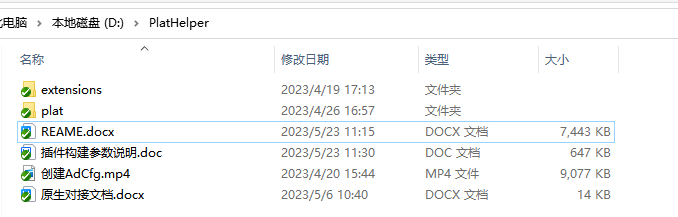
# 简陋的平台打包助手

更新日期：23.07.26

对应支持的 cc3平台版本：3.5.2,3.7.1

SVN地址：https://desktop-dvdsv79/svn/cgdr168/trunk/PlatHelper

#### 文件结构

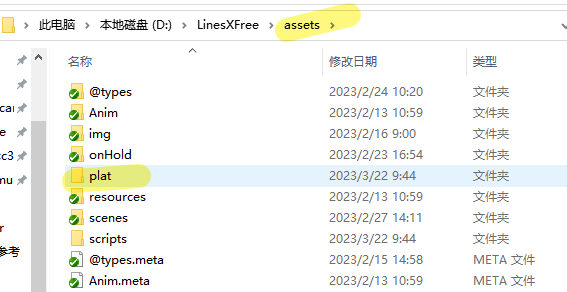


plat 脚本代码

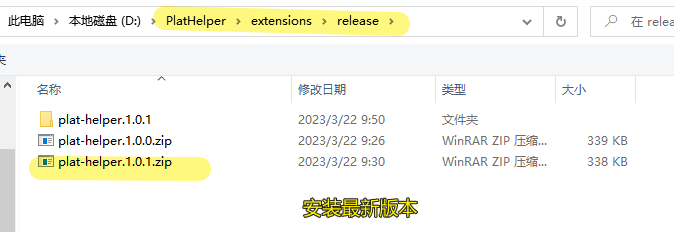
extensions 插件

#### 安装及使用

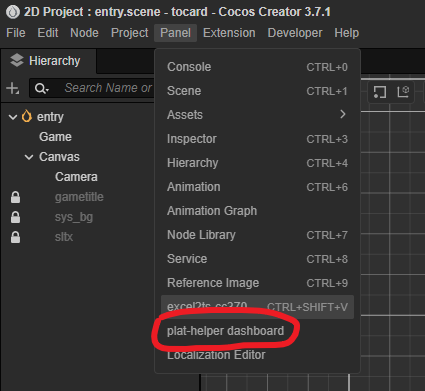
1. **将 plat 移动到cc项目的assets 内：**



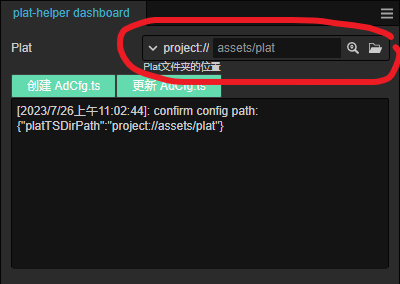
1. **安装插件。插件地址：**



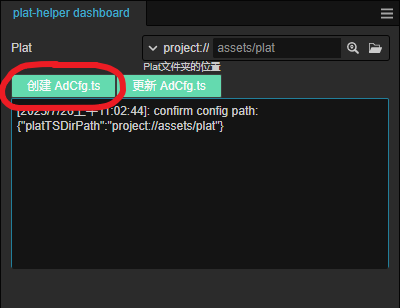
1. **打开插件面板：**



1. **更新 plat 路径信息**



1. **创建 AdCfg.ts：点击一下 “创建AdCfg.ts” 即可**





插件安装完成后，在构建界面中会有额外的选项，请按要求填写。

具体填写规则参考下面这个附件



# TS中使用

##### 必接

1. 在游戏入口调用

Plat.inst.init({ 按照注释填写初始项 });

Plat.inst.setLoadingProgress(0);

1. 在确定玩家同意 用户协议/隐私声明 后调用

Plat.inst.setLoadingProgress(33);

Plat.inst.checkPlatReady(() => {

    Plat.inst.setLoadingProgress(66);

    Plat.inst.login(() => {

        Plat.inst.loadingComplete();

    }, uid);

});

**checkPlatReady**

等待原生平台原生代码初始化完毕。

**login**

处理“登录”及“防沉迷”

如果没有账户系统，uid可参考如下代码生成

let uid = sys.localStorage.getItem('NOW')

if (!uid) {

    uid = Date.now().toString() + "\_" + (Math.random() \* 100 << 0);

    sys.localStorage.setItem('NOW', uid);

}

1. 退出游戏时统一调用：

Plat.inst.endGame();

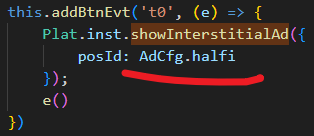
##### 广告播放

AdCfg.ts 介绍

配置时：



使用：



AdCfg.halfi 的实际值在不同的平台上将根据配置调整。

接口

激励视频 showRewardAd

Banner showBannerAd

hideBannerAd

原生/模板广告 showTemplateAd

hideTemplateAd

插屏 showInterstitialAd

##### 热更新

如果存在 热更新 逻辑，热更新逻辑必须放在 Plat.inst.login 调用之前。

Plat.inst.checkPlatReady(() => {

// 热更新逻辑可以放在这。或则在 checkPlatReady 之前。

Hotfix.me.do(() => {

Plat.inst.setLoadingProgress(66);

    Plat.inst.login(() => {

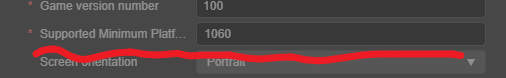
        Plat.inst.loadingComplete();

    }, uid);

});

});

# 小游戏打包



最小平台版本号推荐 参考 构建项 中的 [查表]最小支持平台版本

## 引擎修复

【3.7.1 上小米打包时需处理】[3.7.1 空包小米快游戏 crash - Creator 3.x - Cocos中文社区](https://forum.cocos.org/t/topic/146715/12)

【3.7.1 上华为打包时需处理】[华为快游戏由于引擎代码错误，导致资源加载出现问题，卡Loading页 - Creator 3.x - Cocos中文社区](https://forum.cocos.org/t/topic/151409)

# 安卓打包

## 引擎修复

【3.7.1/3.7.2/3.7.3 上安卓打横屏游戏时需处理】https://forum.cocos.org/t/topic/147427/5

## 环境搭建

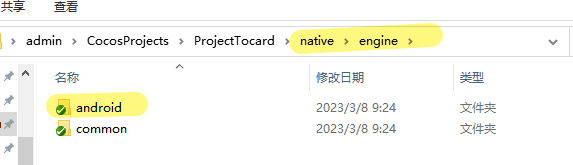


进入 [安装配置原生环境 · Cocos Creator](https://docs.cocos.com/creator/3.5/manual/zh/editor/publish/setup-native-development.html" \l "android-%E5%B9%B3%E5%8F%B0%E7%9B%B8%E5%85%B3%E4%BE%9D%E8%B5%96)，完成 Android平台相关依赖的安装

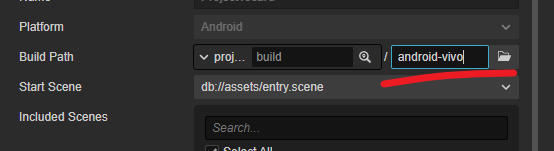
## 首次打包

下面以首次打vivo安卓包为例

1. 如果存在需要删除 native/engine/android



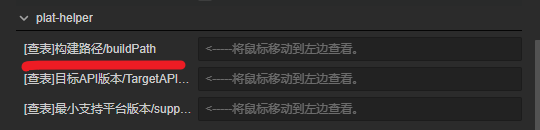
1. 新建打包配置



**BuildPath**

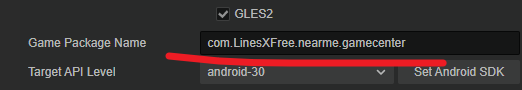
BuildPath 目录名命名为 android-vivo

（可参考 构建项 中的 [查表]构建路径）

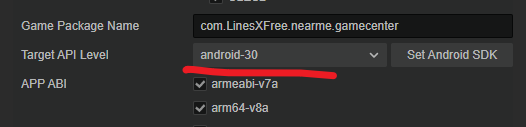


**包名**

请务必正确填写对应平台的管理后台中的包名



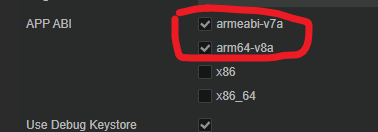
**TargetAPILevel 选择**



（可参考 构建项 中的 [查表]目标API版本）

**APP ABI**

APP ABI 勾上 armeabi-v7a 和 arm64-v8a



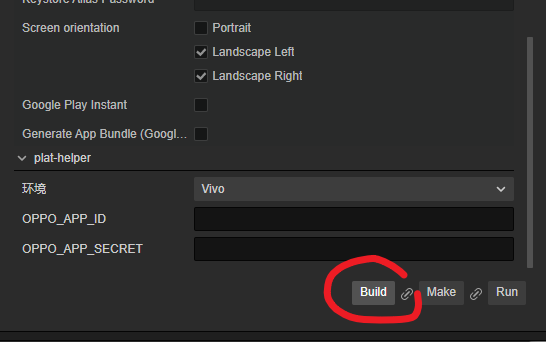
**Plat-helper**

参考

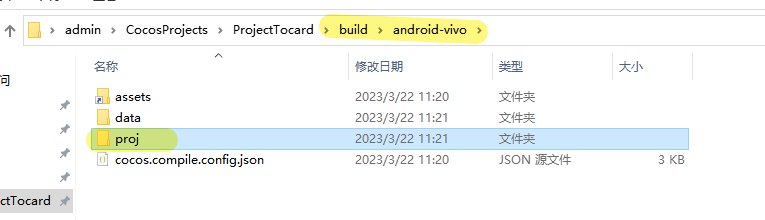


1. 构建。

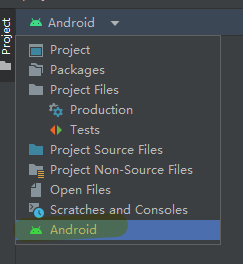
参数填写完毕后，点击 Build/构建



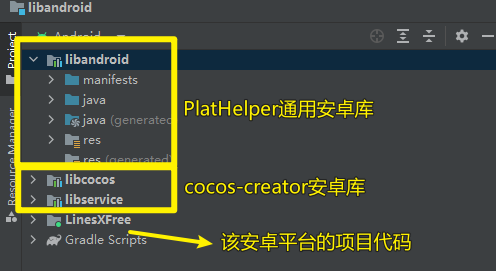
1. 等待构建结束。
2. 用Android-studio 打开安卓项目



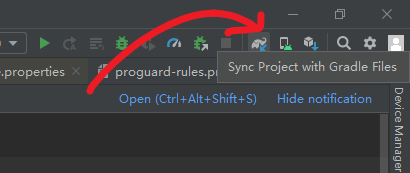
切换Project模式到 Android，如果没有这个选项，那么你可能项目地址选错了。



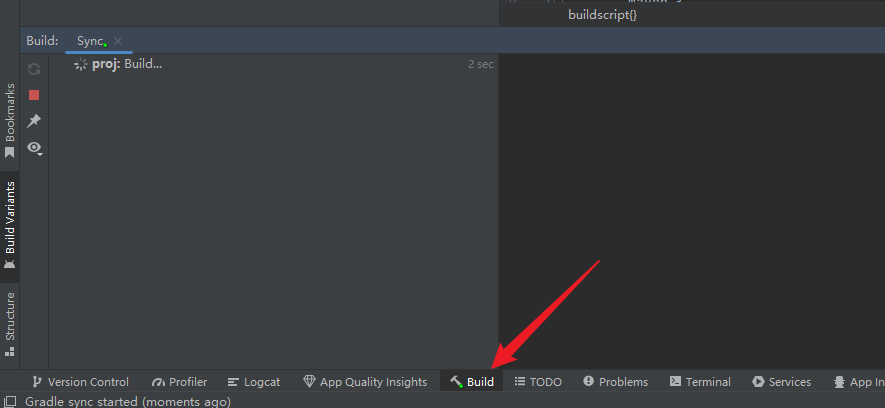
项目结构



1. GradleSync 同步安卓包。

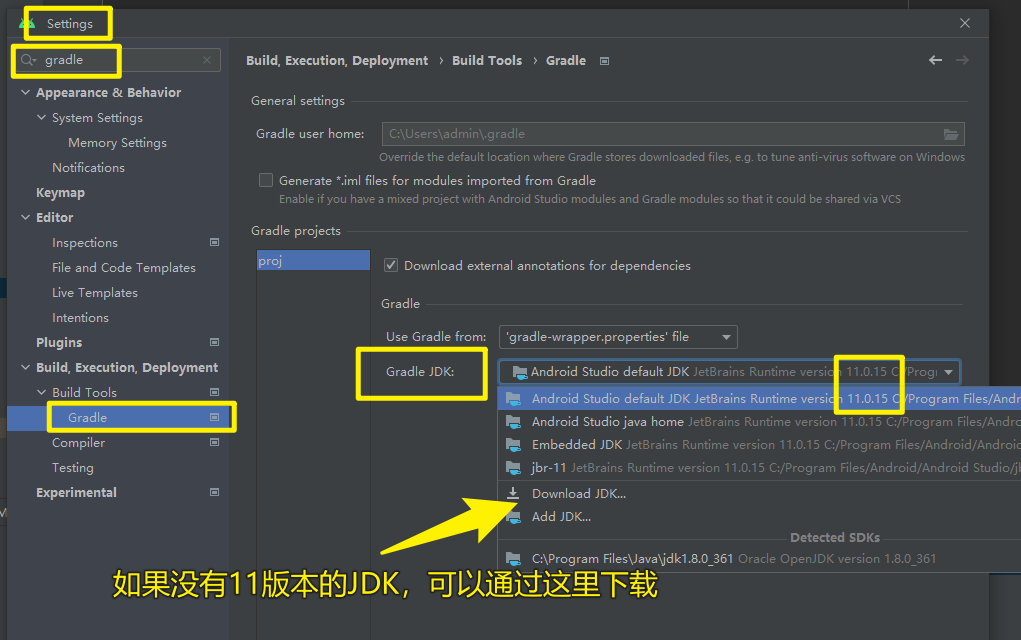


点击这个大象。

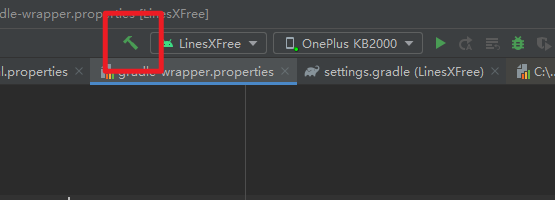


构建日志会显示在下面板的 Build 中。

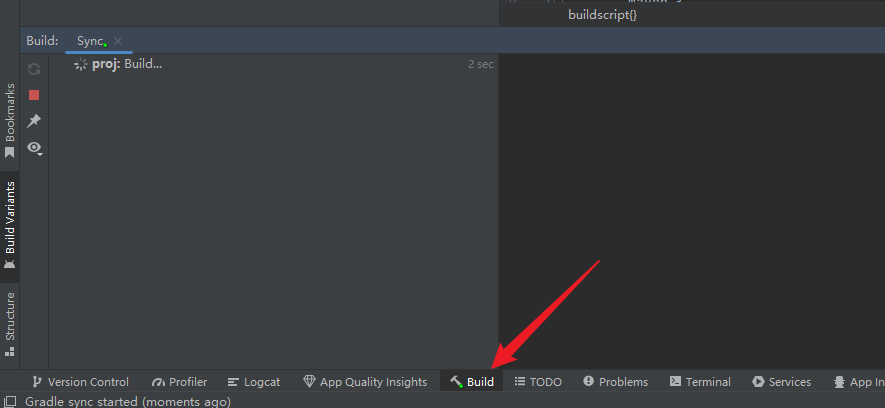
\* 期间如果报错，请确保当前 Gradle 使用的JDK版本为11。



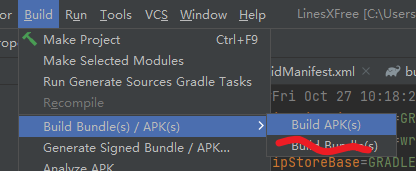
1. Make 构建。GradleSync 如果成功就可以尝试 Make。



构建日志会显示在下面板的 Build 中。



1. 构建 APK。Make如果成功，那么就可以构建APK了。

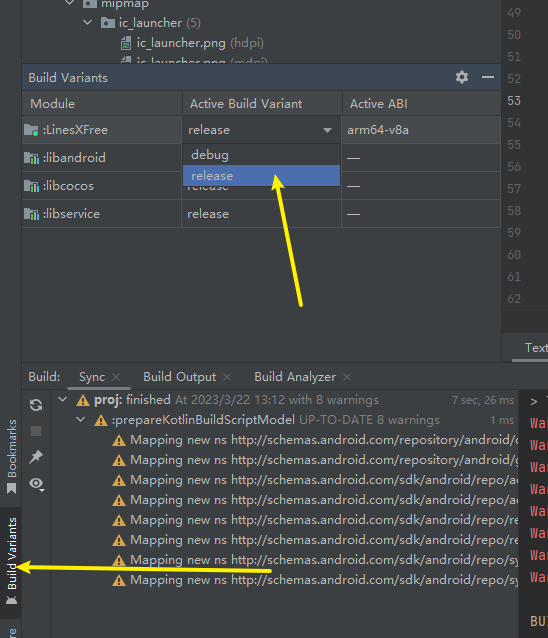


**Build Variants 变体**

\* 如果你正在调试代码，那么建议你切换 Variant 为debug，这样构建的时候会快一点。注意四个Module都要改为 debug。

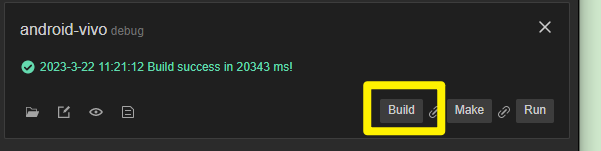
\* 如果你打算出正式包，请务必切换为 release 。

\* TapTap包只能在release下调试。



**后续打包流程**

1. Build



1. 打开 android-studio 项目
2. 选择 Build Variants 变体
3. 构建 APK。

Q: 如何修改应用名称和应用图标

A：修改 native/engine 下平台项目文件夹的 res文件夹中的内容。

