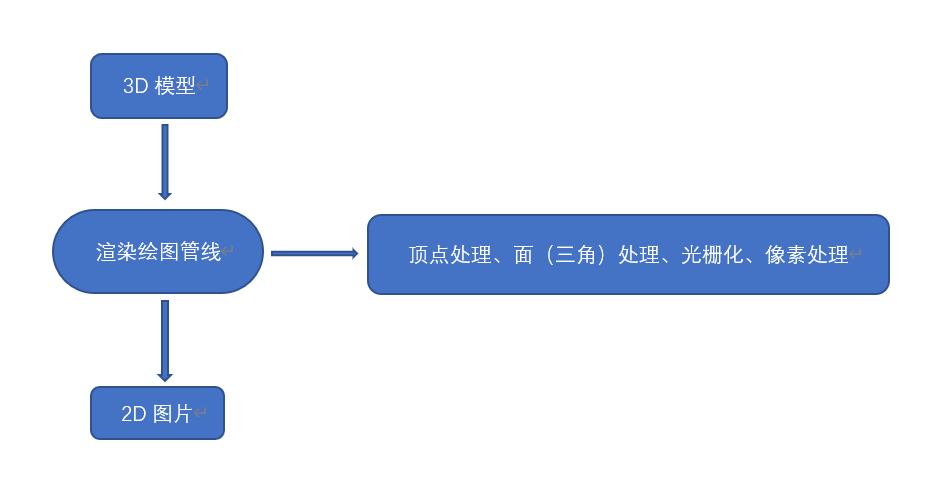
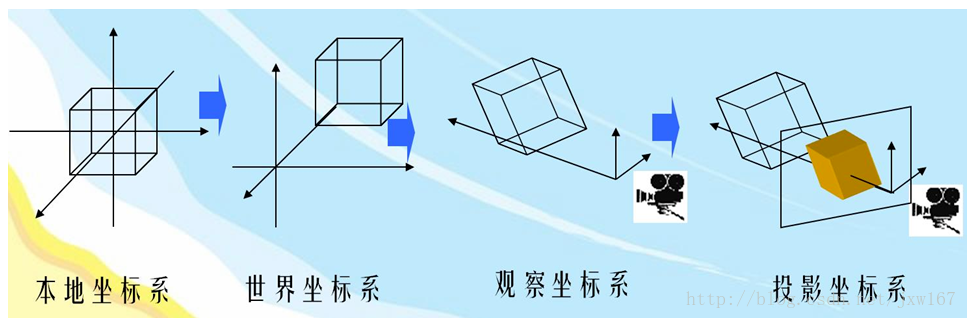
# ShaderGraph的应用，以及why？how？

## 知识点1：渲染管线

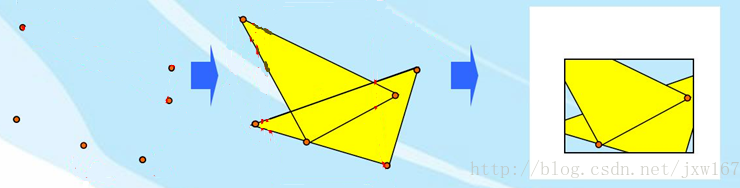
美术3d模型 -> 导入游戏场景渲染 == 把绘制的3D物体通过可编程流水线绘制在2D的屏幕上的过程



1.1、顶点处理（一系列的矩阵变换）



1.2、面处理、光栅化、像素处理

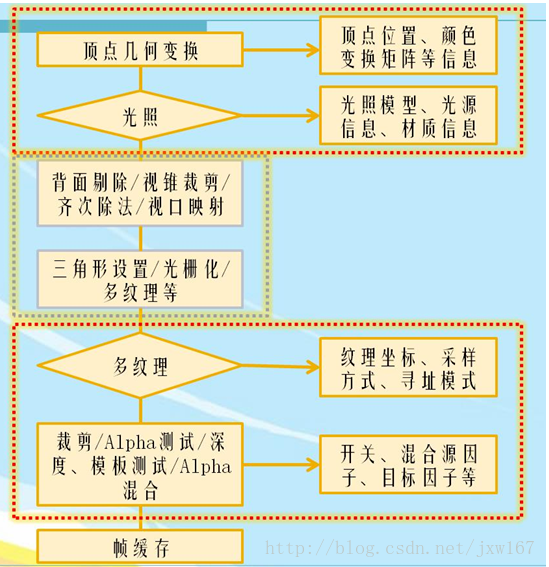


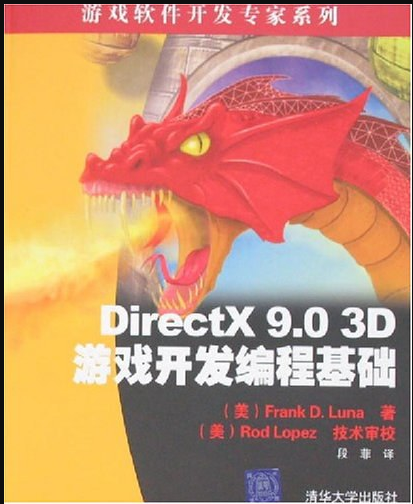
点线成面->面剔除（遮挡、深度）->光栅化（对每个像素区域进行着色，对像素贴上贴图）

所有处理的单元都是gpu，处理结果存放于显存

注：不同教程对渲染管线阶段有些细微的区分，比如：应用程序截断->几何截断->光栅化->像素处理

渲染管线总流程图



偷偷推介一本书（/滑稽.jpg）：

## 知识点2：shader

2.1、着色器（shader）

1、顶点着色器：

顶点渲染的作用是对三维图元的顶点进行坐标变换和光照计算，生成可用于渲染到投影空间的顶点坐标、颜色和纹理坐标。顶点渲染就是定义了一系列针对顶点的渲染指令或渲染语句，当Direct3D处理图元顶点时，自动使用这些渲染指令或者渲染语句对每一个顶点逐一进行处理，完成顶点数据的处理工作。

3D图形都是由一个个三角面片组成的，顶点Shader就是计算每个三角面片上的顶点，并为最终像素渲染做准备

2、像素着色器

对每个像素的颜色进行混合纹理采样，包括迭代颜色和纹理坐标、纹理采样以及将纹理采样与灯光和材质的颜色进行混合。比如：Alpha测试、深度测试、模版测试、计算每个像素的雾化值、Alpha混合等。

顾名思义，就是以像素为单位，计算光照、颜色的一系列算法。 几个不同的图形API都有各自的Shader语言，在DirectX中，顶点shader叫做 Vertex Shader ，像素Shader叫做 Pixel Shader； 在OpenGL中，顶点Shader也叫做 Vertex Shader ，但像素Shader叫做 Fragment Shader，也就是我们常说的片断Shader或者片元Shader。

* 1. 着色器编程语言

1. 基于OpenGL的OpenGL Shading Language，简称GLSL。
2. 基于DirectX的High Level Shading Language,简称HLSL。
3. 还有NVIDIA公司的C for Graphic，简称Cg语言。

2.3、unity 着色器

Unity Shader严格来说并不是传统上的Shader,而是Unity自身封装后的一种便于书写的Shader，又称为ShaderLab。

在Unity中有3种Shader（其实就是三种不同的写法）：

* Surface Shaders 表面着色器(Unity对Vertex/Fragment Shader的又一层包装)
* Vertex/Fragment Shaders 顶点/片断着色器
* Fixed Function Shaders 固定管线着色器，无法编程计算，通过修改参数和配置进行渲染操作（是否想到了shadergraph?!）

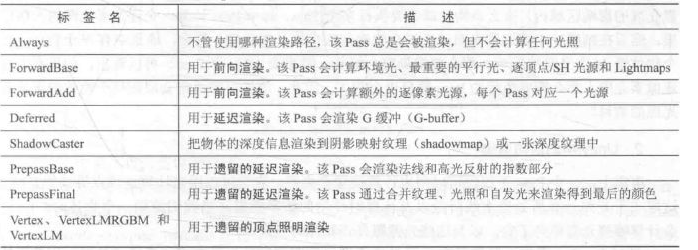
附上一遍简单的shader代码以及注解：

Path：t\_s\_test\Assets\ShaderGraph\_cxh\Shader

## 知识点3：可编程渲染管线SRP(script render pipline)

3.1理由：

上节有讲到unity的固定渲染管线，处理的东西是：顶点处理、面处理、光栅化、像素处理等，but！Unity只支持前向渲染(Forward Renddering Path)和 延迟渲染(Deferred Rendering Path)



其中：

Forward Shading  
原理：每个作用于物体的像素光单独计算一次，drawCall随着物体与光照数量增加而成倍增加  
优点：不受硬件限制  
缺点：光照计算开销成倍增加随着光源和物体数量增加。  
每个物体接受光照数量有限。

Deferred Shading  
原理：物体颜色、法线、材质等信息先渲染到G-Buffer中，光照最后单独渲染，避免每个物体多个光照批次的问题  
优点：作用于每个物体光照数量不再受到限制，光照计算不随着物体增加而增加  
缺点：移动设备需要支持OpenGL3.0。  
不支持MSAA。  
半透明物体仍然使用前向渲染。

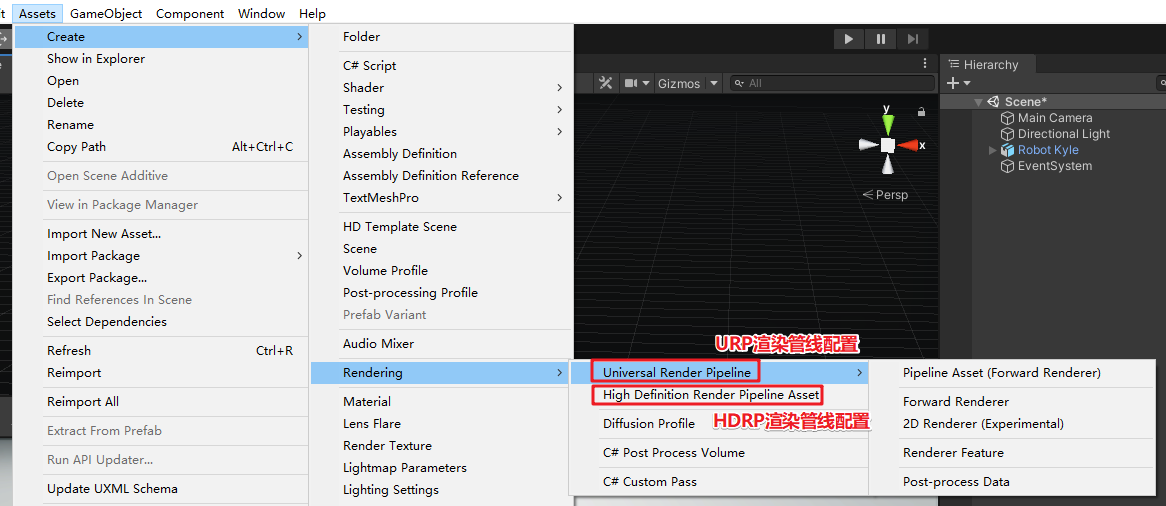
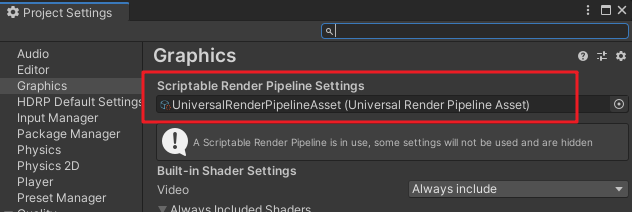
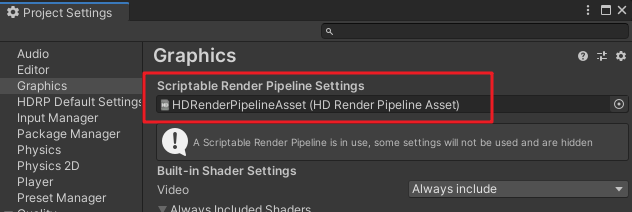
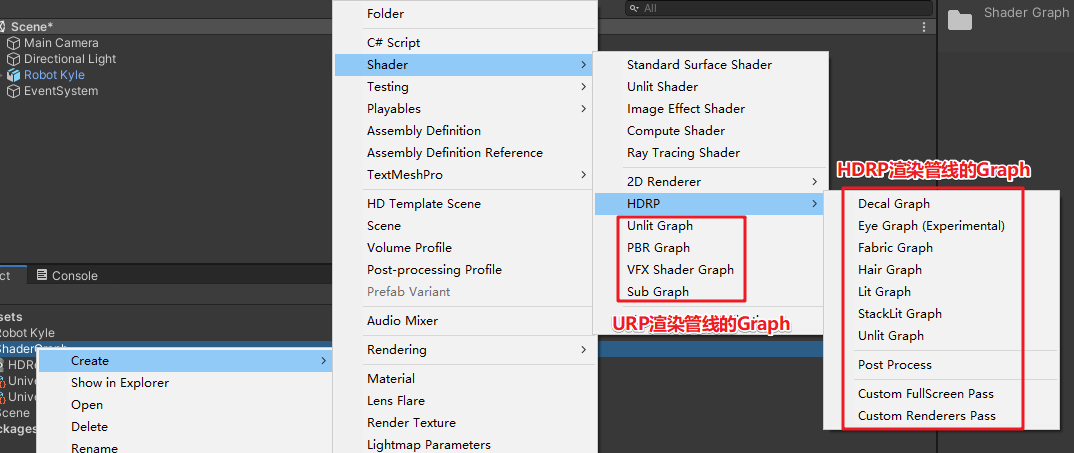
3.2、所以

unity引入可编程渲染管线，方便优化大佬自己编写合适的srp，降低drawcall、提高影响效率等，but！难度太大，具有劝退性质，so！ Unity提供里2个预制的管线：高清渲染管线（HDRP，全称High Definition Render Pipleline）、通用渲染管线（URP，全称Universal Render Pipleline），注：URP的前身是轻量级渲染管线LWRP，2019.3版本重命名

* 一点小小的总结：
* HDRP：更适合大型游戏，追求高品质，对硬件要求极高
* URP: 期望能实现等同甚至更好的品质与性能，且只需一次开发，便能在大量的平台上部署

## 知识点4：ShaderGraph！

4.1首先给一份简单的配置教程，摘自于[林新发](https://linxinfa.blog.csdn.net/" \o "林新发)的整合：

Unity2018推出了一个可编程渲染管线工具ShaderGraph，让开发者们可以通过可视化界面拖拽来实现着色器的创建和编辑。  
目前ShaderGraph支持的URP通用渲染管线，也支持HDRP高清渲染管线。  
我们通过Unity的PackageManager安装ShaderGraph，还需要安装对应的渲染管线工具包，比如要使用URP通用渲染管线，则需要通过PackageManager安装Universal RP，如果要使用HDRP高清渲染管线，则需要通过PackageManager安装High Definition RP。  
安装了渲染管线工具包之后，就可以通过菜单 Assets - Create - Rendering创建对应的渲染管线配置。  
  
接着在Editor - Project Setting - Graphics中的Scriptable Render Pipeline Settings设置渲染管线配置文件。  
如果是使用URP，则设置URP的渲染管线配置  
  
如果是使用HDRP，则设置HDRP的渲染管线配置  
  
然后，我们就可以创建对应的ShaderGraph，愉快得进行连连看了。  


4.2.一些常用节点

Step Node：如果输入值In大于等于Edge，则返回1，否则返回0

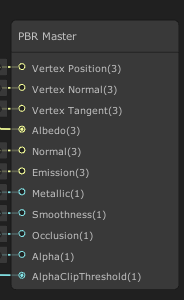
\*Split:分离通道 / Combine：合并通道

\*OneMinus：用一减去输入（即1-x）

4.3 master节点

PBR Master PBR Shader创建时的输出节点（Physicallly-BasedRendering）基于物理规律模拟渲染。

Unlit Master Unlit Shader创建时的输出节点。无反光材质



| 名称 | 类型 | 特殊说明 | 描述 |

| Albedo | Vector 3 | 无 | 材质的albedo值，范围是0-1 |

| Normal | Vector 3 | 切线空间法线 | 材质的法线值，应该是切线空间的法线 |

| Emission | Vector 3 | 无 | 材质的自发光颜色，应该是正数 |

| Metallic | Vector 1 | 无 | 材质的金属度，0代表无金属度，1代表是金属。只有在Workflow为metallic时显示 |

| Specular | Vector 3 | 无 | 材质的高光颜色，范围是0-1。只有在Workflow为Specular时显示 |

| Smoothness | Vector 1 | 无 | 材质的光泽度，范围是0-1 |

| Occlusion | Vector 1 | 无 | 材质的环境光遮挡值，范围是0-1 |

| Alpha | Vector 1 | 无 | 材质的alpha值，用于透明，也可用于alpha clip。应为0-1的值。 |

| Alpha Clip Threshold | Vector 1 | 无 | 低于这个阈值的部分会被裁掉。应为0-1的值。 |