# 18.2 法线贴图

法线贴图（normal map）是一种特殊的纹理，它的每个纹理元素存储的不是RGB数据，而是压缩后的*x*、*y*、*z*坐标（分别存储在R、G、B分量中）。这些坐标定义了一个法线向量；法线贴图的每个像素存储的都是这种法线向量。图18.3说明了法线贴图的实现方式。

****

**图18.3 法线向量存储在一个由向量T（*x*轴）、B（*y*轴）、N（*z*轴）构成的纹理空间坐标系中。向量T水平向右遍历法线贴图，向量B垂直向下遍历法线贴图，向量N垂直于法线贴图平面。**

为便于讲解，我们将采用24位图像格式，为每个颜色分量分配一个字节，故每个颜色分量的取值范围在0到255之间。（当然，也可以采用32位格式，只是alpha分量会被闲置，或者考虑用它存储一些其他的诸如高度贴图或高光贴图之类的标量值。在不需要压缩内存的情况下，也可以采用浮点格式，只是它占用的内存较多。）

**注意**：如图18.3所示，向量一般与z轴对齐。也就是，*z*坐标的值最大。所以，当把法线贴图视为一个彩色图像时（例如在Photoshop中查看法线贴图），它一般呈现为偏蓝的颜色。这是因为*z*坐标存储在蓝色通道中，该通道的数值最大，蓝色占据首要地位。

那么，我们如何将一个单位向量压缩到这种格式中呢首先，对于单位向量来说，每个坐标都在[−1,1]区间内。如果我们将该区间调整为[0,1]、乘以255并舍去小数，那结果就是一个在[0,255]区间内的整数。也就是，如果*x*是一个在[−1,1]区间内的浮点数，那么*f*(*x*)的整数部分就是一个在[0,255]区间内的整数。其中，*f*的定义为：



所以要将一个单位向量存入24位图像，我们只需要将函数*f*应用于每个坐标，并将坐标写入纹理贴图的对应颜色通道即可。

第二个问题是如何解压缩；也就是，当给出一个在[0,255]区间内的压缩纹理坐标时，如何将它恢复为[−1,1]区间内的原值。答案是将函数*f*简单的反转过来，只要稍加思索就可以看出：



也就是，如果*x*是一个在[0,255]区间内的整数，那么*f*-1(*x*)就是一个在[−1,1]区间内的浮点数。

我们不必自己实现压缩处理，因为我们可以用一个Photoshop 插件来完成从图像到法线贴图的转换工作。不过，当我们在像素着色器中对一个法线贴图进行采样时，必须自己实现反转处理，对法线贴图进行解压缩。当我们在着色器中对一个法线贴图进行采样时，可使用如下代码：

float3 normalT = gNormalMap.Sample(gTriLinearSam,pIn.texC);

颜色向量**normalT**将会包含规范化分量(*r*,*g*,*b*)，其中0≤*r*，*g*，*b*≤1。这样，该方法已经替我们完成了一部分解压缩工作（也就是除以255、将[0,255]区间内的整数变换为[0,1]区间内的浮点数）。我们只需要使用函数*g*(*x*)将每个分量从[0,1]区间调整到[−1,1]区间，即可完成的转换工作：



在代码中，我们将每个颜色分量代入该函数：

// 将每个分量从[0,1]解压到[-1,1].

normalT = 2.0f\*normalT - 1.0f;

这条语句可以运行，因为标量1.0会被自动扩展为向量(1,1,1)，以使整个表达式完整有效，并完成分量运算。

**注意**：Photoshop插件可以从[http://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop](https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop)下载。还有一些其他的工具也可以生成法线贴图，比如<http://www.crazybump.com/>。另外，还有一些工具可以根据高精度网格来生成法线贴图（参见<http://developer.nvidia.com/object/melody_home.html>）。

**注意**：如果你想使用压缩纹理格式存储法线贴图，请使用BC7（**DXGI\_FORMAT\_BC7\_UNORM**）格式，因为这种格式可以显著地减少压缩法线贴图带来的错误，所以质量最高。对于BC6和BC7格式来说，DirectX SDK中一个叫做“BC6HBC7EncoderDecoder11”的示例可以用来将纹理文件转换为BC6或BC7。