# Unity中的Shader

## Standard Surface Shader

会产生一个包含了标准光照模型(使用了基于物理的渲染方法)的表面着色器模板。

## Unlit Shader

会产生一个不包括 光照(但包含雾效)的基本的顶点/片元着色器。

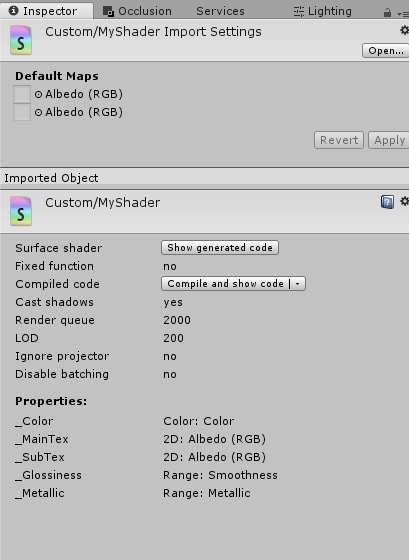
## Image Effect Shader

为我们实现各种 屏幕后处理效果 提过了一个基本模板。

## Compute Shader

会产生一种特殊的Shader文件，这类Shader旨在利用GPU的并行性来进行一些与常规渲染流水线无关的计算。

# Shader的导入信息



shader的默认数据

是否是 表面着色器；是否是 固定着色器

编译的方式

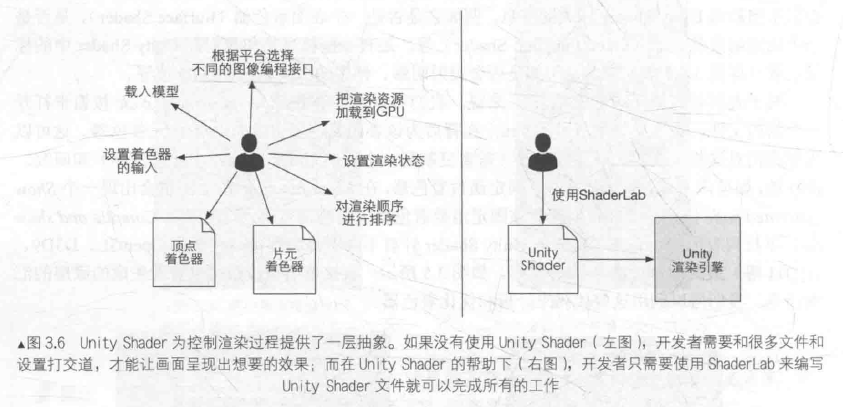
渲染队列，LOD值

是否关闭 批处理

Shader中的属性

# ShaderLab

UnityShader都是使用ShaderLab来编写的，UnityShader是对渲染的一种抽象，而ShaderLab是使用这种抽象的语言。

Unity会把ShaderLab编译成真正的代码和Shader文件。

# UnityShader的结构

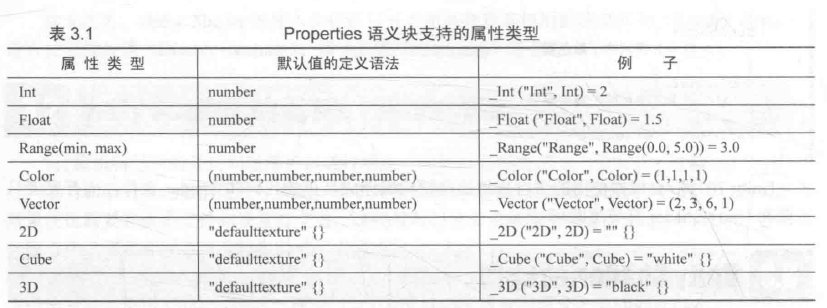
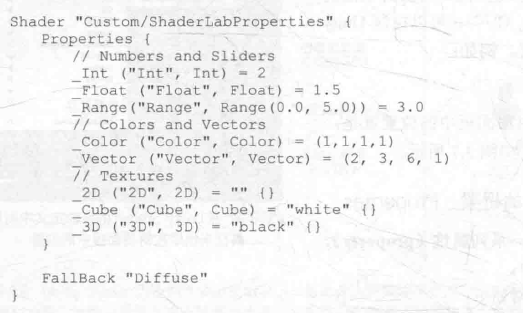
## 1 名字：

Shader “Custom/MyShader” { … … }

## 2 属性 properties:

Properties {

Name(“display\_name”, Type) = DefaultValue

}

## 3 SubShader

结构:

SubShader {

[Tags] // 标签

[RenderSetup] // 状态

Pass{

[Name]

[Tags]

[RenderSetup]

//Code

}

}

SubShader中定义了一系列Pass以及可选的状态和标签。每个Pass都是一次完整的渲染流程，如果Pass数量过多，渲染性能会下降。

状态和标签 同样可以在Pass中声明：

SubShader中的一些标签设置是特定的，即这些标签设置 和 Pass中使用的标签是不一样的。

状态设置 语法是一样的。SubShader进行了设置对所有Pass都起作用。

#### 状态设置

#### SubShader的标签

标签是 字符串类型的键值对，这些键值对是SubShader与渲染引擎之间 的沟通桥梁。它们用来告诉Unity的渲染引擎 怎样以及何时 渲染这个对象。

结构： Tags {“TagName1”=”Value1” “TagName2”=”Value2”}

Queue 渲染队列，控制渲染的顺序。

RenderType 着色器分类，可以用于着色器替换功能。

DisableBatching 一些SubShader使用Unity批处理功能时才会出现的问题。

ForceNoShadowCasting 控制使用该SubShader的物体是否会投射阴影

IgnoreProjector 是否无视 Projector(投影机，用于制造影子) 的影响。常用于半透明物体

CanUseSpriteAtlas 当SubShader用于精灵时，为False

ProviewType 预览材质的类型。

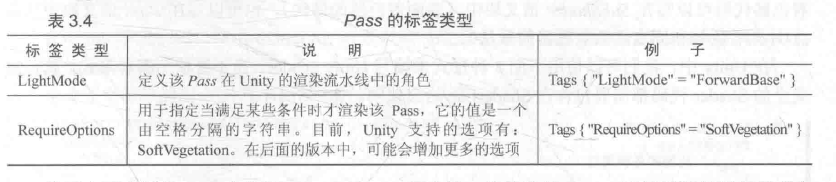
#### Pass语义块

Pass {

[Name] [Tags] [RenderSetup]

//Code

}

1. Name “MyPassName”
2. UsePass “MyShader/MYPASSNAME” //Unity内部会把所有的Pass的名字转为大写。
3. 渲染命令： Pass 不仅可以用SubShader的渲染命令，还可以使用固定管线的着色器命令
4. 标签：Pass的标签不同于SubShader的标签
5. 特殊的Pass：

UsePass：用于复用其他UnityShader中的Pass；

GrabPass：将抓取屏幕，并存放在一张纹理中。

#### Fallback

Fallback “name”

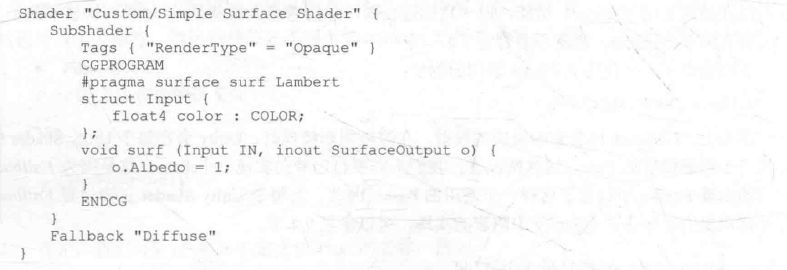
Fallback Off

实际上，Fallback还会影响阴影的投射。在渲染阴影纹理时，Unity会在每个UnityShader中寻找一个阴影投射的Pass，通常情况下，我们不需要自己专门实现一个Pass，因为Fallback使用的内置Shader中包含了这样的一个通用的Pass。

# UnityShader的形式

着色器代码可以写在SubShader语义块中(表面着色器的做法)，也可以写在Pass语义块中（顶点/片元着色器， 固定函数着色器的做法）。

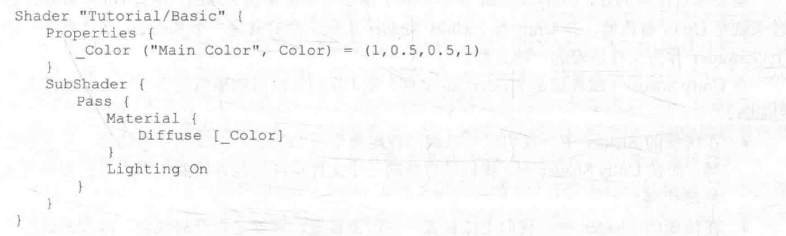
## 表面着色器

表面着色器 是对 顶点/片元着色器 的抽象，Unity还是会把它转化为 顶点片元着色器的，因此渲染代价比较大。但它的价值在于，Unity为我们处理了很多光照细节。

## 顶点/片元着色器

## 固定函数着色器

固定函数着色器完全使用ShaderLab的语法，后台变为 顶点片元着色器，真正意义的固定函数着色器已经不存在了。



## 如何选择UnityShader的形式？

1 能用可编程管线的着色器 不用 固定管线着色器

2 如果光源很多，使用表面着色器，小心移动平台上的性能

3 光照甚少，用顶点片元着色器

4 如果有很多自定义的渲染效果，用顶点片元着色器。