

操作手册

0.0 系统要求及预安装

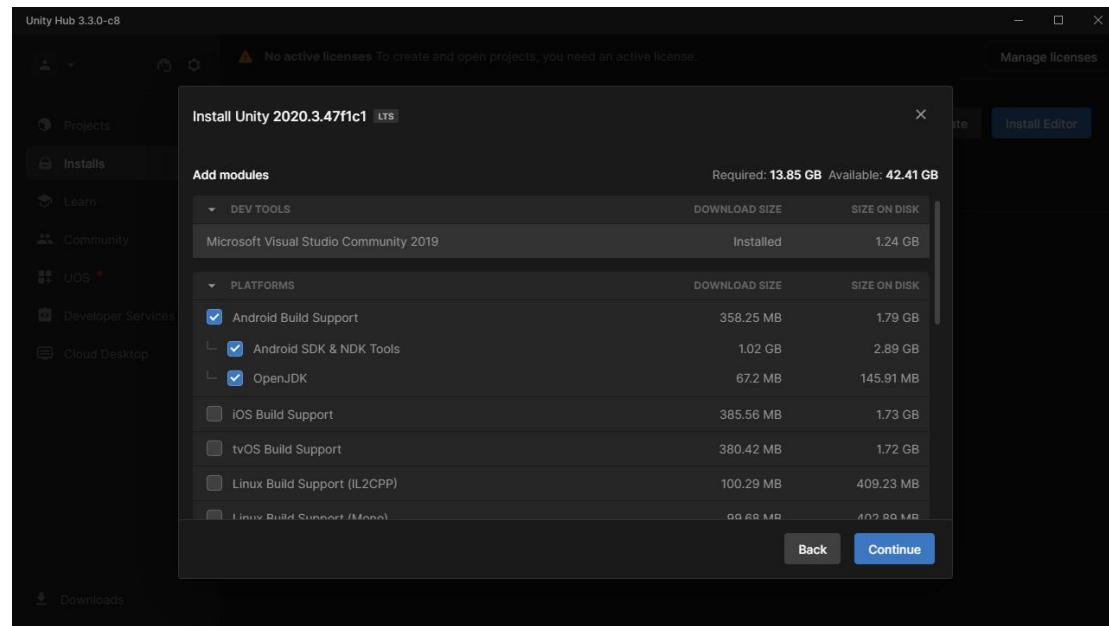
开发系统: Windows 8.0 及以上

预安装软件: 1. Unity Hub; 2. Visual Studio 2019

运行设备: Android 智能手机

0.0.1 Unity Editor 安装

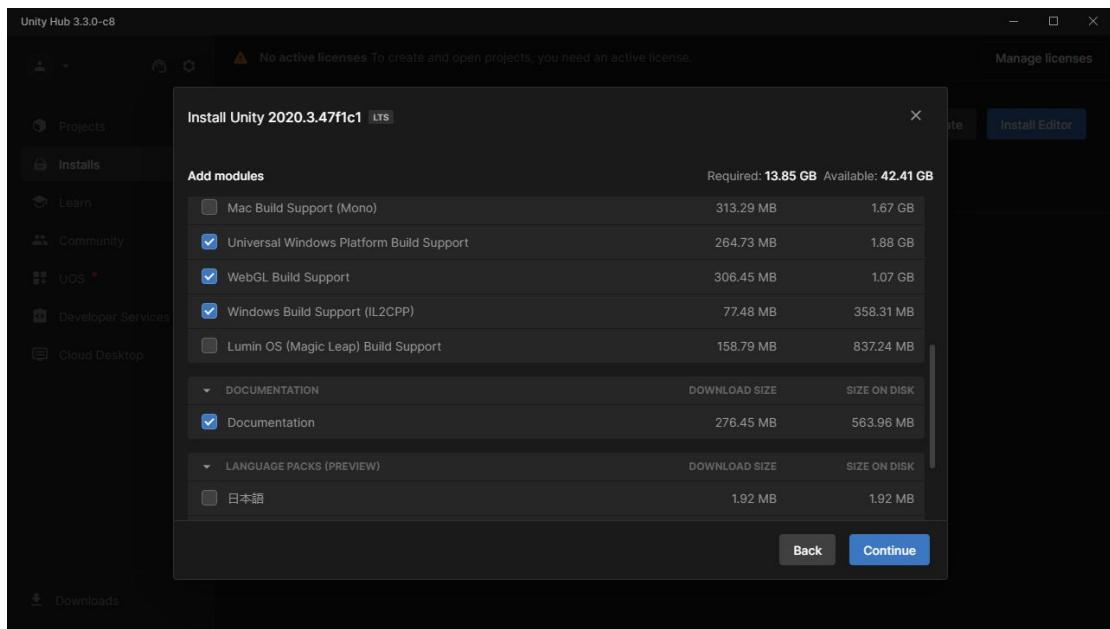
打开 Unity Hub 软件, 从 Installs 边菜单里选择所需安装的 Unity editor 版本 (需要 2020.3.36f1 或更高版本, 建议选择长期支持版本), 并在 Add modules 中勾选 Android Build Support



除了 Android Build Support 以外, 建议也将 Universal Windows Platform Build Support, Windows Build Support (IL2CPP)勾选上, 以备不时之需。

WebGL Build Support 视个人需要可勾选上。

安装全部所需 Modules 大致需要 10 至 20GB 硬盘空间, 请预留足够空间后进行安装。

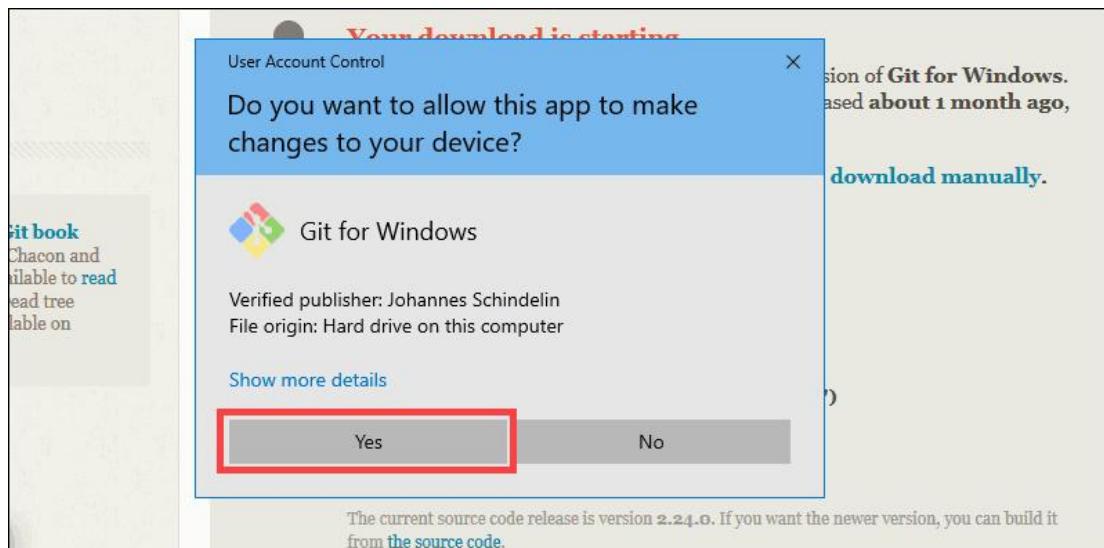


除了以上软件以外，也需要安装 Git 以允许 Unity 从网络服务器下载必要组件和软件包等。浏览器访问 Git 官方网站：<https://git-scm.com/downloads>

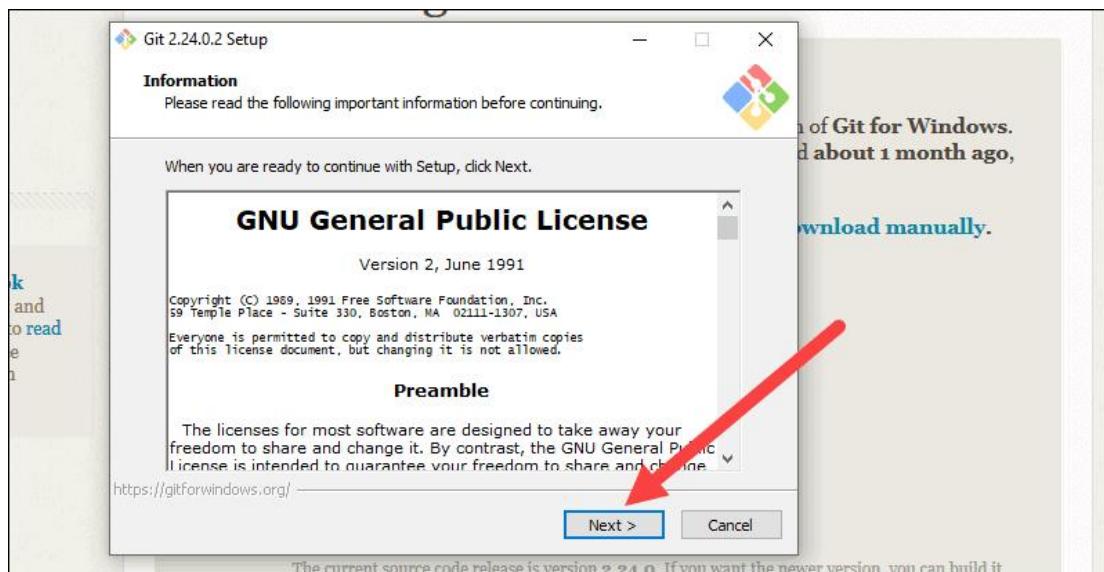
单击 Windows 的下载链接并允许下载完成。



启动 git 安装程序



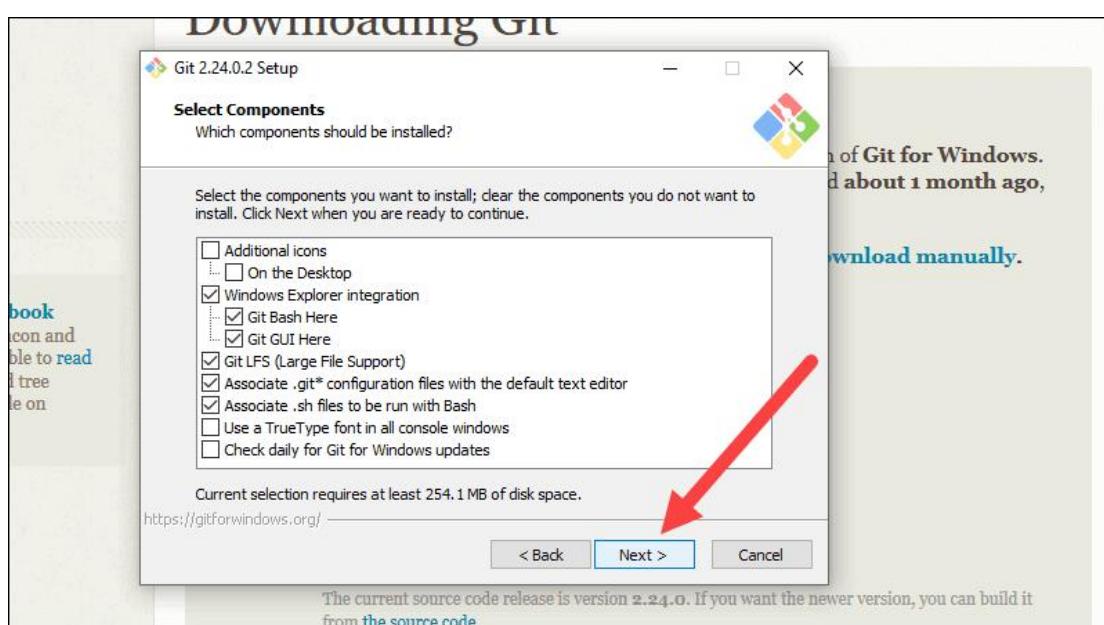
点击 Next



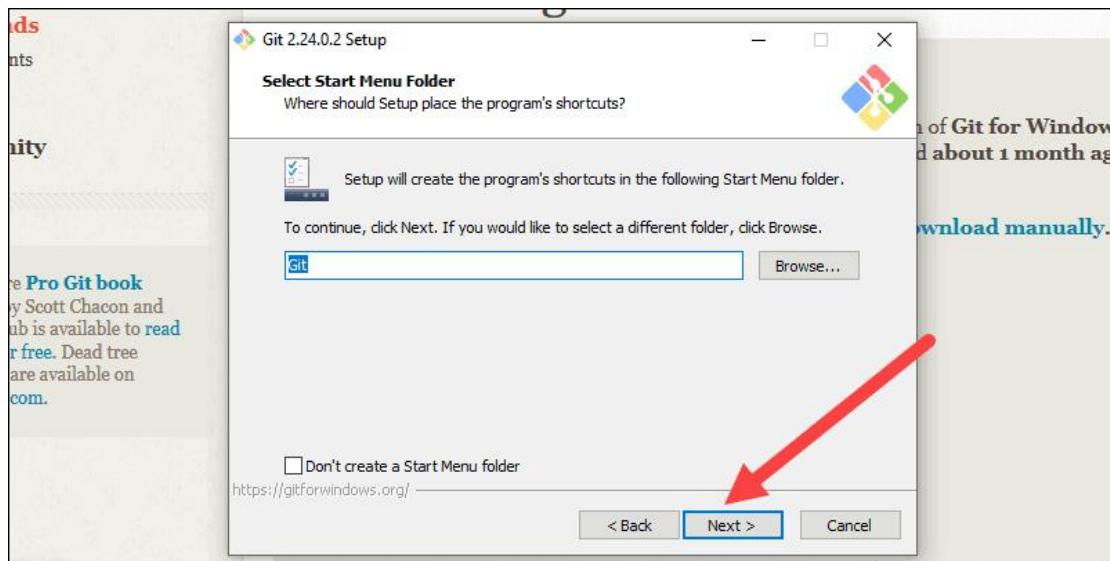
选择安装路径，尽量选非中文路径



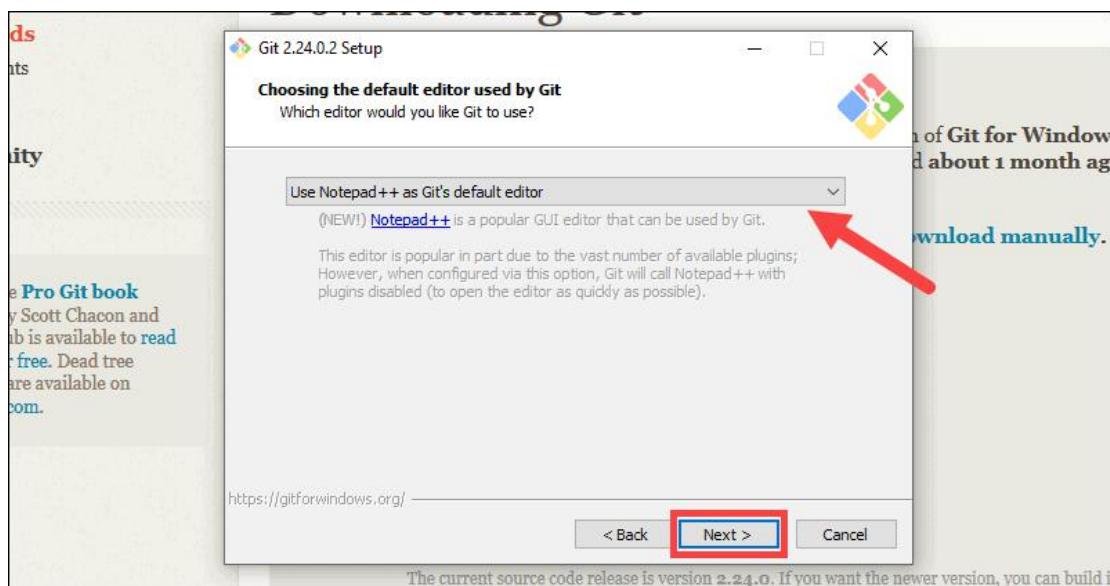
使用安装程序的默认选项



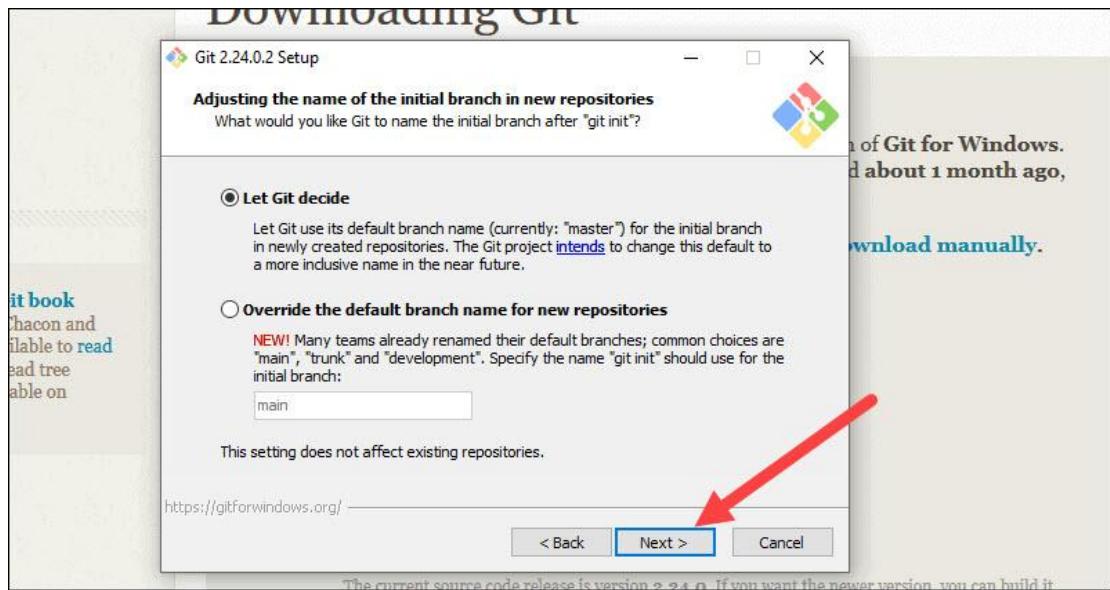
.安装程序将提供创建开始菜单文件夹。只需单击下一步



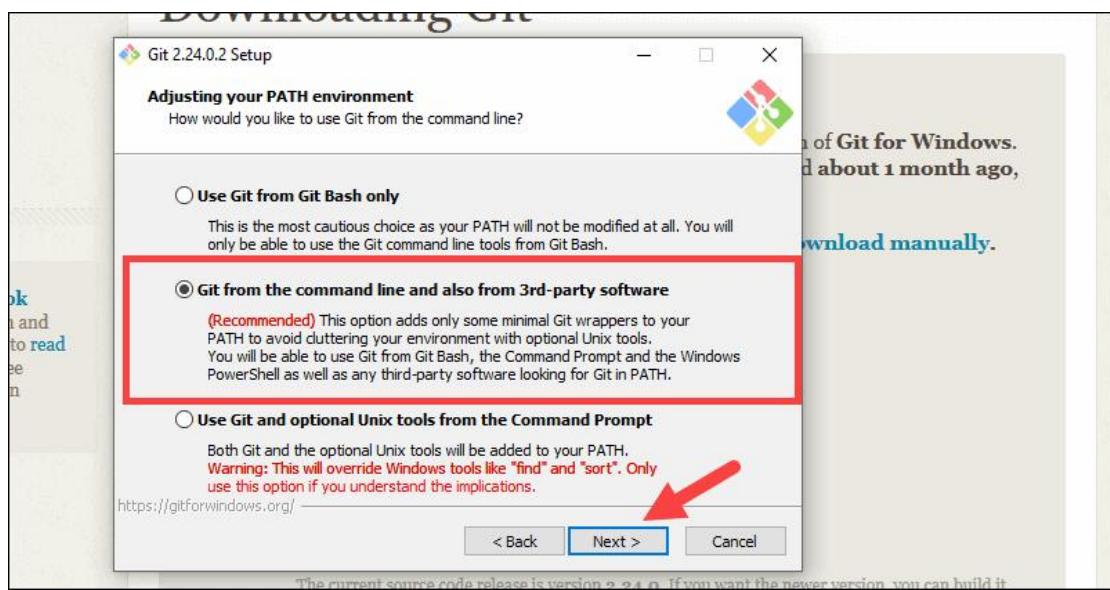
选择要与 Git 一起使用的文本编辑器。使用下拉菜单选择 Notepad++（或任何文本编辑器），然后单击下一步。



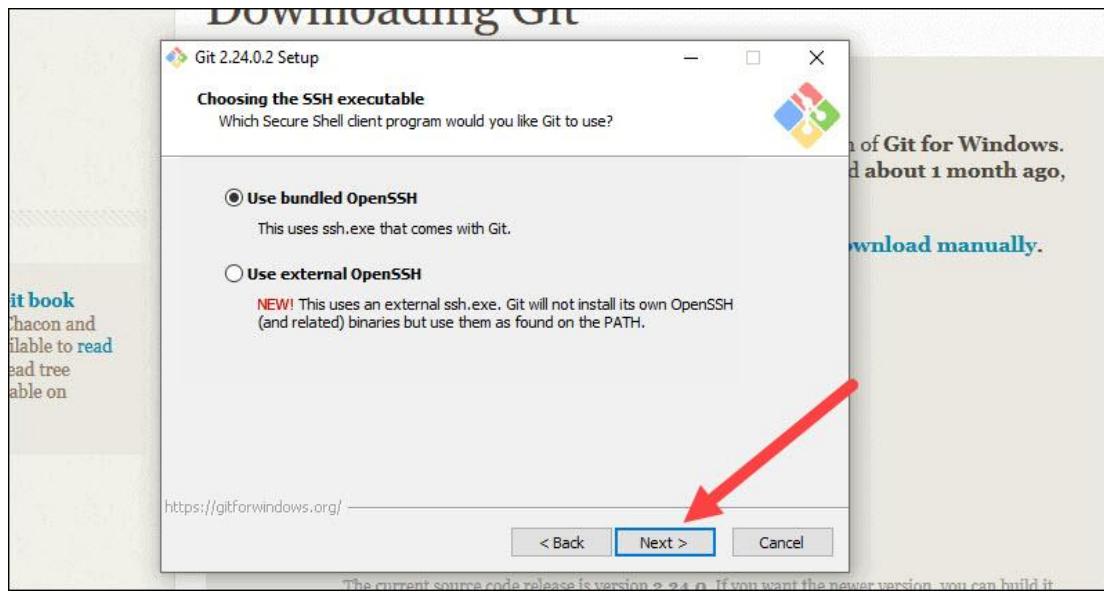
默认选项



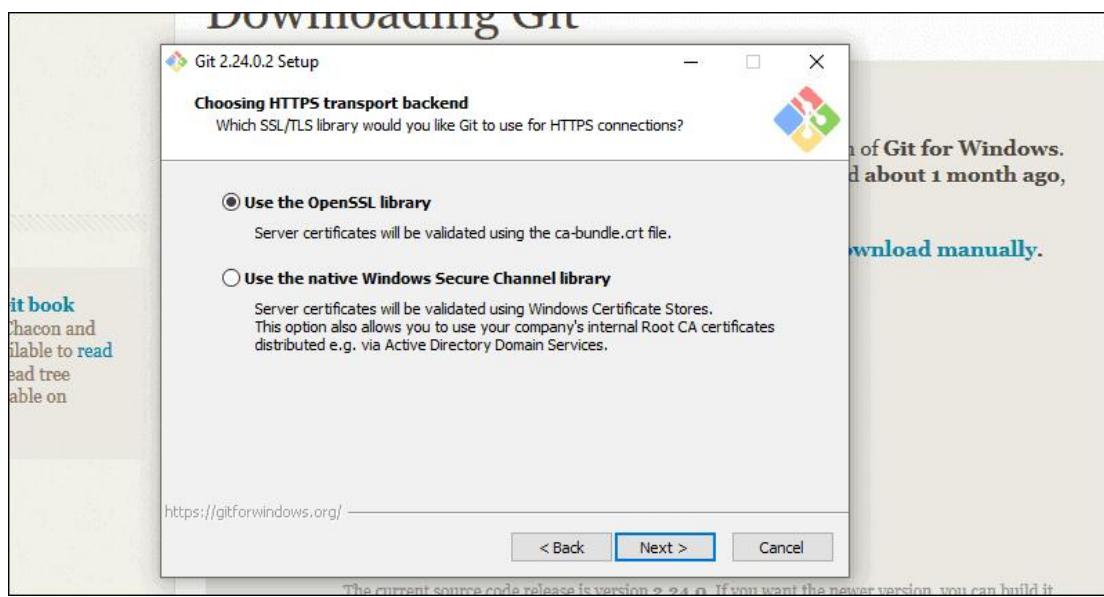
此安装步骤允许更改 **PATH** 环境。**PATH** 是从命令行运行命令时包含的默认目录集。将其保留在中间（推荐）选择中，然后单击下一步。**注意：此步骤非常关键！**



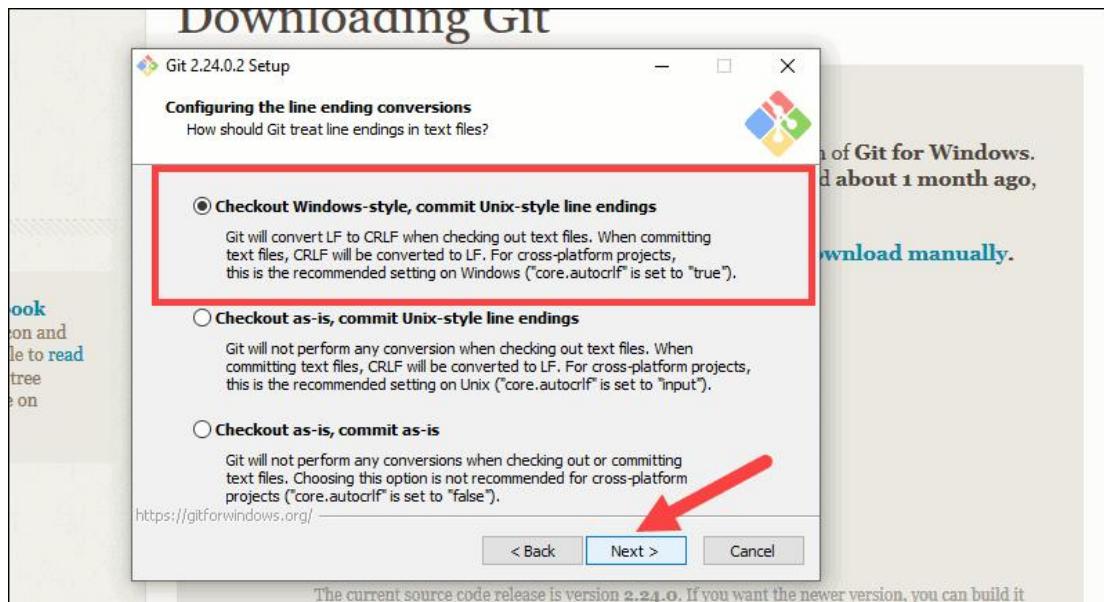
安装程序现在会询问你希望 Git 使用哪个 SSH 客户端。Git 已经带有自己的 SSH 客户端，因此如果不需要特定的客户端，请保留默认选项并单击下一步。



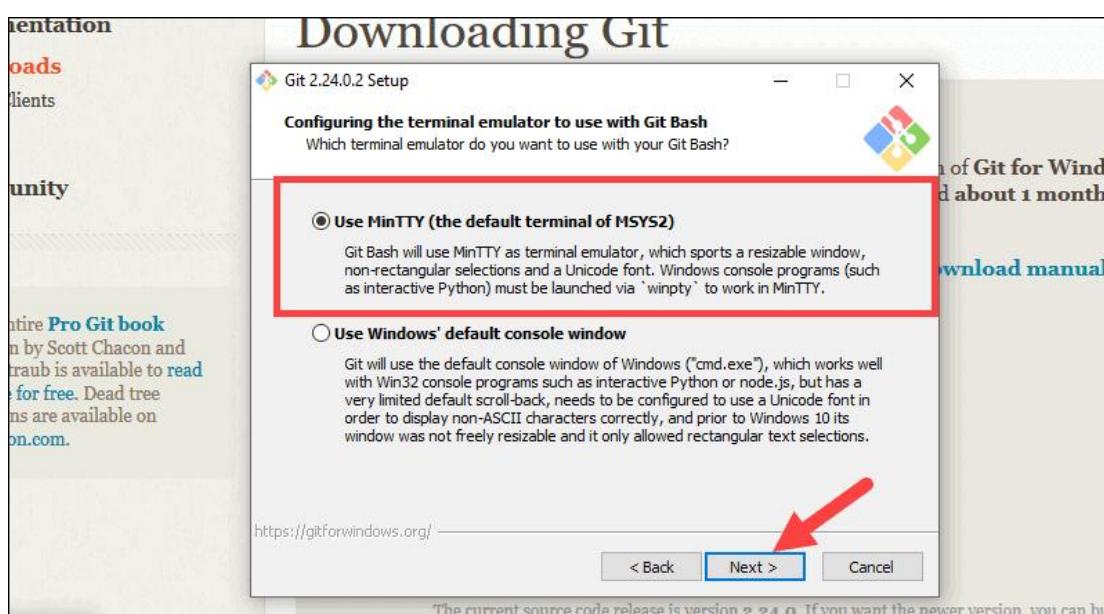
下一个选项与服务器证书有关。大多数用户应使用默认值。如果你在 Active Directory 环境中工作，则可能需要切换到 Windows 应用商店证书。单击下一步。



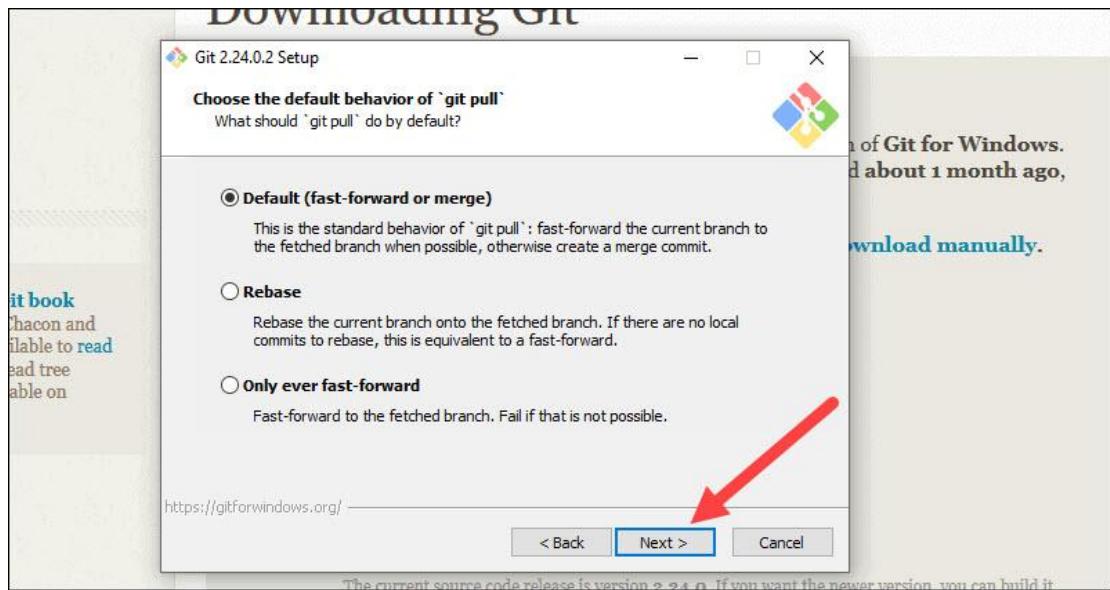
下一个选择转换行尾。建议保留默认选择。这与数据的格式设置方式有关，更改此选项可能会导致问题。单击下一步。



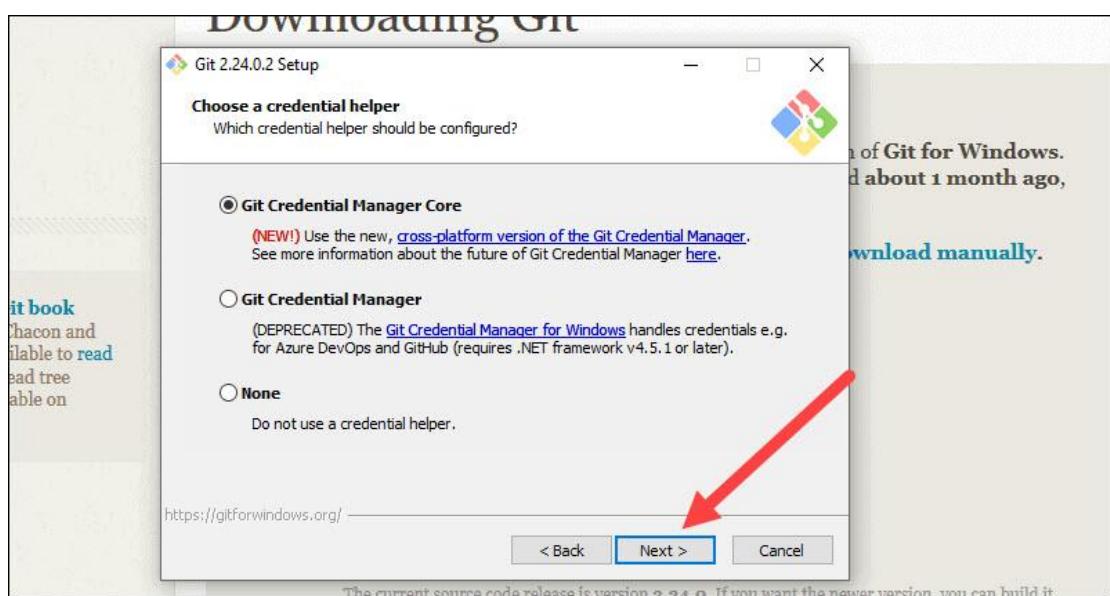
选择要使用的**终端模拟器**。对于其功能，建议使用默认的 MinTTY。单击**下一步**。



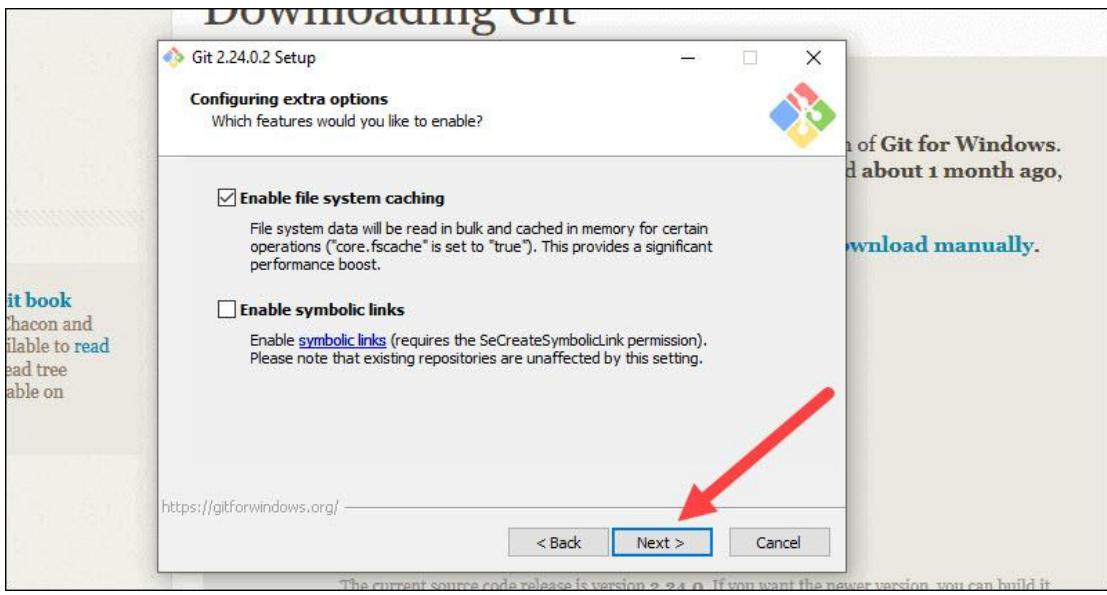
安装程序现在询问 **git pull** 命令应该做什么。建议使用默认选项，除非特别需要更改其行为。单击**下一步**继续安装。



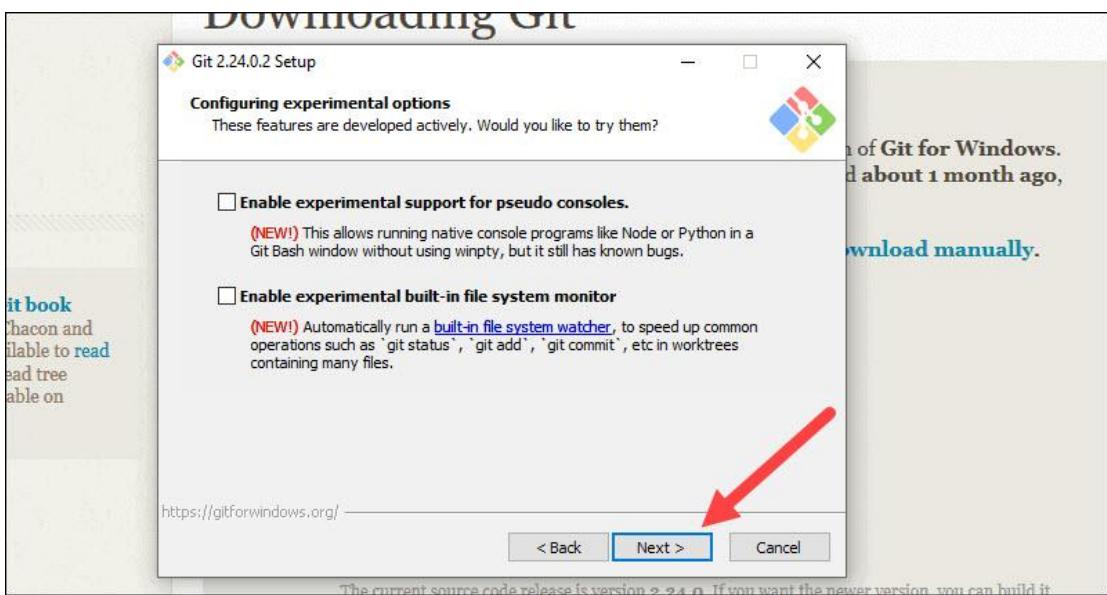
接下来，选择要使用的凭据助手。Git 使用凭据帮助程序来获取或保存凭据。保留默认选项，因为它是最稳定的选项，然后单击下一步。



建议使用默认选项，但此步骤允许你决定要启用的额外选项。如果您使用符号链接（类似于命令行的快捷方式），请勾选该框。单击下一步。

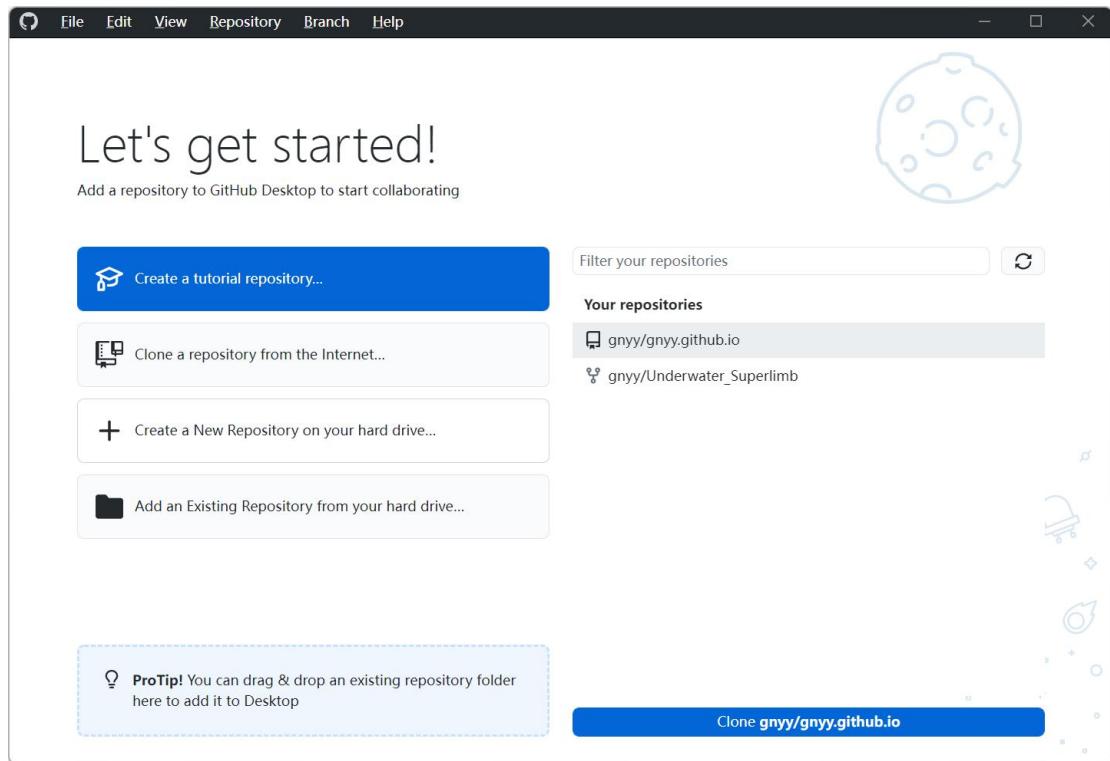


根据安装的 Git 版本，它可能会提供安装实验性功能。建议将其保留为未选中状态，然后单击“安装”。

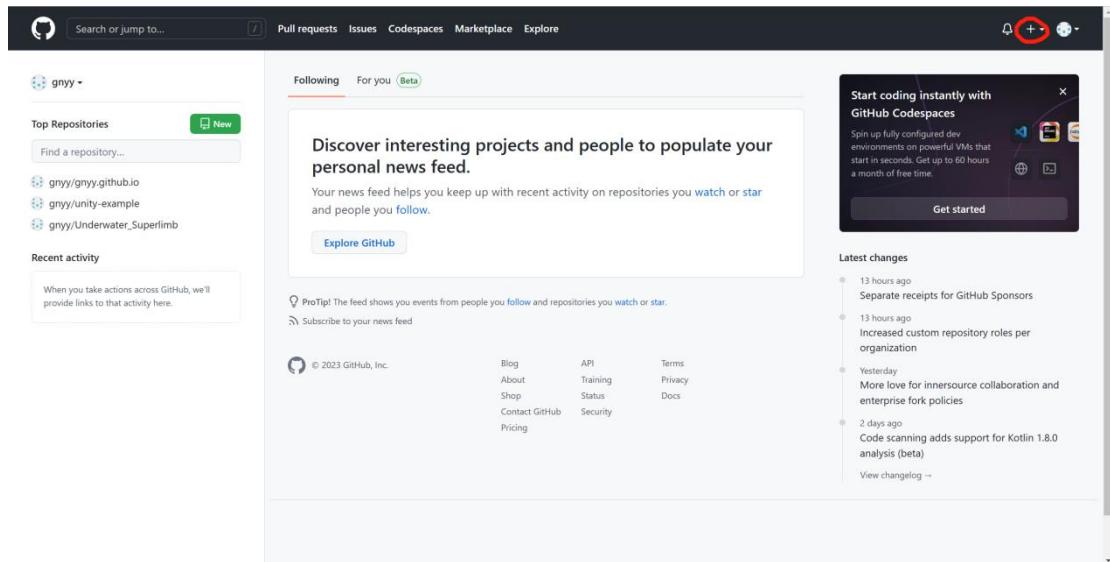


1.0 使用 Github 新建及管理 Unity 项目

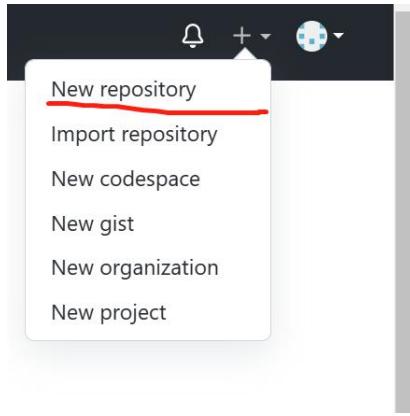
下载并安装 Github Desktop (<https://desktop.github.com>)，如无 Github 账户，请自行注册后登陆。



在 github 的远程仓库创建一个自己的新项目，点击下图所示的红圈标注的加号



选择 New repository



在后续配置页面中修改下面两项

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

gnny / unity-example1

Great repository names are short and memorable. Need inspiration? How about [improved-octo-spork](#)?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license A license tells others what they can and can't do with your code. [Learn more.](#)

License: None

You are creating a public repository in your personal account.

①中填写你的项目名称，如 `unity-example`，②中展开复选框搜索选择 `unity`, 如下图所示：

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

gnny / unity-example1 ✓

Great repository names are short and memorable. Need inspiration? How about [improved-octo-spork](#)?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Unity

Choose a license A license tells others what they can and can't do with your code. [Learn more.](#)

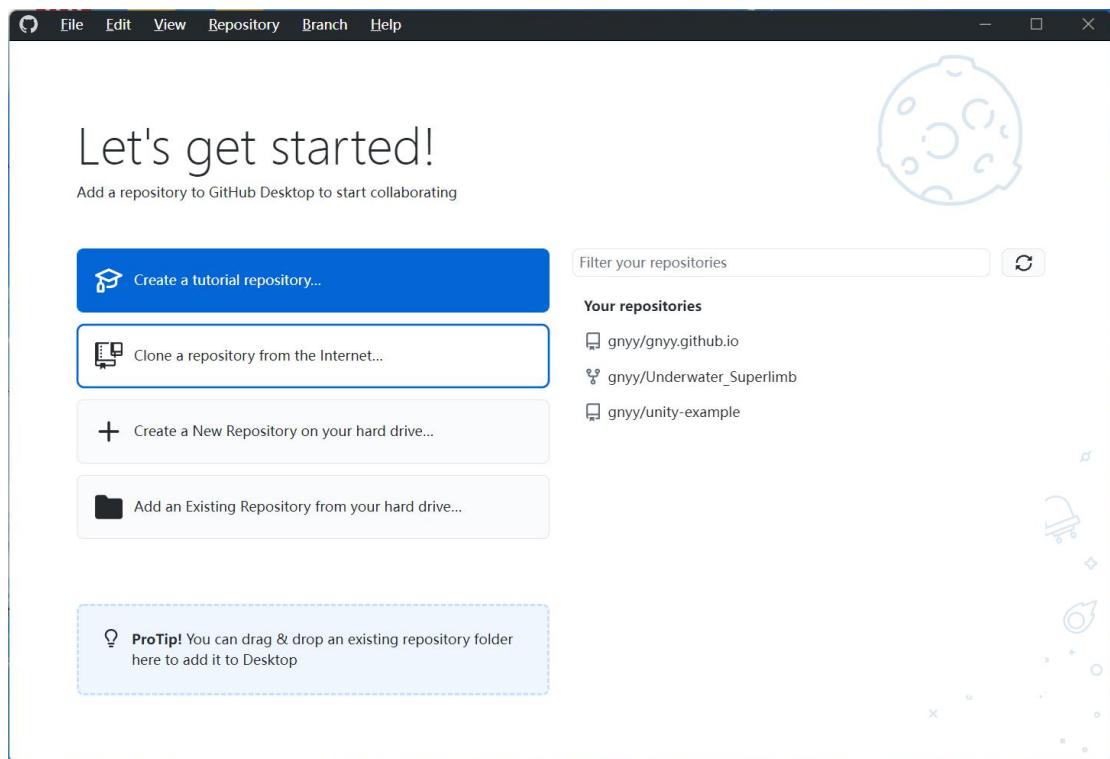
License: None

You are creating a public repository in your personal account.

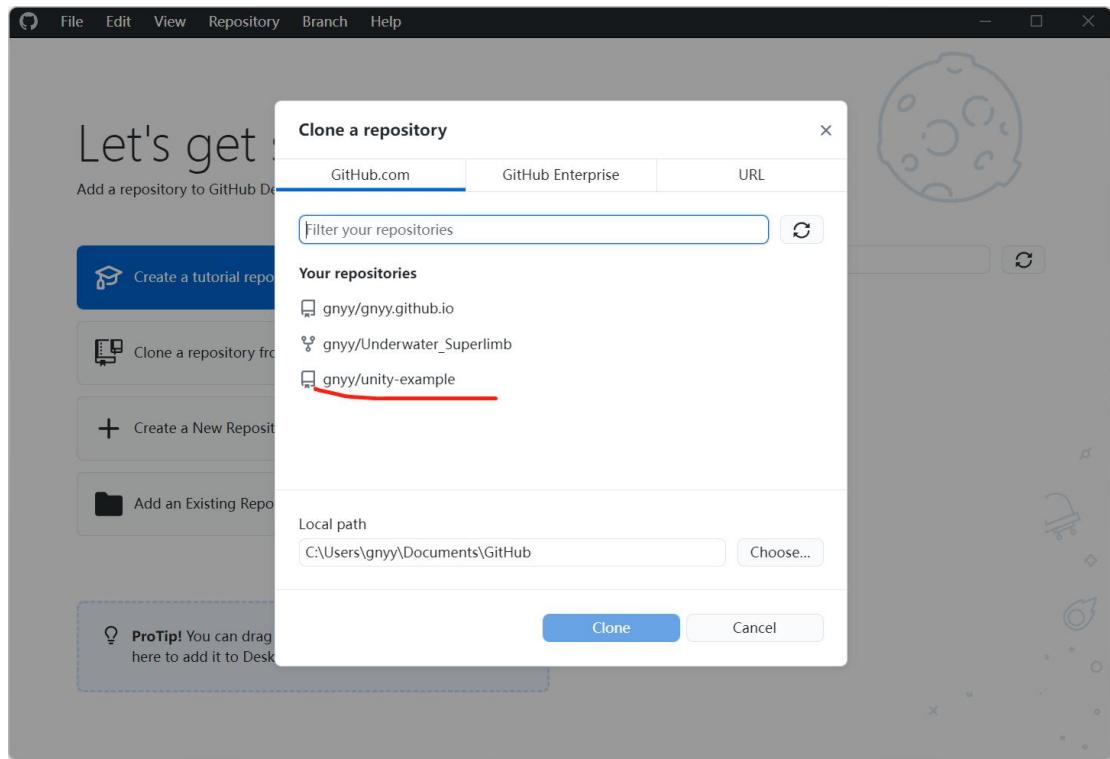
创建成功后的界面如下图所示：

The screenshot shows a GitHub repository page for 'gnyy/unity-example'. At the top, there's a search bar and navigation links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the header, there are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Code' tab is selected, showing a main branch with 1 branch and 0 tags. A commit titled 'gnyy Initial commit' is listed, made by 'gnyy' 4 minutes ago with 1 commit. Below the commit is a '.gitignore' file. On the right side, there's an 'About' section with a message: 'No description, website, or topics provided.' It also shows 0 stars, 1 watching, and 0 forks. Other sections include Releases (no releases), Packages (no packages), and a footer with links to Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

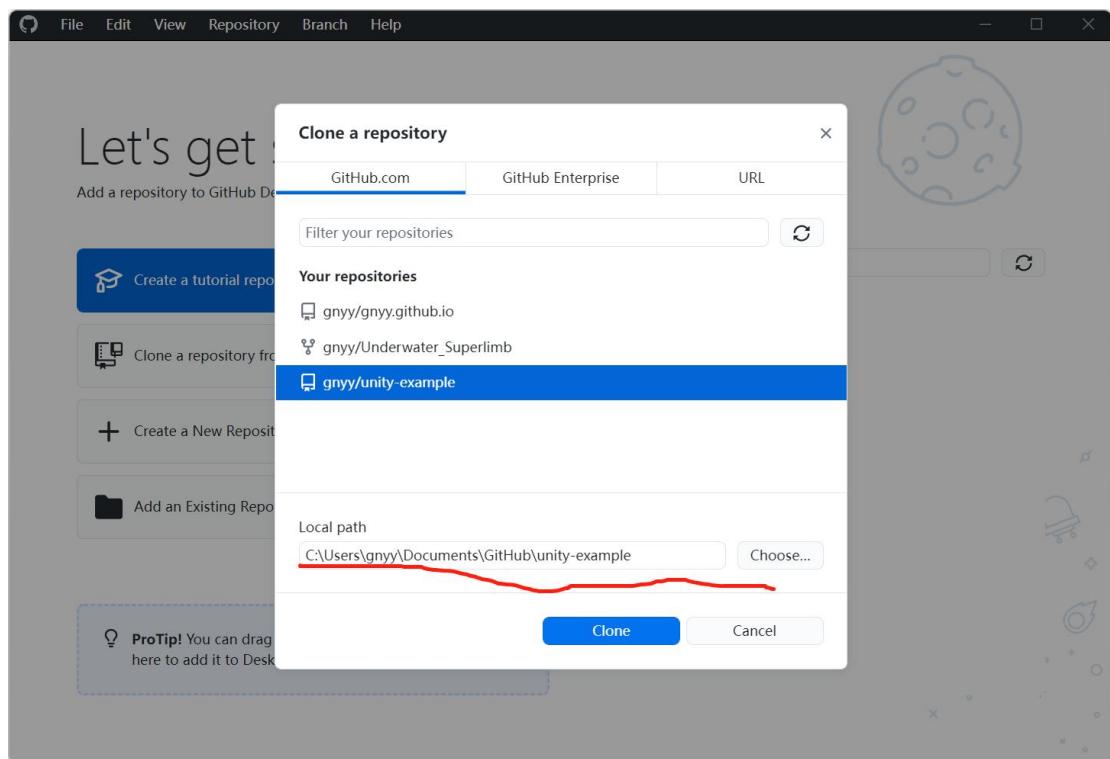
打开 Github Desktop,点击 Clone a repository from the internet



选择刚刚创建的仓库 unity-example



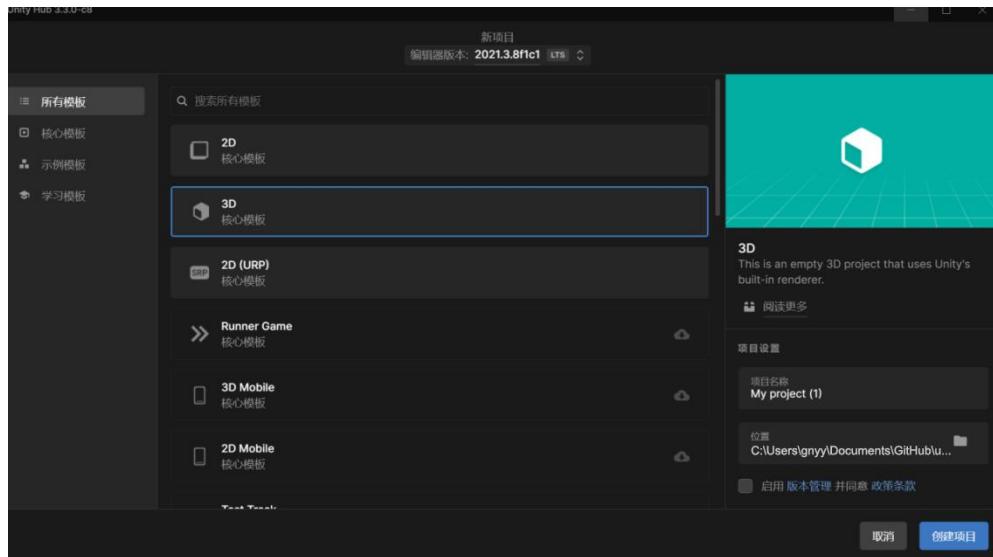
注意克隆至本地的文件路径，该文件路径将会作为 Unity 项目的本地文件路径使用，**注意选择不包含中文字符的文件夹作为项目路径！**



打开 Unity Hub，点击新项目后，选择 3D 模板，设置项目路径为上面 Github 远程代码库对应的本地路径：

建议取消勾选版本管理，容易导致问题

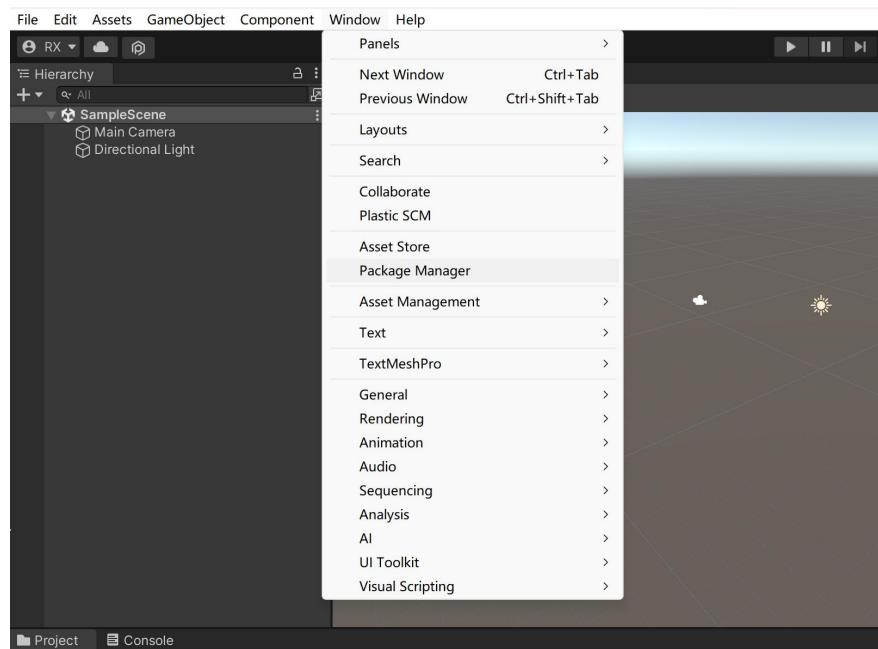
点击创建项目



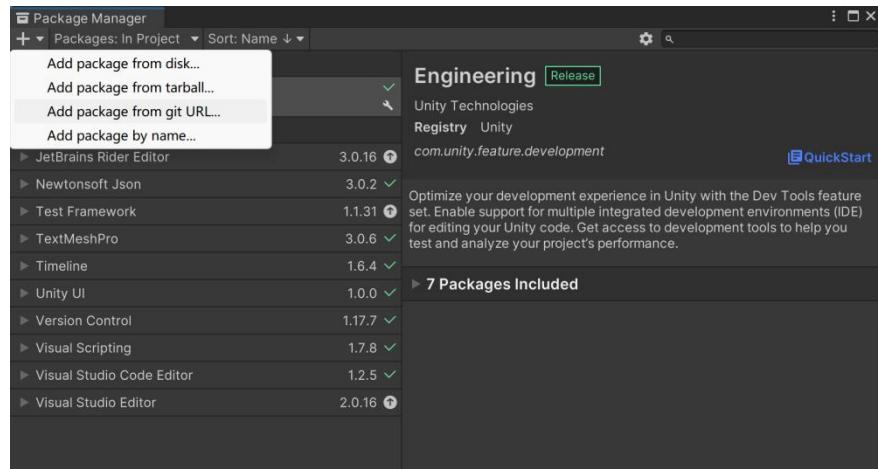
1.1 配置及运行 Google Cardboard

1.1.1 下载 Google Cardboard SDK

点击 Unity 菜单中的 Window，打开 Package Manager

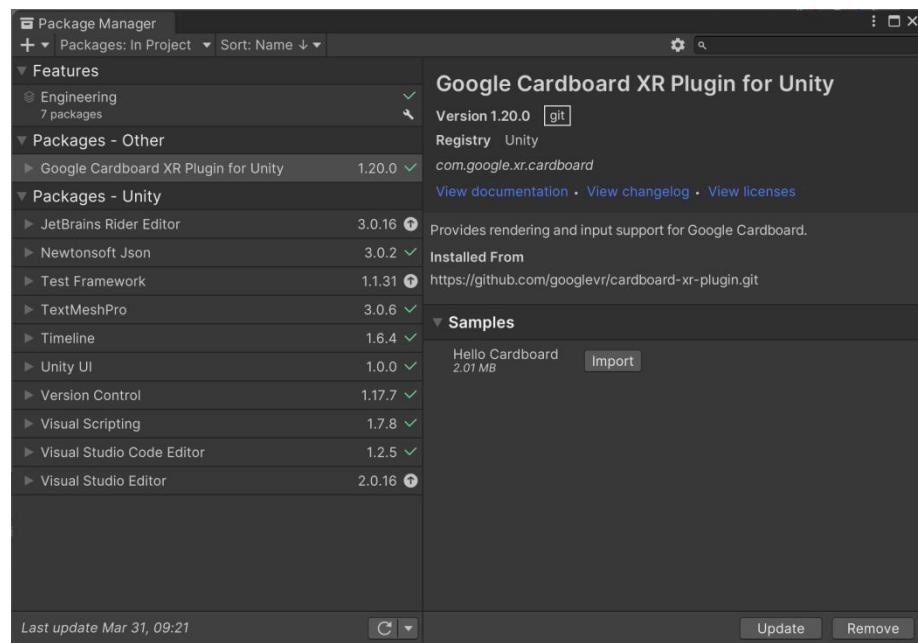


点击+号，选择 Add package from git URL



将 <https://github.com/googlevr/cardboard-xr-plugin.git> 粘贴到文本输入字段中，点击 add。注意：该步骤可能需要使用 VPN 全局系统代理！否则有可能无法下载 github 来源的软件包！

下载可能需要一段时间，完成后在显示的 package 页面上下拉打开 Samples，选择点击 import 导入素材包

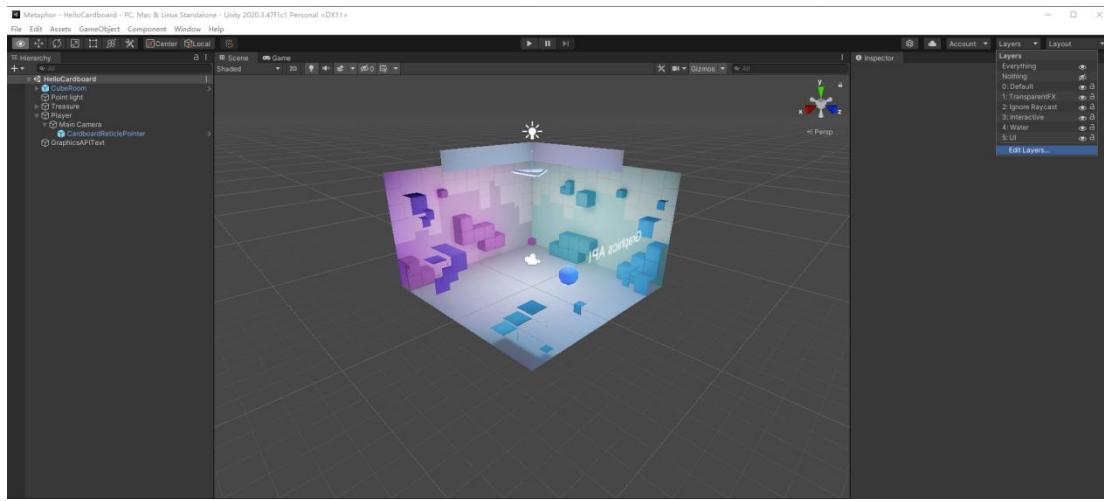


1.1.2 示例场景配置

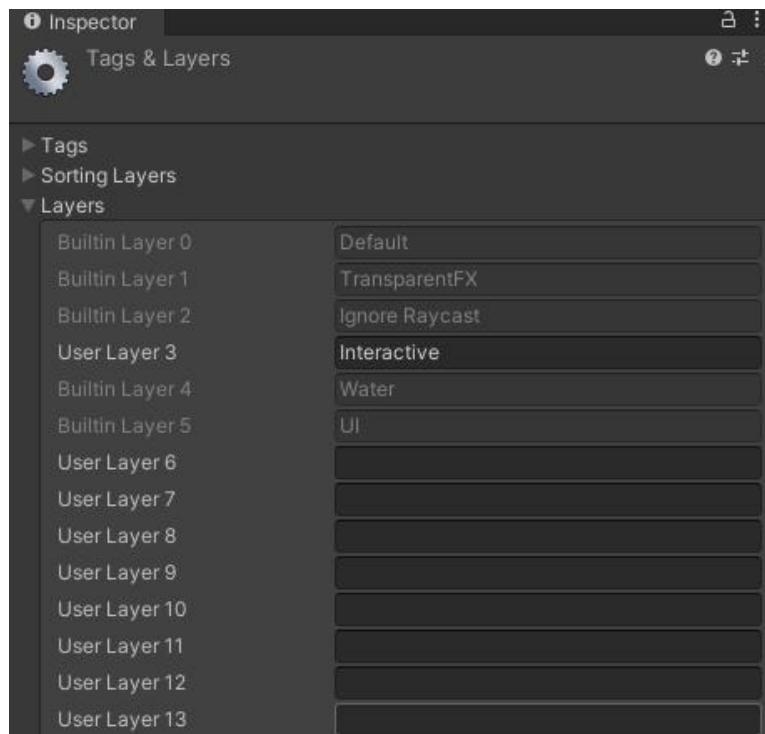
在项目 Assets 窗口中，依次点击 Samples -> Goggle Cardboard XR Plugin for Unity -> 1.19.0 -> Hello Cardboard -> Scenes

双击打开该目录下的 HelloCardboard 场景，加载到 unity 中去

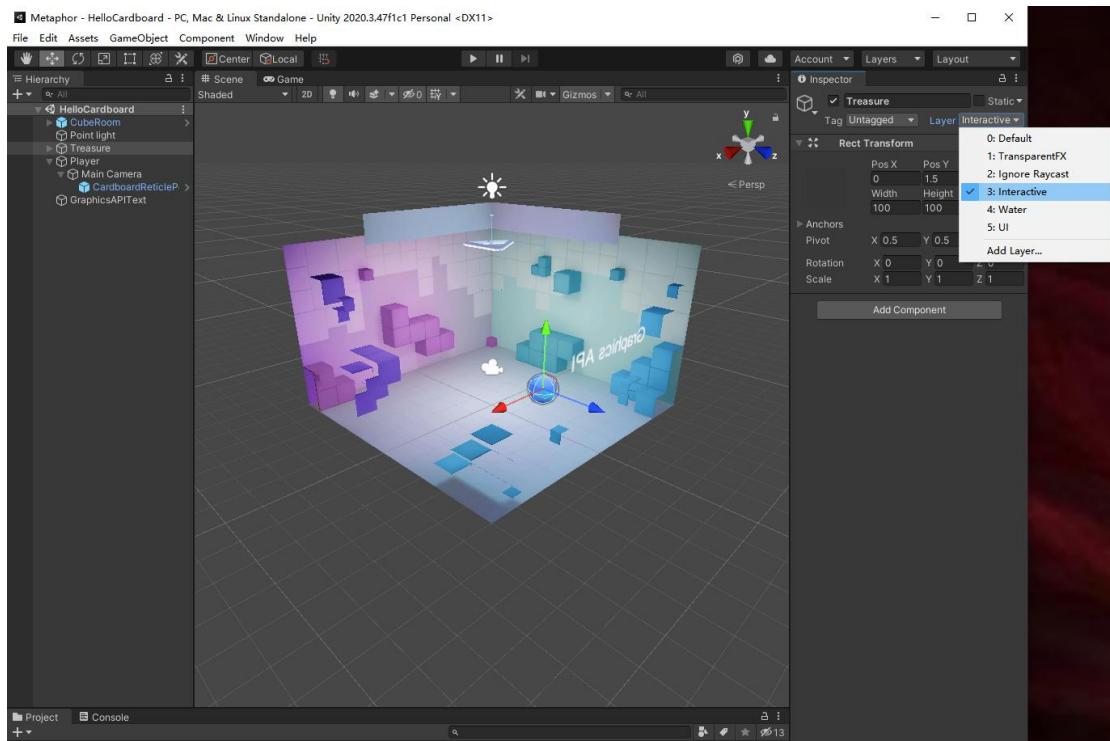
点击 Unity Editor 右上角 Layers 下拉菜单，选择 Edit Layers...



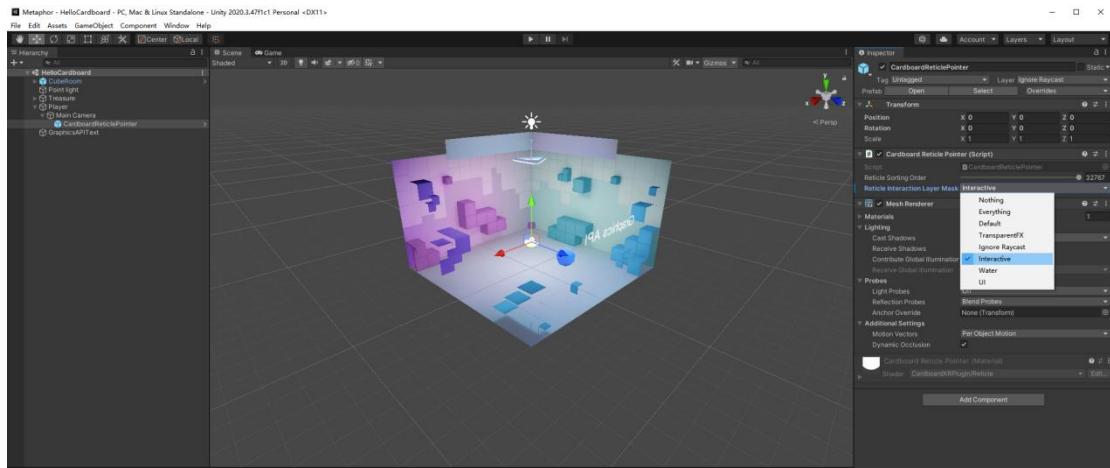
在 Layers 中新定义一个名为 Interactive 的图层



在 Hierarchy 中点击 Treasure 对象，在 Inspector 里将该对象的 Layer 设为 Interactive



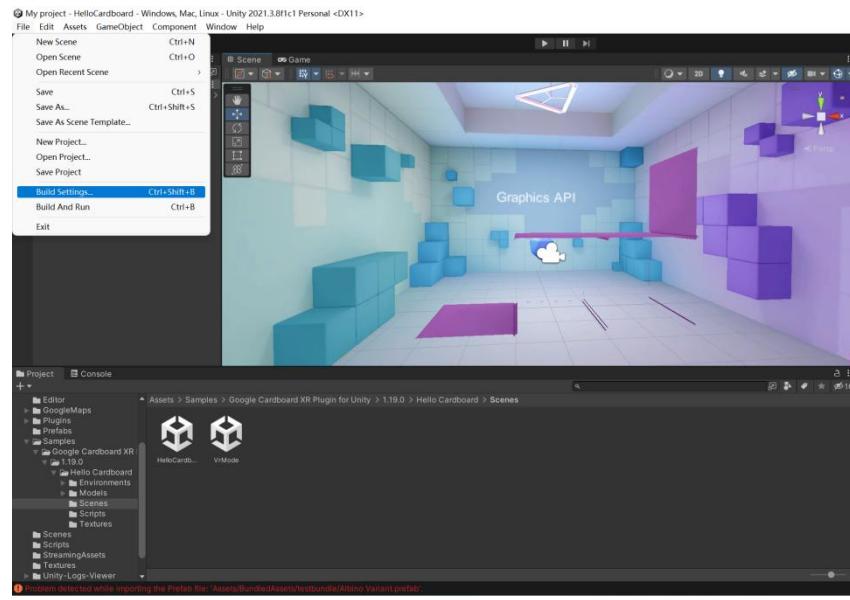
以同样的方法找到 Player -> Main Camera -> CardboardReticlePointer，在 Inspector 中设置 Reticle Interaction Layer Mask 的图层为 Interactive



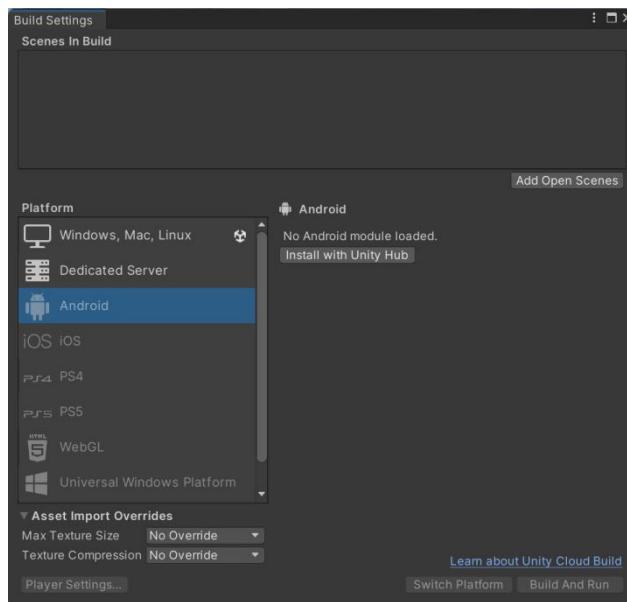
HelloCardboard 场景配置完毕

1.1.3 Build Setting 配置

点击导航栏 File, 选中 Build Settings



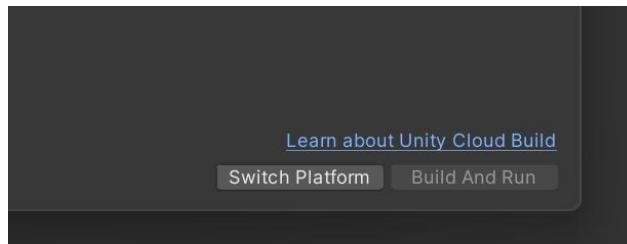
相关界面将如下所示：



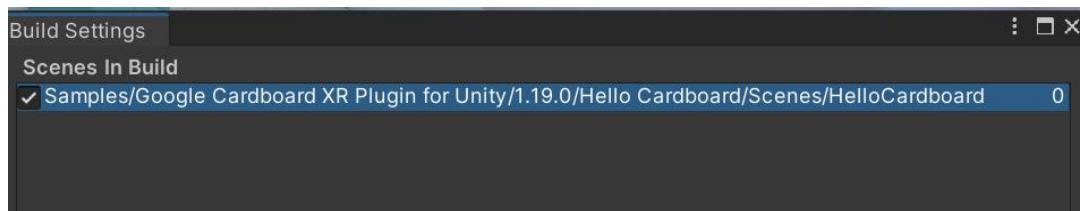
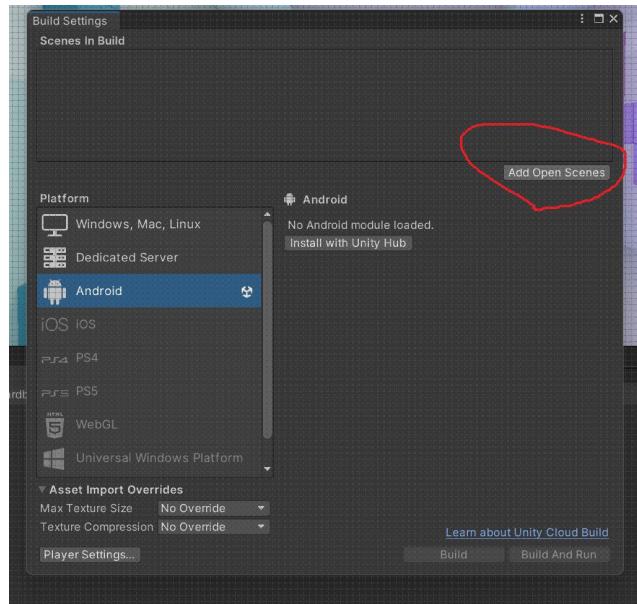
切换目标平台至 Android，若显示灰色，则点击右侧 Install with Unity Hub

(安装完成之后记得重启，不然之后的一些选项将无法看到)

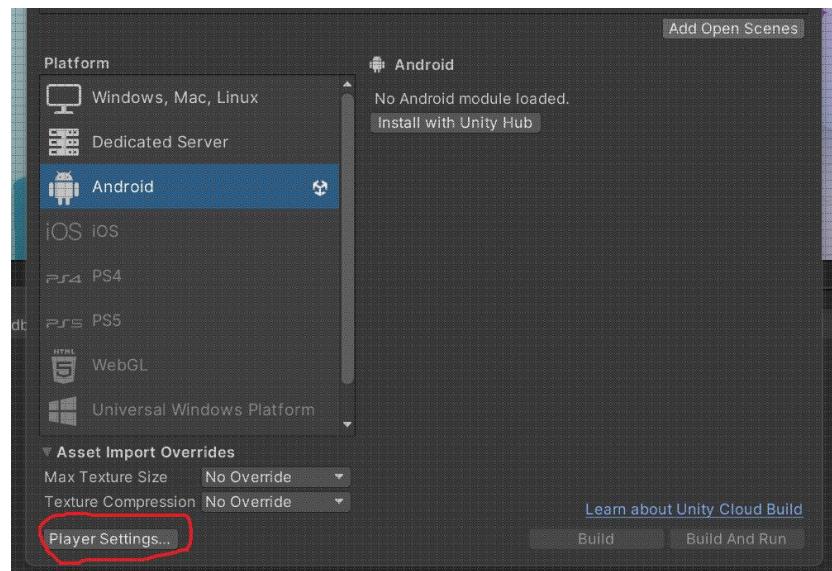
若已安装完成 Android 平台相关环境，则直接选中右下角的 Switch Platform



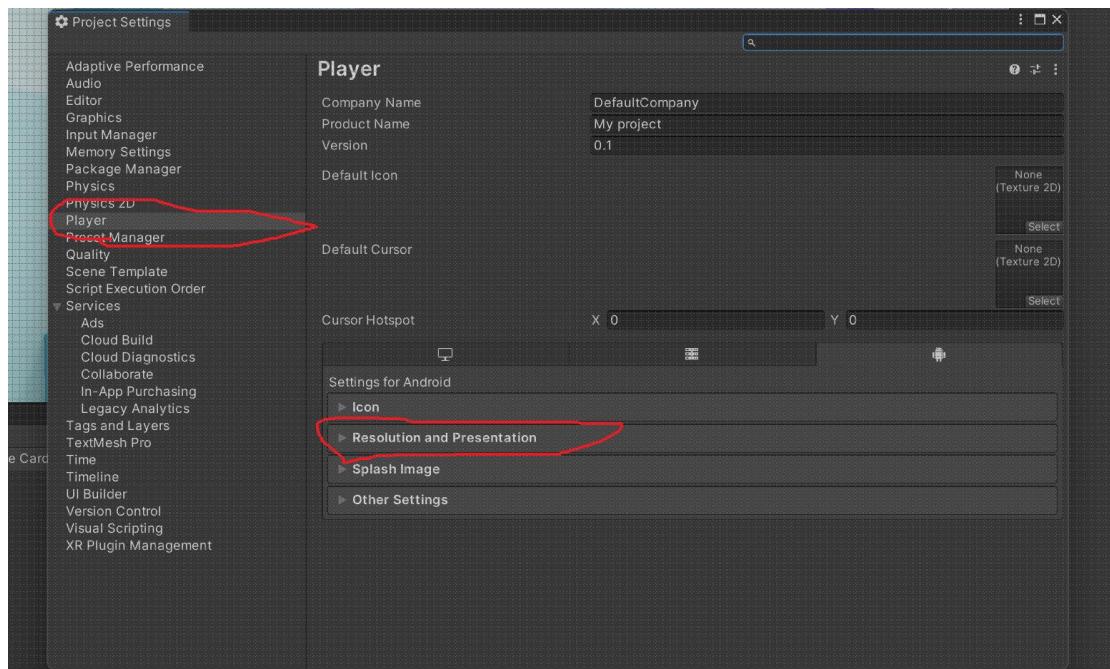
切换平台完成后，点击红圈中的 Add Open Scenes,然后选择 HelloCardboard



选择左下角的 Player Settings

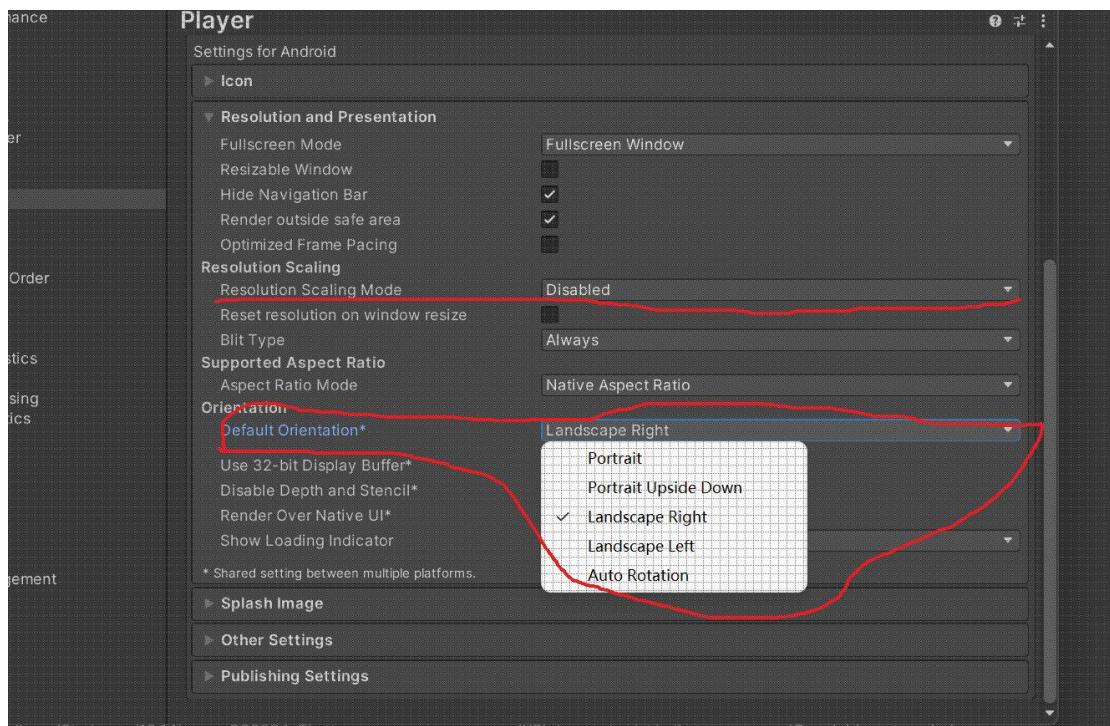


点击之后的界面如下所示，选择 Player-> Resolution and Presentation (如红圈所示)

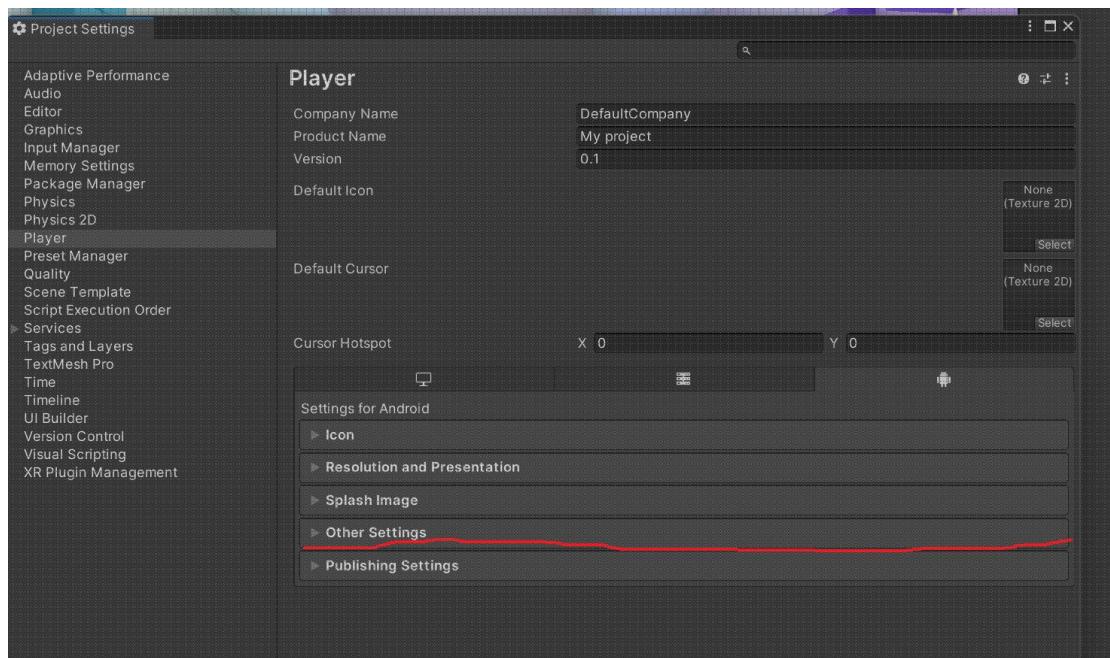


设置 **Default Orientation** 为 **Landscape Left** 或者 **Landscape Right**.

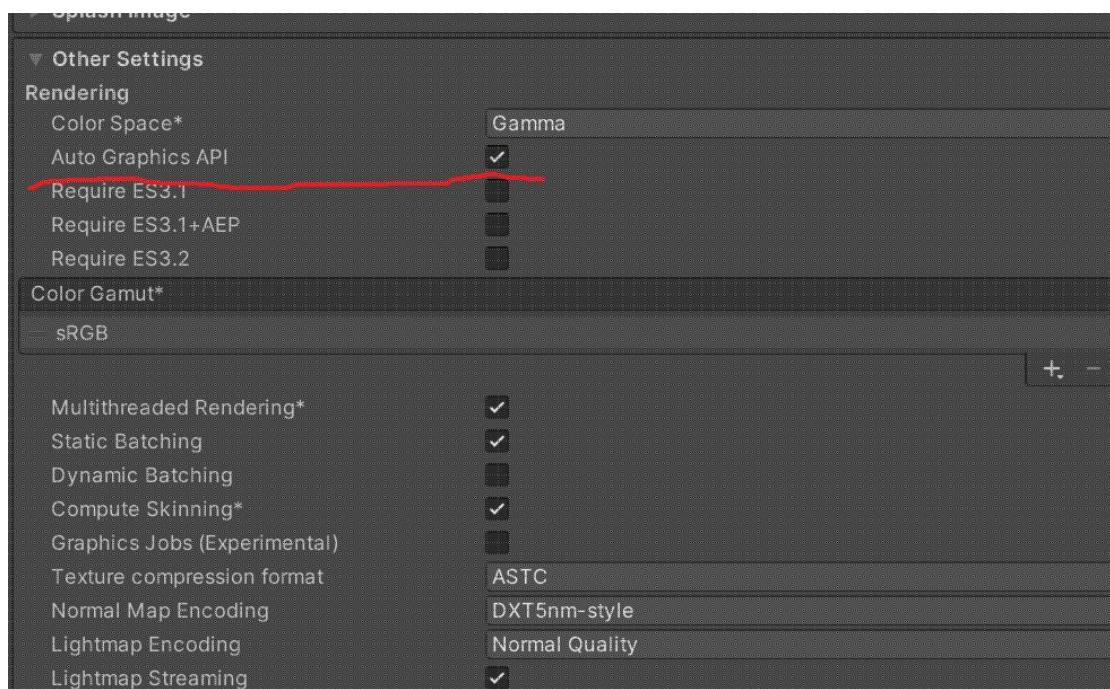
取消勾选 **Optimized Frame Pacing** (此处红线画错, 应为上面一个项目).



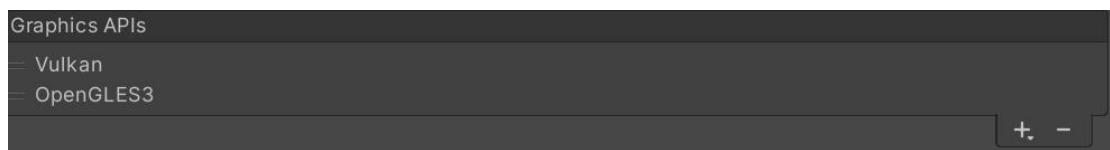
切换至 **Other Settings**



取消勾选 Auto Graphics API

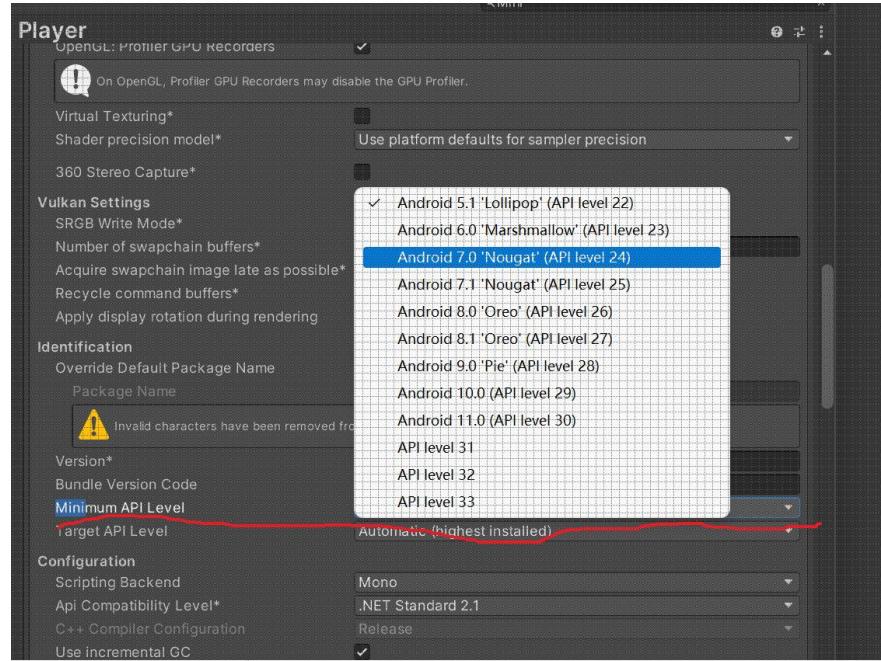


选择 OpenGL ES3。例如下图所示 (注意强烈建议去掉 Vulkan，部分机型可能会报错!)：

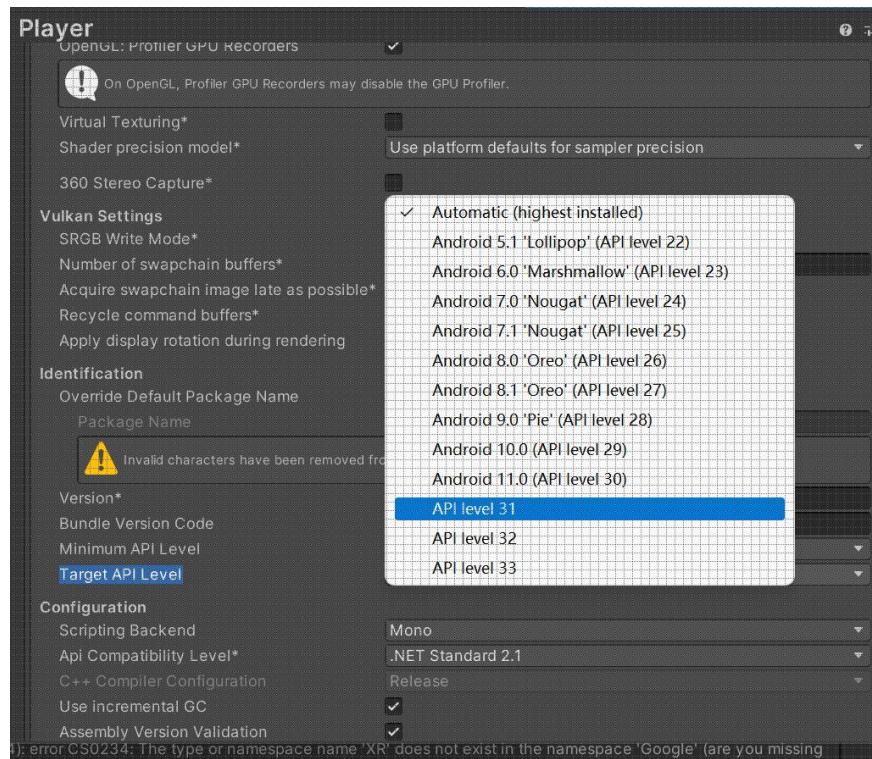


选择 Minimum API Level，并选择 Android 7.0 'Nougat' (API level 24)

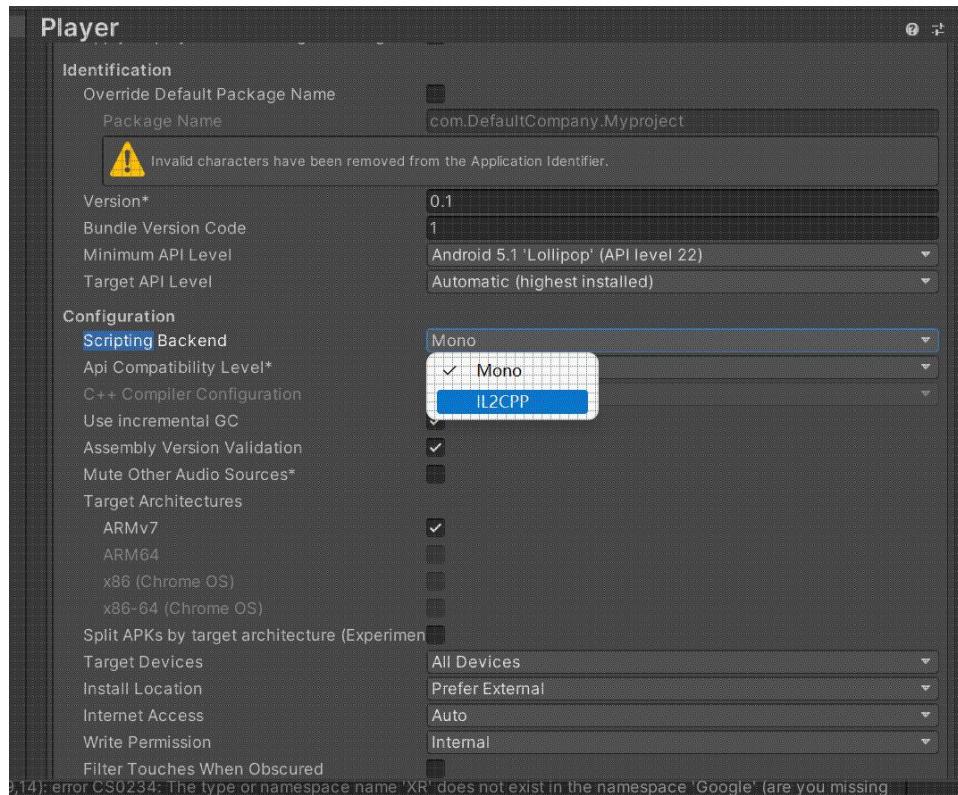
(PS: Minimum API Level 位于 identification 小栏中)



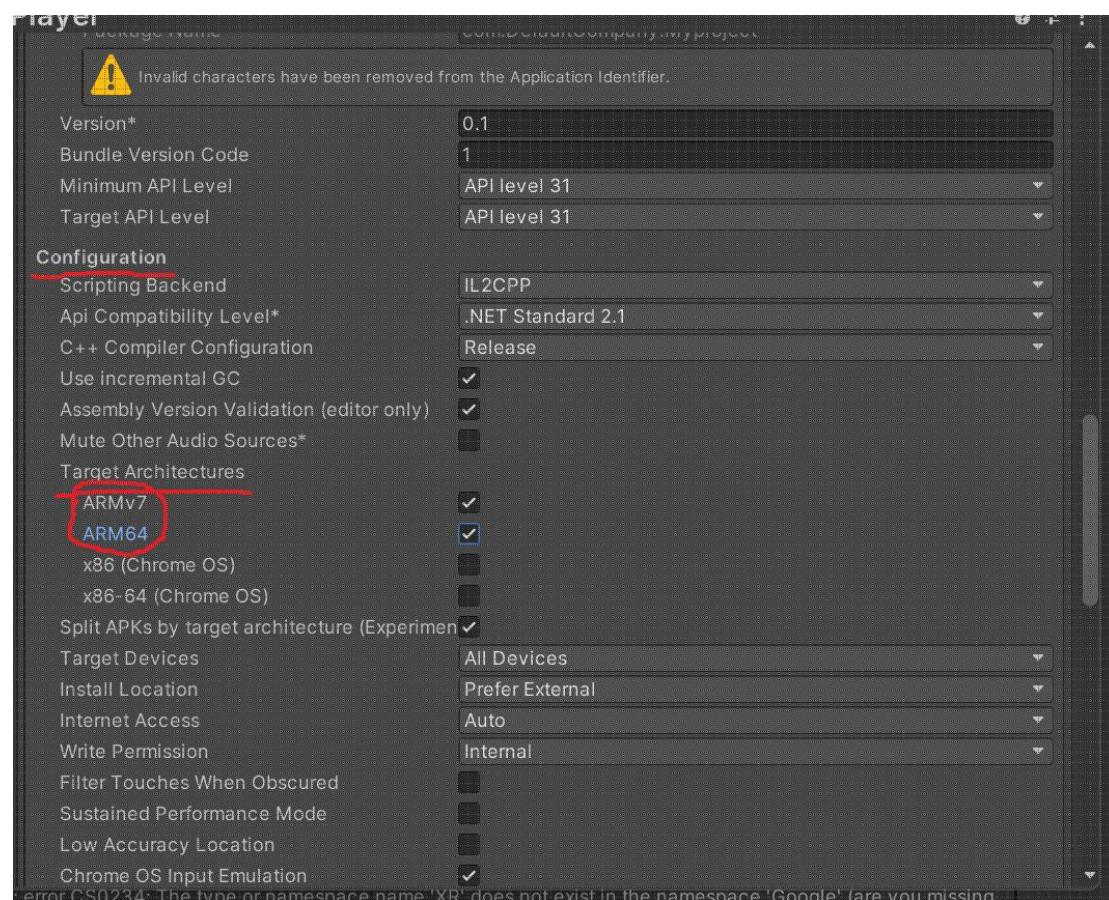
选中 Target API Level，选择 API level 31 或者更高版本



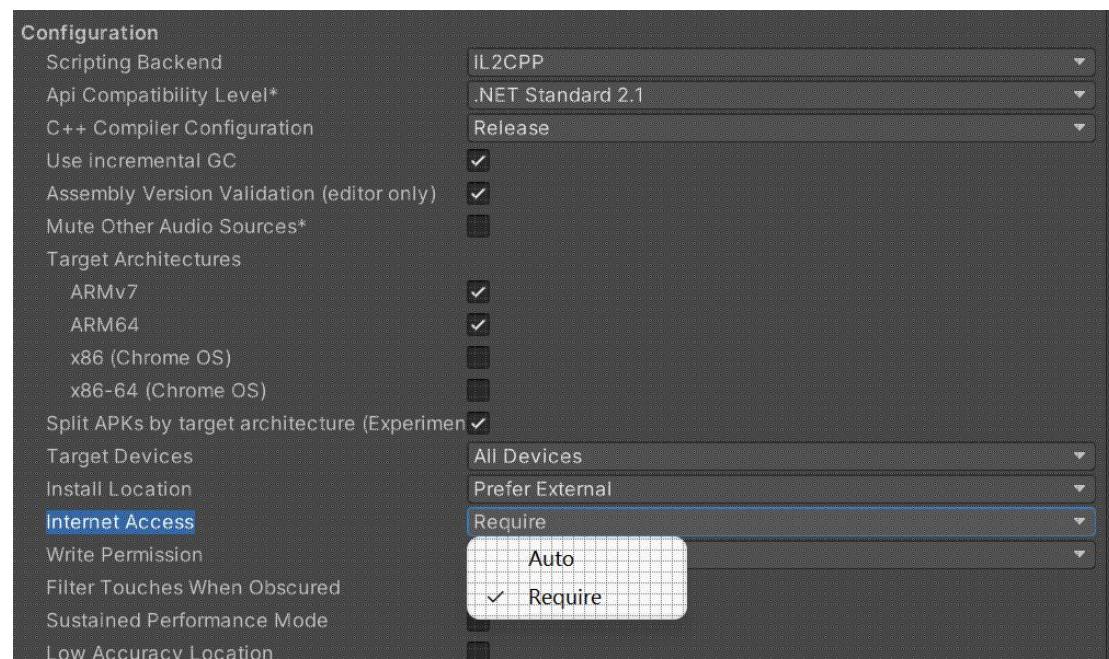
选中 Scripting Backend，选择 IL2CPP



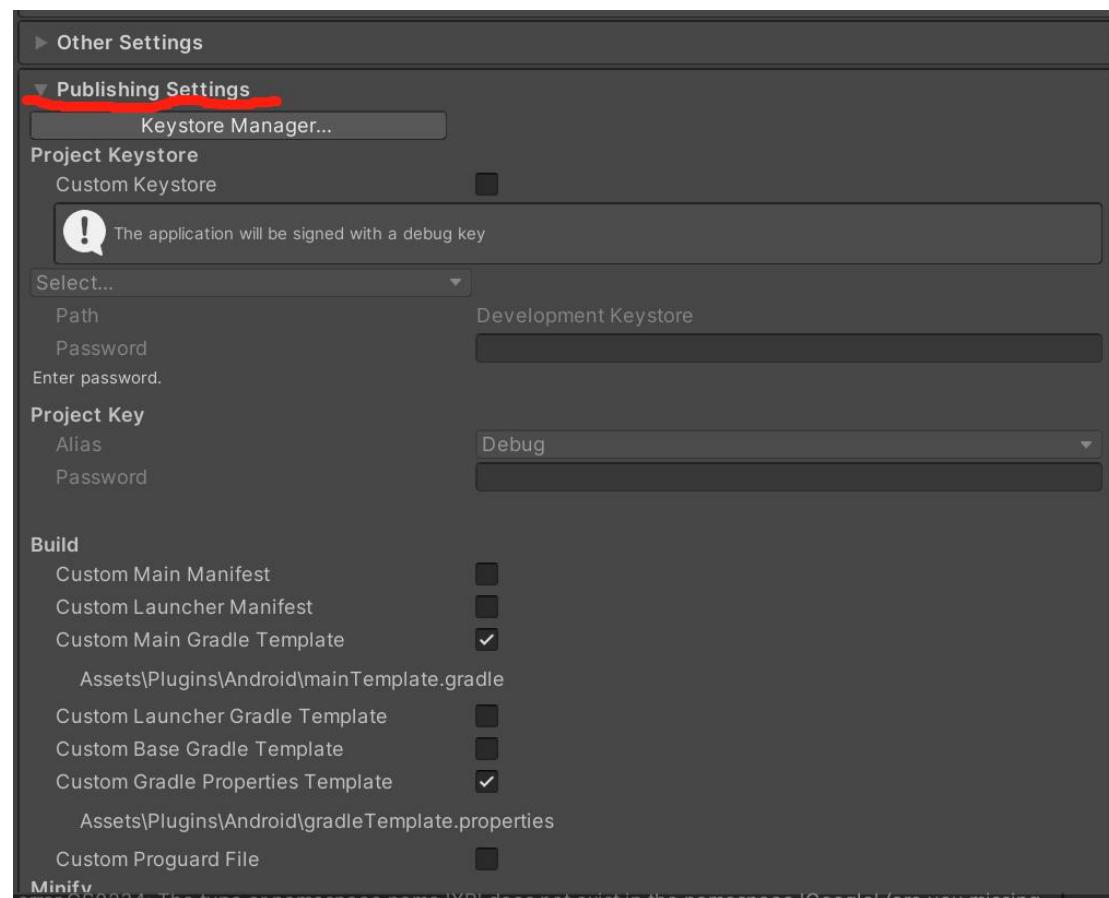
在 Target Architectures 中选择 ARM64 或者 ARMv7，或者干脆都选上



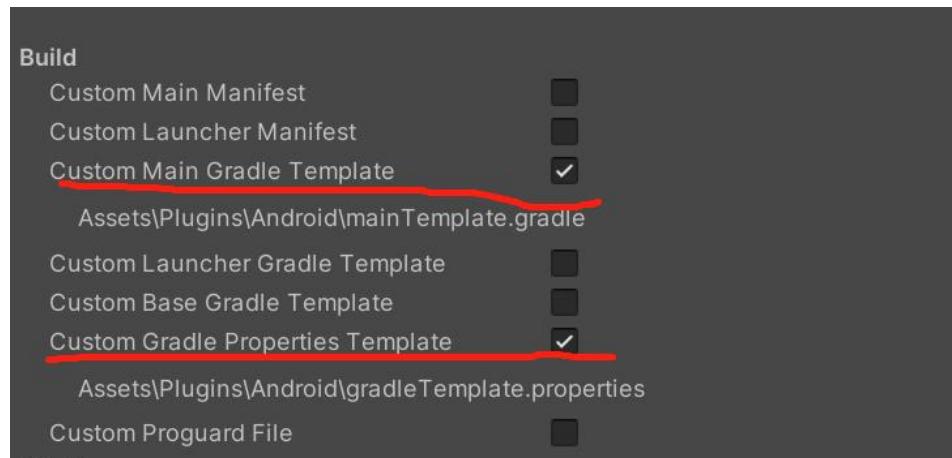
在 Internet Access 中选择 Require



选中 Publishing Settings



在 Build 设置中选择 Custom Main Gradle Template 和 Custom Gradle Properties Template:



配置 Assets/Plugins/Android/mainTemplate.gradle 文件，找到 dependencies:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.4.2'
    implementation 'com.google.android.gms:play-services-vision:20.1.3'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'com.google.protobuf:protobuf-javalite:3.19.4'
    **DEPS**
}
```

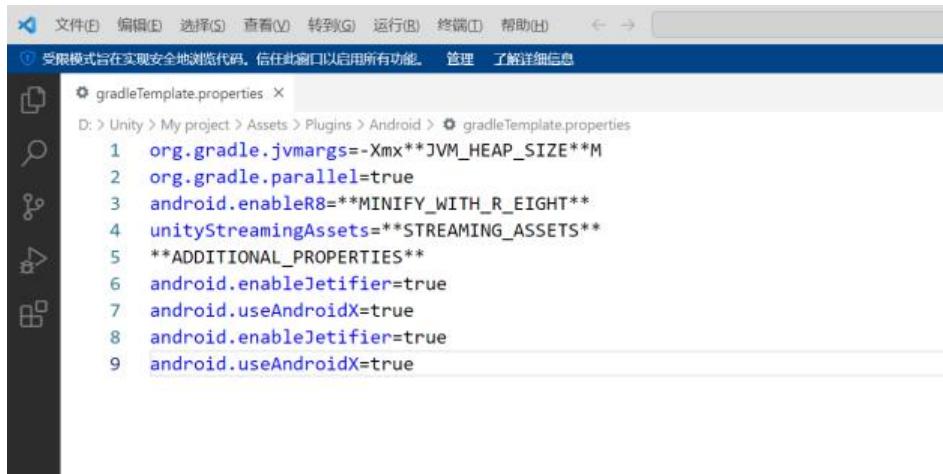
如图所示，添加以下内容:

```
implementation 'androidx.appcompat:appcompat:1.4.2'
implementation 'com.google.android.gms:play-services-vision:20.1.3'
implementation 'com.google.android.material:material:1.6.1'
implementation 'com.google.protobuf:protobuf-javalite:3.19.4'
```

```
1  dependencies {
2      implementation fileTree(dir: 'libs', include: ['*.jar'])
3      implementation 'androidx.appcompat:appcompat:1.4.2'
4      implementation 'com.google.android.gms:play-services-vision:20.1.3'
5      implementation 'com.google.android.material:material:1.6.1'
6      implementation 'com.google.protobuf:protobuf-javalite:3.19.4'
7      implementation 'androidx.appcompat:appcompat:1.4.2'
8      implementation 'com.google.android.gms:play-services-vision:20.1.3'
9      implementation 'com.google.android.material:material:1.6.1'
10     implementation 'com.google.protobuf:protobuf-javalite:3.19.4'
11     **DEPS**
12 }
```

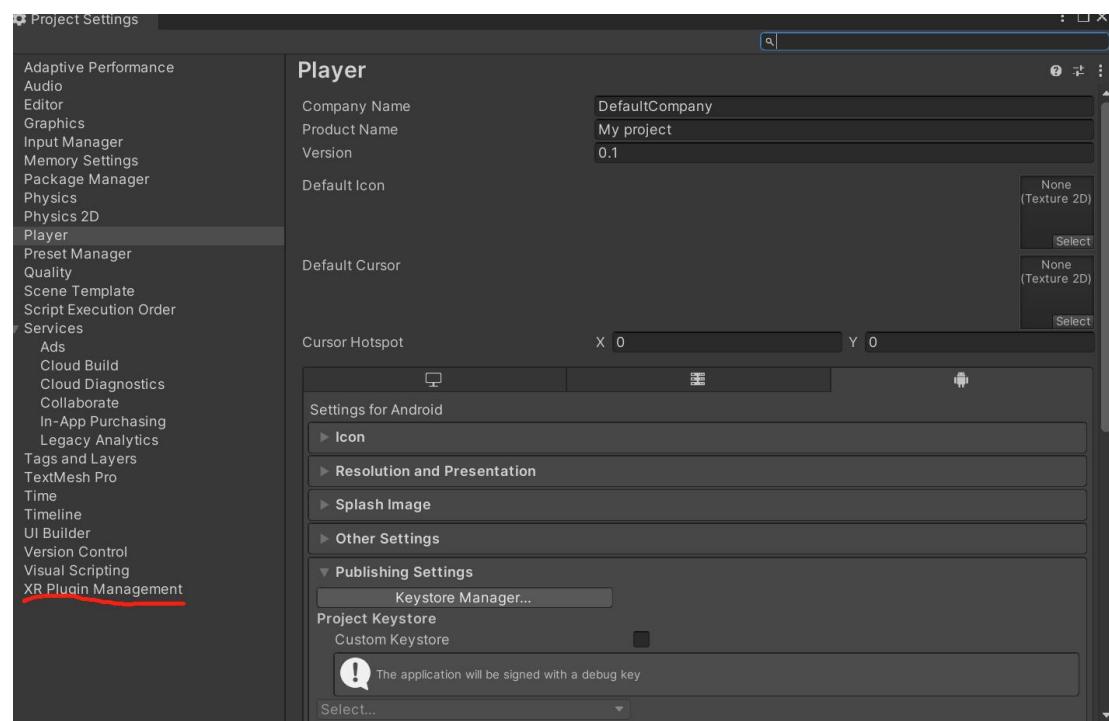
添加以下代码到 Assets/Plugins/Android/gradleTemplate.properties 中去（图示重复添加了）：

```
android.enableJetifier=true  
android.useAndroidX=true
```



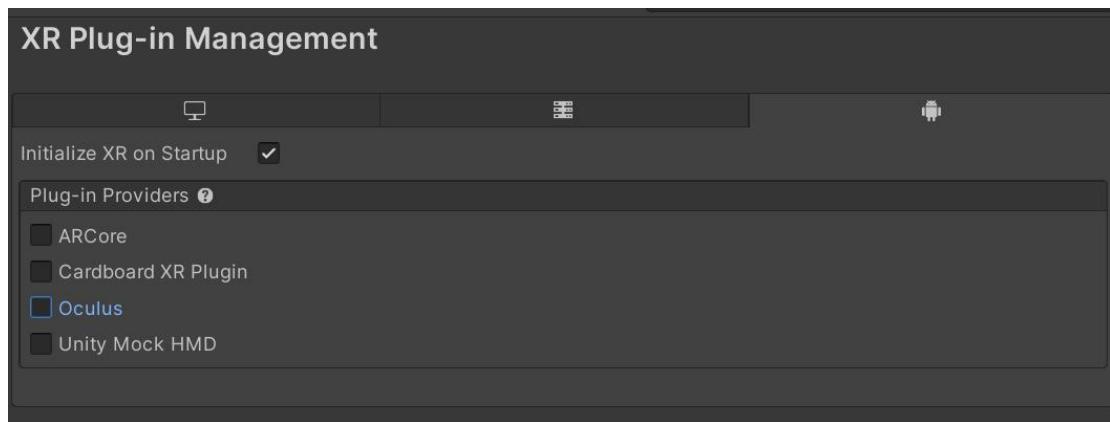
```
gradleTemplate.properties X  
D: > Unity > My project > Assets > Plugins > Android > gradleTemplate.properties  
1 org.gradle.jvmargs=-Xmx**JVM_HEAP_SIZE**M  
2 org.gradle.parallel=true  
3 android.enableR8=**MINIFY_WITH_R_EIGHT**  
4 unityStreamingAssets=**STREAMING_ASSETS**  
5 **ADDITIONAL_PROPERTIES**  
6 android.enableJetifier=true  
7 android.useAndroidX=true  
8 android.enableJetifier=true  
9 android.useAndroidX=true
```

接下来配置 XR 插件，依次进入 Project Settings > XR Plug-in Management



勾选 Cardboard XR Plugin

如有其它选项被选中，则全部取消勾选，否则可能会出现 Bug

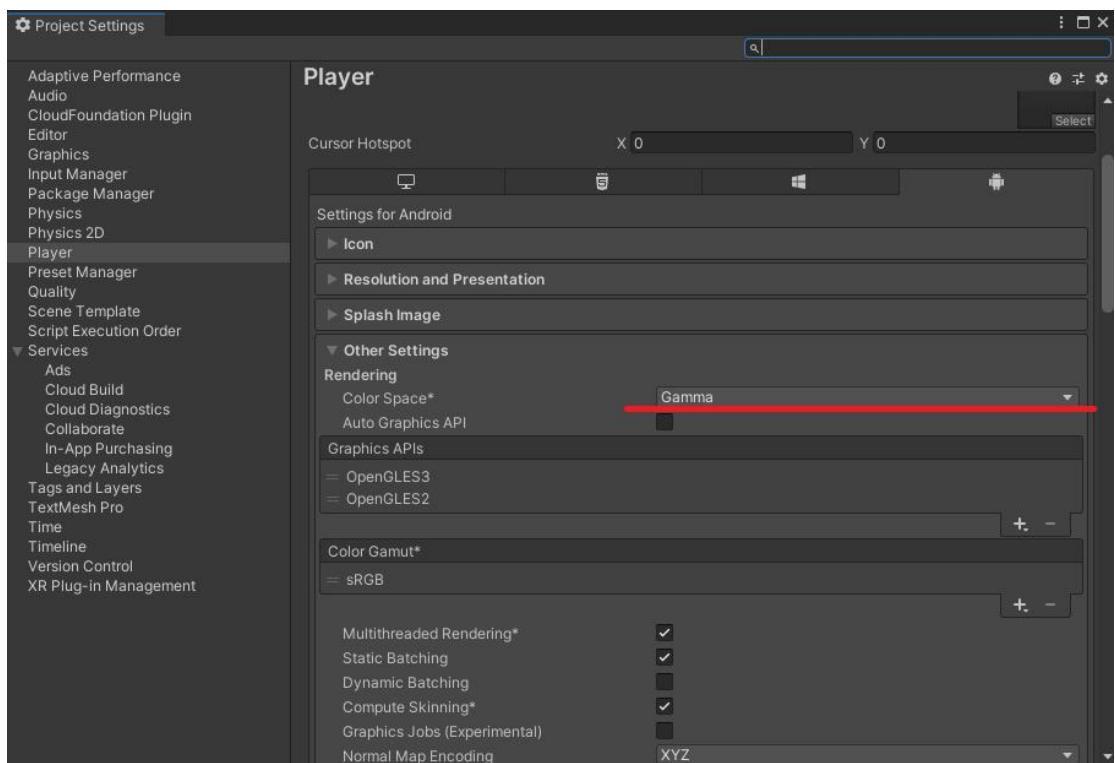


设置完毕后，可在 Unity 窗口内点击 Play 按钮试运行



如果底部调试信息出现报错信息：“**Please initialize Cardboard XR loader before calling this function.**”，可以至项目资源 Assets -> Samples -> Google Cardboard XR Plugin for Unity -> 1.19.0 -> Hello Cardboard -> Scripts 中找到 CardboardStartup.cs，并在 70 行前后找到如下代码：Api.UpdateScreenParams() 在该代码前后添加下面标红的两行代码：

```
#if !UNITY_EDITOR  
    Api.UpdateScreenParams();  
#endif
```



如果在 Build and Run 菜单中显示 Color Key 不兼容相关报错信息，可以前往 Player -> Other Settings -> Color Space 中将 Color Space 调整为 Gamma 即可解决问题。

1.1.4 在 Android 手机上运行 Cardboard

在安卓手机上依次进入设置->关于手机->多次点按手机版本号

启用开发者选项（根据不同机型该步骤有可能有所不同，请以具体情况为准）

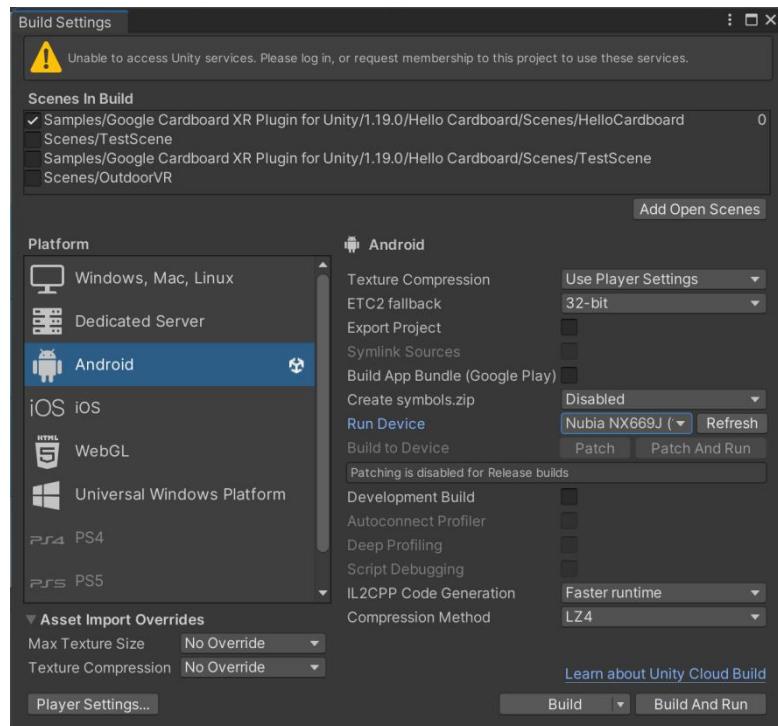


进入开发者选项，开启 USB 调试

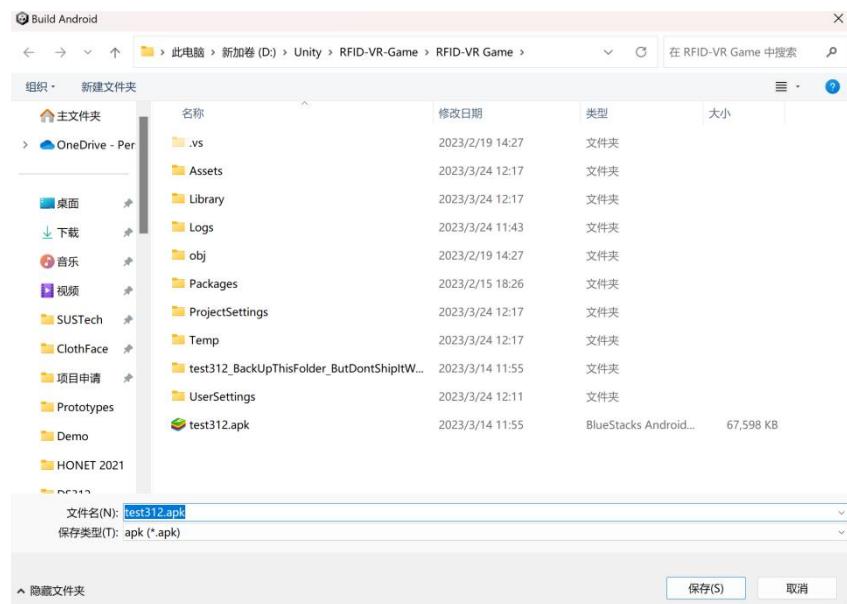


将手机用 USB 连接至运行 Unity 的电脑上

若 USB 调试成功启用，在 Unity 中打开 File->Build Settings.. 菜单，在 Run Device 一栏点击 Refresh，应该可以从下拉菜单中看见自己的 Android 手机型号，选择该手机，点击 Build and Run



命名 APK 安装包，点击保存



第一次编译运行时，有部分 Unity Editor 可能会弹出提示是否进行 Android SDK

Update 的会话窗，**选择 Update Android SDK**

通常第一次编译运行将花费较长时间，期间不要断开 USB 连接，若成功编译运行，将可以在安卓手机上自动打开 HelloCardboard 应用（**请注意部分型号安卓手机在最后阶段会弹出询问是否允许，在规定倒数时间内点击确定后才能正常安装打开**）



成功运行手机界面如下：



1.2 在 Unity Cardboard 中使用 RFID

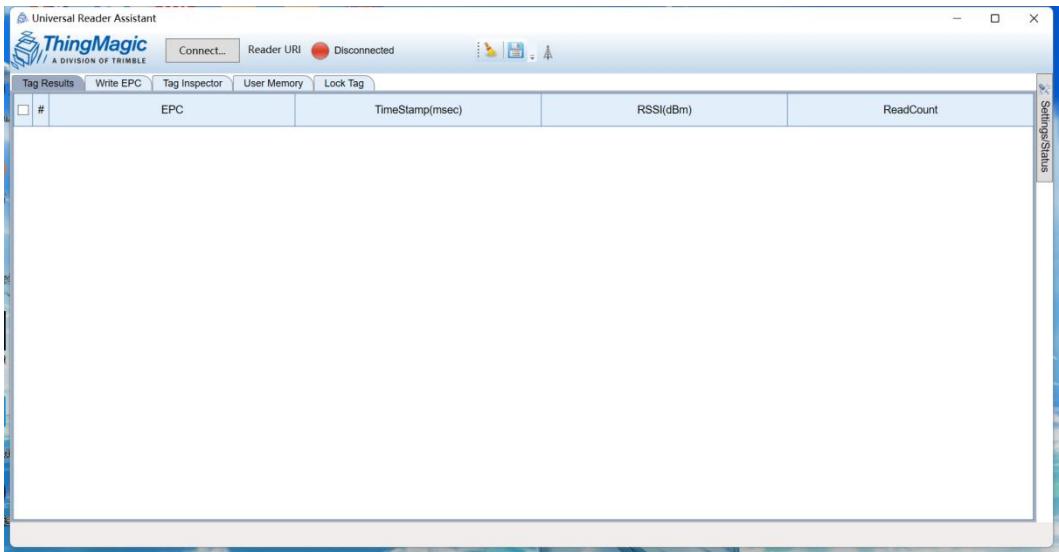
1.2.1 连接 RFID 读卡器与 Windows 电脑

使用 MiniUSB 线缆连接 RFID 读卡器与电脑，第一次连接时，需要等待一段时间自动下载读卡器硬件驱动。



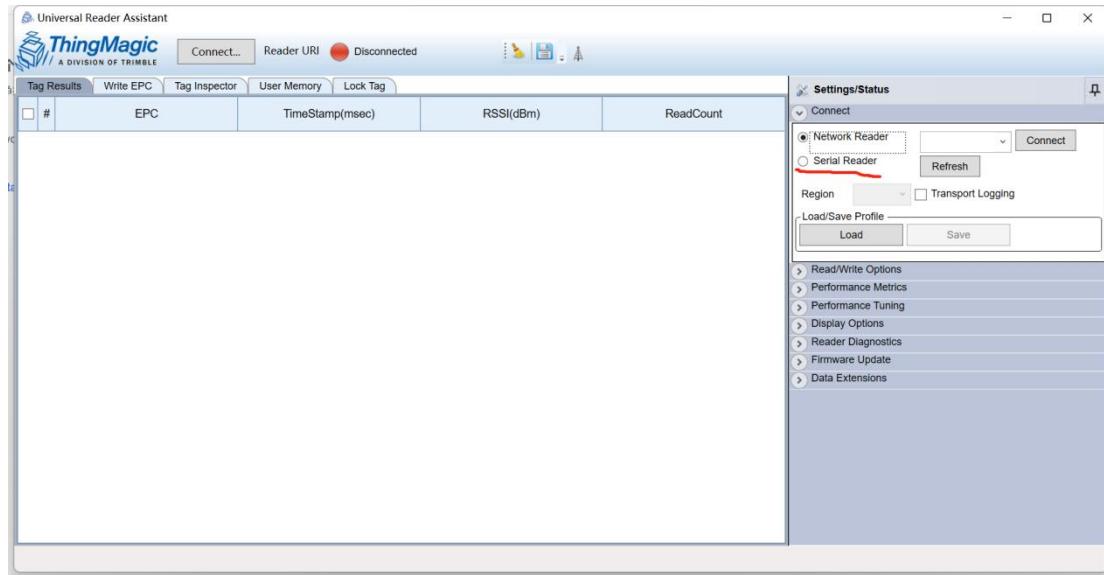
安装 ThingMagic Universal Reader Assistant。在 Documents that may be used.zip 中选择合适的版本，如果你的 windows 是 64 位的就选择 ura2.6bonjoursetup64.zip，32 位选择 ura2.6bonjoursetup32.zip

解压后运行安装，打开安装完成的 Universal Reader Assistant，界面如下所示



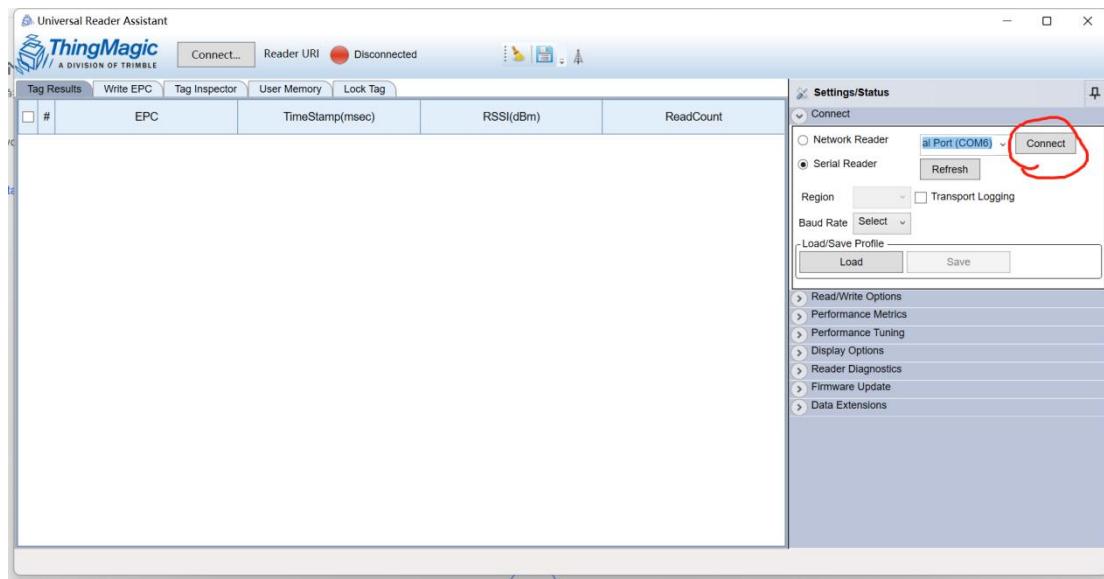
第一次使用 USB 连接 RFID 读卡器，可能需要重启电脑以启用默认安装的驱动程序。
请注意系统弹出的提示。

在 RFID 读卡器正常连接并可被系统检测到的情况下，选择 Serial Reader，点击 Refresh 按钮以后，下拉菜单里应该会出现目前 RFID 所连接的端口号，如 COM3, COM6

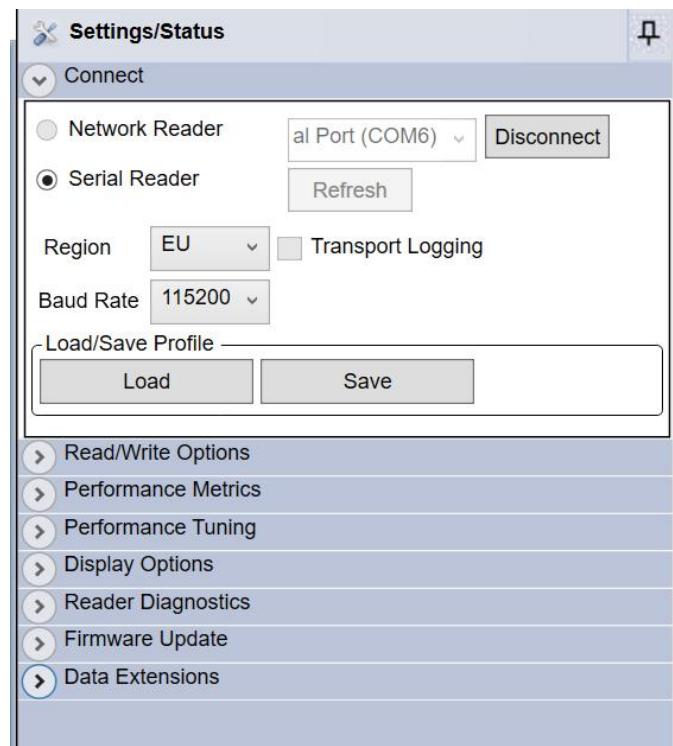


记下该端口号，每台电脑都不一样，比如 COM6, COM3 等。后面需要使用这个根据不同的端口修改后续代码。注意！如果中途将 RFID 连接到另外的 USB 接口，该端口号有可能会变化。

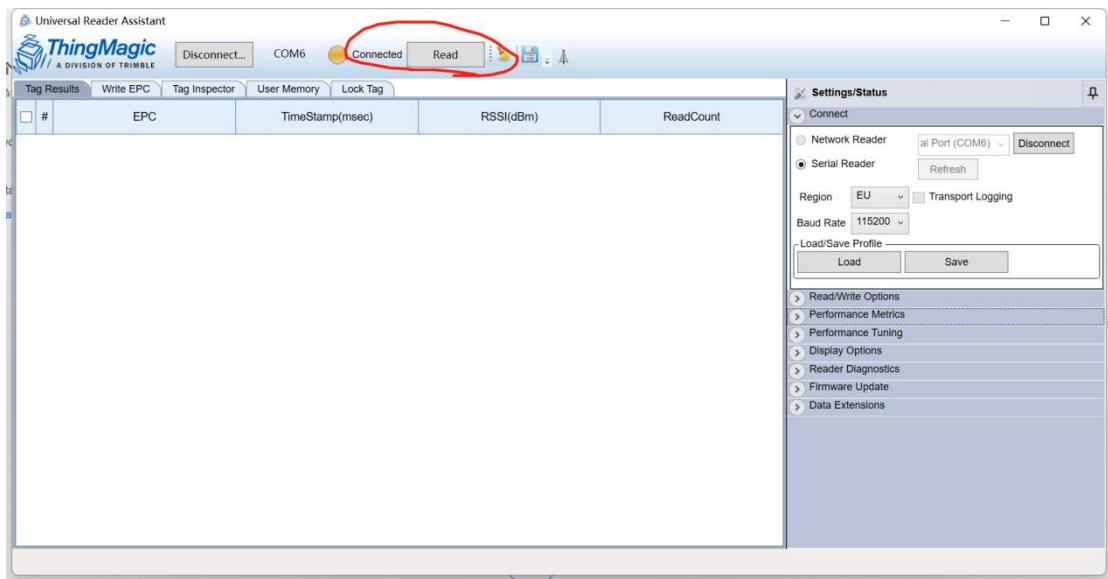
选择该端口号，点击 Connect



连接完成后，Region 地区位选择 EU，Baud Rate 波特率选择 115200，界面应如下图所示：



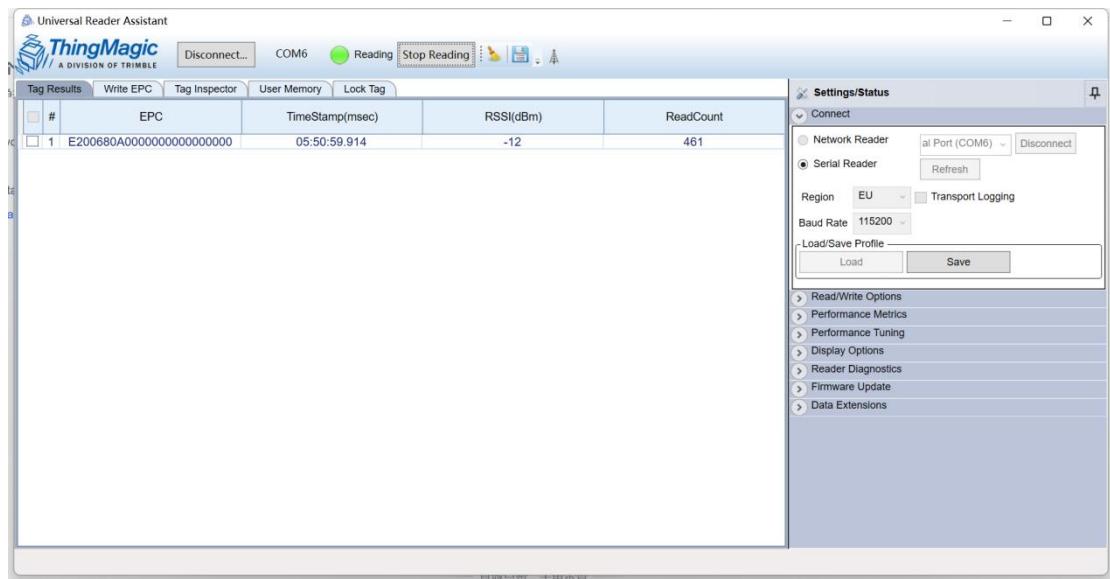
在 Connected 状态下，点击窗口上方 Read 按钮，此时读卡器开始检测 RFID 标签：



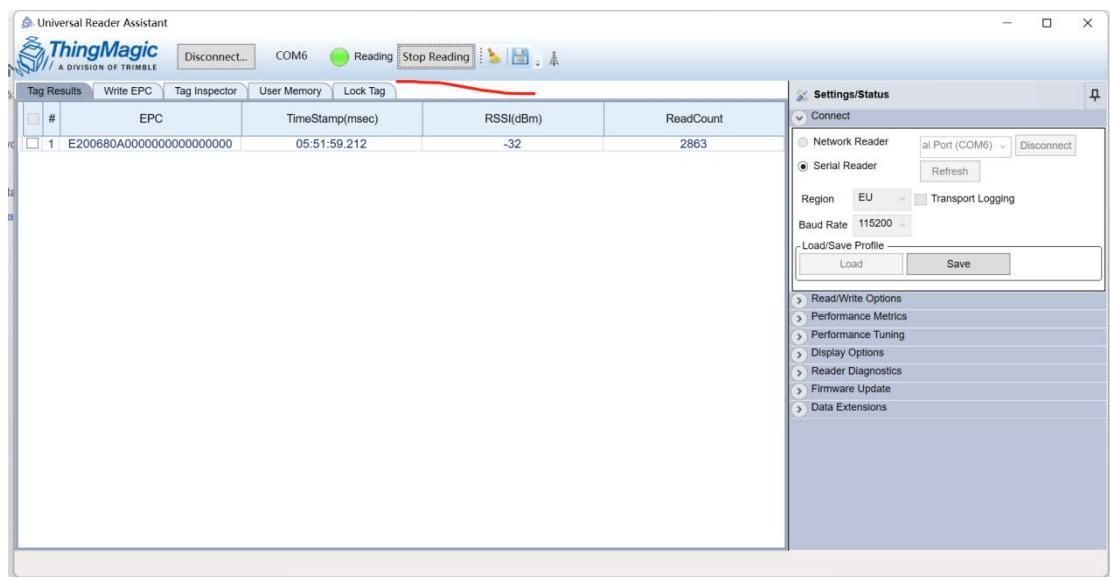
取一张 RFID 标签，靠近 RFID 读卡器：



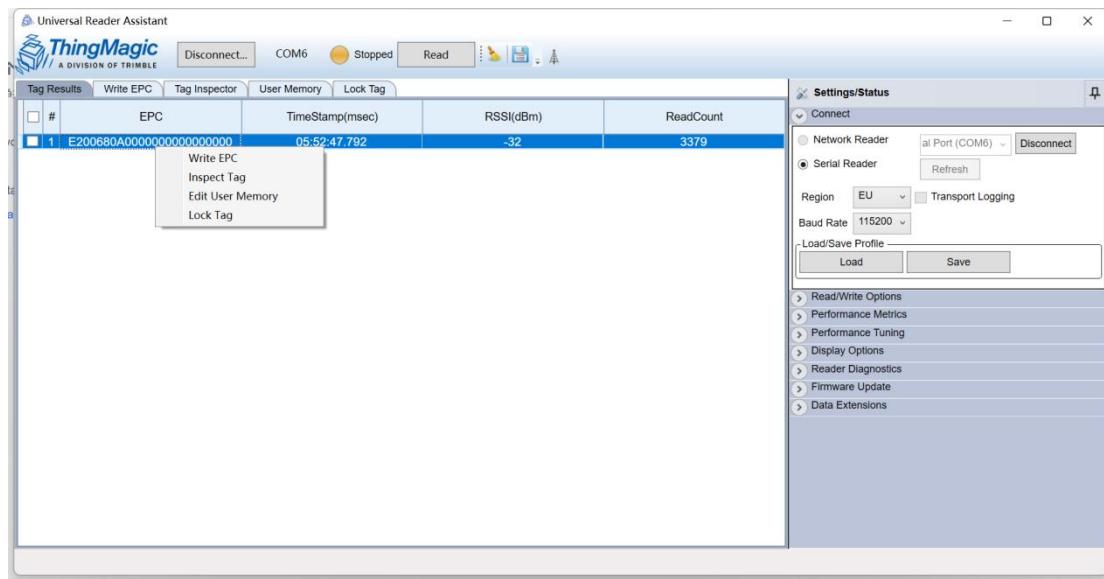
如果 RFID 标签能够被正常读取，结果应该类似下图：



一般情况下，RFID 标签自带的 EPC（类似每个标签的 ID）过长，难以记忆与使用，因此我们需要重新为用到的标签写入新的 EPC（类似于标签重命名）。由于读卡器不能同时读写，因此要先点击 Stop Reading

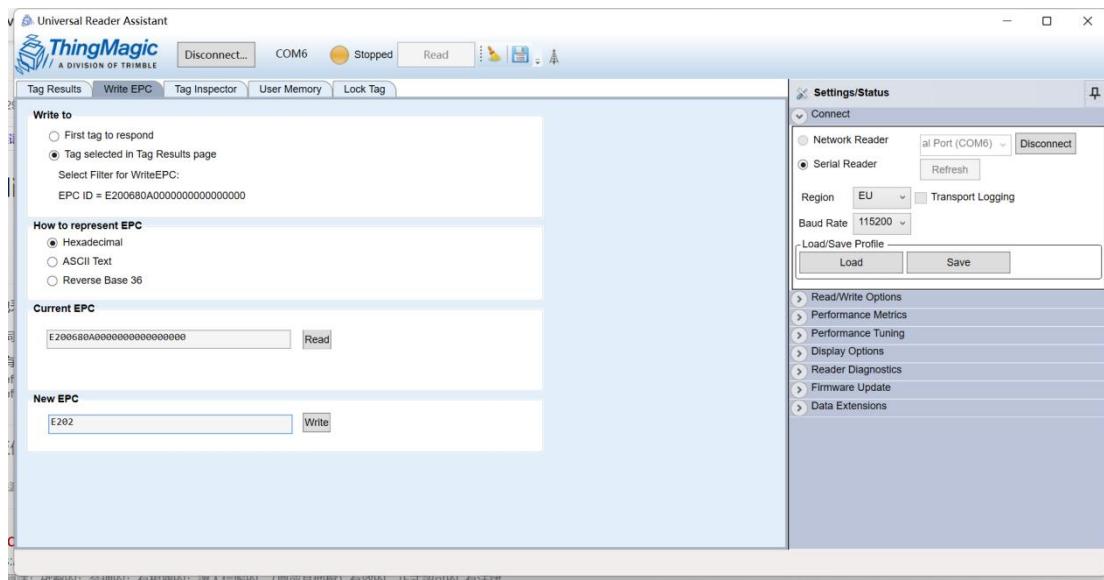


右键点击需要改名的 RFID 标签，选择 Write EPC

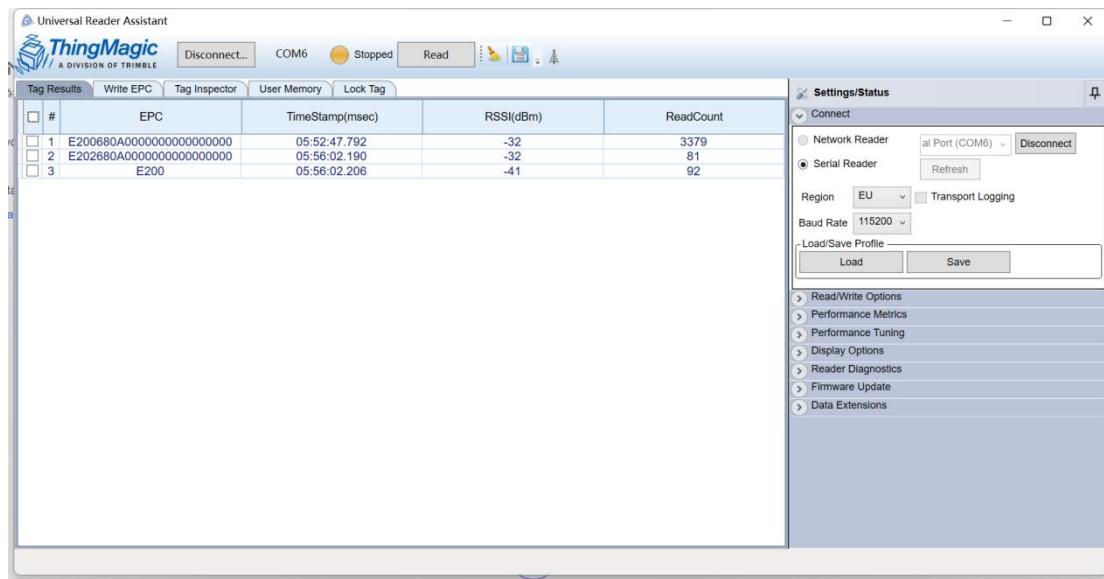


EPC 表达选择 Hexadecimal 16 进制，在 New EPC 的输入栏输入新的 4 位 EPC，需要为 16 进制的字母或数字。如不确定，则简单指定 4 位数字即可。

(注意：后续程序判定有效 EPC 位数为 4 位，输入其它长度字符或违法字母会引发报错)



点击 Write，提示成功写入后，再次 Read，结果应类似下图：



1.2.2 在 Windows 电脑上运行 RFID 服务器

作为前置准备，作为服务器的 Windows 电脑必须已安装 Visual Studio 2019

并确认.NET Framework 版本号为 4.8:

文件路径: C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework

如下图所示：

名称	修改日期	类型	大小
.NETCore	2022/8/28 12:51	文件夹	
.NETFramework	2023/2/18 16:56	文件夹	
v3.0	2022/12/29 0:49	文件夹	
v3.5	2022/8/29 16:01	文件夹	

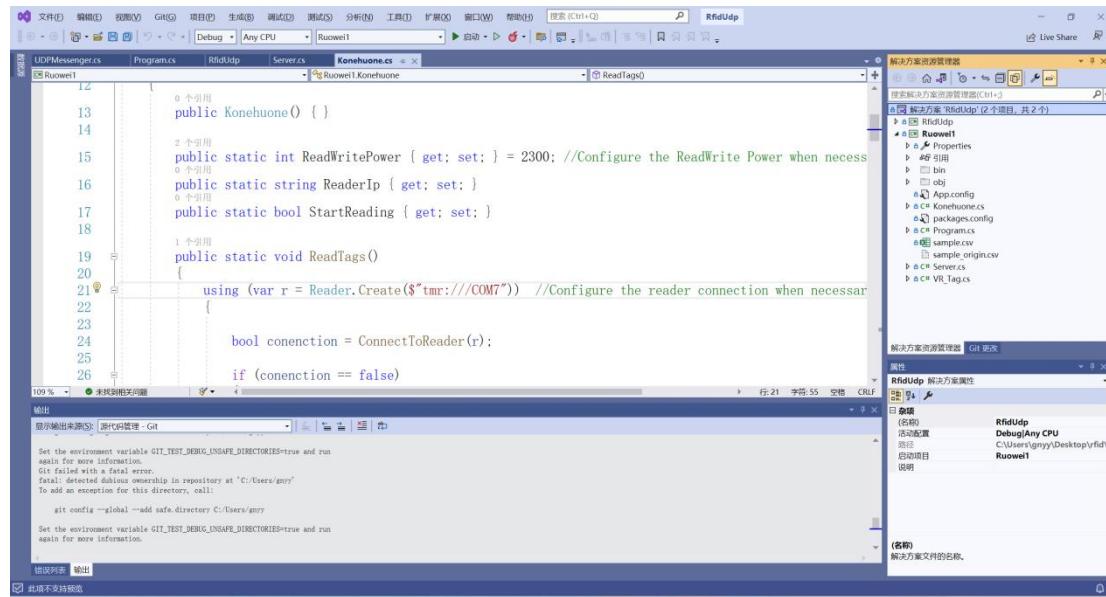
点开.NETFramework，查看是否有 4.8 版本，若没有则将 Documents that may be used.zip 中的.NETFramework 粘贴至 C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework

PS：不用担心覆盖问题

名称	修改日期	类型	大小
v4.7.1	2022/8/28 12:30	文件夹	
v4.8	2023/2/18 17:04	文件夹	

解压 RfidUdp.zip 后，打开目标文件夹下的 RfidUdp.sln 文件，将菜单栏中的启动项目切换为 Ruwei1（初始项目可能为 RfidUdp）

打开该项目中的 Konehuone.cs 文件，如下图所示



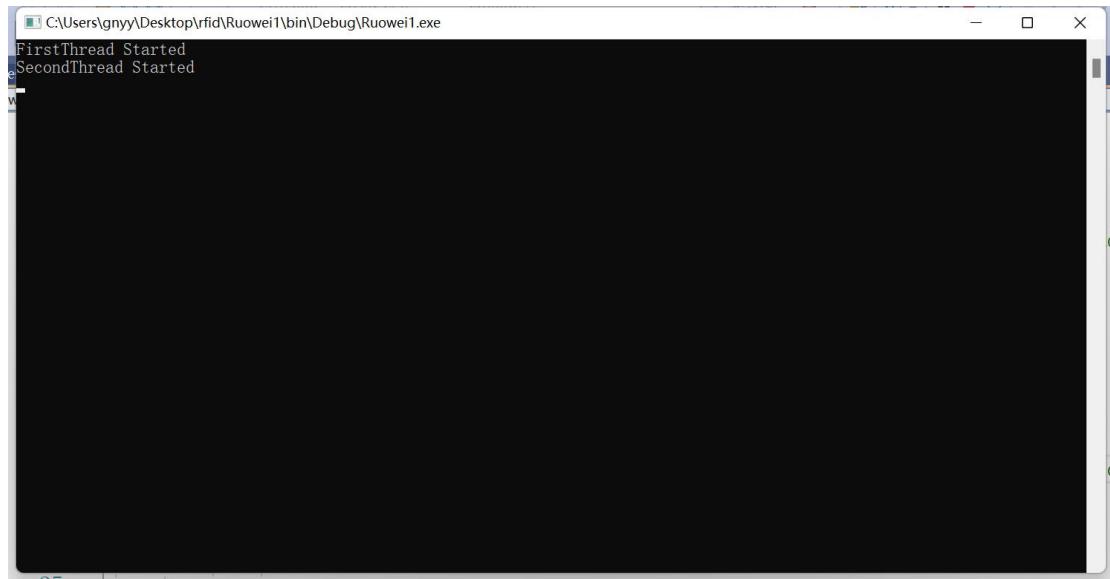
修改第二十一行圆圈所示的地方为 1.2.1 步骤中记录的端口号

```
1 个引用
public static void ReadTags()
{
    using (var r = Reader.Create($"tmr://COM7")) //Configure the reader connection when necessary
    {

        bool conenction = ConnectToReader(r);

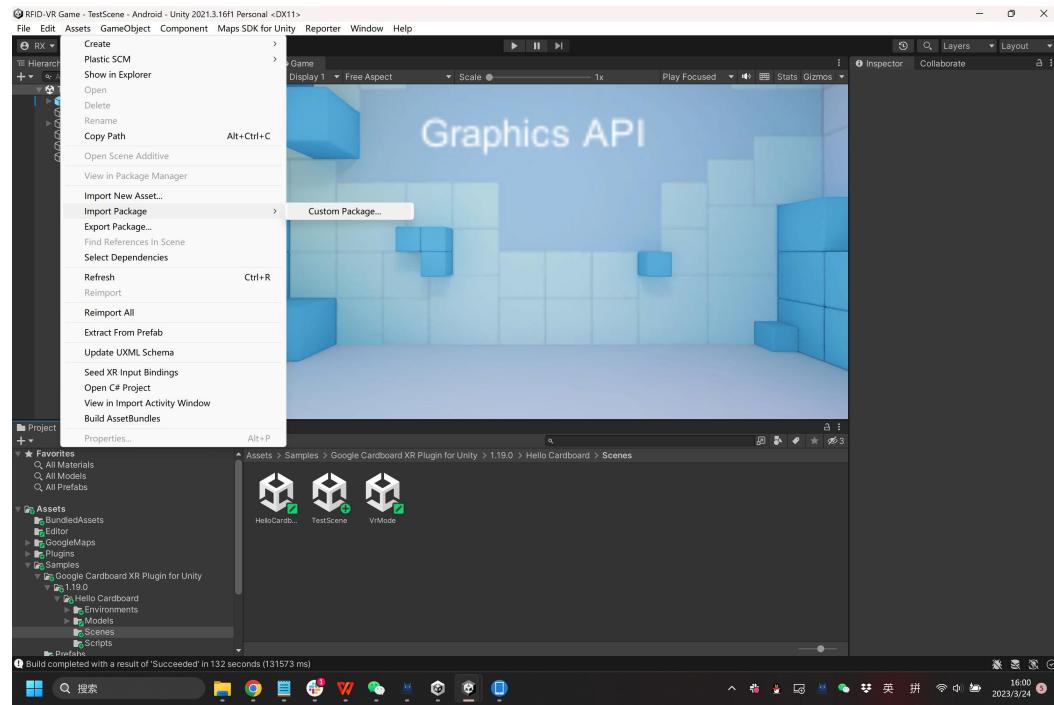
        if (conenction == false)
```

点击旁边的绿色三角形，启动程序，若效果如下图所示表示 RFID 服务器已能够成功启动

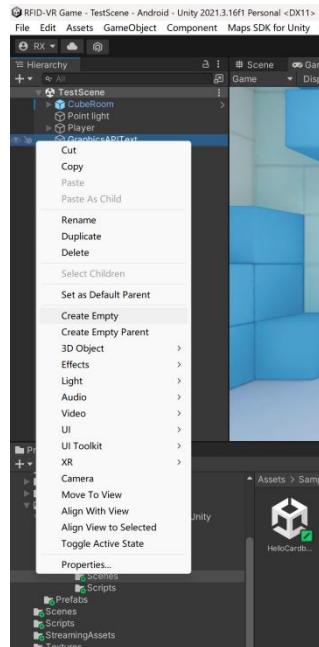
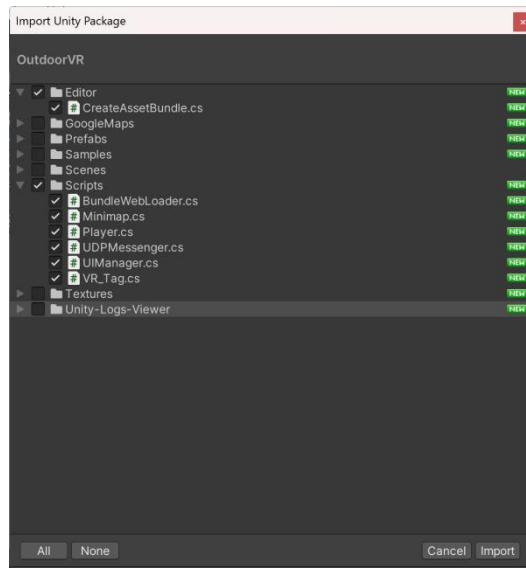


1.2.3 在 Unity 中建立与 RFID 服务器的通讯

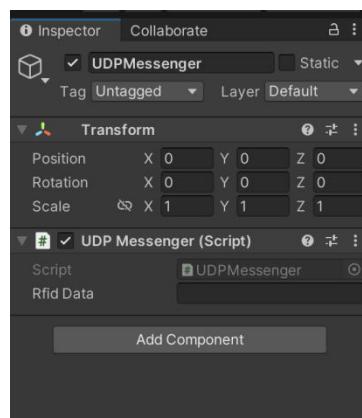
在上一步的 Cardboard Unity 项目中，依次点击 Assets -> Import Package -> Custom Package，选择 OutdoorVR.unitypackage 文件，点击 import 导入



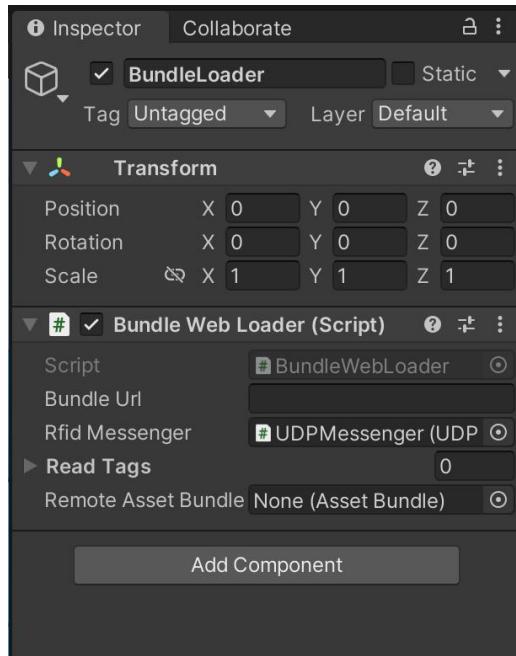
注意在这一步，不需要将该 package 里的文件全部导入，只需要勾选 Editor 以及 Scripts 两个文件夹，将里面包含的脚本导入即可。如下图所示：



在项目 Hierarchy 栏中点击右键，从菜单中选择 Create Empty，创建一个空对象，并分别命名为 UDPMessenger 和 BundleLoader



选中创建的 UDPMessenger 对象，将 Assets -> Scripts 文件夹中的 UDPMessenger.cs 拖拽至 Inspector 其中，成功后应如上图所示



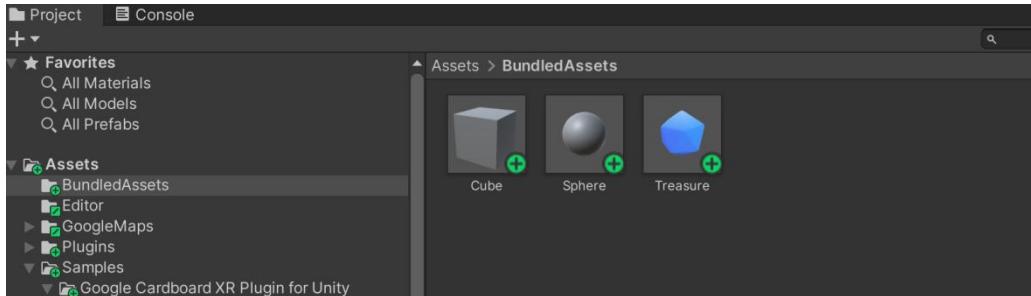
同样的方式将 Assets -> Scripts 文件夹中的 BundleWebLoader.cs 拖拽至 BundleLoader 对象上，并将刚刚创建的 UDPMessenger 对象拖拽到 Bundle Web Loader 中 Rfid Messenger 一栏内。成功后应如上图所示。

1.2.4 将预制的游戏对象打包上载至网络服务器

在 Assets 中新建一个文件夹用以放置用来被打包的游戏对象，可命名为 BundledAssets

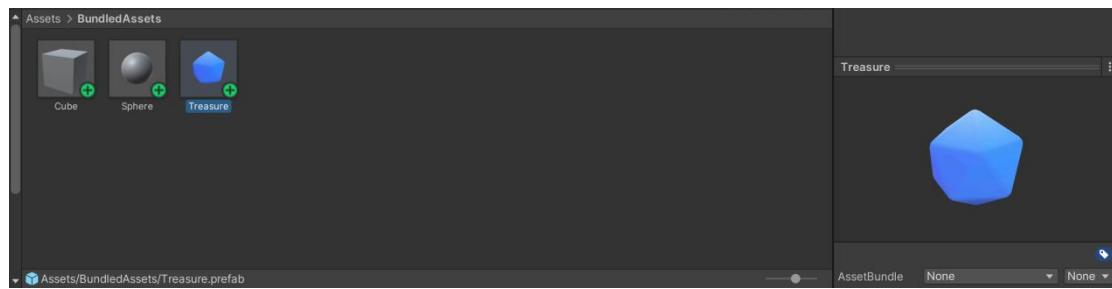
在 HelloCardboard 项目的 Hierarchy 中找到名为 Treasure 的游戏对象，将其拖拽到上面新建的文件夹中

之后，将 Treasure 从 Hierarchy 中删除

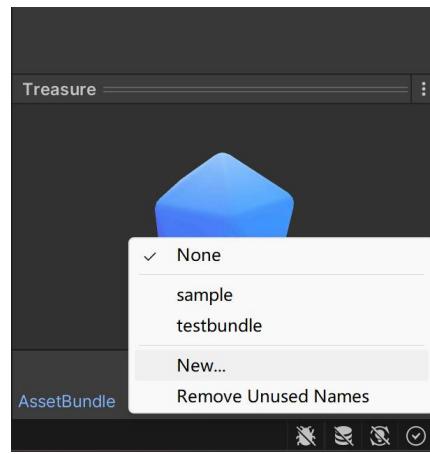


在 BundledAssets 文件夹中选中 Treasure，在 Inspector 可以看见底部应有

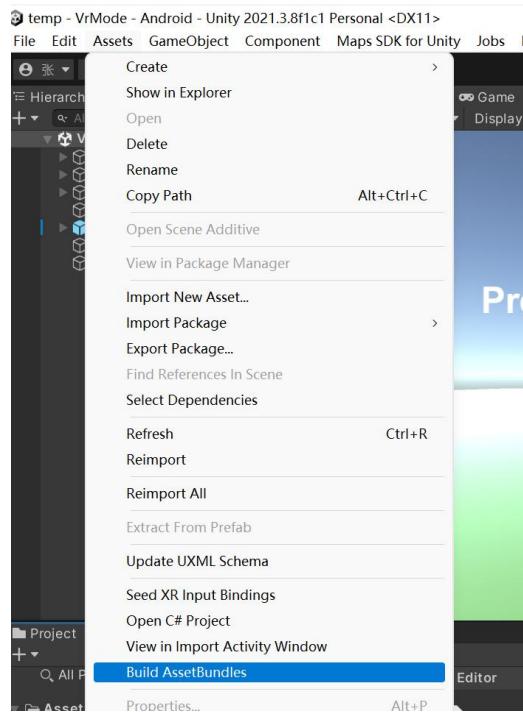
AssetBundle 一栏，初始状态为 None



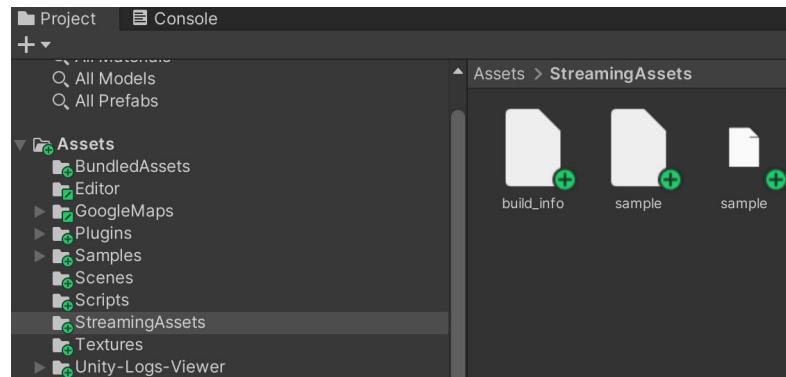
点击 AssetBundle 一栏的下拉箭头，选择 New... 后，输入该捆包的名字，如 sample



捆包命名完后，可打开 Unity 的 Assets 菜单，点击 Build AssetBundles



若成功生成捆包，可在 Assets -> StreamingAssets 中看见刚刚捆包生成的、命名为 sample 的文件



将捆包后的游戏对象文件上传至网络服务器之后，即可通过对应的网络服务器地址下载该文件

1.2.5 使用 CSV 文件关联 RFID 与网络资源

打开 1.2.4 中 RfidUdp 项目的文件夹，在 Ruowei1 子文件夹中找到 sample.csv 文件

名称	修改日期	类型	大小
bin	2022/5/19 6:33	文件夹	
obj	2022/5/19 6:33	文件夹	
Properties	2022/5/19 6:33	文件夹	
App.config	2022/5/19 6:33	XML Configuration ...	1 KB
Konehuone.cs	2023/2/6 0:09	C# Source File	5 KB
packages.config	2022/5/19 6:33	XML Configuration ...	1 KB
Program.cs	2022/5/22 19:05	C# Source File	2 KB
Ruowei1.csproj	2022/5/19 6:33	C# Project file	4 KB
Ruowei1.csproj.user	2022/5/19 6:33	Per-User Project O...	1 KB
<input checked="" type="checkbox"/> sample.csv	2023/3/11 15:48	Microsoft Excel 逗...	1 KB
sample_origin.csv	2022/5/22 19:05	Microsoft Excel 逗...	1 KB
Server.cs	2022/5/22 19:05	C# Source File	5 KB
VR_Tag.cs	2023/2/6 0:28	C# Source File	2 KB

使用记事本打开 sample.csv (注意：最好不要用默认的 Excel 或类似程序打开，否则若 csv 文件内含有以 0 开头的 RFID 标签的 epc，有可能被自动去掉 0 而导致数位变化，引发程序出错)

文件内容格式如下：

The screenshot shows a CSV file titled "sample.csv" in a text editor. The first row contains column headers: "EPC,Lat,Lng,BundleUrl,AssetName". The subsequent five rows provide data for an RFID tag:

	EPC	Lat	Lng	BundleUrl	AssetName
0011	35.657518	139.747775		http://121.37.8.3/Vera/testbundle	GoldBoarPBR Variant
0012	61.496609	23.779807		http://121.37.8.3/Vera/sample	Treasure
0013	,,			http://121.37.8.3/Vera/sample	Cube
E000	,,			http://121.37.8.3/Lance/1111111	Spaceship
0001	,,			http://121.37.8.3/Anna/cuuube	Cube

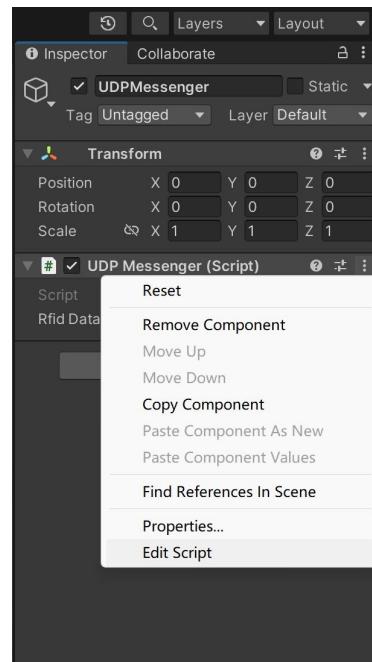
该文件第一行表示了每个数据下包含各字段名：EPC，经度，纬度，游戏资源捆包地址及捆包中特定资源名称

第二行起，每一行数据记录代表一个 RFID 标签所关联的数据

在 RFID 服务器未运行的情况下，将 EPC 修改为 1.2.1 步骤中指定的 RFID 标签 EPC，将 BundleUrl 和 AssetName 分别修改为 1.2.3 步骤中捆包的游戏资源文件地址和捆包中对应的资源名（此处应为 Treasure）

修改完后，保存 csv 文件，再次运行 RFID 服务器程序

在 Unity 端选中 UDPMessenger，在 Inspector 中点击 UDP Messenger 脚本组件最右端的三点，在弹出的下拉菜单中选择 Edit Script



在 VS2019 编辑器中，找到第 44 行代码，并修改该代码中的 IP 地址为目前所使用的 RFID 服务器地址（如不确定服务器地址，可在该服务器电脑上运行命令行，输入 ipconfig 查询到具体的网络 IP 地址）

保存项目后，将项目再次编译加载至手机端

```
34     }
35 
36     void OnApplicationQuit()
37     {
38         thread.Abort();
39     }
40 
41     private async void ThreadMethod()
42     {
43         UdpClient udpSend = new UdpClient();
44         IPEndPoint ep = new IPEndPoint(IPAddress.Parse("10.12.223.190"), 11001); //Fill in here your RFID server address;
45         //IPEndPoint ep = new IPEndPoint(IPAddress.Parse("192.168.2.103"), 11001);
46         string msg = "StartReading";
47         byte[] packet = Encoding.UTF8.GetBytes(msg);
48         await udpSend.SendAsync(packet, packet.Length, ep);
49 
50         //Create a UdpClient object and specify 11000 as receive port
51         UdpClient udpReceive = new UdpClient(11000);
52 
53         //Specify remote host machine, allowing all data sent by any IP port
54         IPEndPoint remoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0); //IPEndPoint remoteIpEndPoint = null;
55 
56         Debug.Log("Waiting for a client...");
57 }
```

先运行 RFID 服务器，再运行手机端 Cardboard VR 应用

如手机端成功与 RFID 服务器建立连接，则服务器端 RFID 控制台程序会显示如下信息：

```
StartReading : true
Connecting...
Reading started!
```

有时第一次运行会显示如下报错信息：“Connection Cancelled”，此时可关闭

Cardboard VR 应用并再次打开，通常可以解决该报错；如未解决，请检查是否打开了

VPN 系统代理，如有，请关闭再尝试。

```
D:\Clothface\CodeRepository' > 
FirstThread Started
SecondThread Started
StartReading : true
Connecting...
Connection Cancelled
```

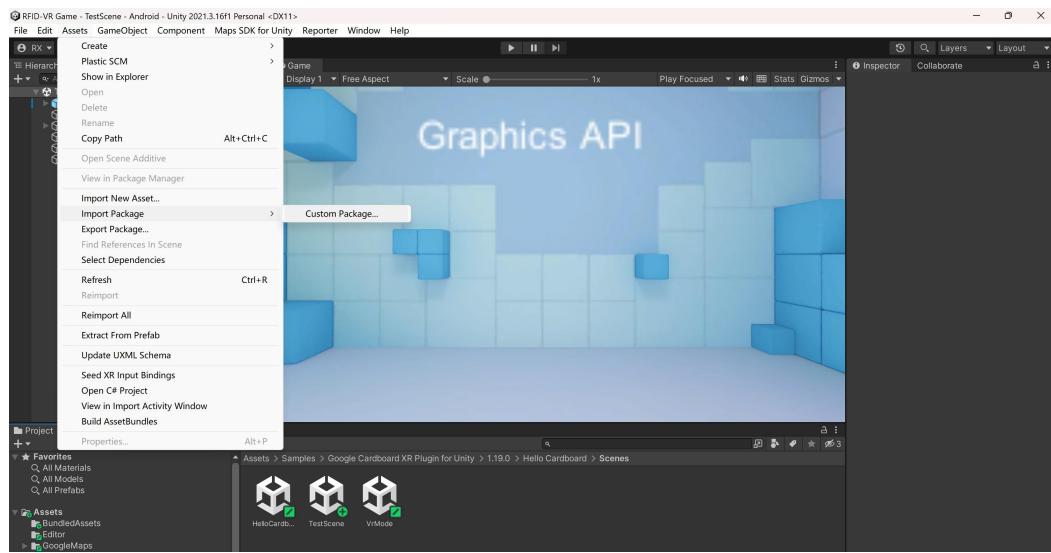
如程序没有错误，则在手机端打开 Cardboard VR 应用以后，读取对应 RFID 标签，可看见 VR 虚拟房间中实时加载了 Treasure 对象，可以原本方式与该对象进行互动。

1.3 搭建户外 VR 环境

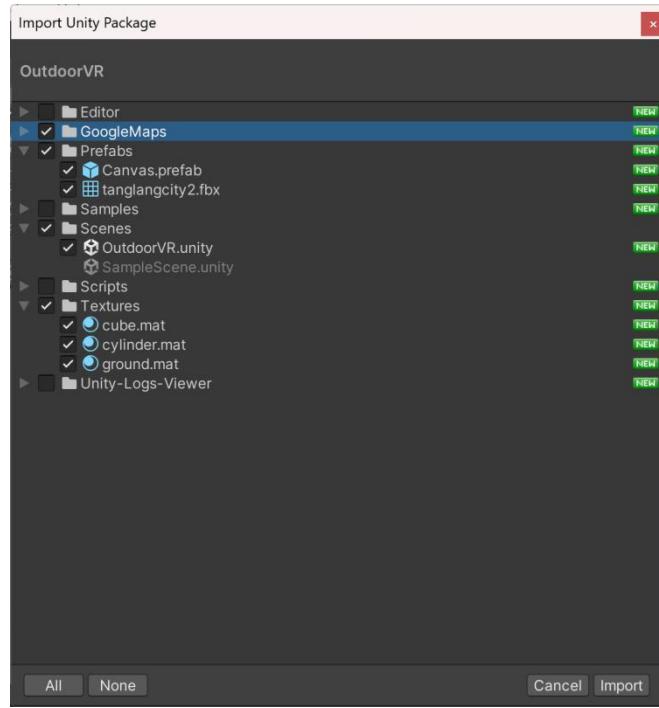
1.3.1 导入户外 VR 环境相关资源

在 Cardboard Unity 项目中，依次点击 Assets -> Import Package -> Custom Package，选择 GoogleMaps.unitypackage, GoolgeMapsExamples.unitypackage，点击 import 全部导入。

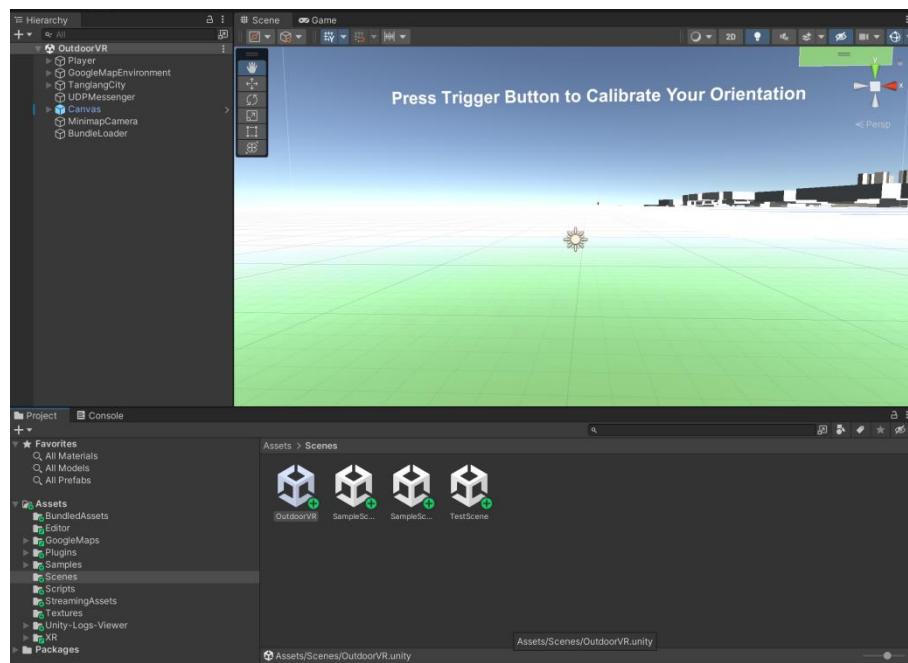
以同样的方式选择 OutdoorVR.unitypackage 文件，点击 import 导入



勾选如下组件导入至项目中



打开项目文件中的 Assets -> Scenes -> OutdoorVR.unity，将该游戏场景加载进 Unity，如下图所示



该场景中已事先创建了 UDPMessenger 和 BundleLoader 等组件，使用方法与 1.2 中描述一致，仅将 Cardboard 自带的游戏场景 CubeRoom 替换成塘朗地区五公里范围内的户外地图，并且支持通过手机 GPS 信号进行实时的地图定位

可利用该场景进行户外游戏场景的设计

