本章重点介绍我在设计噪声函数的新改进版本时所做出的决策。我进行此重新设计的原因有三个：（1）使Noise更适合逐步过渡到硬件；（2）以某些重要方式改善Noise函数的视觉特性；（3）引入一个单一的标准版本,在所有硬件和软件平台上将返回相同的值。

本章的结构如下：首先，我描述噪声函数本身的目标，并简要回顾一下原始实现。然后，我将讨论在实施改进的“噪声”以提高其质量并减少视觉伪像时所做出的特定选择。最后，我讨论了逐步迁移到硬件所需的条件。目标是允许噪声在越来越少的指令中实现，因为连续几代的GPU都遵循一条通往功能更强大的指令的道路。

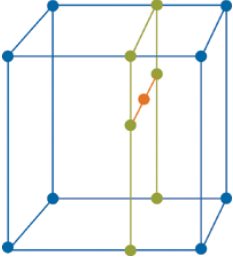
5.1 噪声函数

噪声功能的目的是在(三维空间)上提供有效实施且可重复的伪随机信号，该信号受频带限制（其大部分能量集中在一个空间频率附近）并且在视觉上各向同性（统计旋转-不变）。

总体思路是创建一种与完全随机的信号（即白噪声）具有相同效果的信号，该信号已通过低通滤波器滤除，从而消除了所有高空间频率。一个例子是沙丘的丘陵和山谷逐渐上升和下降，如图5-1所示。

5,2 原始实现

最初的实现方式于1983年首次使用，并于1985年首次发布（Perlin 1985），它使用以下算法定义了任意点的噪声：

1. 在具有整数坐标的空间的每个点上,分配零值和一个从散列的伪随机梯度,
2. 将的坐标定义为整数值加上小数部分:.考虑围绕该点的单位立方体的八个角:.
3. 通过这八个点拟合Hermite样条曲线,并使用和作为内插值,在处评估该样条曲线.如果使用表查找来预定义Hermite三次混合函数,则此插值只需要七个标量线性插值:例如,四个,两个,z一个,如图5-2中所示。

这种方法相当有效。“噪声”函数的二维切片如图5-3所示。

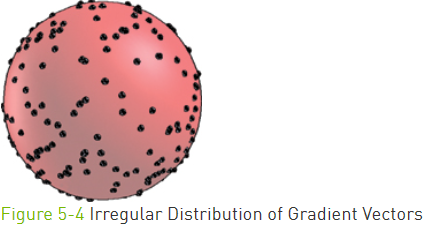
5.3原始实施的缺陷

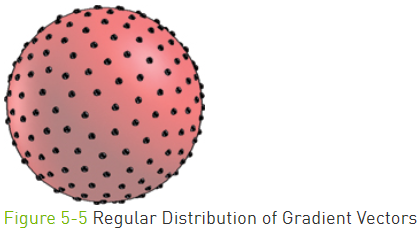
不幸的是，Noise具有以下缺陷：

每个维度上的插值函数选择为,其二阶导数包含非零值.当采用噪声导数时（例如进行凹凸贴图时），这可能会导致视觉伪影.

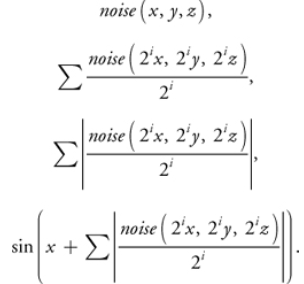
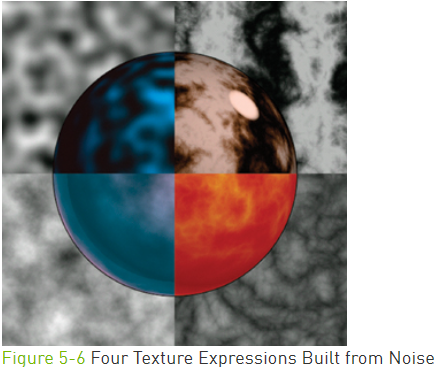
来自的梯度散列产生了对3D球体上播出的256个梯度向量的伪随机集合的查找.这种分布的不规则性会在产生的噪声函数中产生不必要的高频率.请参阅图5-4.

回想起来，我意识到我应该在预处理步骤中应用松弛算法来选择伪随机梯度方向，将这些点在球体上彼此排斥，以形成更均匀的，类似于泊松的分布，这将减少不需要的高空间频率，其结果类似于图5-5所示。





最近，我意识到在视觉上甚至不需要这么多不同的梯度方向。相反，与许多不同的梯度方向相反，使梯度方向的统计分布均匀更为重要。这与感知研究相一致，该研究表明尽管人类注意力集中的视觉对纹理特征方向的统计异常高度敏感，但是我们的视觉对方向的粒度相对不敏感，因为我们的低级视觉系统将转换足够密集的视觉 一组离散的方向转变成等效的连续信号（Grill Spector等人1995）。人类视觉系统在很早的阶段就完成了这种转换，远低于意识的阈值。

 假设使用噪声实现时，梯度方向在统计上是均匀分布的，那么构建可创建外观复杂纹理的过程表达式非常简单。例如，图5-6依次显示了根据Noise构建的四个更复杂的纹理表达式。

5.4 改善噪声

对噪声函数的改进在两个特定领域：内插值的性质和伪随机梯度场的性质。

正如我在Perlin 2002中讨论的那样，改进插值的关键只是消除二阶不连续性。选择原始三次插值是因为它在和时都具有零导数。不幸的是，它的二阶导数是，在或时都不为零。当在凹凸贴图中使用噪声时，这会导致出现伪像。因为在凹凸贴图的表面上照明的效果随相应的高度函数的导数而变化，所以二阶导数间断是可见的。

通过切换到五阶插值器可解决此问题：，它的一阶和二阶导数在和时均为零。

这两个插值之间的差异可以在图5-7中看到。绿色曲线是旧的三次插值，在和时具有二阶不连续性。蓝色曲线是新的第五阶插值，在和时具有二阶连续性。

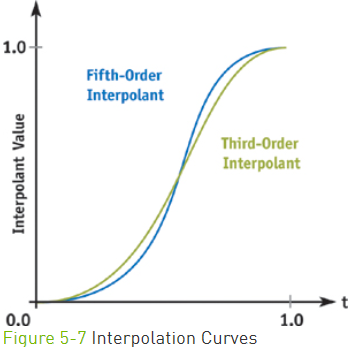
关于图形加速器的一件好事是，您可以通过一维纹理查找表执行一维插值。当您采用这种方法时，进行高阶插值没有额外的成本。 在实践中，我发现纹理表的长度为256，足够了，精度为16位就足够了。甚至具有10位尾数的半精度浮点（fp16）也足够。

图5-8显示了噪声置换超二次方的两个不同渲染。左边的使用旧的三次插值； 右边的使用较新的五阶插值。视觉上的改善可以看作是形状正面中4x4网格状伪像的减少。

另一个改进是仅用12个伪随机梯度替换了256个伪随机梯度，其中包括以原点为中心的立方体的边缘中心：（0，±1，±1），（±1、0，± 1）和（±1，±1、0）。如图5-9所示，使用旧方法和新方法从Noise的实现中获取的平面切片的并排比较可以看出，这导致了“斑点状”分布的减少。

进行此改进的原因是，这12个渐变方向中的任何一个都不离其他方向太近，因此它们永远不会聚集得太近。正是由于这种不必要的相邻梯度堆积而导致原始噪声实现的某些部分出现了斑点。

这种方法的另一个好处是，它可以避免与评估梯度函数相关的许多乘法，因为与的内积可以是简单地计算为.

为了使散列到这组梯度中的过程与最快的硬件实现兼容,实际上定义了16个梯度,因此散列函数可以简单地返回一个随机的4位值.额外的四个渐变仅重复（1、1、0），（-1、1、0），（0，-1、1）和（0，-1，-1）。

5.5 如何在像素着色器中产生良好的伪噪声

完全在像素着色器中具有某种合理的体积噪声基元会很好，但是此方法的直接实现需要大量指令。我们如何利用当今的像素着色器的功能，以很少的指令数量实现噪声的合理近似值？ 假设我们愿意使用分段三线性插值而不是更高的多项式插值（因为当今的GPU提供了三线性插值的直接硬件加速），并且我们愿意忍受使用三线性插值时出现的较小的梯度不连续性 。

一种解决方案是将全部噪声放入三线性插值纹理中。 不幸的是，这将非常占用空间。 例如，从（0，0，0）到（72，72，72）的多维数据集采样噪声，每个线性单位距离包含三个样本，最后端点包含一个样本，或者每个维度包含（3 x 72）+1个样本，则需要217 x 217 x 217 = 10,218,313索引位置，这不方便。

但是，今天的GPU可以在较小体积的体积上执行快速三线性插值的事实可以通过其他方式加以利用。例如，以相同方式采样的8x8x8体积在每个方向上需要（3 x 8）+ 1个采样，或者仅25x25x25 = 15,625个索引位置。我一直在做的事情是通过将多个引用定义到此类中等大小的存储卷中来定义噪声。

那么如何从个多维数据集中得到个多维数据集呢？诀窍是分两个步骤完成：

1. 在该空间中填充个样本的副本，这些样本是环形拼贴。换句话说，构造此样本，以使其形成无限重复的连续模式，顶部与底部无缝连接，左侧与右侧无缝连接，背面与前端无缝连接。当这些样品以立方砖的形式堆叠在空间中时，它们将产生连续但重复的，充满空间的纹理。
2. 使用以九倍的比例放大的同一采样模式，作为一种低频模式，该模式可以改变这种精细的砖瓦，使得精细模式不再显得重复。

详细信息如下：而不是将此重复磁贴中的值定义为实数，而是将它们定义为复数。请注意，这会使表的存储成本加倍（在这种情况下，将存储成本增加到31,250个索引位置）。如果我们为表中的每个数字标量使用16位数量，则存储卷所需的总内存成本为31,250x2x2字节= 125,000字节。

评估低频和高频噪声纹理。然后将低频纹理的实分量用作旋转相移，如下所示：假设我们从低频纹理中检索了的复数值，而从高频纹理中检索了的复数值。我们通过评估以下表达式，使用低频纹理来旋转高频纹理的复数值：

图5-10中的三幅图像依次显示（a）通过高频纹理的平面切片的实数部分，（b）通过低频纹理的平面切片的实数部分和（c）通过将它们组合而获得的最终噪声纹理。

5.6 在不看相邻顶点的情况下进行颠簸

一旦可以直接在像素着色器中实现“噪波”，就可以按以下方式实现凹凸贴图：

1. 考虑一个点和一些通过噪声函数组合在像素着色器中定义的函数.
2. 选择一些小的值并计算以下量：
3. 这可以通过以下方式很好地近似在处的梯度矢量（或导数）：
4. 从表面法线减去此梯度，然后将结果重新归一化为单位长度：

图5-11显示了将这种方法准确地应用于球体上的三个不同过程凹凸贴图的结果：简单的块状图案，浮雕的大理石图案和分形的皱纹图案。

假设模型是单位半径球，则实现这些凹凸图案的表达式为：

.03 \* noise(x, y, z, 8); //LUMPY

.01 \* stripes(x + 2 \* turbulence(x, y, z, 1), 1.6); //MARBLED –

.10 \* turbulence(x, y, z, 1); //CRINKLED

在以下函数之上定义：

// STRIPES TEXTURE (GOOD FOR MAKING MARBLE)

double stripes(double x, double f) {

double t = .5 + .5 \* sin(f \* 2\*PI \* x);

return t \* t - .5;

}

// TURBULENCE TEXTURE

double turbulence(double x, double y, double z, double f) {

double t = -.5;

for ( ; f <= W/12 ; f \*= 2) // W = Image width in pixels

t += abs(noise(x,y,z,f) / f);

return t;

}

5.7 总结

使用体积噪声的过程纹理在要创建看起来自然的材质印象的情况下非常有用，而无需创建明确的纹理图像。此外，基于噪声的纹理的体积性质使您可以简单地在像素着色器的（x，y，z）位置上对其进行评估，而不是试图找出如何将纹理图像参数地映射到形状表面上的方法。通过这种方式，您可以有效地从固体材料中雕刻出纹理，这通常比尝试制定合理的未失真参数映射要简单得多。

基于噪声的纹理还允许您以与分辨率无关的方式进行工作：当您靠近对象时，可以使纹理保持清晰细腻，而不是在靠近对象时使纹理模糊，而可以将其保持清晰细腻。 程序。

与程序纹理一样，您应尽量不要添加超出像素采样率的非常高的频率。这样的超像素频率不会增加视觉质量，并且在纹理动画化时会导致不必要的斑点。

能够在GPU上有效使用基于噪声的纹理的关键在于，能否真正利用GPU的巨大计算能力来实现噪声。在这里，我们概述了这样的实现，并显示了如何在像素着色器中使用它来以过程方式定义凹凸贴图的几个示例。

这类基于体积噪声的程序纹理长期以来一直是故事片的主要内容，其中着色器不需要实时性能，但需要对天然材料的视觉外观具有很高的保真度。 实际上，今天所有的特效电影都大量使用了基于噪点的程序着色器。 仅举两个例子，例如《完美风暴》中的海浪以及《指环王》三部曲中许多表面纹理和大气效果的效果具有说服力，这在很大程度上取决于像素着色器中噪声功能的广泛使用。 用Pixar的RenderMan等语言编写。 借助可在GPU中实时使用的噪声功能的良好实现，我期待从故事片中看到一些令人兴奋的视觉创意，这些故事片已被整合到台式机和控制台上的下一代交互式娱乐中。