金属，塑料，傻傻分不清楚

叛逆者

计算机图形学、C++ 话题的优秀回答者

很多时候可以听到游戏玩家说，某某游戏看起来什么都是塑料做的。最近还听一个设计师说，你用Blinn-Phong吧，那东西渲染出来就是浓浓的塑料感的。

金属和塑料，区别在哪里，真的是因为BRDF模型吗？

Diffuse/Specular

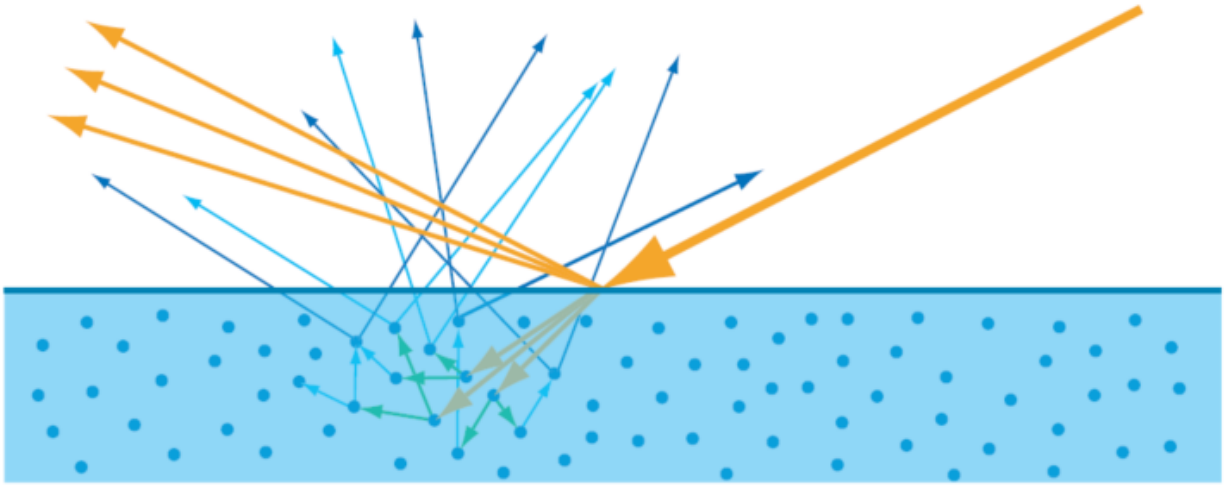
就从Blinn-Phong模型入手。最简单的基于物理的Blinn-Phong是这样的：

$$L_o(\mathbf{v}) = (\mathbf{c}_{diff} + \frac{\alpha + 2}{8}(\mathbf{n} \cdot \mathbf{h})^{\alpha} F(\mathbf{c}_{spec}, \mathbf{l_c}, \mathbf{h})) \otimes \mathbf{c}_{light}(\mathbf{n} \cdot \mathbf{l_c})$$

其中 $\mathbf{c_{diff}}$ 和 $\mathbf{c_{spec}}$ 分别是diffuse和specular的颜色， α 是表述光滑程度的。跟物体材质相关的，也就是这三项了。话句话说，决定一个物体看起来像什么，在Blinn-Phong模型里，就是通过调整这三项完成的。

那么，是光滑度造成的吗？不是。塑料也有粗糙和光滑，金属也有粗糙和光滑。并不会因为是金属就不那么粗糙。所以，现在的问题就是，什么决定了材质的diffuse和specular颜色。

回到基于物理的BRDF这个出发点，看看从物理的角度，diffuse和specular来自哪里。

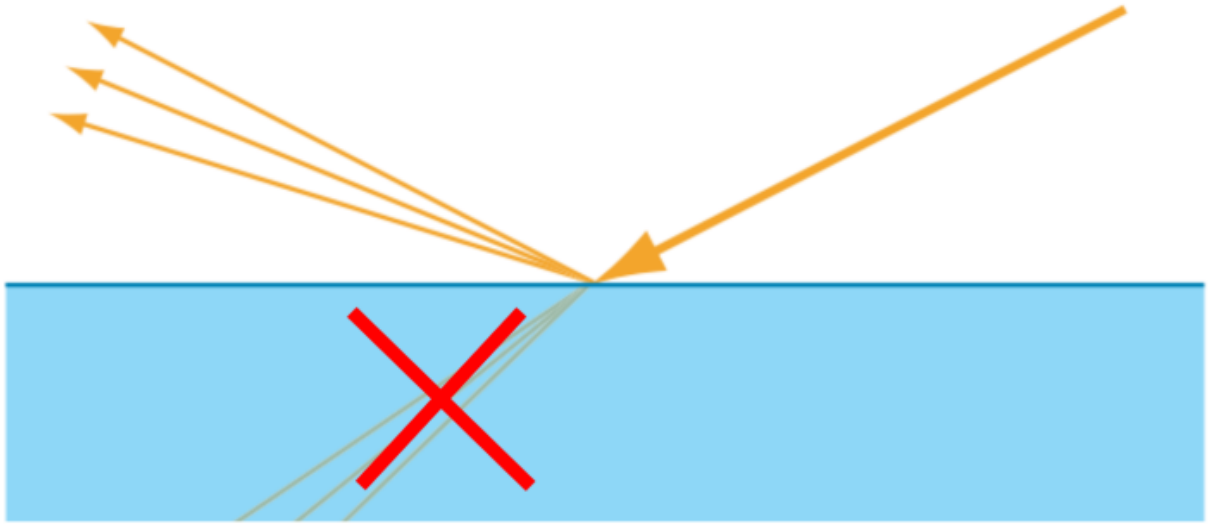


上图来自于Real-time Rendering第三版。箭头根据颜色可以分成几种。简化起见，就看最终出射的情况。深蓝色和浅蓝色表示diffuse（分别来自于single scattering和multiple scattering，但这不重要），金黄色表示specular。原理上说，diffuse是光线

折射入物体后，在内部经过散射，重新穿出表面的光。而specular是光线在物体表面直接反射的结果。并没有经过转换，还是原来的光子。

（从这张图还可以看出，因为物体的密度和材料不同，diffuse可能分布到一个很广的范围内，不是一个点，而是一个光斑。specular就是一个点。而且，这两者位置上不重合。不过这已经超出了实时渲染甚至离线渲染的范畴，这里不讨论）

然而，对于金属（导体）来说，折射入物体的光子会被完全吸收掉，就成了这个样子：



好了，这下明白了。对于金属来说，diffuse永远是0，材质的颜色来自于 specular。对于非金属来说，材质的颜色一部分来自于diffuse，一部分来自于specular。

颜色分布

既然如此，也就是说如果我们有能力控制diffuse和specular的颜色，就一定能在渲染上复现出金属（导体）和塑料（绝缘体）的区别，甚至介于其间的半导体。在此之前，先找一些现实中的材料，看看它们的diffuse和specular颜色分布究竟如何。

这里列举一下来自Physics and Math of Shading的几张表。

Metal	F_0 (Linear, Float)	F_0 (sRGB, U8)	Color
Titanium	0.542,0.497,0.449	194,187,179	
Chromium	0.549,0.556,0.554	196,197,196	
Iron	0.562,0.565,0.578	198,198,200	
Nickel	0.660,0.609,0.526	212,205,192	
Platinum	0.673,0.637,0.585	214,209,201	
Copper	0.955,0.638,0.538	250,209,194	
Palladium	0.733,0.697,0.652	222,217,211	
Zinc	0.664,0.824,0.850	213,234,237	
Gold	1.022,0.782,0.344	255,229,158	
Aluminum	0.913,0.922,0.924	245,246,246	
Silver	0.972,0.960,0.915	252,250,245	

第一张是金属的specular颜色。前面说到金属的diffuse = 0，所以你看到的金属颜色，就是它的specular。从这张表可以看出，金属的specular非常强烈，而且各通道都几乎大于0.5。唯一的例外是金，蓝色通道较弱，而红色通道已经超出了sRGB的表达范围。（如此独特的反射曲线，也是金子为什么有着独特的文化和经济地位的原因之一吧）

再来看看绝缘体阵营

Dielectric	F_0 (Linear, Float)	F_0 (sRGB, U8)	Color
Water	0.020	39	
Plastic, Glass	0.040 – 0.045	56 – 60	
Crystalware, Gems	0.050 – 0.080	63 – 80	
Diamond-like	0.100 – 0.200	90 – 124	

同样是specular颜色，非金属则显得弱了很多。塑料、玻璃、水这些常见的绝缘体，specular都低于0.04。而水晶、钻石这些绝缘体， 你们知道为什么贵了吧。另一个特点是，绝缘体的specular都是单色的。各个通道的光都会被等同地反射，而不像金属那样有着不同的吸收偏好。

而半导体呢？如你所想，介于两者之间。

Substance	F_0 (Linear, Float)	F_0 (sRGB, U8)	Color
Diamond-like	0.100 – 0.200	90 – 124	
Crystalline Silicon	0.345,0.369,0.426	159,164,174	
Titanium	0.542,0.497,0.449	194,187,179	

有了这些信息，我们就可以进一步把它用于渲染了。

GBuffer

这几年的渲染引擎，几乎都是用的deferred rendering框架，其中一个兵家必争之地就是GBuffer。每年都有不同的文章在讲如何把更多的信息挤到GBuffer中，正因为GBuffer空间极其有限，而想放入的信息有越来越多。所以，如何利用信息之间的关系，把数据编码到有限的空间里，对deferred rendering来说非常重要。

如果按照前面的结论，需要放diffuse和specular颜色，则需要6个通道，完全超出了GBuffer可以承受的范围。GBuffer里留给diffuse和specular颜色的通道只有4个。所以游戏引擎常见的做法是，保存个diffuse颜色和specular的亮度。就当作specular是单色的来处理。这样显然对于金属不利，也是为什么会画面看着都像塑料的重要原因。

有了上一节的信息，直觉告诉我们一个想法，对于导体和绝缘体，应该分开用不同的保存方法。导体存specular颜色，因为它的diffuse是0。绝缘体存diffuse，因为specular非常弱，都用0.04这个常量表达就可以了。那么半导体呢？就需要一个“金属性”的系数，在两者之间插值。

用公式来表达，就是

diffuse = albedo * (1 - metalness)
specular = lerp(0.04, albedo, metalness)

其中，albedo就是3通道的反照率，metalness就是金属性系数。lerp是线性插值的意思。这样就用一个方法统一了导体、半导体、绝缘体的颜色分布。metalness = 0表示绝缘体，diffuse就是albedo，specualr = 0.04。metalness = 1表示导体，diffuse = 0，specular = albedo。这么一来，只要在GBuffer里面存albedo（3通道）和metalness（1通道）。

结果和总结

这么做的渲染结果如何呢？咱们来看看一组对比，用的是前面提到的基于物理的Blinn-Phong，用的同样的albedo。

粗糙的绝缘体

光滑的绝缘体

粗糙的导体

光滑的导体

从图上还是很容易分辨那些是绝缘体那些是导体。而且很明显可以看出，这件事情和光滑度无关，和BRDF模型也无关。

在来对比一下在引入albedo+metalness之前能达到的效果。

对比强烈吧，同样的参数，原先的做法看起来就是油腻的塑料。

最后，来一组金属性和粗糙度变化的阵列。

我们用albedo+metalness的方法，在没有增加存储空间，和几乎没有增加计算开销的情况下，实现了更高质量的金属渲染。从此，只要做一些微小的工作，调调材质，就能避免塑料感。

在KlayGE里面，GBuffer的格式也已经改成这里说的存储方式，材质系统也做了相应的修改，以更好地表达这样的材质分类。

在有些游戏引擎，比如Dying Light，用了一个有点区别的编码方式。那里面只有导体和绝缘体两种切分，没有metalness的系数可以做过渡。这样就可以把绝缘体的specular给编码进去。又因为绝缘体specular非常的小，只要占用很少的位数。空出来的位可以把translucent、皮肤、叶片等信息都编码进去。但对导体绝缘体的分析和本文相同。