UniversalRenderPipelineAsset

> URP管线梦开始的地方，继承了**RenderPipelineAsset**，通过重写**CreatePipeline** 方法，创建一个管线实例，去做真正的渲染。
> 该类主要是用来配置各种全局设置，如AA,HDR,SHADOW等。在创建一个Asset时，会同时创建一个**Renderer**。
> 主要看一些关键的函数。

CreatePipeline()

管线的入口函数。

- 在这个方法里，首先会判断**m_RendererDataList** 是否为null，如果为null则创建一个**ScriptableRendererData** 队列。
- 然后清空Asset拥有的**ScriptableRenderer**，**ScriptableRenderer** 基类清理管理的rendererFeature。随后会进入子类清理一些列数据。

```
m_ForwardLights.Cleanup();
m_GBufferPass?.Dispose();
m_PostProcessPasses.Dispose();
m_ColorBufferSystem.Dispose();
m_MainLightShadowCasterPass?.Dispose();
m_AdditionalLightsShadowCasterPass?.Dispose();
m_CameraDepthAttachment?.Release();
m_XRTargetHandleAlias?.Release();
m_DepthTexture?.Release();
m_NormalsTexture?.Release();
m_OpaqueColor?.Release();
m_MotionVectorColor?.Release();
m_MotionVectorDepth?.Release();
CoreUtils.Destroy(m_BlitMaterial);
CoreUtils.Destroy(m_CopyDepthMaterial);
CoreUtils.Destroy(m_SamplingMaterial);
CoreUtils.Destroy(m_StencilDeferredMaterial);
CoreUtils.Destroy(m_CameraMotionVecMaterial);
CoreUtils.Destroy(m_ObjectMotionVecMaterial);
```

> Blitter.Cleanup();
> LensFlareCommonSRP.Dispose();

- 随后创建一个Pipeline实例

```
var pipeline = new UniversalRenderPipeline(this);
```

- 最后创建Renderers。
  有多少个RendererData就创建几个Renderer，创建时调用Renderer的**InternalCreateRenderer()**方法。

一些属性

URP的所有全局属性都在这里了。用到的时候看一下就好。

s_LightCookieFormatListp[][]

这个属性存了URP里大部分默认的贴图格式

Texture Name	Texture size	Texture Format
Grayscale Low	R8_UNorm	8bit
Grayscale High	R16_UNorm	16bit
Color Low	RGB565或RGBA5551	16bit
Color High	A2B10G10R10或RGBA8_SRGB或BGRA8_SRGB	32
Color HDR	B10G11R11_UFloatPack32	32

scriptableRenderer

该属性为保存的Renderer，在获取时，会判断RendererData是否还生效，如果失效会重建一个。返回的被设置为默认的Renderer。