**1 自定义渲染管线**

* 1. **自定义渲染管线**

本系列教程参考了Jsper Flick的《Custom SRP》系列文章，可以去他的网站学习更多技术。

本课程参考书籍《Unity Shader入门精要》和《Unity3D 内建着色器源码剖析》等。

推荐雨松的《URP从原理到应用》教程。

**1.1.1 前置工作**

1. 通过Player Setting将Color Space由Gamma空间切换为Linear线性空间；
2. 从PackageManager下载扩展包；

**1.1.2 新建渲染管线资产**

**1.1.3 创建渲染管线实例**

**1.2 正式渲染**

ScriptableRenderContext和CommandBuffer是SRP用于渲染的最底层接口。

**1.2.1 相机渲染**

**1.2.2 绘制天空盒**

通过调用ScriptableRenderContext渲染接口的DrawSkybox()来绘制一个天空盒。通过ScriptableRenderContext发送的渲染命令都是缓冲的，最后需要通过调用Submit()方法来正式提交渲染命令。

我们想要控制相机需要设置视图-投影变换矩阵，我们通过context.SetupCameraProperties方法来设置矩阵和相机的其他属性。

**1.2.3 CommandBuffer**

某些任务，比如绘制天空盒，可以直接调用context的专用方法发出命令，而其它命令需要通过单独的命令缓冲区（CommandBuffer）间接发出，我们需要这样一个缓冲区来绘制场景中其它几何图形。CommandBuffer是一个容器，它保存了这些将要执行的渲染命令。

执行缓冲区命令是通过context.ExecuteCommandBuffer(buffer)来执行，这个操作会从缓冲区复制命令但不会清除缓冲区，我们如果要重用buffer，一般会在执行完该命令后调用Clear()清除。

**1.2.4 清除渲染目标**

颜色缓冲，深度缓冲，模板缓冲和自定义的缓冲区一起称为帧缓冲。

清除渲染目标是通过调用buffer.ClearRenderTarget方法来完成。

**1.2.5 剔除**

通过camera.TryGetCullingParameters方法得到需要进行剔除检查的所有物体，正式的剔除是通过context.Cull()实现的，最后会返回一个CullingResults的结构，里面存储了我们相机剔除后的所有视野内可见物体的数据信息。

**1.2.6 绘制几何体**

调用context.DrawRenderers方法进行正式绘制。它需要三个参数：CullingResults，DrawingSettings和FilteringSettings。

DrawingSettings用于设置是哪个Shader的哪个Pass进行渲染。

在SRP中，旧的着色器大部分无法使用，但没有光照的内置着色器Unlit被保留了下来，我们需要获取Pass名字为SRPDefaultUnlit的着色器标识ID。

SortSettings的作用是确定相机的透明排序模式是否使用正交或基于距离的排序。

FilteringSettings，用于过滤给定的一组可见对象以便渲染。

**1.2.7 透明和不透明几何分开绘制**

一般情况下，我们应当遵守 不透明物体->绘制天空盒->绘制透明物体 的绘制顺序。

**1.3 编辑器渲染**

**1.3.1 绘制SRP不支持的着色器类型**

在编辑器开发项目的过程中应该把那些不支持的着色器类型给暴露出来进行集中解决。

**1.3.2 动静代码分离：局部类**

局部类在很多项目的开发中比较常用，经常用于分离编辑器中静态编辑的相关代码和运行时动态调用的相关代码。

**1.3.3 绘制Gizmos**

我们通过context.DrawGizmos()来绘制Gizmos辅助线框，它在工程的测试和编辑时是比较有用的，该方法放到Editor脚本中来定义实现。Handles.ShouldRenderGizmos决定是否绘制Gizmos。

**1.3.4 绘制UI**

判断相机是在Scene视图渲染，ScriptableRenderContext.EmitWorldGeometryForSceneView方法将UI发送到Scene视图进行渲染。

**1.4 多摄像机**

**1.4.1 两个摄像机**

游戏场景中的Main Camera深度值默认是-1，若场景中有多个相机，它们的渲染顺序是按深度递增渲染的。

为了区分两个相机渲染的条目，我们在CameraRenderer.Editor脚本中添加一个PrepareBuffer()方法，使用相机的名字去设置命令缓冲区的名字。

**1.4.2 Culling Mask和Clear Flags**

多个相机的渲染效果取决于Clear Flags。

buffer.ClearRenderTarget()的第一个参数代表是否要清除深度缓冲，第二个参数代表是否要清除颜色缓冲，第三个参数设置用于清除缓冲区的颜色值。