现在,我们将描述一组接口和类,这些接口和类使我们能够将纹理合并到材料模型中.回想一下,第9章中的材料都是基于描述其特性(漫反射率,光泽度等)的各种参数.由于现实世界中的材料特性通常会在整个表面上发生变化,因此必须能够表示这种空间变化.在pbrt中,纹理抽象用于此目的.它们的定义方式是将图案生成方法与反射模型实现分开,从而可以轻松地以任意方式组合它们,从而使创建各种外观变得更加容易.

在pbrt中,纹理是一个相当笼统的概念:它是将某个域(例如,曲面的（u，v）参数空间或（x，y，z）对象空间)中的点映射到其他某个域中的值的函数(例如,频谱或实数).系统中有多种纹理类的实现方式.例如,pbrt具有表示零维函数的纹理,这些纹理返回一个常数,以适应所有位置具有相同参数值的曲面.图像贴图纹理是（s，t）参数值的二维函数,这些参数值使用像素值的2D数组来计算特定点的纹理值(它们在10.4节中进行了介绍).甚至还有基于其他纹理函数计算出的值来计算值的纹理函数.

纹理可能是最终图像中高频变化的来源.图10.1显示了由于纹理而导致严重锯齿的图像.尽管可以使用第7章中的非均匀采样技术来减少这种混叠的视觉影响,但对此问题的更好解决方案是实现纹理函数,该纹理函数根据采样率来调整其频率含量.对于许多纹理函数而言,以这种方式计算出频率分量的合理近似值和抗锯齿并不太困难,并且比通过增加图像采样率来减少混叠效果要有效得多.

本章的第一部分将讨论纹理混叠的问题以及解决该问题的一般方法.然后,我们将描述基本的纹理接口,并通过一些简单的纹理函数来说明其用法.在本章的其余部分中,我们将介绍各种更复杂的纹理实现,演示了在此过程中使用多种不同的纹理抗锯齿技术.

10.1 采样和抗锯齿

第7章采样任务令人沮丧,因为已知混叠问题从一开始就无法解决.无论图像采样率有多高,几何边缘和硬阴影的无限频率内容都可以确保最终图像中出现混叠.(我们唯一的安慰是,使用足够数量的适当放置的样本，可以将这种残留混叠的视觉影响降低到无可争议的水平.)

幸运的是，对于纹理，从一开始就没有那么困难：要么通常有一个方便的解析可以使用纹理函数的形式，从而可以在对其进行采样之前去除过高的频率，或者在评估函数时要小心，以免一开始就不引入高频。当在纹理实现中仔细解决了这个问题时（如本章其余部分所述），通常无需为每个像素提供一个以上的样本即可渲染图像而无需纹理混叠.

为了消除纹理函数的混叠,必须解决两个问题:

1. 必须计算纹理空间中的采样率.从图像分辨率和像素采样率可以知道屏幕空间采样率,但是这里我们需要确定场景表面上的最终采样率,以便找到对纹理函数进行采样的速率.
2. 在给定纹理采样率的情况下,必须应用采样理论来指导频率变化不超过采样率所表示的纹理值的计算(例如,从纹理函数中去除超出奈奎斯特极限的多余频率).

本节其余部分将依次解决这两个问题.

10.1.1 寻找纹理采样率

考虑任意定义在曲面位置上的函数,.如果我们忽略了可见性问题带来的复杂性-另一个物体可能会在附近的图像样本处遮挡该表面,或者该表面在图像平面上的程度可能有限-该纹理函数也可以表示为点上的函数,其中是将图像点映射到表面上的点的函数.因此,给出了在图像位置处看到的纹理函数的值.

作为该想法的简单示例,请考虑将二维纹理函数应用于垂直于z轴并在世界空间点（0，0，0），（1、0 ，0），（1、1、0）和（0、1、0）.如果放置正交摄影机,并沿z轴向下看,以使四边形精确填充图像平面,并且如果将四边形上的点映射到2D纹理坐标,则

那么和屏幕像素之间的关系很简单:

整体图像分辨率为(图10.2).因此,给定图像平面中一个像素的采样间隔,纹理参数空间中的采样间隔为,并且纹理函数必须移出所有高于给定采样率的高频细节.

像素坐标和纹理坐标之间的这种关系,以及因此它们的采样率之间的关系,是确定纹理函数中允许的最大频率内容的信息的关键位.举一个稍微复杂的例子,给定一个在顶点具有纹理坐标的三角形,并使用透视投影进行观察,就可以分析地找到图像平面上各个采样点的和的差异.这是图形处理器中基本纹理贴图抗锯齿的基础.

对于更复杂的场景几何,相机投影以及到纹理坐标的映射,要精确确定图像位置和纹理参数值之间的关系要困难得多.幸运的是,对于纹理抗锯齿,我们不需要能够针对任意评估,而只需要找到像素样本位置变化与纹理样本位置变化之间的关系即可.在图像上的特定点.这种关系由该函数的偏导数和y给出.例如,这些可用于查找值的一阶近似值,

如果这些偏导数相对于距离和变化缓慢,则这是一个合理的近似值.更重要的是,这些偏导数的值分别针对在x和y方向上偏移一个像素提供了纹理采样位置变化的近似值,因此可以直接得出纹理采样率.例如,在前面的四边形示例中,以及.

通常情况下,找到这些偏导数的值的关键在于第2.5.1节中定义的RayDifferential结构. 通过Camera::GenerateRayDifferential()方法为每个摄像机射线初始化此结构.它不仅包含实际在场景中跟踪的光线,而且还包含另外两条光线,其中一条光线相对于摄影机光线水平偏移一个像素样本,另一偏移垂直于一个像素样本.所有的几何射线相交测试仅使用主摄影机射线进行计算;辅助射线将被忽略(这很容易做到,因为RayDifferential是Ray的子类).

在这里,我们将使用偏移射线来估计从图像位置到世界空间位置的映射的偏导数以及从映射和偏导数参数坐标,给出世界空间位置和的偏导数以及参数坐标偏导数,和.在第10.2节中,我们将看到如何将它们用于基于p或（u，v）计算任意数量的屏幕空间导数，进而计算这些数量的采样率。 这些偏导数在交点处的值存储在SurfaceInteraction结构中。 它们被声明为可变的，因为它们是在采用该对象的const实例的方法中设置的.

计算这些估计值的关键是假设表面相对于阴影点处的采样率局部平坦。 实际上这是一个合理的近似值，很难做得更好.因为光线追踪是一种点采样技术,所以我们在追踪的光线之间没有关于场景的其他信息.对于高度弯曲的表面或轮廓边缘,此近似值可能会分解,尽管实际上很少会引起明显的误差.

对于这种近似,我们需要一个平面,该平面通过与主切线相交的点p相交.该平面由隐式方程式给出

其中,且.然后,我们可以计算辅助射线和与该平面之间的交点和(图10.3).这些新点基于正向差异,近似表示表面和上位置的偏导数:

由于差异光线在每个方向上都偏移一个像素样本,因此无需将这些差异除以值,因为.

**注意:上述内容可能存在错误.因为根据曲面的切平面近似定理**:,而.(参考教材939和940页内容)

3.1.2节中描述的射线平面相交算法给出值,其中由原点和方向所描述的射线与由所描述的平面相交:

为了计算两条辅助射线的该值,首先要计算平面的d系数.无需计算a,b和c系数,因为它们在n中可用.然后,我们可以直接应用公式.

利用位置和,可以利用表面的偏导数和形成a（不一定是正交的）这一事实来找到它们各自的（u，v）坐标的近似值。 平面上的坐标系，并且根据该坐标系的辅助交点的坐标是相对于（u，v）参数化的坐标（图10.4）。 给定平面上的位置，我们可以通过以下方式计算其相对于坐标系的位置:

或等价为

对于辅助点或之一的线性方程组的解给出相应的屏幕空间偏导数或.

这个线性系统包含三个方程,两个未知数,也就是说,它是过度约束的.我们需要注意,因为其中一个方程可能是简并的-例如,如果∂p/∂u和∂p/∂v在xy平面上,使得它们的z分量都为零,那么第三个方程将简并.因此,我们想使用两个不退化系统的方程来求解方程组.一种简单的方法是取∂p/∂u和∂p/∂v的叉积,查看结果的哪个坐标具有最大的幅度,并使用其他两个.他们的叉积已经在n中可用,因此使用这种方法很简单.即使所有这些,线性系统也可能没有解（通常是由于偏导数未在平面上形成坐标系）.在那种情况下,所有可以做的就是返回任意值.

10.1.2 过滤纹理函数

有必要删除纹理函数中超过纹理采样速率的奈奎斯特极限的频率.目标是以尽可能少的近似值计算理想纹理重采样过程的结果，该结果表明，要在不产生混叠的情况下评估,我们必须首先对其进行频带限制,然后通过将其与Sinc滤波器进行卷积去除超出奈奎斯特极限的频率:

然后,应将带限函数与要在其上评估纹理函数的屏幕上点居中的像素过滤器卷积:

这为投影到屏幕上的纹理提供了理论上理想的值.

实际上，可以对此过程进行许多简化，而视觉质量几乎没有下降.例如,盒滤波器可以用于带宽限制步骤，而第二步通常被完全忽略，有效地发挥了作用，就好像像素滤波器是盒滤波器一样，这使得可以在纹理空间中完全进行抗锯齿工作。 并大大简化了实施。 第10.4.5节中的EWA过滤算法是一个显着的例外，因为它假定了高斯像素过滤器.

盒式滤波器易于使用,因为它可以通过计算适当区域上纹理函数的平均值来进行分析性应用。直观地讲，这是解决纹理过滤问题的一种合理方法，可以直接为许多纹理函数进行计算。实际上，在本章的其余部分中，我们将经常使用盒式滤波器对样本之间的纹理函数值求平均，并非正式地使用术语“滤波器区域”来描述要求平均值的区域。这是过滤纹理函数时最常用的方法。 在许多情况下，即使是盒式过滤器也有其所有缺点，也可以为纹理过滤提供可接受的结果。一个有用的因素是这样的事实，即通常在每个像素中采集许多样本。因此，即使每个纹理中使用的滤波纹理值都不理想，一旦通过像素重建滤波器对其进行滤波，最终结果通常也不会受到太大影响。

使用盒式滤波器对纹理函数进行滤波的另一种方法是，观察到理想的Sinc滤波器的作用是使低于奈奎斯特极限的频率分量不变地通过，但去除超过该极限的频率。因此，如果我们知道纹理函数的频率含量（例如，如果它是一项的和，则每个频率项具有已知的频率含量），那么如果我们将高频项替换为其平均值，则可以有效地进行sinc预过滤器的工作。这种方法称为钳位，是基于10.6节中的噪声函数对纹理进行抗锯齿的基础。

最后，对于不容易应用这些技术的纹理函数，最后的选择是超级采样-在靠近主评估点的多个位置对函数进行评估和过滤，从而提高了纹理空间中的采样率。如果使用盒式过滤器过滤这些样本值，则等效于对该函数的值求平均值。如果纹理函数的计算复杂，则此方法可能会很昂贵，并且与图像采样一样，可能需要非常大量的采样才能消除混叠。尽管这是一种蛮力解决方案，但它仍然比增加图像采样率更有效，因为它不会增加在场景中跟踪更多光线的成本。

10.1.3 用于镜面反射和投射的射线微分

考虑到光线微分在寻找用于相机光线的纹理抗混叠的滤镜区域方面的有效性，扩展该方法以使确定通过镜面反射或折射间接看到的对象的纹理空间采样率成为可能是有用的。 例如，在镜子中看到的对象也不应比直接可见的对象具有更多的纹理混叠。 Igehy（1999）针对如何找到适当的微分射线以进行镜面反射和折射开发了一种精巧的解决方案，这是pbrt中使用的方法。

图10.5说明了用于镜面反射和透射的适当纹理过滤可以产生的差异。图10.5（a）显示了平面上的玻璃球和镜像球，其纹理贴图包含高频分量。射线微分确保通过球的反射和折射看到的纹理图像没有混叠伪影。图10.5（b）和（c）展示了玻璃球中反射的近视图。图10.5（b）的反射和透射光线没有光线差异，图10.5（c）的光线具有差异。 （所有图像均以每个像素一个样本进行渲染。）消除了左侧图像中的混叠错误，而不会过度模糊纹理。为了计算在表面相交点处反射或透射的射线微分，我们需要对撞击表面的射线微分中的两个偏移射线在交点处跟踪的光线进行近似计算（图10.6）。

主射线的新射线是由BSDF计算的，因此这里我们只需要计算rx和ry差分的出射射线.

对于反射和折射,很容易找到每条微分射线的起源.SurfaceInteraction :: ComputeDifferentials()方法先前已计算出表面位置相对于图像平面和上（x，y）位置变化的近似值.将这些偏移量添加到主光线的交点可以得出新光线的近似原点.如果入射光线没有微分,则不可能计算反射光线的微分,因此将跳过此步骤.

找到这些射线的方向有些棘手.Igehy（1999）观察到，如果我们知道反射方向相对于像素样本在图像平面在x和y方向上的位移有多少变化，则可以使用此信息来近似偏移光线的方向。 例如，x中的射线偏移方向为

从公式（8.6）中回想起,对于一般世界空间表面的法线和出射方向,理想镜面反射的方向为

幸运的是，该表达式的偏导数很容易计算:

使用点积的属性,可以证明

10.2 纹理坐标生成

本章中几乎所有的纹理都是采用2D或3D坐标并返回纹理值的函数。有时，有明显的方法可以选择这些纹理坐标。 对于参数化曲面（例如第3章中的二次曲面），曲面具有自然的2D（u，v）参数化，对于所有曲面，着色点p是3D坐标的自然选择。

在其他情况下，没有自然参数设置，或者自然参数设置可能是不希望的。例如，球极附近的（u，v）值严重失真。同样，对于任意细分曲面，也没有简单的通用方法来分配纹理值，以使整个[0，1] 2空间被连续覆盖并且没有变形。实际上，创建具有低失真的复杂网格的平滑参数化是计算机图形学研究的一个活跃领域。本节从介绍两个抽象基类（TextureMapping2D和TextureMapping3D）开始，它们提供了用于计算这些2D和3D纹理坐标的接口。然后，我们将使用此接口实现许多标准映射（图10.7显示了其中的许多映射）。纹理实现会适当地存储一个指向2D或3D映射函数的指针，并使用它来计算每个点的纹理坐标。因此，可以轻松地向系统添加新的映射，而无需修改所有Texture的实现，并且可以将不同的映射用于与同一表面关联的不同纹理。在pbrt中，我们将使用2D纹理坐标由（s，t）表示的约定；这有助于弄清楚底层表面的固有（u，v）参数化与用于纹理化的（可能不同）坐标值之间的区别。 TextureMapping2D基类具有单个方法TextureMapping2D :: Map（），该方法在着色点处具有SurfaceInteraction，并通过Point2f返回（s，t）纹理坐标。它还返回有关dstdx和dstdy参数中像素x和y坐标的（s，t）变化的估计值，以便使用映射的纹理可以确定（s，t）采样率并相应地进行过滤。

10.2.1 2D(u,v)映射

最简单的映射使用SurfaceInteraction中的（u，v）坐标来计算纹理坐标。 它们的值可以偏移，并可以使用每个维度中用户提供的值进行缩放。

10.2.2 球面映射

另一个有用的映射有效地将球体包裹在对象周围。每个点都沿着矢量从球体中心到该点投影，直至球体表面。 在那里，使用了球形的（u，v）映射。 SphericalMapping2D存储在执行此映射之前应用于点的转换。 这有效地允许映射球相对于对象任意定位和定向。

10.5 固体和程序纹理

一旦人们开始将2D纹理函数所使用的（s，t）纹理坐标视为可以通过任意函数计算而不仅仅是从表面的参数坐标计算出的量,自然就可以将纹理函数泛化为3D域（通常称为实体纹理），而不只是2D（s，t）。实体纹理特别方便的一个原因是，所有对象都具有自然的3D纹理映射-对象空间位置。对于不具有自然2D参数化的对象（例如，三角形网格和隐式曲面）以及具有扭曲的参数化（例如，靠近球体的极点）的对象进行纹理化，这是一个很大的优势。为准备这个想法，第10.2.5节定义了一个通用的TextureMapping3D接口来计算3D纹理坐标以及TransformMapping3D实现。但是，实体纹理会带来一个新问题：纹理表示。 3D图像贴图占用了大量的存储空间，比2D纹理贴图要难得多，后者可以从照片中提取或由艺术家绘画。因此，过程纹理化（可以执行程序以在场景表面上任意位置生成纹理值的想法）在开发实体纹理化的同时开始使用。过程纹理的一个简单示例是过程正弦波。如果我们想使用正弦波进行凹凸贴图（例如，模拟水中的波），则在点网格处预先计算函数的值然后将其存储在图像图中将是效率低下且可能不准确的。相反，根据需要在表面上的点上评估sin（）函数会更有意义。例如，如果我们找到一个描述实木块中谷物颜色的3D函数，那么我们就可以生成似乎是用木头雕刻而成的复杂物体的图像。多年来，随着技术的发展以描述越来越复杂的程序表面，程序纹理已经在应用中得到了很大的发展。程序化纹理具有许多有趣的含义。首先，它可以通过减少存储大型，高分辨率纹理贴图的需求来减少渲染所需的内存。另外，程序阴影为潜在的无限细节提供了希望。当观察者接近物体时，在阴影处评估纹理功能，这自然会导致可见数量的细节。相反，当观看者离图像纹理贴图太近时，它们会变得模糊。另一方面，与使用图像贴图相比，过程纹理外观的细微细节要难得多。

程序纹理的另一个挑战是抗锯齿.程序纹理的评估通常很昂贵,而且无法完整描述特征集的点样本,就象图像地图一样.因为我们想在纹理函数中删除高频信息,然后再对其进行采样,因此我们需要了解在此过程中采取的各个步骤的频率内容,以便避免引入高频.尽管这听起来令人生畏,但仍有一些技巧可以很好地解决此问题.

10.5.1 UV纹理

我们的第一个程序纹理将表面的（u，v）坐标转换为Spectrum的红色和绿色分量（图10.17）。 例如，在调试新Shape的参数化时，它特别有用。 它在textures / uv.h和textures / uv.cpp中定义。

10.5.2 棋盘格

棋盘格是规则的程序纹理（图10.18）.（s，t）纹理坐标用于将参数空间分解为正方形区域，这些正方形区域用交替的图案阴影。 此处的实现方式不仅支持在两种固定颜色之间切换的棋盘，还允许用户传入两种纹理来为交替区域着色。 传统的黑白棋盘格是通过传递两个ConstantTextures获得的。 它的实现在textures / checkerboard.h和textures / checkerboard.cpp文件中。

为简单起见，检查功能的频率为（s，t）空间中的1：检查在每个方向上的宽度为一个单位。 TextureMapping2D类始终可以使用（s，t）坐标的适当比例来更改有效频率。

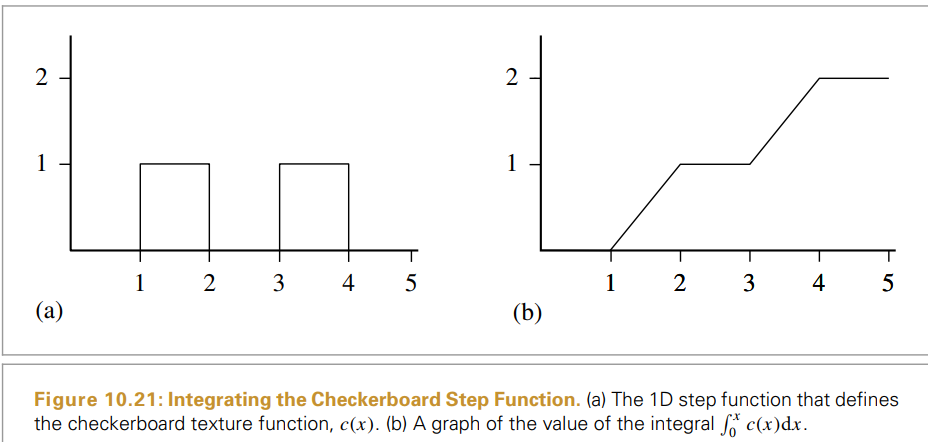
棋盘格非常适合用来说明在程序纹理的各种抗锯齿方法之间的取舍。 此处的实现既支持简单点采样（无抗锯齿），又支持在滤波器区域上评估的封闭式盒式滤波器。 本节末尾的图10.23中的图像序列显示了这些方法的结果。 aaMethod枚举器选择使用哪种方法。

评估例程执行通常的纹理坐标和微分计算，然后使用适当的片段来计算抗锯齿的棋盘值（如果选择了点采样，则不进行抗锯齿）。

最简单的情况是忽略抗锯齿，仅在该点对棋盘纹理进行点采样。 对于这种情况，在从TextureMapping2D获得（s，t）纹理坐标之后，对该（s，t）位置的整数棋盘格坐标进行计算，相加并检查奇数或偶数奇偶校验以确定两个纹理中的哪一个 评估。

鉴于点采样棋盘纹理中的混叠效果有多差，我们将投入一些精力来适当地消除混叠。 最简单的情况是整个过滤器区域都位于单个检查内（图10.19）。 在这种情况下，我们只需要确定我们属于哪种检查类型并进行评估即可。 只要该检查中的“ Texture”本身能够进行适当的抗锯齿处理，此情况下的结果就将得到适当的抗锯齿处理。

通过计算边界框并查看其范围是否在同一支票内,可以很容易地检查整个过滤器区域是否在单个棋盘内。在本节的其余部分中,我们将使用偏导数等给定的过滤器区域的轴对齐边界框作为要过滤的区域，而不是 尝试像EWA过滤器一样过滤由偏导数定义的椭圆（图10.20）。 尽管在某种程度上增加了滤波值的模糊性，但这简化了此处的实现。 后面的变量ds和dt在每个方向上保持滤镜宽度的一半，因此，经过滤波的总面积从（s-ds，t-dt）到（s + ds，t + dt）。



否则，查找方法将通过首先计算一个浮点值来近似过滤值，该浮点值指示过滤区域的哪一部分覆盖两种检查类型中的每一种。这等效于在滤波器区域上计算二维步进函数的平均值，当我们在tex1中时该值为2，而在tex2中则为1。图10.21（a）显示了棋盘功能c（x）的图形，定义为

给定平均值,我们可以根据可见的每个滤镜区域的比例在两个子纹理之间进行混合.一维棋盘格函数c（x）的积分可用于在某种程度上计算该函数的平均值.查看该图发现，

为了在二维中计算阶跃函数的平均值,我们分别计算每个1D方向上棋盘的积分，以计算其在滤波器区域上的平均值.

10.5.3 固体棋盘格

上一节中的Checkerboard2DTexture类在参数空间中的对象周围包装了一个棋盘图案。我们还可以基于3D纹理坐标定义实体棋盘格图案，以使对象看起来像是从3D棋盘格立方体雕刻而成的（图10.22）。 像2D变体一样，此实现基于查找位置在纹理函数之间进行选择。 请注意，这两个纹理本身不必是实体纹理。 Checkerboard3DTexture仅根据点的3D位置在它们之间进行选择。

忽略抗锯齿，查看点p是否在3D检查器区域内的基本计算是

Checkerboard3DTexture没有任何内置的抗锯齿支持，因此其实现时间很短.

10.6 Noise

为了编写描述复杂表面外观的实体纹理，能够在过程中引入一些受控变化是有帮助的。考虑由单个木板制成的木地板；每块木板的颜色可能与其他颜色略有不同。或者考虑一个风吹拂的湖泊；我们可能希望整个湖面都具有类似振幅的波，但我们不希望它们在湖的所有部分上都是均匀的（例如，如果它们是由正弦波之和构成的，则可能是均匀的）。对纹理中的这种变化进行建模有助于使最终结果看起来更逼真。

开发此类纹理的一个困难是，渲染器在不规则分布的一组点上评估表面的纹理功能，其中每个评估完全独立于其他评估。因此，程序纹理必须通过回答关于所有这些点上的模式值的查询来隐式定义复杂模式。相反，显式模式描述方法是通过PostScript®语言实现的，例如，该语言使用一系列绘图命令来描述页面上的图形。隐式方法引入的一个困难是，纹理不能仅在被评估为引入随机性的每个点上调用RNG :: UniformFloat（）：因为每个点将具有与其邻居完全不同的随机值，因此没有一致性在生成的模式中将是可能的。

解决此问题的一种优雅方法是应用所谓的噪声函数，该问题是将可控制的随机性引入图形的过程纹理中。通常，图形中使用的噪声函数是平滑变化的函数，Rn→[-1，1]，至少n = 1，2，3，没有明显的重复。 实际噪声功能的最关键特性之一是它受到已知最大频率的限制。

这样就可以控制由于噪声函数而添加到纹理的频率含量，从而不会引入比奈奎斯特极限所允许的频率高的频率。

已开发的许多噪声函数都是基于R3上的整数晶格的思想构建的。首先，将值与空间中的每个整数（x，y，z）位置相关联。然后，给定空间中的任意位置，找到八个周围的晶格值。然后对这些晶格值进行插值以计算特定点的噪声值。可以将这种想法推广或限制为更多或更少的尺寸d，其中晶格点的数量为2d。此方法的一个简单示例是值噪声，其中-1和1之间的伪随机数与每个晶格点相关联，并且实际噪声值通过三线性插值法或更复杂的样条插值法计算得出，这可以给出更平滑的结果通过避免从一个晶格单元移动到另一个晶格单元时的导数不连续性。

对于这样的噪声函数，给定整数（x，y，z）晶格点，必须有可能以始终将相同值与每个晶格点相关联的方式有效地计算其参数值。因为存储所有可能的（x，y，z）点的值是不可行的，所以需要一些紧凑的表示形式。一种选择是使用哈希函数，该函数先对坐标进行哈希处理，然后再从固定大小的预计算伪随机参数值表中查找参数。

10.6.1 Perlin噪声

在pbrt中,我们将实现Ken Perlin（1985a，2002）引入的噪声函数;因此,这被称为Perlin噪声.在所有（x，y，z）整数晶格点处，其值为零。它的变化来自每个晶格点处的梯度向量,这些向量指导在点之间进行平滑函数的插值（图10.24）.该噪声函数具有上述噪声函数的许多期望特性,计算效率高,并且易于实现.图10.25显示了其在球体上呈现的值.

该实现首先计算包含给定点的单元格的整数坐标以及该点距下单元格角的分数偏移：

接下来，它调用Grad（）以获取八个权重值，该点位于其中的单元格的每个角对应一个。 Grad（）使用单元格索引来索引表； 为了提高效率，我们将所有会使组件超出表大小的高位清零，以确保所有索引都在此表的范围内。 （要使此技巧起作用，表的大小必须为2的幂-否则，将需要昂贵的整数模运算来代替按位“和”。）

每个整数晶格点都有一个与之关联的梯度向量。 梯度向量对单元格内任何点的影响是通过计算从梯度角到查找点的向量的点积与梯度向量获得的（图10.26）； 这由Grad（）函数处理。 注意，除左下角以外的其他角上的向量可以轻松地基于该向量进行增量计算。

通过索引到整数值预先计算的表NoisePerm中，可以找到特定整数晶格点的梯度向量。 格点表值的四个低阶位决定了与之关联的16个梯度向量中的哪一个。 在预处理步骤中，此大小为NoisePermSize的表填充了从0到NoisePermSize-1的数字，然后随机排列。 然后复制这些值，形成一个大小为2 \* NoisePermSize的数组，该数组连续两次保存该表。 该表的第二个副本使以下代码中的查找效率更高。

10.6.2 随机波尔卡圆点

噪声函数的基本用途是作为圆点纹理的一部分，该纹理将（s，t）纹理空间划分为矩形单元（图10.27）。 每个单元格中有50％的机会在其中有一个点，并且这些点随机地放置在其单元格中。 DotsTexture具有通常的2D映射功能以及两个Texture，一个用于点外表面的区域，另一个用于内面的区域。 它在文件textures / dots.h和textures / dots.cpp中定义。

10.6.3 噪声成语与光谱合成

噪声是一个带限函数的事实意味着，可以通过缩放评估它的域来调整其频率含量。 例如，如果Noise（p）具有一些已知的频率含量，则Noise（2 \* p）的频率含量将是原来的两倍。 这就像sin（x）和sin（2x）的频率含量之间的关系一样。 该技术可用于创建具有所需变化率的噪声函数。

对于程序纹理的许多应用而言，在多个比例上进行变化很有用，例如，为基本噪声函数添加更精细的变化。 解决噪声的一种有效方法是通过频谱合成来计算模式，其中复杂函数由另一个函数的贡献之和定义：

10.6.4 凹凸不平的纹理

fBm和湍流功能作为凹凸贴图的随机变化源特别有用。FBmTexture是使用FBm（）计算偏移量的Float值纹理，而WrinkledTexture使用Turbulence（）来计算偏移量。它们在图10.30和10.31中进行了演示，并在textures / fbm.h，textures / fbm.cpp，textures / wrinkled.h和textures / wrinkled.cpp中实现。

10.6.5 风浪

fBm的应用可以合理地令人信服地表示波浪（Ebert等，2003）。 图1.11和4.1在这些场景中使用这种纹理作为水。 此纹理基于两个观察结果。 首先，在风吹拂的湖泊表面（例如），某些区域相对较平整，而某些区域则更不稳定。 这种影响来自风的强度随区域的自然变化。 其次，表面上单个波浪的整体形式可以很好地描述，该模型由基于fBm的波型按风强度定标。 该纹理在textures / windy.h和textures / windy.cpp中实现。

10.6.6 大理石

噪声函数的另一种经典用法是在使用另一个纹理或查找表之前先扰动纹理坐标。例如，可以通过将大理石材料建模为一系列分层的地层，然后使用噪声扰动用于在层间寻找值的坐标来制造大理石的传真。 本节中的MarbleTexture实现了这种方法。 图10.32说明了此纹理的思想。 在左侧，使用球体上该点的y坐标直接索引大理石图层。 在右侧，fBm已用于扰动y值，从而引入变化。 该纹理在textures / marble.h和textures / marble.cpp中实现。