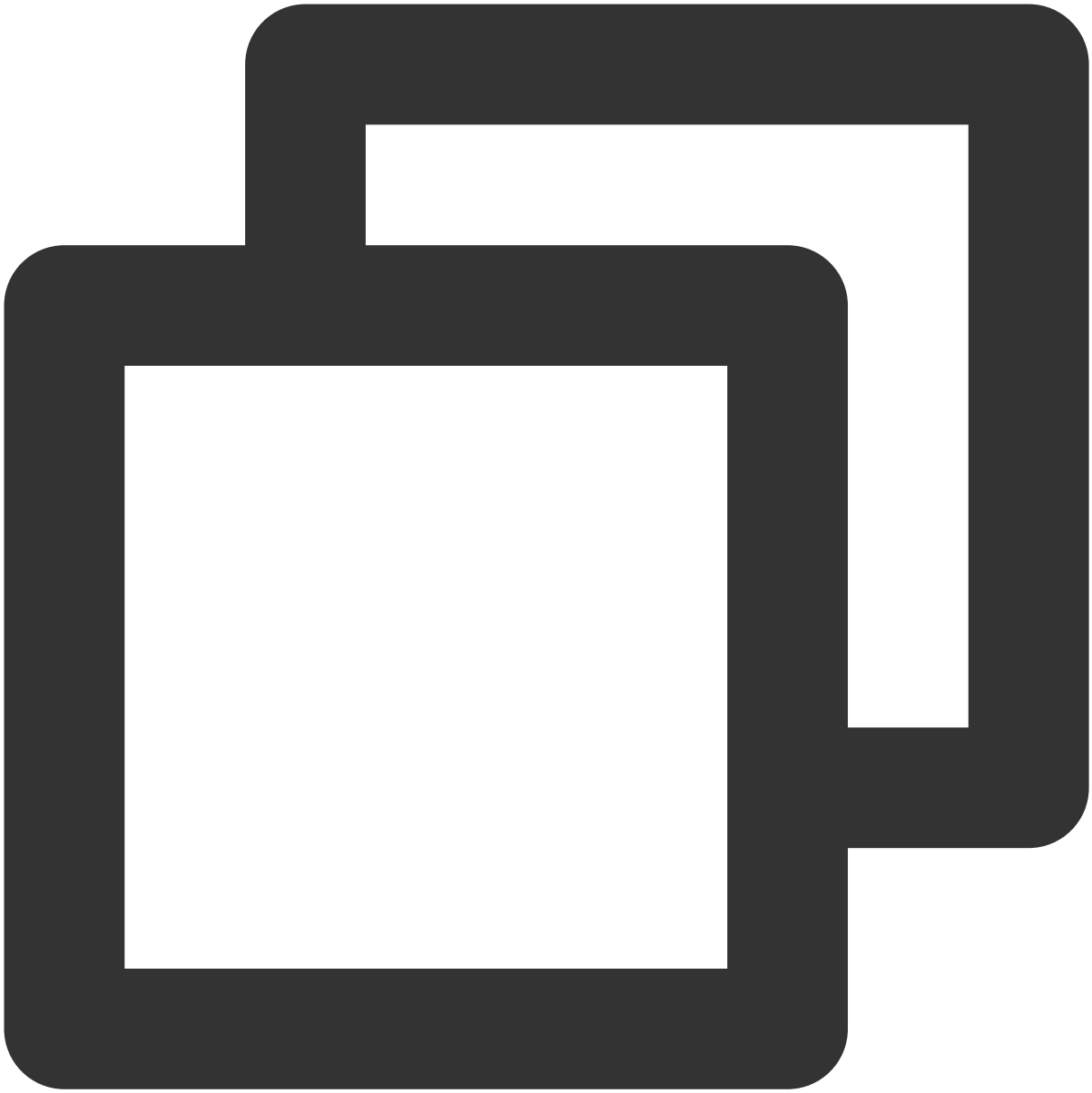
步骤二：导入 TUILiveKit 组件

使用 CocoaPods 导入组件，如果您遇到问题，请先参见 环境准备。导入组件具体骤如下：
1. 请在您的 `Podfile` 文件中添加 `pod 'TUILiveKit'` 依赖，如果您遇到任何问题，请参见 Example 工程。
Swift



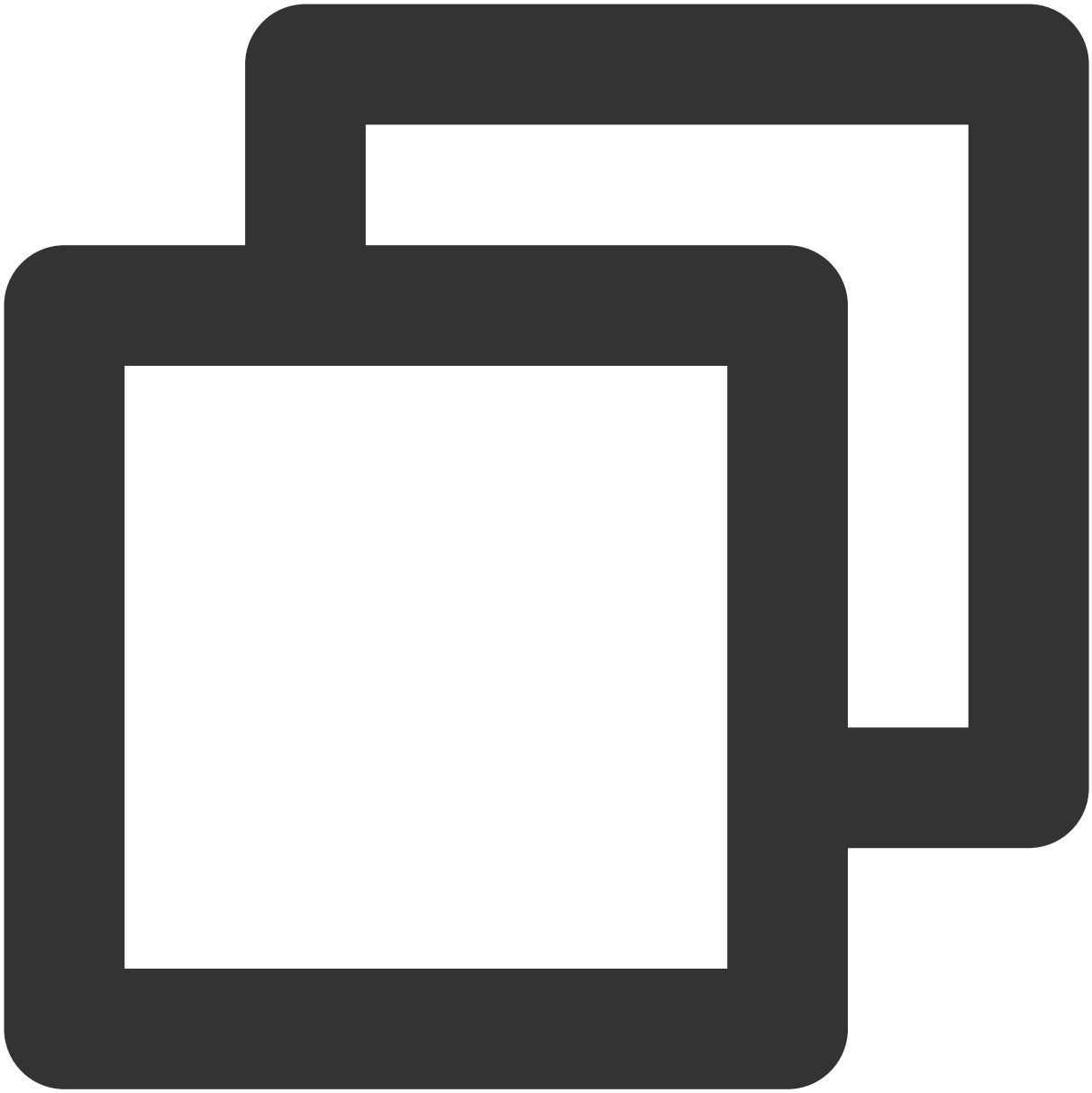
```
target 'xxxx' do
  ...
  ...
  pod 'TUILiveKit'
end
```

如果您没有 `Podfile` 文件，首先终端 `cd` 到 `xxxx.xcodeproj` 目录，然后通过以下命令创建：



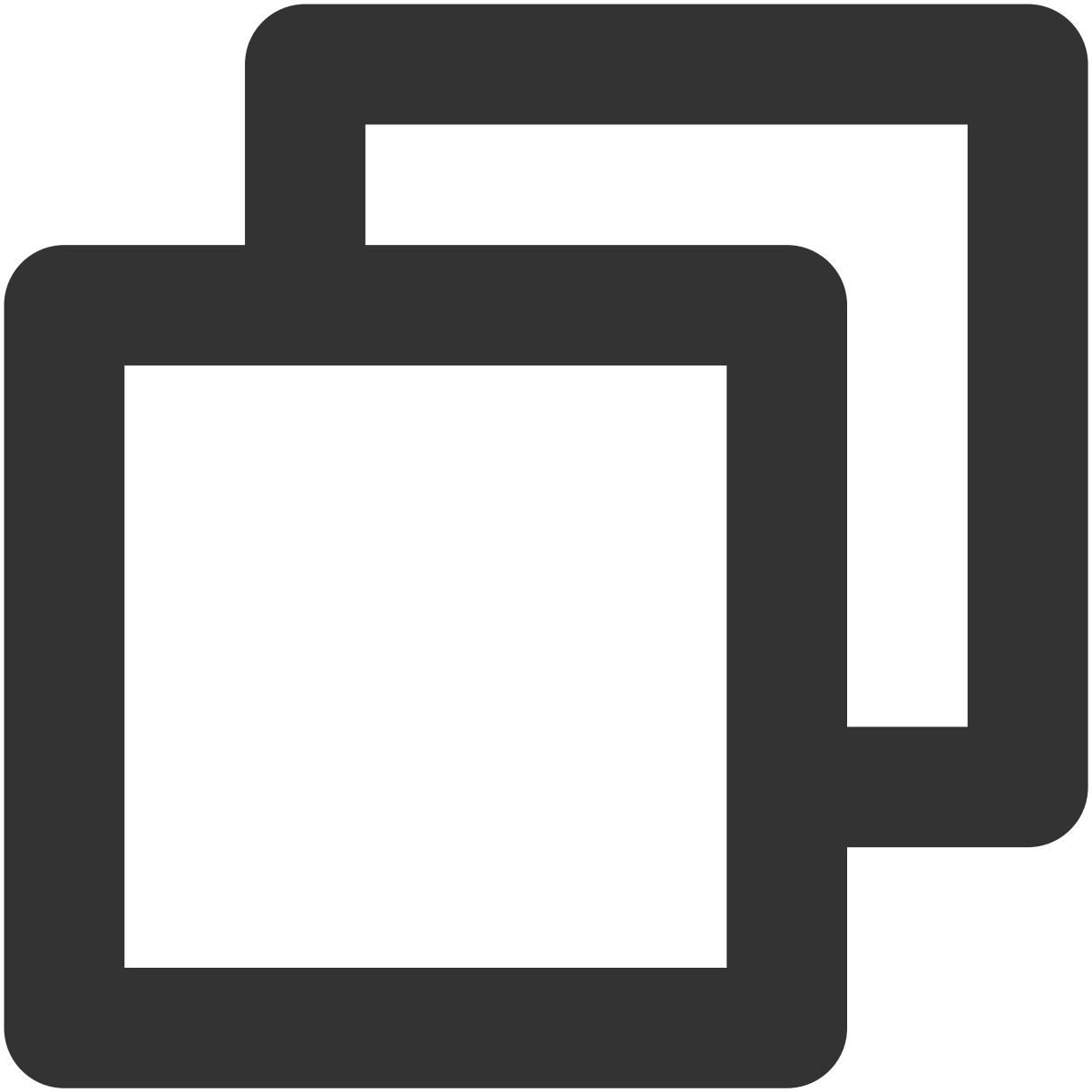
```
pod init
```

2. 在终端中，首先 `cd` 到 `Podfile` 目录下，然后执行以下命令，安装组件。



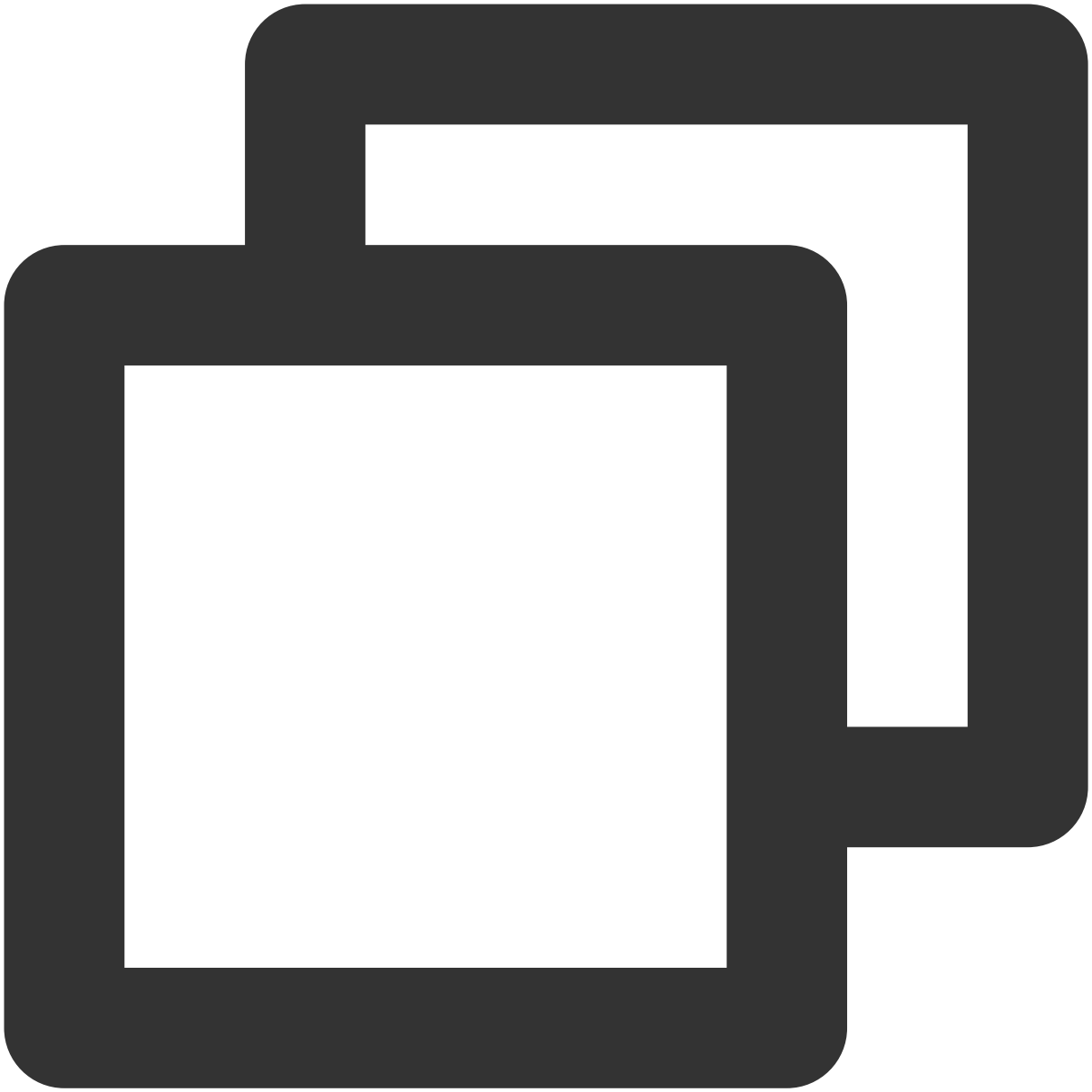
```
pod install
```

如果无法安装 TUILiveKit 最新版本，可以先删除**Podfile.lock**和**Pods**。然后执行以下命令更新本地的 CocoaPods 仓库列表。



```
pod repo update
```

之后执行以下命令，更新组件库的 Pod 版本。



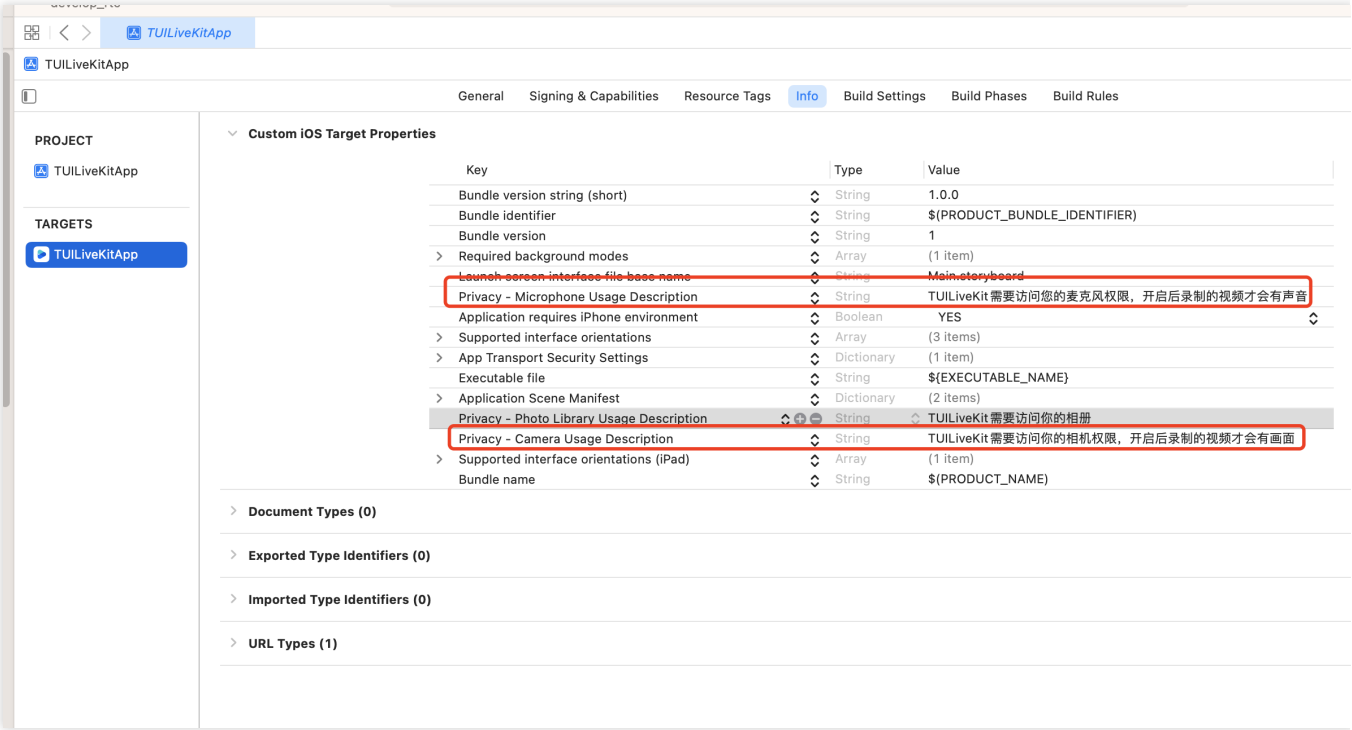
```
pod update
```

3. 可以先编译运行一下，如果遇到问题，请参见 常见问题。问题如果依然无法解决，可以先去跑一下我们的 Example工 程。您在接入和使用过程中遇到的任何问题，欢迎给我们 反馈。

步骤三：工程配置

使用音视频功能，需要授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

```
<key>NSCameraUsageDescription</key>
<string>TUILiveKit需要访问你的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUILiveKit需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```

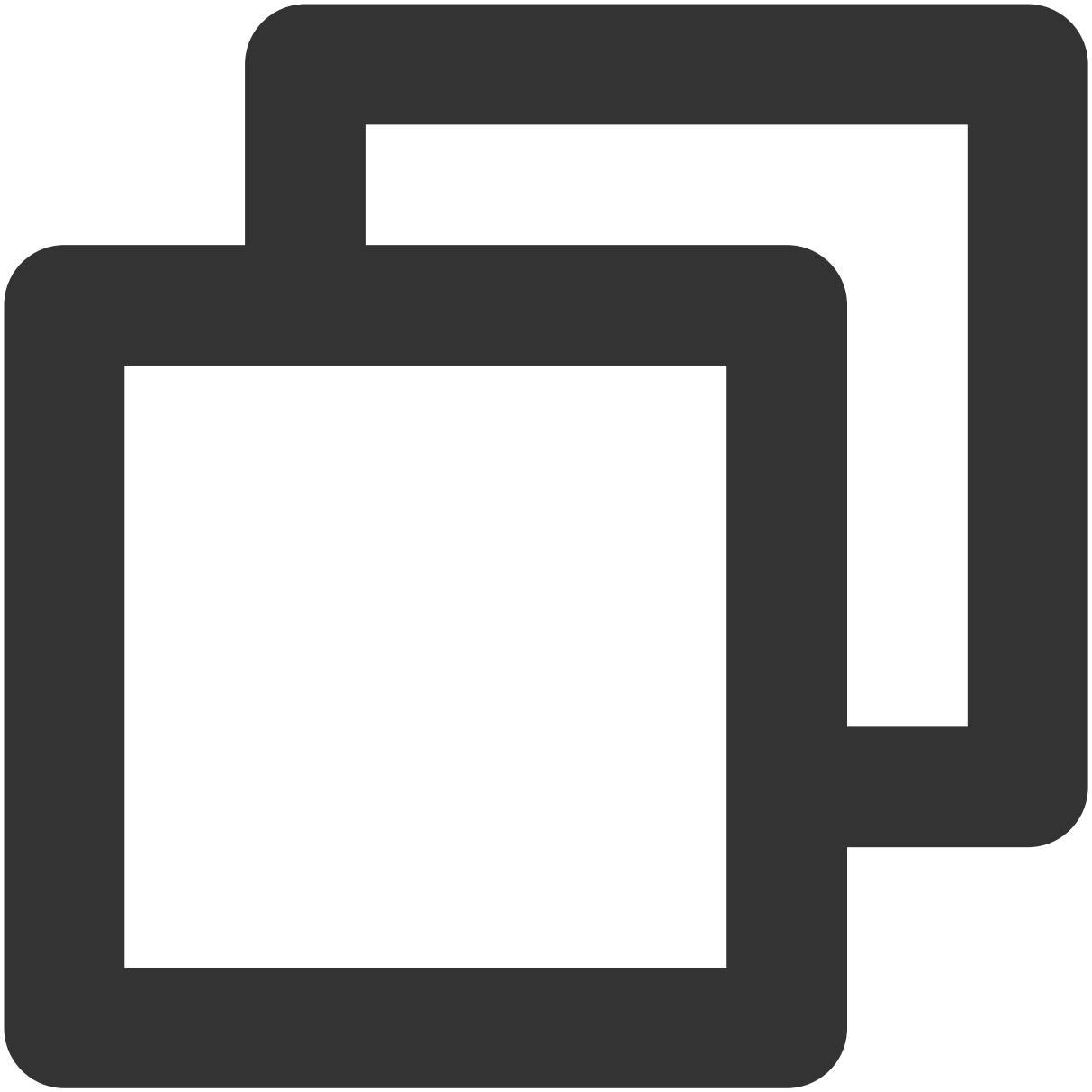


步骤四：登录

在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤非常关键，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

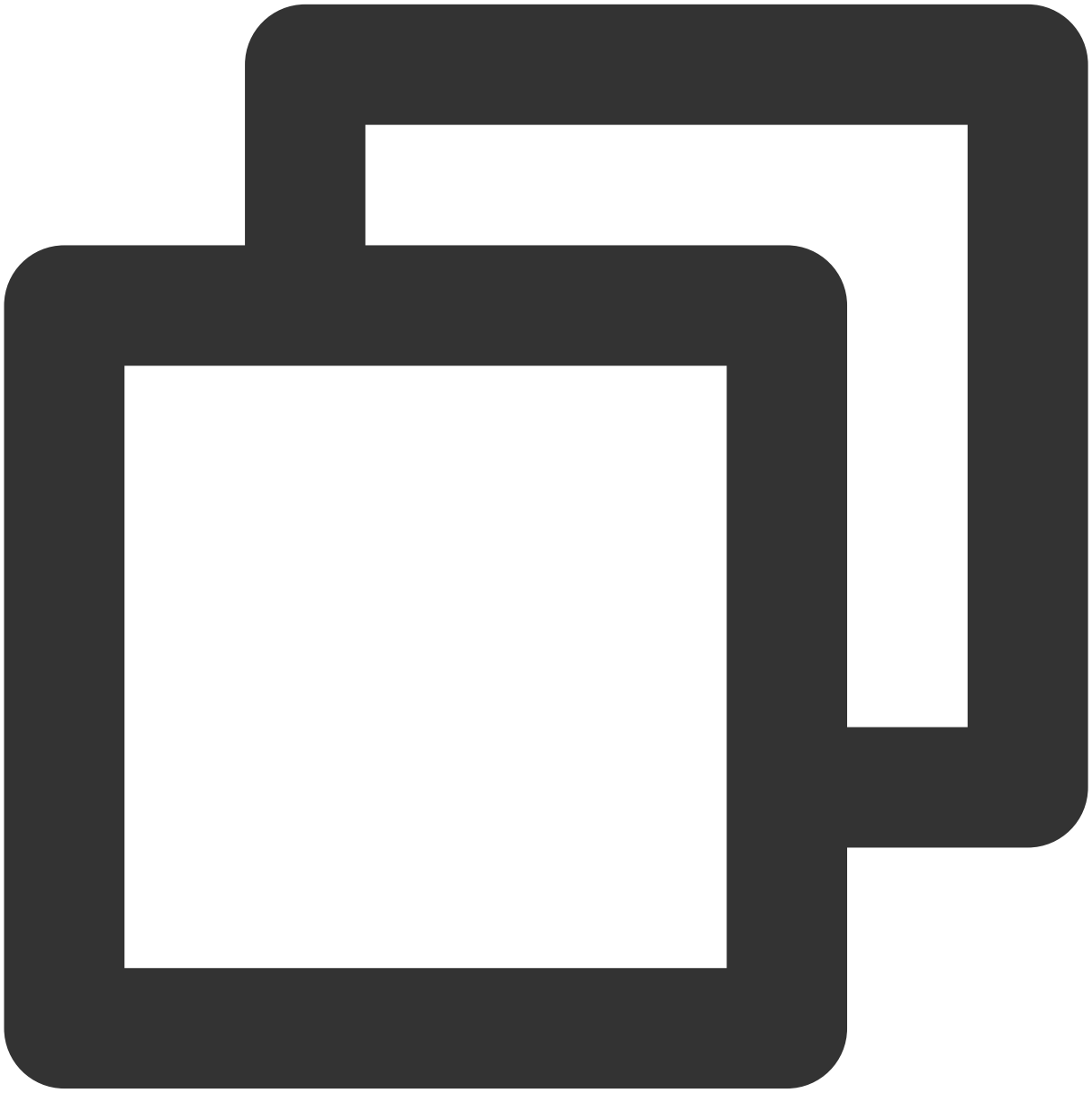
Swift

Objective-C



```
//
//  AppDelegate.swift
//


import TUICore
func application(_ application: UIApplication, didFinishLaunchingWithOptions launch

    TUILogin.login(1400000001,                  // 请替换为步骤一取到的 SDKAppID
            userID: "denny",                    // 请替换为您的 UserID
            userSig: "xxxxxxxxxxx") {           // 您可以在控制台中计算一个 UserSig 并填在这
```

```
        print("login success")
    } fail: { (code, message) in
        print("login failed, code: \\(code), error: \\(message ?? "nil")")
    }

    return true
}
```



```
//
//  AppDelegate.m
//
```

```
#import <TUICore/TUILogin.h>
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    [TUILogin login:1400000001          // 请替换为步骤一取到的 SDKAppID
             userID:@"denny"            // 请替换为您的 UserID
            userSig:@"xxxxxxxxxxx"      // 您可以在控制台中计算一个 UserSig 并填在这个位置
               succ:^{
        NSLog(@"login success");
    } fail:^(int code, NSString * _Nullable msg) {
        NSLog(@"login failed, code: %d, error: %@", code, msg);
    }];
    return YES;
}
```

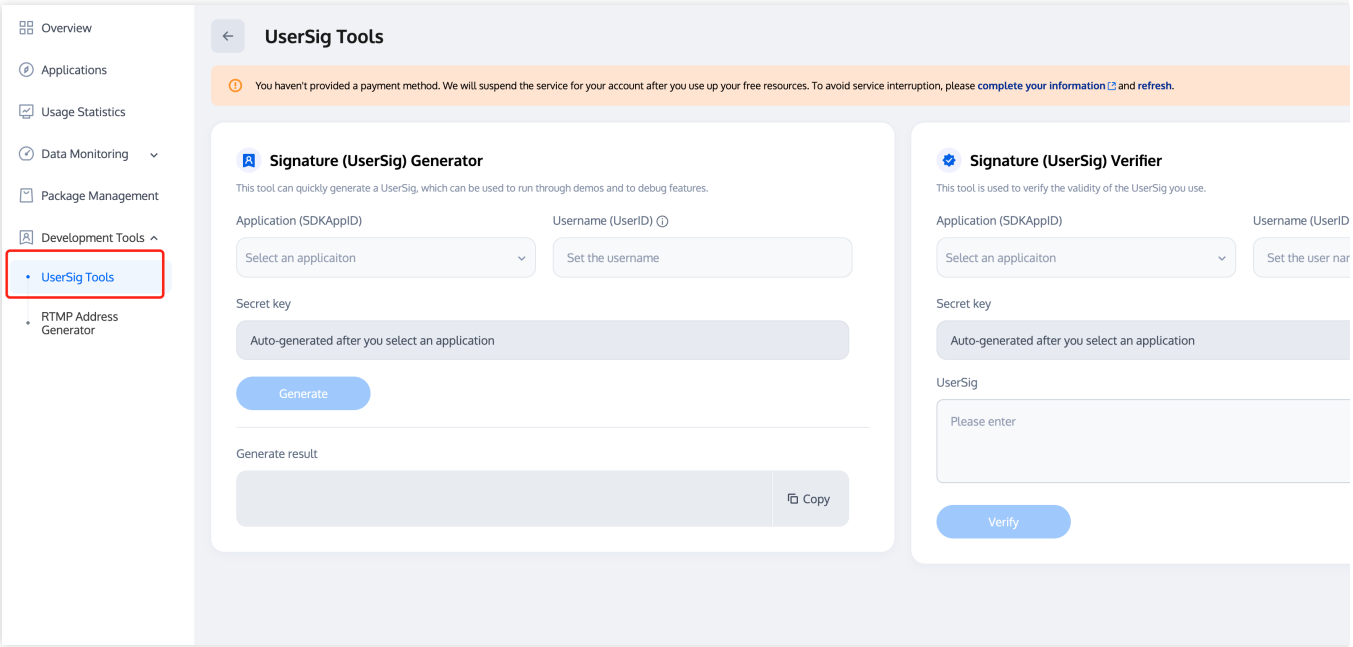
参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

SDKAppID：在 步骤一 中的最后一步中您已经获取到，这里不再赘述。

UserID：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

UserSig：使用 开通服务（TUILiveKit）获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台左侧项目栏中的 UserSig 工具，创建一个临时可用的 UserSig。



更多信息请参见 UserSig 相关。

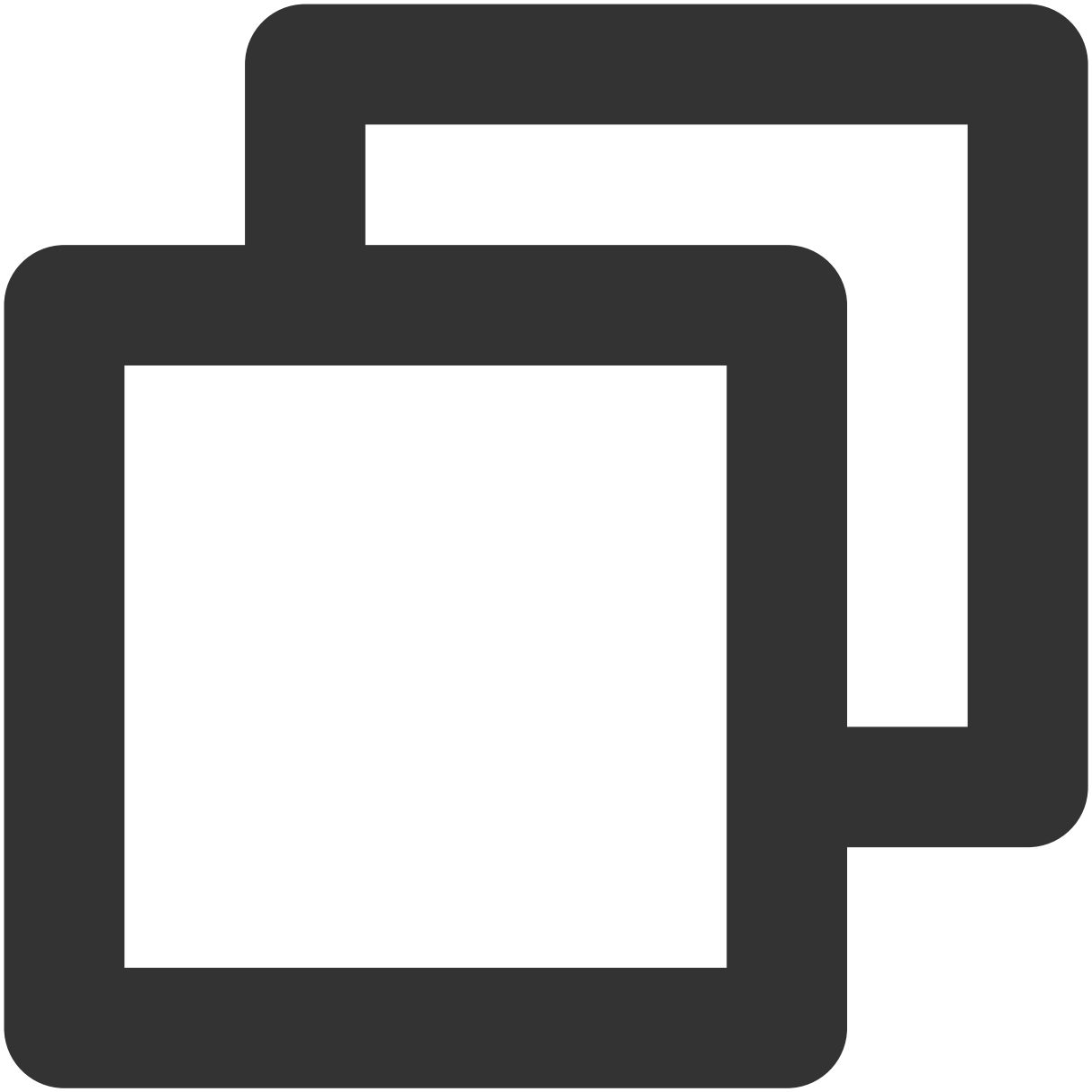
注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误。
userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 userSig。
userSig 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userID。
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TULiveKit 组件时向您的服务器请求实时计算出的 userSig。

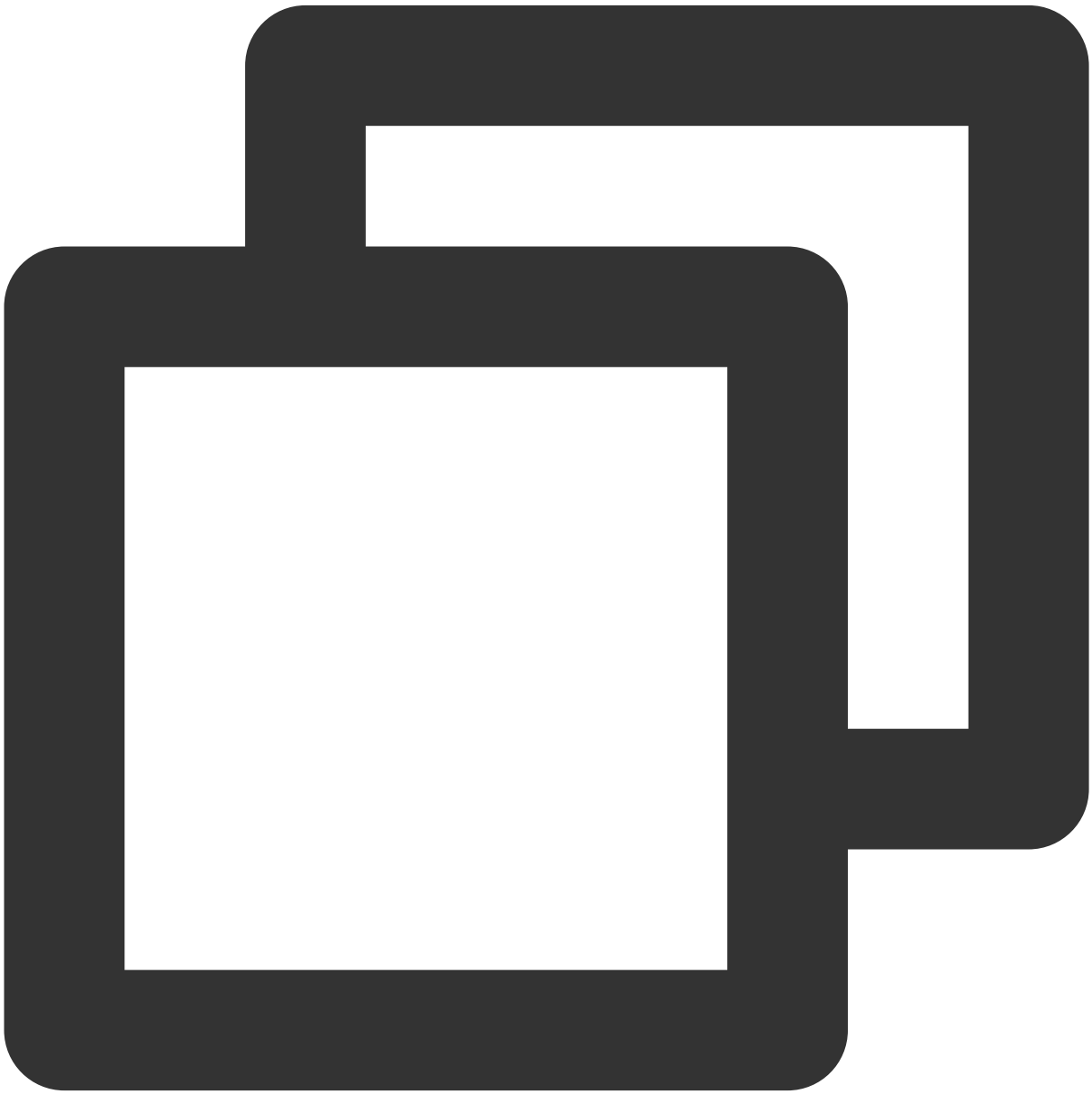
步骤五：进入语音聊天室预览画面

注意：
请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。
通过加载 TUILiveKit 的TUIVoiceRoomViewController页面，就可以拉起预览画面，点击“开始直播”即可创建语音聊天室。
Swift
Objective-C



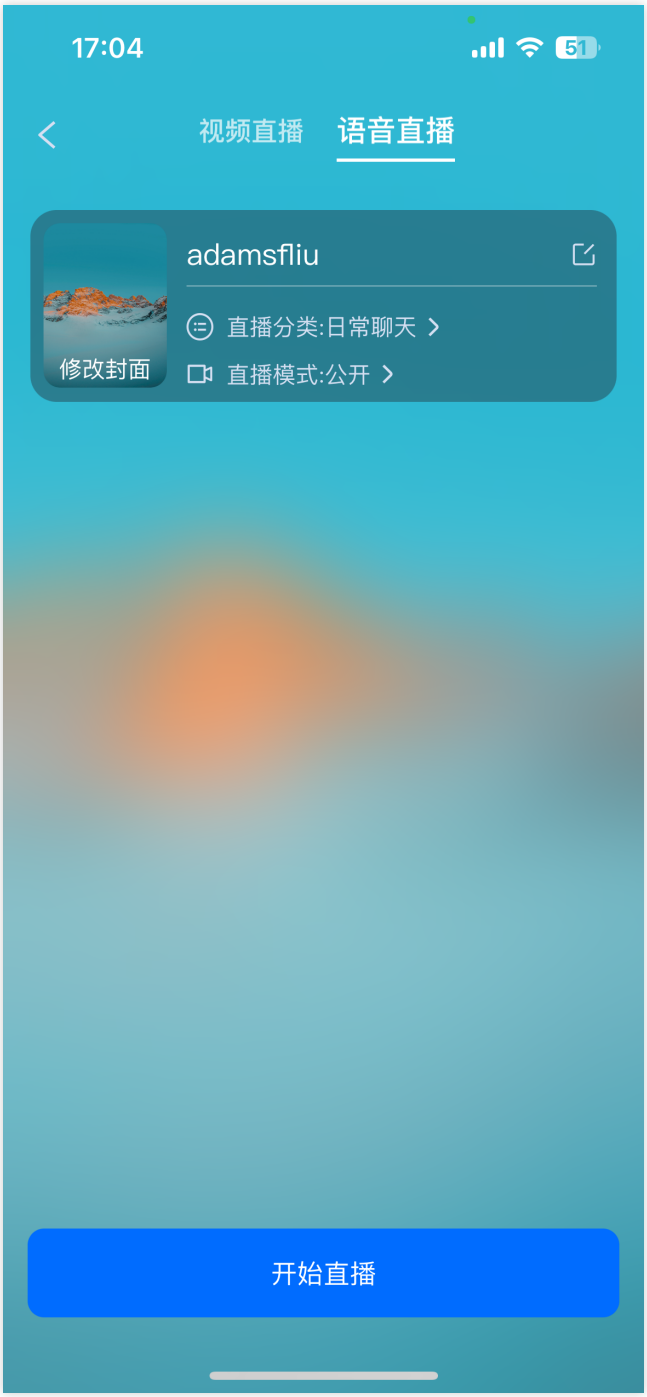
```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    //RoomId可以自定义生成，建议使用 LiveIdentityGenerator.shared.generateId(TUILogin.g
    let roomId = "123666";
```

```
    // 进入语音聊天室预览画面
    var roomParams = RoomParams()
    // 房间最大麦位数，默认套餐包支持的最大麦位数量
    roomParams.maxSeatCount = 0
    roomParams.seatMode = .applyToTake

    let voiceRoomController = TUIVoiceRoomViewController(roomId: roomId, behavior:
    self.navigationController?.pushViewController(voiceRoomController, animated: tr
}
```

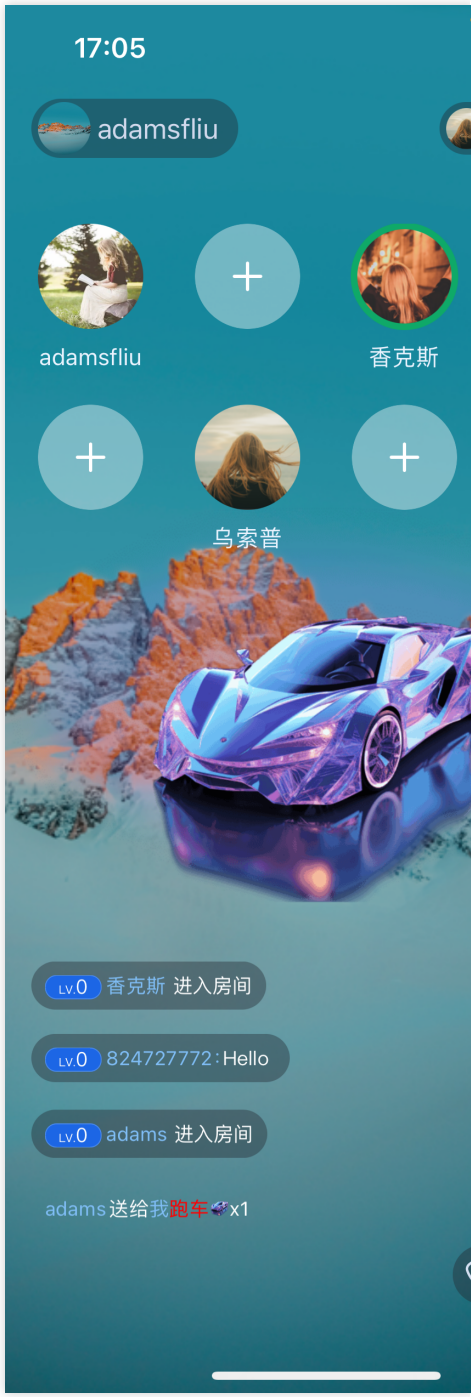


```
//
```

```
//  MainViewController.m
//

#import <TUILiveKit/TUILiveKit-Swift.h>
#import <RTCRoomEngine/RTCEngine.h>

- (void)buttonTapped:(UIButton *)sender {
    //RoomId可以自定义生成，建议使用 [LiveIdentityGenerator.shared generateId:[TUILogir
    NSString *roomId = @"123666";

    // 进入语音聊天室预览画面
    RoomParams *roomParams = [[RoomParams alloc] initWithMaxSeatCount:0  // 房间最大
                                                                seatMode:TUISeatModeAp
    TUIVoiceRoomViewController *voiceRoomController = [[TUIVoiceRoomViewController


    [self.navigationController pushViewController:voiceRoomController animated:true
}
```



语音聊天室预览画面



语音聊天室房间内画面

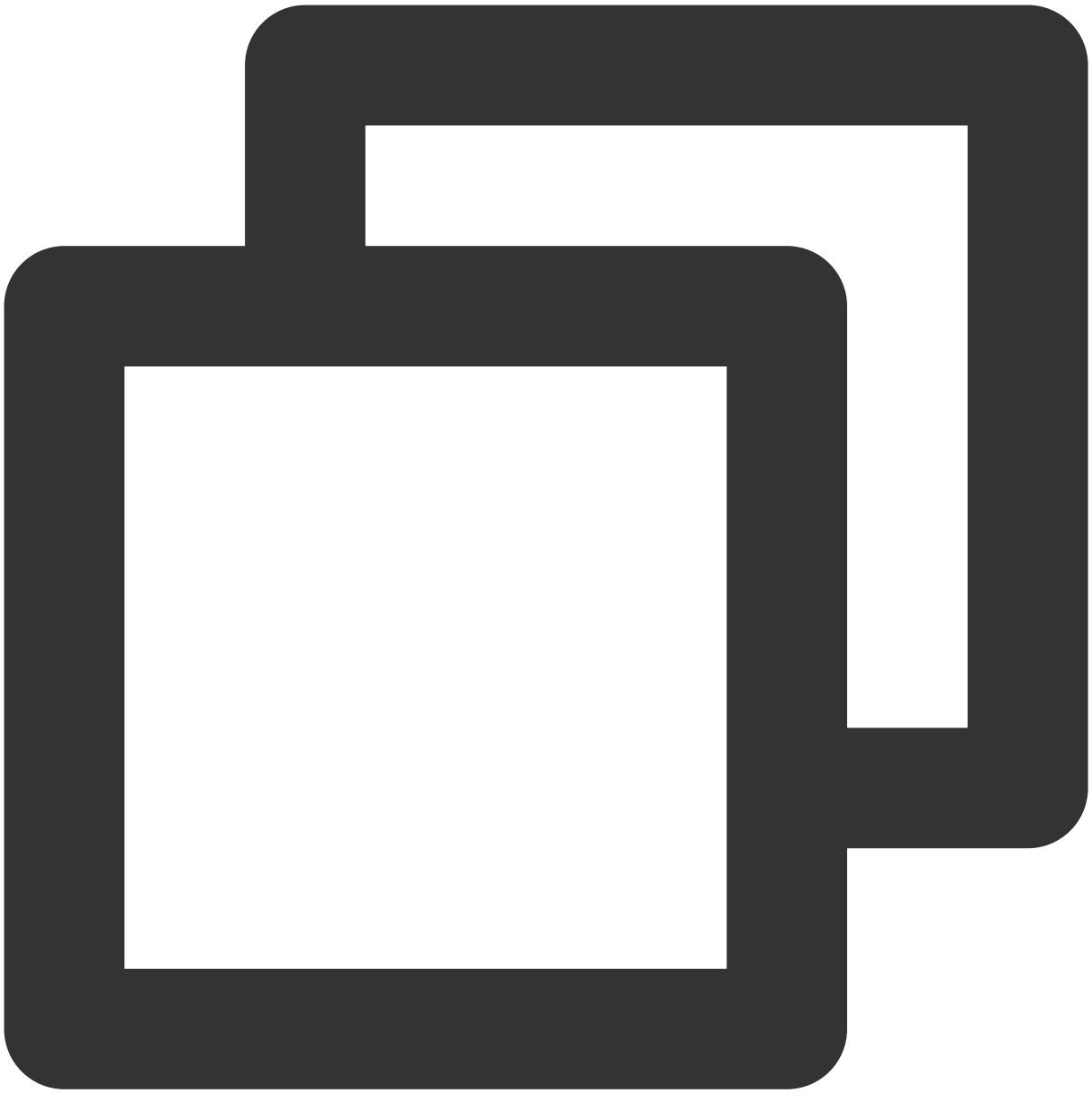
步骤六：拉取房间列表

注意：

请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。

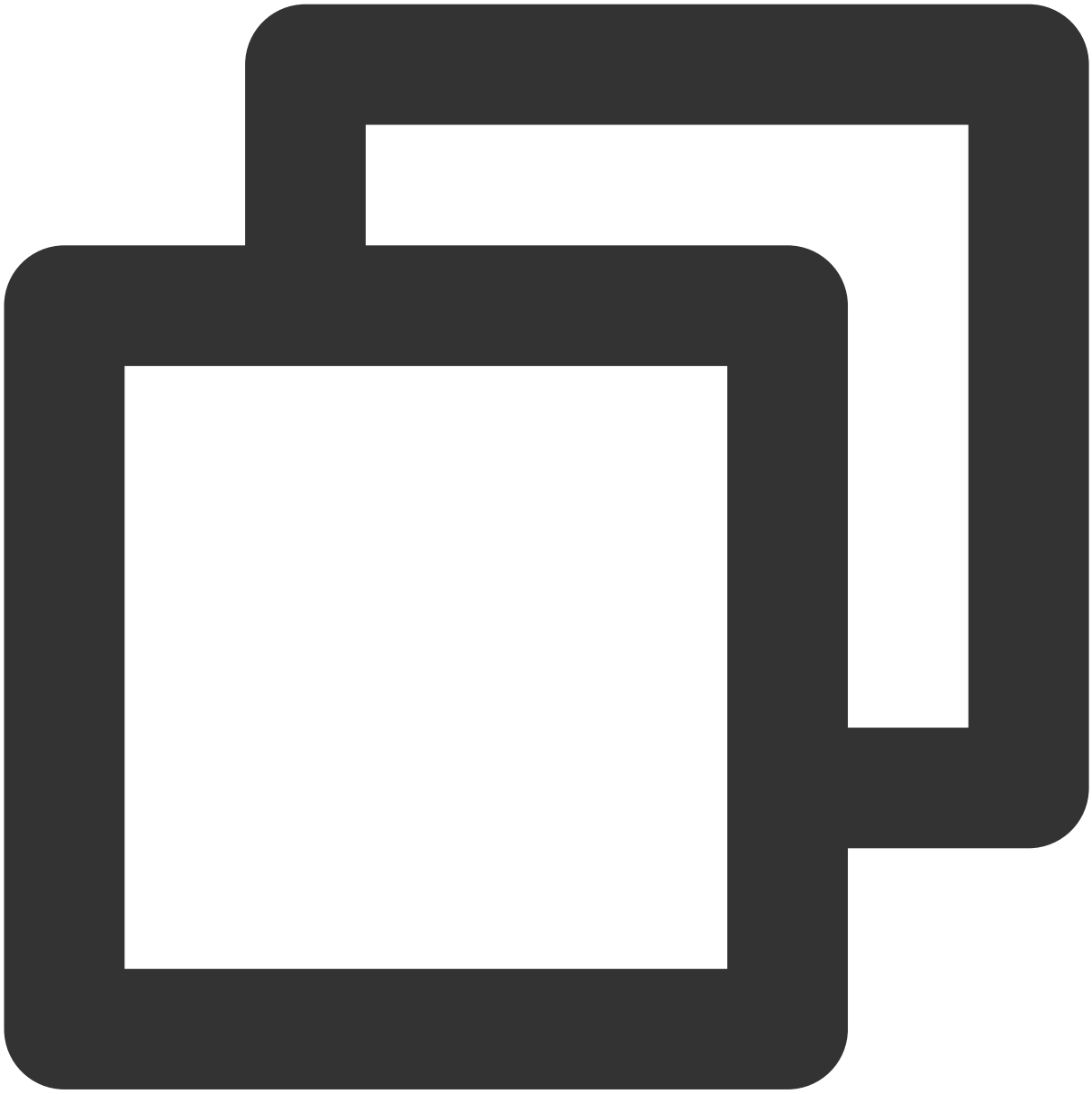
Swift

Objective-C



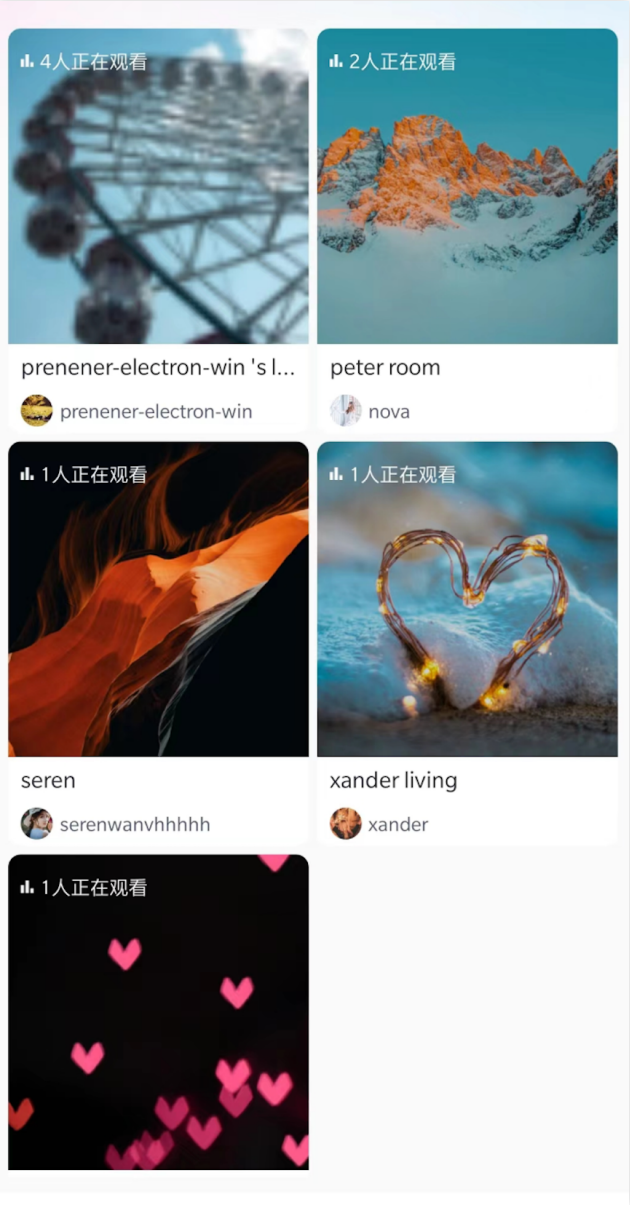
```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    // 进入房间列表
```

```
    let liveListViewController = TUILiveListViewController()
    self.navigationController?.pushViewController(viewController, animated: true)
}
```



```
//
//  MainViewController.m
//

#import <TUILiveKit/TUILiveKit-Swift.h>

- (void)buttonTapped:(UIButton *)sender {
```

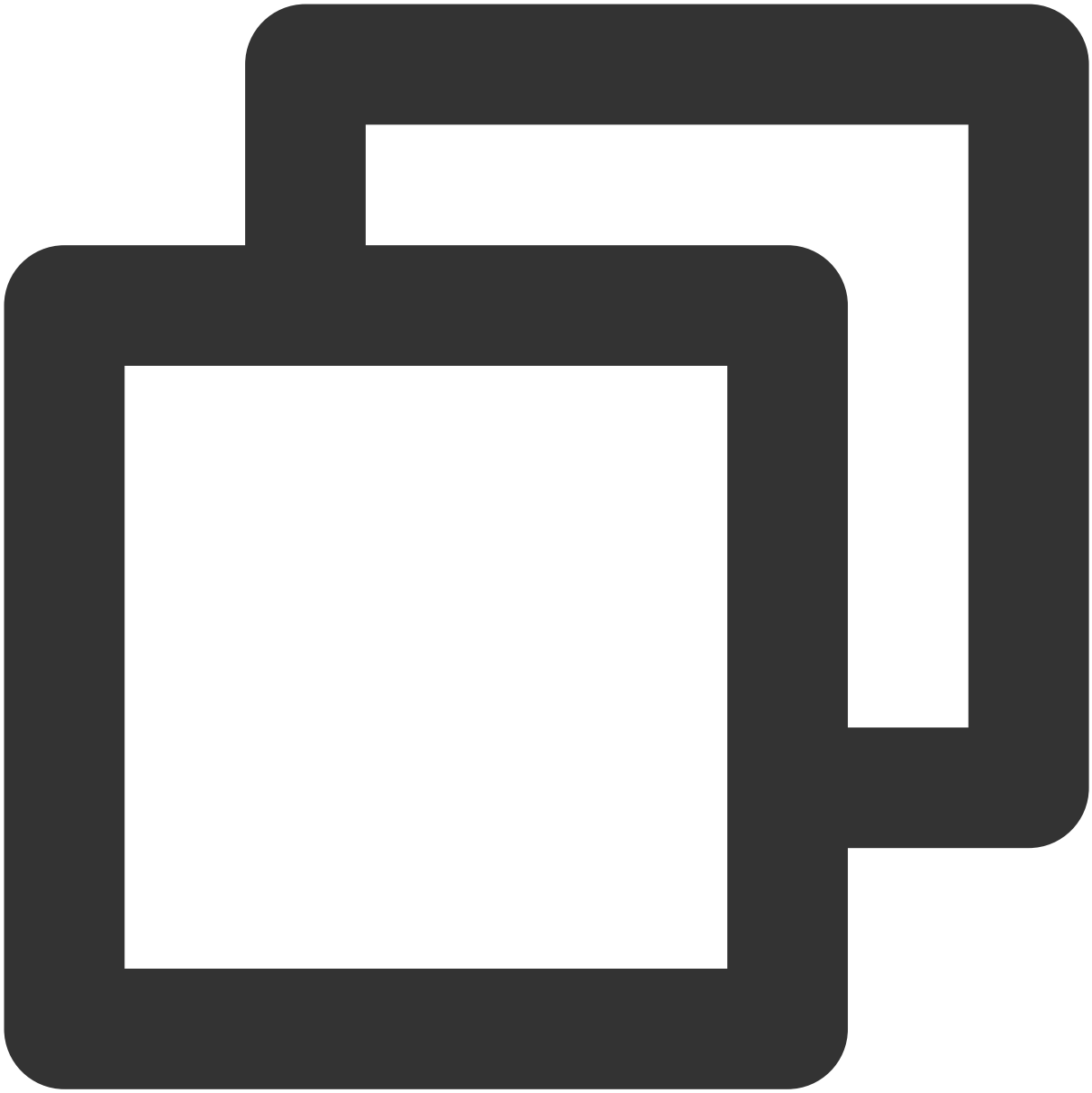
```
    // 进入房间列表
    TUILiveListViewController *liveListViewController = [[TUILiveListViewController
    [self.navigationController pushViewController:liveListViewController animated:t
}
```



步骤七：观众进入直播间

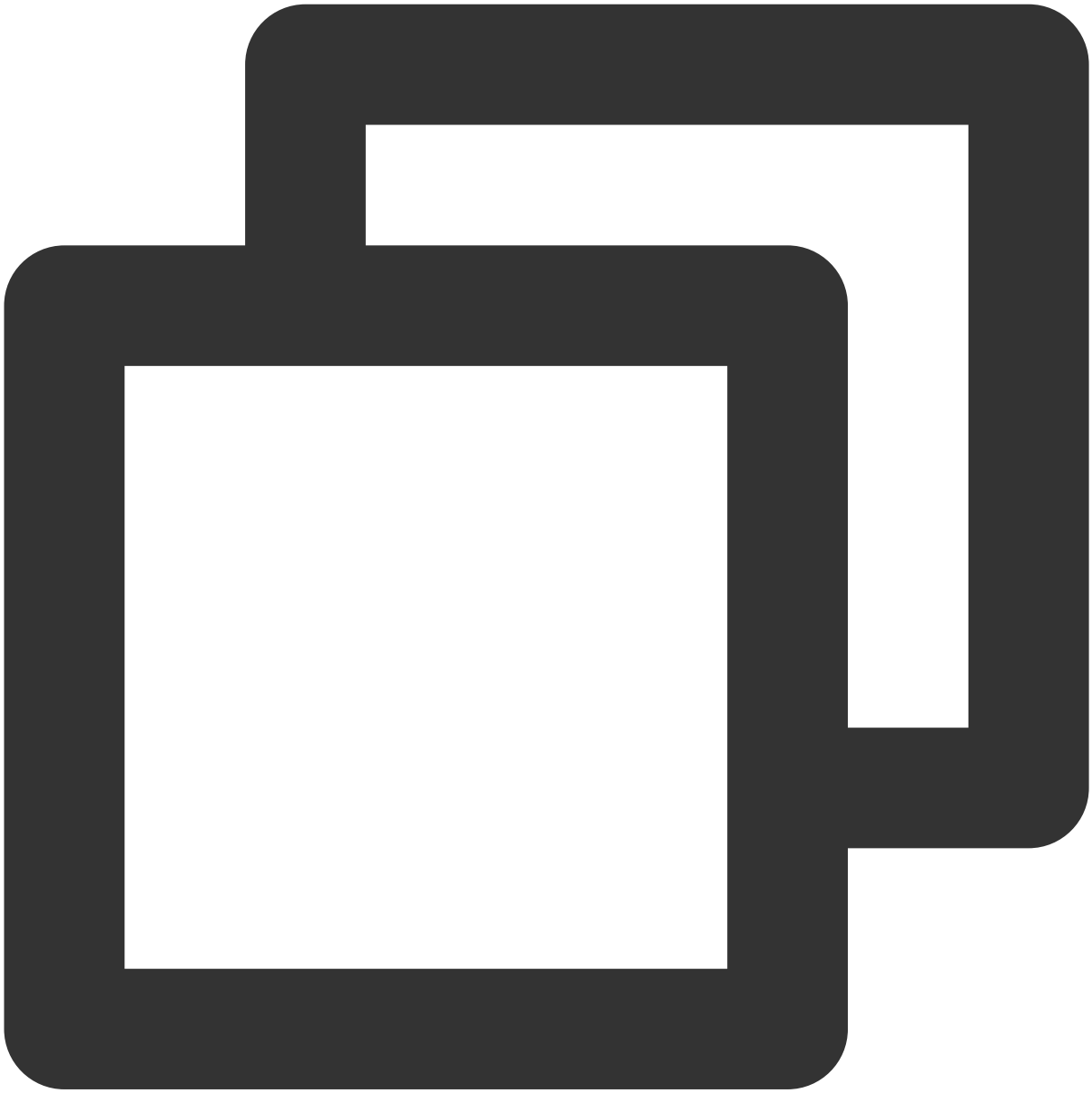
Swift

Objective-C



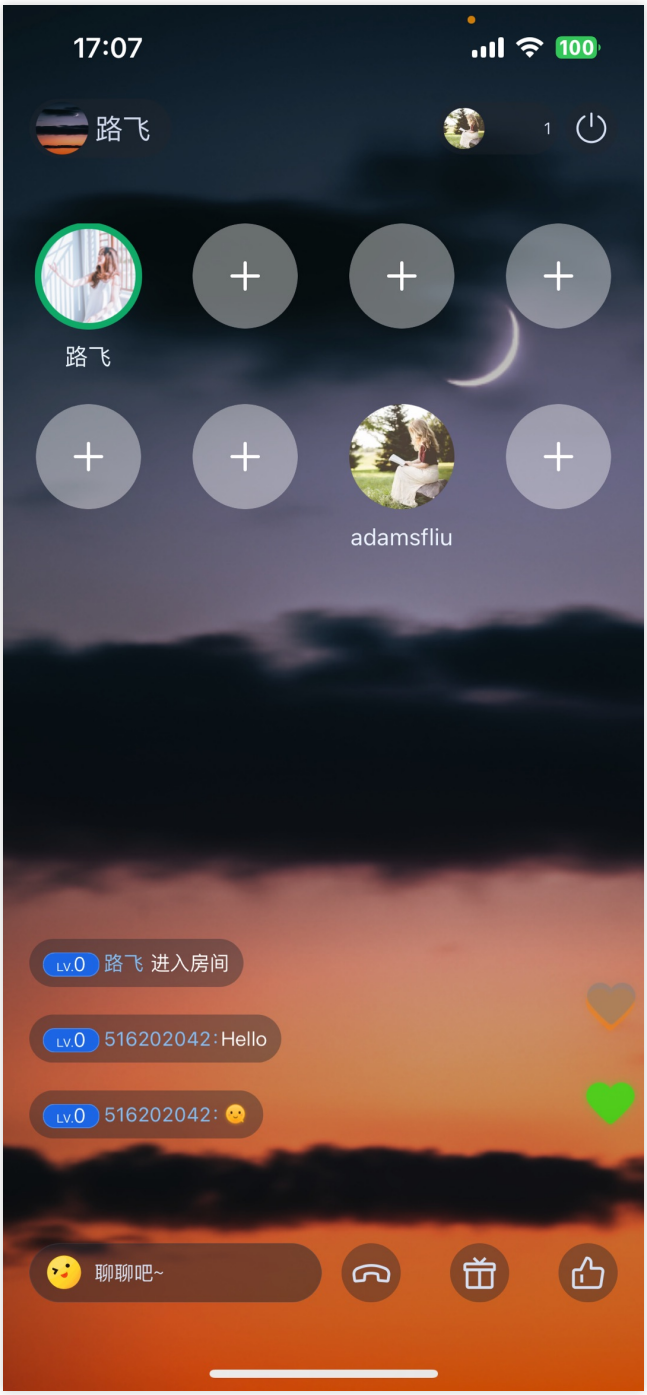
```
//
//  MainViewController.swift
//

import UIKit
import TUILiveKit

@objc private func buttonTapped(_ sender: UIButton) {
    //RoomId是创建语音聊天室的房间id
    let roomId = "123666";
```

```
    // 进入语音聊天室
    let viewController = TUIVoiceRoomViewController(roomId: roomId, behavior: .join
    self.navigationController?.pushViewController(viewController, animated: true)
}
```

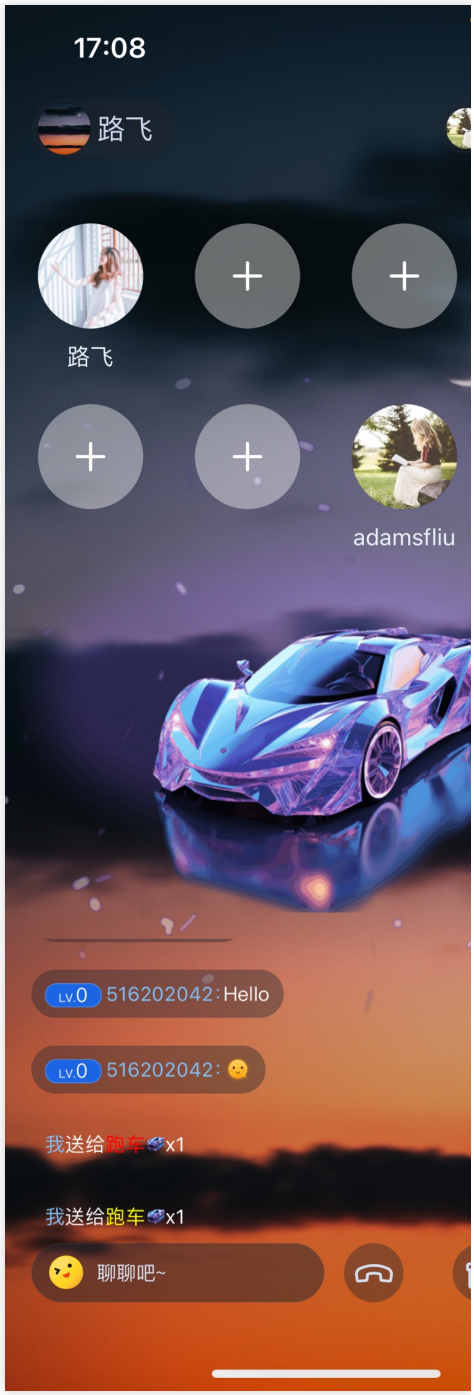


```
//
//  MainViewController.swift
//

#import <TUILiveKit/TUILiveKit-Swift.h>
```

```
- (void)buttonTapped:(UIButton *)sender {

    //RoomId是创建语音聊天室的房间id
    NSString *roomId = @"123666";

    // 进入语音聊天室
    TUIVoiceRoomViewController *voiceRoomController = [[TUIVoiceRoomViewController


    [self.navigationController pushViewController:voiceRoomController animated:true
}
```



语音聊天室



语音聊天室

更多特性

弹幕

礼物

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2024-07-03 14:56:53

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您可以在10分钟内完成接入工作，并最终得到一个包含完备 UI 界面的直播功能。

环境准备

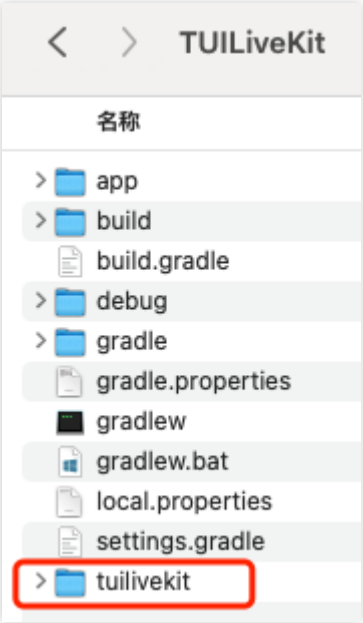
Android 5.0（SDK API Level 21）及以上版本。
Gradle 4.2.1 及以上的版本。
Android 5.0 及以上的手机设备。

步骤一：开通服务

在使用腾讯云提供的音视频服务前，您需要前往控制台，为应用开通音视频服务。具体步骤请参见 开通服务（TUILiveKit）

步骤二：下载 TUILiveKit 组件

在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuilivekit 子目录到您当前工程中的 app 同一级目录中，如下图：



步骤三：工程配置

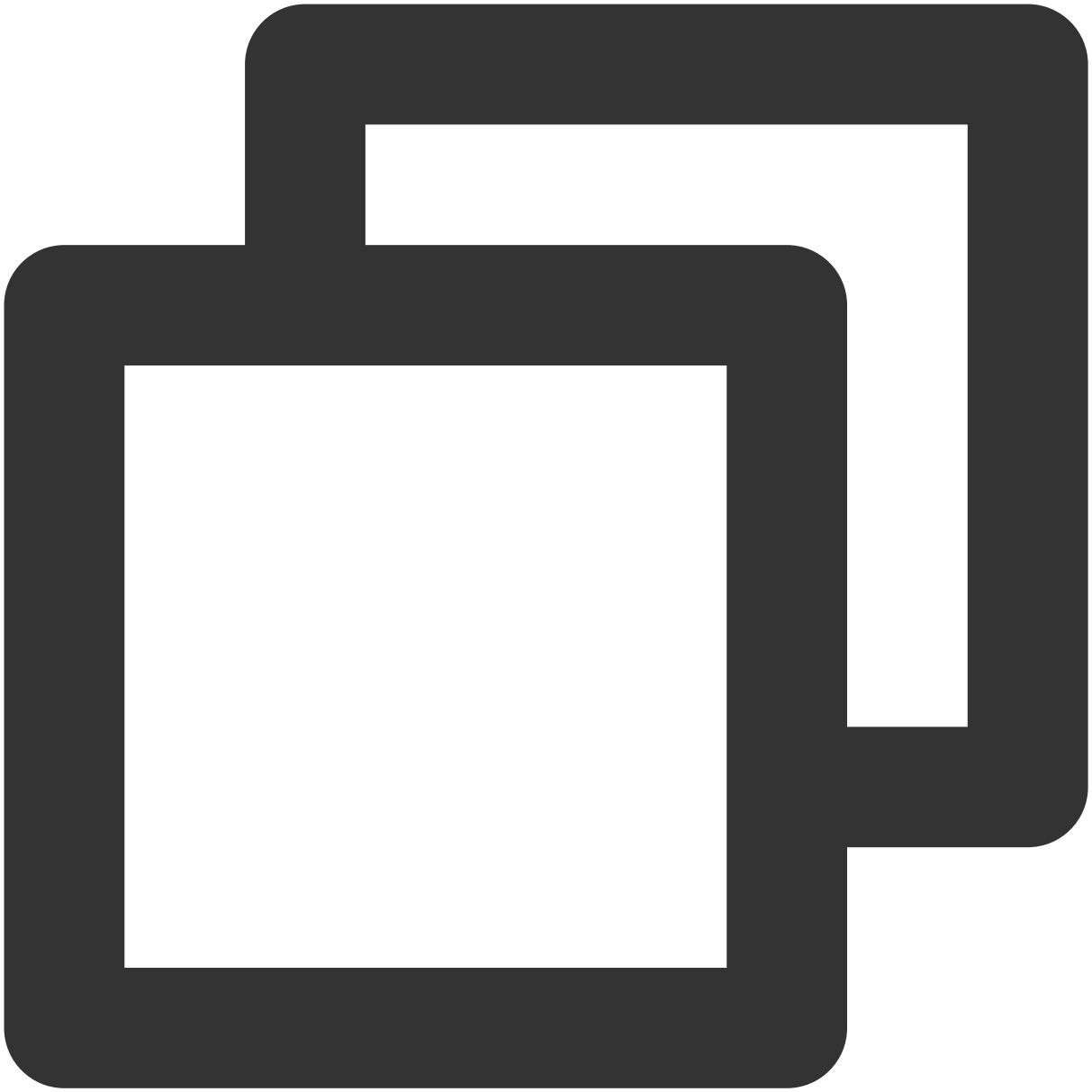
Java

1. 在工程中添加 `jitpack` 仓库依赖（下载播放礼物 svg 动画的三方库 `SVGAPlayer` ）：

gradle7.0以下版本

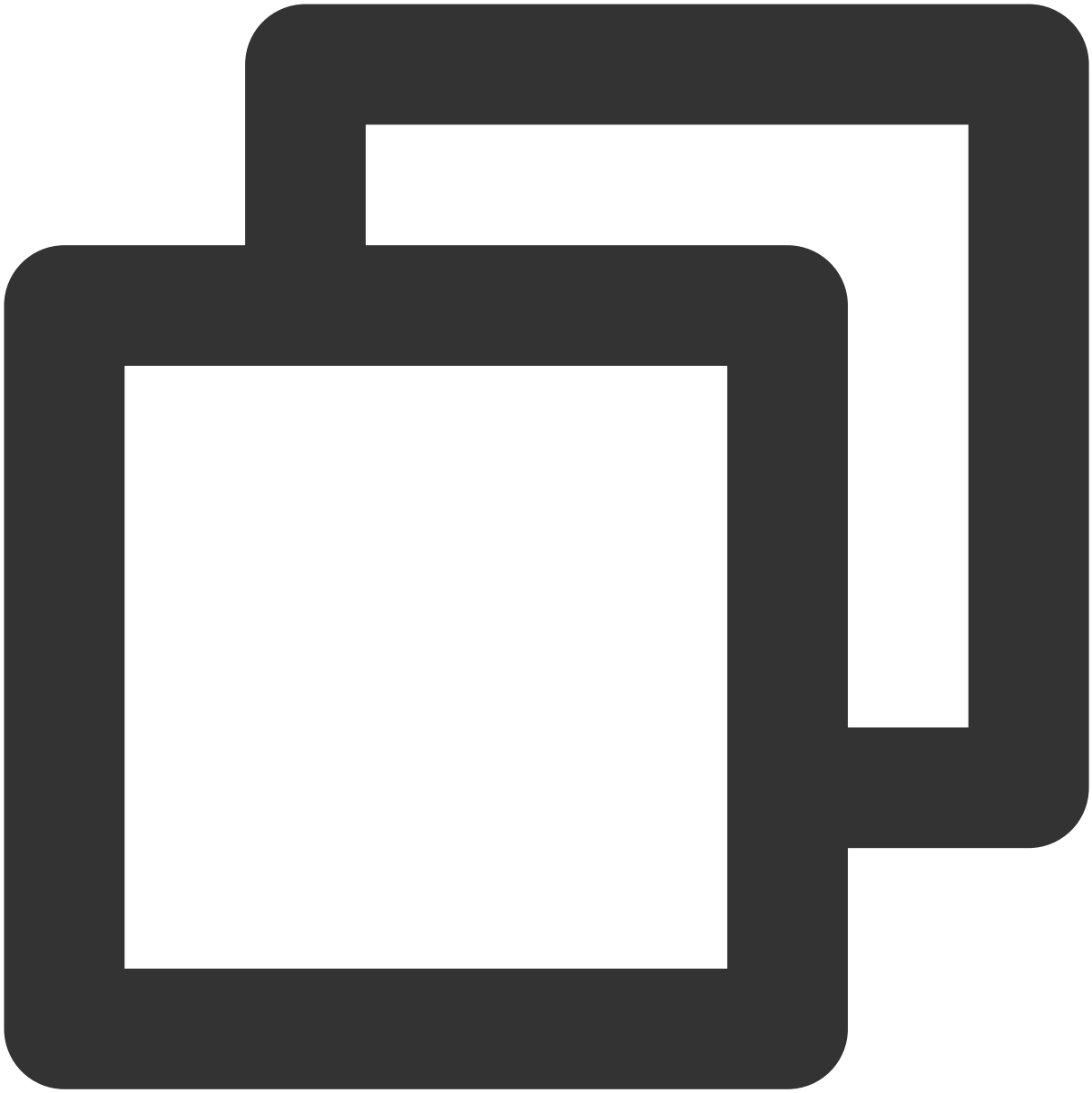
gradle7.0及以上版本

在工程根目录下的 build.gradle 文件中添加 jitpack 仓库地址：



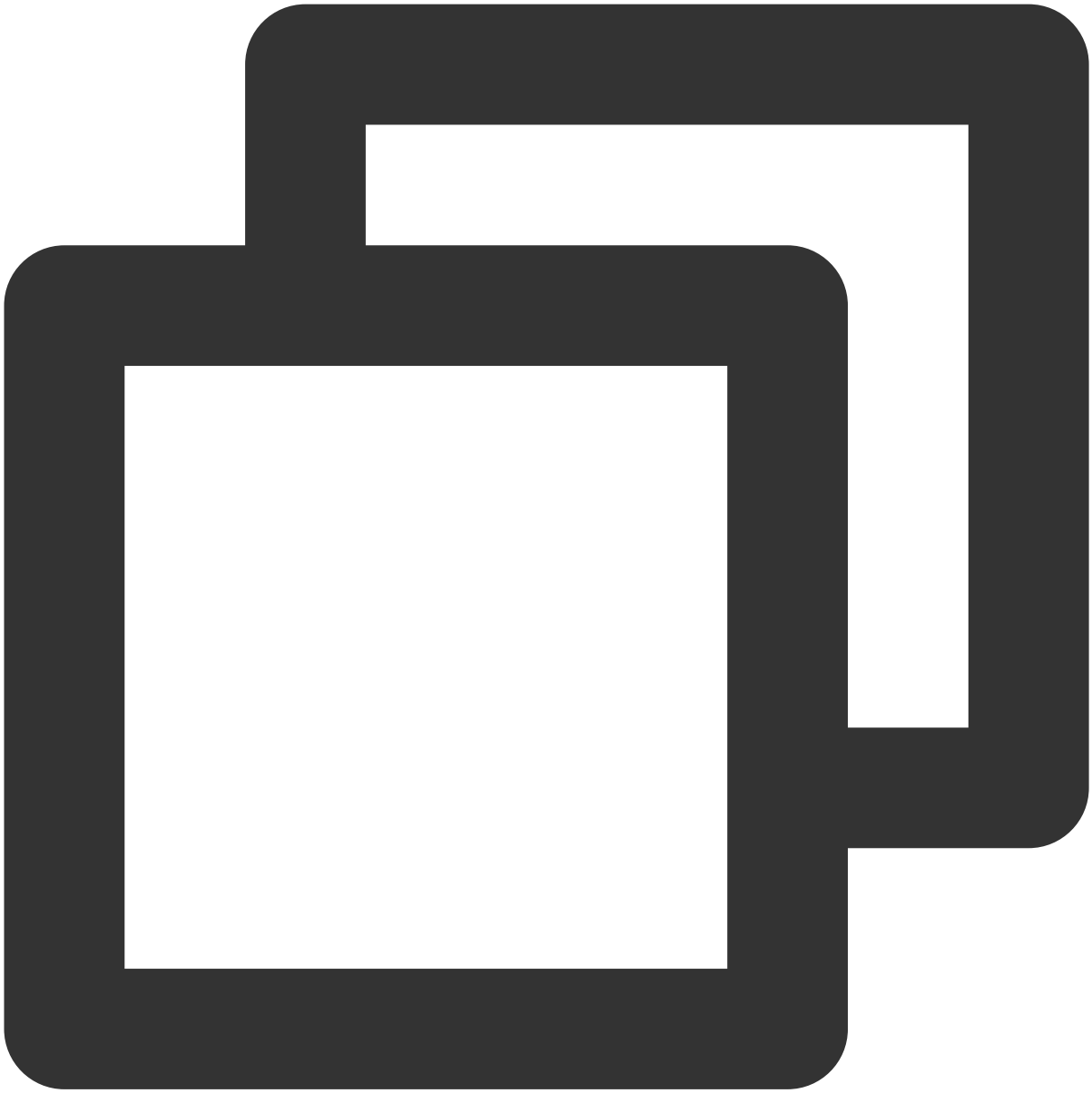
```
allprojects {
    repositories {
        google()
        mavenCentral()
        // 添加 jitpack 仓库地址
        maven { url 'https://jitpack.io' }
    }
}
```

在工程根目录下的 settings.gradle 文件中添加 jitpack 仓库地址：



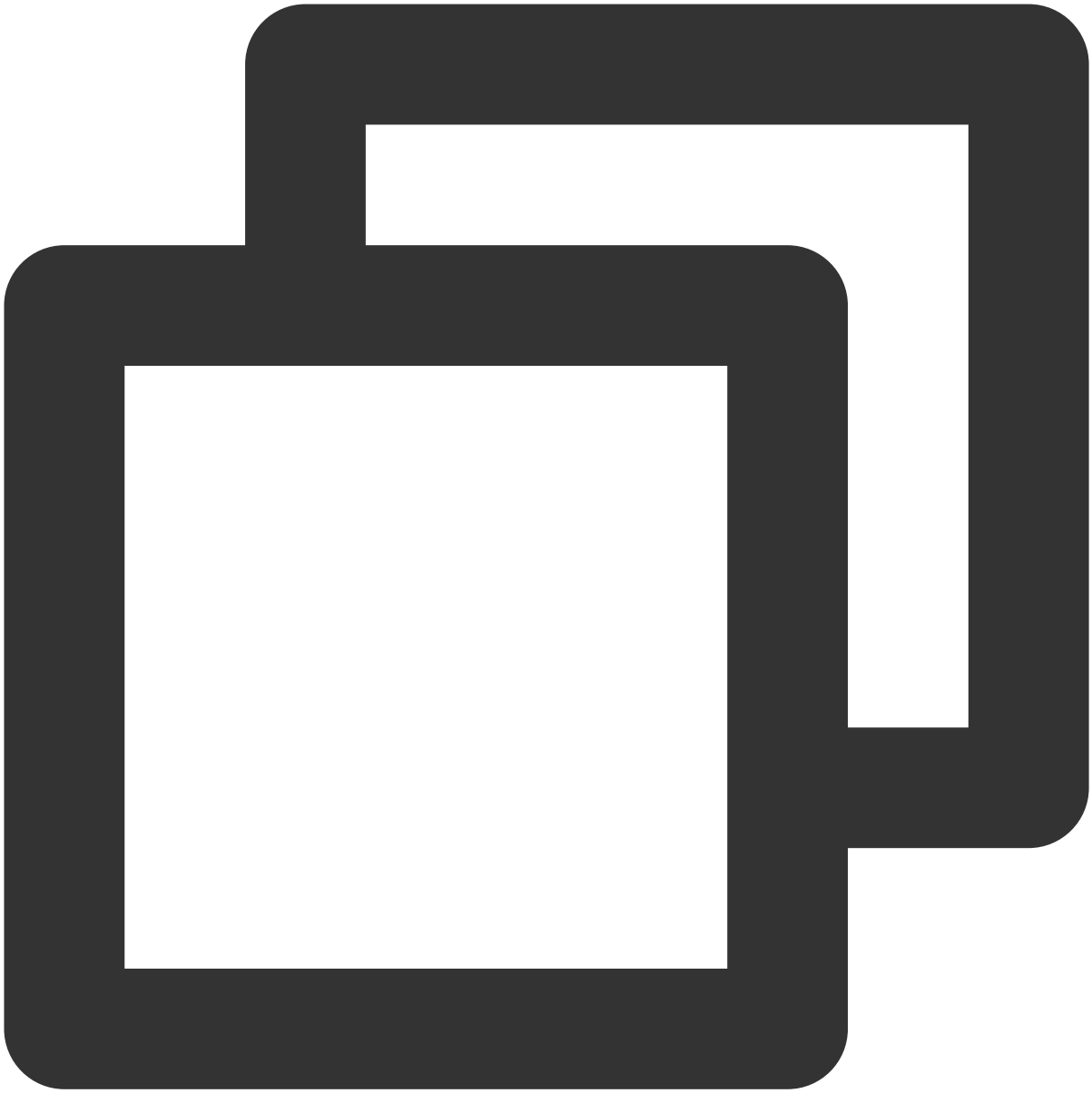
```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        // 添加 jitpack 仓库地址
        maven { url 'https://jitpack.io' }
    }
}
```

2. 在工程根目录下找到 `settings.gradle` 文件，并在其中增加如下代码，它的作用是将 步骤二 中下载的 tuilivekit 组件导入到您当前的项目中：



```
include ':tuilivekit'
```

3. 在 app 目录下找到 `build.gradle` 文件，并在其中增加如下代码，它的作用是声明当前 app 对新加入的 tuilivekit 组件的依赖：

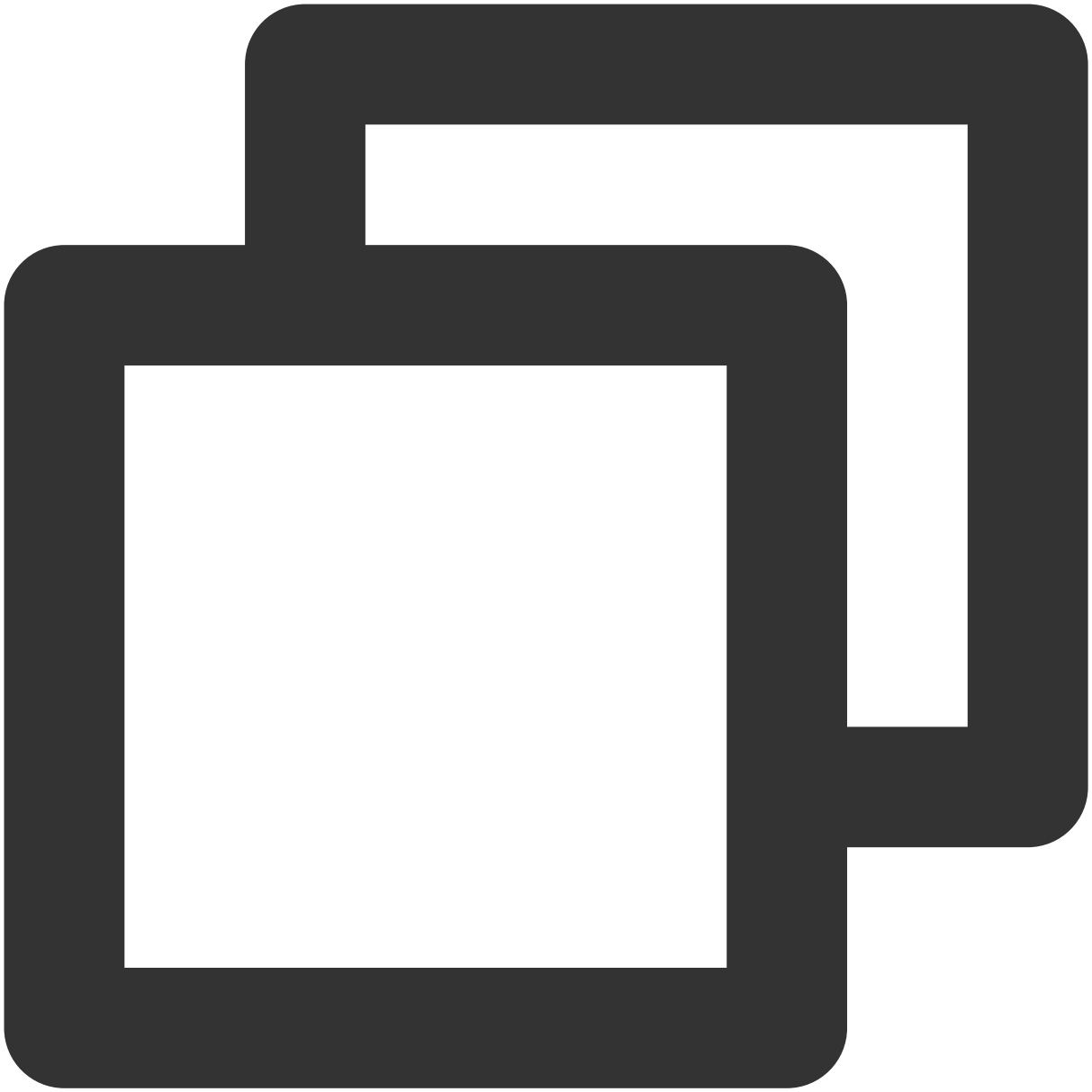


```
api project(':tuilivekit')
```

说明：

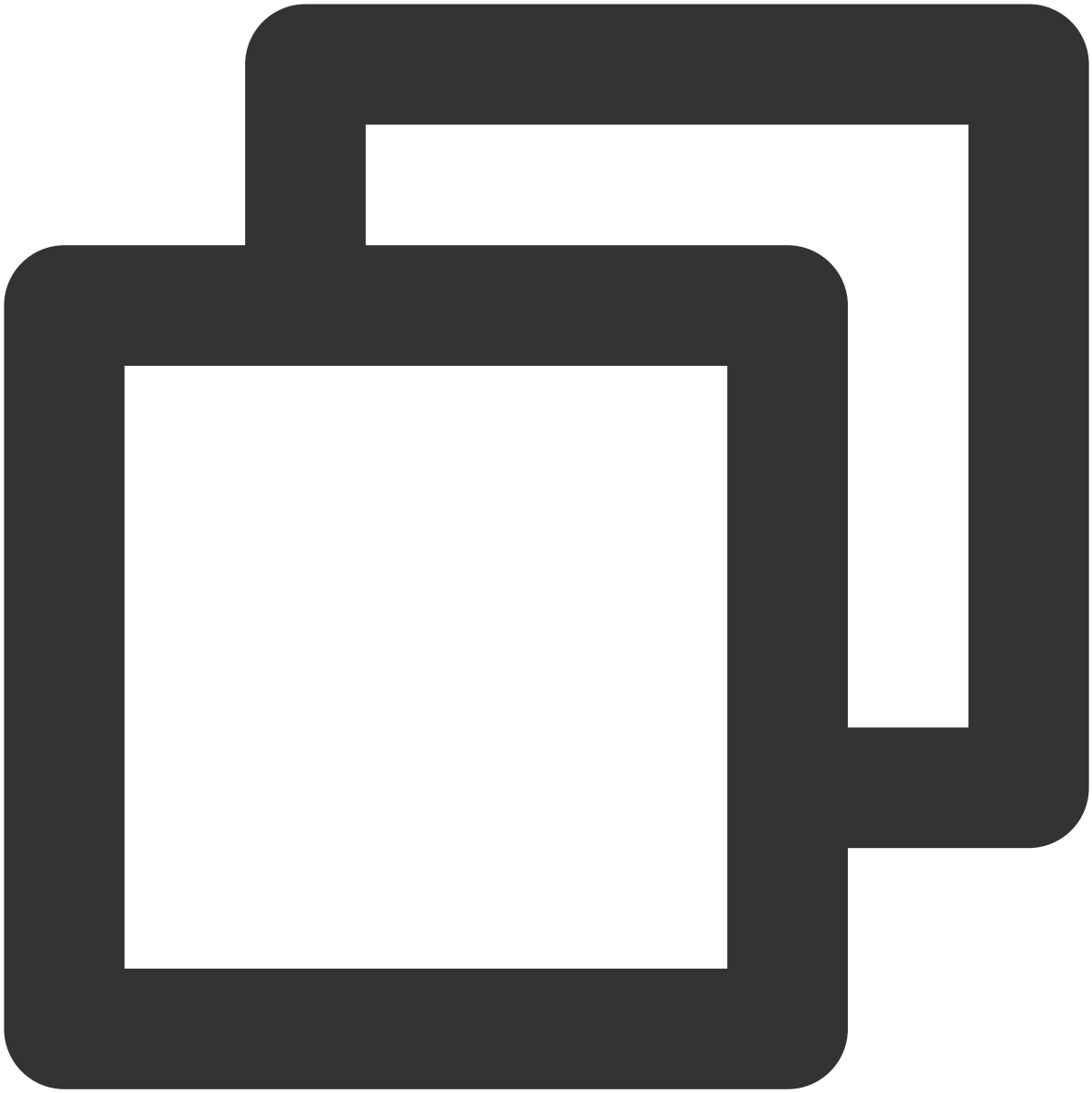
TUILiveKit 工程内部已经默认依赖： `TRTC SDK` 、 `IM SDK` 、 `tuiroomengine` 以及公共库 `tuicore` ，不需要开发者单独配置。如需进行版本升级，则修改 `tuilivekit/build.gradle` 文件即可。

4. 由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 `proguard-rules.pro` 文件中添加如下代码：



```
-keep class com.tencent.** { *; }
```

5. 在 `AndroidManifest.xml` 里，给 `application` 的 `android:theme` 属性配置一个 `Theme.AppCompat` 主题：



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
        ...
    </application>
</manifest>
```

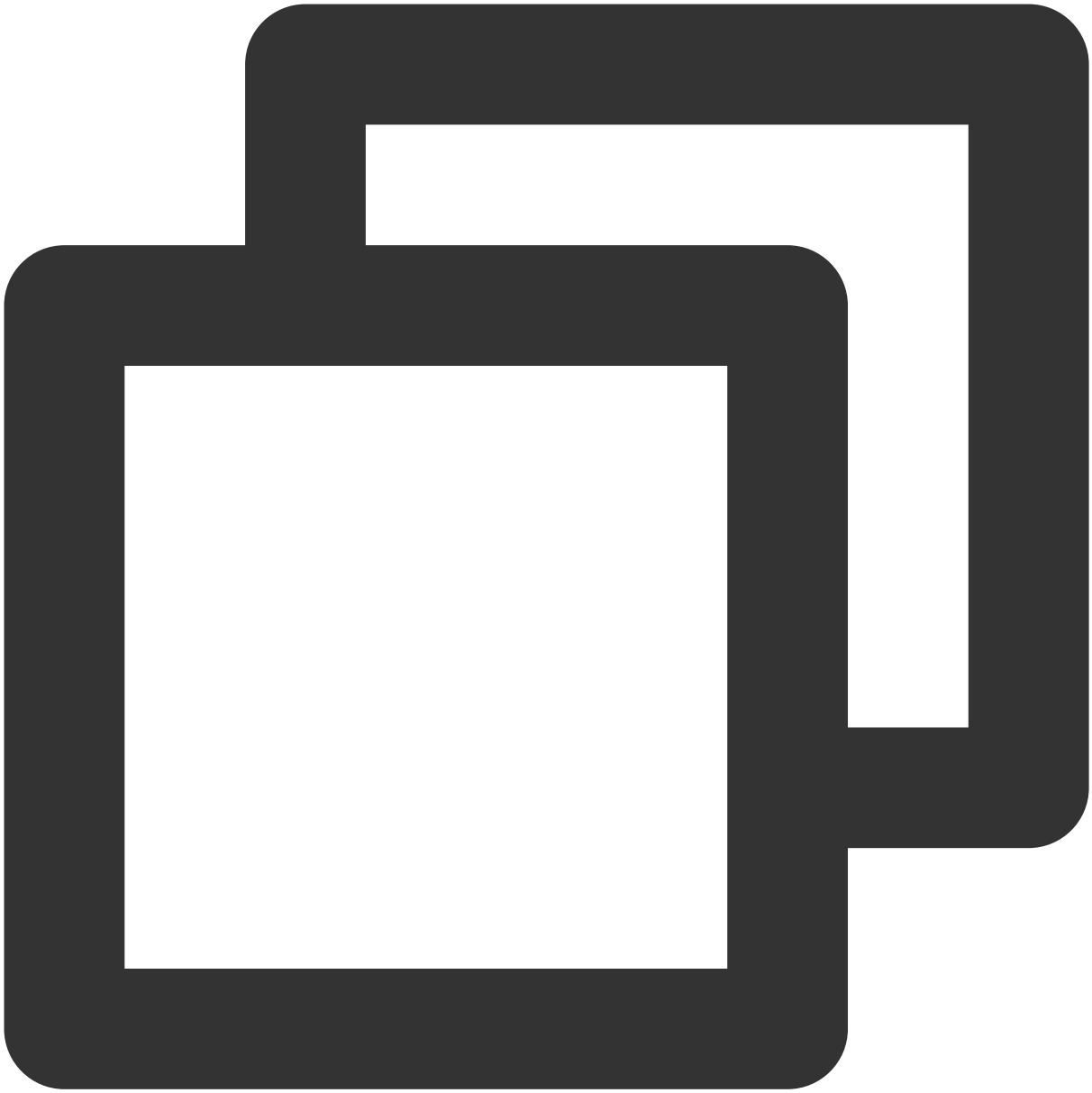
注意：

TUILiveKit 会在内部帮助您动态申请相机、麦克风、读取存储权限等，如果因为您的业务问题需要删减，可以请修改 `tuilivekit/src/main/AndroidManifest.xml` 。
如果遇到 `AndroidManifest.xml` 提示 `allowBackup` 相关异常，请参见 allowBackup 异常。
如果遇到 `Theme.AppCompat` 相关问题，请参见 Activity 主题问题。

步骤四：登录

在调用 TUILiveKit 组件各项功能之前，先要执行 TUI 组件的登录。在您的项目中，建议在您的业务登录场景里或者 App 的首个启动 Activity 里，添加执行如下登录代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤关键，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Java



```
//登录
TUILogin.login(context,
    1400000001,      // 请替换为步骤一取到的 SDKAppID
    "denny",         // 请替换为您的 UserID
    "xxxxxxxxxxx",   // 您可以在控制台中计算一个 UserSig 并填在这个位置
    new TUICallback() {
    @Override
    public void onSuccess() {
        Log.i(TAG, "login success");
    }
```

```
    @Override
    public void onError(int errorCode, String errorMessage) {
        Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + errorMessage
    }
});
```

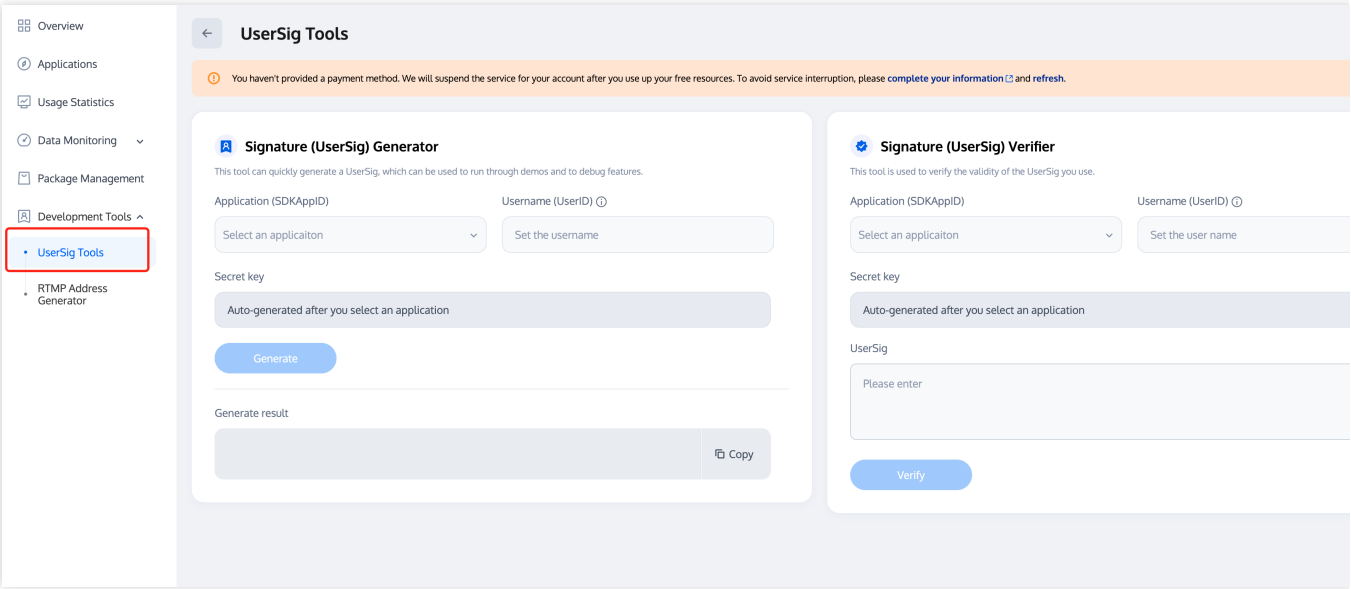
参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

SDKAppID：在 开通服务（TUILiveKit） 中您已经获取到，这里不再赘述。

UserID：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。

UserSig：使用 开通服务（TUILiveKit） 获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台左侧项目栏中的 UserSig 工具，创建一个临时可用的 UserSig。



更多信息请参见 UserSig 相关。

注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下:

SDKAppID 设置错误。

userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 userSig。

userSig 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userID。

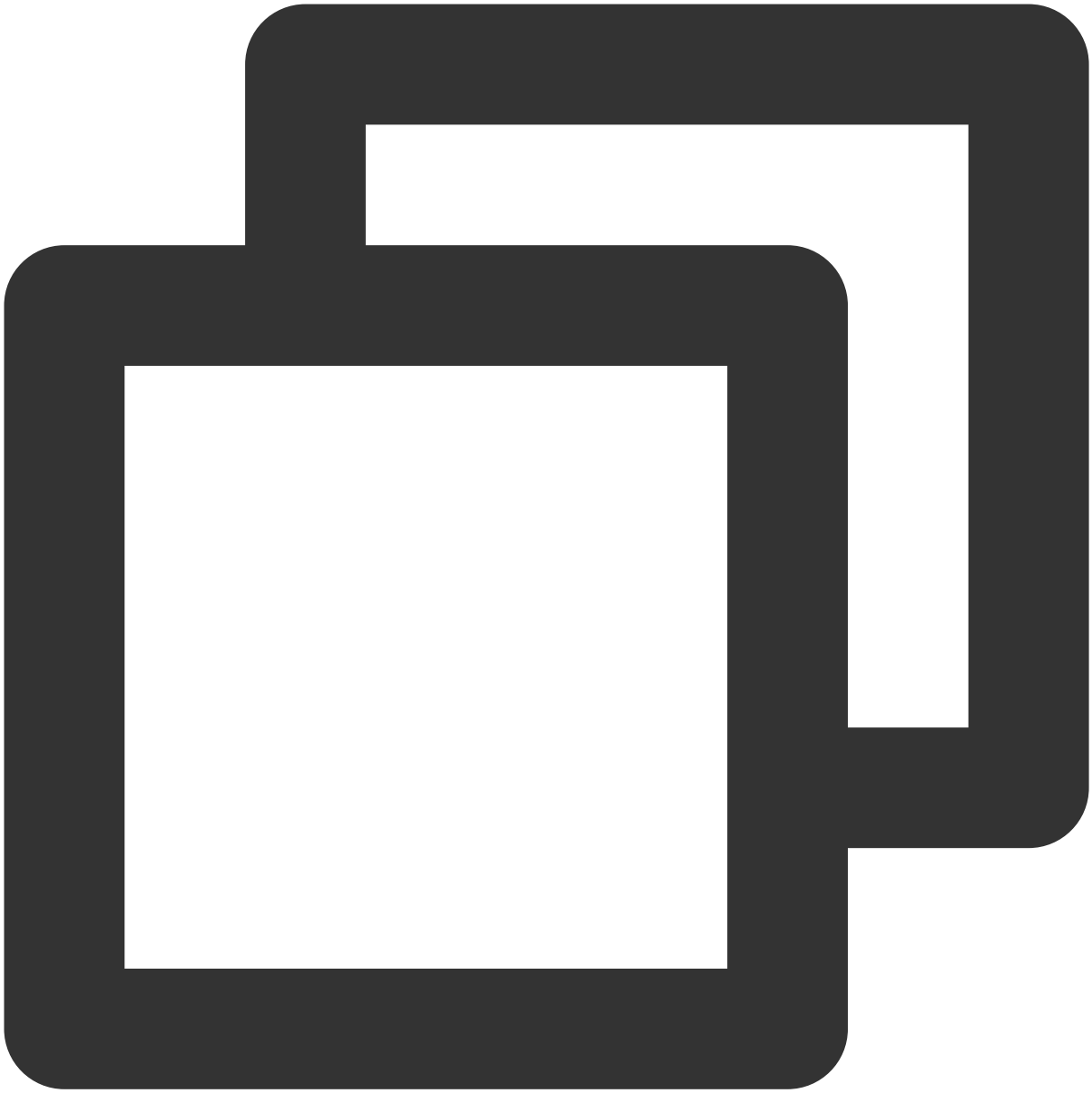
Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUILiveKit 组件时向您的服务器请求实时计算出的 userSig。

步骤五：进入直播预览画面

注意：

请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。

1. 新建 app_activity_anchor.xml 文件（默认路径：app/src/main/res/layout/app_activity_anchor.xml）。



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

2. 新建 AnchorActivity.java，并在 `AndroidManifest.xml` 里注册，通过加载 TUILiveKit 的 TUIVoiceRoomFragment 页面，就可以拉起预览画面。

Java

```
public class AnchorActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_anchor);
        //主播的房间 ID
        String roomId = "123666";
```

```
        //创建房间的参数
        LiveDefine.RoomParams params = new LiveDefine.RoomParams();
        //房间最大麦位数，默认套餐包支持的最大麦位数量
        params.maxSeatCount = 0;
        //上麦模式
        params.seatMode = TUIRoomDefine.SeatMode.APPLY_TO_TAKE;

        //在 Activity 中显示 主播预览页面，点击预览页面的开始直播按钮，即可发起在线语音直播
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction(
        TUIVoiceRoomFragment fragment = new TUIVoiceRoomFragment(roomId,
        LiveDefine.RoomBehavior.PREPARE_CREATE, params);
        fragmentTransaction.add(R.id.fl_container, fragment);
        fragmentTransaction.commit();
    }
}
```

在 app 项目的 `AndroidManifest.xml` 里注册 `AnchorActivity` （请使用您的 `AnchorActivity` 实际包名）：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application>
        ...
        <!-- 示例：注册AnchorActivity，请使用您的实际包名 -->
        <activity android:name="com.trtc.uikit.livekit.example.main.AnchorActivity"
                  android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"/>
        ...
    </application>
</manifest>
```

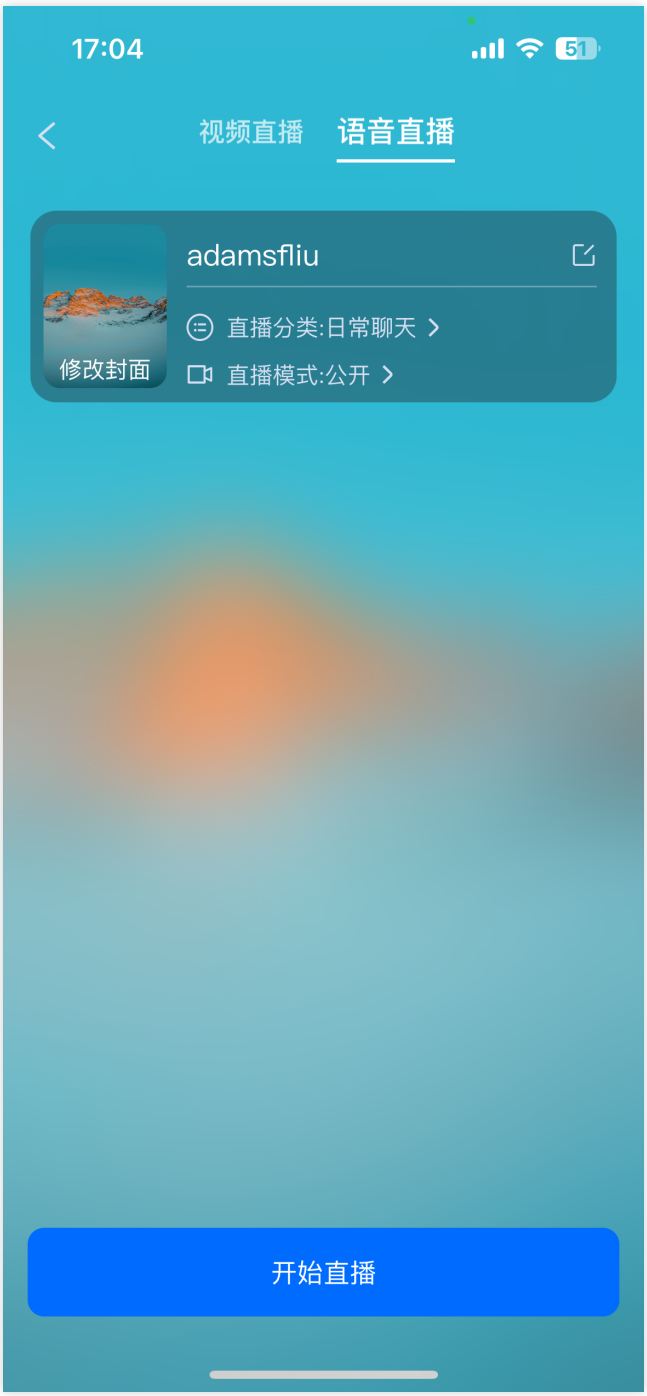
注意：

由于 `AnchorActivity` 继承自 `AppCompatActivity`，所以要给 `AnchorActivity` 设置一个 `Theme.AppCompat` 主题。您可以修改成自己的 `Theme.AppCompat` 主题，如果遇到 Theme.AppCompat 相关问题，请参见 Activity 主题问题。

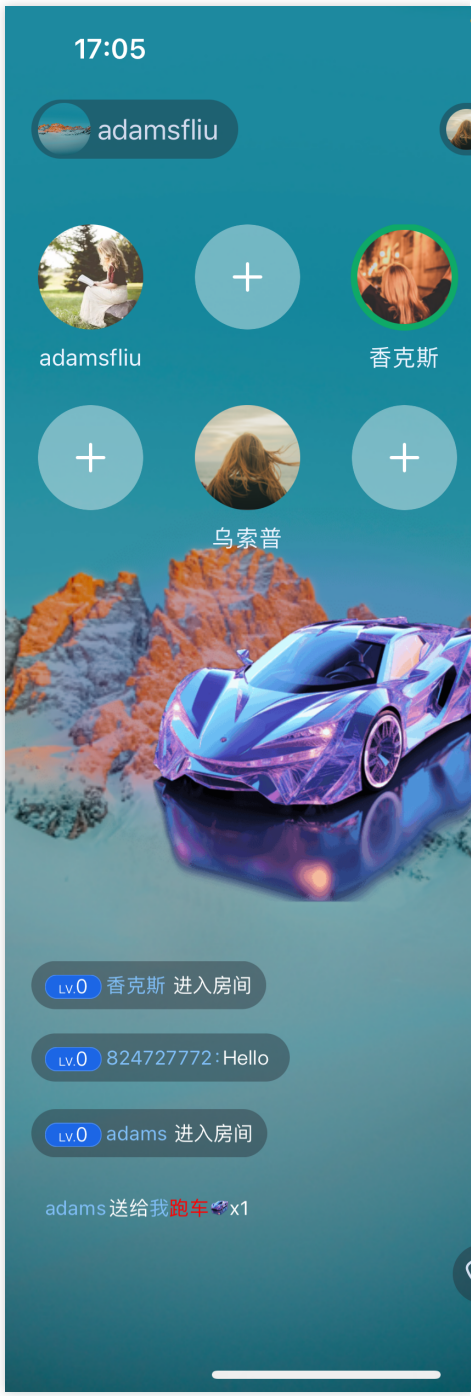
3. 在您需要开启直播的地方（具体由您的业务决定，默认情况下可在 MainActivity 某点击事件里执行），执行如下操作，拉起主播开播页面：

Java

```
Intent intent = new Intent(context, AnchorActivity.class);
startActivity(intent);
```



语音聊天室预览画面



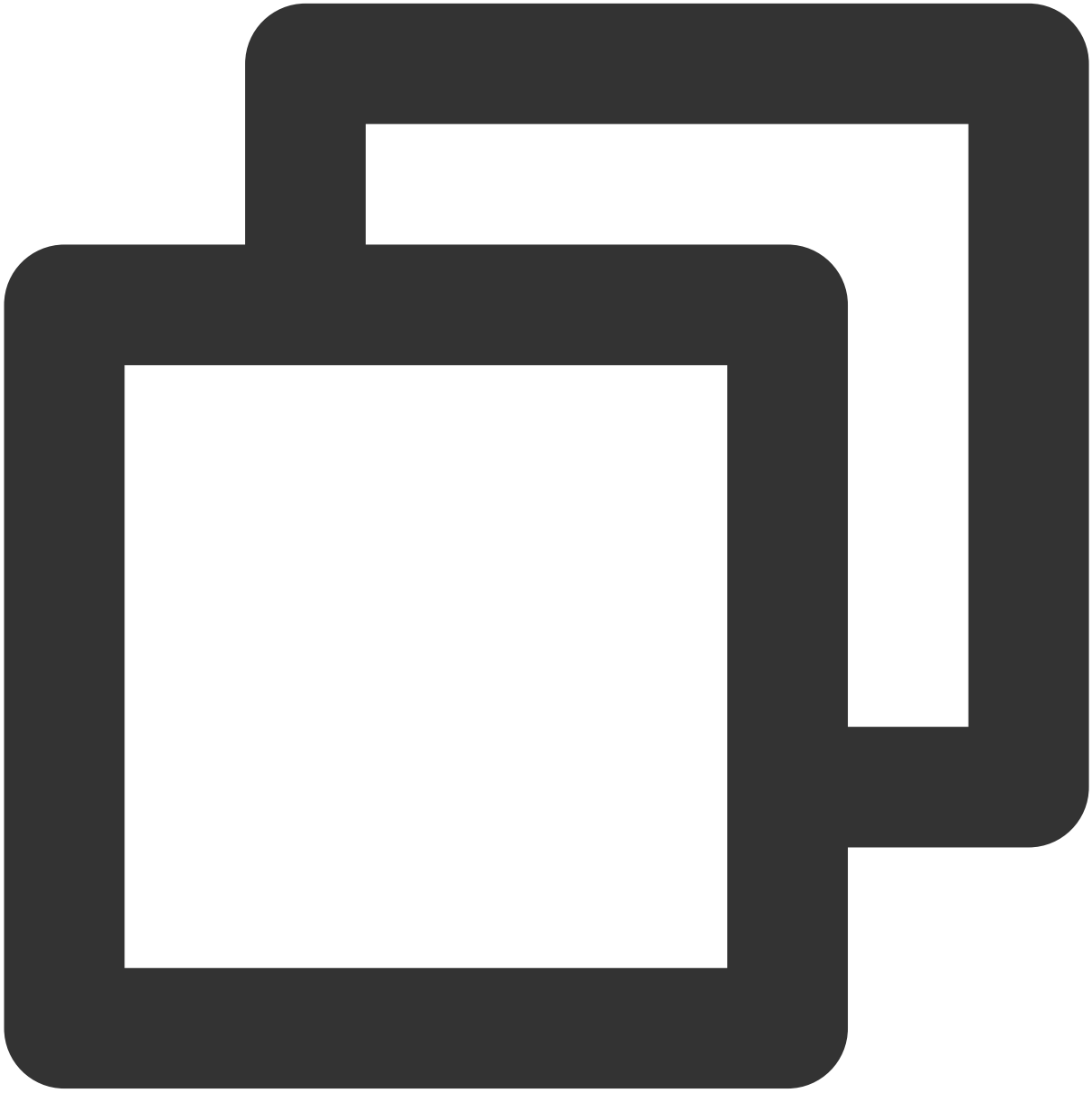
语音聊天室房间内画面

步骤六：观众进入直播间

注意：

请务必确保已经按照 步骤四 完成登录操作。只有 `TUILogin.login` 登录成功后观众才能正常进入直播间。并且，登录的观众 `UserID` 不要与主播 `UserID` 相同。

1. 新建 app_activity_audience.xml 文件（默认路径：app/src/main/res/layout/app_activity_audience.xml）。



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

2. 新建 AudienceActivity.java，并在 `AndroidManifest.xml` 里注册， 通过加载 TUILiveKit 的 TUIVoiceRoomFragment 页面，进入直播间。

Java

```
public class AudienceActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_audience);
        //主播的房间 ID
        String roomId = "123666";

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction(
        TUIVoiceRoomFragment fragment = new TUIVoiceRoomFragment(roomId,
        LiveDefine.RoomBehavior.JOIN, null);
        fragmentTransaction.add(R.id.fl_container, fragment);
        fragmentTransaction.commit();
    }
}
```

在 app 项目的 `AndroidManifest.xml` 里注册 `AudienceActivity` （请使用您的 `AudienceActivity` 实际包名）：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application>
        ...
        <!-- 示例：注册AudienceActivity，请使用您的实际包名 -->
        <activity android:name="com.trtc.uikit.livekit.example.main.AudienceActivit
                  android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"/>
        ...
    </application>
</manifest>
```

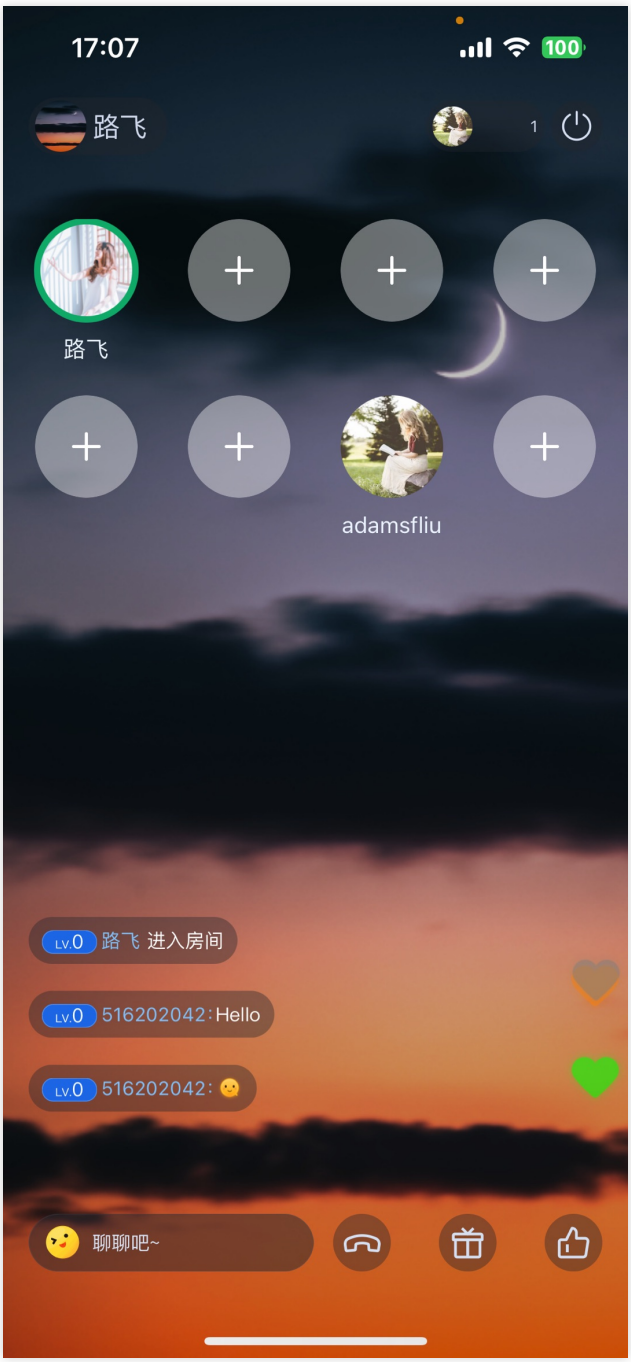
注意：

由于AudienceActivity继承自 `AppCompatActivity` ，所以要给 `AnchorActivity` 设置一个 `Theme.AppCompat` 主题。您可以修改成自己的 `Theme.AppCompat` 主题。如果遇到 Theme.AppCompat 相关问题，请参考 Activity 主题问题。

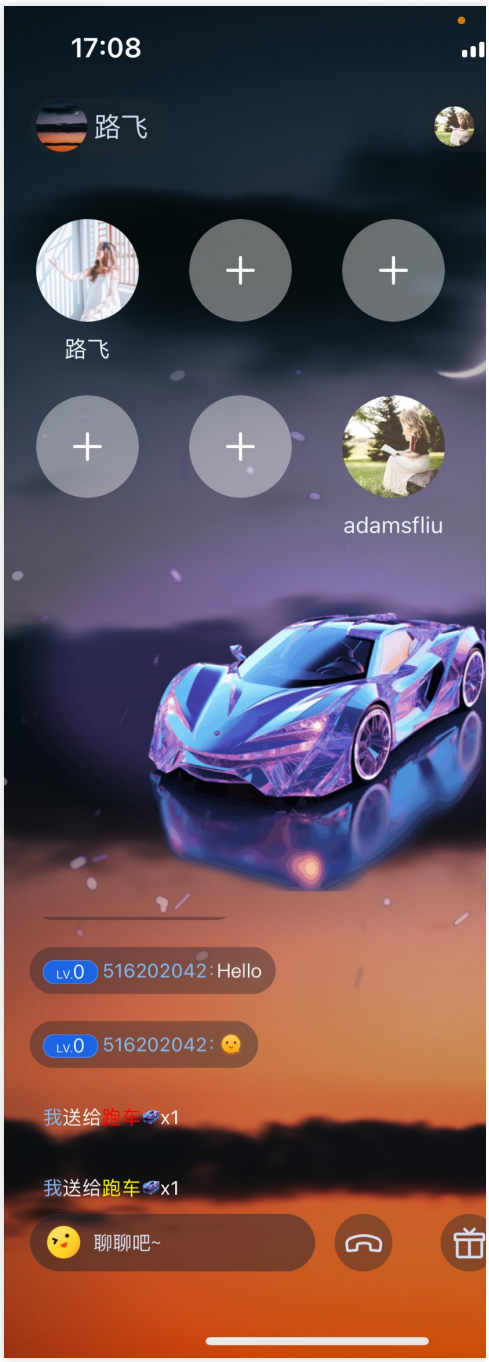
3. 在您需要观众进房的地方（具体由您的业务决定，默认情况下可在 MainActivity 某点击事件里执行），执行如下操作，拉起观众进房页面：

Java

```
Intent intent = new Intent(context, AudienceActivity.class);
startActivity(intent);
```



语音聊天室



语音聊天室

更多特性

弹幕

礼物

常见问题

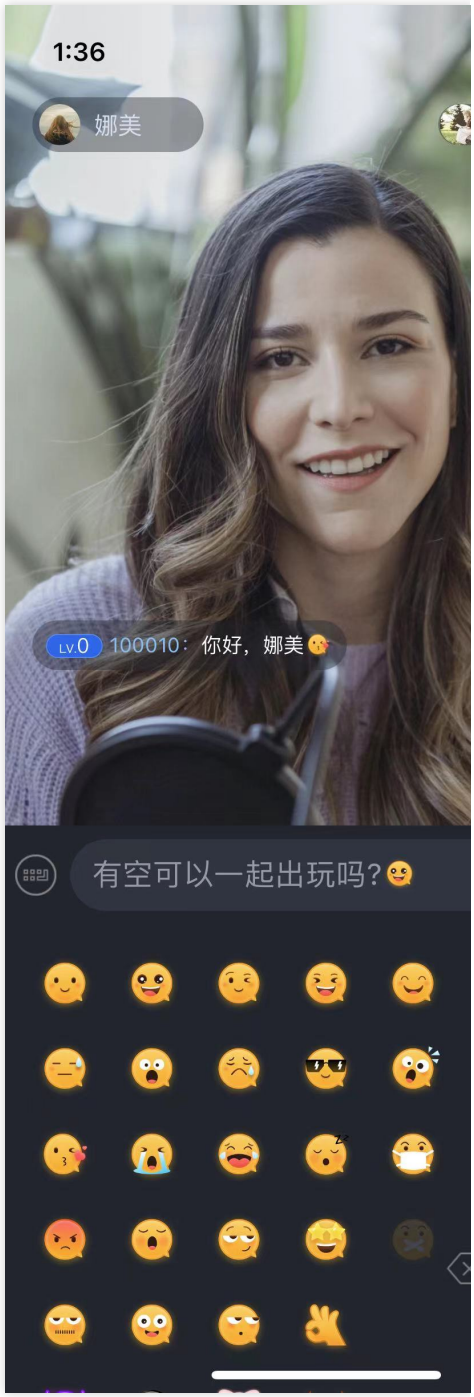
如果您的接入和使用中遇到问题，请参见 常见问题。

互动弹幕（TUILiveKit）
iOS

最近更新时间：2024-08-09 22:25:01

弹幕展示

弹幕显示



弹幕发送

说明：

支持**系统键盘**和**表情键盘**切换。

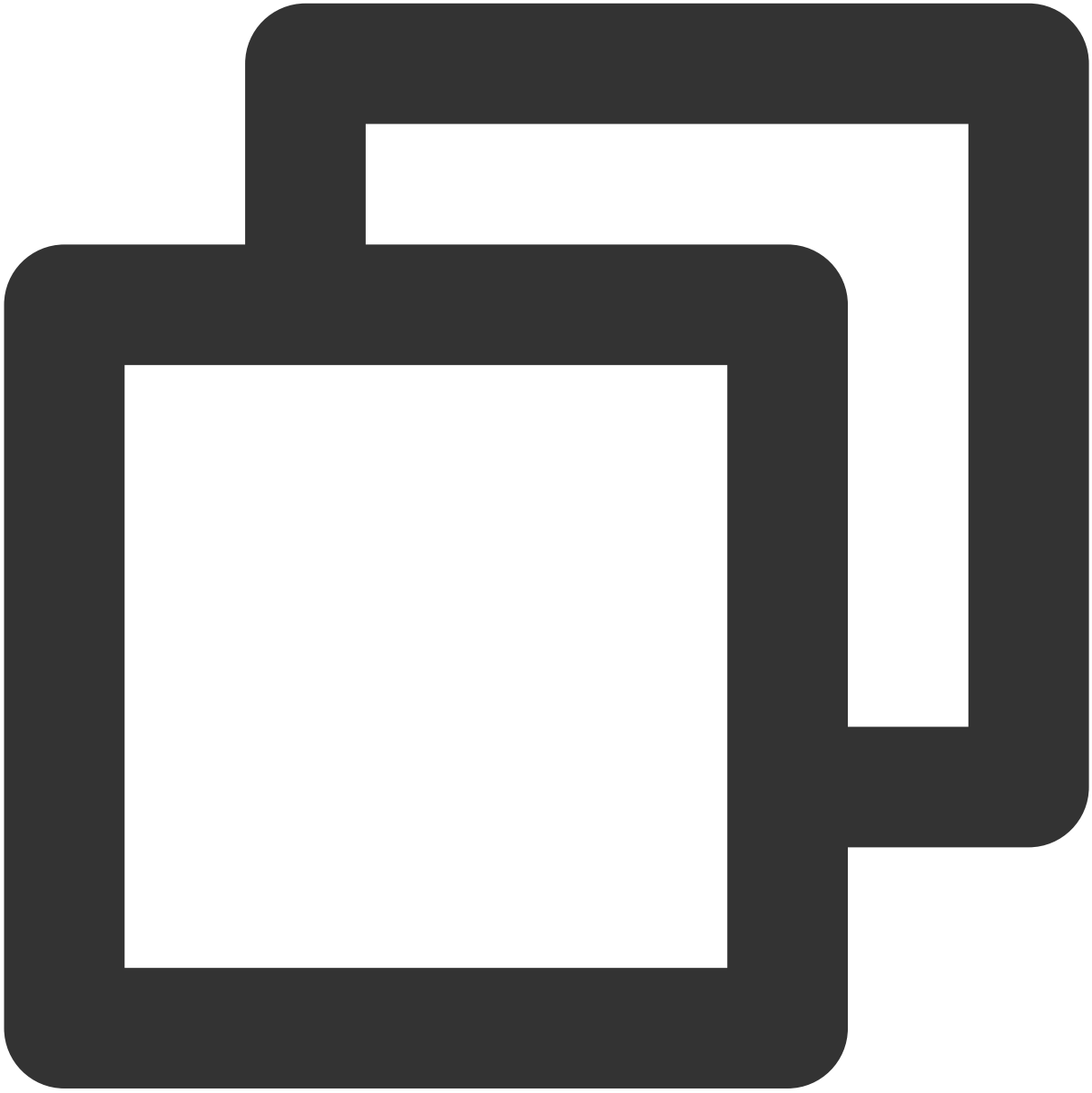
快速接入

弹幕组件主要包提供2个API：

`TUIBarrageButton` ：点击后可以拉起输入界面。

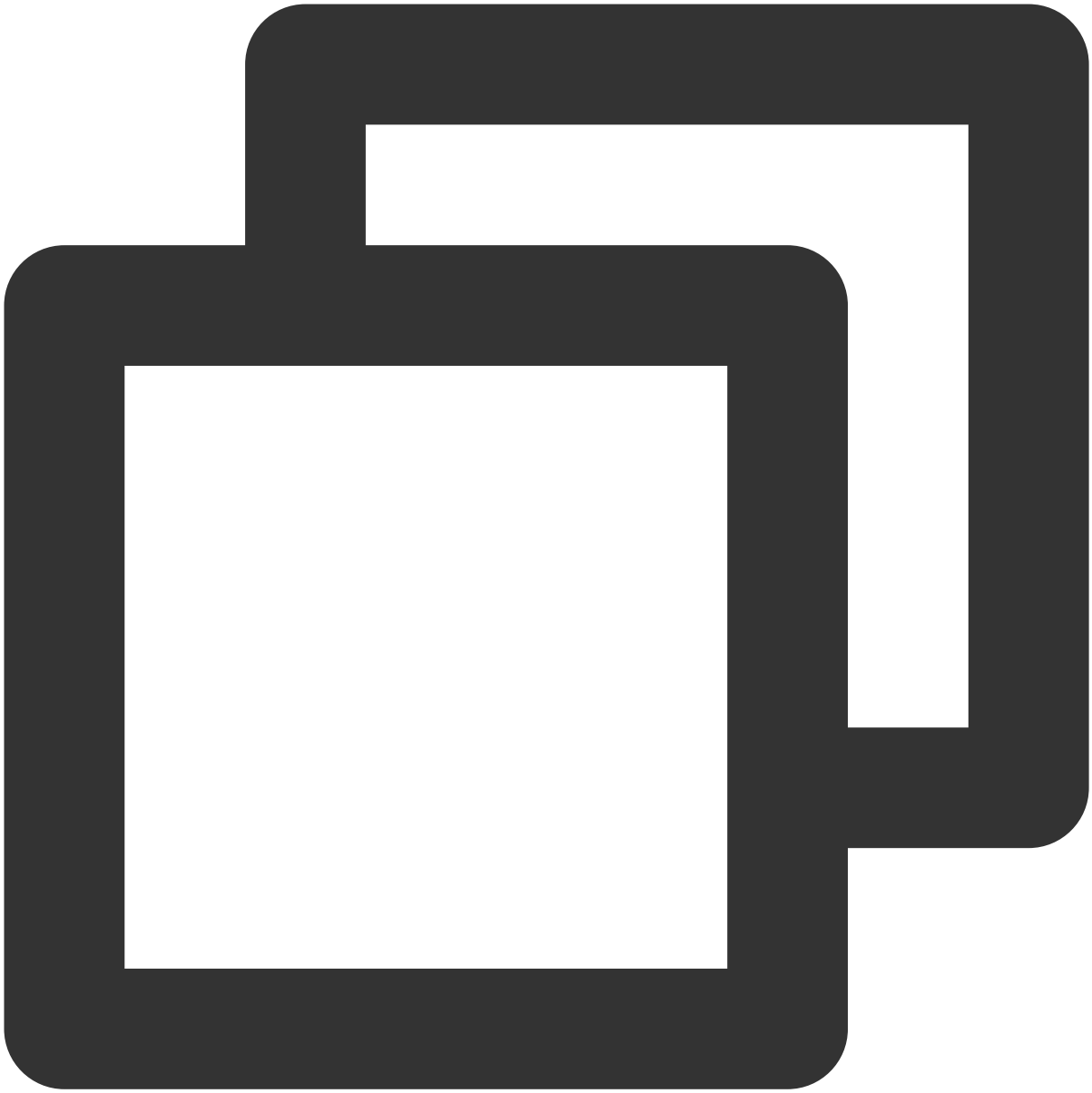
`TUIBarrageDisplayView` ：用于展示弹幕消息。

在需要发送弹幕的场景，创建 `TUIBarrageButton` ，点击后可以拉起输入界面：



```
let barrageButton: TUIBarrageButton = TUIBarrageButton(roomId: xxx)
view.addSubview(barrageButton)
//布局barrageButton
```

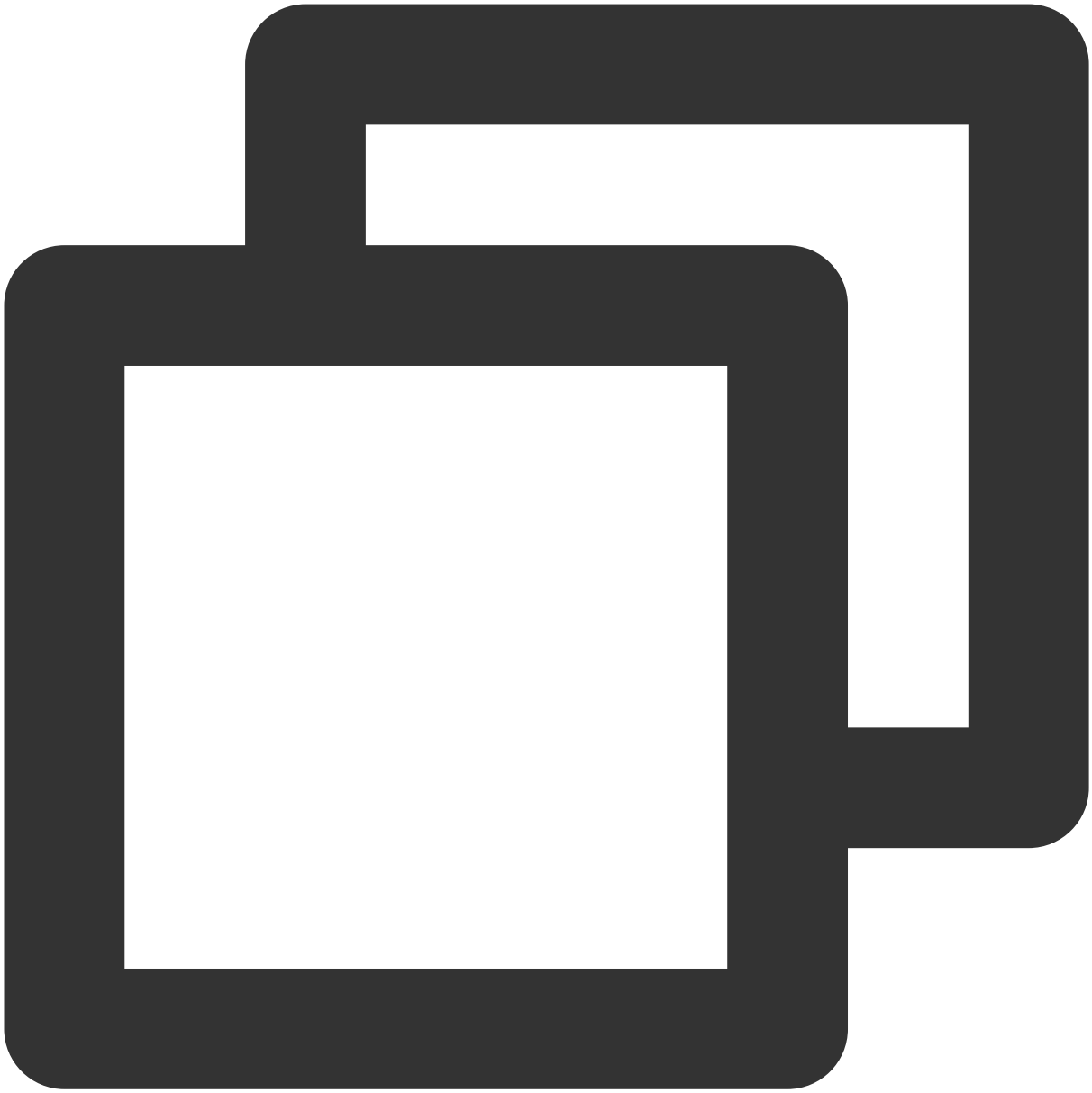
在需要展示弹幕的场景，使用 `TUIBarrageDisplayView` 来展示弹幕消息：



```
let barrageDisplayView: TUIBarrageDisplayView = TUIBarrageDisplayView(roomId: xxx)
view.addSubview(barrageDisplayView)
//布局barrageDisplayView
```

自定义消息样式

实现 `TUIBarrageDisplayView` 的代理 `TUIBarrageDisplayViewDelegate` 中的 `createCustomCell` 代理函数，用于自定义弹幕消息样式。



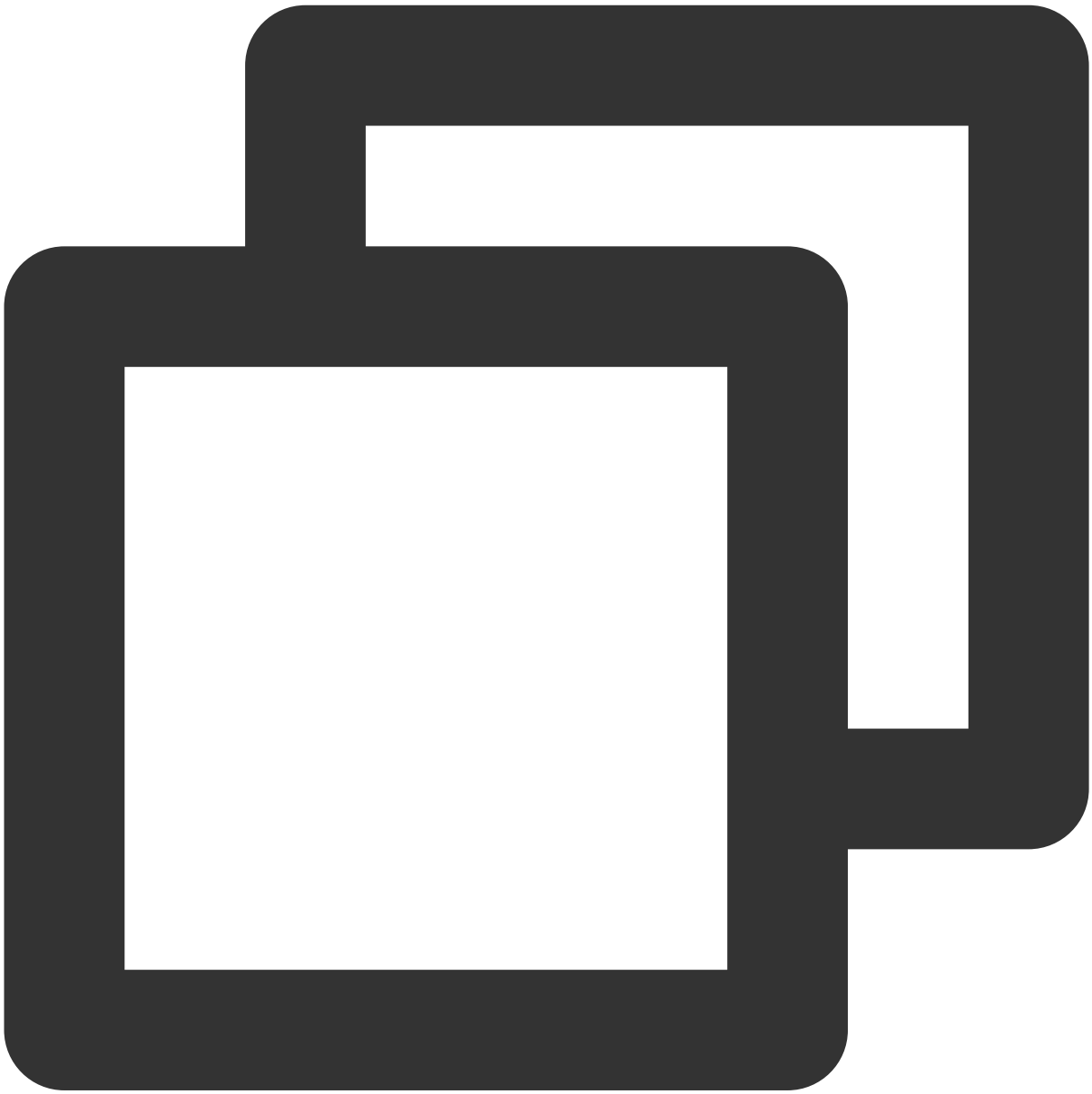
```
barrageDisplayView.delegate = self
extension UIViewController: TUIBarrageDisplayViewDelegate {
   func barrageDisplayView(_ barrageDisplayView: TUIBarrageDisplayView, createCusto
        //此处返回自定义弹幕UI
    }
}
```

说明：

`TUIBarrageDisplayView` 在展示消息时会先调用代理函数 `barrageDisplayView:createCustomCell` 用于获取用户对某条弹幕的自定义样式，若返回nil，则将采用 `TUIBarrageDisplayView` 的默认弹幕样式。

插入自定义消息

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息，通常自定义消息配合自定义样式，实现不一样的展示效果。



```
// 示例：在弹幕区插入一条礼物消息
let barrage = TUIBarrage()
barrage.content = "gift"
barrage.user.userId = sender.userId
barrage.user.userName = sender.userName
```

```
barrage.user.avatarUrl = sender.avatarUrl
barrage.user.level = sender.level
barrage.extInfo["xxx"] = "xxx"
barrageDisplayView.insertBarrages(barrage);
```

说明：

`TUIBarrage` 的 `extInfo` 是一个 `Map`，用于存放自定义数据。

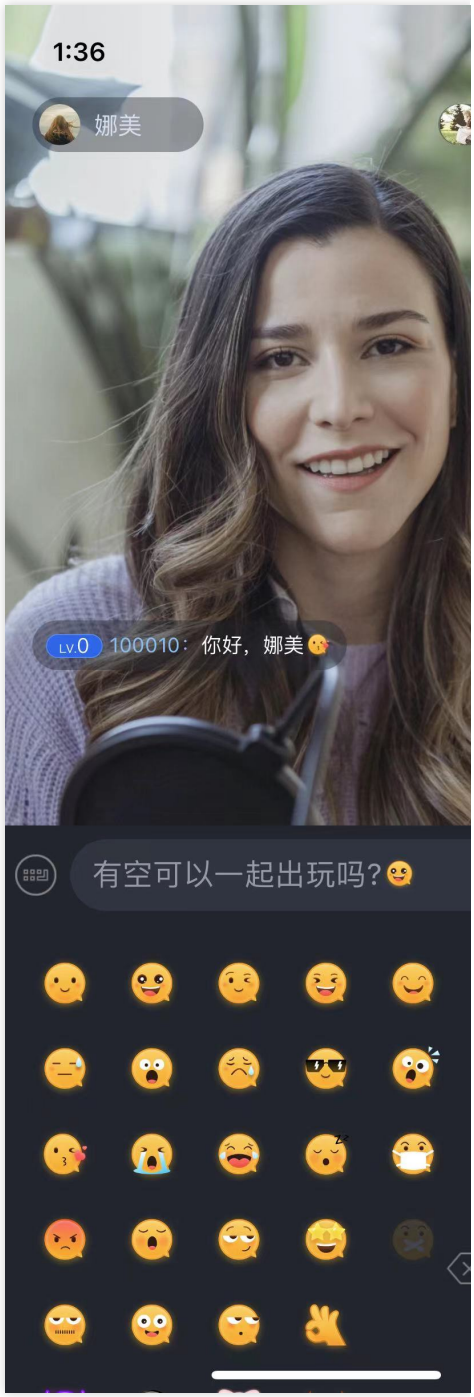
Android

最近更新时间：2024-08-09 22:25:01

互动弹幕功能，支持如下功能：发送弹幕消息、插入自定义消息、自定义消息样式等。弹幕消息支持表情输入，增加消息趣味性，让互动更愉悦。

弹幕展示

弹幕显示



弹幕发送

说明：

支持**系统键盘**和**表情键盘**切换。

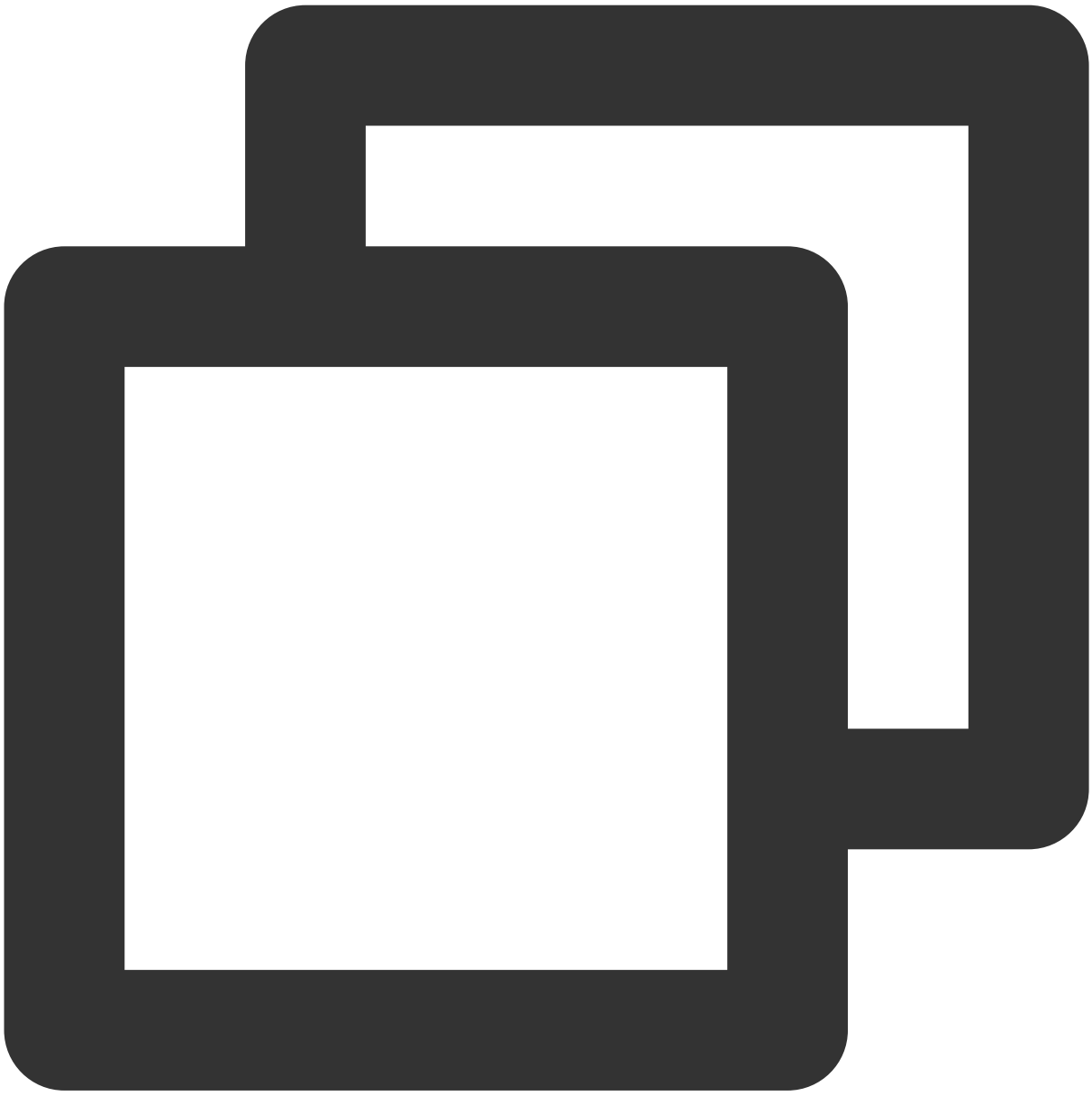
快速接入

弹幕组件主要包提供2个 API：

`TUIBarrageButton` ：点击后可以拉起输入界面。

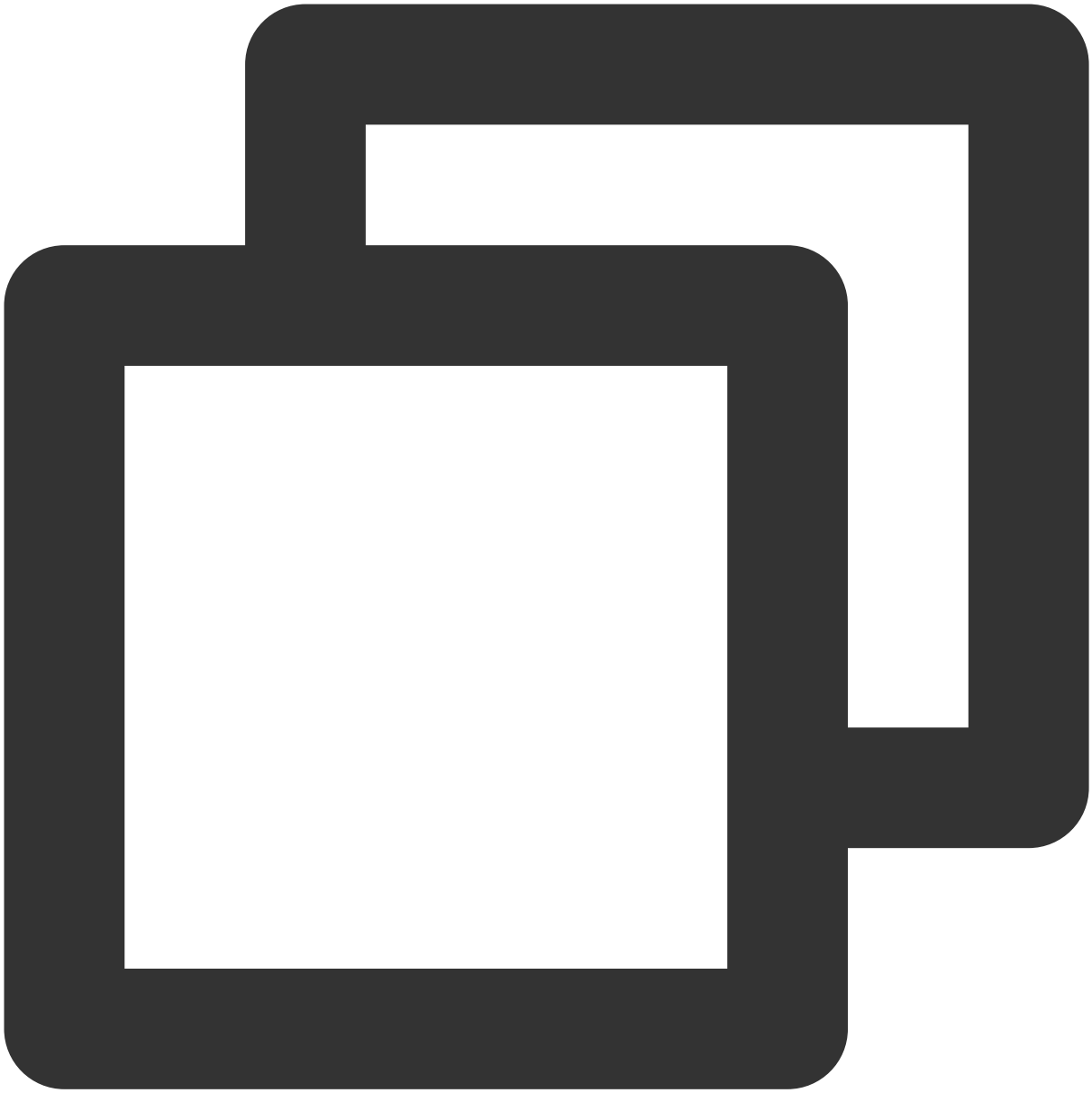
`TUIBarrageDisplayView` ：用于展示弹幕消息。

在需要发送弹幕的场景，创建 `TUIBarrageButton` ，点击后可以拉起输入界面：



```
TUIBarrageButton barrageButton = new TUIBarrageButton(mContext, roomId);
mBarrageButtonContainer.addView(barrageButton);
```

在需要展示弹幕的场景，使用 `TUIBarrageDisplayView` 来展示弹幕消息：

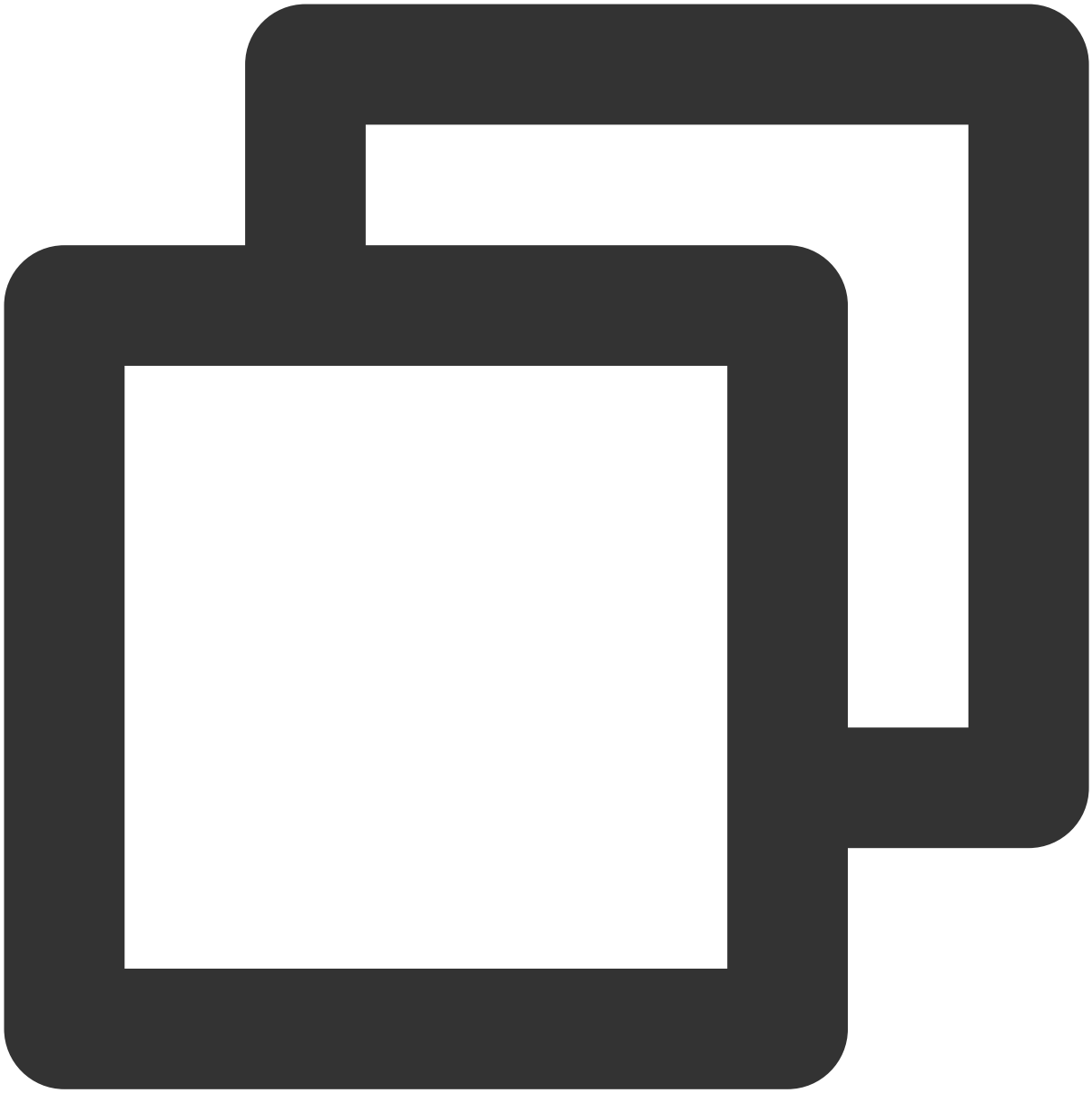


```
TUIBarrageDisplayView barrageDisplayView = new TUIBarrageDisplayView(mContext, room
mLayoutBarrageContainer.addView(barrageDisplayView);
```

自定义消息样式

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `setAdapter` 和 `TUIBarrageDisplayAdapter` ，用于自定义弹幕消息 `Item` 样式：

```
public interface TUIBarrageDisplayAdapter {
    RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType);

    void onBindViewHolder(RecyclerView.ViewHolder holder, TUIBarrage barrage);

    int getItemViewType(int position, TUIBarrage barrage);
}
```



```
public class GiftBarrageAdapter implements TUIBarrageDisplayAdapter {
    private final Context       mContext;

    public GiftBarrageAdapter(Context context) {
        mContext = context;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewTyp
        if (viewType == GIFT_VIEW_TYPE_1) {
```

```
            // 自定义样式1的处理
            LinearLayout ll = new LinearLayout(mContext);
            ll.addView(new TextView(mContext));
            return new GiftViewHolder(ll);
        }
        return null;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, TUIBarrage barrage
        if (holder instanceof GiftViewHolder) {
            GiftViewHolder viewHolder = (GiftViewHolder) holder;
            viewHolder.bind(barrage);
        }
    }

    @Override
    public int getItemViewType(int position, TUIBarrage barrage) {
        if (...) { // 如果当前 barrage 需要自定义 Item 样式，则返回对应样式type
            return GIFT_VIEW_TYPE_1;
        }
        return 0; // 0 表示使用默认样式
    }

    private static class GiftViewHolder extends RecyclerView.ViewHolder {

        public GiftViewHolder(View itemView) {
            super(itemView);
            // ...
        }

        public void bind(TUIBarrage barrage) {
            // ...
        }
    }
}

// 设置自定义 Adapter
barrageDisplayView.setAdapter(new GiftBarrageAdapter(mContext));
```

`TUIBarrage` 定义如下：

```
public class TUIBarrage {
    public final TUIBarrageUser    user = new TUIBarrageUser(); //发送内容
    public String                  content;                     //扩展信息
    public HashMap<String, Object> extInfo = new HashMap<>();   //扩展信息
}

public class TUIBarrageUser {
    public String userId;
    public String userName;
    public String avatarUrl;
```

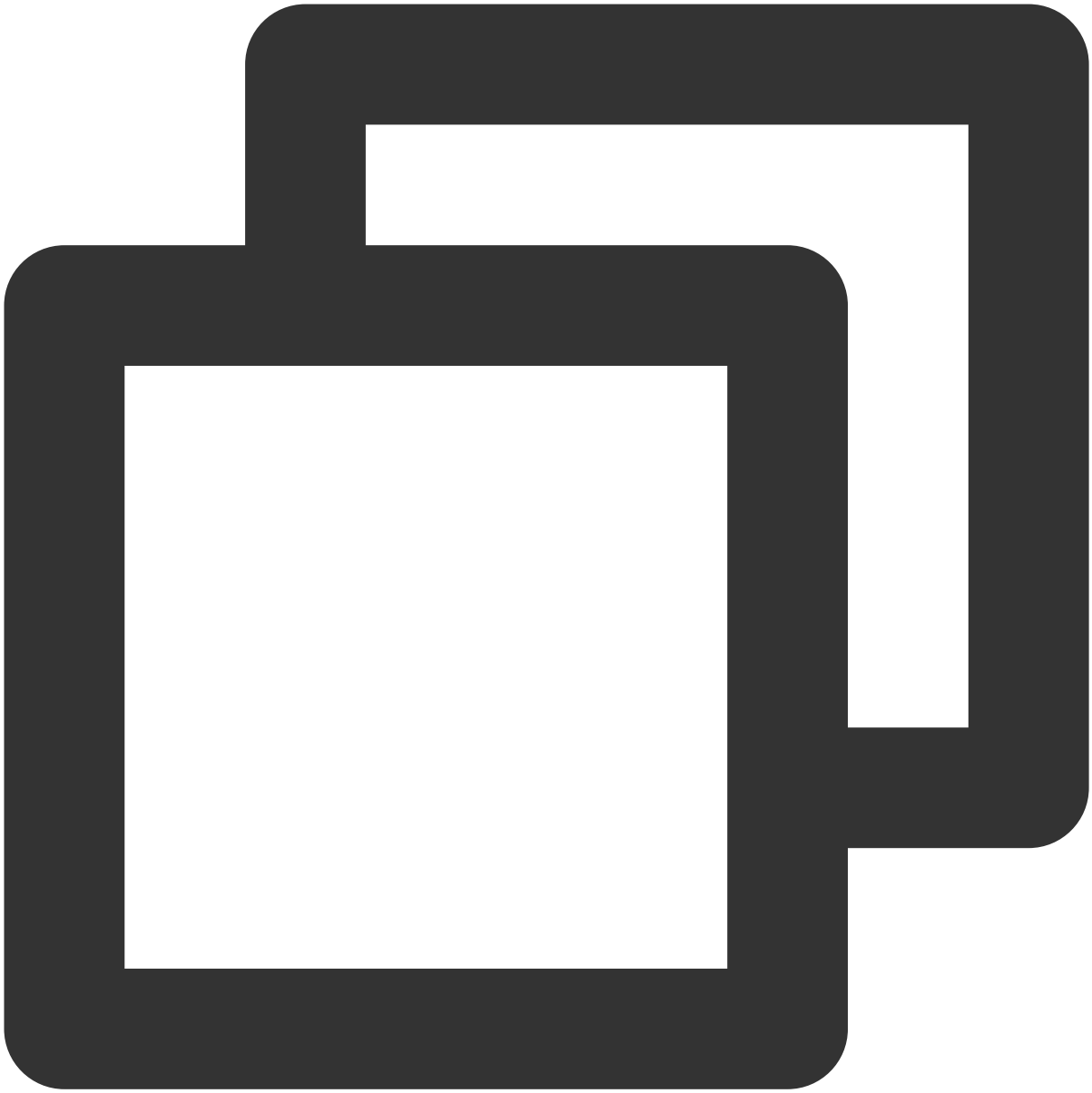
```
    public String level;
}
```

说明：

支持多种自定义样式（通过 `getItemViewType` 指定多种返回值），0 表示默认样式。

插入自定义消息

弹幕展示组件 `TUIBarrageDisplayView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息，通常自定义消息配合自定义样式，实现不一样的展示效果。



```
// 示例：在弹幕区插入一条礼物消息
TUIBarrage barrage = new TUIBarrage();
barrage.content = "gift";
barrage.user.userId = sender.userId;
barrage.user.userName = sender.userName;
barrage.user.avatarUrl = sender.avatarUrl;
barrage.user.level = sender.level;
barrage.extInfo.put(Constants.GIFT_VIEW_TYPE, GIFT_VIEW_TYPE_1);
barrage.extInfo.put(GIFT_NAME, barrage.giftName);
barrage.extInfo.put(GIFT_COUNT, giftCount);
barrage.extInfo.put(GIFT_ICON_URL, barrage.imageUrl);
```

版权所有：腾讯云计算（北京）有限责任公司

```
barrage.extInfo.put(GIFT_RECEIVER_USERNAME, receiver.userName);
barrageDisplayView.insertBarrages(barrage);
```

说明：

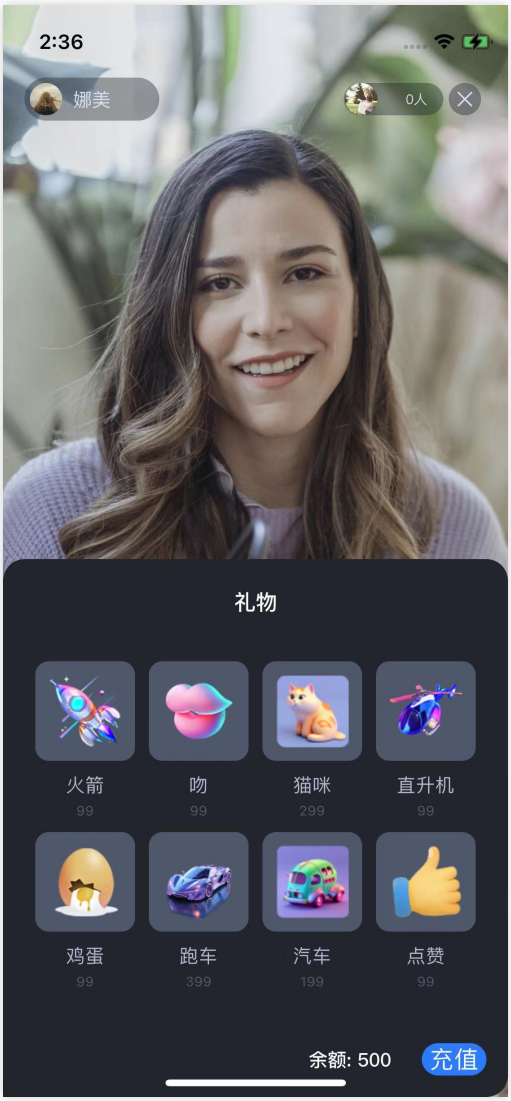
`TUIBarrage` 的 `extInfo` 是一个 `Map`，用于存放自定义数据。

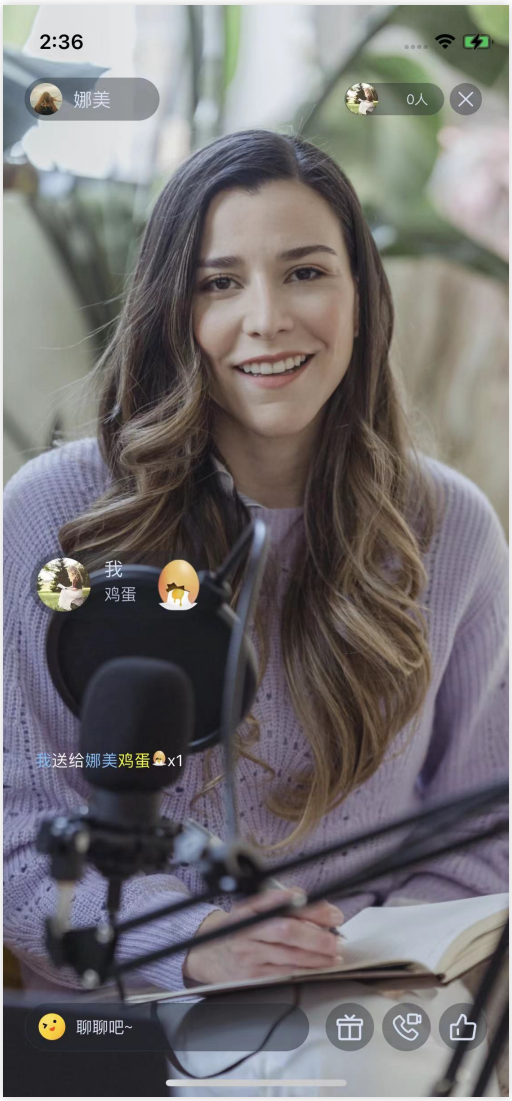
互动礼物（TUILiveKit）
iOS

最近更新时间：2024-07-10 16:31:15

互动礼物组件是一款虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。借助这一组件，用户能够向自己欣赏的直播主播送上虚拟礼物，以此展示他们的赞赏、喜爱以及支持。
互动礼物组件支持设置**礼物素材、余额显示、普通礼物播放与全屏礼物播放、充值按钮**等。

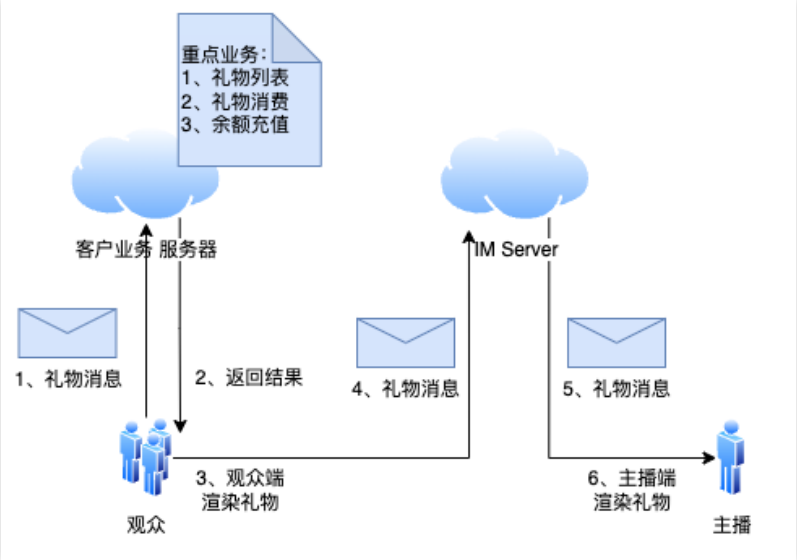
礼物展示

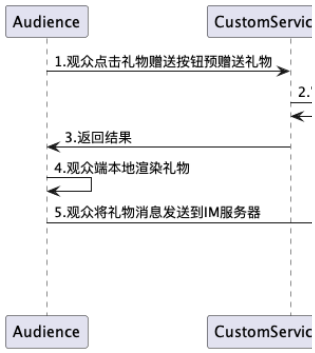




礼物展示面板	普通礼物播放效果

礼物系统





礼物系统结构图	礼物系统时序图

客户端短连接请求到自己的业务服务器，涉及到礼物计费逻辑。

1. 计费后，发送人直接看到 XXX 送了 XXX 礼物（以确保发送人自己看到自己发的礼物，消息量大的时候，可能会触发抛弃策略）。
2. 计费结算后，调用 `GiftListView.sendGift` 发送消礼物消息。

快速接入

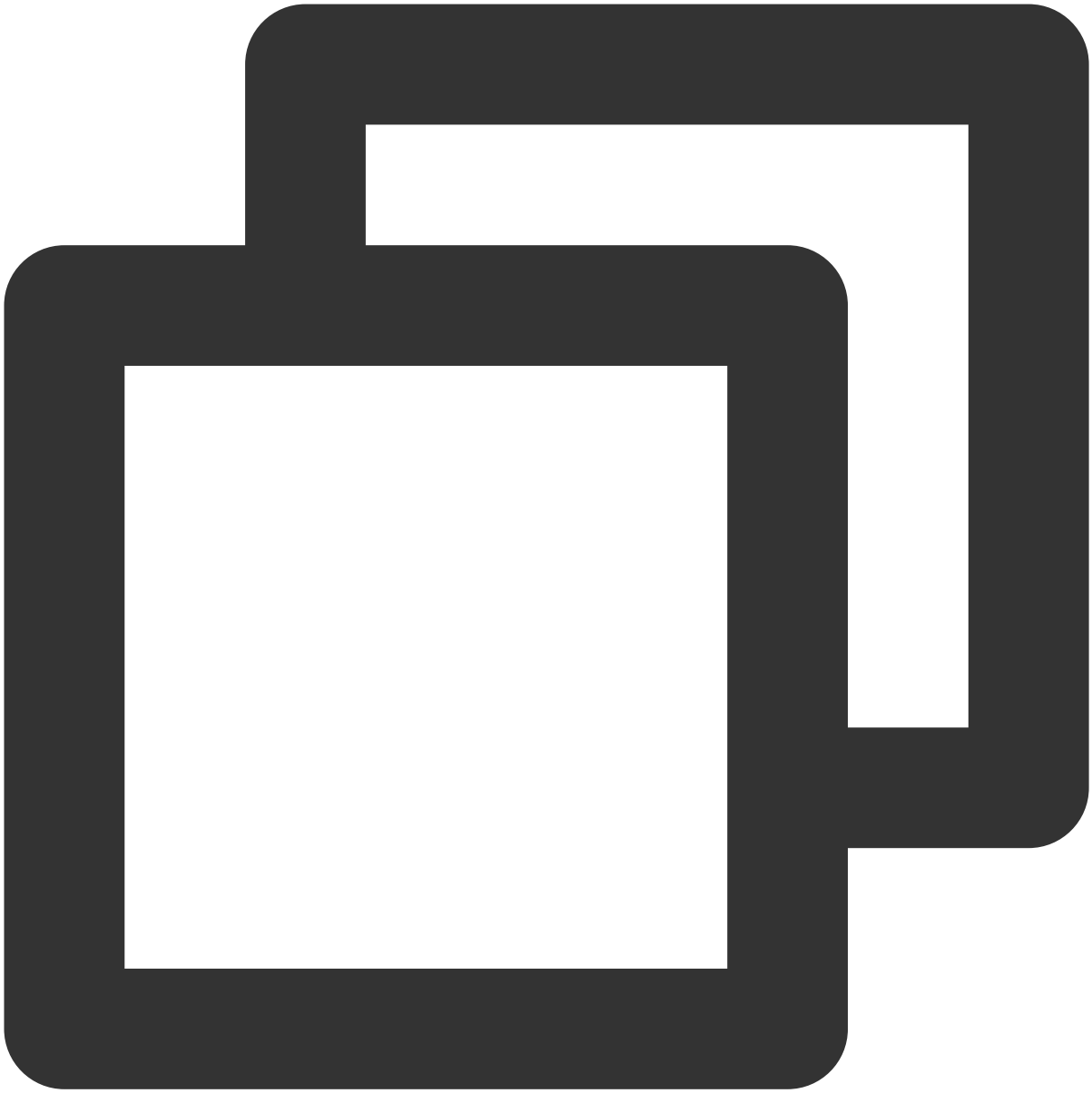
礼物组件主要提供2个 API：

`TUIGiftListView` ：礼物面板，呈现礼物列表，发送礼物及充值。

`TUIGiftPlayView` ：播放礼物的面板，自动监听礼物消息。

设置礼物素材

礼物面板组件 `TUIGiftListView` 提供了 `setGiftList` 接口，可用于设置礼物素材。



```
let giftListView: TUIGiftListView = TUIGiftListView(groupId: xxx) //生成礼物面板对象
let giftList: [TUIGift] = ... //这里可替换成自己的礼物素材数组
giftListView.setGiftList(giftList) //设置礼物面板的素材
```

说明：

`TUIGift` 的参数及说明如下：

`giftId: String`	礼物ID
`giftName: String`	礼物名称
`imageUrl: String`	礼物面板展示图像

`animationUrl: String` SVGA动画Url

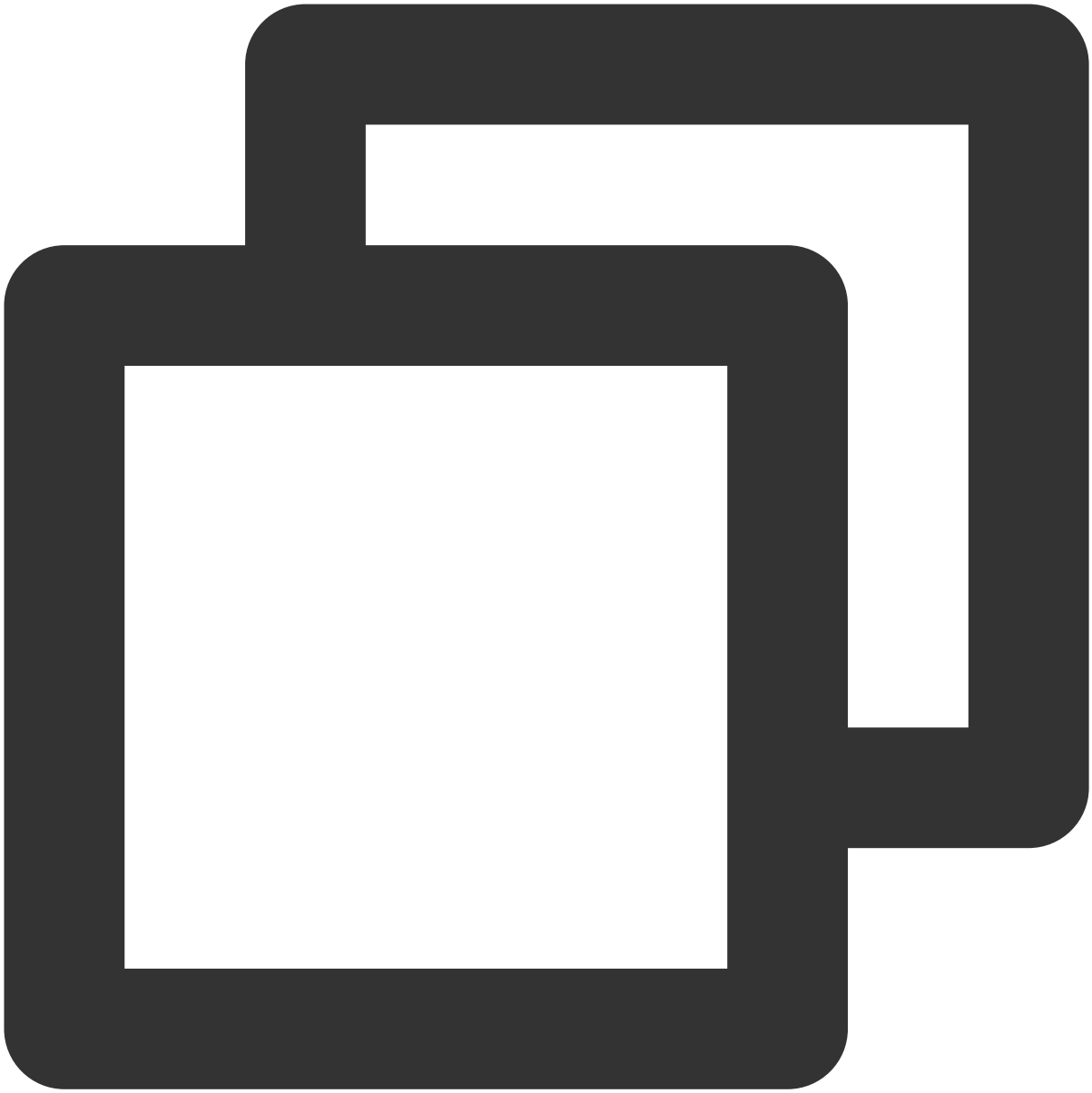
`price: Int` 礼物价格

`extInfo: [String: AnyCodable]` 自定义扩展信息

互动礼物组件支持设置自己的礼物素材。若 `animationUrl` 为空，则礼物播放效果为普通播放，播放的内容为imageUrl所链接的图像；若 `animationUrl` 不为空，则播放效果为全屏播放，播放的内容为对应的svga动画。

赠送礼物

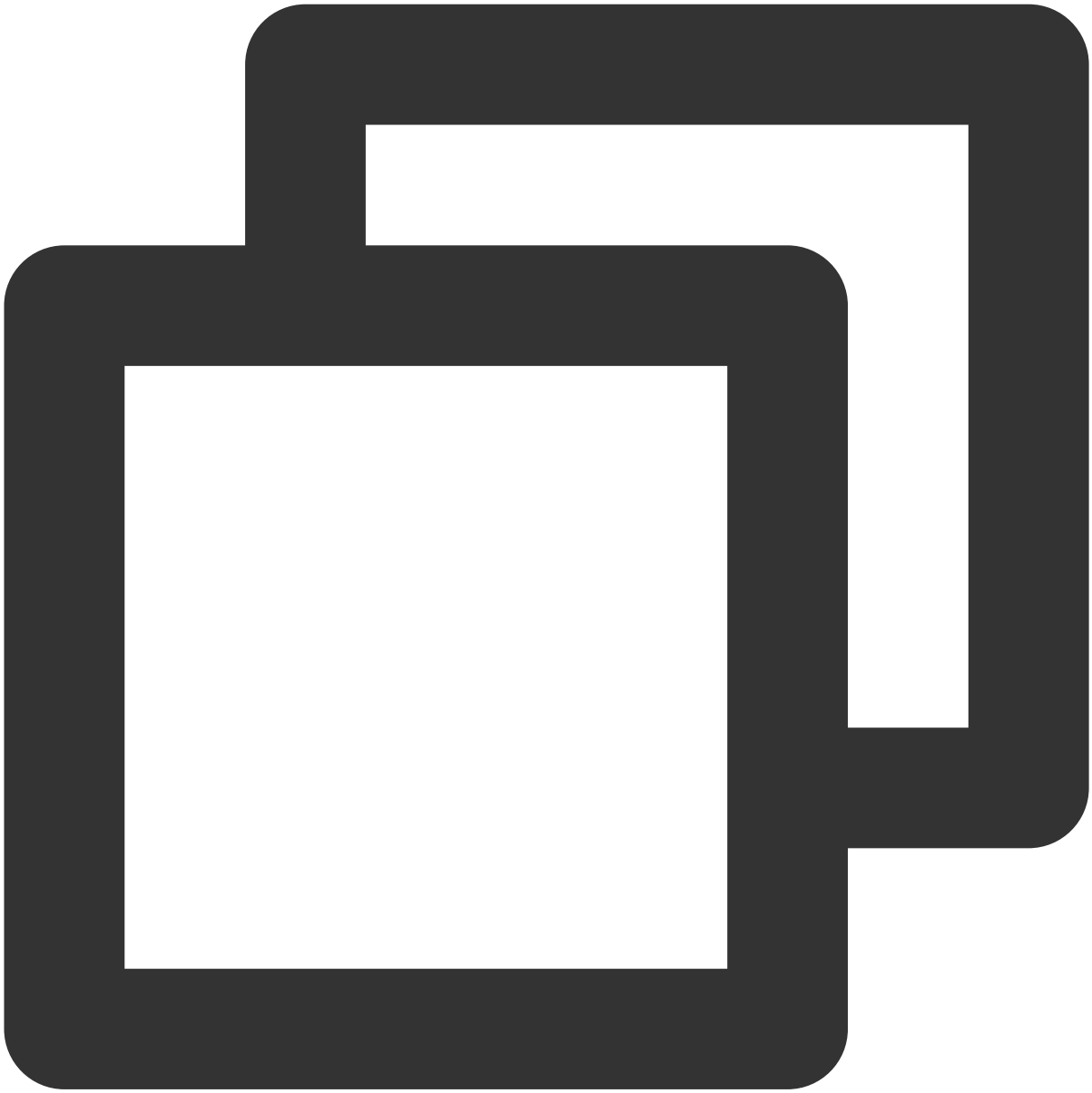
实现 `TUIGiftListView` 的代理 `TUIGiftListViewDelegate` 中的 `onSendGift` 代理函数，获取礼物个数和礼物信息，在预处理完后可调用 `TUIGiftListView` 的 `sendGift` 函数用于礼物的实际发送。



```
giftListView.delegate = self

extension ViewController: TUIGiftListViewDelegate {
    func onSendGift(giftListView view: TUIGiftListView, giftModel: TUIGift, giftCou
        //...此处为预处理，如校验当前用户的余额等操作
        let receiver = TUIGiftUser()
        //...此处设置礼物接受者信息
        view.sendGift(giftModel: giftModel, giftCount: giftCount, receiver: receive
    }
}
```

接收礼物

礼物展示组件 `TUIGiftPlayView` 自身会接收并播放礼物消息。

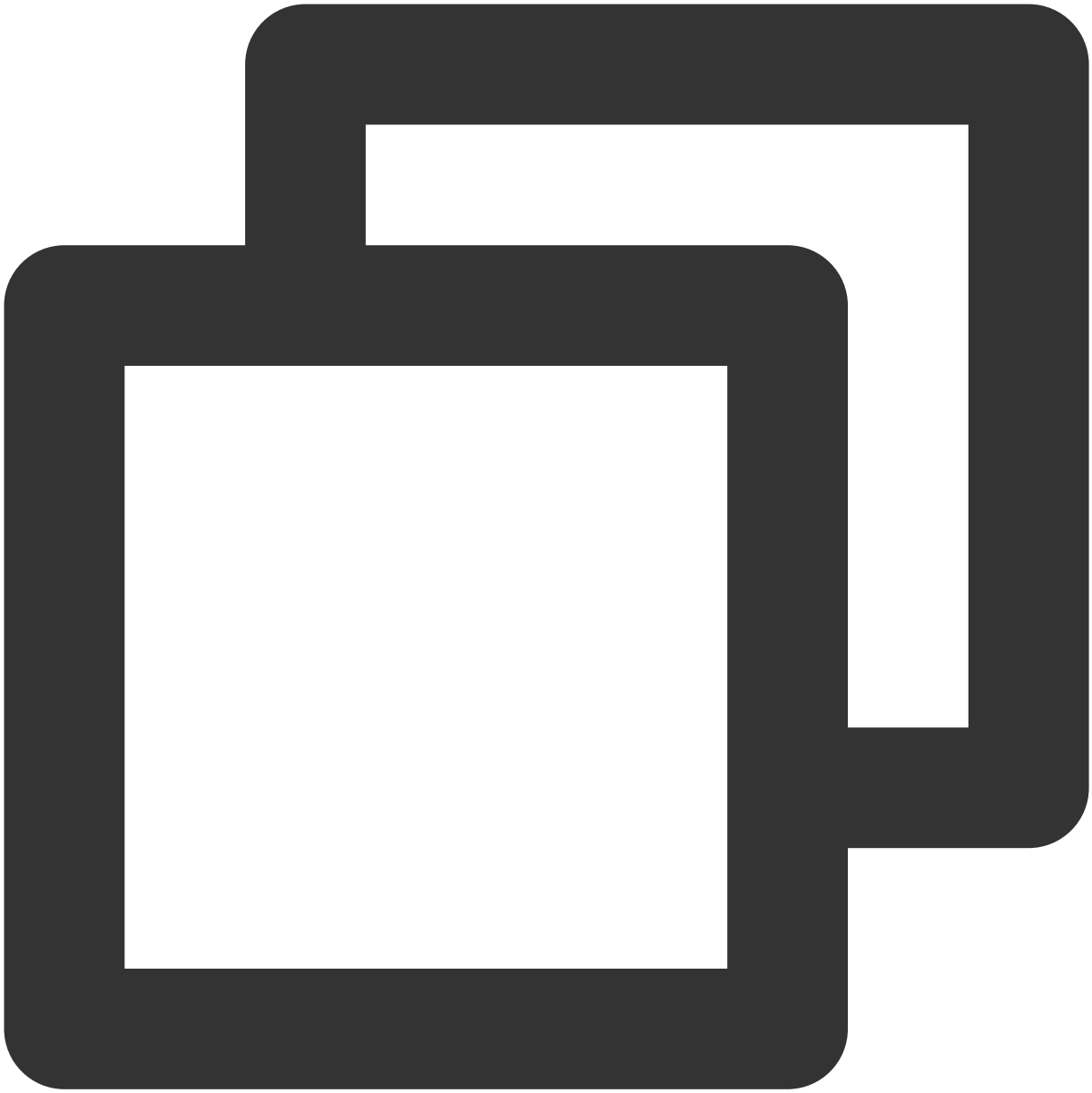


```
let giftDisplayView: TUIGiftPlayView = TUIGiftPlayView(groupId:xxx)
```

说明：

`TUIGiftPlayView` 需要全屏接入。

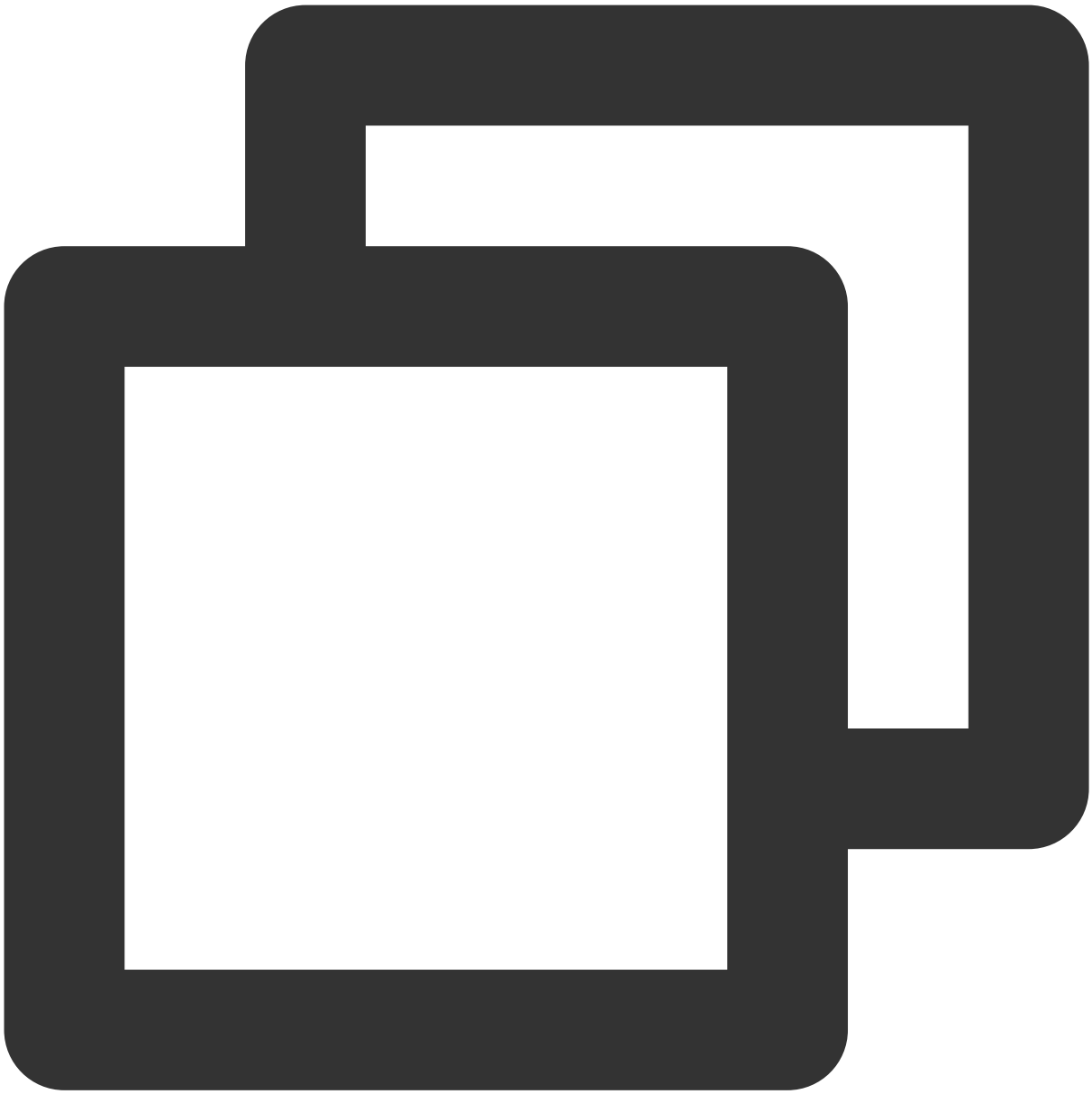
若需要获取接收礼物的回调信息，可将实现 `TUIGiftPlayView` 的代理 `TUIGiftPlayViewDelegate` 中的 `giftPlayView:onPlayGift` 代理函数。



```
extension ViewController: TUIGiftPlayViewDelegate {
    func giftPlayView(_ giftPlayView: TUIGiftPlayView, onPlayGift gift: TUIGift, gi
     //...可在此处自行处理
    }
}
```

设置余额

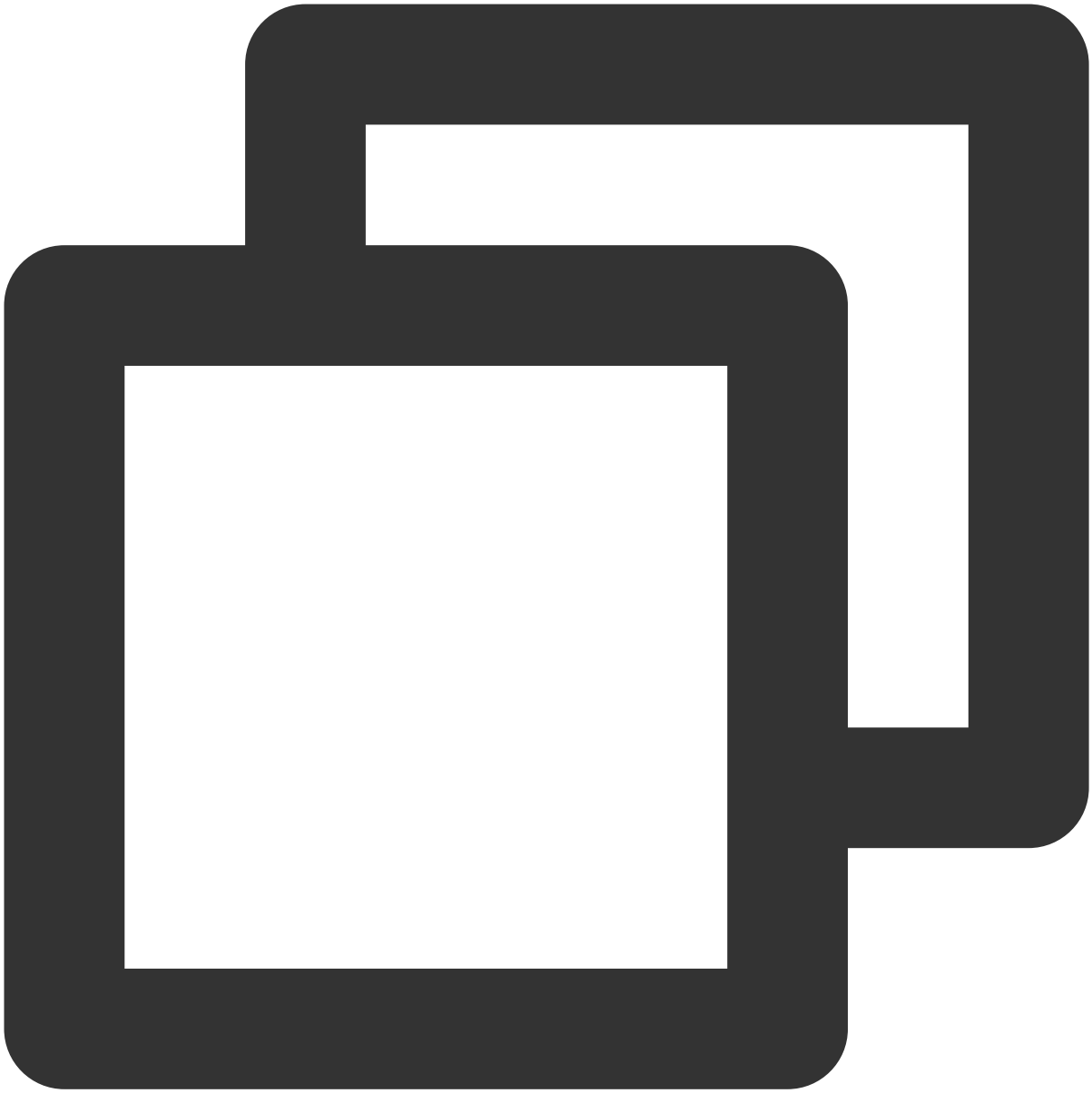
礼物面板组件 `TUIGiftListView` 提供了 `setBalance` 接口，可用于设置礼物面板上显示的余额值。



```
giftListView.setBalance(xxx)
```

充值

实现 `TUIGiftListView` 的代理 `TUIGiftListViewDelegate` 中的 `onRecharge` 代理函数可用于接收礼物展示面板抛出的充值按钮点击事件，在这里对接自己的充值系统。



```
extension ViewController: TUIGiftListViewDelegate {
   func onRecharge(giftListView view: TUIGiftListView) {
        //...此处可用于对接自己的充值系统，充值完毕后，可调用view.setBalance(xxx)设置礼物展示
    }
}
```

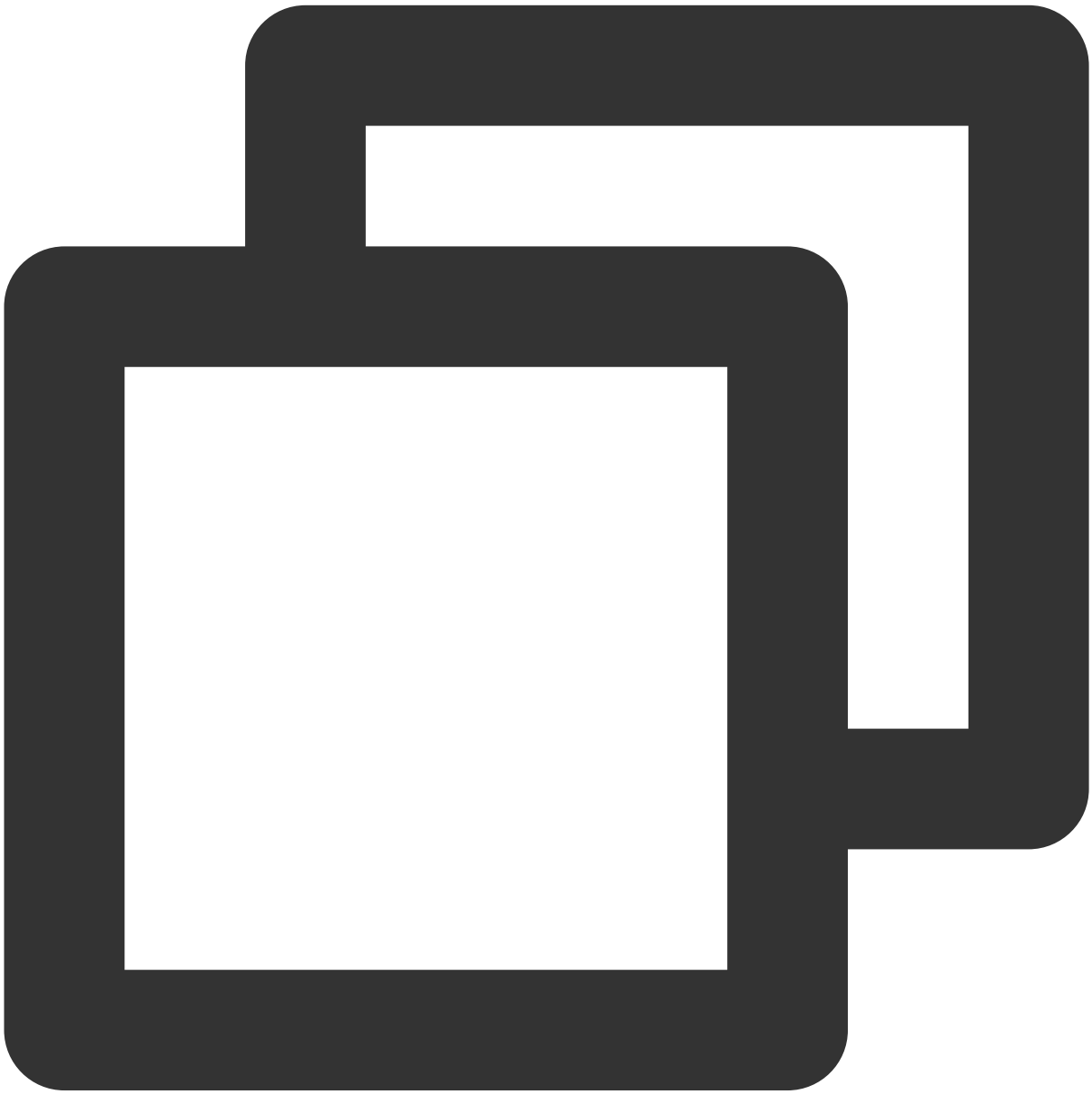
注意：

礼物余额是个虚拟币的概念，并不是真实货币。

礼物充值逻辑，由外部实现，客户可以接入自己的充值系统。充值完毕后再更新礼物余额。

计费统计

实现 `TUIGiftListView` 的 `TUIGiftListViewDelegate` 中的 `onSendGift` 代理函数，连接客户自己的业务服务器，完成余额校验、礼物计费、消费统计后，再调用 `TUIGiftListView` 的 `sendGift` 发送礼物消息。

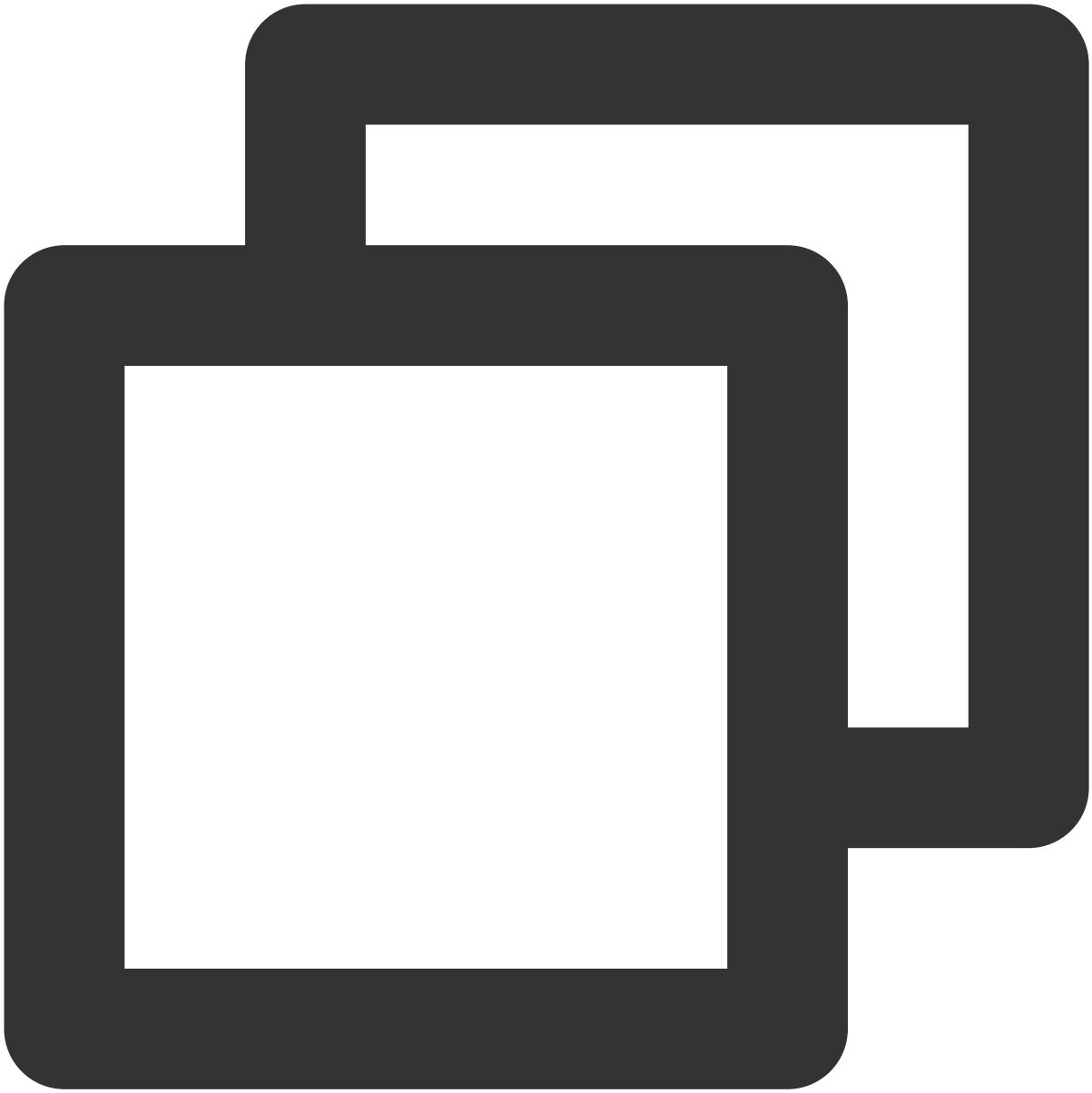


```
giftListView.delegate = self

extension ViewController: TUIGiftListViewDelegate {
    func onSendGift(giftListView view: TUIGiftListView, giftModel: TUIGift, giftCou
```

```
        //...此处连接客户自己的业务服务器，完成余额校验、礼物计费、消费统计等
        let receiver = TUIGiftUser()
        //...此处设置礼物接受者信息
        view.sendGift(giftModel: giftModel, giftCount: giftCount, receiver: receive
    }
}
```

自定义礼物列表

修改观众端礼物面板的礼物列表：



```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw

private lazy var giftPanelView: TUIGiftListView = {
    let view = TUIGiftListView(groupId: liveRoomInfo.roomId.value)
    giftCloudServer.queryGiftInfoList { [weak self] error, giftList in
        guard let self = self else { return }
        DispatchQueue.main.async {
            if error == .noError {
                view.setGiftList(giftList)
            } else {
                self.makeToast("query gift list error, code = \\(error)")
```

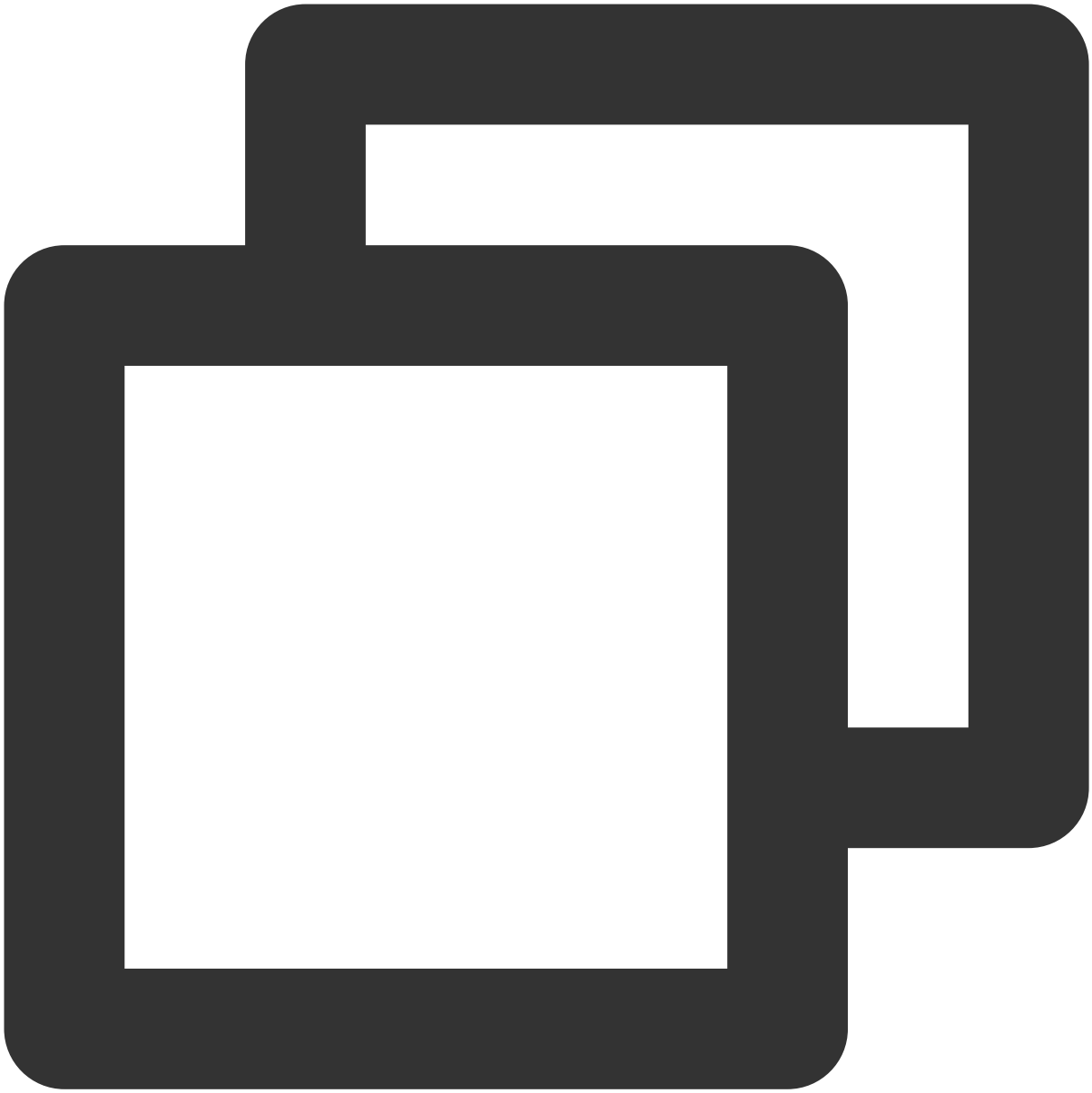
```
                }
            }
        }
        return view
    }()
```

说明：

客户自行实现 `giftCloudServer.queryGiftInfoList` 的逻辑，得到一个自定义的礼物列表 `[TUIGift]`，通过 `view.setGiftList` 设置礼物列表即可。

礼物的 `animationUrl 要求是一个SVGA动画。`

自定义礼物余额充值



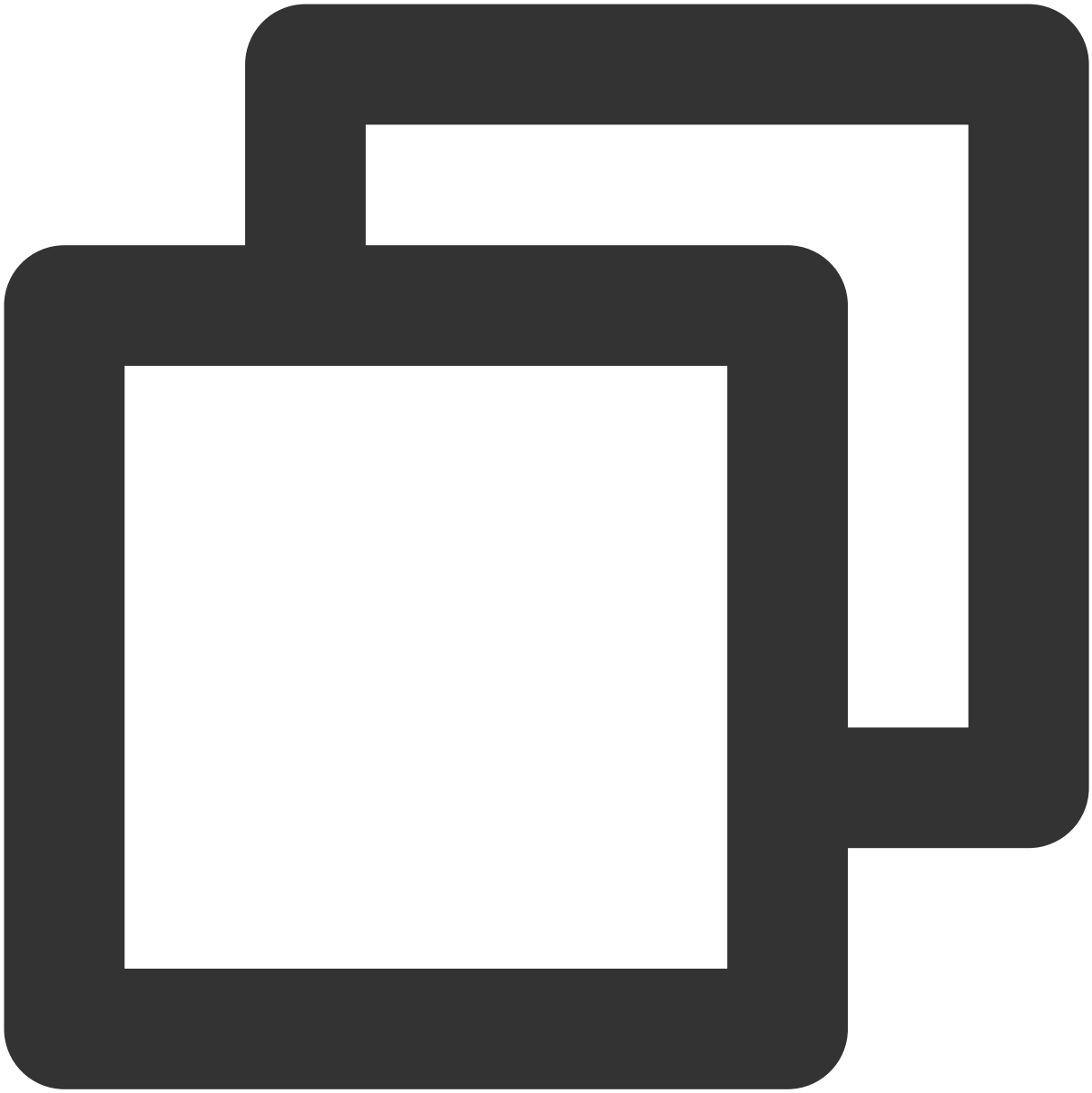
```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw

private lazy var giftPanelView: TUIGiftListView = {
    let view = TUIGiftListView(groupId: liveRoomInfo.roomId.value)
    giftCloudServer.queryBalance { [weak self] error, balance in
        guard let self = self else { return }
        DispatchQueue.main.async {
            if error == .noError {
                view.setBalance(balance)
            } else {
                self.makeToast("query balance error, code = \\(error)")
```

```
                }
            }
        }
        return view
    }()
```

说明：

客户自行实现 `giftCloudServer.queryBalance` 的逻辑，得到礼物余额，通过 `view.setBalance` 更新礼物余额即可。

自定义礼物消费逻辑



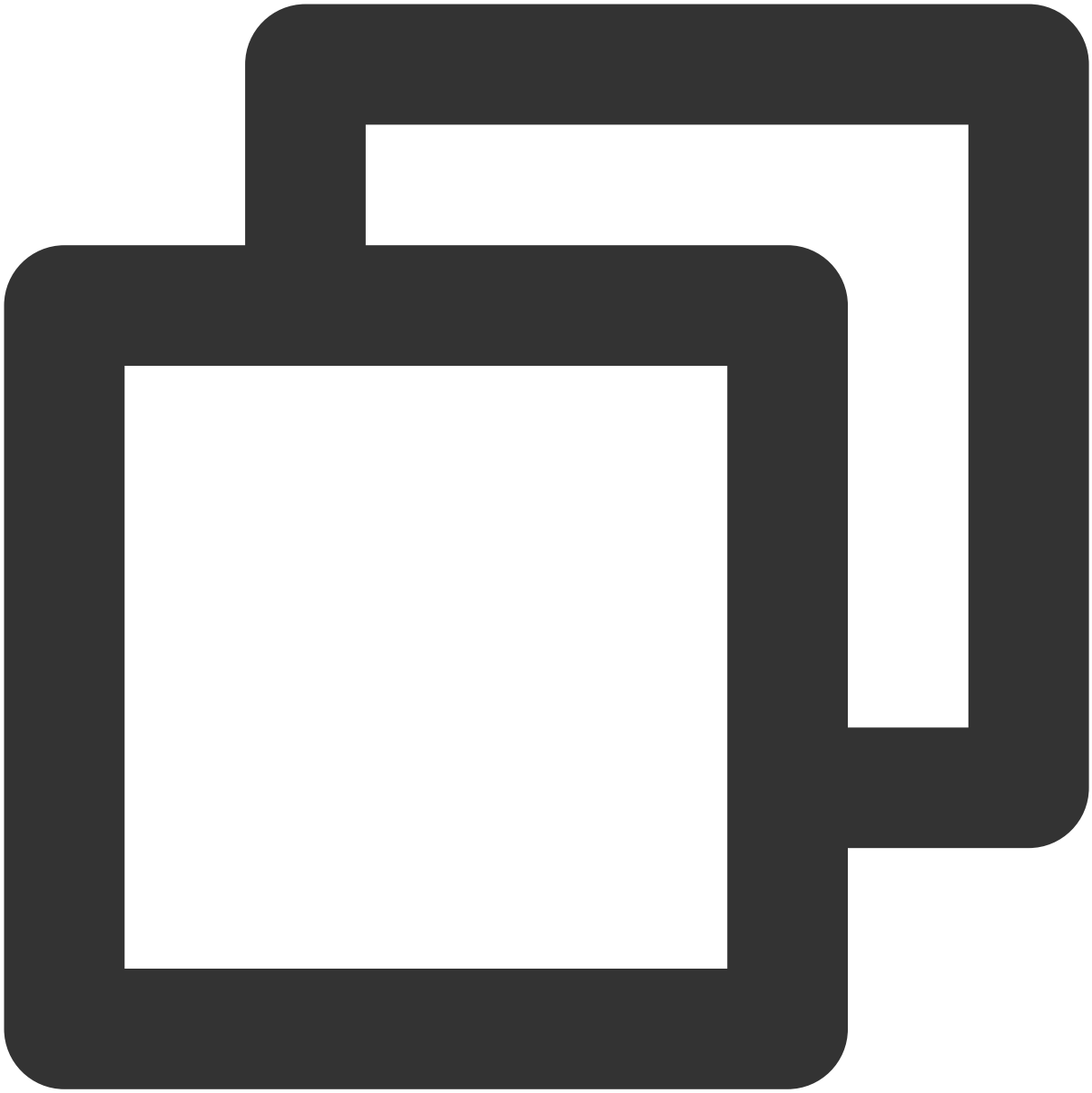
```
// 源码路径：TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.sw

func onSendGift(giftListView view: TUIGiftListView, giftModel: TUIGift, giftCount:
    let receiver = TUIGiftUser()
    receiver.userId = liveRoomInfo.anchorInfo.value.userId
    receiver.userName = liveRoomInfo.anchorInfo.value.name.value
    receiver.avatarUrl = liveRoomInfo.anchorInfo.value.avatarUrl.value
    receiver.level = "0"
    giftCloudServer.sendGift(sender: TUILogin.getUserID() ?? "",

                             receiver: receiver.userId,
```

```
                                giftModel: giftModel,

                                giftCount: giftCount) { [weak self] error, balance in
            guard let self = self else { return }
            if error == .noError {
                view.sendGift(giftModel: giftModel, giftCount: giftCount, receiver: rec
                view.setBalance(balance)
            } else {
                self.makeToast(.balanceInsufficientText)
            }
        }
    }
```

说明：

客户自行实现 `giftCloudServer.sendGift` 的逻辑，主要流程是，首先连接客户自己的业务服务器，进行余额校验，校验通过后，由服务器进行计费并统计消费记录，最后将结果回调给客户端。客户端收到成功回调后，通过 `GiftListView` 的 `sendGift` 发送礼物消息，之后再通过 `setBalance` 更新礼物余额。

自定义加载礼物动画并播放



```
// 源码路径：
// TUILiveKit/Source/LiveRoom/View/Audience/Component/AudienceLivingView.swift
// TUILiveKit/Source/LiveRoom/View/Anchor/Living/AudienceLivingView.swift

func giftPlayView(_ giftPlayView: TUIGiftPlayView, onPlayGiftAnimation gift: TUIGif
    giftCacheService.request(urlString: gift.animationUrl) { error, data in
        guard let data = data else { return }
        if error == 0 {
            DispatchQueue.main.async {
                giftPlayView.playGiftAnimation(animationData: data)
```

```
                }
            }
        }
    }
```

说明：

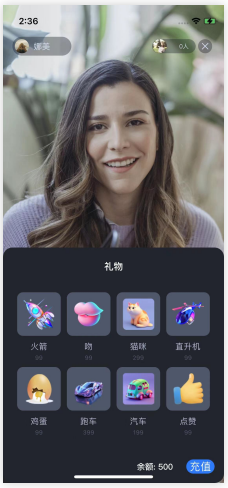
客户自行实现 `giftCacheService.request` 的逻辑，加载动画成功得到 `data` （ `Data` 类型），然后通过 `TUIGiftPlayView` 的 `playGiftAnimation` 播放礼物动画。

Android

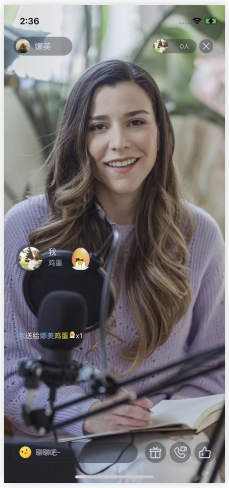
最近更新时间：2024-07-10 16:31:15

互动礼物组件是一款虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。借助这一组件，用户能够向自己欣赏的直播主播送上虚拟礼物，以此展示他们的赞赏、喜爱以及支持。
互动礼物组件支持设置**礼物素材、余额显示、普通礼物播放与全屏礼物播放、充值按钮**等。

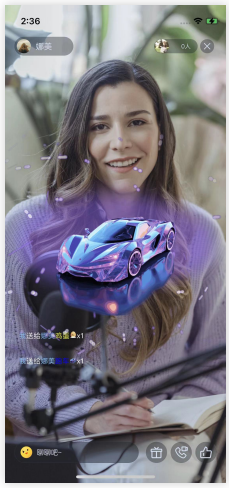
礼物展示



礼物展示面板

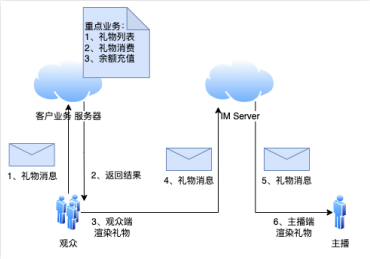


普通礼物播放效果

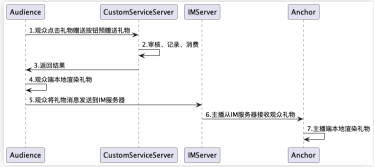


全屏礼物播放效果

礼物系统



礼物系统结构图



礼物系统时序图

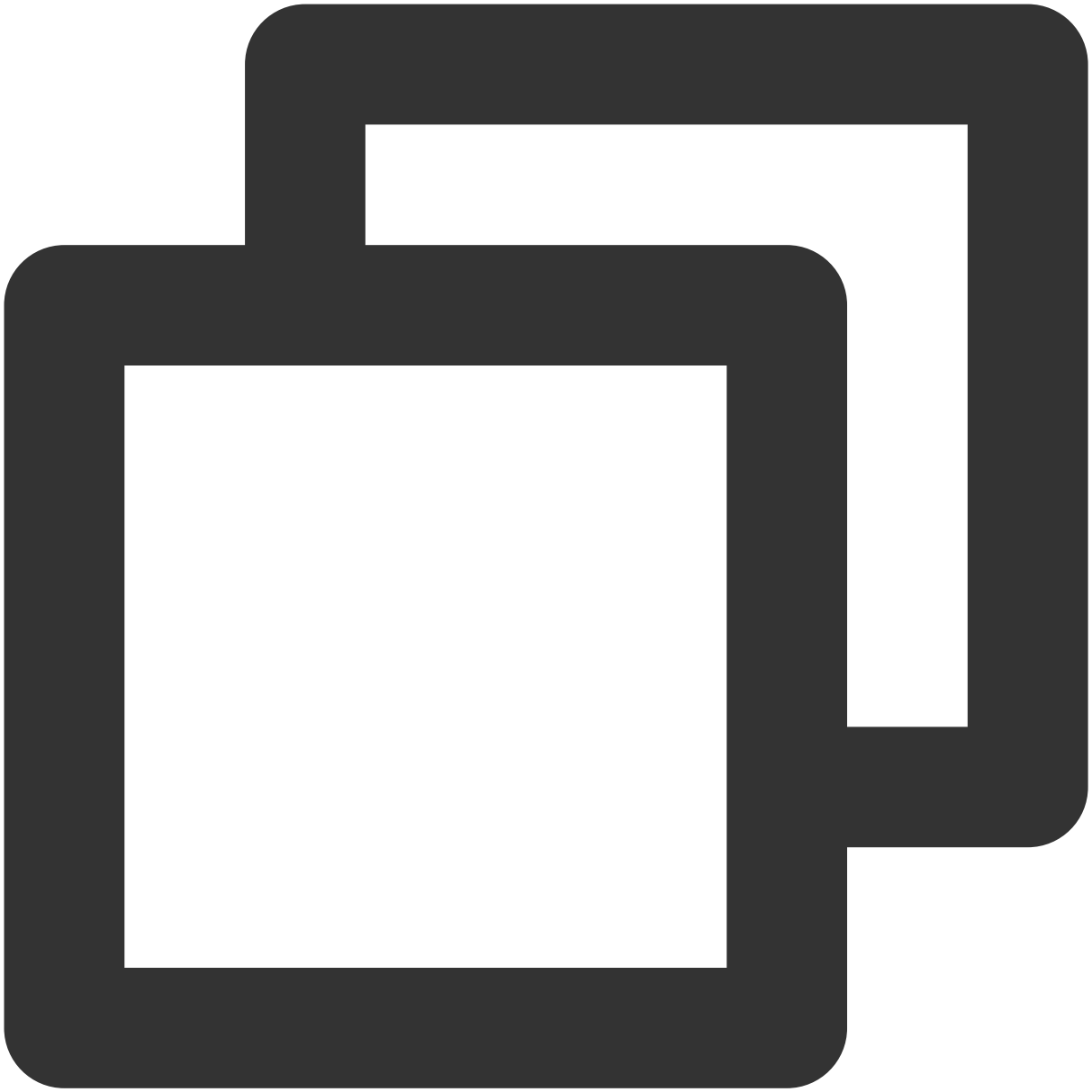
客户端短连接请求到自己的业务服务器，涉及到礼物计费逻辑。
1. 计费后，发送人直接看到 XXX 送了 XXX 礼物（以确保发送人自己看到自己发的礼物，消息量大的时候，可能会触发抛弃策略）。
2. 计费结算后，调用 `GiftListView.sendGift` 发送消礼物消息。

快速接入

礼物组件主要提供2个 API：
`TUIGiftListView` ：礼物面板，呈现礼物列表，发送礼物及充值。
`TUIGiftPlayView` ：播放礼物的面板，自动监听礼物消息。

设置礼物素材

礼物面板组件 `TUIGiftListView` 提供了 `setGiftList` 接口，可用于设置礼物素材。



```
TUIGiftListView giftListView = new TUIGiftListView(mContext, roomId); //生成礼物面板对
List<GiftModel> giftList = new ArrayList<>() //这里可替换成自己的礼物素材数组
giftListView.setGiftList(giftList) //设置礼物面板的素材
```

说明：

`TUIGift` 的参数及说明如下：

`giftId: String` ： 礼物 ID

`giftName: String` ： 礼物名称

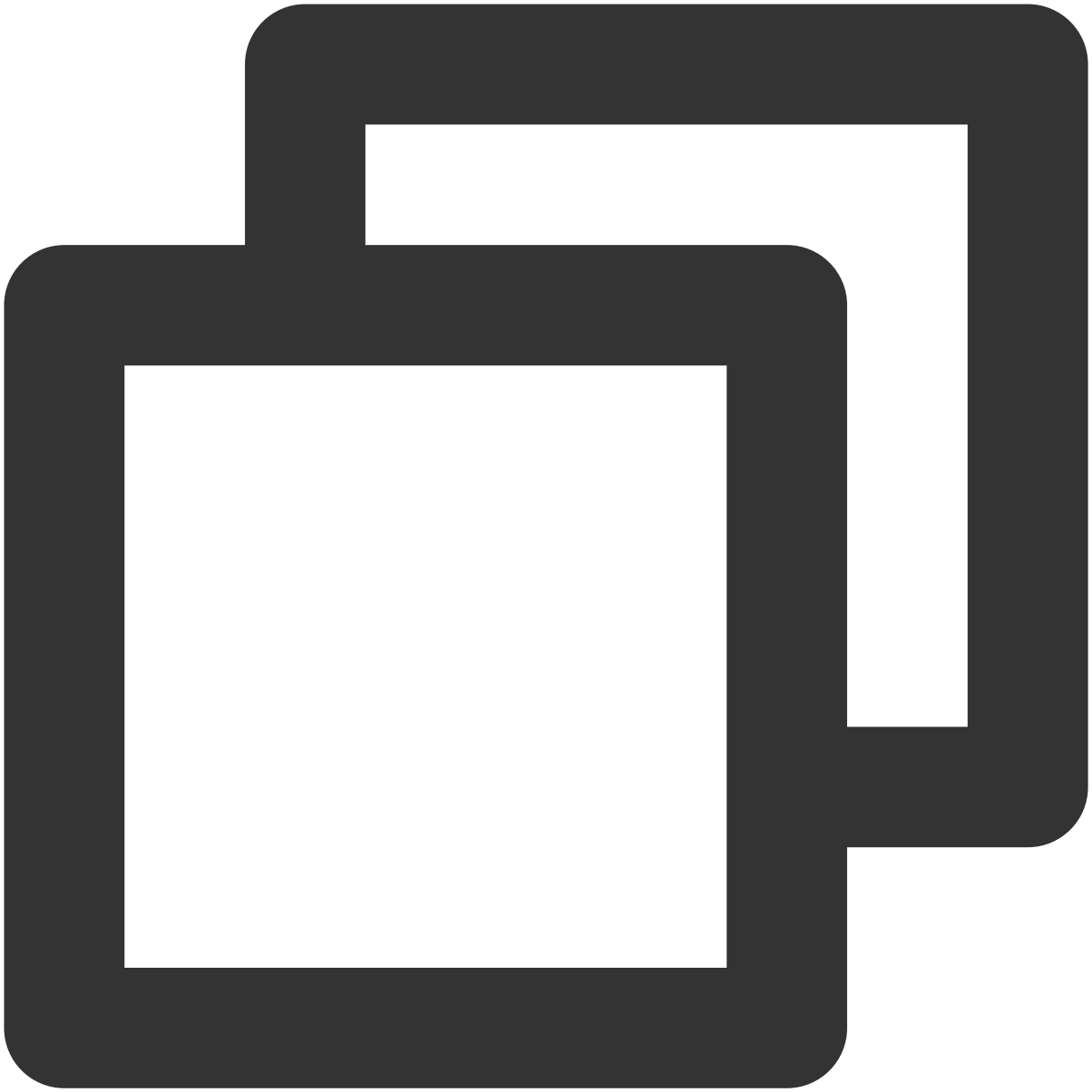
`imageUrl: String` ： 礼物面板展示图像

`animationUrl: String` ： SVGA 动画 Url
`price: int` ： 礼物价格
`extInfo: <String, Object>` ： 自定义扩展信息

互动礼物组件支持设置自己的礼物素材。若 `animationUrl` 为空，则礼物播放效果为普通播放，播放的内容为 imageUrl 所链接的图像，效果如图2；若 `animationUrl` 不为空，则播放效果为全屏播放，播放的内容为对应的 svga 动画，效果如图3。

赠送礼物

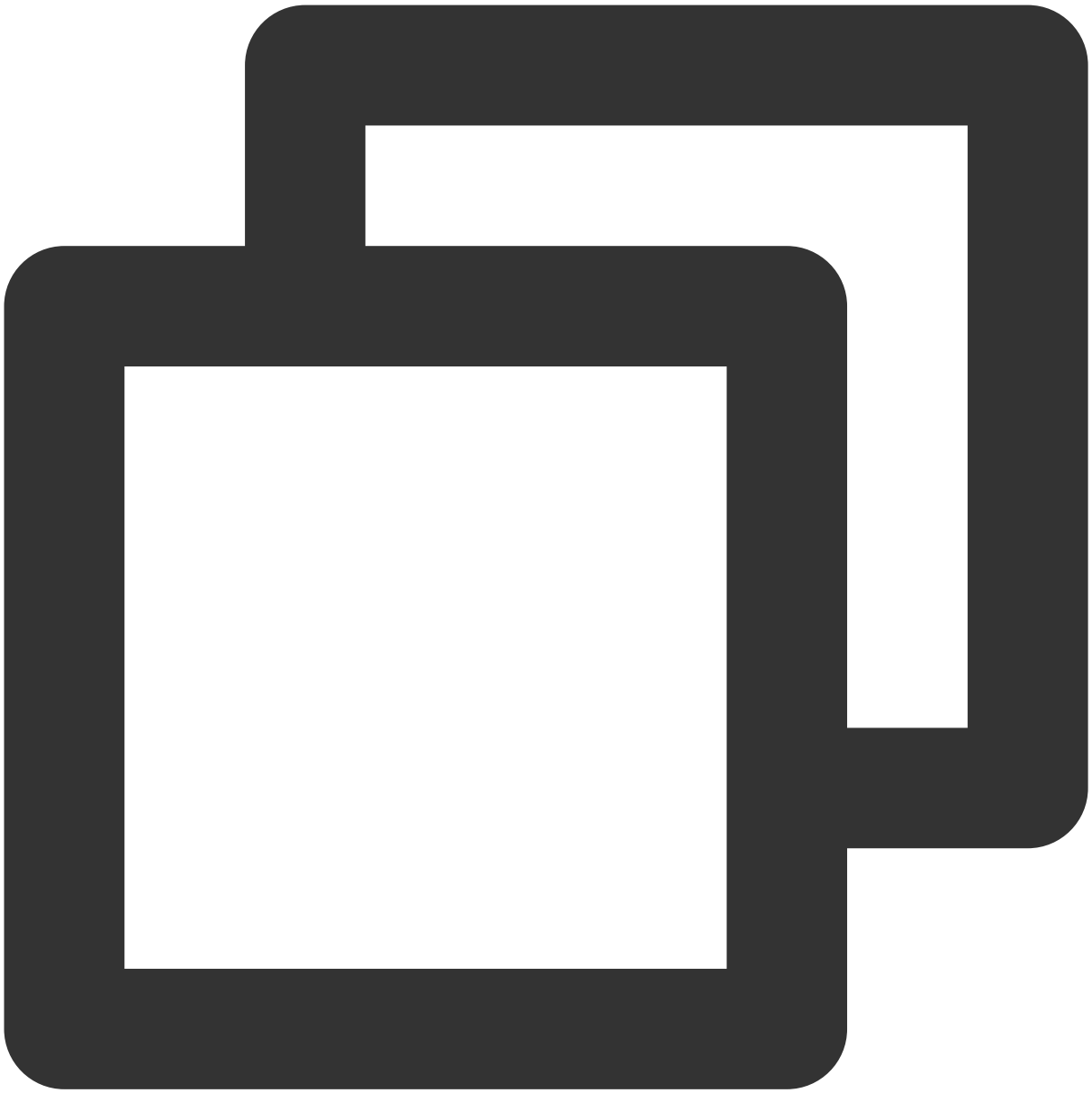
实现 `TUIGiftListView` 的 `OnGiftListener` 中的 `onSendGift` 回调，获取礼物个数和礼物信息，在预处理完后可调用 `TUIGiftListView` 的 `sendGift` 函数用于礼物的实际发送。



```
public void onSendGift(TUIGiftListView giftListView, TUIGift gift, int giftCount) {
    //...此处为预处理，如校验当前用户的余额等操作
    TUIGiftUser receiver = new TUIGiftUser();
    //...此处设置礼物接受者信息
    giftListView.sendGift(gift, giftCount, receiver);
}
```

接收礼物

礼物展示组件 `TUIGiftPlayView` 自身会接收并播放礼物消息。

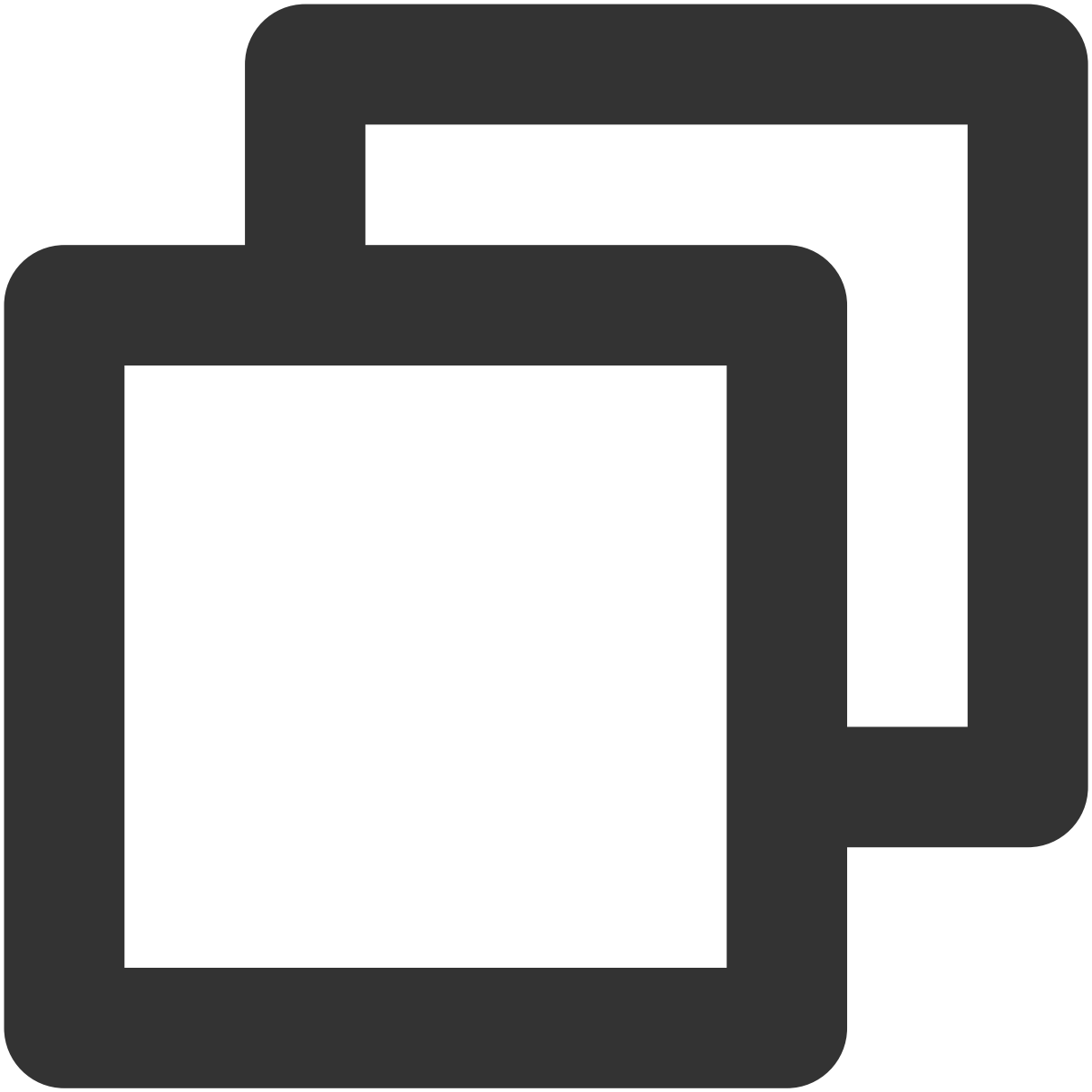


```
TUIGiftPlayView giftPlayView = new TUIGiftPlayView(mContext, roomId);
```

说明：

`TUIGiftPlayView` 需要全屏接入。

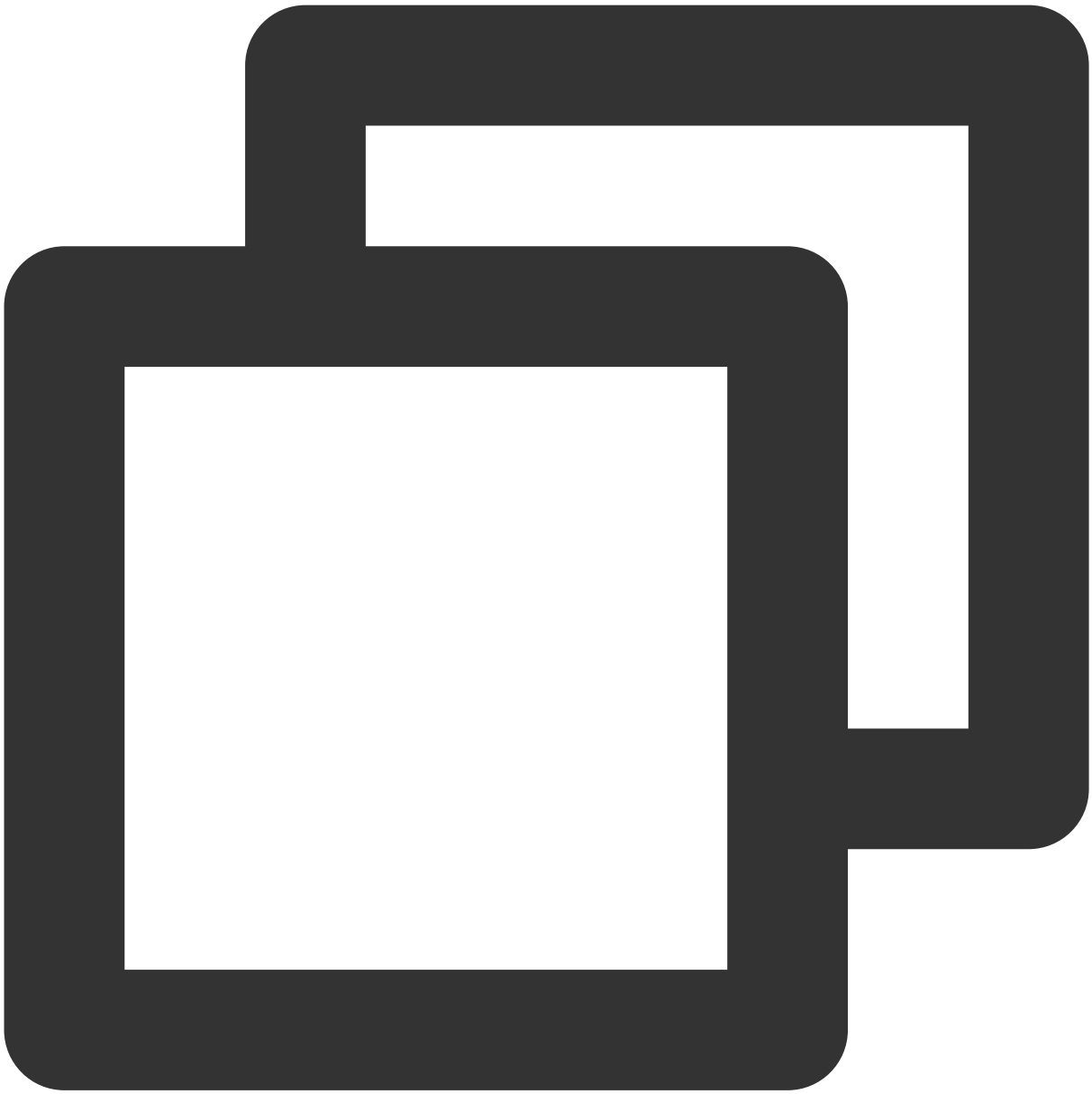
若需要获取接收礼物的回调信息，可实现 `TUIGiftPlayView` 的 `TUIGiftPlayViewListener` 中的 `onReceiveGift` 函数。



```
public interface TUIGiftPlayViewListener {
    void onReceiveGift(TUIGift gift, TUIGiftUser sender, TUIGiftUser receiver, int
    //...
}
```

播放礼物动画

需要主动调用 `TUIGiftPlayView` 的 `playGiftAnimation` 进行动画播放，调用时机是在收到 `TUIGiftPlayViewListener` 的 `onPlayGiftAnimation` 回调。



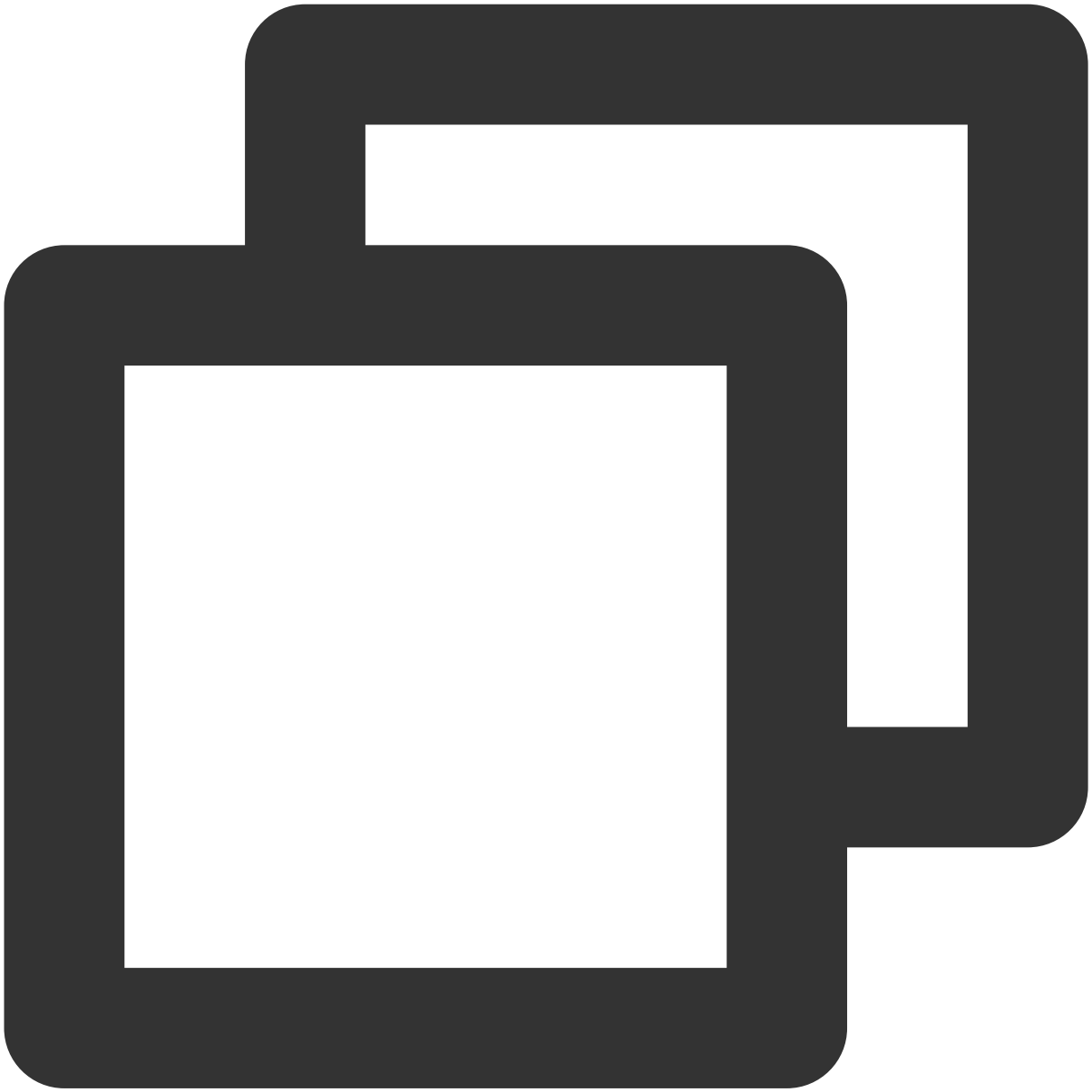
```
public interface TUIGiftPlayViewListener {
    void onPlayGiftAnimation(TUIGiftPlayView view, TUIGift gift);
    //...
}
```

说明：

仅支持 SVGA 动画。

设置余额

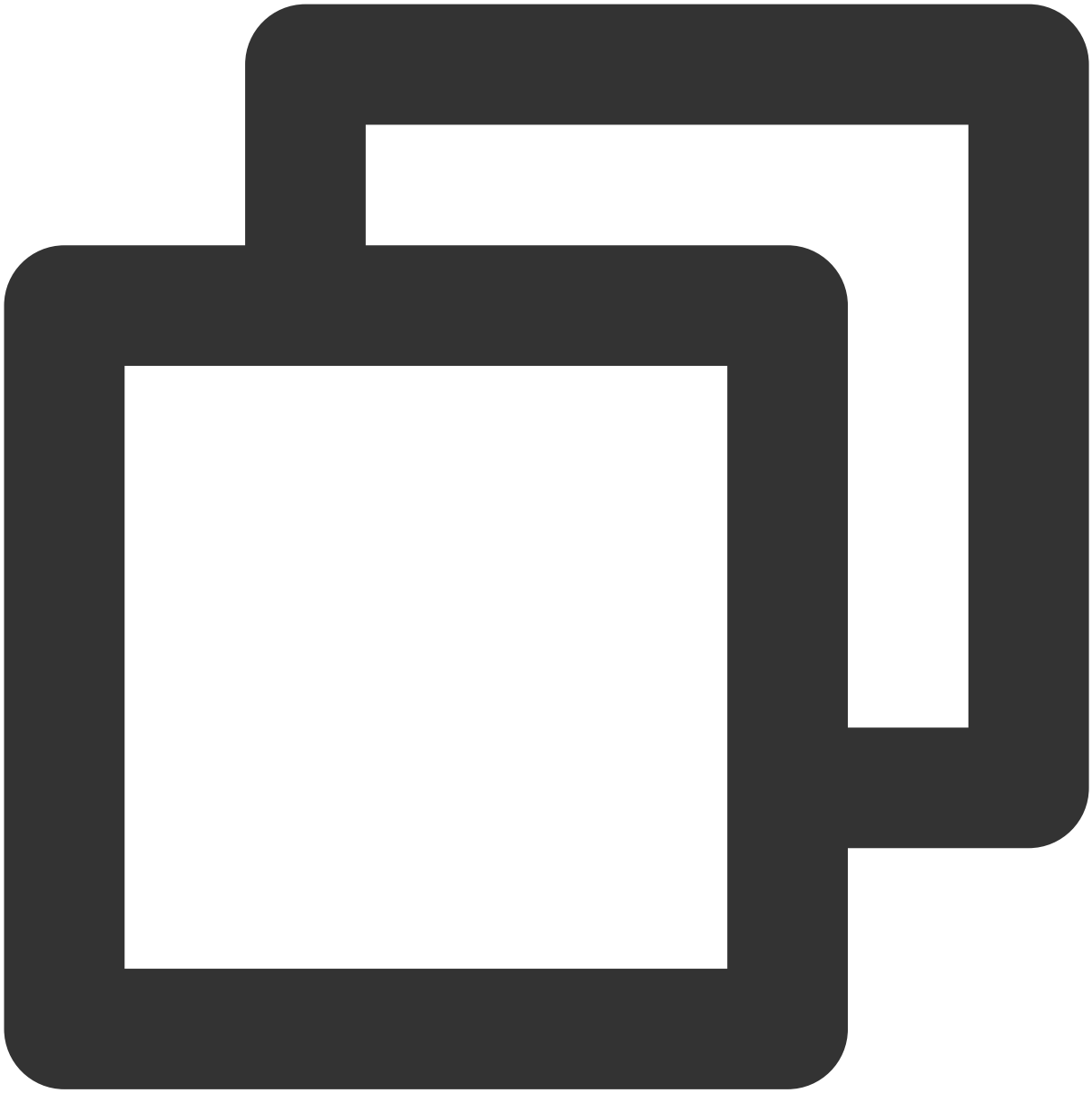
礼物面板组件 `TUIGiftListView` 提供了 `setBalance` 接口，可用于设置礼物面板上显示的余额值。



```
giftListView.setBalance(xxx);
```

充值

实现 `TUIGiftListView` 的 `OnGiftListener` 中的 `onRecharge` 回调可用于接收礼物展示面板抛出的充值按钮点击事件，在这里对接自己的充值系统。



```
public void onRecharge(TUIGiftListView view) {
    //...去充值
    //设置最新的余额
    giftListView.setBalance(balance);
}
```

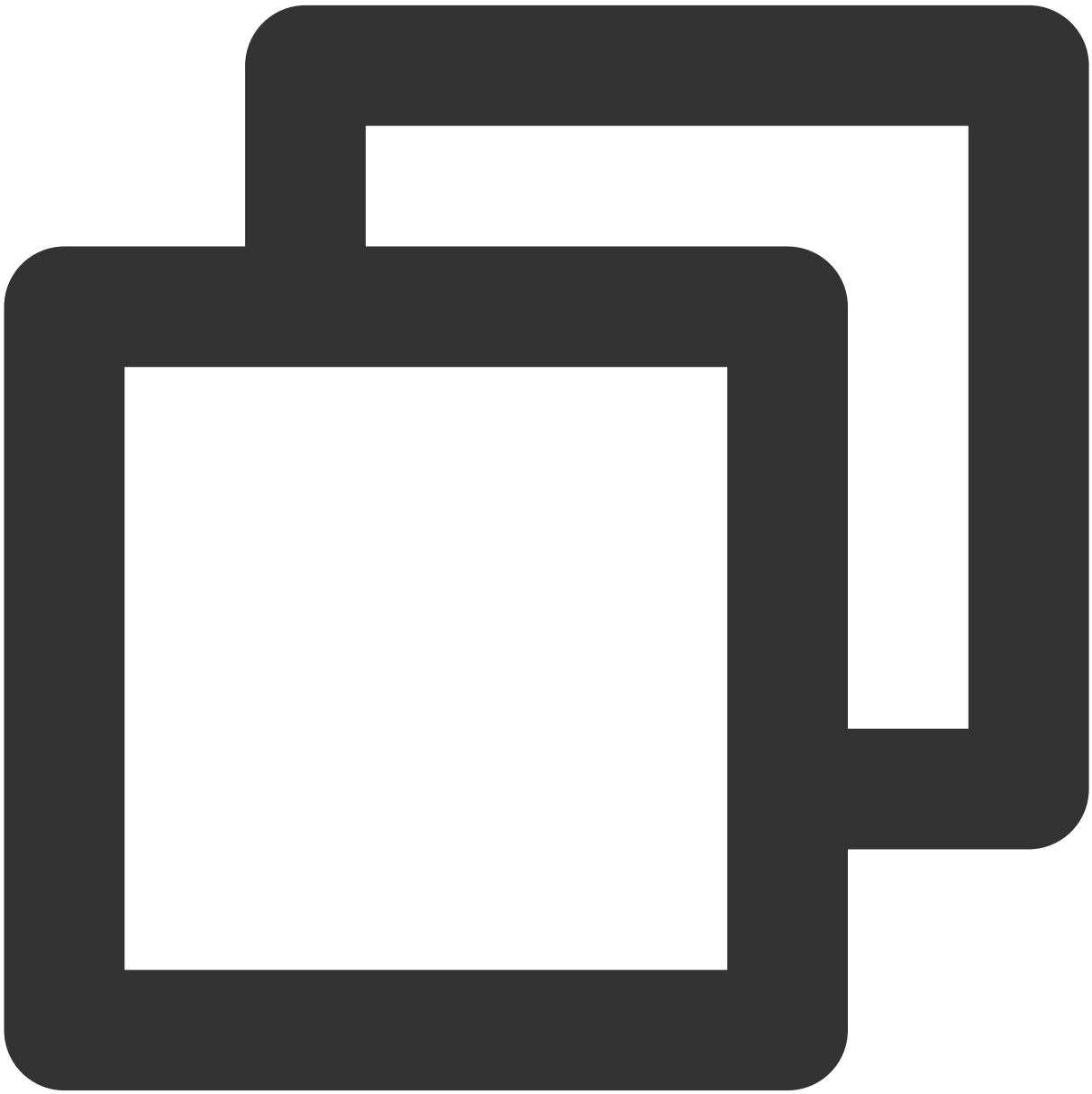
注意：

礼物余额是个虚拟币的概念，并不是真实货币。

礼物充值逻辑，由外部实现，客户可以接入自己的充值系统。充值完毕后再更新礼物余额。

计费统计

实现 `TUIGiftListView` 的 `OnGiftListener` 中的 `onSendGift` 回调，连接客户自己的业务服务器，完成余额校验、礼物计费、消费统计后，再调用 `TUIGiftListView` 的 `sendGift` 发送礼物消息。



```
public void onSendGift(TUIGiftListView giftListView, TUIGift gift, int giftCount) {
    //...此处连接客户自己的业务服务器，完成余额校验、礼物计费、消费统计等
```

```
    TUIGiftUser receiver = new TUIGiftUser();
    //...此处设置礼物接受者信息
    giftListView.sendGift(gift, giftCount, receiver);
}
```

自定义礼物列表

修改观众端礼物面板的礼物列表：

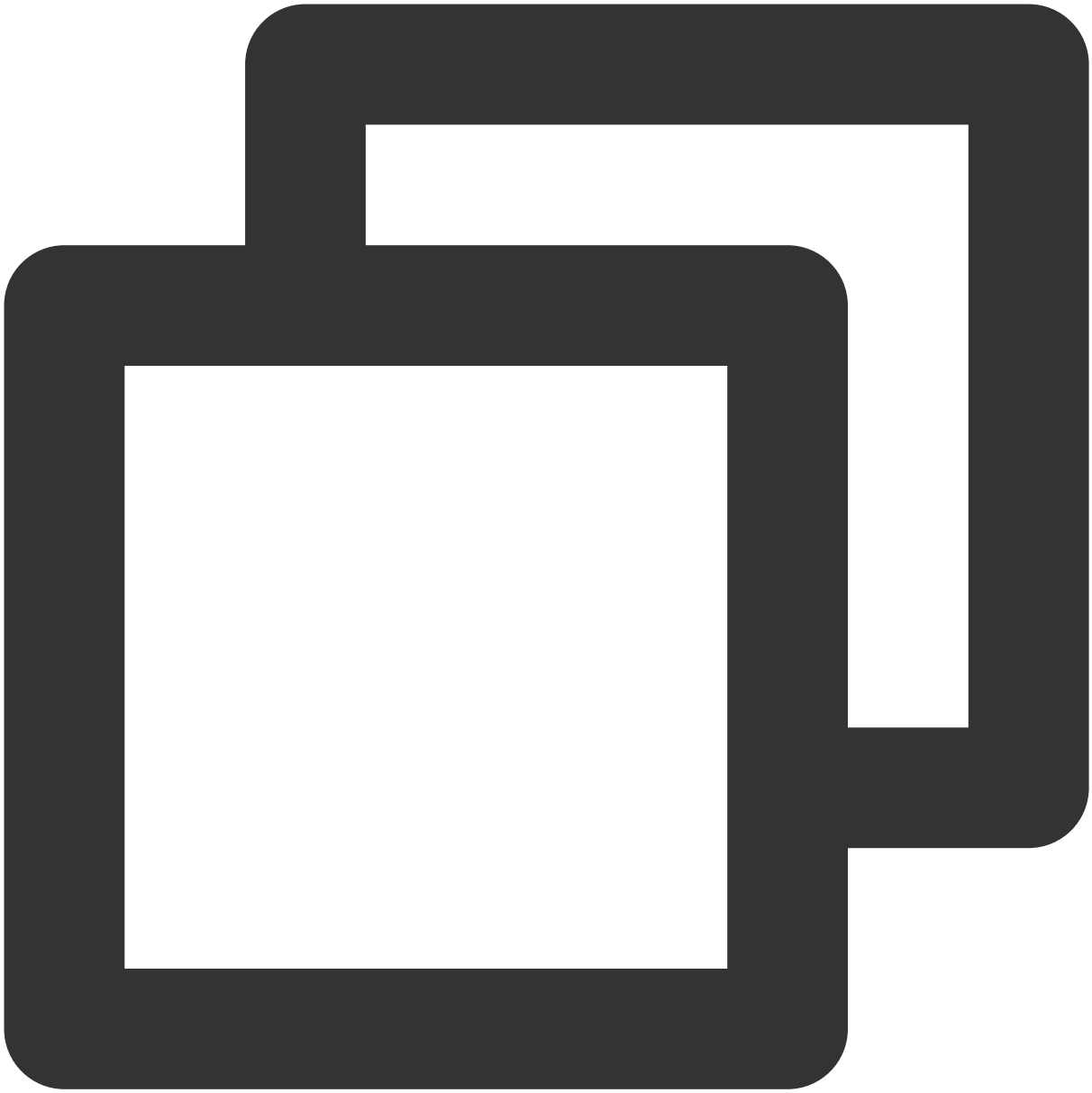
```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c

mGiftCloudServer.queryGiftInfoList((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setGiftList(result);
    } else {
        ToastUtil.toastLongMessage("query gift list error, code = " + error);
    }
}));
```

说明：

客户自行实现 `mGiftCloudServer.queryGiftInfoList` 的逻辑，得到一个自定义的礼物列表 `List<TUIGift>` ，通过 `GiftListView.setGiftList` 设置礼物列表即可。

礼物的 `animationUrl` 要求是一个 `SVGA` 动画。

自定义礼物余额充值

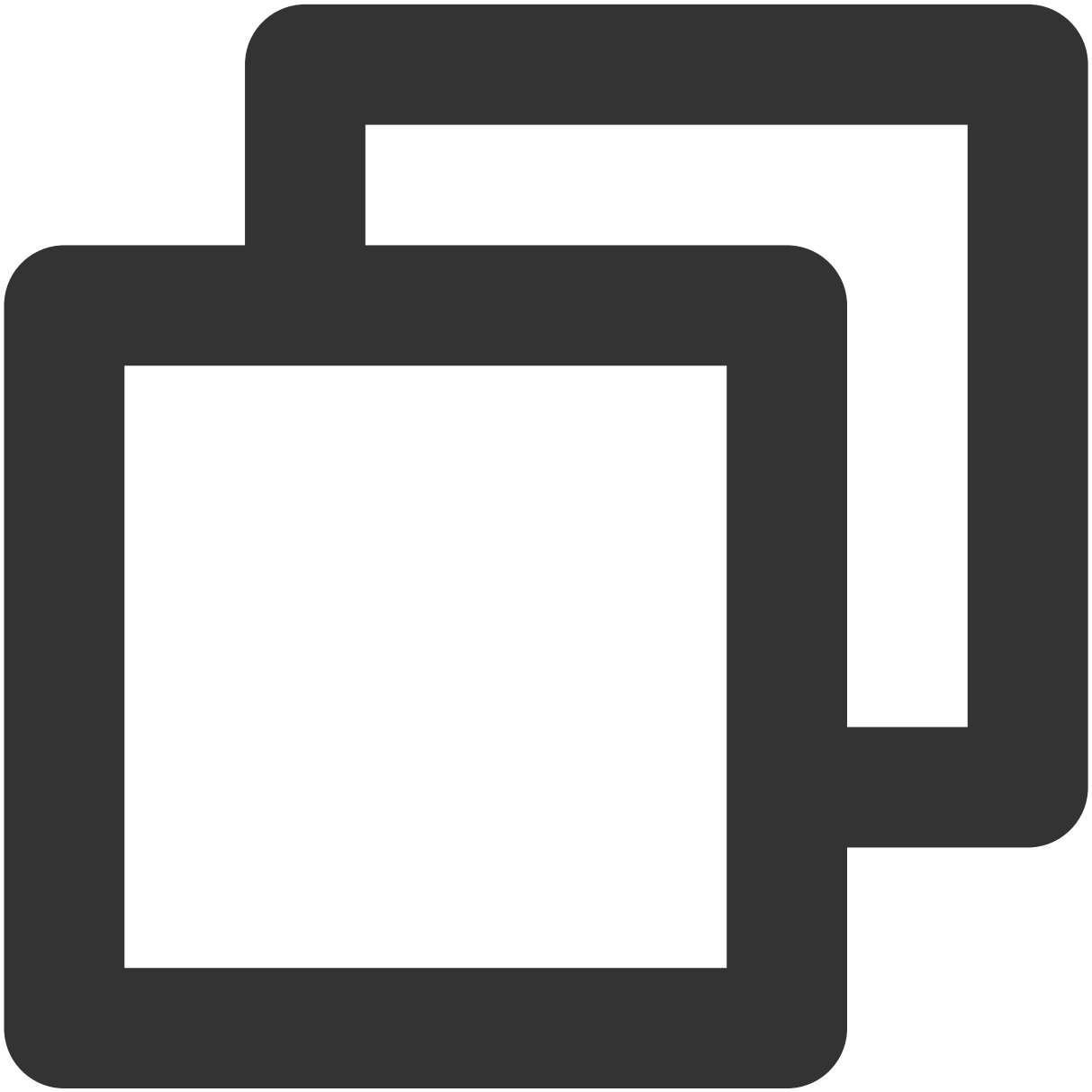


```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c

mGiftCloudServer.queryBalance((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setBalance(result);
    } else {
        ToastUtil.toastLongMessage("query balance error, code = " + error);
    }
}));
```

说明：

客户自行实现 `mGiftCloudServer.queryBalance` 的逻辑，得到礼物余额，通过 `GiftListView.setBalance` 更新礼物余额即可。

自定义礼物消费逻辑



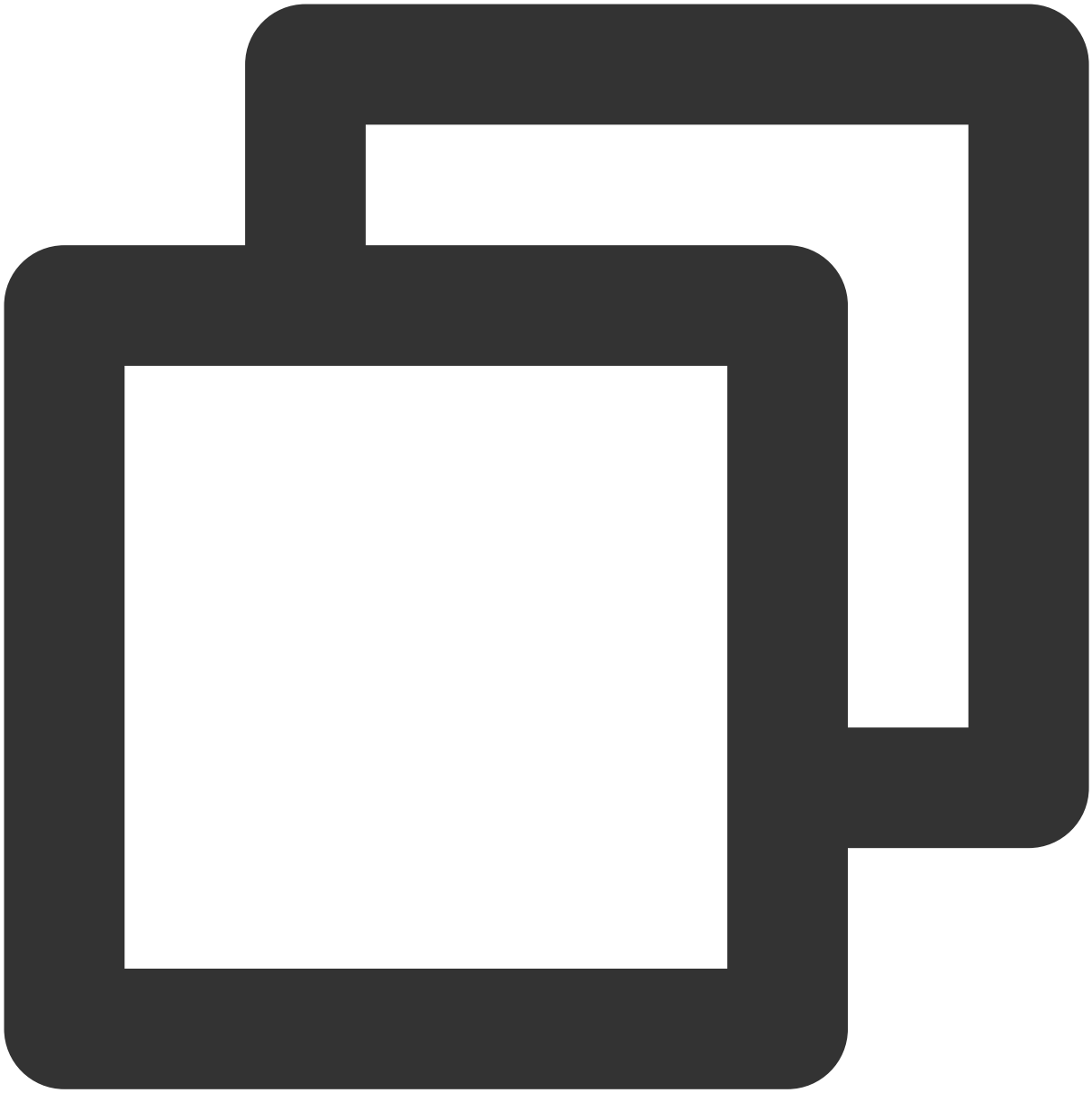
```
// 源码路径：tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/c
```

```
@Override
public void onSendGift(TUIGiftListView view, TUIGift gift, int giftCount) {
    TUIGiftUser receiver = new TUIGiftUser();
    receiver.userId = mLiveRoomInfo.anchorInfo.userId;
    receiver.userName = mLiveRoomInfo.anchorInfo.name.get();
    receiver.avatarUrl = mLiveRoomInfo.anchorInfo.avatarUrl.get();
    receiver.level = "0";
    mGiftCloudServer.sendGift(TUILogin.getUserId(), receiver.userId, gift, giftCoun
            (error, result) -> post(() -> {
        if (error == Error.NO_ERROR) {
            view.sendGift(gift, giftCount, receiver);
            view.setBalance(result);
        } else {
            if (error == Error.BALANCE_INSUFFICIENT) {
                String info = getResources().getString(R.string.livekit_gift_balance
                ToastUtil.toastLongMessage(info);
            } else {
                ToastUtil.toastLongMessage("send gift error, code = " + error);
            }
        }
    }));
}
```

说明：

客户自行实现 `mGiftCloudServer.sendGift` 的逻辑，主要流程是，首先连接客户自己的业务服务器，进行余额校验，校验通过后，由服务器进行计费并统计消费记录，最后将结果回调给客户端。客户端收到成功回调后，通过 `GiftListView` 的 `sendGift` 发送礼物消息，之后再通过 `setBalance` 更新礼物余额。

自定义加载礼物动画并播放



```
// 源码路径：
// tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/audience/component
// tuilivekit/src/main/java/com/trtc/uikit/livekit/liveroom/view/anchor/component/l

@Override
public void onPlayGiftAnimation(TUIGiftPlayView view, TUIGift gift) {
    mGiftCacheService.request(gift.animationUrl, (error, result) -> {
        if (error == 0) {
            view.playGiftAnimation(result);
        }
```

```
        });
    }
```

说明：

客户自行实现 `mGiftCacheService.request` 的逻辑，加载动画成功得到 `result` （ `InputStream` 类型），然后通过 `TUIGiftPlayView` 的 `playGiftAnimation` 播放礼物动画。

客户端API（TUILiveKit）
iOS
UIKit API

最近更新时间：2024-07-05 19:55:56

简介

TUILiveKit 是语音聊天室的 UI 开源套件，目前 iOS 平台支持 Swift 语言，通过简单 API 调用即可唤起直播 UI。

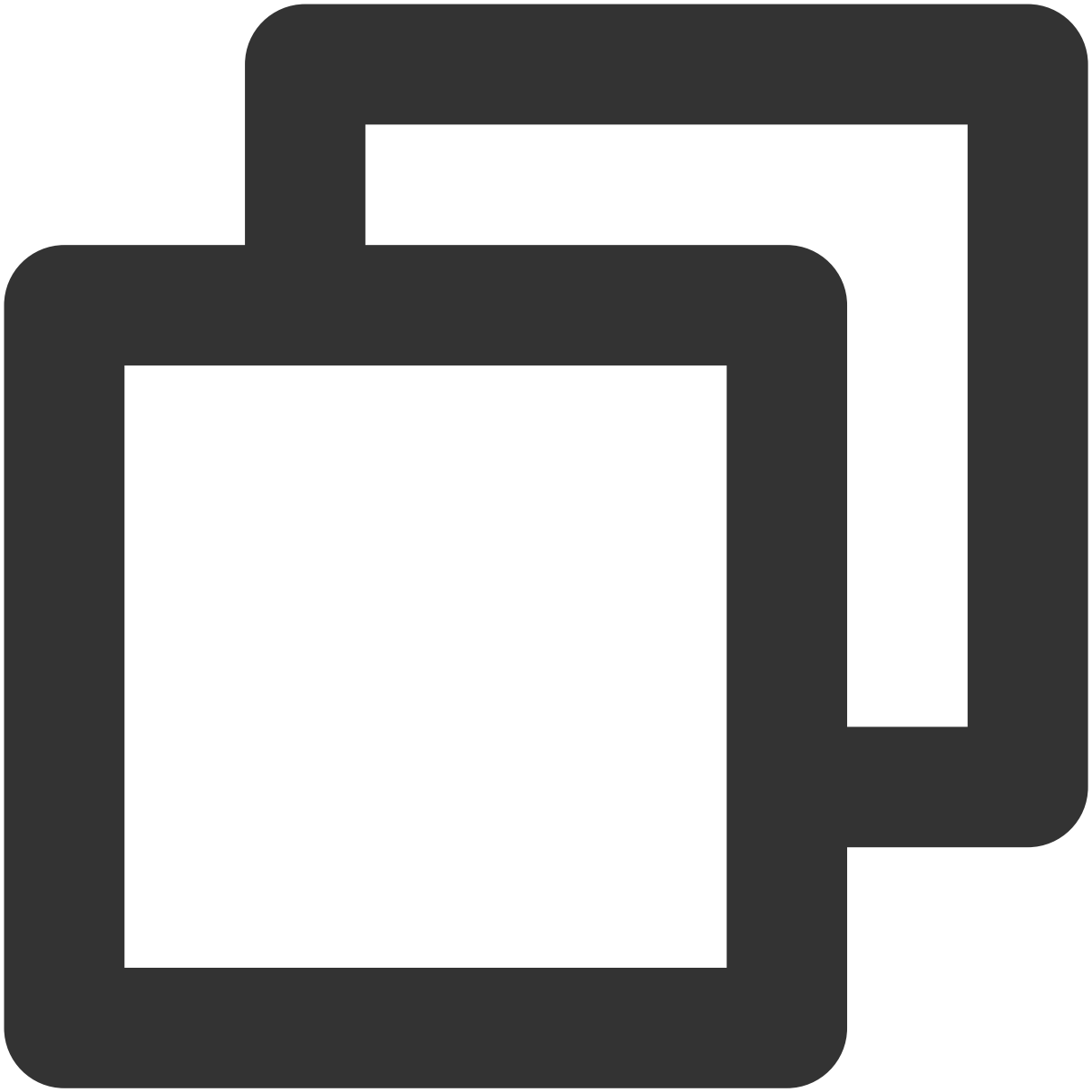
TUIVoiceRoomViewController

API	描述
init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)	初始化 TUIVoiceRoomViewController 对象

init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)

初始化 TUIVoiceRoomViewController 对象。

版权所有：腾讯云计算（北京）有限责任公司



```
public init(roomId: String, behavior: RoomBehavior, roomParams: RoomParams? = nil)
```

参数如下表所示：

参数	类型	含义
roomId	String	语音聊天室房间 Id
behavior	RoomBehavior	初始化语音聊天室类型
roomParams	RoomParams	创建语音聊天室必传参数，加入房间（behavior为join）参数可为

		空

RoomBehavior

语音聊天室类型

类型	描述
autoCreate	直接创建语音聊天室
prepareCreate	预览页创建语音聊天室
join	加入语音聊天室

RoomParams

语音聊天室房间参数

类型	描述
maxSeatCount	房间最大麦位数，默认为套餐包支持的最大麦位数量
seatMode	上麦：自由上麦、申请上麦（默认申请上麦）

版权所有：腾讯云计算（北京）有限责任公司

Engine API
API 概览

最近更新时间：2024-07-05 19:55:32

TUIRoomEngine API 列表

TUIRoomEngine API 是音视频通话组件的无 UI 接口，您可以使用这套 API 根据您的业务需求自定义封装。

TUIRoomEngine

TUIRoomEngine 核心方法

API	描述
init	创建 TUIRoomEngine 实例。
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作。
logout	退出登录接口，会有主动离开房间操作、销毁资源。
setSelfInfo	设置本地用户名称和头像。
getSelfInfo	获取本地用户登录的基本信息。
addObserver	设置事件回调。
removeObserver	移除事件回调。

房间相关主动接口

API	描述
createRoom	创建房间。
destroyRoom	解散房间。
enterRoom	进入房间。
exitRoom	离开房间。
connectOtherRoom	连接其他房间。
disconnectOtherRoom	断开与其他房间的连接。

版权所有：腾讯云计算（北京）有限责任公司

fetchRoomInfo	获取房间信息。
updateRoomNameByAdmin	更新房间名称（只有管理员或群主能够调用）。
updateRoomSpeechModeByAdmin	设置房间麦控模式（只有管理员或群主能够调用）。

本地用户视图渲染、视频管理

API	描述
setLocalVideoView	设置本地用户视频渲染的控件。
openLocalCamera	打开本地摄像头。
closeLocalCamera	关闭本地摄像头。
updateVideoQuality	更新本地视频编码质量设置。
startPushLocalVideo	开始推送本地视频。
stopPushLocalVideo	停止推送本地视频。
startScreenCapture	开始屏幕分享（该接口仅支持移动端）。
startScreenCapture	开始屏幕分享（该接口仅支持桌面端 Mac OS 系统）。
stopScreenCapture	结束屏幕分享。
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）。
selectScreenCaptureTarget	选取要分享的屏幕或窗口（该接口仅支持 Mac OS 系统）。

本地用户音频管理

API	描述
openLocalMicrophone	打开本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
updateAudioQuality	更新本地音频编码质量设置。
startPushLocalAudio	开始推送本地音频。
stopPushLocalAudio	停止推送本地音频。

远端用户视图渲染、视频管理

API	描述
setRemoteVideoView	设置远端用户视频渲染的控件。
startPlayRemoteVideo	开始播放远端用户视频。
stopPlayRemoteVideo	停止播放远端用户视频。
muteRemoteAudioStream	将远端用户禁音。

房间内用户信息

API	描述
getUserList	获取房间内的成员列表。
getUserInfo	获取成员信息。

房间内用户管理

API	描述
changeUserRole	修改用户角色（只有管理员或群主能够调用）。
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或群主能够调用）。

房间内用户发言管理

API	描述
disableDeviceForAllUserByAdmin	控制当前房间内所有用户是否可打开音频流、视频流采集设备的权限状态，例如：全员禁止打开麦克风、全员禁止打开摄像头、全员禁止打开屏幕分享（目前仅会议场景下可用，并且只有管理员或群主能够调用）。
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或群主能够调用）。
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或群主能够调用）。
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）。

房间内麦位管理

API	描述

setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）。
getSeatList	获取麦位列表。
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）。
takeSeat	申请上麦（自由发言模式下，不需要申请）。
leaveSeat	申请下麦（自由发言模式下，不需要申请）。
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦。
kickUserOffSeatByAdmin	主持人/管理员 将用户踢下麦。

信令管理

API	描述
cancelRequest	取消请求。
responseRemoteRequest	回复请求。

发送消息

API	描述
sendTextMessage	发送本文消息。
sendCustomMessage	发送自定义消息。
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或群主能够调用）。
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或群主能够调用）。

高级功能：获取 TRTC 实例

API	描述
getDeviceManager	获取原生 TRTC 设备管理类。
getAudioEffectManager	获取原生 TRTC 音效类。
getBeautyManager	获取原生 TRTC 美颜类。

getTRTCCloud	获取原生 TRTC 实例类。

TUIRoomObserver 回调事件

TUIRoomObserver 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己需要的回调事件。
TUIRoomObserver

TUIRoomObserver

错误回调

API	描述
onError	错误事件回调。

登录状态事件回调

API	描述
onKickedOffLine	其他终端登录被踢下线。
onUserSigExpired	用户凭证超时事件。

房间内事件回调

API	描述
onRoomNameChanged	房间名称更改事件。
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件。
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件。
onSendMessageForAllUserDisableChanged	房间内所有用户发送文本消息被禁用事件。
onKickedOutOfRoom	被踢出房间事件。
onRoomDismissed	房间被解散事件。
onRoomSpeechModeChanged	房间麦控模式发生变化

房间内用户事件回调

API	描述
onRemoteUserEnterRoom	远端用户进房事件。
onRemoteUserLeaveRoom	远端用户离开房间事件。
onUserRoleChanged	用户角色发生变化事件。
onUserVideoStateChanged	用户视频状态发生变化事件。
onUserAudioStateChanged	用户音频状态发生变化事件。
onUserScreenCaptureStopped	用户屏幕采集停止事件。
onUserVoiceVolumeChanged	用户音量变化事件。
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件。

房间内麦位事件回调

API	描述
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged	麦位列表发生变化事件。
onKickedOffSeat	收到用户被踢下麦事件。

请求信令事件回调

API	描述
onRequestReceived	收到请求消息事件。
onRequestCancelled	收到请求被取消的事件。

房间内消息事件回调

API	描述
onReceiveTextMessage	收到普通文本消息事件。
onReceiveCustomMessage	收到自定义消息事件。

版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司

Android
UIKit API

最近更新时间：2024-07-05 19:57:03

简介

TUILiveKit 是语音聊天室的 UI 开源套件，目前 Android 平台仅支持 Java 语言，通过简单 API 调用即可唤起直播 UI。

TUIVoiceRoomFragment 类

API	描述
TUIVoiceRoomFragment(String roomId, LiveDefine.RoomBehavior behavior, LiveDefine.RoomParams params)	构造语音聊天室主界面对象

TUIVoiceRoomFragment

构造 语音聊天室主界面 对象

```
public TUIVoiceRoomFragment(String roomId, LiveDefine.RoomBehavior behavior, LiveDe
```

参数如下表所示：

参数	类型	含义
roomId	String	语音聊天室房间 Id
behavior	RoomBehavior	初始化语音聊天室类型
params	RoomParams	创建语音聊天室必传参数，加入房间（behavior 为

		JOIN）参数可为空

RoomBehavior

语音聊天室类型

类型	描述
AUTO_CREATE	直接创建语音聊天室
PREPARE_CREATE	预览页创建语音聊天室
JOIN	加入语音聊天室

RoomParams

语音聊天室房间参数

类型	描述
maxSeatCount	房间最大麦位数，默认为套餐包支持的最大麦位数量
seatMode	上麦：自由上麦、申请上麦（默认申请上麦）

Engine API
API 概览

最近更新时间：2024-07-05 19:57:32

TUIRoomEngine (无 UI 接口)

TUIRoomEngine API 是音视频通话组件的**无 UI 接口,**您可以使用这套 API 根据您的业务需求自定义封装。

TUIRoomEngine

TUIRoomEngine 核心方法

API	描述
createInstance	创建 TUIRoomEngine 实例
destroyInstance	销毁 TUIRoomEngine 实例
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作。
logout	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfo	设置本地用户名称和头像
getSelfInfo	获取本地用户基本信息
addObserver	设置事件回调
removeObserver	移除事件回调

房间相关主动接口

API	描述
createRoom	创建房间
destroyRoom	解散房间
enterRoom	进入房间
exitRoom	离开房间
connectOtherRoom	连接其他房间

版权所有：腾讯云计算（北京）有限责任公司

disconnectOtherRoom	断开与其他房间的连接
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间名称
updateRoomSpeechModeByAdmin	设置房间管理模式（只有管理员或群主能够调用）

本地用户视图渲染、视频管理

API	描述
setLocalVideoView	设置本地用户视频渲染的视图控件
openLocalCamera	打开本地摄像头
closeLocalCamera	关闭本地摄像头
updateVideoQuality	更新本地视频编码质量设置
startScreenSharing	开始屏幕分享
stopScreenSharing	结束屏幕分享
startPushLocalVideo	开始推送本地视频
stopPushLocalVideo	停止推送本地视频

本地用户音频管理

API	描述
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateAudioQuality	更新本地音频编码质量设置
startPushLocalAudio	开始推送本地音频
stopPushLocalAudio	停止推送本地音频

远端用户视图渲染、视频管理

API	描述
setRemoteVideoView	设置远端用户视频渲染的视图控件

版权所有：腾讯云计算（北京）有限责任公司

startPlayRemoteVideo	开始播放远端用户视频
stopPlayRemoteVideo	停止播放远端用户视频
muteRemoteAudioStream	将远端用户禁音

房间内用户信息

API	描述
getUserList	获取房间内的成员列表
getUserInfo	获取成员信息

房间内用户管理

API	描述
changeUserRole	修改用户角色（只有管理员或群主能够调用）
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或群主能够调用）

房间内用户发言管理

API	描述
disableDeviceForAllUserByAdmin	全体用户媒体设备管理（只有管理员或群主能够调用）
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或群主能够调用）
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或群主能够调用）
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）

房间内麦位管理

API	描述
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList	获取麦位列表
lockSeatByAdmin	锁定麦位（包括位置锁定，音频状态锁定，视频状态锁定）
takeSeat	本地上麦

版权所有：腾讯云计算（北京）有限责任公司

	会议场景：SPEAK_AFTER_TAKING_SEAT 模式需要向主持人或管理员发起申请才允许上麦，其他模式不支持上麦。 直播场景：FREE_TO_SPEAK 模式可以自由上麦，上麦后开麦发言；SPEAK_AFTER_TAKING_SEAT 模式需要向主持人或管理员发起申请才允许上麦；其他模式不支持上麦
leaveSeat	本地下麦
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin	主持人/管理员 将用户下麦

信令管理

API	描述
cancelRequest	取消请求
responseRemoteRequest	回复请求

发送消息

API	描述
sendTextMessage	发送本文消息
sendCustomMessage	发送自定义消息
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或群主能够调用）
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或群主能够调用）

高级功能：获取 TRTC 实例

API	描述
getTRTCCloud	获得 TRTC 实例对象
getDeviceManager	获得设备管理对象
getAudioEffectManager	获得音效管理对象
getBeautyManager	获得美颜管理对象

版权所有：腾讯云计算（北京）有限责任公司

事件类型定义

TUIRoomObserver 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己需要的回调事件。
TUIRoomObserver

错误回调

事件	描述
onError	错误回调事件

登录状态事件回调

API	描述
onKickedOffLine	用户被踢下线事件
onUserSigExpired	用户凭证超时事件

房间内事件回调

API	描述
onRoomNameChanged	房间名称更改事件
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged	房间内所有用户发送文本消息被禁用事件
onRoomDismissed	房间被解散事件
onKickedOutOfRoom	被踢出房间事件
onRoomSpeechModeChanged	房间麦控模式发生变化

房间内用户事件回调

API	描述
onRemoteUserEnterRoom	远端用户进房事件
onRemoteUserLeaveRoom	远端用户离开房间事件

版权所有：腾讯云计算（北京）有限责任公司

onUserRoleChanged	用户角色发生变化事件
onUserVideoStateChanged	用户视频状态发生变化事件
onUserAudioStateChanged	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件
onUserScreenCaptureStopped	屏幕分享结束事件

房间内麦位事件回调

API	描述
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged	麦位列表发生变化事件
onKickedOffSeat	收到用户被踢下麦事件

请求信令事件回调

API	描述
onRequestReceived	收到请求消息事件
onRequestCancelled	收到请求被取消的事件

房间内消息事件回调

API	描述
onReceiveTextMessage	收到普通文本消息事件
onReceiveCustomMessage	收到自定义消息事件

版权所有：腾讯云计算（北京）有限责任公司

错误码（TUILiveKit）

最近更新时间：2024-05-17 11:20:40

客户端错误码

通用错误码

错误码	描述
0	操作成功
-1	暂未归类的通用错误
-2	请求被限频，请稍后重试
-1000	未找到 SDKAppID，请在 腾讯云视立方 SDK 控制台 确认应用信息
-1001	调用 API 时，传入的参数不合法，检查入参是否合法
-1002	未登录,请调用 Login 接口
-1003	获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限
-1004	该功能需要开通额外的套餐，请在 腾讯云视立方 SDK 控制台 按需开通对应套餐

本地用户渲染，视频管理，音频管理 API 回调错误定义

错误码	描述
-1100	系统问题，打开摄像头失败。检查摄像头设备是否正常
-1101	摄像头没有系统授权, 检查系统授权
-1102	摄像头被占用，检查是否有其他进程使用摄像头
-1103	当前无摄像头设备，请插入摄像头设备解决该问题
-1104	系统问题，打开麦克风失败。检查麦克风设备是否正常
-1105	麦克风没有系统授权，检查系统授权
-1106	麦克风被占用
-1107	当前无麦克风设备

-1108	获取屏幕分享对象失败，检查屏幕录制权限
-1109	开启屏幕分享失败，检查房间内是否有人正在屏幕分享

房间管理相关 API 回调错误定义

错误码	描述
-2100	进房时房间不存在，或许已被解散
-2101	需要进房后才可使用此功能
-2102	房主不支持退房操作，Conference（会议）房间类型: 可以先转让房主，再退房。LivingRoom（直播）房间类型: 房主只能解散房间
-2103	当前房间类型下不支持该操作
-2104	当前发言模式下不支持该操作
-2105	创建房间 ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
-2106	房间 ID 已被使用，请选择别的房间 ID
-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
-2108	当前用户已在别的房间内，单个 roomEngine 实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的 roomEngine 实例

房间内用户信息 API 回调错误定义

错误码	描述
-2200	未找到该用户
-2201	房间内未找到该用户

房间内用户发言管理 API 回调错误定义&房间内麦位管理 API 回调错误定义

错误码	描述
-2300	需要房主权限才能操作
-2301	需要房主或者管理员权限才能操作
-2310	信令请求无权限，比如取消非自己发起的邀请。
-2311	信令请求 ID 无效或已经被处理过。

-2340	最大麦位超出套餐包数量限制
-2341	当前用户已经在麦位上
-2342	当前麦位已经有人了
-2343	当前麦位被锁
-2344	麦位编号不存在
-2345	当前用户没有在麦上
-2346	上麦人数已满
-2360	当前麦位音频被锁
-2361	需要向房主或管理员申请后打开麦克风
-2370	当前麦位视频被锁, 需要由房主解锁麦位后，才能打开摄像头
-2371	需要向房主或管理员申请后打开摄像头
-2380	当前房间已开启全员禁言
-2381	当前房间内，您已被已禁言

服务端错误码

错误码	描述
83007	请求被限频
84002	房间不存在
84003	房间内人数已超出最大套餐包限制
84004	房间不存在该用户
84005	房间已存在
84006	房主不能退房
85001	麦位已满
85002	麦位被锁定

85003	用户已上麦
85004	麦位被使用
85005	麦位号不存在
85006	麦位音频被锁定
85007	麦位视频被锁定
85008	非直播场景不能使用麦位
87001	不允许修改成员身份或房间信息
87002	权限不足，需要房主或管理员身份
87003	IM 群属性更新错误
87005	IM 群属性获取异常
87006	权限不足，需要房主才可以操作
87007	权限不足，需要管理员才可以操作
87008	不允许销毁 IM 房间
41001	创建 IM 群组错误
41002	销毁 IM 群组错误
41003	添加 IM 成员错误
41004	删除 IM 成员错误
41005	IM 改变角色错误
42001	预定会议错误
42002	预定会议无效
42003	预定会议必须是会议场景
42004	预定会议未开始
42005	预定会议已经开始
42006	预定会议邀请成员太多，超出数量限制

版权所有：腾讯云计算（北京）有限责任公司

发布日志（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:20:41

2024年5月

发布动态	描述	发布时间
Version 1.0.0	支持自定义礼物 支持自定义弹幕 支持音效&混响 支持点赞 支持麦控	2024.05.17

Android

最近更新时间：2024-05-17 11:20:40

2024年4月

发布动态	描述	发布时间
Version 1.0.0	支持自定义礼物 支持自定义弹幕 支持音效&混响 支持点赞 支持麦控	2024.05.17

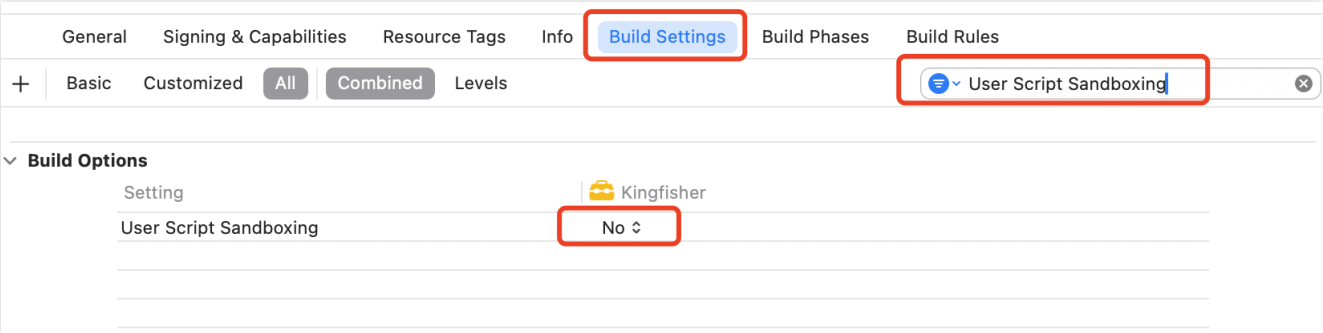
常见问题（TUILiveKit）
iOS

最近更新时间：2024-05-17 11:48:33

Xcode 15 编译报错？

1. 出现 **Sandbox: rsync**，编译报错截图：

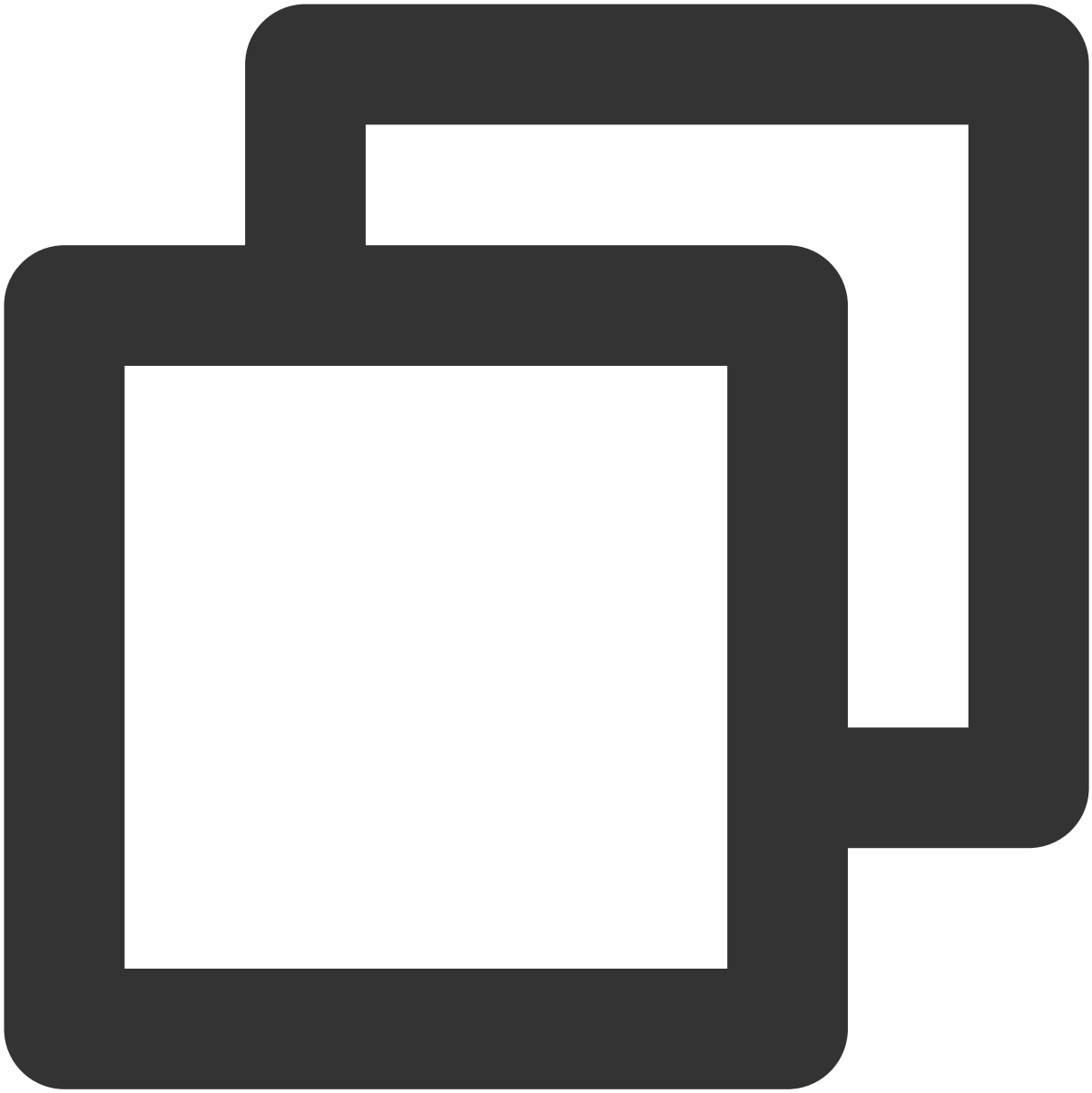
可以在 **Build Settings** 中把 **User Script Sandboxing** 设置为 **NO**：



2. 如果出现 **SDK does not contain**，编译报错截图：

```
All   Recent   All Messages   All Issues   Errors Only                    Export...   Filter
Compile SVGAExporter.m (arm64) 0.1 seconds
Compile SVGAContentLayer.m (arm64) 0.2 seconds
Compile SVGABitmapLayer.m (arm64) 0.1 seconds
Compile SVGABezierPath.m (arm64) 0.1 seconds
Compile SVGAAudioLayer.m (arm64) 0.2 seconds
Compile SVGAAudioEntity.m (arm64) 0.1 seconds
Compile SVGA.m (arm64) 0.1 seconds
Write SVGAPlayer.LinkFileList (arm64) 0.1 seconds
Link SVGAPlayer (arm64) 0.1 seconds                                        1
    Ld /Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/SVGAPlayer/
      SVGAPlayer.framework/SVGAPlayer normal (in target 'SVGAPlayer' from project 'Pods')
        cd /Users/wesleylei/Downloads/12333/Pods
        /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang -Xlinker -reproducible -target arm64-
      apple-ios8.0 -dynamiclib -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/
      iPhoneOS17.4.sdk -O0 -L/Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/
      Intermediates.noindex/EagerLinkingTBDs/Debug-iphoneos -L/Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-
      camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/SVGAPlayer -F/Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-
      camchkqjvdeemtekovsoevsdgbhj/Build/Intermediates.noindex/EagerLinkingTBDs/Debug-iphoneos -F/Users/wesleylei/Library/Developer/Xcode/
      DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/SVGAPlayer -F/Users/wesleylei/Library/Developer/Xcode/
      DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/Protobuf -F/Users/wesleylei/Library/Developer/Xcode/
      DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/SSZipArchive -filelist /Users/wesleylei/Library/
      Developer/Xcode/DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Intermediates.noindex/Pods.build/Debug-iphoneos/
      SVGAPlayer.build/Objects-normal/arm64/SVGAPlayer.LinkFileList -install_name @rpath/SVGAPlayer.framework/SVGAPlayer -Xlinker -rpath
      -Xlinker @executable_path/Frameworks -Xlinker -rpath -Xlinker @loader_path/Frameworks -dead_strip -Xlinker -object_path_lto
      -Xlinker /Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Intermediates.noindex/
      Pods.build/Debug-iphoneos/SVGAPlayer.build/Objects-normal/arm64/SVGAPlayer_lto.o -Xlinker -export_dynamic -Xlinker -no_deduplicate
      -fobjc-arc -fobjc-link-runtime -lz -framework AVFoundation -framework Protobuf -framework SSZipArchive -framework Security -framework
      AVFoundation -framework Foundation -compatibility_version 1 -current_version 1 -Xlinker -dependency_info -Xlinker /Users/wesleylei/
      Library/Developer/Xcode/DerivedData/12333-camchkqjvdeemtekovsoevsdgbhj/Build/Intermediates.noindex/Pods.build/Debug-iphoneos/
      SVGAPlayer.build/Objects-normal/arm64/SVGAPlayer_dependency_info.dat -o /Users/wesleylei/Library/Developer/Xcode/DerivedData/12333-
      camchkqjvdeemtekovsoevsdgbhj/Build/Products/Debug-iphoneos/SVGAPlayer/SVGAPlayer.framework/SVGAPlayer

    clang: error: SDK does not contain 'libarclite' at the path '/Applications/Xcode.app/Contents/Developer/Toolchains/
      XcodeDefault.xctoolchain/usr/lib/arc/libarclite_iphoneos.a'; try increasing the minimum deployment target
SDK does not contain 'libarclite' at the path '/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/arc/libarclite_iphoneos.a'; try increa...   more
Build target Protobuf
Project Pods | Configuration Debug | Destination tf手机13 | SDK iOS 17.4
The iOS deployment target 'IPHONEOS_DEPLOYMENT_TARGET' is set to 10.0, but the range of supported deployment target versions is 12.0 to 17.4.99.
```

请在 Podfile 添加如下代码：

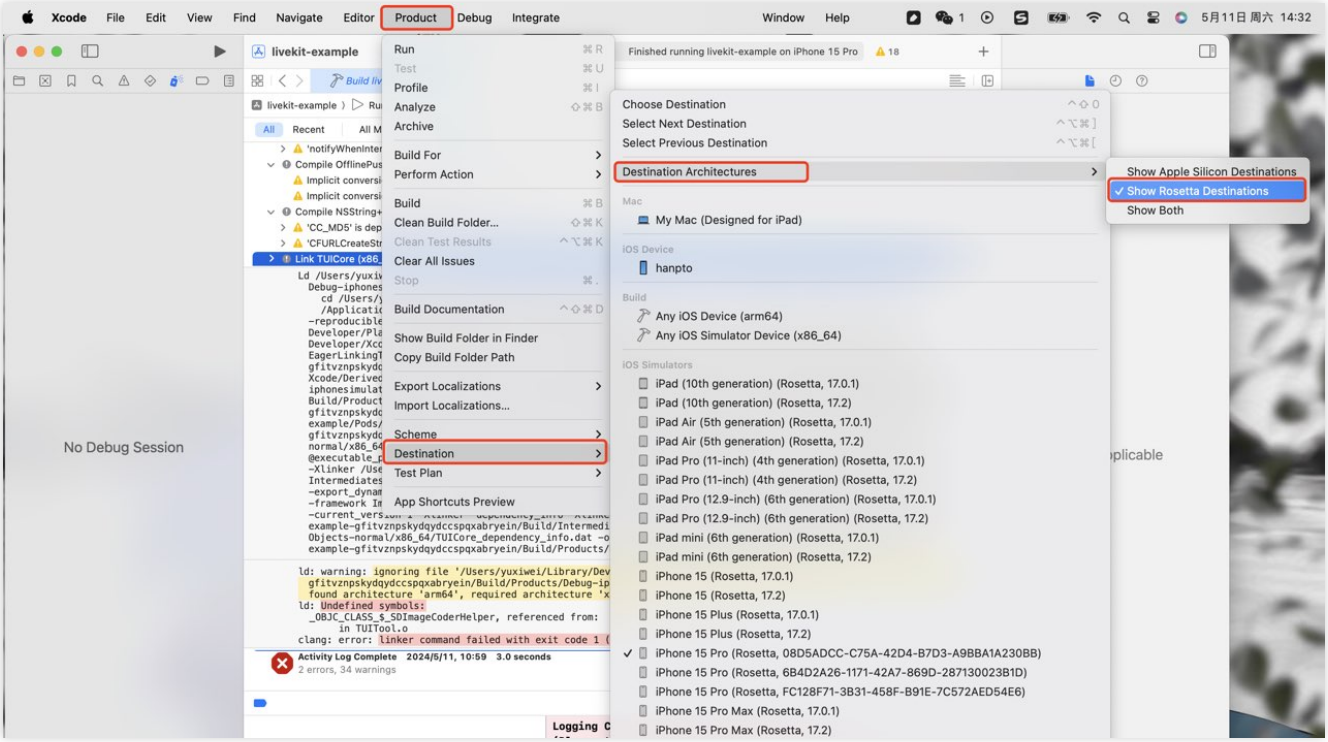


```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '13.0'
    end
  end
end
```

3. 如果在 **M 系列电脑**上运行模拟器，可能会出现 **Linker command failed with exit code 1 (use -v to see invocation)**，编译报错截图：

```
'notifyWhenInteractionEndsUsingBlock:' is deprecated: first deprecated in iOS 10.0
Compile OfflinePushExtConfigInfo.m (x86_64) 0.2 seconds                                  2
  Implicit conversion loses integer precision: 'NSInteger' (aka 'long') to 'int'
  Implicit conversion loses integer precision: 'NSInteger' (aka 'long') to 'int'
Compile NSString+TUIUtil.m (x86_64) 0.2 seconds                                          2
  'CC_MD5' is deprecated: first deprecated in iOS 13.0 - This function is cryptographically broken and should not be used in security contexts. Clie... more
  'CFURLCreateStringByAddingPercentEscapes' is deprecated: first deprecated in iOS 9.0 - Use [NSString stringByAddingPercentEncodingWithAll... more
Link TUICore (x86_64) 0.4 seconds                                                        1  2

Ld /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfitvznpskydqydccspqxabryein/Build/Products/
  Debug-iphonesimulator/TUICore/TUICore.framework/TUICore normal (in target 'TUICore' from project 'Pods')
    cd /Users/yuxiwei/Downloads/livekit-example/Pods
    /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang -Xlinker
  -reproducible -target x86_64-apple-ios13.0-simulator -dynamiclib -isysroot /Applications/Xcode.app/Contents/
  Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator17.0.sdk -O0 -L/Users/yuxiwei/Library/
  Developer/Xcode/DerivedData/livekit-example-gfitvznpskydqydccspqxabryein/Build/Intermediates.noindex/
  EagerLinkingTBDs/Debug-iphonesimulator -L/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfitvznpskydqydccspqxabryein/Build/Products/Debug-iphonesimulator/TUICore -F/Users/yuxiwei/Library/Developer/
  Xcode/DerivedData/livekit-example-gfitvznpskydqydccspqxabryein/Build/Intermediates.noindex/EagerLinkingTBDs/Debug-
  iphonesimulator -F/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfitvznpskydqydccspqxabryein/
  Build/Products/Debug-iphonesimulator/TUICore -F/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfitvznpskydqydccspqxabryein/Build/Products/Debug-iphonesimulator/SDWebImage -F/Users/yuxiwei/Downloads/livekit-
  example/Pods/TXIMSDK_Plus_iOS -filelist /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfitvznpskydqydccspqxabryein/Build/Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/Objects-
  normal/x86_64/TUICore.LinkFileList -install_name @rpath/TUICore.framework/TUICore -Xlinker -rpath -Xlinker
  @executable_path/Frameworks -Xlinker -rpath -Xlinker @loader_path/Frameworks -dead_strip -Xlinker -object_path_lto
  -Xlinker /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-gfitvznpskydqydccspqxabryein/Build/
  Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/Objects-normal/x86_64/TUICore_lto.o -Xlinker
  -export_dynamic -Xlinker -no_deduplicate -Xlinker -objc_abi_version -Xlinker 2 -fobjc-arc -fobjc-link-runtime
  -framework ImSDK_Plus -framework ImageIO -framework SDWebImage -framework Foundation -compatibility_version 1
  -current_version 1 -Xlinker -dependency_info -Xlinker /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-
  example-gfitvznpskydqydccspqxabryein/Build/Intermediates.noindex/Pods.build/Debug-iphonesimulator/TUICore.build/
  Objects-normal/x86_64/TUICore_dependency_info.dat -o /Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-
  example-gfitvznpskydqydccspqxabryein/Build/Products/Debug-iphonesimulator/TUICore/TUICore.framework/TUICore

ld: warning: ignoring file '/Users/yuxiwei/Library/Developer/Xcode/DerivedData/livekit-example-
  gfitvznpskydqydccspqxabryein/Build/Products/Debug-iphonesimulator/SDWebImage/SDWebImage.framework/SDWebImage':
  found architecture 'arm64', required architecture 'x86_64'
ld: Undefined symbols:
  _OBJC_CLASS_$_SDImageCoderHelper, referenced from:
      in TUITool.o
clang: error: linker command failed with exit code 1 (use -v to see invocation)

Activity Log Complete  2024/5/11, 10:59  3.0 seconds
2 errors, 34 warnings
```

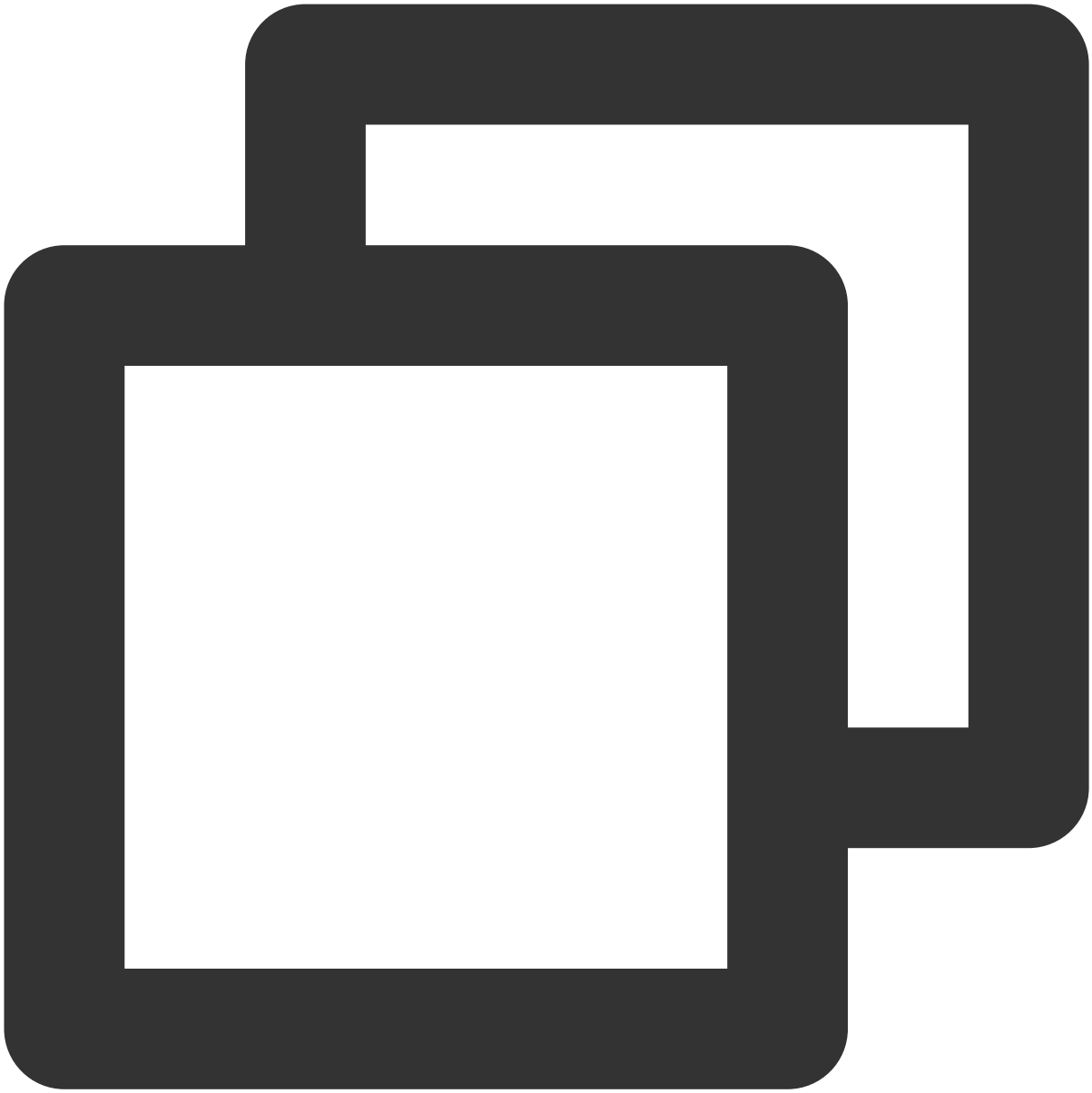
需要修改 xcode 配置。xcode 打开项目 > Product > Destination > **Destination Architectures** 可以选择用哪种模式的模拟器打开，需要选择 （**Rosetta**）结尾的模拟器。



TUILiveKit 和自己集成的音视频库冲突了？

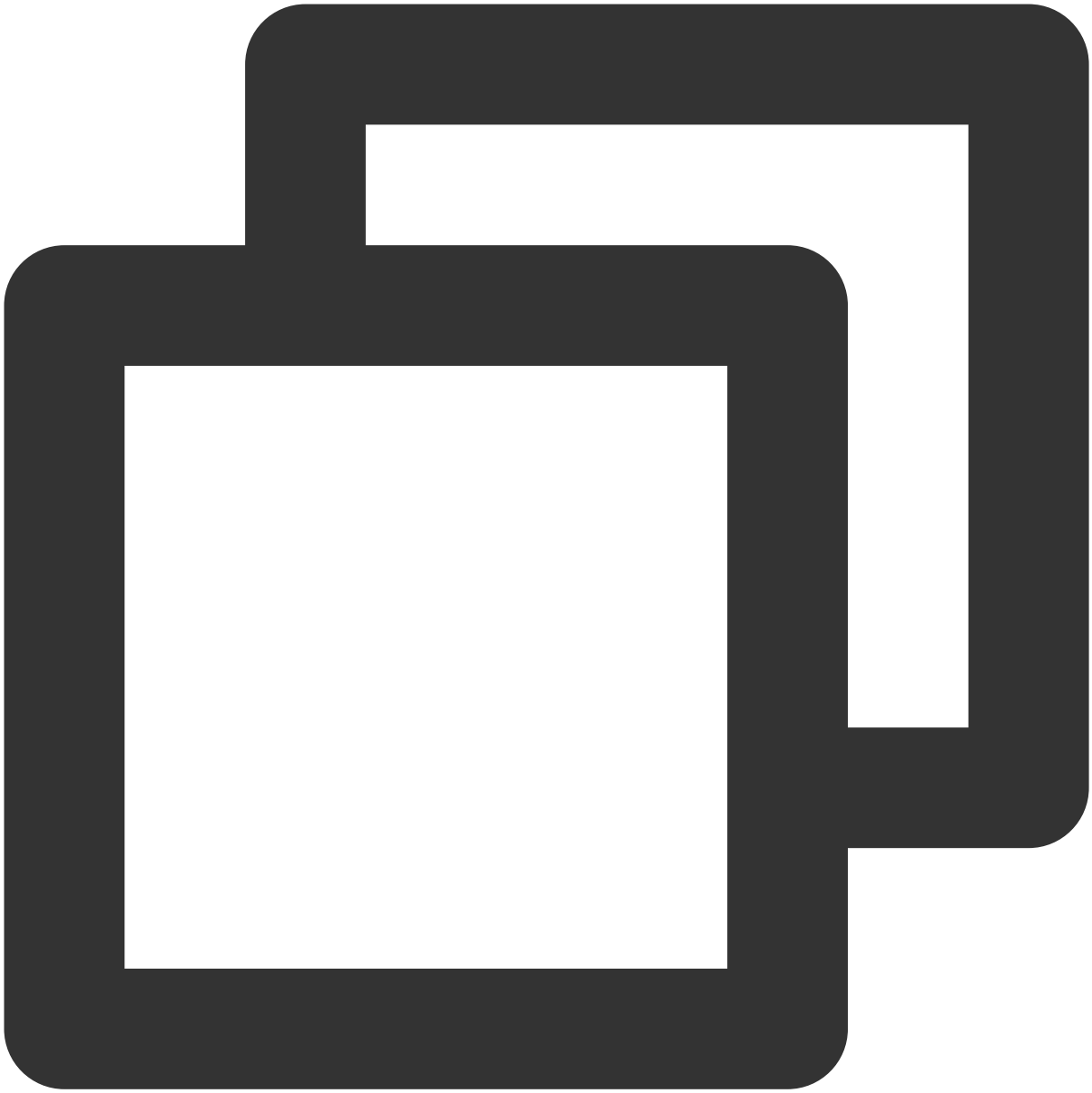
腾讯云的 `音视频库` 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。

1. 如果您使用了 `TXLiteAVSDK_TRTC` 库，不会发生符号冲突。可直接在 `Podfile` 文件中添加依赖，



```
pod 'TUILiveKit'
```

2. 如果您使用了 `TXLiteAVSDK_Professional` 库，会产生符号冲突。您可在 `Podfile` 文件中添加依赖，



```
pod 'TUILiveKit/Professional'
```

3. 如果您使用了 `TXLiteAVSDK_Enterprise` 库，会产生符号冲突。建议升级到 `TXLiteAVSDK_Professional` 后使用 `TUILiveKit/Professional` 。

如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 .xlog。日志是否加密是可以通过 setLogCompressEnabled 来控制，生成的文件名里面含 C(compressed) 的就是加密压缩的，含 R(raw) 的就是明文的。

iOS： `sandbox的Documents/log`

说明：

查看 .xlog 文件需要下载 解密工具，在 Python 2.7环境中放到 xlog 文件同目录下直接使用 `python decode_mars_log_file.py` 运行即可。

查看 .clog 文件（9.6 版本以后新的日志格式）需要下载 解密工具，在 Python 2.7 环境中放到 clog 文件同目录下直接使用 `python decompress_clog.py` 运行即可。

Android

最近更新时间：2024-05-17 11:48:33

TUILiveKit 是否可以不引入 IM SDK，只使用 TRTC？

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，例如创建房间信令、连麦信令等核心逻辑都使用 IM服务，如果您已经购买有其他 IM 产品，也可以参照 TUILiveKit 逻辑进行适配。

运行异常：allowBackup 异常，如何处理？

```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-3
    is also present at [com.github.yyued:SVGAPlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(tr
    Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.xm
```

问题原因：多个模块的 `AndroidManifest.xml` 中都配置了 `allowBackup` 属性，造成冲突。
解决方法：您可以在您工程的 `AndroidManifest.xml` 文件中删除 allowBackup 属性或将该属性改为 false，表示关闭备份和恢复功能；并在 `AndroidManifest.xml` 文件的 application 节点中添加 `tools:replace="android:allowBackup"` 表示覆盖其他模块的设置，使用您自己的设置。修复示例如图所示：

```
oginActivity.java    AndroidManifest.xml    VideoViewFactory.java    Ar
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        tools:replace="android:allowBackup"
        android:allowBackup="false"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

Activity 主题问题：Activity need to use a Theme.AppCompat theme，如何处理？

```
FATAL EXCEPTION: main
Process: com.trtc.uikit.livekit.example, PID: 15190
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.trtc.uikit.livekit.example/com.trtc.u
 .LoginActivity}: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) w
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3730)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3885)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:101)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2332)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:230)
    at android.app.ActivityThread.main(ActivityThread.java:8115) <1 internal line>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:526)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1034)
```

问题原因：由于 `LoginActivity` 继承自 `AppCompatActivity` ，所以要给 `LoginActivity` 设置一个 `Theme.AppCompat` 主题。

解决方法：您可以在您工程的 `AndroidManifest.xml` 文件中 `LoginActivity` 的配置里，增加一个 `Theme.AppCompat` 主题。您也可以使用自己的 `Theme.AppCompat` 主题。修复示例如图所示：

```
<activity
    android:name=".login.LoginActivity"
    android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
</activity>
```