片元着色器的输入是上一个阶段对顶点信息插值得到的结果，更具体的说，是根据那些从顶点着色器中输出的数据插值得到的。而片元着色器的输出是一个或者多个颜色值

4种shader

1. Standard Surface Shader会产生一个包含了标准光照模型（使用了unity5中新添加的基于物理的渲染方法）的表面着色器模板。
2. Unlit Shader则会产生一个不包含光照（但包含雾效）的基本的顶点/片元着色器。
3. Image Effect Shader则为我们事先各种屏幕后处理效果提供了一个基本模板。
4. 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。

Unity是左手坐标系

坐标系变换

Ap = Mc->pAc

M是4 4变换矩阵，可以逆向思维，提取它的第一列再进行归一化来得到模型空间的x轴在世界空间下的单位矢量表示，只需要用M乘[1 0 0 0]的逆矩阵，就能得到M的第一列了

另外，矢量没有位置信息，所以坐标空间的原点变换是可以忽略的，所以可以用3x3表示，在shader中，我们常常会看到截取变换矩阵的前三行和前三列来对法线方向、光照方向来进行空间变换，这正是原因所在。

P87，统一缩放那里可以当做正交矩阵那里没看懂

float:最高精度，通常32位  
　　half:中等精度，通常16位，-60000到60000，  
　　fixed:最低精度，通常11位，-2.0到2.0，1/256的精度。  
　　尽量使用低精度。对于color和unit length vectors，使用fixed，其他情况，根据取值范围尽量使用half，实在不够则使用float。  
　　在移动平台，关键是在fragment shader中尽可能多的使用低精度数据。另外，对于多数移动GPU，在低精度和高精度之间转换是非常耗的，在fixed上做swizzle操作也是很费事的。

编写shader时的建议

<https://www.cnblogs.com/sifenkesi/p/4716791.html>

带SV的就代表是裁剪空间(齐次)，用SV开头的变量是不可以随便赋值的，因为流水线需要使用它们来完成特定的目的，例如，会吧SV\_POSITION修饰的变量经过光栅化后显示在屏幕上。

在绝大多数平台，POSITION和SV\_POSITION是等价的，但是某些平台（ps4）上必须使用SV\_POSITION来修饰顶点着色器的输出，否则无法让shader正常工作，同样的例子还有COLOR和SV\_Target，所以，最好输出都用SV；

UNITY\_LIGHTMODEL\_AMBIENT 环境光的颜色和强度信息

一个重要特殊点，法线变换的特殊性，发现变换需要 原变换矩阵的逆转置矩阵来变换

所以！！！ 把法线从模型空间转到世界空间，需要Mul(normal, \_World2Object), 首先需要原矩阵的逆矩阵，正常是需要\_Obect2World，他的逆矩阵就是\_World2Object，又因为需要转置矩阵，所以向量需要左乘，所以就是这样了,详见以下连接

<https://blog.csdn.net/jxw167/article/details/77666599>

反射向量r计算 l入射向量，N法线

r = l – 2(n\*l)\*n

<https://blog.csdn.net/yinhun2012/article/details/79466517>

可以直接用函数 reflect(l,n)求出反射向量

Unity中内置的各种方法，都没有保证得到的方向矢量是单位矢量，因此，我们需要在使用前归一化

注意！！！，通常在变换顶点时，使用右乘按列矩阵进行乘法，因为unity内置矩阵都是按列存储的。

纹理坐标变换：

o.uv = v.texcoord.xy \* \_MainTex\_ST.xy + \_MainTex\_ST.zw;

也可以用

o.uv = TRANSFORM\_TEX(v.texcoord, MainTex\_ST);

渐变纹理需要把Wrap Mode设为Clamp

TANGENT\_SPACE\_ROTATION 代替通过切线和法线求模型到切线空间的转换矩阵

一般情况下，比如一个shader又有纹理又有法线又有遮罩，这时其实只需要用纹理的uv就可以，因为法线和遮罩要跟着纹理动的

延迟渲染不受灯光数量影响，但是不支持真正的抗锯齿功能，不能处理半透明物体

\_LightColor0已经是颜色和强度相乘后的结果

计算光照衰减需要用到光照衰减纹理，这样可以减少计算，式子是这样的

#if defined (POINT)

float3 lightCoord = mul(unity\_WorldToLight, i.worldPos);

//fixed atten = tex2D(\_LightTexture0, dot(lightCoord, lightCoord).rr).UNITY\_ATTEN\_CHANNEL;

fixed atten = tex2D(\_LightTexture0, dot(lightCoord, lightCoord).rr).UNITY\_ATTEN\_CHANNEL;

至于为什么要dot他的平方，因为衰减纹理是这个样子



所以需要算出离光源的距离，通过世界转光源坐标的矩阵算出光源坐标系中的顶点位置后，需要求模来算出距离，但是求模需要开方，所以可能官方就设定成距离平方的形式

经测验



这么写也可以，自身点积自身其实就是模的乘积…

.rr的意思是，把变量变成2维，比如3.rr 就是（3,3）

“DisableBatching” = “True” 关闭批处理，有些shader在使用批处理的时候会有问题，比如模型空间的顶点动画的shader，这是因为批处理会合并所有相关的模型，而这些模型各自的模型空间就会丢失

用了ComputeScreenPos之后的点不是最终结果，详情见92页，需要用xy分量除w分量，结果后补0 1