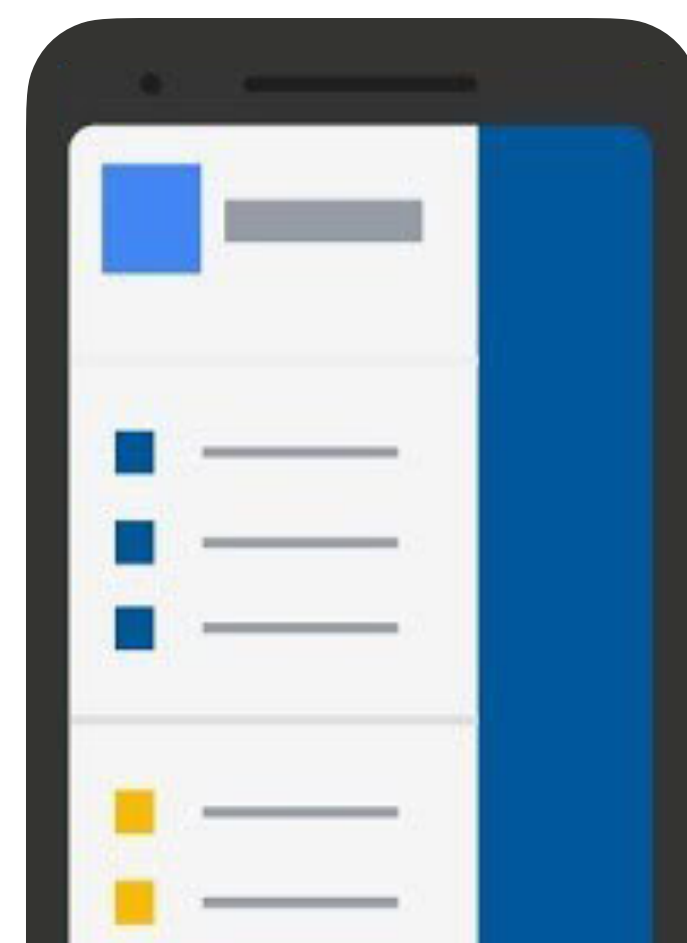
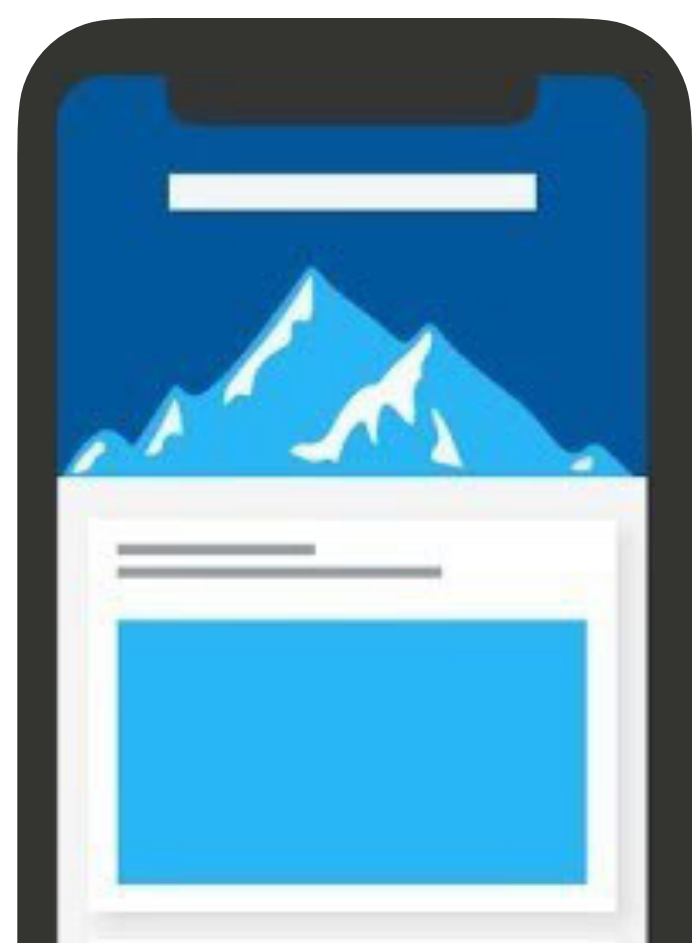


跨平台移动应用开发技术

章国锋



Flutter



Unity



跨平台移动应用开发技术

Cross-platform mobile application development



浙江大学
ZHEJIANG UNIVERSITY

原生应用 (Native application)

某一个移动平台（比如iOS或安卓）所特有的应用

iOS原生应用：Objective-C 或 Swift

Android原生应用：Java 或 Kotlin

- 可访问平台全部功能（GPS、摄像头）；
- 速度快、性能高、可以实现复杂动画及绘制；
- 整体用户体验好；
- 不同平台必须维护不同代码，开发成本高；
- 内容固定，动态化弱

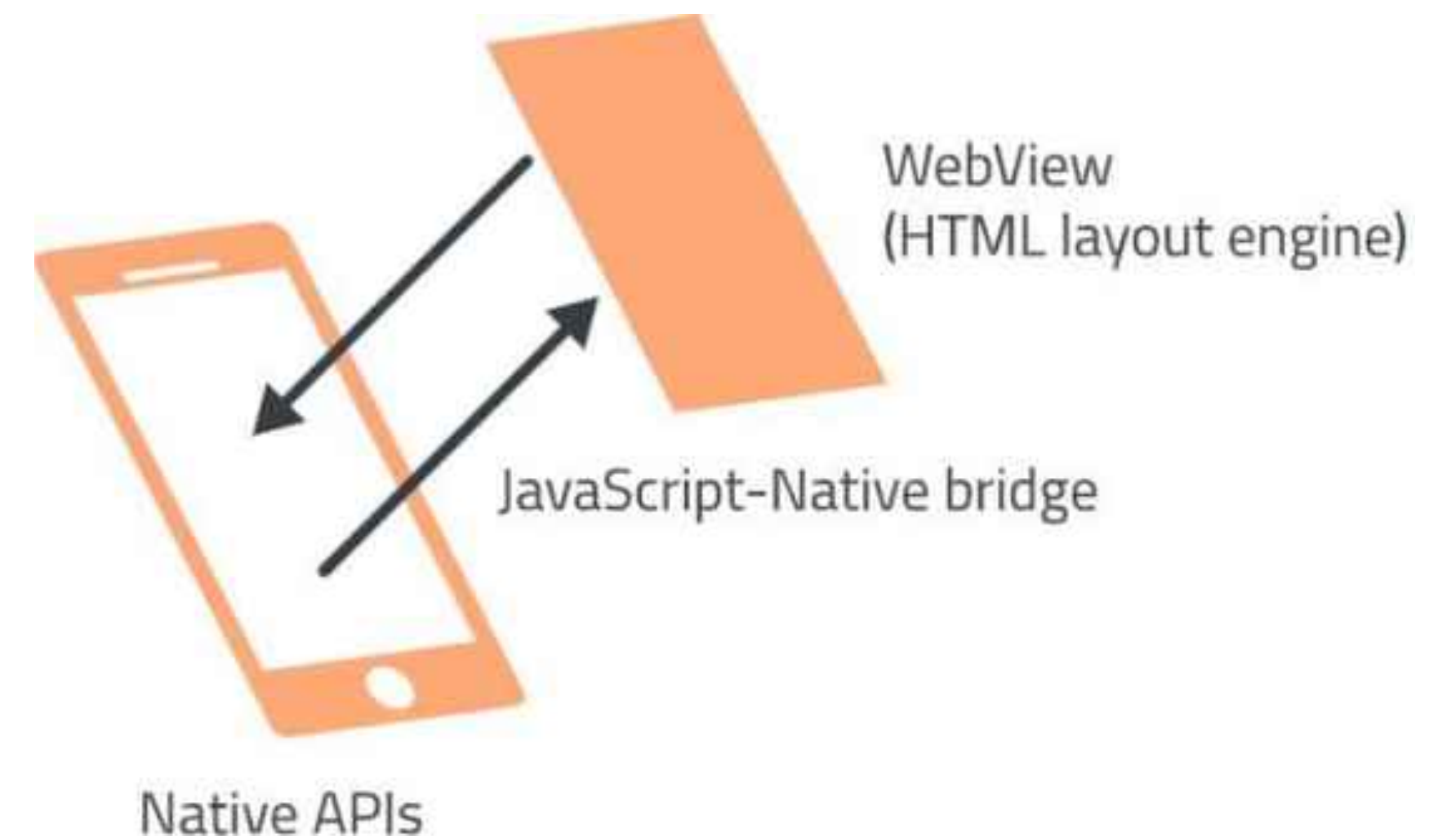
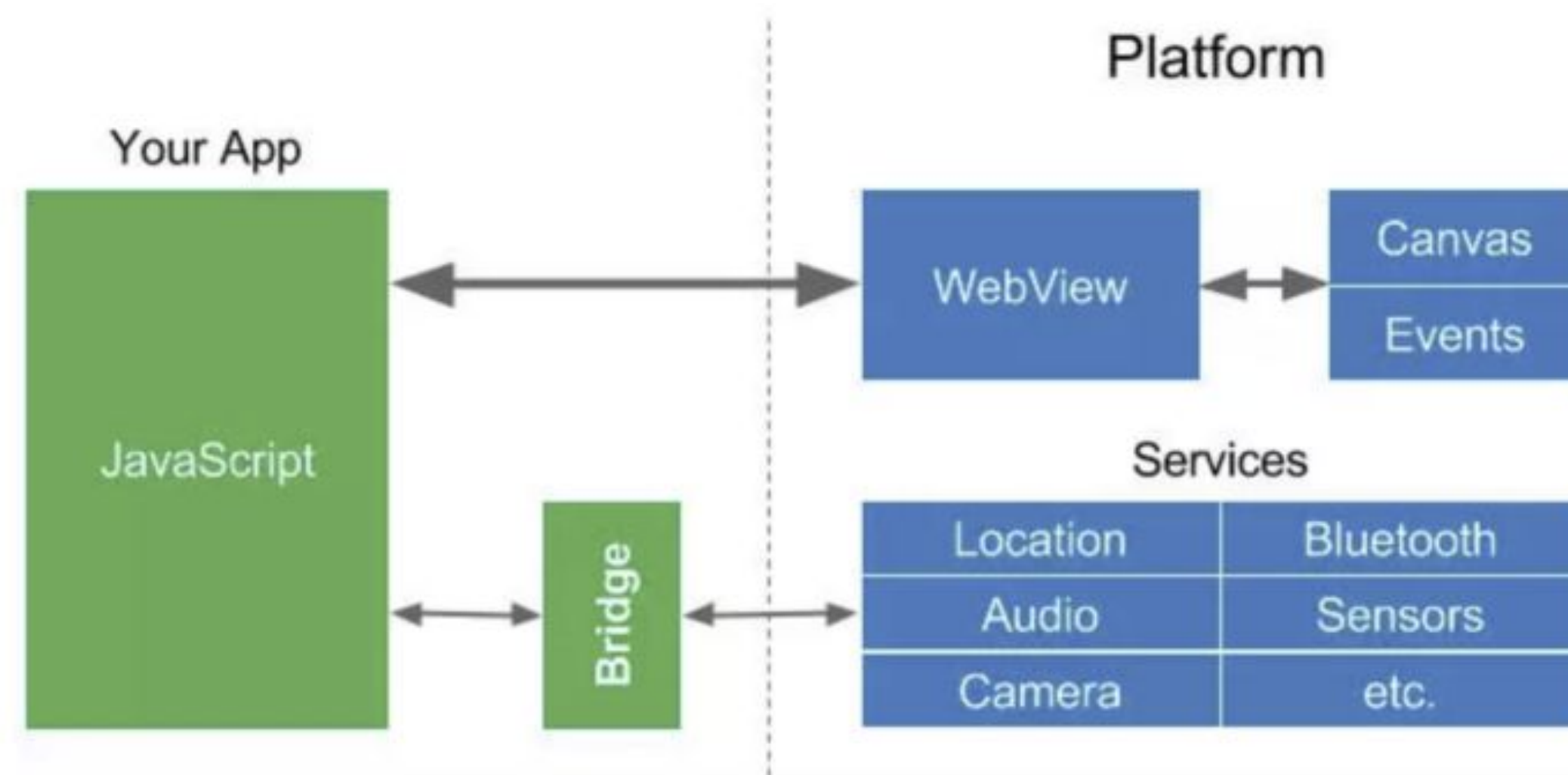


跨平台移动应用开发技术

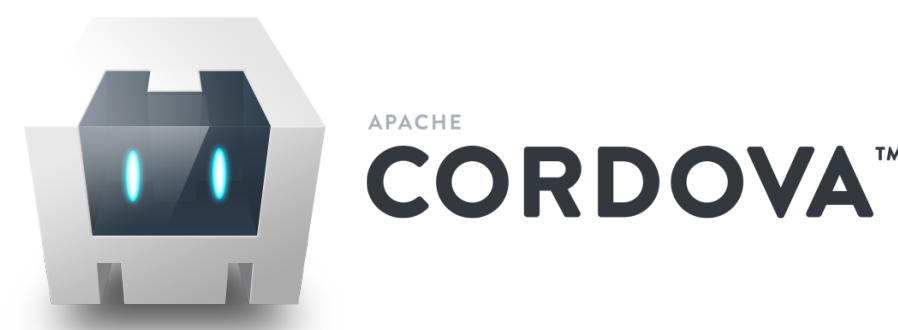
Cross-platform mobile application development



Solution one: HTML5 + Native app (混合开发技术)



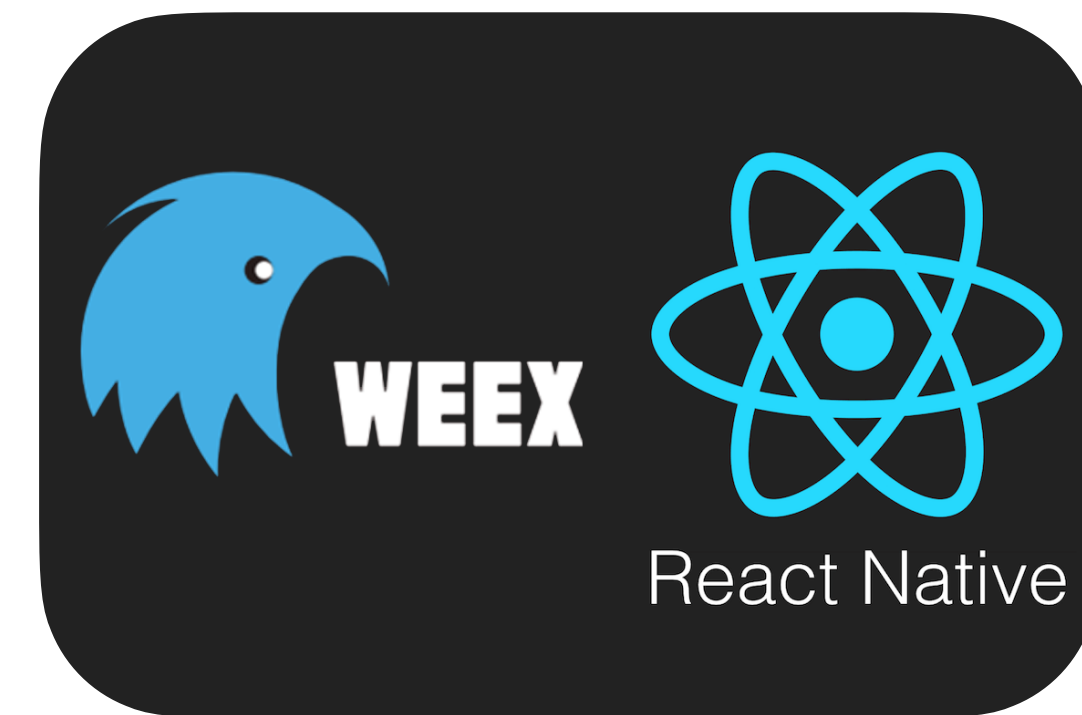
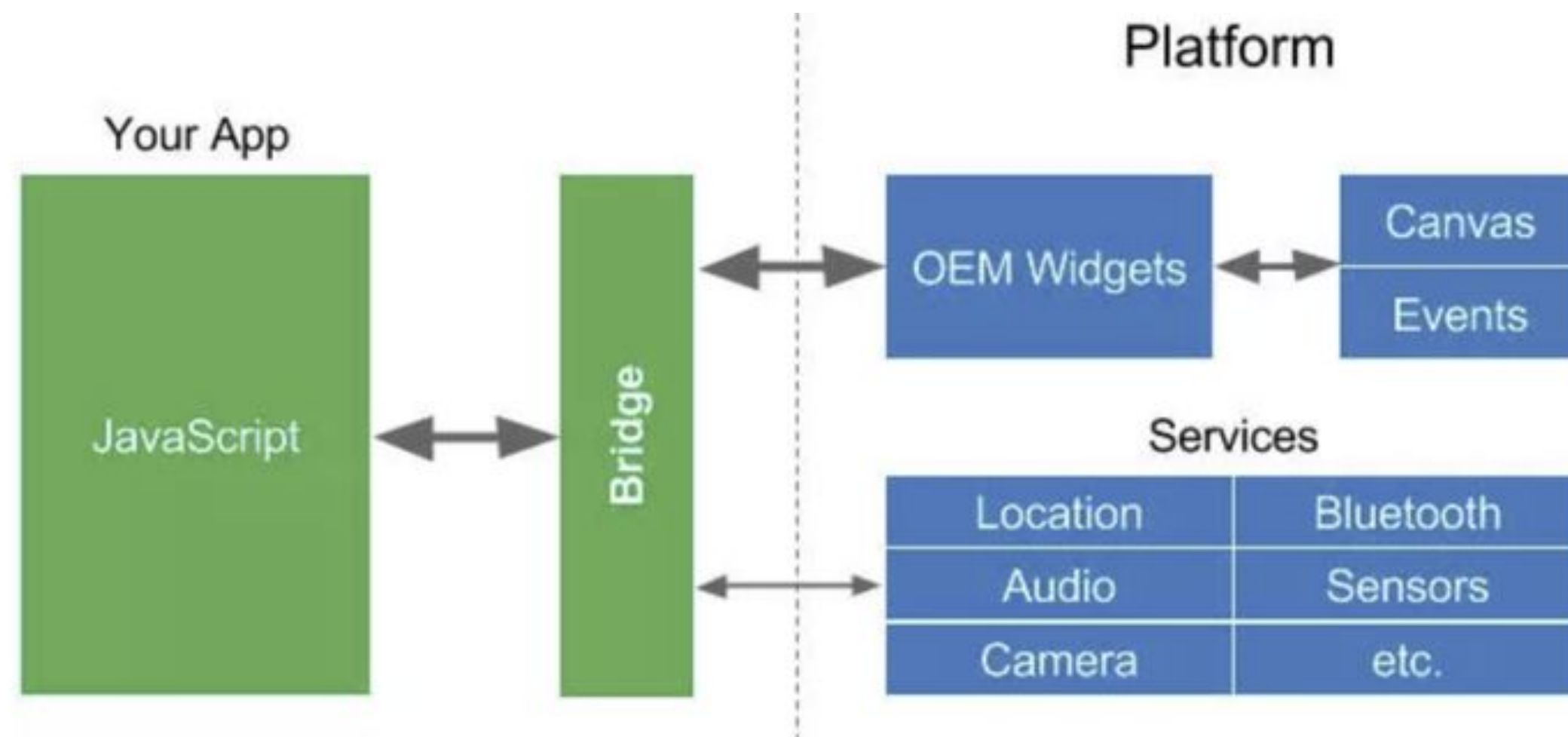
representative product:



跨平台移动应用开发技术

Cross-platform mobile application development

Solution two: JavaScript + Native render



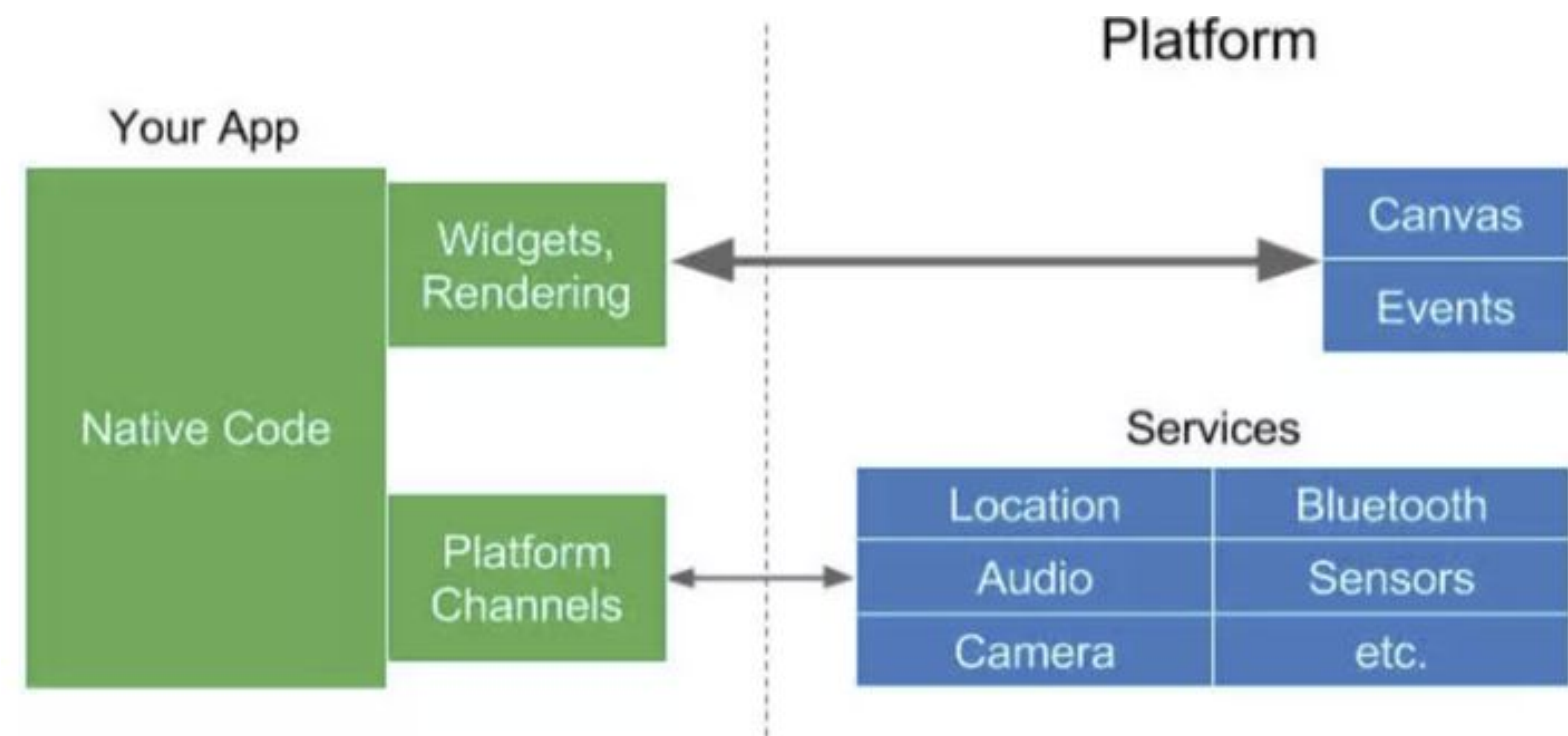
Representative product:



跨平台移动应用开发技术

Cross-platform mobile application development

Solution three: Custom UI + Native API



Representative product:



总结一下

Summary



技术类型	UI渲染方式	性能	开发效率	动态化	框架代表
H5 + 原生	WebView渲染	一般	高	支持	Cordova、Ionic
JavaScript + 原生渲染	原生控件渲染	好	中	支持	RN、Weex
自绘UI + 原生	调用系统API渲染	好	Flutter高, Qt低	默认不支持	Qt、Flutter

所以我们选择什么?

So what should we choose?



浙江大学
ZHEJIANG UNIVERSITY



1

使用Flutter跨平台移动应用开发

2

使用Unity跨平台AR开发

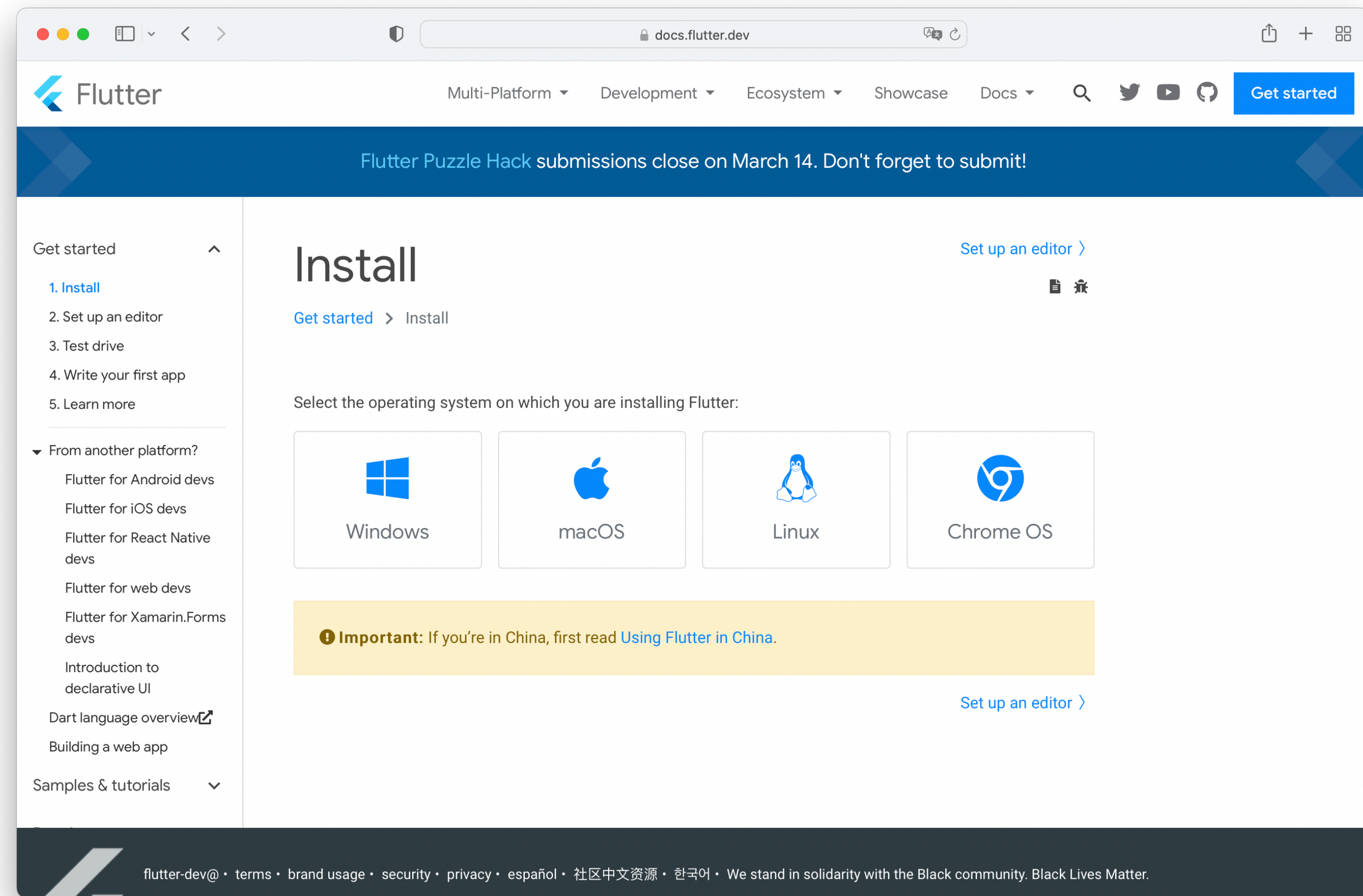
Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



跨平台移动应用开发技术

利用VSCode作为IDE

1. View > Command Palette
2. install > Extensions... > flutter
3. Restart VSCode

检查VSCode中Flutter是否安装成功

1. View > Command Palette
2. doctor > Flutter: Run Flutter Doctor

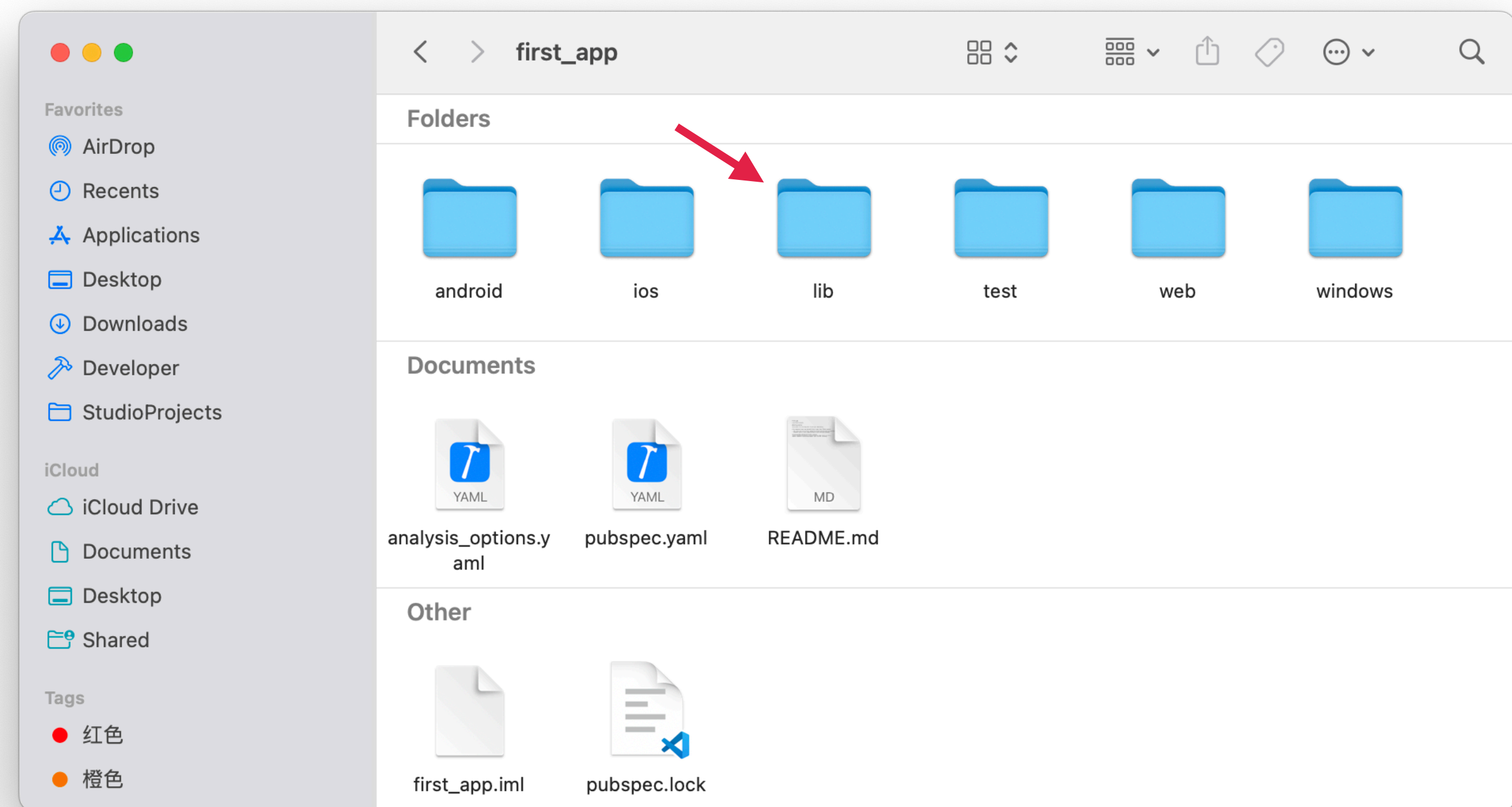
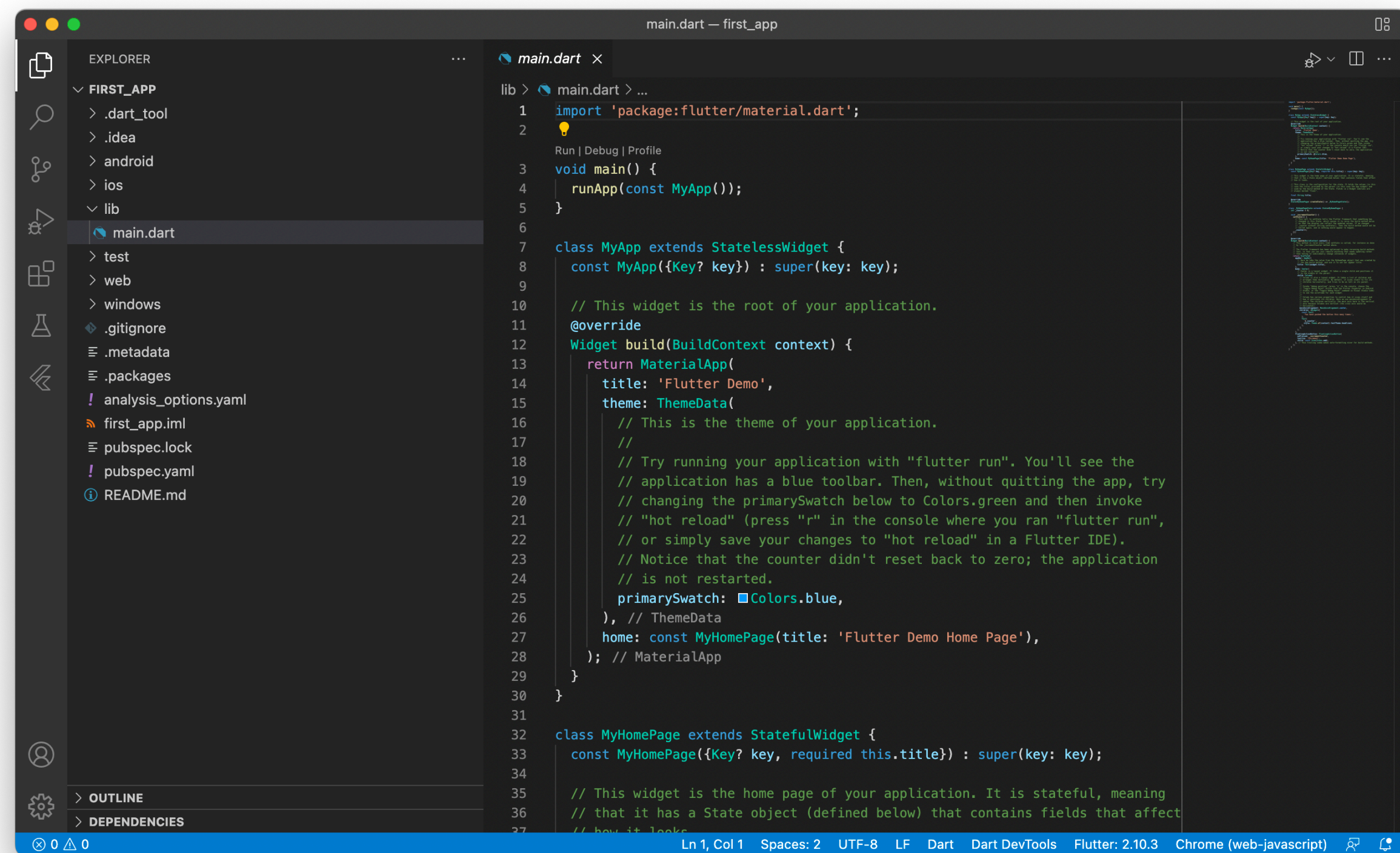
利用VSCode创建第一个Flutter程序

1. View > Command Palette
2. Flutter: New Project

Flutter跨平台开发技术



Cross-platform mobile application development with Flutter



Flutter跨平台开发技术



Cross-platform mobile application development with Flutter

Run flutter app:

```
flutter run
```

Open iOS simulator:

```
xcrun simctl list
```

```
open -a Simulator --args -CurrentDeviceUDID <Your device UDID>
```

Open Android emulator:

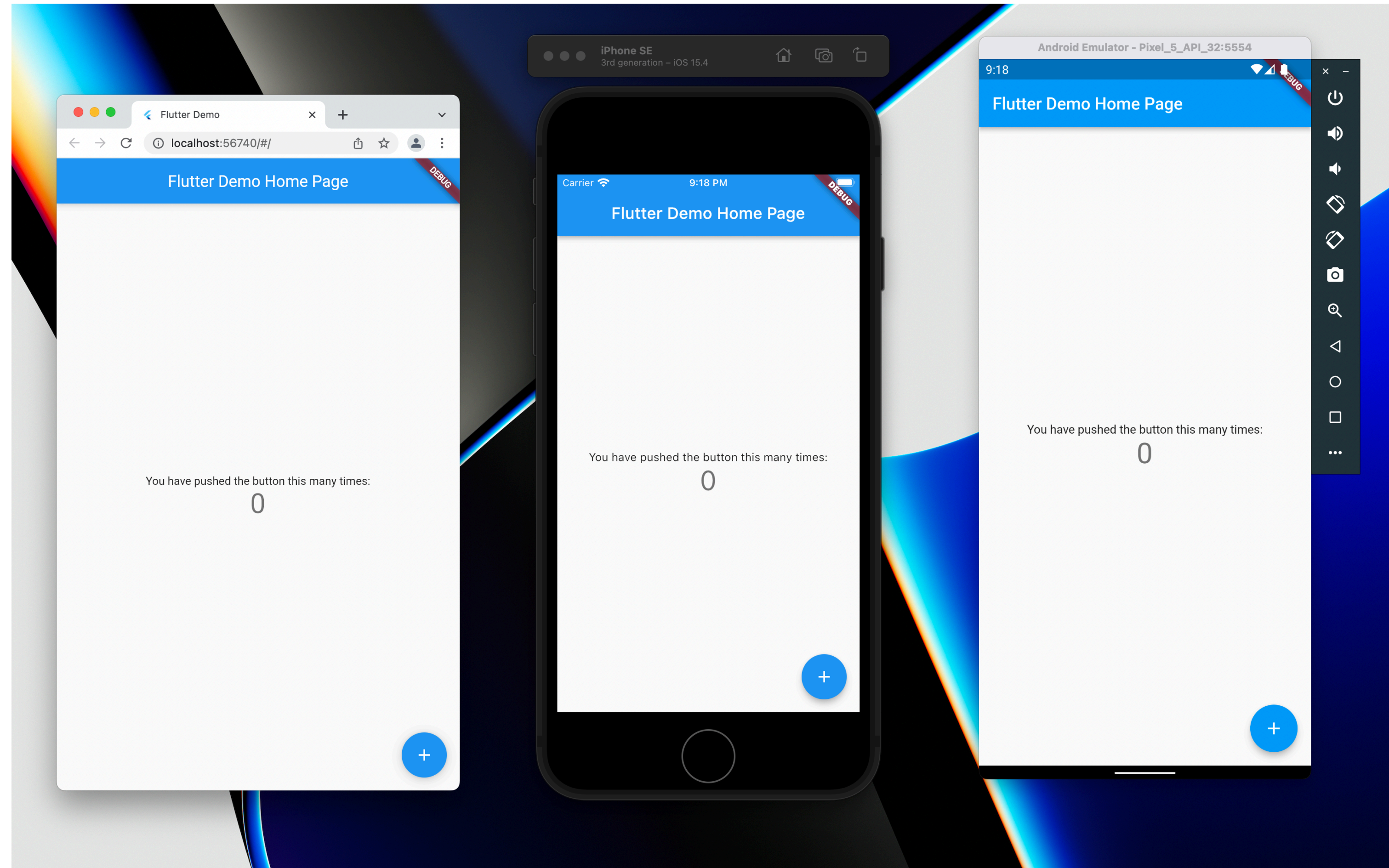
```
/Users/your/path/to/Android/sdk/emulator/emulator -avd Your_Pixel
```

Flutter run key commands:

```
r Hot reload. 🔥🔥🔥  
R Hot restart.  
q Quit (terminate the application on the device).
```


Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



Flutter跨平台开发技术



浙江大学
ZHEJIANG UNIVERSITY

Cross-platform mobile application development with Flutter

```
import 'package:flutter/material.dart'; ←

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

Flutter跨平台开发技术

Cross-platform mobile application development with Flutter

```
import 'package:flutter/material.dart';
```

```
void main() {  
  runApp(const MyApp());  
}
```

← MyApp是一个Widget参数

```
class MyApp extends StatelessWidget {  
  const MyApp({Key? key}) : super(key: key);
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(  
      title: 'Flutter Demo',  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
      ),  
      home: const MyHomePage(title: 'Flutter Demo Home Page'),  
    );  
  }
```


Flutter跨平台开发技术

Cross-platform mobile application development with Flutter

```
import 'package:flutter/material.dart';
```

```
void main() {  
  runApp(const MyApp());  
}
```

```
class MyApp extends StatelessWidget {  
  const MyApp({Key? key}) : super(key: key);
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(  
      title: 'Flutter Demo',
```

```
      theme: ThemeData(  
        primarySwatch: Colors.blue,
```

```
      ),  
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
```

```
    );  
  }
```

```
}
```

跨平台移动应用开发技术

← 应用名称

← 主题颜色

← 首页

Flutter跨平台开发技术



Cross-platform mobile application development with Flutter

```
class MyHomePage extends StatefulWidget {  
  const MyHomePage({Key? key, required this.title}) : super(key: key);  
  
  final String title;  
  
  @override  
  State<MyHomePage> createState() => _MyHomePageState();  
}
```

```
class _MyHomePageState extends State<MyHomePage> {  
  
  . . .  
  
}
```

Flutter跨平台开发技术



浙江大学
ZHEJIANG UNIVERSITY

Cross-platform mobile application development with Flutter

```
class _MyHomePageState extends State<MyHomePage> {  
  int _counter = 0;  ← 状态量  
  
  void _incrementCounter() {  
    setState(() {  
      _counter++;  ← 维护状态量的函数  
    });  
  }  
  
  . . .  
}  
  
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text(widget.title),  
    ),  
    body: Center(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: <Widget>[  
          const Text(  
            'You have pushed the button this many times:',  
          ),  
          Text(  
            '$_counter',  
            style: Theme.of(context).textTheme.headline4,  
          ),  
        ],  
      ),  
    ),  
    floatingActionButton: FloatingActionButton(  
      onPressed: _incrementCounter,  
      tooltip: 'Increment',  
      child: const Icon(Icons.add),  
    ),  
  );  
}
```


Flutter跨平台开发技术



浙江大学
ZHEJIANG UNIVERSITY

Cross-platform mobile application development with Flutter

```
class _MyHomePageState extends State<MyHomePage> {  
  int _counter = 0;  ← 状态量  
  
  void _incrementCounter() {  
    setState(() {  
      _counter++;  ← 维护状态量的函数  
    });  
  }  
  
  . . .  
}  
  
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text(widget.title),  
    ),  
    body: Center(  
      child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: <Widget>[  
          const Text(  
            'You have pushed the button this many times:',  
          ),  
          Text(  
            '$_counter',  
            style: Theme.of(context).textTheme.headline4,  
          ),  
        ],  
      ),  
    ),  
    floatingActionButton: FloatingActionButton(  
      onPressed: _incrementCounter,  
      tooltip: 'Increment',  
      child: const Icon(Icons.add),  
    ),  
  );  
}
```

Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



Widget

StatelessWidget

StatelessWidget用于不需要维护状态的场景，它通常在build方法中通过嵌套其它 widget 来构建UI，在构建过程中会递归的构建其嵌套的 widget

StatefulWidget

和StatelessWidget一样，StatefulWidget也是继承自Widget类，相比StatelessWidget需要多重写一个函数createState

State

表示与其对应的 StatefulWidget 要维护的状态

Flutter跨平台开发技术



Cross-platform mobile application development with Flutter

Text

```
Text("Hello world",
  textAlign: TextAlign.left,
);
```

Button

```
ElevatedButton(
  child: Text("normal"),
  onPressed: () {},
);
```

Image

```
Image(
  image: AssetImage("images/avatar.png"),
  width: 100.0
);
```

Switch

Checkbox

```
Switch(
  value: _switchSelected, // 当前状态
  onChanged: (value) {
    // 重新构建页面
    setState(() {
      _switchSelected = value;
    });
  },
);
```

LinearProgressIndicator

TextField

```
TextField(
  autofocus: true,
  decoration: InputDecoration(
    labelText: "用户名",
    hintText: "用户名或邮箱",
    prefixIcon: Icon(Icons.person)
  ),
);
```

Constraints

Row and Column

Align

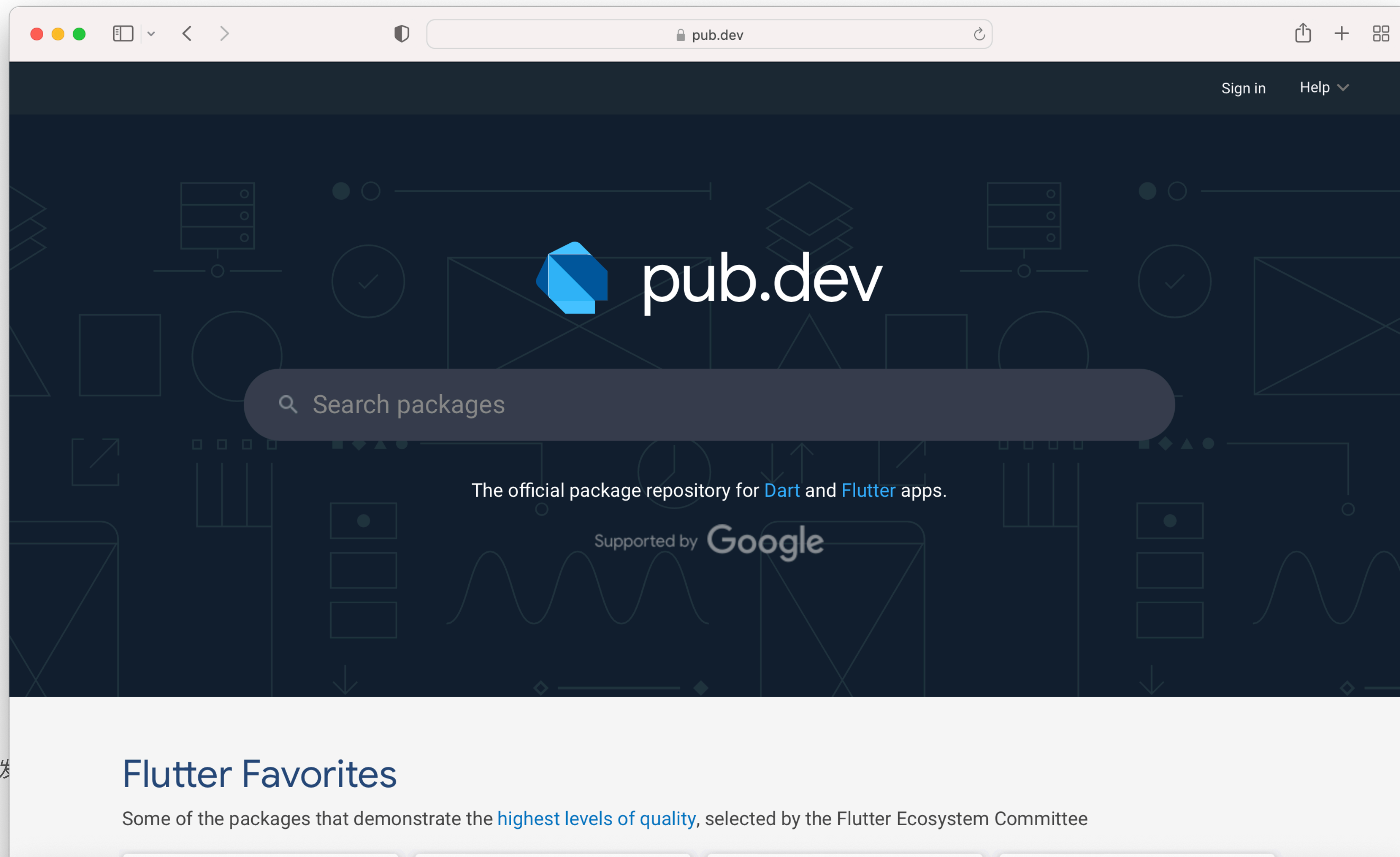
Animation

Gesture

Flutter跨平台开发技术



Cross-platform mobile application development with Flutter



跨平台移动应用开发

Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



pub.dev

Search Sign in Help

arkit_plugin 1.0.5

Published 7 months ago Null safety

SDK FLUTTER PLATFORM IOS 278

Readme Changelog Example Installing Versions Scores

Use this package as a library

Depend on it

Run this command:

With Flutter:

\$ flutter pub add arkit_plugin

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:
 arkit_plugin: ^1.0.5

278 120 89%
LIKES PUB POINTS POPULARITY

Publisher

unverified uploader

Metadata

Flutter Plugin for ARKit - Apple's augmented reality (AR) development platform for iOS mobile devices.

Repository (GitHub)
View/report issues

Documentation

API reference

License

Flutter跨平台开发技术

Cross-platform mobile application development with Flutter



One more thing...

1

使用Flutter跨平台移动应用开发

2

使用Unity跨平台AR开发

使用Unity跨平台AR开发

Cross-platform AR application development with Unity

ices around the world



使用Unity跨平台AR开发

Cross-platform AR application development with Unity



跨平台移动应用开发技术

AR 应用：游戏

AR application: Games



Pokemon GO



网易悠梦



AR 应用：生活

AR application: Life



浙江大学
ZHEJIANG UNIVERSITY



AR试妆



商汤室内WebAR导航



跨平台移动应用开发技术

AR 应用：工业

AR application: industry



Waveoptics 物流辅助AR



在实际应用中

商汤电网巡检辅助AR



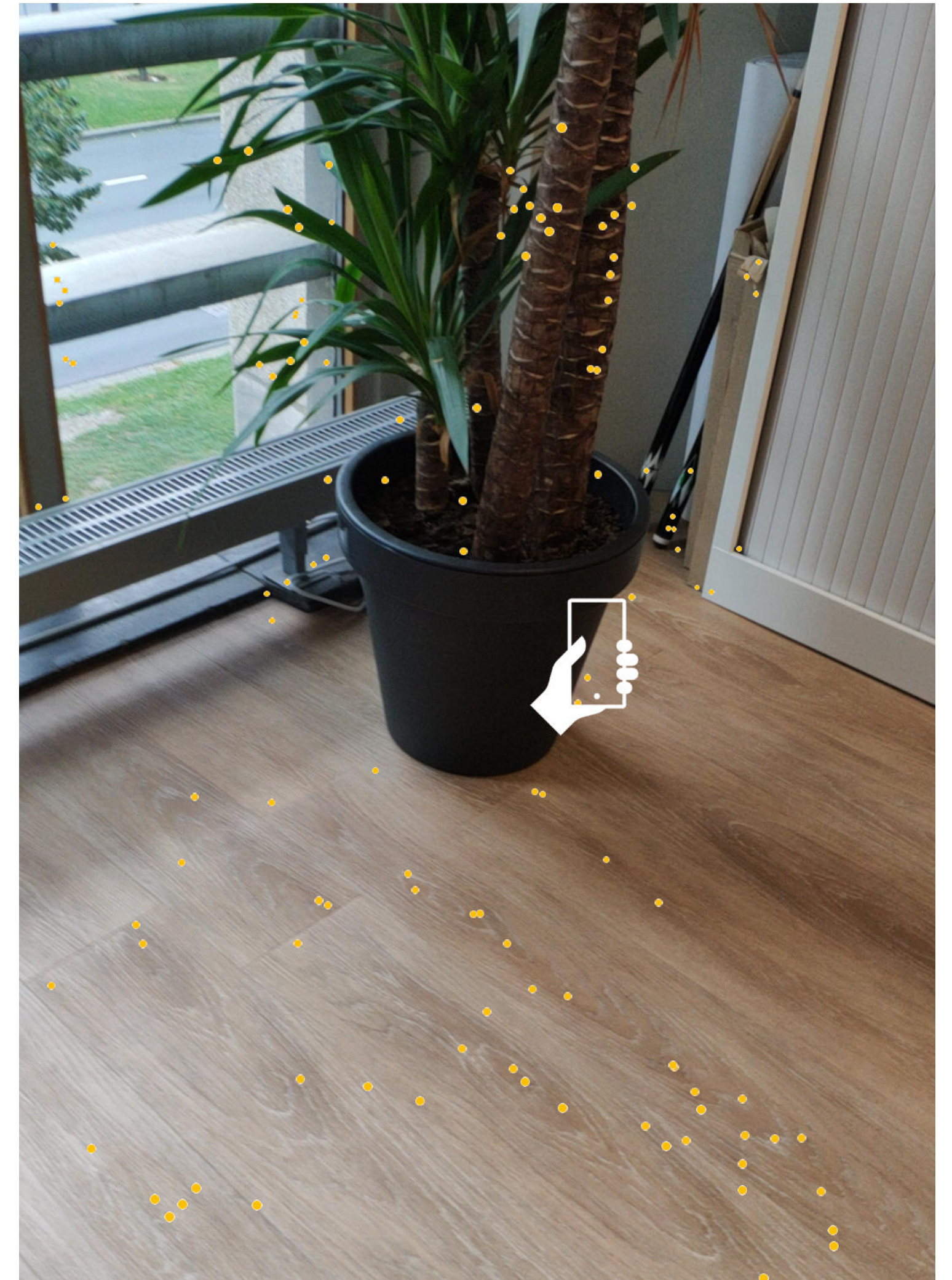
复习一下AR背后的原理

How does AR works

- AR App使用相机和深度传感器持续检测并跟踪真实世界的特征点



浙江大学
ZHEJIANG UNIVERSITY



复习一下AR背后的原理

How does AR works

- AR App使用相机和深度传感器持续检测并跟踪真实世界的特征点
- 根据特征点，AR app 在真实世界中创建水平或者竖直的虚拟平面，映射虚拟世界与真实世界



复习一下AR背后的原理

How does AR works

- AR App使用相机和深度传感器持续检测并跟踪真实世界的特征点
- 根据特征点，AR app 在真实世界中创建水平或者竖直的虚拟平面，映射虚拟世界与真实世界
- 通过用户交互，将ray投射到现实世界，以对准找到平面或者特征点以放置内容



复习一下AR背后的原理

How does AR works

- AR app 持续**跟踪设备**在真实世界的**位置和姿态**，以绘制不同位姿下的虚拟物体
- 估算**平均亮度、色温和色彩校正**，用此刻调整实时定向光的值



浙江大学
ZHEJIANG UNIVERSITY



跨平台移动应用开发技术

从AR, VR到MR

from AR to MR

Mixed Reality |



不同移动平台的AR SDK

different AR sdk



跨平台移动应用开发技术

Unity AR Foundation



AR Foundation

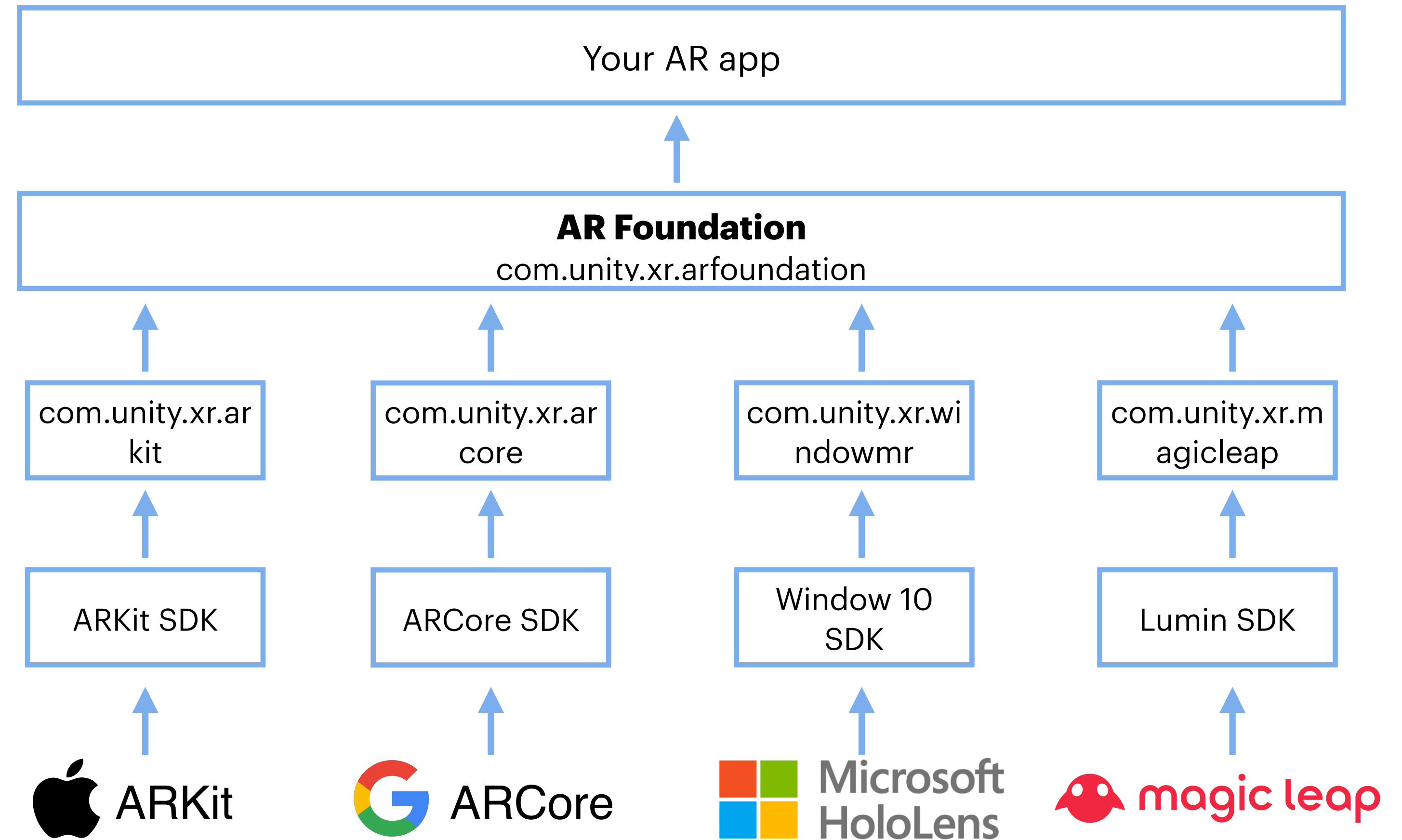


Unity AR Foundation



浙江大学
ZHEJIANG UNIVERSITY

- Unity AR Foundation是Unity设计的与平台无关的AR接口库，它允许开发者使用统一的开发接口，而不用考虑各平台间的底层实现的差异

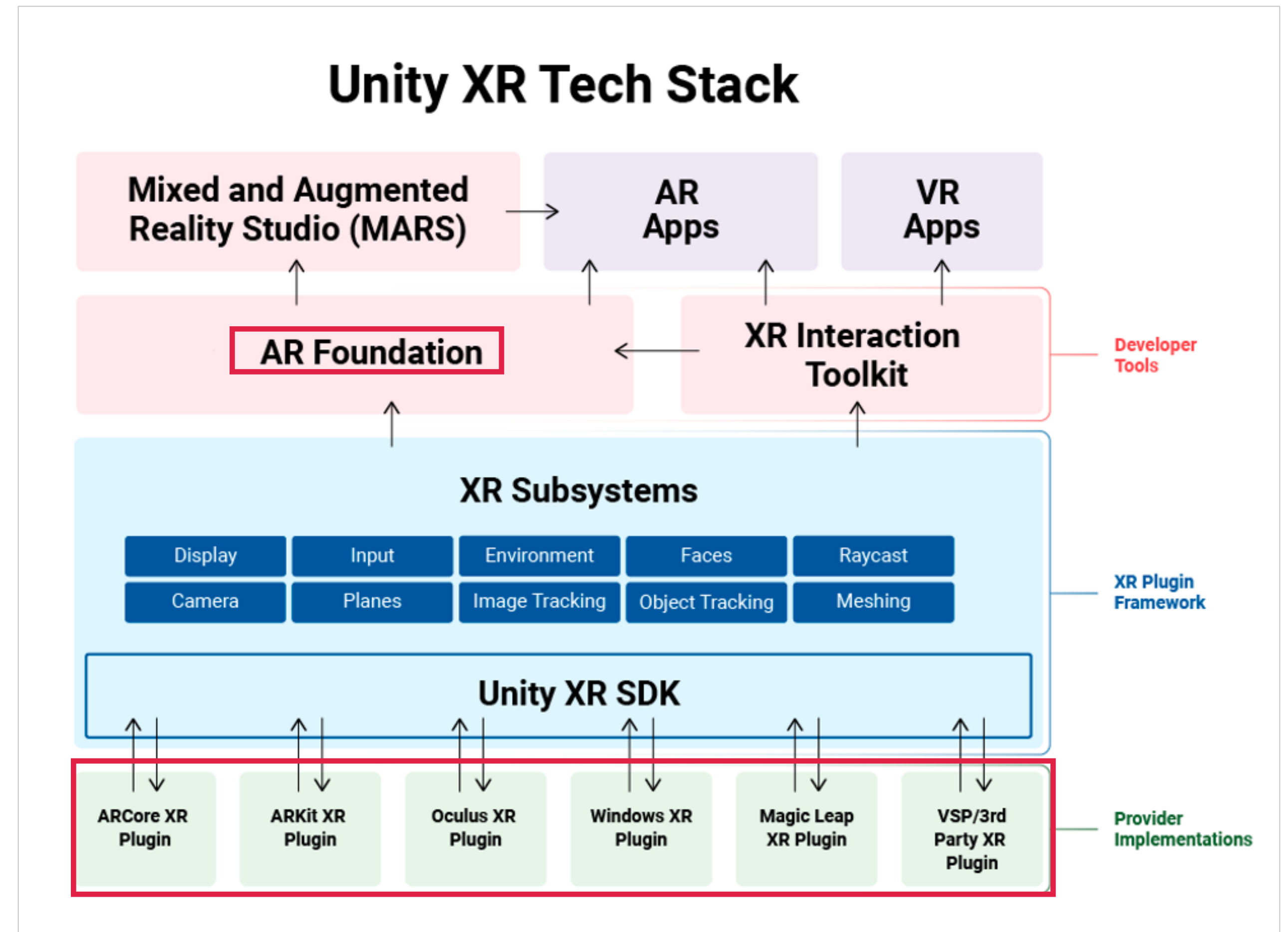


Unity AR Foundation



浙江大学
ZHEJIANG UNIVERSITY

- Unity的XR框架支持虚拟现实（VR）、增强现实（AR）和混合现实（MR）三种类型的应用，它的结构图如右图



AR Foundation功能支持



	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
2D Image tracking	✓	✓	✓	
3D Object tracking		✓		
Meshing		✓	✓	✓
2D & 3D body tracking		✓		
Collaborative participants		✓		
Human segmentation		✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓
Occlusion	✓	✓		

Unity's AR Foundation Supported Features

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

AR Foundation 2.1 (verified in Unity 2019 LTS)

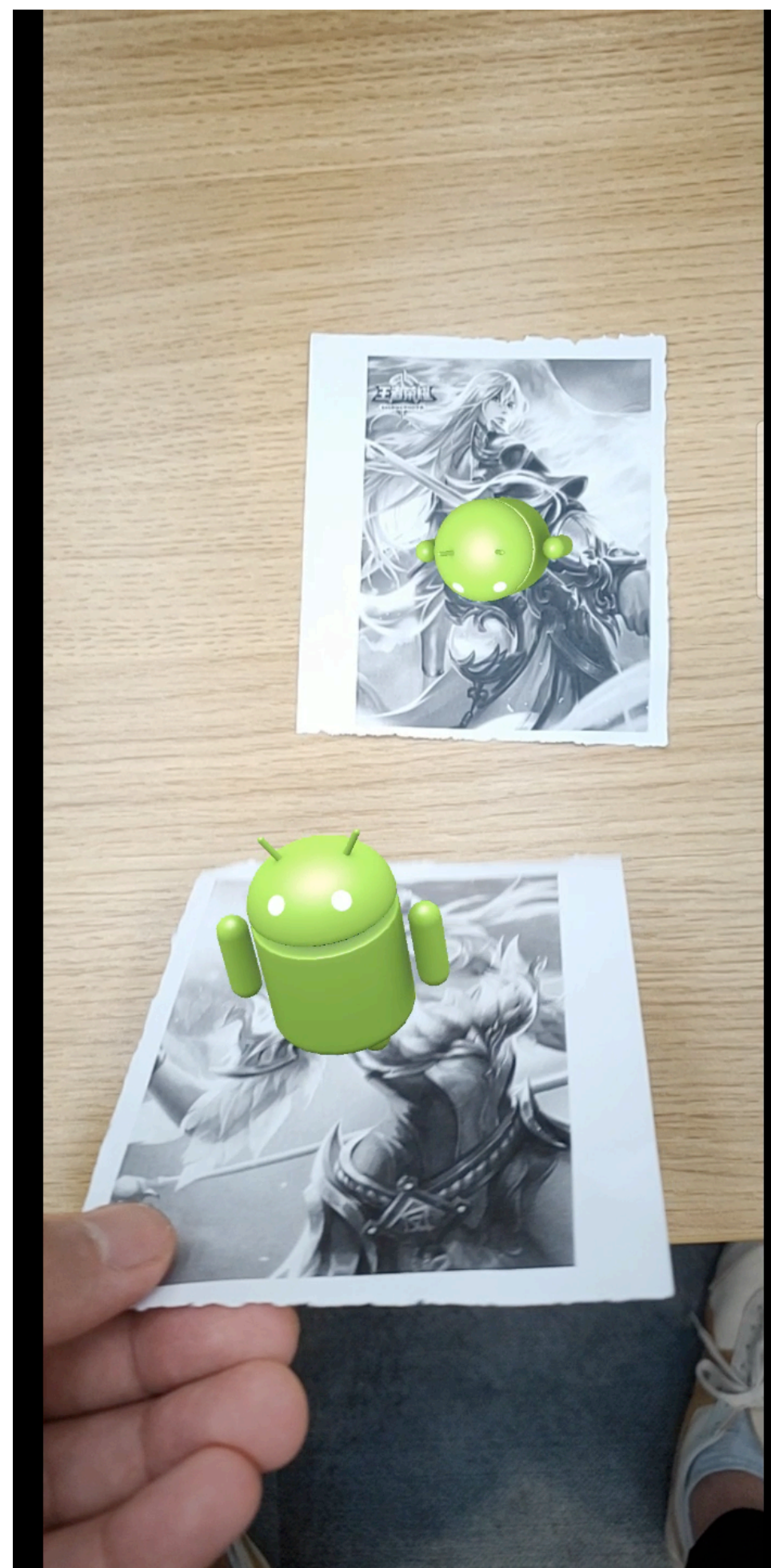


AR foundation: Image Tracking



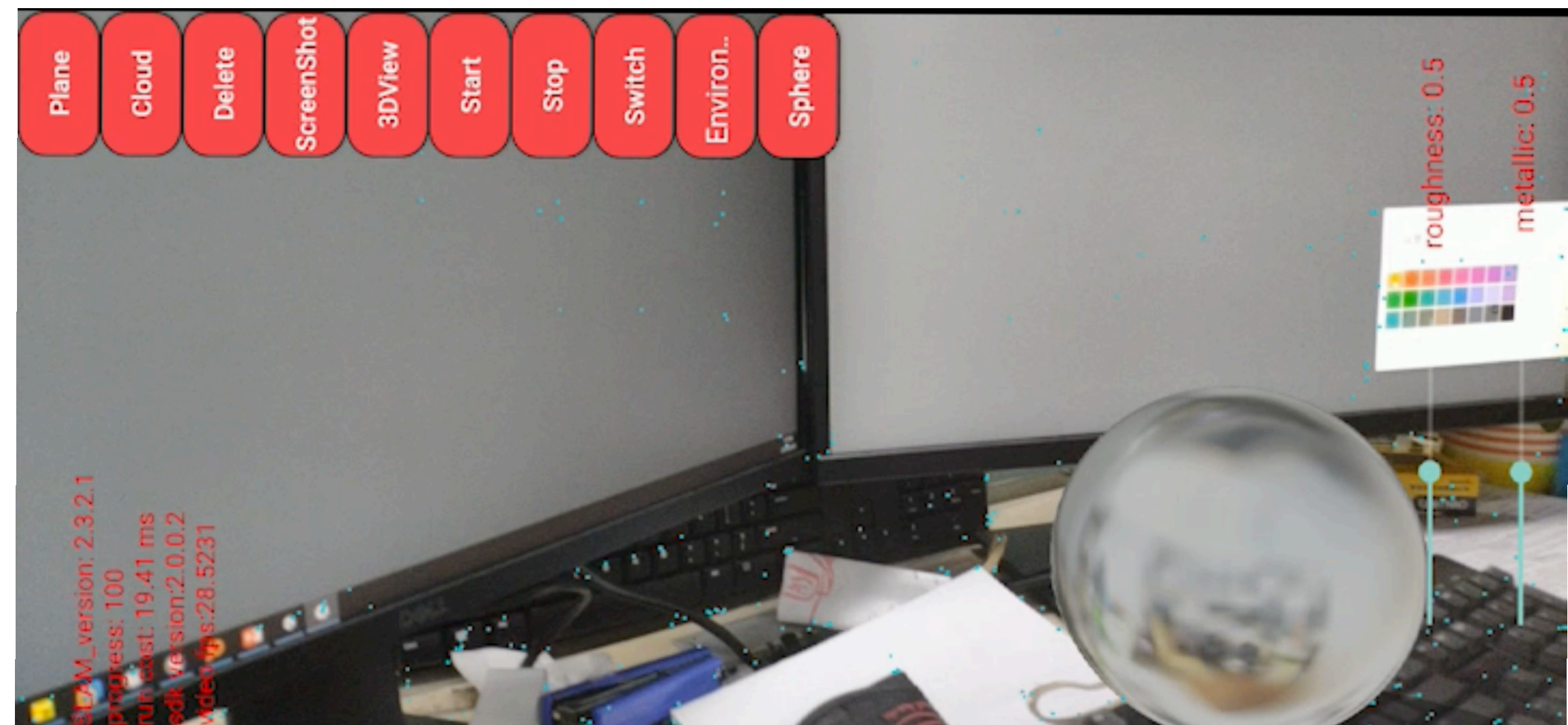
浙江大学
ZHEJIANG UNIVERSITY

- **图像识别**是AR应用中较为常见的展现形式，通过增强图像功能，在**识别图片后，可以在图片上方加载出三维立体的虚拟内容**。图片识别主要应用于构建识别平面图像，实现海报照片、书籍、卡片或产品包装等图片的AR效果。



AR foundation: Environments probes

- **环境纹理**主要用于为AR虚拟内容的光滑表面实现**反光效果**，将**真实场景中的光线**和物体渲染在虚拟对象的表面上，进一步实现虚实融合的效果。

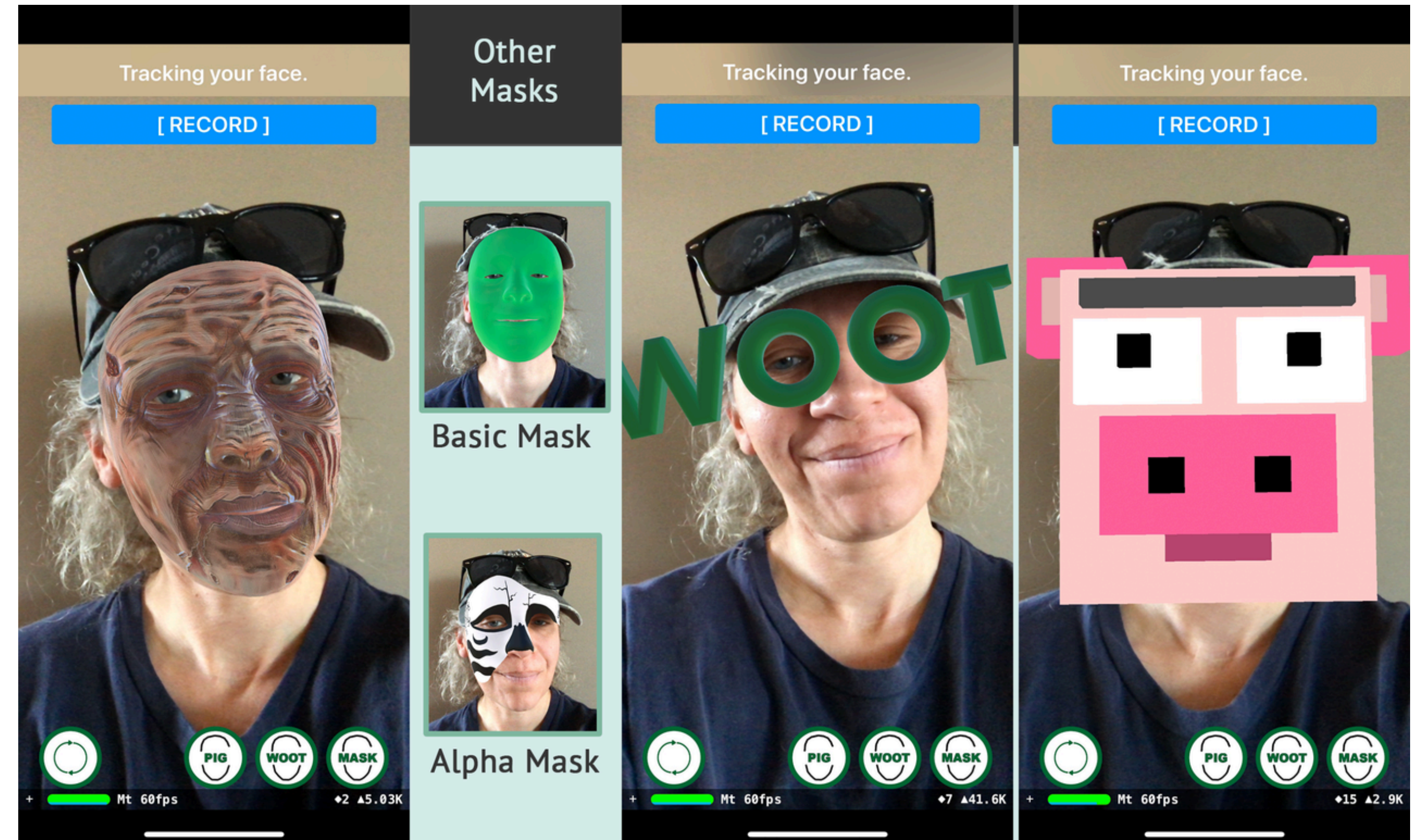


AR foundation: Face Tracking



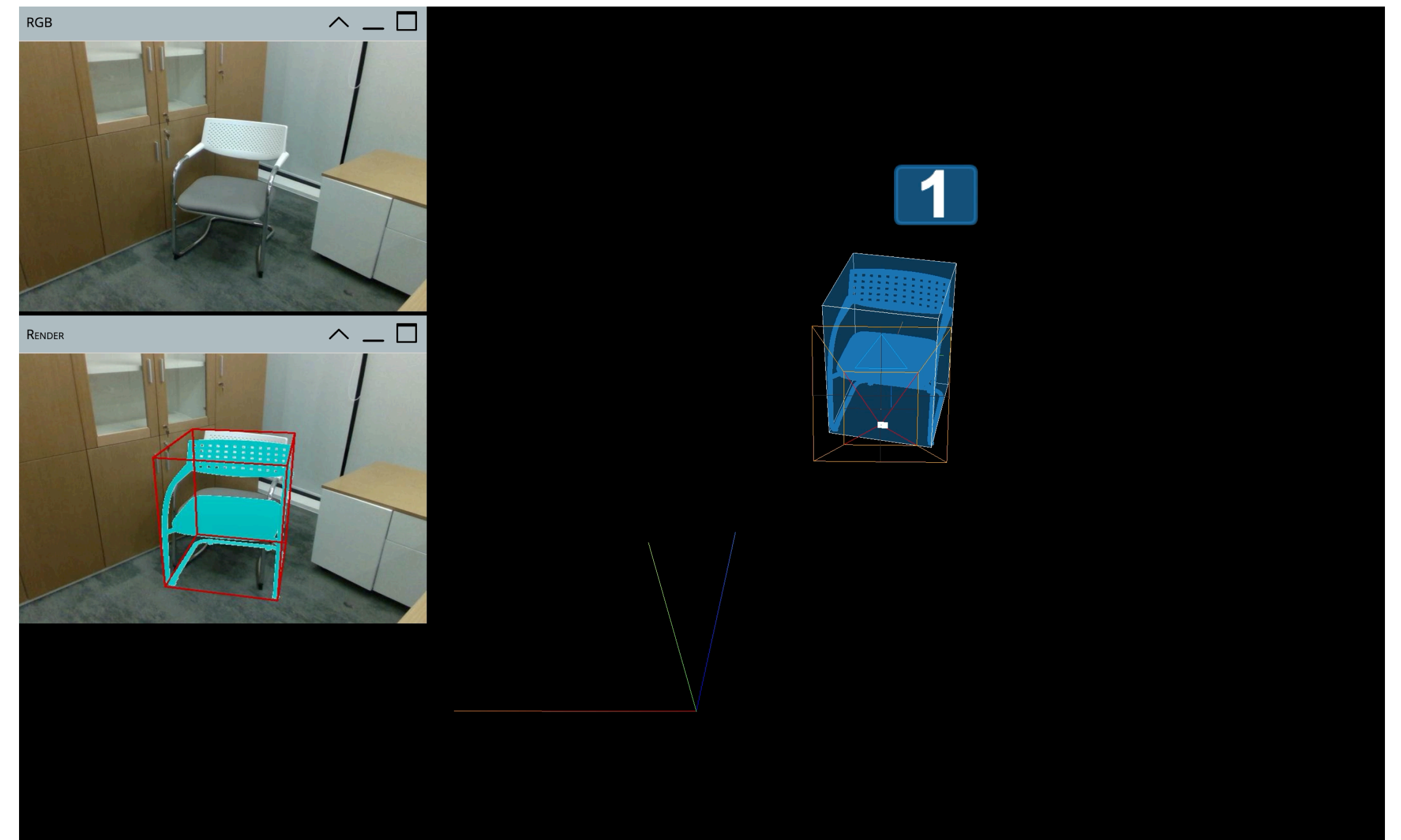
浙江大学
ZHEJIANG UNIVERSITY

- 脸部追踪通过深度相机追踪用户脸部，在运行时生成网格以及混合系数的值，将表情符号覆盖在跟踪的脸上。



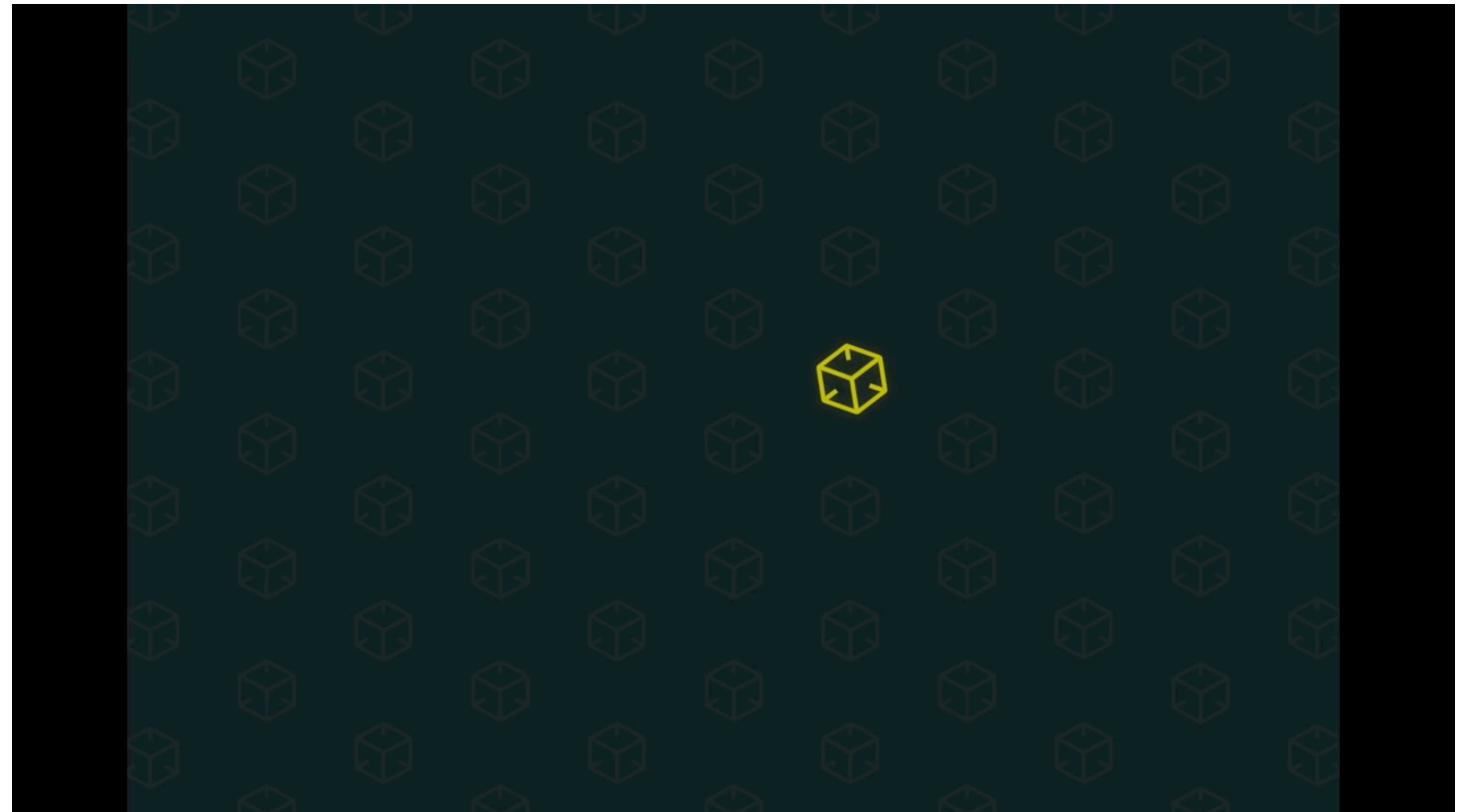
AR foundation: Object detection

- 物体检测 (Object Detection)
也叫三维物体识别、模型识别，
主要用于记录现实世界中三维物
体的空间特征。
- 使用物体识别技术，当用户在当
前环境中扫描，当检测到这个物
体时，就会触发相应的AR内
容。



AR foundation: Occlusion

- **虚实遮挡**的目标是在创建AR场景时，保留现实场景中的**视线视距规则**。这意味着任何位于真实对象后面的虚拟对象应该被“遮挡”，或隐藏在该真实对象后面。

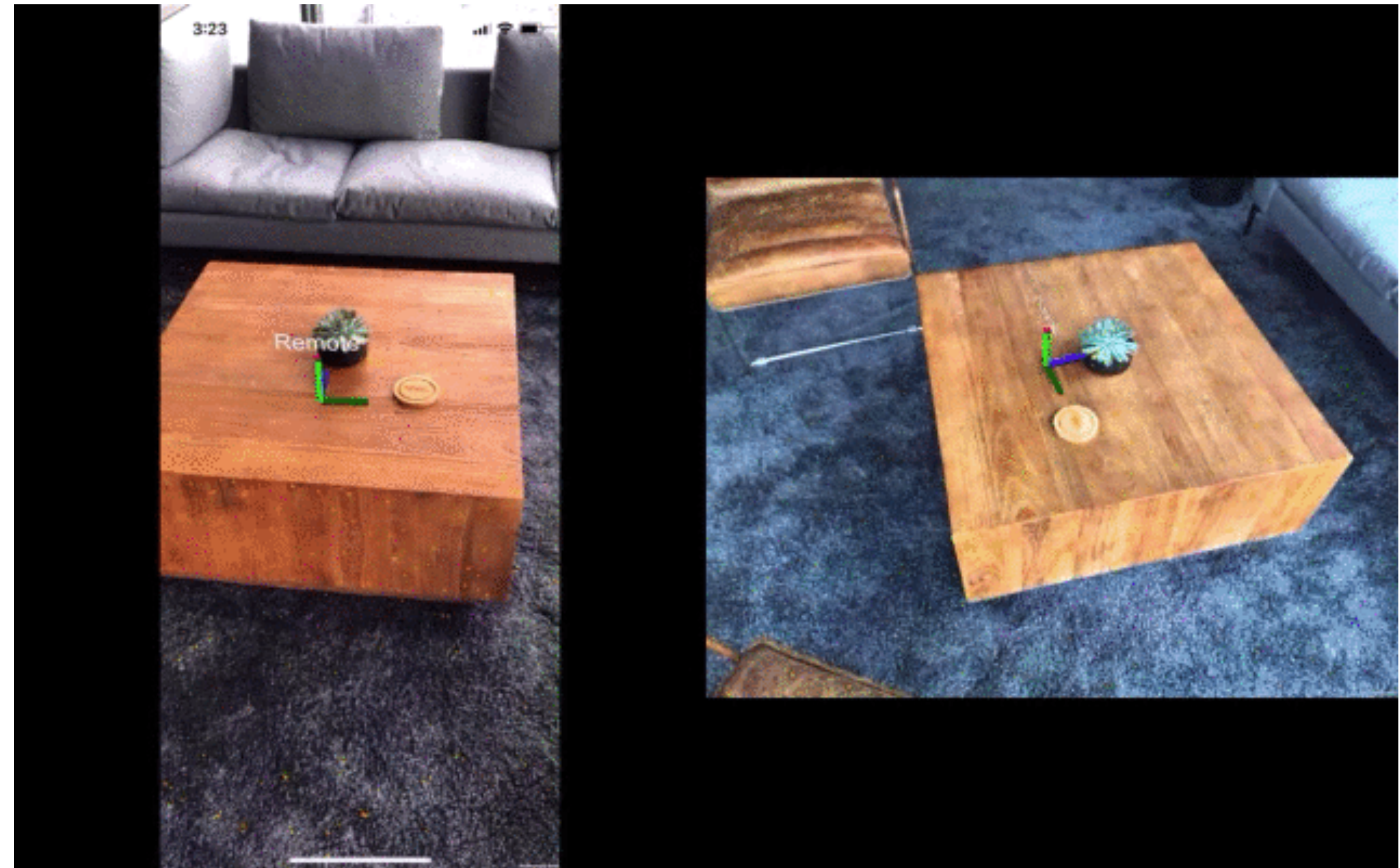


AR foundation: Participant tracking



浙江大学
ZHEJIANG UNIVERSITY

- 多个AR协同，跟踪共享AR设备的位置和朝向
 - ARKit: **World map**通过扫描环境并将每个用户重新定位到相同的坐标空间来创建多用户体验
 - ARCore: 在现实世界中设置**Cloud Anchors**并将其同步到云端，在另一台设备上解析次锚点以创建多用户体验



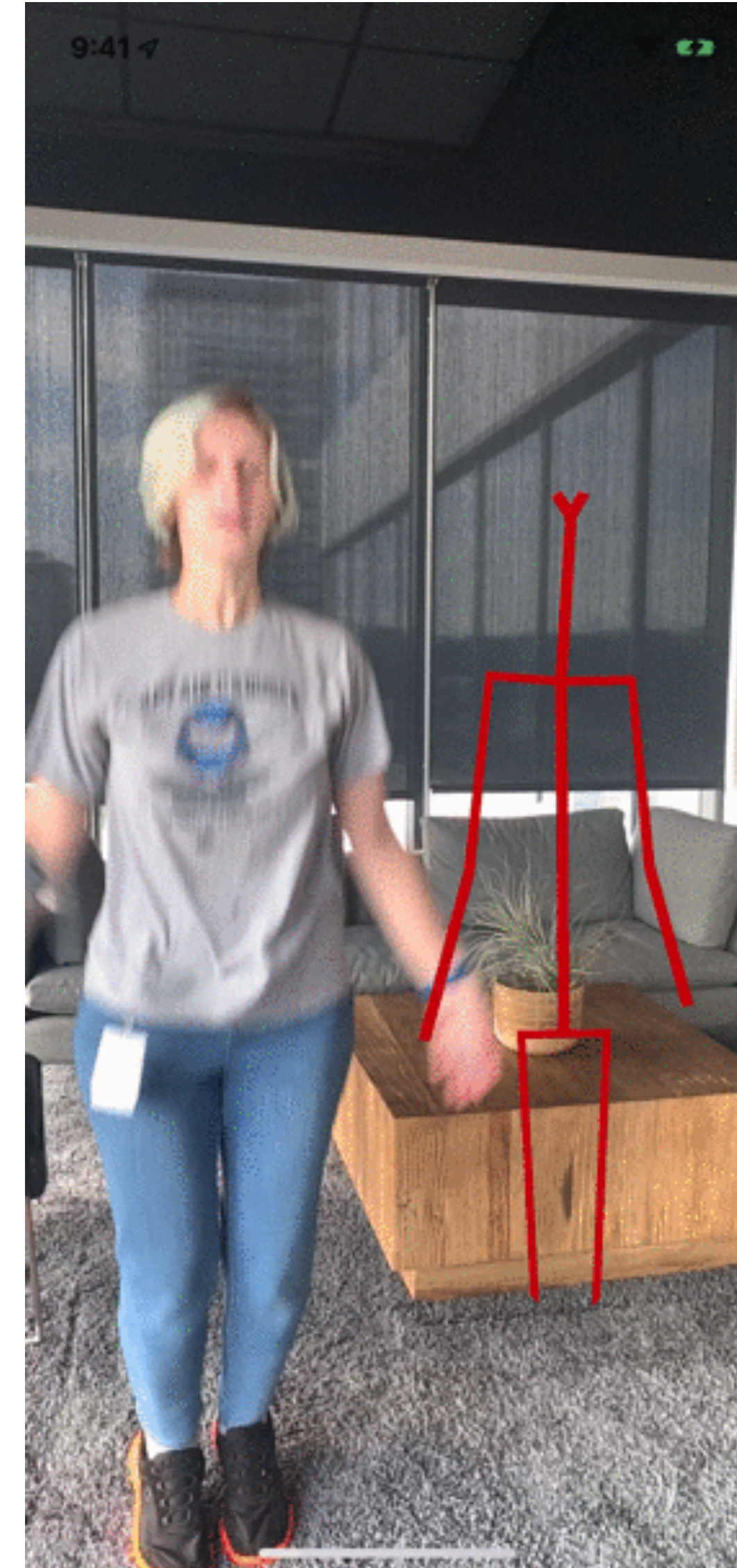
AR foundation: Motion Capture



浙江大学
ZHEJIANG UNIVERSITY

Work with ARKit

- 通过追踪多达91个独特关节来捕捉人体的身体动作，2D或3D跟踪都可



跨平台移动应用开发技术

AR foundation: Meshing



浙江大学
ZHEJIANG UNIVERSITY

Work with ARKit

- 通过检测环境，实时生成**真实场景的Mesh结构**，可以被用于实现碰撞，遮挡等效果



Attention!



Unity Hub

- Unity Hub是一个方便我们管理Unity项目和不同版本Unity引擎编辑器的工具。
- 支持功能：
 - 管理我们的Unity账号和Unity编辑器证书。
 - 创建使用不同版本Unity编辑器的Unity项目。
 - 从项目视图使用特定版本的Unity编辑器打开Unity项目。
 - 同时运行多个不同版本的Unity编辑器，但这样做可能会引发编辑器之间的冲突。
 - 为已经安装的Unity编辑器添加新的组件，比如Visual Studio，离线文档等等。
 - 使用项目模板来创建具有共同初始设置的Unity项目。



安装Unity Hub



Unity

产品列表

解决方案

企业支持服务

教育服务

游戏服务

开发者社区

中文文档

用户案例

下载Unity

所有版本

长期支持版

补丁程序版

Beta版

切换Hub 2.5

下载Unity Hub

所有版本

你好！我们知道您想快速下载并开始使用Unity，这里有所有正式发布的Unity版本，但我们更建议您从安装Unity Hub开始！现在就开始吧！

Unity 2022.x

Unity 2021.x

Unity 2020.x

Unity 2019.x

Unity 2018.x

Unity 2017.x

Unity 5.x

Unity 4.x

Unity 3.x

搜索Release Notes

Unity 2022.x

2022.1.0

13 May, 2022

从Hub下载

下载 (Mac)

下载 (Win)

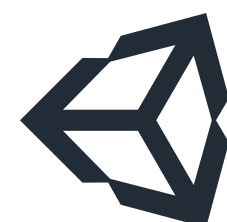
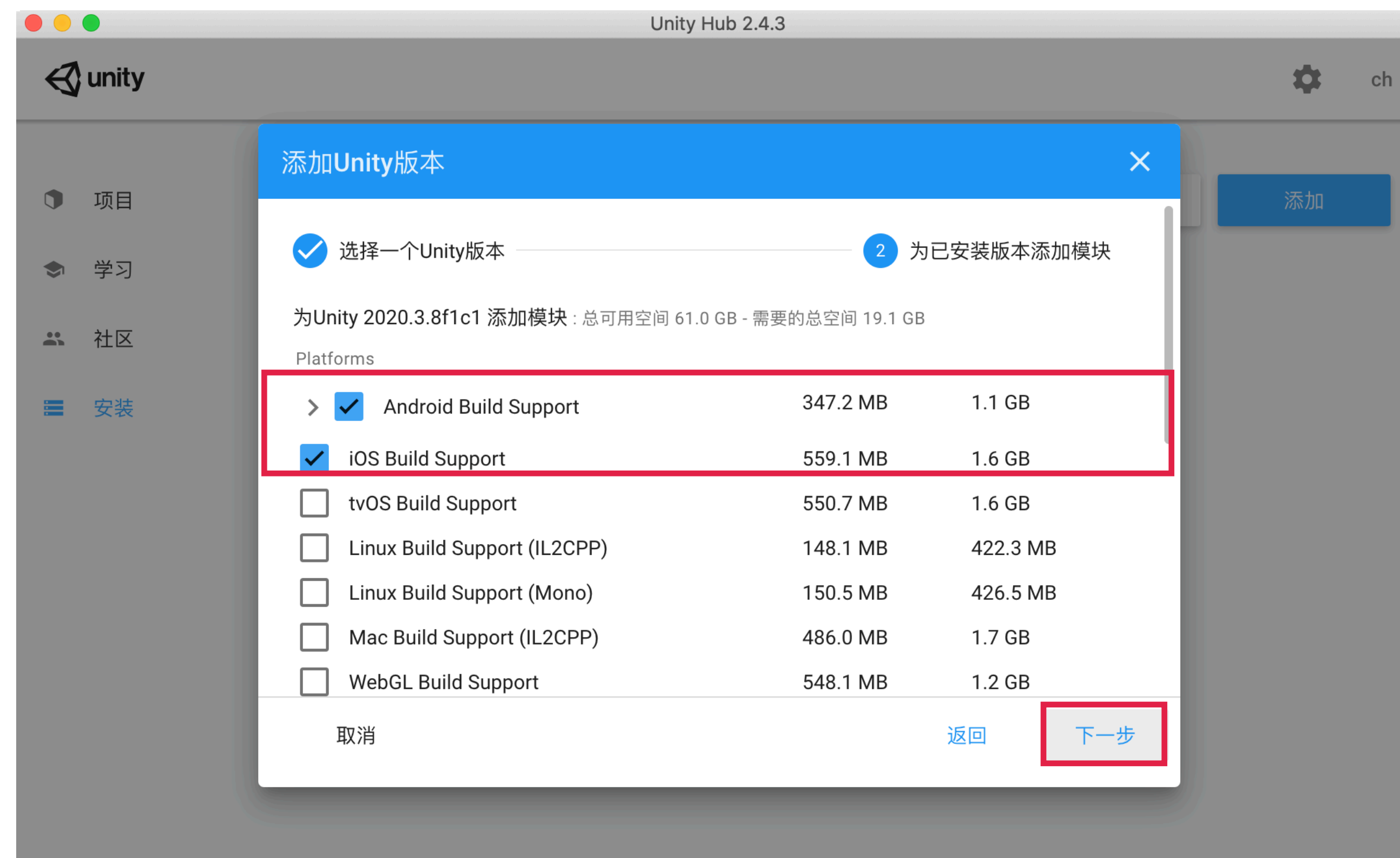
Release notes

移动平台AP

52

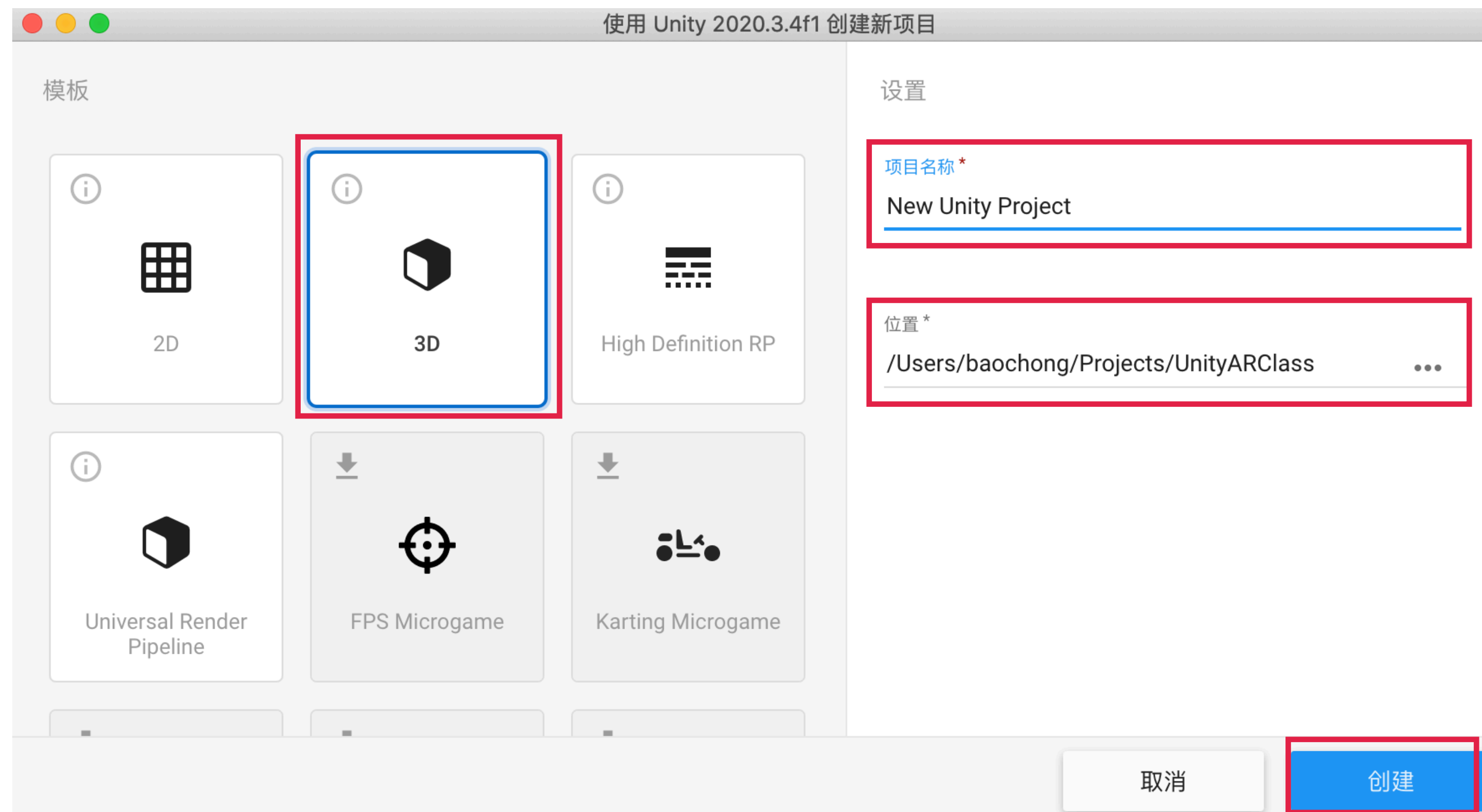
通过Unity Hub安装Unity

- 下载Unity Hub并注册账号和许可证；
- 选择Unity版本和所需要的组件；
- 安装对应组件的SDK，例如需要用于安卓开发，则需要安装Android SDK。



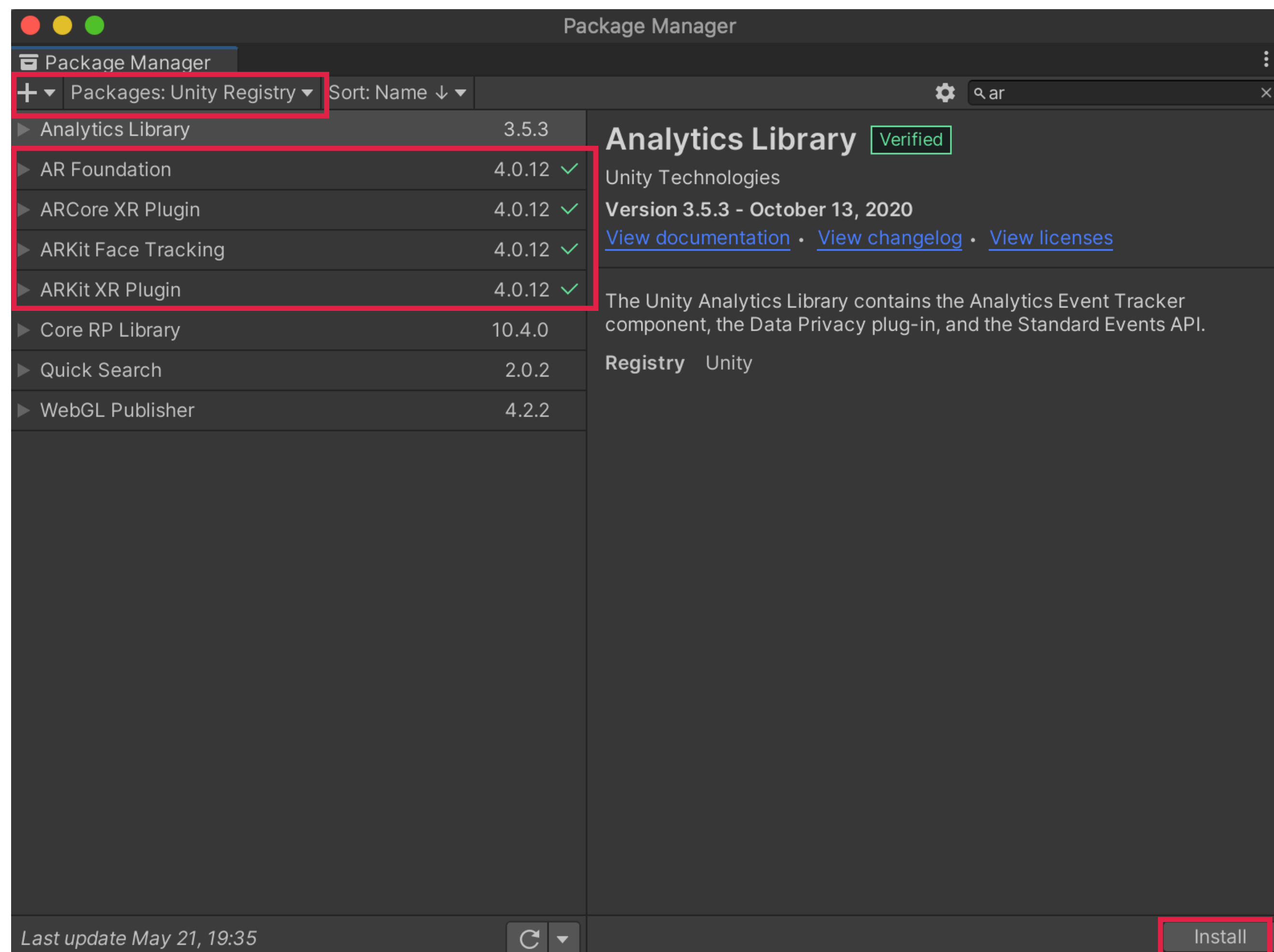
创建项目

- 依次点击项目 -> 新建 -> 3D
- 填写项目名称和位置，并点击创建



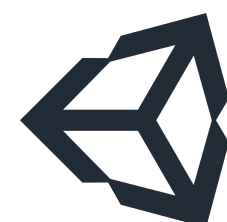
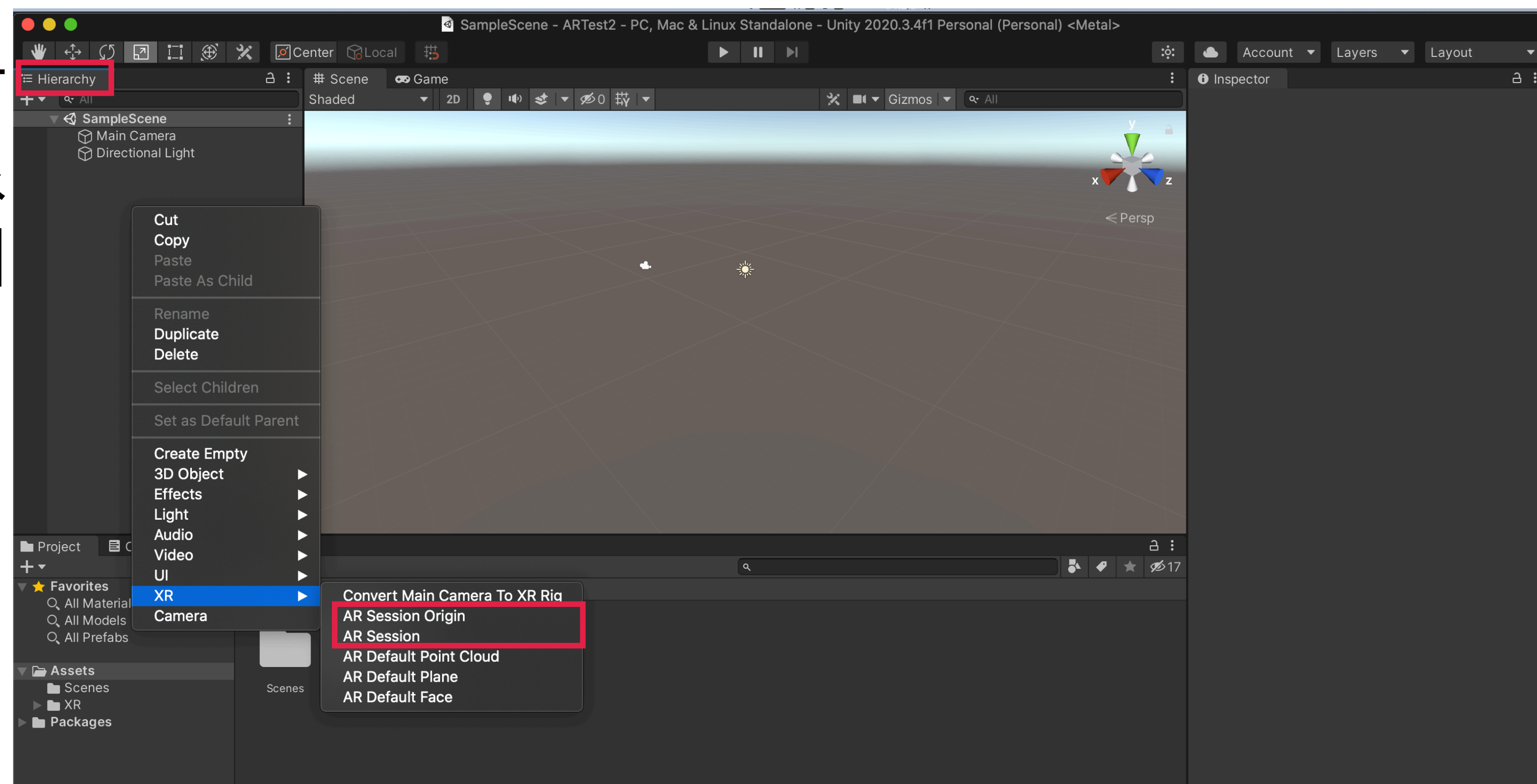
添加AR Foundation

- 依次选择Window -> Package Manager，在弹出框搜索AR Foundation并安装
 - 支持iOS系统的开发，需要安装ARKit XR Plugin；
 - 支持Android系统的开发，需要安装ARCore XR Plugin或者SenseAR XR Plugin（中国增强版）。



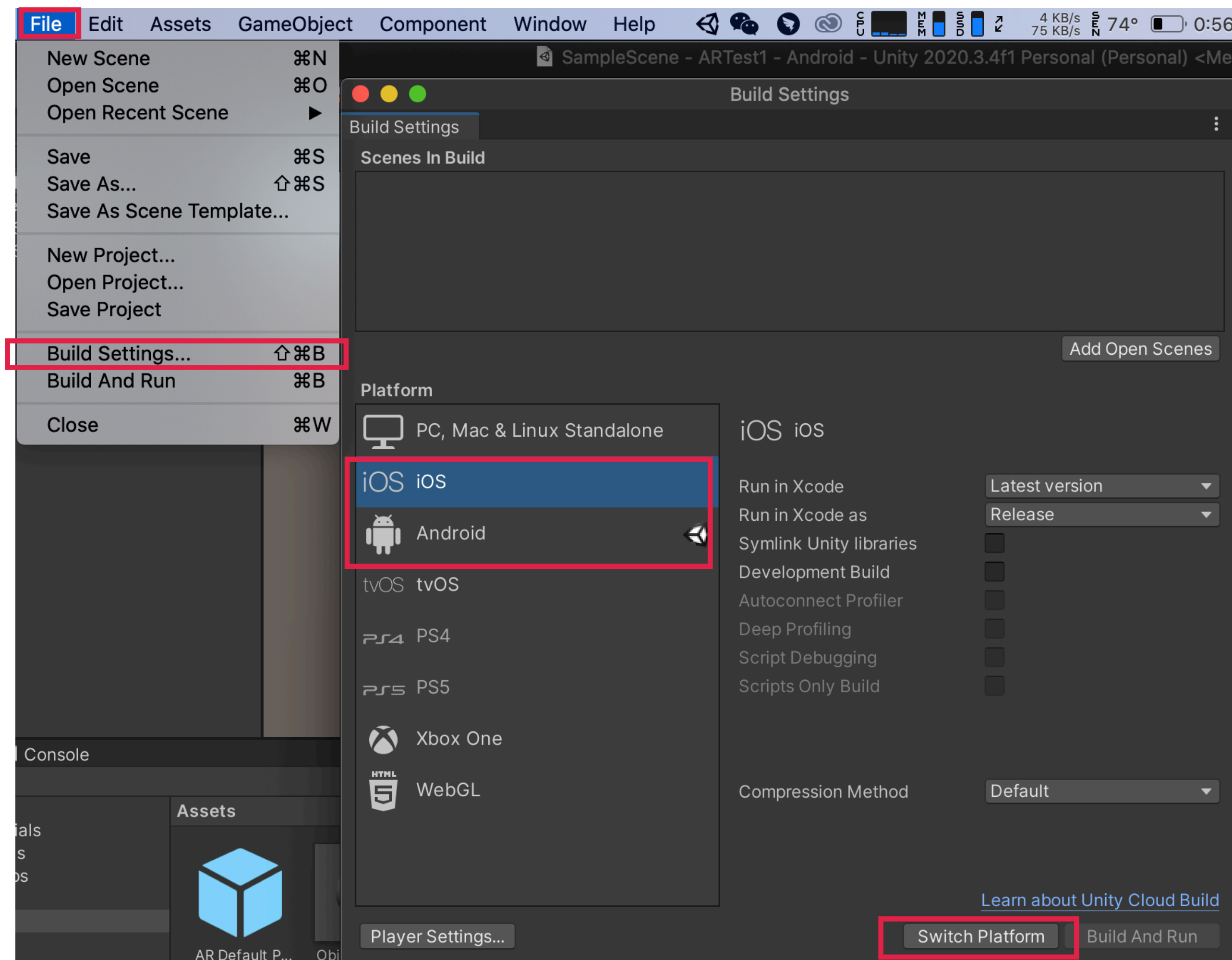
Hierarchy

- Hierarchy 窗口会列出**当前场景中的每个游戏对象**,这是构成游戏大部分内容的自定义对象。在场景中添加和删除对象时,这些对象也会在层级视图中相应显示和消失。
- 安装完成后在Hierarchy界面中右键单击,可以在 XR 选项中找到**ARSession**、**ARSession Origin**、**ARDefault Plane**、**ARDefault Point Cloud** 等。



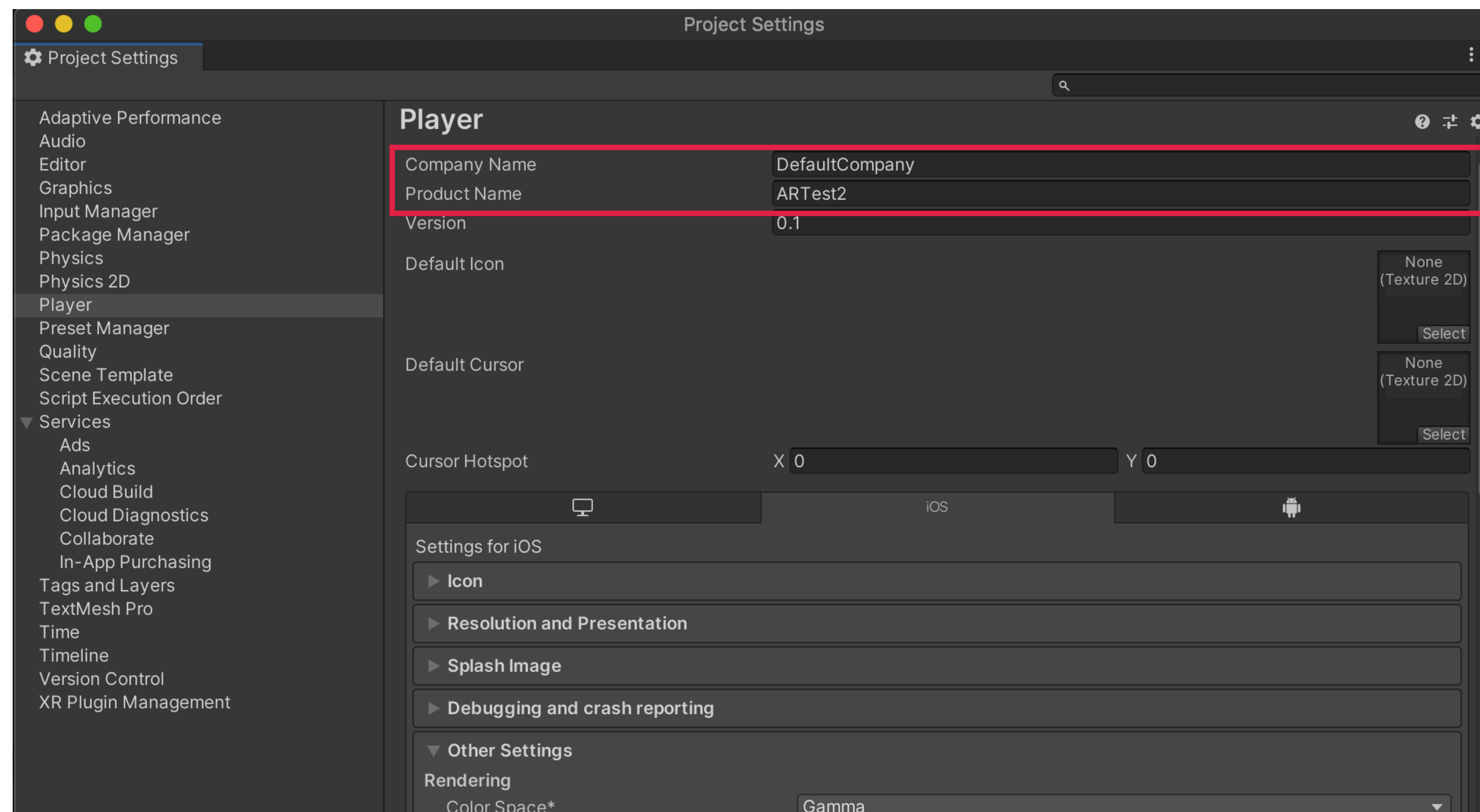
多平台架构

- 依次点击File -> Build Settings;
- 选择目标平台Android/iOS, 点击Switch Platform, Unity 将会切换到对应的平台构建项目;
- 在Player Settings中选择一些平台相关的属性。



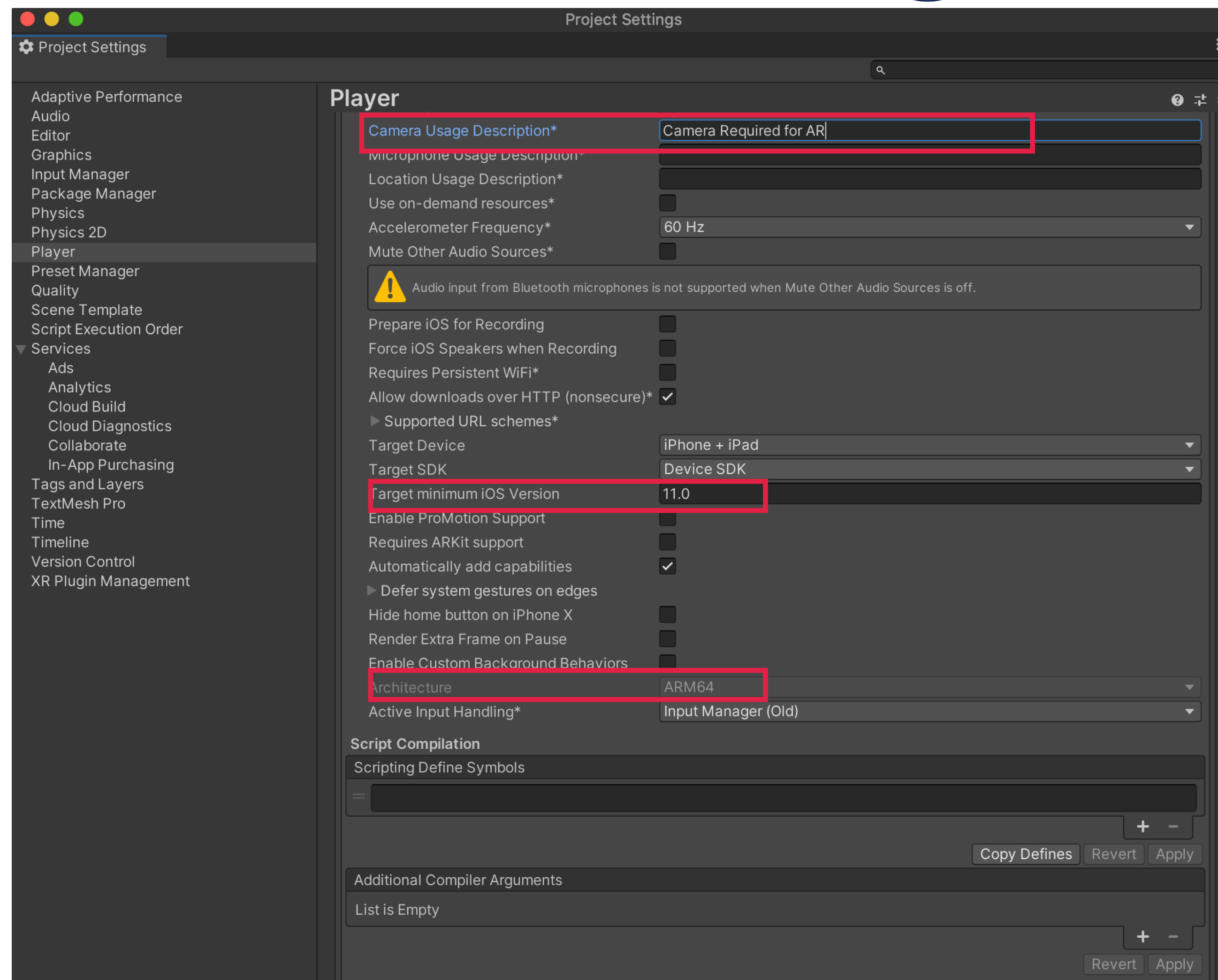
Play Settings(iOS,Android)

- Company Name, Product Name保持默认。两个设置影响生成的移动应用的文件名
`com.[Company Name].[Product Name]`
 - iOS:Bundle Identifier
 - Android: Package Name



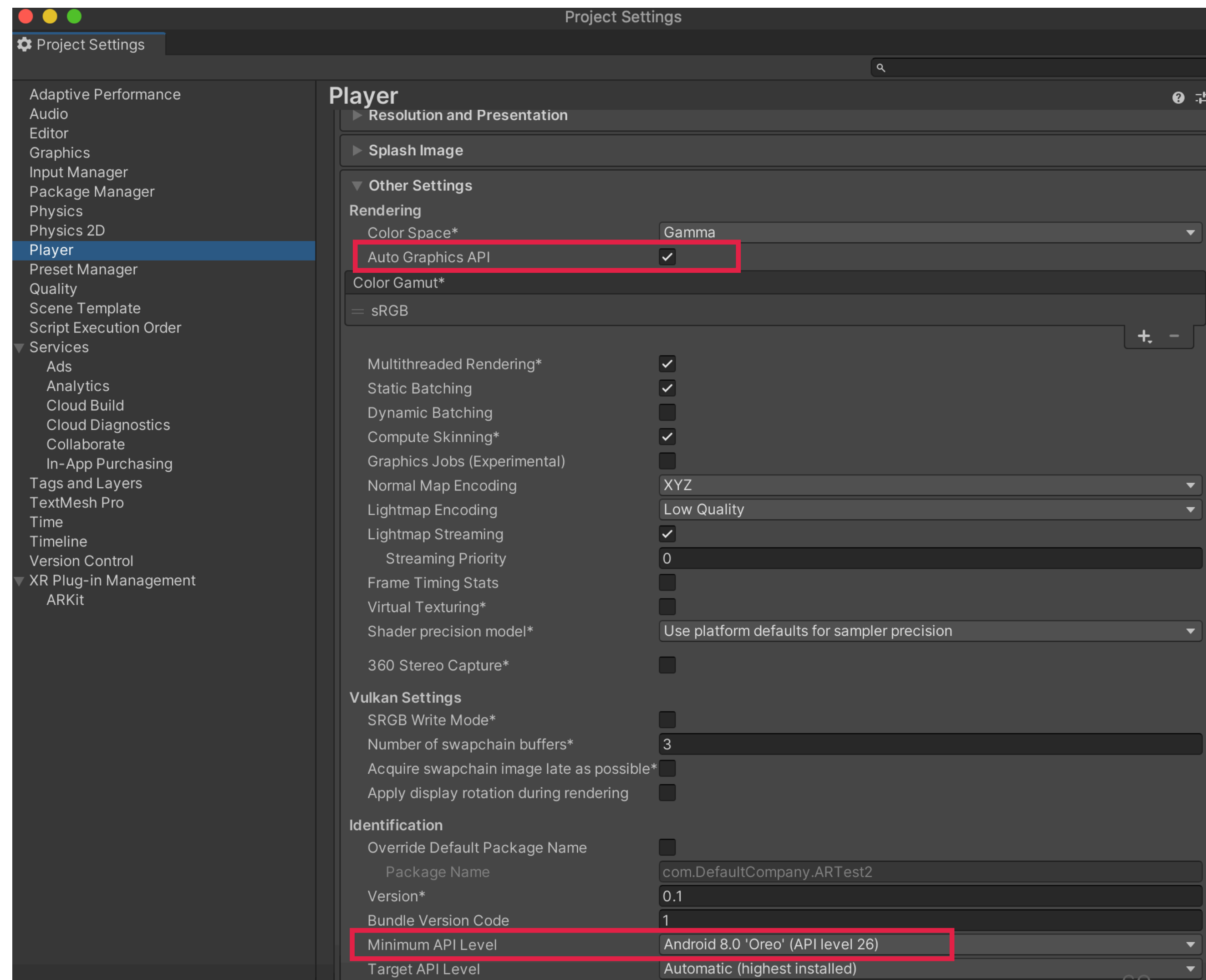
Play Setting(iOS)

- iOS平台需要在**Camera Usage Description**输入一些字符，提示用户数据是怎样被使用的，这样才能通过iOS的NSCameraUsageDescription检查。
- iOS平台需要**11.0**以上才能支持ARKit；
- iOS平台需要在Player Settings中将架构设置为**ARM64**；



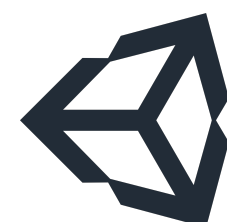
Player Setting(Android)

- Android平台需要API level 26 即 Android 8.0以上的系统才支持ARCore和SenseAR;
- Android平台不支持Vulkan的API和多线程渲染 (Multithreaded Rendering) , 需要在Player Settings中禁用, 或设置**Auto Graphic API**
- Unity可以直接连接Android设备进行调试。



Subsystems

- AR Foundation是建立在subsystems上的， subsystem用于显示不同类型的信息， 是与平台无关的接口， 与AR相关的subsystem在AR Subsystems包中定义， 并使用命名空间 `UnityEngine.XR.ARSubsystems`。
- 每一个subsystem会实现一些特别的功能
 - 例如， 平面检测功能接口是由XRPlaneSubsystem提供的。

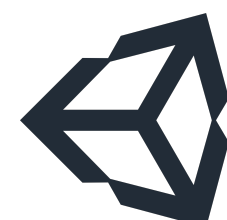


Providers

- Provider是subsystem的**具体实现**。例如，ARCore XR Plugin包含了許多AR Subsystems的ARCore实现。
- 因为不同的subsystem对特定的特性有不同的支持，所以每个subsystem也有一个描述符来指示它支持哪些特定的subsystem特性。
 - 例如，XRPlaneSubsystemDescriptor包含指示是否支持水平或垂直平面检测的属性。
- Provider决定他们将如何实现每个subsystem，通常是对系统**原生的SDK进行封装**（例如，iOS上的ARKit和Android上的ARCore）。



- AR场景包含ARSession组件，ARSession通过在目标平台上启用或禁用AR来控制着AR应用的**生命周期**。
- 如果禁用ARSession，则系统将不再跟踪环境中的特征，但是如果稍后启用了该功能，则系统将尝试恢复并维护以前检测到的特征。
- 注意： ARSession是全局构造，因此ARSession组件可以管理全局Session，如果创建多个AR Session组件，则它们会管理相同的全局Session。



设备支持检测

- 一些平台可能支持有限的设备类型。你的程序需要检测设备是否对AR Foundation提供支持，以便在不支持AR时提供替代的方案。
- ARSession组件具有一个静态协程，可用于确定运行时是否支持AR。
- 右边的代码段展示了简单的设备检测功能。

```
public class MyComponent {  
    [SerializeField] ARSession m_Session;  
  
    IEnumerator Start() {  
        if ((ARSession.state == ARSessionState.None) ||  
            (ARSession.state ==  
                ARSessionState.CheckingAvailability))  
        {  
            yield return ARSession.CheckAvailability();  
        }  
  
        if (ARSession.state == ARSessionState.Unsupported)  
        {  
            // Start some fallback experience for  
            // unsupported devices  
        }  
        else  
        {  
            // Start the AR session  
            m_Session.enabled = true;  
        }  
    }  
}
```



Session State

- 要确定Session的当前状态（例如，是否支持该设备，是否安装了AR软件以及该Session是否正常工作），可查看 **ARSession.state**，还可以订阅ARSession.stateChanged事件，当状态变化时得到通知。



Session State

- ARSessionState包括
 - **None**: AR系统尚未初始化, 可用性未知。
 - **Unsupported**: 当前设备不支持AR。
 - **CheckingAvailability**: 系统正在检查当前设备上AR的可用性。
 - **NeedsInstall**: 当前设备支持AR, 但需要额外安装软件。
 - **Installing**: 正在安装AR软件。
 - **Ready**: 支持AR并准备就绪。
 - **SessionInitialized**: AR会话正在初始化 (即正在启动)。这通常意味着AR正在运行, 但尚未收集到足够的有关环境的信息。
 - **SessionTracking**: AR会话正在运行并且正在跟踪 (即设备能够确定其在世界上的位置和方向)。



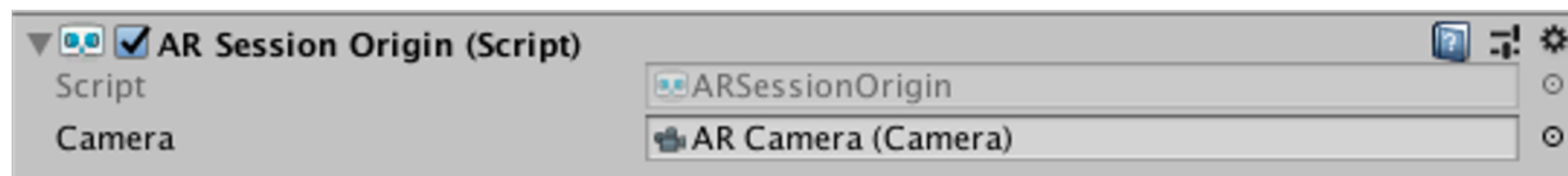
ARSessionOrigin

- ARSessionOrigin的目的是将检测和跟踪到的特征（例如平面和特征点）转换到Unity场景的坐标系下。因为AR设备提供的数据基于其自身的坐标系，而ARSessionOrigin提供了对应的变换（缩放，平移，旋转），将其统一到标准的Unity场景坐标系中。
- 这个概念类似于在Unity使用其他资源时，例如，如果从Maya导入房屋模型时，则门将在相对于建模者原点的位置。这通常称为“模型空间”或“局部空间”。在Unity中实例化它时，相对于Unity的原点，它还将具有一个世界空间。



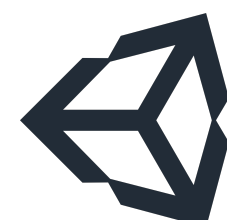
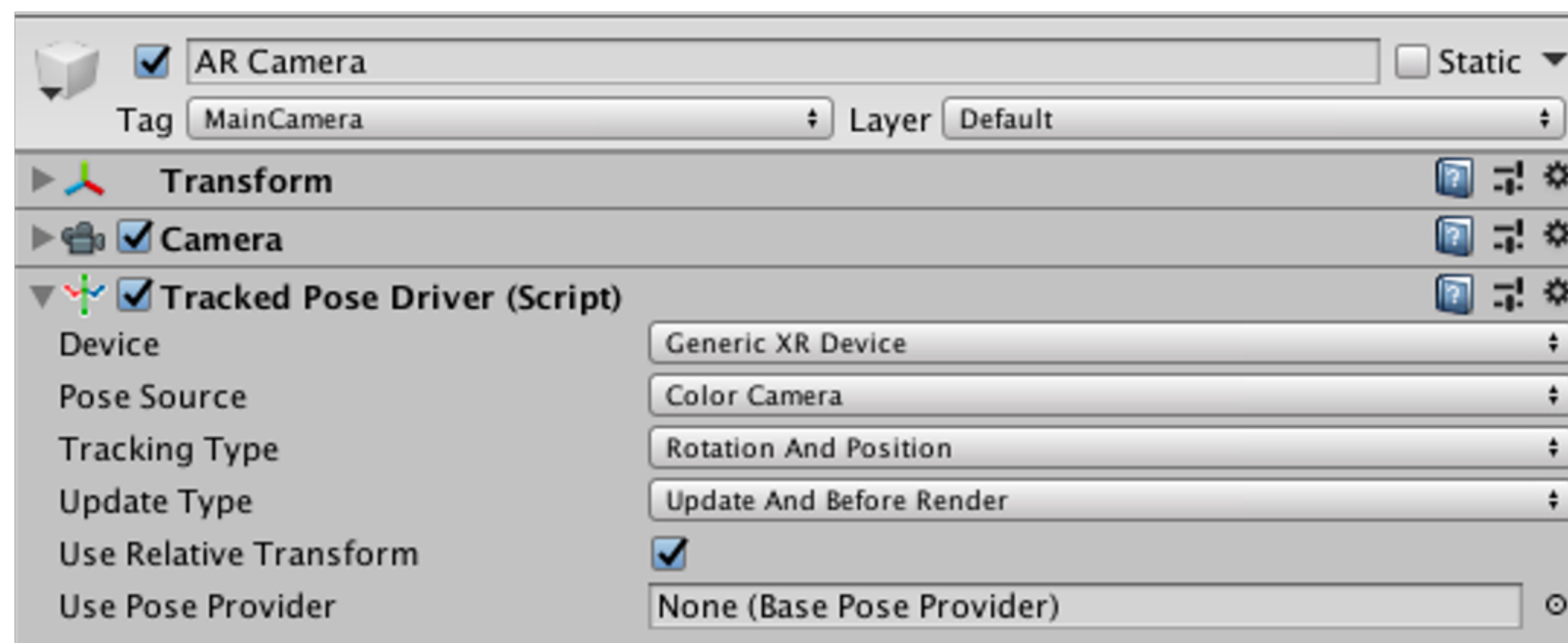
ARSessionOrigin

- ARSessionOrigin还允许变换到AR Camera坐标系上，此时AR Camera应该声明为ARSessionOrigin的子类。由于Camera是由Session驱动的，因此Camera和所检测跟踪的对象可以一起移动



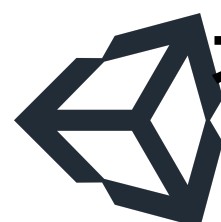
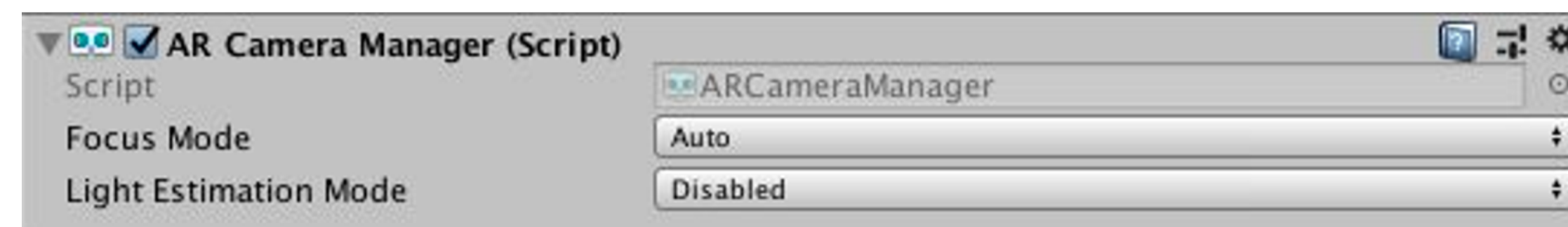
Tracked Pose Driver

- **AR Camera**用于可视化跟踪的物体。由于AR Camera应当与实际的设备保持**运动的一致**，需要在AR Camera上设置**TrackedPoseDriver**组件，该组件将根据设备的跟踪信息来驱动AR Camera的移动和旋转。



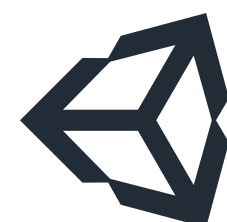
AR Camera manager

- **ARCameraManager**控制AR Camera的部分功能，例如**设备相机获取的纹理**以及**光照估计模式**等。
- 设置的功能包括：
 - **对焦模式**： 可设置为**Auto**（自动变焦）或**Fixed**（固定焦距）。Auto启用硬件相机的自动对焦模式，而Fixed则禁用该功能。
 - **光照估计**： 可以是Disabled（禁用）或Ambient intensity（环境光）。如果未禁用，它将估计物理环境中的平均光强度和颜色。这可能会影响性能，因此如果不使用它，请禁用它。



AR Camera Background

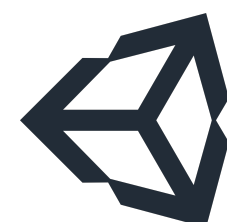
- 如果要~~将设备相机获取的视频作为纹理渲染到背景~~中，则需要添加Camera中ARCameraBackground组件，否则背景将来源于Camera.clearFlags的预设值。
- 该Use Custom Material属性是可选的，通常不需要设置它，因为ARCore和ARKit都提供了自己的着色器用于背景渲染。如果选中了Use Custom Material，则需要自己提供渲染材质。



AR Camera Background

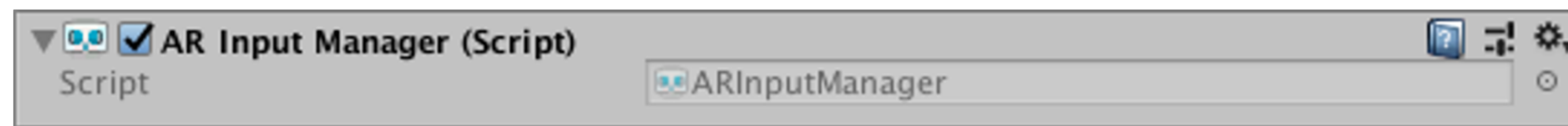
- 相机的纹理显示通常来源于外部，例如设备相机捕获，因此不会持续超过一帧，如果想要进一步对该纹理进行处理，则需要将其**拷贝到渲染纹理**中，可使用右示代码完成该操作：

```
Graphics.Blit(null,  
               m_MyRenderTexture,  
               m_ARBackgroundCamera.material);
```



AR Input Manager

- AR Input Manager组件用于启动**跟踪功能**，否则无法获取设备位置信息，该组件可以放置在场景中的任意位置，但只能存在一个。



Trackable managers

- 在AR Foundation中，**Trackable**指的是真实世界中可以被检测和跟踪的物体，例如：平面，点云，锚点，图像等。
- 每一个Trackable都有对应的manager，但每个manager必须都放置在**同一个AR Session Origin**中，否则无法保证坐标系的一致性

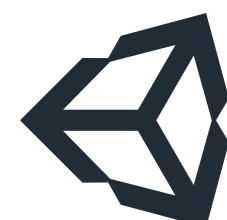
Trackable Manager	Trackable	Purpose
ARPlaneManager	ARPlane	Detects flat surfaces.
ARPointCloudManager	ARPointCloud	Detects feature points.
ARAnchorManager	ARAnchor	Manages anchors. You can manually add and remove them with <code>ARAnchorManager.AddAnchor</code> and <code>ARAnchorManager.RemoveAnchor</code> .
ARRaycastManager	ARRaycast	Repeats and updates a raycast automatically.
ARTrackedImageManager	ARTrackedImage	Detects and tracks 2D images.
AREnvironmentProbeManager	AREnvironmentProbe	Creates cubemaps that represent the environment.
ARFaceManager	ARFace	Detects and tracks human faces.
ARTrackedObjectManager	ARTrackedObject	Detects 3D objects.
ARParticipantManager	ARParticipant	Tracks other users in a multi-user collaborative session.



Trackable的访问

- 通过manager可以遍历其中包含的所有trackable成员，**manager.trackables**属性返回了**TrackableCollection**，可以用foreach遍历，若想获取特定的Trackable，可以用**TryGetTrackable**方法实现。

```
var planeManager = GetComponent<ARPlaneManager>();  
foreach (ARPlane plane in planeManager.trackables)  
{  
    // Do something with the ARPlane  
}
```



添加删除Trackable



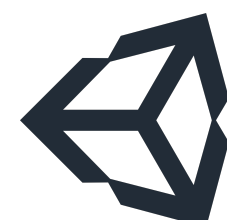
- 有些Trackable可以手动**添加和删除**，例如锚点，有些则是自动添加删除的，例如平面，特征点。
- **不要尝试直接Destroy某个Trackable**或它的GameObject，要通过它的manager完成此类操作。
- 在手动添加或删除某个Trackable后，可能需要subsystem处理完该事件后才会有显式的反应，在此期间可通过检查该Trackable是否在pending状态来判断是否在处理。



添加删除Trackable

- 实际处理添加删除的时间取决于实现，例如添加锚点之后，程序将在下一帧才会对其进行处理。

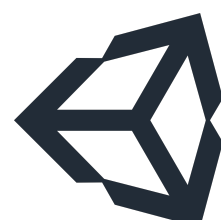
```
var anchor = AnchorManager.AddAnchor(  
    new Pose(position, rotation));  
Debug.Log(anchor.pending); // "true"  
  
// -- next frame --  
void OnAnchorsChanged(ARAnchorsChangedEventArgs eventArgs)  
{  
    foreach (var anchor in eventArgs.added)  
    {  
        // anchor added above now appears in this list.  
    }  
}
```



停用Trackable

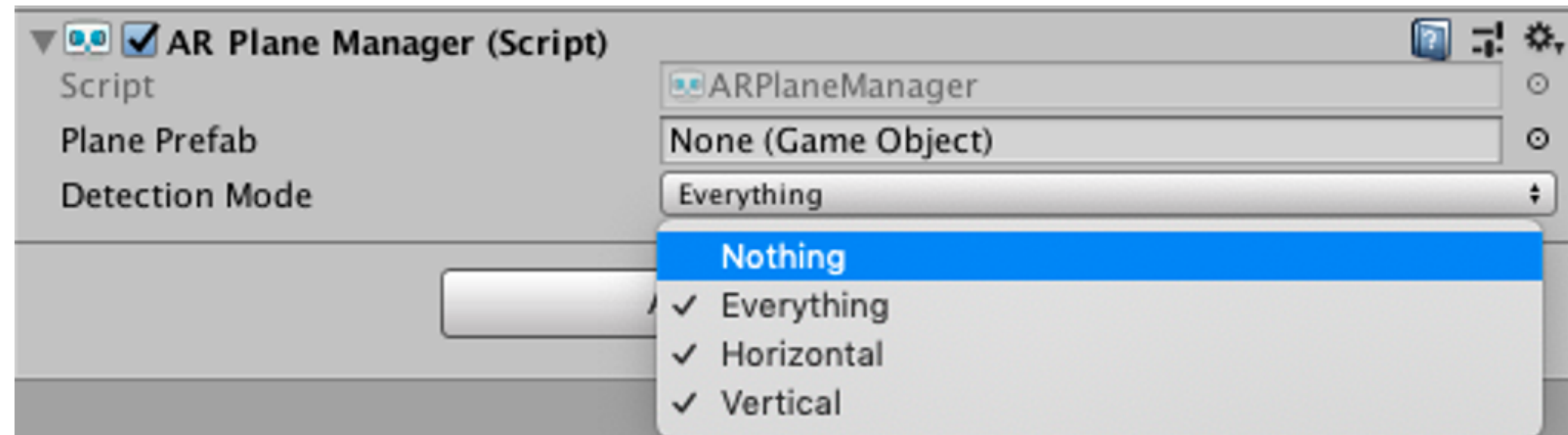
- 有时你并不想删除某个Trackable，只是不想Manager对它有进一步的操作，例如停止对某个平面做渲染，则可以通过改变active属性停用它，如下所示。

```
var planeManager = GetComponent<ARPlaneManager>();  
foreach (var plane in planeManager.trackables)  
{  
    plane.gameObject.SetActive(false);  
}
```



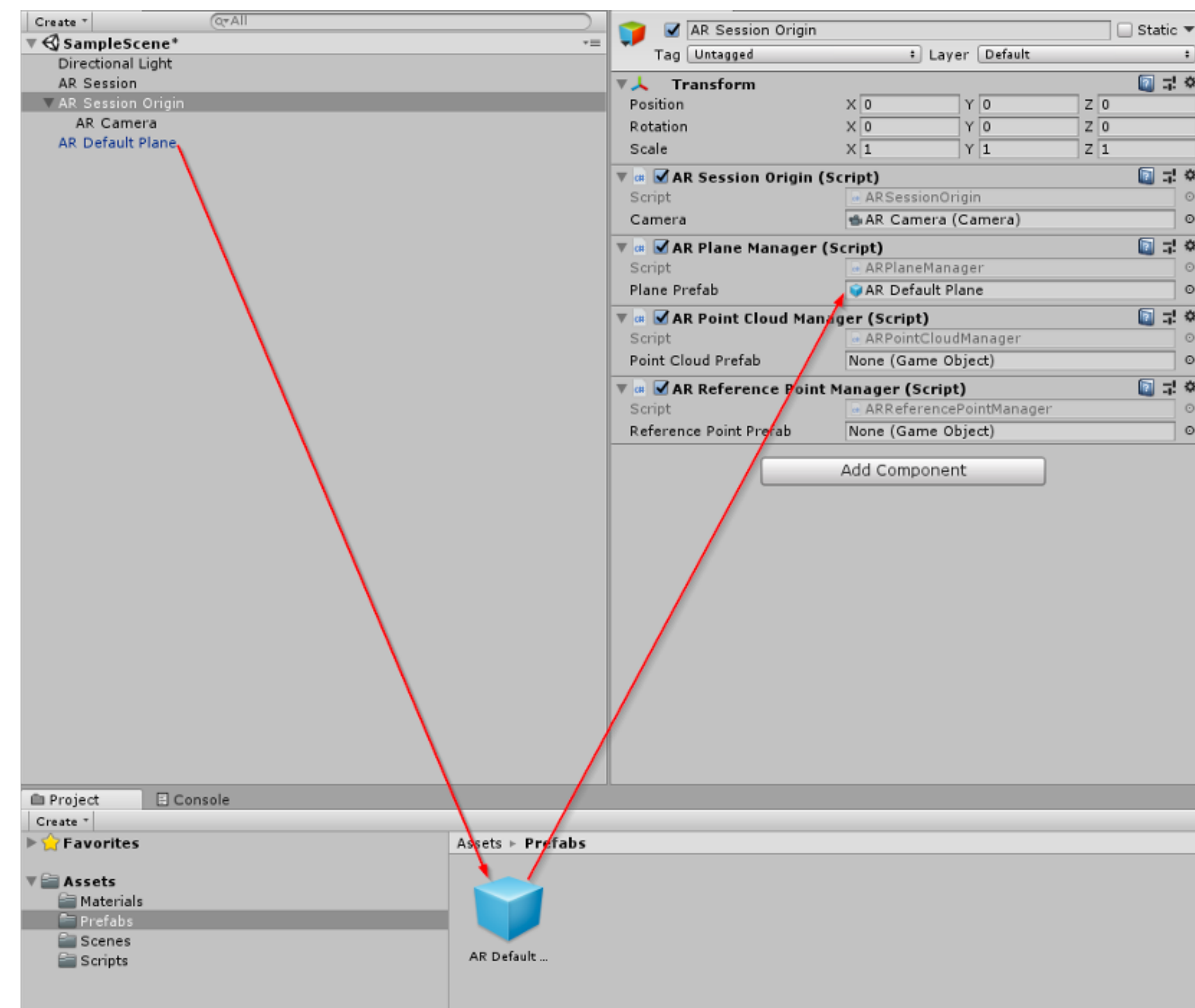
AR plane manager

- AR plane manager为检测到的平面创建GameObject。
- 平面的属性包括位置，纬度和轮廓点，注意平面一定是凸多边形。
- Manager中可以通过Detection Mode设置检测水平面，垂直面或者两者都检测



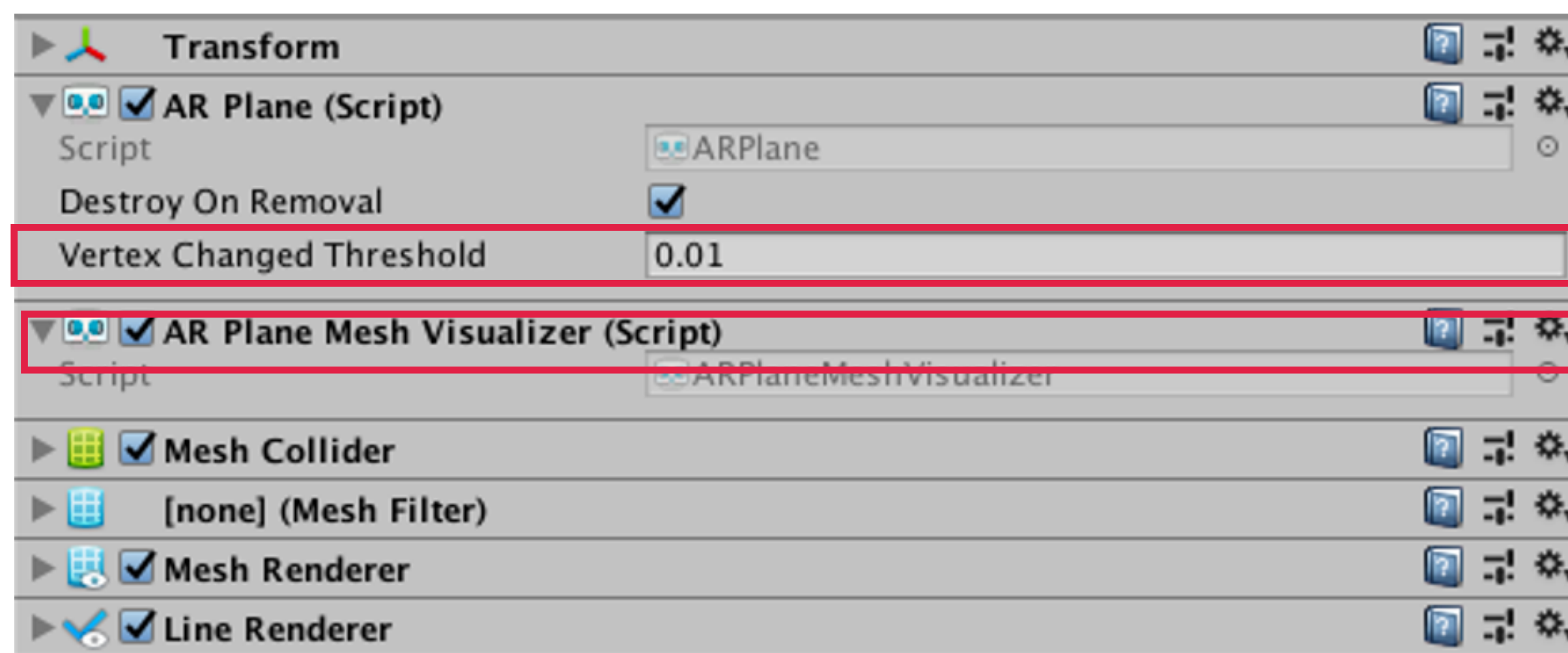
AR plane manager

- 为检测到的平面添加AR效果
 - 在Hierarchy界面中右键XR→AR Default Plane
 - 将AR Default Plane拉至Asset文件夹下，形成prefab
 - 将 Asset文件夹下的平面prefab添加到AR Plane manager的 Plane Prefab中
- 如果AR Plane Manager 检测到平面就会显示Plane Prefab中的效果。



AR plane manager

- 平面可以被添加，更新和删除，在某一帧中平面被添加或者删除时会触发planesChanged事件，而更新的轮廓点超过预设的阈值时会触发boundaryChanged事件。
- AR Foundation提供了ARPlaneMeshVisualizer组件用于从轮廓点生成网格结构，并将它放入MeshCollider, MeshFilter, LineRenderer等组件中，实现碰撞，渲染等效果。



QUESTIONS?