实时渲染 第4版

第5章 Shading Basics 基础着色

When you render images of three-dimensional objects, the models should not only

have the proper geometrical shape, they should also have the desired visual

appearance. Depending on the application, this can range from photorealism—an

appearance nearly identical to photographs of real objects—to various types of

stylized appearance chosen for creative reasons. See Figure 5.1 for examples of both.

This chapter will discuss those aspects of shading that are equally applicable to

photorealistic and stylized rendering. Chapter 15 is dedicated specifically to stylized

rendering, and a significant part of the book, Chapters 9 through 14, focuses on

physically based approaches commonly used for photorealistic rendering.

当你渲染三维对象图像时，模型不仅仅要有合适的几何外形，还要有理想的视觉外观。依据应用程序的不同，这可以包括从几乎与真实物体照片相同的光照到出于创造性原因而选择的各种类型的程序式外观。图5.1给出了两者的示例。这一章将主要从着色的角度，讨论真实和非真实渲染。第15章是专门为风格化渲染，书的重要部分，第9章到第14章，集中在物理基础上的方法，通常用于真实渲染。

5.1 Shading Models 着色器模型

The first step in determining the appearance of a rendered object is to choose a

shading model to describe how the object’s color should vary based on factors such

as surface orientation, view direction, and lighting.

决定渲染物体的呈现外观的第一步是选择一个着色器模型去描述物体的颜色应该如何根据表面方向，视角方向和光照因素而变化。

As an example, we will use a variation on the Gooch shading model [561]. This is a

form of non-photorealistic rendering, the subject of Chapter 15. The Gooch shading

model was designed to increase legibility of details in technical illustrations.

例如，我们将使用Gooch着色模型的一个变体[561]。这是非真实渲染的一种形式，是第15章的主题。Gooch着色模型是为了增加技术插图的细节而设计的。

The basic idea behind Gooch shading is to compare the surface normal to the light’s

location. If the normal points toward the light, a warmer tone is used to color the

surface; if it points away, a cooler tone is used. Angles in between interpolate

between these tones, which are based on a user-supplied surface color. In this

example, we add a stylized “highlight” effect to the model to give the surface a shiny

appearance. Figure 5.2 shows the shading model in action.

Gooch着色的基础思想是比较表面法线朝向光照的位置，如果法线点是朝向光照的话，就在表面使用暖色色调的颜色；如果偏离光照，会使用冷色色调的颜色。在色调之间使用角度插值。在这个例子中，我们增加一个非真实“高亮”效果在模型上，给表面一种光泽的表现。图5.2显示了着色器模型。

Shading models often have properties used to control appearance variation. Setting

the values of these properties is the next step in determining object appearance. Our

example model has just one property, surface color, as shown in the bottom image

of Figure 5.2.

着色器模型常常会提供属性来控制外观变化。设置这些属性的值是决定物体表面的下一步。我们的例子只有一个属性：表面颜色，显示在图5.2的底部图片。

Like most shading models, this example is affected by the surface orientation relative

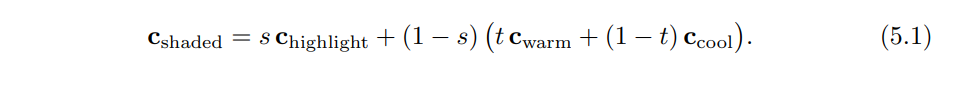
to the view and lighting directions. For shading purposes, these directions are

commonly expressed as normalized (unit-length) vectors, as illustrated in Figure 5.3.

像大部分着色器模型，这个例子受到了表面方向相对于视角以及光照方向的影响。出于着色目的，这些方向通常表示为标准化（单位长度）向量，如图5.3所示。

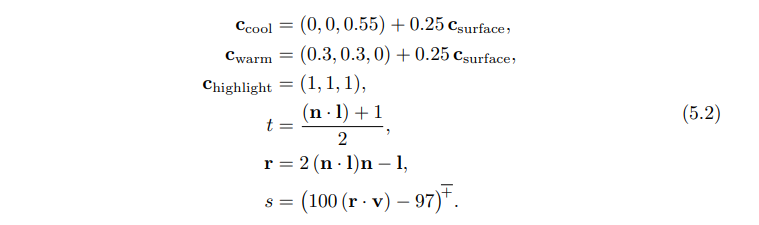
Now that we have defined all the inputs to our shading model, we can look at the

mathematical definition of the model itself:



现在我们定义了着色器模型的输出，我们可以看看模型本身的数学定义。

In this equation, we have used the following intermediate calculations:



在这个方程中个，我们可以使用下面中间计算过程。

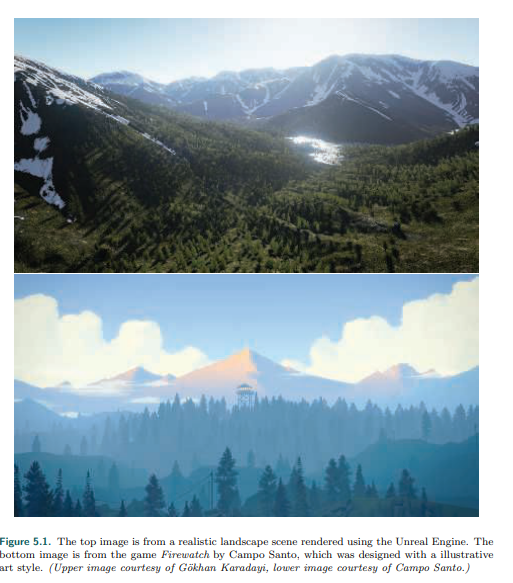


图5.1 顶部的照片是一张使用虚幻引擎的真实风景渲染。底部照片来自Campo Santo制作的游戏Firewatch，使用了插图艺术风格设计（上面图片来自G¨okhan Karadayi，下部图片来自Campo Santo）。

Several of the mathematical expressions in this definition are often found in other

shading models as well. Clamping operations, typically clamping to 0 or clamping

between 0 and 1, are common in shading. Here we use the x + notation, introduced

in Section 1.2, for the clamp between 0 and 1 used in the computation of the

highlight blend factor s. The dot product operator appears three times, in each case

between two unit-length vectors; this is an extremely common pattern. The dot

product of two vectors is the product of their lengths and the cosine of the angle

between them. So, the dot product of two unit-length vectors is simply the cosine,

which is a useful measure of the degree to which two vectors are aligned with each

other. Simple functions composed of cosines are often the most pleasing and

accurate mathematical expressions to account for the relationship between two

directions, e.g., light direction and surface normal, in a shading model.

在这个定义中的几个数学公式经常在其他的着色器模型中看到。钳位操作，在常规着色中，典型的是钳位到0或者钳位到0到1之间。在这里我们使用x(+-)符号，在章节1.2中介绍过，钳位到0到1之间用在计算高光混合因子s。点积操作出现了三次，每种情况都是在两个单位长度之间；这是极其常规的操作。两向量的点积是指的是他们的长度以及两者之间角度的cos之积。因此两个单位长度的向量积就可以简单的用cos表示，这对于衡量两个向量的角度是否一致非常有用。在着色器模型中，cos的简单的函数组合，通常是描述两个方向（例如光照方向的表面法线）之间关系的最令人愉快和准确的数学表达式。

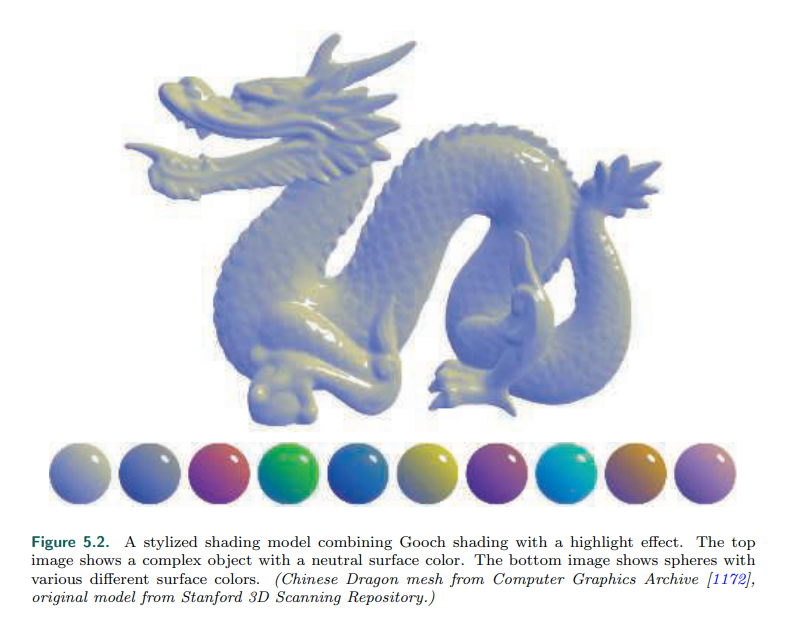


图5.2 一个风格化的着色模型联合了Gooch着色使用了高光影响。上部图像显示了混合物体使用了中性的表面颜色。底部的照片显示了使用了不同颜色表面颜色的球（中国龙网格来自计算机图形档案[1172]，原始模型来自斯坦福三维扫描库）。

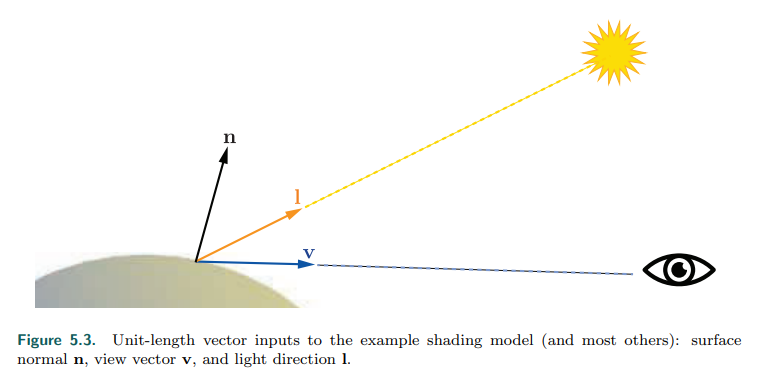


图5.3：单位长度向量输入着色器模型例子（其他大部分模型也是）：表面法线n，视角v，以及光照方向l。

Another common shading operation is to interpolate linearly between two colors

based on a scalar value between 0 and 1. This operation takes the form tca + (1 –

t)cb that interpolates between ca and cb as the value of t moves between 1 and 0,

respectively. This pattern appears twice in this shading model, first to interpolate

between cwarm and ccool and second to interpolate between the result of the

previous interpolation and chighlight. Linear interpolation appears so often in shaders

that it is a built-in function, called lerp or mix, in every shading language we have

seen.

另一个常规着色操作是两个基础颜色以0到1之间的标量进行线性插值。这种操作tca + (1 − t)cb通过t在0到1之间的移动，在ca和cb之间插值。这个模式在着色器模型中出现了两次，第一次在cwarm和ccool之前插值，第二次是在之前结果和chighlight之间插值。线性插值在着色器模型中的出现如此频繁，因此特定建立了函数，称作lerp或者mix，在我们看到的每种着色器语言中。

The line “r = 2 (n · l)n − l” computes the reflected light vector, reflecting l about n.

While not quite as common as the previous two operations, this is common enough

for most shading languages to have a built-in reflect function as well.

“r = 2 (n · l)n − l”这一行计算光向量的反射向量，关于n反射l。并不是如先前的两个操作那样常规，但是足够常规到大部分着色器语言都建立了反射函数。

By combining such operations in different ways with various mathematical

expressions and shading parameters, shading models can be defined for a huge

variety of stylized and realistic appearances.

通过联合各种各样的数学表达式和着色参数的此类操作，着色模型可以被各种风格和现实定义以及呈现。

5.2 Light Source 光源

The impact of lighting on our example shading model was quite simple; it provided a

dominant direction for shading. Of course, lighting in the real world can be quite

complex. There can be multiple light sources each with its own size, shape, color,

and intensity; indirect lighting adds even more variation. As we will see in Chapter 9,

physically based, photorealistic shading models need to take all these parameters

into account.

光照的影响在我们着色器模型例子中十分简单；他提供了一个显著的方向。当然，光照在真实世界中十分复杂。这里可能会有多重光源，每个都有自己的尺寸，形状，颜色以及强度；间接照明增加了更多的变种。我们会在第9章讲述基于物理的真实着色模型需要将这些所有参数考虑进去。

In contrast, stylized shading models may use lighting in many different ways,

depending on the needs of the application and visual style. Some highly stylized

models may have no concept of lighting at all, or (like our Gooch shading example)

may only use it to provide some simple directionality.

相反，非写实着色模型可能以很多不同的方式使用光照，这依据于程序和虚拟风格的需要而定。一些高度非写实模型可能没有光照的概念，或者（例如我们的Gooch着色例子）仅仅用它提供一些简单的方向。

The next step in lighting complexity is for the shading model to react to the presence

or absence of light in a binary way. A surface shaded with such a model would have

one appearance when lit and a different appearance when unaffected by light. This

implies some criteria for distinguishing the two cases: distance from light sources,

shadowing (which will be discussed in Chapter 7), whether the surface is facing away

from the light source (i.e., the angle between the surface normal n and the light

vector l is greater than 90◦ ), or some combination of these factors.

光照复杂性的下一步是对于着色模型以二进制的方式对光的存在或不存在做出反应。用这种模型的表面着色在光照下会有一种外观，在不受光照影响的情况下会有不同的外观。这意味着区分这两种情况的一些标准：光源的距离，阴影（将在第7章节讨论），表面是否朝向光源（即表面法线n和光向量l之间的角度是否大于90度），或者是这些因素的组合。

It is a small step from the binary presence or absence of light to a continuous scale

of light intensities. This could be expressed as a simple interpolation between

absence and full presence, which implies a bounded range for the intensity, perhaps

0 to 1, or as an unbounded quantity that affects the shading in some other way. A

common option for the latter is to factor the shading model into lit and unlit parts,

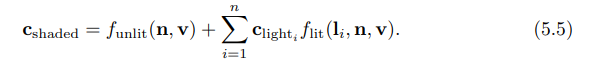
with the light intensity klight linearly scaling the lit part:



This easily extends to an RGB light color clight,



and to multiple light sources,



这是一个小的步骤从二元存在或者没有光到一个连续规模的光强度。这可以表示为一个简单的无光到满光的插值运算，这意味着一个有界的范围的强度，也许就是0到1，或者作为一个无界的数量，以某种方式影响着色。后者的一个常见的选择是将着色模型分解为光照的非光照部分，以及光照强度klight线性影响光照部分。

很容易拓展到RGB光照颜色。

以及拓展到多重光源。

The unlit part funlit(n, v) corresponds to the “appearance when unaffected by light”

of shading models that treat light as a binary. It can have various forms, depending

on the desired visual style and the needs of the application. For example, funlit() =

(0, 0, 0) will cause any surface unaffected by a light source to be colored pure black.

Alternately, the unlit part could express some form of stylized appearance for unlit

objects, similar to the Gooch model’s cool color for surfaces facing away from light.

Often, this part of the shading model expresses some form of lighting that does not

come directly from explicitly placed light sources, such as light from the sky or light

bounced from surrounding objects. These other forms of lighting will be discussed in

Chapters 10 and 11.

无光部分funlit(n, v)对应于将光视为二进制的着色模型的“不受光照影响时的外观”。他可以有多种格式，依据于所需的视觉风格和应用程序的需要。例如，funlit() = (0, 0, 0)将使任何不受光源影响的表面呈现纯黑色。另外，非光照部分可以为非真实对象表示某种形式的风格化外观，类似于Gooch模型为背向光的表面使用的冷色。通常，着色模型的这一部分表达了某种形式的照明并不直接来自于显示放置的光源，比如来自天空或者来自周围物体的反弹光。这种其他形式的照明将在第10章和第11章中讨论。

We mentioned earlier that a light source does not affect a surface point if the light

direction l is more than 90◦ from the surface normal n, in effect coming from

underneath the surface. This can be thought of as a special case of a more general

relationship between the light’s direction, relative to the surface, and its effect on

shading. Although physically based, this relationship can be derived from simple

geometrical principles and is useful for many types of non-physically based, stylized

shading models as well.

我们之前提到过光源并不会影响当表面法线n与光照方向l角度大于90度的情况。这可以看做是光照方向，相对于表面和他对着色影响之间更一般关系的一个特例。虽然是基于物理的，这种关系可以从简单几何法则中推导出来，并且对于许多类型是非基于物理的、非真实着色模型也是有用的。

The effect of light on a surface can be visualized as a set of rays, with the density of

rays hitting the surface corresponding to the light intensity for surface shading

purposes. See Figure 5.4, which shows a cross section of a lit surface. The spacing

between light rays hitting the surface along that cross section is inversely

proportional to the cosine of the angle between l and n. So, the overall density of

light rays hitting the surface is proportional to the cosine of the angle between l and

n, which, as we have seen earlier, is equal to the dot product between those two

unit-length vectors. Here we see why it is convenient to define the light vector l

opposite to the light’s direction of travel; otherwise we would have to negate it

before performing the dot product.

光照的影响可以形象化为一组射线，射线的密度击中表面对应于光强作用于表面着色。如图5.4，显示了光照表面的横截面。沿着横截面击中表面的光射线之间的空隙与l和n之间的夹角的cos值成负相关。因此，光射线击中表面的整体密度与l和n之间夹角的cos值成正相关，正如我们之前所见，与这两个单位长度向量的点积相等。这里我们可以看到为什么定义光向量l与光的运动方向相反是很方便的；否则我们做点积之前要先做负处理。

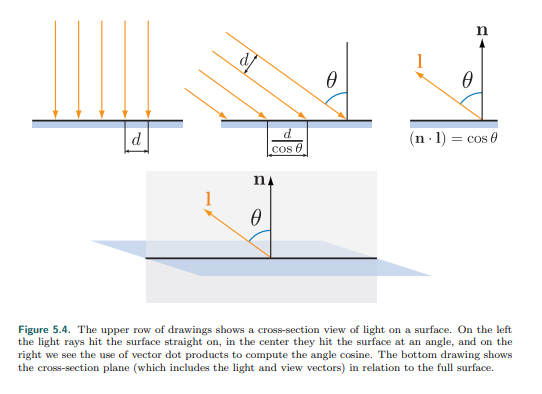


图5.4 上面一行显示的是光在表面的横截面处理。在左侧光射线垂直射向表面，在中间他们以一定的角度射向表面，在右侧我们看到计算l和n的点积来计算其角度。底部图片显示了整个平面的横截面（包括了光和视角向量）。

More precisely, the ray density (and thus the light’s contribution to shading) is

proportional to the dot product when it is positive. Negative values correspond to

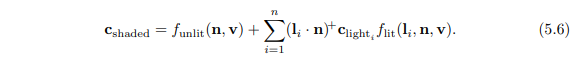
light rays coming from behind the surface, which have no effect. So, before

multiplying the light’s shading by the lighting dot product, we need to first clamp the

dot product to 0. Using the x + notation introduced in Section 1.2, which means

clamping negative values to zero, we have

更确切的说，射线密度（光对于着色的贡献）只有当点积为正值的时候才与点积成正相关。负值对应的光射线是来自表面的背部，这一部分光线并没有什么影响。因此，在通过点积增加光的着色影响时，我们需要映射点积到0，使用在1.2节介绍的x +符号，意味着将负值映射为0，因此公式如下。



Shading models that support multiple light sources will typically use one of the

structures from Equation 5.5, which is more general, or Equation 5.6, which is

required for physically based models. It can be advantageous for stylized models as

well, since it helps ensure an overall consistency to the lighting, especially for

surfaces that are facing away from the light or are shadowed. However, some

models are not a good fit for that structure; such models would use the structure in

Equation 5.5.

支持多重光源的着色器模型可以使用公式5.5中结构，这个更为普遍一些，或者使用公式5.6，常用在基于物理的模型。他同样适用于非真实渲染，因为他可以帮助确定表面与光的一致性，尤其是表面朝向光还是阴影。然而，一些模型并不适用这个公式；这些模型将使用公式5.5。

The simplest possible choice for the function flit() is to make it a constant color,



which results in the following shading model:



最简单的可能性选择是将函数flit()设置为常量颜色。那么着色模型的结果就如下所示。

The lit part of this model corresponds to the Lambertian shading model, after Johann

Heinrich Lambert [967], who published it in 1760! This model works in the context of

ideal diffusely reflecting surfaces, i.e., surfaces that are perfectly matte. We present

here a somewhat simplified explanation of Lambert’s model, which will be covered

with more rigor in Chapter 9. The Lambertian model can be used by itself for simple

shading, and it is a key building block in many shading models.

模型中光的部分对应于兰伯特着色模型，由Johann Heinrich Lambert[967]出版与1760年。这个模型是在理想漫反射表面的情况下工作，即表面是完美光泽。我们这里只对兰伯特模型做一个简单的介绍，第9章将会给出更严谨的说明。兰伯特模型可以使用在简单的着色情况，并且他也是很多着色模型的关键点。

We can see from Equations 5.3–5.6 that a light source interacts with the shading

model via two parameters: the vector l pointing toward the light and the light color

clight. There are various different types of light sources, which differ primarily in how

these two parameters vary over the scene.

从公式5.3-5.6，我们可以看出光源通过两个参数影响着色模型：向量l指向光的方向和光的颜色clight。有各种不同的光源类型，他们的主要区别在于如何使用这两个参数随场景而变化。

We will next discuss several popular types of light sources, which have one thing in

common: At a given surface location, each light source illuminates the surface from

only one direction l. In other words, the light source, as seen from the shaded

surface location, is an infinitesimally small point. This is not strictly true for real-world

lights, but most light sources are small relative to their distance from illuminated

surfaces, making this a reasonable approximation. In Sections 7.1.2 and 10.1, we

will discuss light sources that illuminate a surface location from a range of directions,

i.e., “area lights.”

我们接下来讨论几种常见的光源类型，他们有一点相同：在给定表面位置，每一个光源，只有一个方向l照亮表面。用其他话来说，光源从着色表面位置来看，他只是一个极小的点。这对于真实世界光来说，并不完全正确，但是大部分光源相对于他们照亮表面的距离来讲都很小，这似乎是一个合理的近似。在章节7.1.2和10.1，我们将讨论光源从一个范围方向照亮表面位置，即区域光。

5.2.1 Directional Lights 平行光

Directional light is the simplest model of a light source. Both l and clight are constant

over the scene, except that clight may be attenuated by shadowing. Directional lights

have no location. Of course, actual light sources do have specific locations in space.

Directional lights are abstractions, which work well when the distance to the light is

large relative to the scene size. For example, a floodlight 20 feet away illuminating a

small tabletop diorama could be represented as a directional light. Another example

is pretty much any scene lit by the sun, unless the scene in question is something

such as the inner planets of the solar system.

平行光是光源的最简单的模型。L和clight都是常量，除了clight可能被阴影衰减。平行光没有位置。当然，在空间中，真实的光源都有特定的位置。平行光是抽象的，当光源距离相对于场景尺寸非常大的时候，才工作良好。例如，一个照明灯在20尺以外的位置照明了一个小的桌面模型，此时可以称照明灯为平行光。另外一个例子在场景光中更常见就是太阳，除非这个场景是太阳系内行星之类的东西。

The concept of a directional light can be somewhat extended to allow varying the

value of clight while the light direction l remains constant. This is most often done to

bound the effect of the light to a particular part of the scene for performance or

creative reasons. For example, a region could be defined with two nested (one inside

the other) box-shaped volumes, where clight is equal to (0, 0, 0) (pure black)

outside the outer box, is equal to some constant value inside the inner box, and

smoothly interpolates between those extremes in the region between the two boxes.

平行光的概念可以稍微拓展，允许改变clight的值，而光的方向l保持不变。这通常是为了表现或创在性的原因，将灯光效果绑定到场景的特定部分。例如，可以定义一个区域有两个嵌套的盒子（一个在另一个内部）着色体，clight等于(0, 0, 0)（纯黑）以外的外框，等于一个常量在内盒内，两个盒子之间区域平滑插值。

5.2.2 Punctual Lights 点光源