目录

[一些说明 1](#_Toc39064490)

[内置管线转urp管线配置 2](#_Toc39064491)

[1包管理中安装Universal RP 2](#_Toc39064492)

[2 Create/Rendering/Universal Render Pipeline/Pipeline Asset 2](#_Toc39064493)

[3 把这个创建出的UniversalRenderPipelineAsset 2](#_Toc39064494)

[4 buildIn shader升级到urp的设置 3](#_Toc39064495)

[Urp渲染（管线） 5](#_Toc39064496)

[Universal Additional Light Data 6](#_Toc39064497)

[Camera 6](#_Toc39064498)

[The Universal Additional Camera Data component 6](#_Toc39064499)

[Render Type 6](#_Toc39064500)

[清屏，渲染顺序和overdraw 8](#_Toc39064501)

[BaseCamera 8](#_Toc39064502)

[Overlay Camera 8](#_Toc39064503)

[Camera culling and rendering order 9](#_Toc39064504)

[Camera的一些设置 10](#_Toc39064505)

[Stop NaN : 10](#_Toc39064506)

[Dithering: 10](#_Toc39064507)

[Priority: 10](#_Toc39064508)

[Opaque Texture: 10](#_Toc39064509)

[Depth Textuer: 10](#_Toc39064510)

[Volume mask: 10](#_Toc39064511)

[Volume Trigger： 10](#_Toc39064512)

[Post-processing 11](#_Toc39064513)

[Volumes 11](#_Toc39064514)

[FAQ 13](#_Toc39064515)

[URP7.3.1目前对旧版Build in 功能的支持 14](#_Toc39064516)

# 一些说明

使用的是2019.3的urp

Urp预计在2020才会支持自定义后处理

包里有个post processing version2，如果要用默认后处理，要先把这个移除掉

# 内置管线转urp管线配置

## 1包管理中安装Universal RP

## 2 Create/Rendering/Universal Render Pipeline/Pipeline Asset

这个pipelineAsset的作用:

通用渲染管线资产控制着项目的一些全局渲染和质量设置，并创建渲染管线实例

这个pipelineAsset继承自RenderPipelineAsset类

Ninfo1:

注意这里会创建出两个文件

UniversalRenderPipelineAsset

UniversalRenderPipelineAsset\_Renderer

猜测这两个就对应srp中创建的两个脚本的功能

Ninfo2:

UniversalRenderPipelineAsset\_Renderer中Post Process Data默认是null的,而且默认无法浏览找到任何可用对象

看了下正常urp的配置，这个post Process Data是来自

Packages/Universal RP/Runtime/Data/PostProcessData

## 3 把这个创建出的UniversalRenderPipelineAsset

拖动到Editor/Project Setting/Graphics

Ninfo:

注意一点，有时候也要配置到Editor/Project Setting/Quatity，比如ar foundation使用urp时就要配置这个

## 4 buildIn shader升级到urp的设置

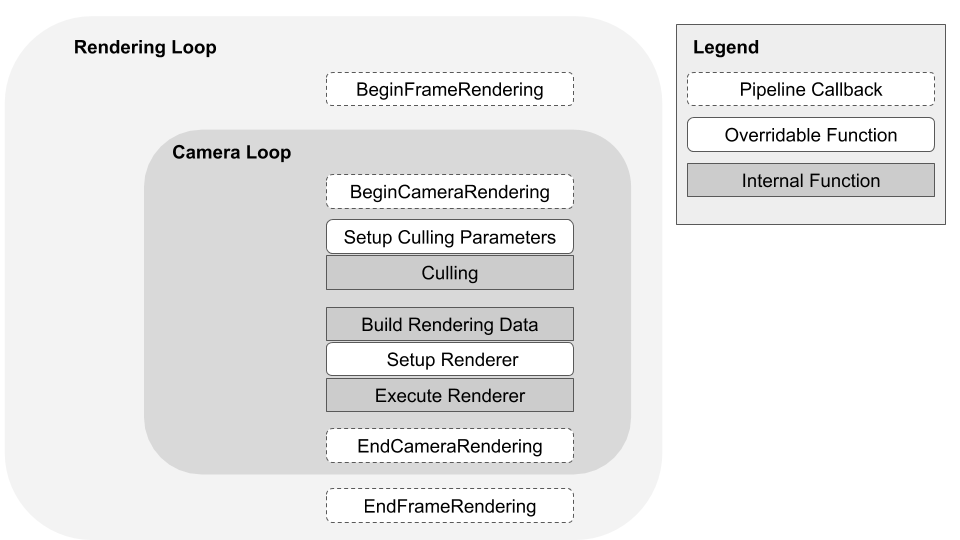
**Edit > Render Pipeline > Universal Render Pipeline/Upgrade Project Materials to URP Materials**

**Edit > Render Pipeline > Universal Render Pipeline/** **Upgrade Selected Materials to URP Materials.**

Build in shader升级对照表

| **Unity built-in shader** | **Universal Render Pipeline shader** |
| --- | --- |
| Standard | Universal Render Pipeline/Lit |
| Standard (Specular Setup) | Universal Render Pipeline/Lit |
| Standard Terrain | Universal Render Pipeline/Terrain/Lit |
| Particles/Standard Surface | Universal Render Pipeline/Particles/Lit |
| Particles/Standard Unlit | Universal Render Pipeline/Particles/Unlit |
| Mobile/Diffuse | Universal Render Pipeline/Simple Lit |
| Mobile/Bumped Specular | Universal Render Pipeline/Simple Lit |
| Mobile/Bumped Specular(1 Directional Light) | Universal Render Pipeline/Simple Lit |
| Mobile/Unlit (Supports Lightmap) | Universal Render Pipeline/Simple Lit |
| Mobile/VertexLit | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Bumped Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Bumped Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Self-Illumin/Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Self-Illumin/Bumped Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Self-Illumin/Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Self-Illumin/Bumped Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Bumped Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Bumped Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Cutout/Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Cutout/Specular | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Cutout/Bumped Diffuse | Universal Render Pipeline/Simple Lit |
| Legacy Shaders/Transparent/Cutout/Bumped Specular | Universal Render Pipeline/Simple Lit |

# Urp渲染（管线）



Urp在循环流程中嵌入了一些回调，就可以修改里面的一些内容

# [Universal Additional Light Data](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/api/UnityEngine.Rendering.Universal.UniversalAdditionalCameraData.html)

切换到urp后，只要想go上添加light就会自动添加这个组件, 这个urp用来存储内部数据

这个组件urp用来扩展和复写unity 标准光照组件

# Camera

## The Universal Additional Camera Data component

作用与上面[Universal Additional Light Data](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/api/UnityEngine.Rendering.Universal.UniversalAdditionalCameraData.html)类似

## Render Type

这里有两种类型

BaseCamera:绘制图像到屏幕或者RenderTexture

OverlayCamera：渲染到其他相机输出之上,同一个OverlayCamera可以被添加到多个CameraStack上

CameraStack: 每个BaseCamera上都有一个CameraStack

对应的一个CameraStatck可以有一个BaseCamera和多个OverlayCamera

所有能对BaseCamera做的操作都可以对CameraStack做，比如把CameraStack渲染到RenderTexture,或者对CameraStack使用后处理特效

注意OverlayCamera 和CamearStack只有在使用ForwadRender时才被支持

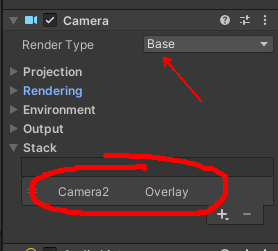
注意，后处理不能用于overlay camera，只能用于base camera或者camera stack

Ninfo这个跟之前的相机叠加类似

不过这回是在RenderType被设置为Base的Camera上，有一个Camera stack

新创建的OverlayCamera只有放到这个CameraStack中才能被显示，

这个CameraStack也可以对多个OverlayCamera进行排序



最后，多个base Camera，以及多个camera stack可以同时存在

通常情况下意义不大，可以用来做切割屏幕的特效，(可以想下监视器多屏幕的效果)

用代码来改变相机viewPort的方法

myUniversalAdditionalCameraData.rect = new Rect(0.5f, 0f, 0.5f, 0f);

注意:cameraStack不支持 2DRenderer和VR MultiPassmode，据说后续版本会支持

使用代码向CameraStack上添加OverrideCamera的方法

var cameraData = camera.GetUniversalAdditionalCameraData();

cameraData.cameraStack.Add(myOverlayCamera);

## 清屏，渲染顺序和overdraw

### BaseCamera

#### Color buffer

Environment/Background Type中有3个选项

Skybox：渲染循环开始前，把color buffer clear成skybox

SolidColor:渲染循环开始前，把color buffer clear 成 solidColor

Uninitialized:这个比较特殊，是不进行clear

Unity描述这个选项说，这种情况下color buffer的内容是由具体平台决定的，有的平台color buffer是前一帧的内容，有的平台会包含unintialized memory(貌似是上次没清理掉的内存)

那么这个uninitialized有什么用呢，上面两种，skybox，solidColor，都会clear一次屏幕，但是uninitialized不会clear，显然效率更高，但是，必须确保camera会渲染整个屏幕的像素(这样才能盖住上一次的渲染结果)

#### Depth buffer

Base Camera每次渲染循环前都会清理depth buffer

### Overlay Camera

#### Color buffer

每次渲染循环开始，overlay camera 接收到的color bufer中，

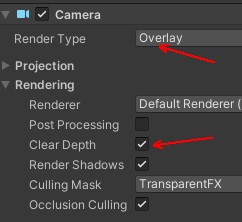
会包含Camera stack中上一个camera的的颜色数据

Overlay camera不会clear color buffer

#### Depth buffer

与上面类似，会拿到camera stack中上一个camera的depth数据

但是overlay camera可以通过选项选择是否clear Depth



## Camera culling and rendering order

多相机情况，unity的渲染按一定顺序执行

每帧unity执行如下操作 ：

1. 获取场景中所有激活的BaseCamera
2. Unity会把激活的BaceCamera组织到2个group中，1一个是渲染到render texture的，另一个组是渲染到屏幕的
3. Unity对渲染到render texture的base camera按优先级进行排序，以便优先级高的camera会在最后渲染
4. 对于每个渲染到render texture的base camera，unity会执行如下步骤

1 cull掉base camera

2 把base camera渲染到render texture

3 对于每个在按顺序在camera stack中的overlay camera

1 cull掉overlay camera

2 把overlay camera渲染到render texture

1. Unity 按优先级排序渲染到scene 的base camera
2. 对每个base camera执行下面步骤

1 cull base camera

2 渲染base camera到屏幕

3 对于每个camera stack中的override camera

1. cull overlay camera
2. 渲染overlay camera到sceen

注意:overlay相关暂不支持2d render

## Camera的一些设置

### Stop NaN :

默认不开启

启用此复选框，使此摄像机将非数字(NaN)的值替换为黑色像素。这阻止了某些效果的破坏，但这是一个资源密集型的过程。只有当您遇到无法修复的NaN问题时，才启用此功能。

Ninfo：个人理解就是把没像素值的位置替换为黑色

### Dithering:

抖动，默认不开启

开启这个，会对最后渲染的图像做一个8 bit的抖动，可以帮助减少颜色渐变区和低光照区的条纹效果。

### Priority:

渲染优先级，优先级越高，越靠后渲染

### Opaque Texture:

控制相机是否创建一个CameraOpaqueTexture,这个textue的内容就是相机的render view

### Depth Textuer:

控制相机是否创建深度贴图，里面存的就是深度信息

### Volume mask:

大概是指定使用volume的layer

### Volume Trigger：

设定一个Transform（位置）， volume系统会使用这个位置去处理相机位置

Eg:

可以把这个trm设置为一个第三人称游戏的主角，那么，相机使用volume的后处理和场景设定就以这个trm的位置为准，如果不设定，就以相机自己的trm位置为准

# Post-processing

后处理会耗费frame time，所以用于移动设备

下面特效可能比较合适

* Bloom (with **High Quality Filtering** disabled)
* Chromatic Aberration
* Color Grading
* Lens Distortion
* Vignette

注意：

Depth of field特效

低端设备用 gaussian（高斯） dof

Pc用 bokeh dof

注意：

对于anti-aliasing在移动端，unity推荐 fxaa

注意：

VR中使用后处理会引起🤮

在一些节奏比较快的，或者需要高速运动vr app中，

使用Vignette effect

避免使用Lens Distortion,Chromatic Aberration,Motion Blur

## Volumes

Volume可以是global或者有边界

为了计算最终值，每个Volume包含 urp用来做插值用的 （依赖相机位置的）场景设置数据。

Eg:可以使用local volume改变环境设置，比如雾的颜色和密度，来改变场景中不同区域的环境范围

Volume组件可以添加到任何go上，最好的做法是每个volume添加到一个独立的go上。

Volume本身不包含数据，（用于做插值的）数据是存到Volume Profile上的

Volume Profile包含默认数据，想要修改就要在上面添加Volume override

Volume上mode，weight，priority用于处理与其他Volume的交互方式

Mode:

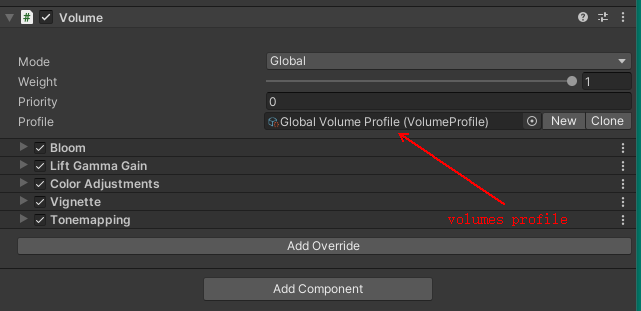
Global: 不管camera在什么位置，Volume都会作用到camera，

Local:选中这个后，需要设置一个collider，如果camera在这个collider之内才会受到影响

Local选中后，会多出一个Blend Distance参数，代表(最大)距离collider多远开始产生效果，如果这个参数是0，则代表，camera进入collider后Volumn 立刻开始对camera 产生影响

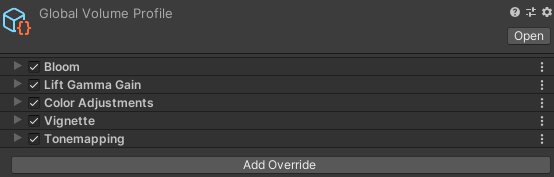
Runtime时，urp会关注所有激活的volumes,检查每个volumes对scene setting的贡献。

Urp具体是使用相机位置和volume参数来计算具体的贡献值

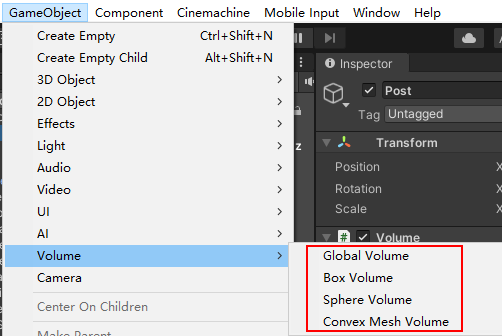


Ninfo:Volume profile就是一个asset，存数据用的,可以看到volume profile中内容与volume中展示的是一致的





GameObject/Volumn里可以直接创建4种Volumn



# FAQ

**Where has Dynamic Batching gone?**

The Dynamic Batching checkbox has moved from the **Player Settings** to the [**URP Asset**](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/manual/universalrp-asset.html).

**How do I enable Double Sided Global Illumination in the Editor?**

In the Material Inspector, find **Render Face**, and select **Both**. This means that both sides of your geometry contribute to global illumination, because URP doesn’t cull either side.

**Does URP support a Deferred Renderer?**

现在不支持，以后会支持

**I’ve upgraded my Project from the Built-in render pipeline to URP, but it’s not running faster. Why?**

Urp与build in qualitySetting不同，build in的设置分散在 Quality setting,Garphics setting,Player setting中，而urp的设置全部存到 urp asset中。

so首先要确定urp asset的设定跟之前build in 的设定是否相同，比如，是否禁用msaa或者hdr。

其次，要确保在渲染上做公平的比较，比如，urp只支持前向渲染，build in也得是前向，才算公平比较

除了前面2条，如果urp还是比build in差，报bug

**My Project takes a long time to build. Is this expected?**

我们正在研究如何更有效的剥离shader keywords,

你可以在urp asset禁用一些不需要的特性，以达到优化shader stripper(剥离器)的作用

更多设置,shader variants,构建时间的问题，看

[shader stripping documentation](https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/manual/shader-stripping.html).

<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/manual/shader-stripping.html>

观与shader stripping相关unity blog的一篇文章

<https://blogs.unity3d.com/2018/05/14/stripping-scriptable-shader-variants/?_ga=2.79395735.2017362426.1587894973-95776593.1586748444>

**What rendering space does URP work in?**

By default, URP uses a linear color space while rendering. You can also use a gamma color space, which is non-linear. To do so, toggle it in the Player Settings.

**扩展 URP scriptable render pass?**

To create a scriptable render pass, you have to create a ScriptableRendererFeature script. This is because the scriptable render feature is a container that can have the pass in it. To create the scriptable render feature in the Editor, click on Asset > Create > Rendering > Universal Render Pipeline > Renderer Feature.

# URP7.3.1与旧版Build in 功能比较

新旧shader升级对比

<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/manual/upgrading-your-shaders.html>

# 关于shader variants，shader stripper，shader keywords

Shader stripping原文

<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.3/manual/shader-stripping.html>

unity 从单一shader文件种编译出多个shader variants.

Shader variants的数量取决于你在shader文件种使用了多少shader keywords

在默认的shader种，unity使用一些keywords,用于lighting与shadows ，urp会排除一些已经在urp asset中特性被激活的shader variants

当你在urp asset中禁用某些特性，渲染管线在build时会剥离与之对应的shader variants,剥离shader中的shader variants会减少shader体积，减少shader 构建时间，如果你的工程从没用过某些特性和keyword，这（种操作）会非常有用

Eg:如果你的工程没使用平行光阴影，如果没有shader stripping,平型光阴影相关的shader variants在build时依然存在，如果你知道根本不会使用这些阴影，可以把urp asset中

主平型光，附加平行光的Cast Shadows勾掉，urp build shader时就会strips（剥离）掉这些shader variants

# 关于urp render feature

官网暂无资料