**什么是GPU?**

GPU:Graphic Processing Unit，中文翻译为“图形处理器”。显卡包括（GPU,显存，显卡BIOS,显卡PCB板）。

**什么是Shader?**

Shader程序:GPU执行的，针对3D对象进行操作的程序。

**Shader编程有哪几种？**

CG：与DirectX 9.0以上以及OpenGL 完全兼容。运行时或事先编译成GPU汇编代码。

HLSL: 主要用于Direct3D。平台:windows。

GLSL: 主要用于OpenGL。  平台:移动平台（iOS，安卓），mac(only use when you target Mac OS X or OpenGL ES 2.0)

**为什么Shader中选择CG?**

因为CG/HLSL 比GLSL支持更多的平台。

**Unity3d里CG输出什么？**

windows平台：Direct3D, GPU汇编代码

mac:OpenGL GPU汇编代码

flash:flash GPU汇编代码

ios/android:unity会将CG转换成GLSL代码。

总结：也就是除了移动平台会把CG转换成GLSL代码，其余平台都是转换成汇编代码。

**什么是缓冲？**

一个像素有如下缓冲

颜色缓存color buffer/pixel buffer：储存该点即将显示的颜色，RGBA值

深度缓存depth buffer/z buffer：储存该点的深度,z

模板缓存stencil buffer：通常用作限制渲染区域。更高级用法需结合深度缓冲，例如某像素的模板缓冲值会随着其是否通过深度缓冲测试而改变。

累积缓存Accumulation Buffer: 与颜色缓冲类似，同样储存一个RGBA值。累积缓存是为合成多幅图像而设计的，累积缓存提供了一种在保持好的颜色分辨率下实现在场景中“多重曝光（multiple exposures）”的方法。使用累积缓存可以产生许多图像效果来提高图像的真实性，其中包括：反走样、运动模糊、软阴影、深度域（景深）和卷积。要产生这些效果，必须将图像渲染多次，对场景位置（或所选的物体）进行微小的、渐增的改变，然后累积结果。

**什么是图元装配（Primitive Assembly）**

经过变换的顶点 被装配成几何图元

**什么是光栅化（又译作栅格化，Rasterization）**

栅格化这个术语可以用于任何将[矢量图形](http://zh.wikipedia.org/wiki/%E7%9F%A2%E9%87%8F%E5%9B%BE%E5%BD%A2" \o "矢量图形)转换成[栅格图像](http://zh.wikipedia.org/wiki/%E4%BD%8D%E5%9B%BE)的过程。

在3D渲染中主要是指，三角形等图元（矢量）转换成像素碎片的过程。或者说决定哪些像素几何图元覆盖的过程。光栅化的结果是像素位置的集合和片段的集合

**什么是光栅操作（Raster Operation）**

指在碎片fragment处理后，在更新帧缓存前最后执行的一系列操作。通过包括裁剪，深度测试，alpha测试，alpha混合等。

**碎片Fragment等于像素吗？**

像素点：（屏幕上能显示）的最小图像单元

像素:帧缓存中某个像素点的内容，通常即指颜色。

碎片：更新像素潜在需要的一个状态。

碎片输出的是当前的fragment函数在这个像素点的颜色，并不代表这像素点的最终颜色。最后显示的颜色是这个点的所有碎片经过叠加等运算形成的最终结果。

针对3D对象进行操作、并被[GPU](http://baike.baidu.com/view/1196.htm" \t "_blank)所执行的程序

* Unity中的三种自定义Shader:（参照帮助文档Shader Reference）
* [surface shaders](file:///D:\soft\MyUnity5.3.2\Unity\Editor\Data\Documentation\en\Manual\SL-SurfaceShaders.html), 表面着色器（之前默认创建的Shader类型）它是[vertex and fragment shaders](file:///D:\\soft\\MyUnity5.3.2\\Unity\\Editor\\Data\\Documentation\\en\\Manual\\SL-ShaderPrograms.html) 的包装,让我们可以不用去关心这些顶点和片段程序的细节，可以直接得到我们想要的着色器。
* [vertex and fragment shaders](file:///D:\soft\MyUnity5.3.2\Unity\Editor\Data\Documentation\en\Manual\SL-ShaderPrograms.html) 顶点和片元着色器
* fixed function shaders. 固定功能管线着色器，在可编程渲染管线硬件出现出现之前，很多光照都会放在硬件级处理(可理解为对固定管线硬件的操作)，一般在项目前对绝大多数硬件都可支持，应用就可以使用，比如光照、纹理采样

先从Shaderlab基本语法开始入手，再去阅读[surface shaders](file:///D:\\soft\\MyUnity5.3.2\\Unity\\Editor\\Data\\Documentation\\en\\Manual\\SL-SurfaceShaders.html),或者[vertex and fragment shaders](file:///D:\\soft\\MyUnity5.3.2\\Unity\\Editor\\Data\\Documentation\\en\\Manual\\SL-ShaderPrograms.html)

[vertex and fragment shaders](file:///D:\soft\MyUnity5.3.2\Unity\Editor\Data\Documentation\en\Manual\SL-ShaderPrograms.html) 顶点和片元着色器是ShaderLab中的语言片段，不仅可应用在ShaderLab，也可通过CG或者GLSL、HLSL进行编写，但fixed function shaders固定功能管线着色器必须在ShaderLab中进行编写。

# ShaderLab基本结构

Unity中的Shader都是要通过ShaderLab的基本语法进行编写，unity就是想通过Shaderlab的方案进行Shader的编写。将三种定义的Shader通过同一种格式进行编写，避免不同Shader使用不同的语法。

**Shader “name”｛**

**【Properties】//属性** **查看ShaderLab：Properties**

**作用 在可视化面板提供美工可使用的属性**

**SubShaders //shader算法 查看ShaderReference**

**ShaderLab中最少有一个SubShader 也可以多个**

**【FallBack】//后退 一般会填写所有硬件都支持的渲染方式**

**｝**

关于SubShaders（处理ShaderLab中的语言片段）

在ShaderLab中至少有一个SubShader，当然也可多个。

但是，显卡每次渲染处理的时候只能选择一个SubShaders执行。那多个SubShader的作用是为了不同硬件的渲染支持，为了Shader能在比较老的图形显卡中也能支持。一般比较越往下的SubShader要简化，运算指令要简单。

# Fixed function shader固定功能管线

所有硬件平台都可支持，针对硬件能够执行的基本命令的Shader，当然，功能有限，但是，速度最快。

## Properties 属性

## Material 材质

## Lighting光照

## Settexture设置纹理

## Pass通道(存储图像的色彩)

# [surface shaders](file:///D:\soft\MyUnity5.3.2\Unity\Editor\Data\Documentation\en\Manual\SL-SurfaceShaders.html)

surfaceOutput输出

Input输入

Lighting光照