# 什么是柏林噪声

柏林噪声是学者 Ken Perlin 提出的一种很强大的随机生成算法，以二维柏林噪声为例，你输入两个确定的坐标值：x 和 y，柏林噪声函数就会给你一个位于 0 到 1 的随机值，把这个值化为一个灰度值，你在一个二维平面就会得到一个图像：

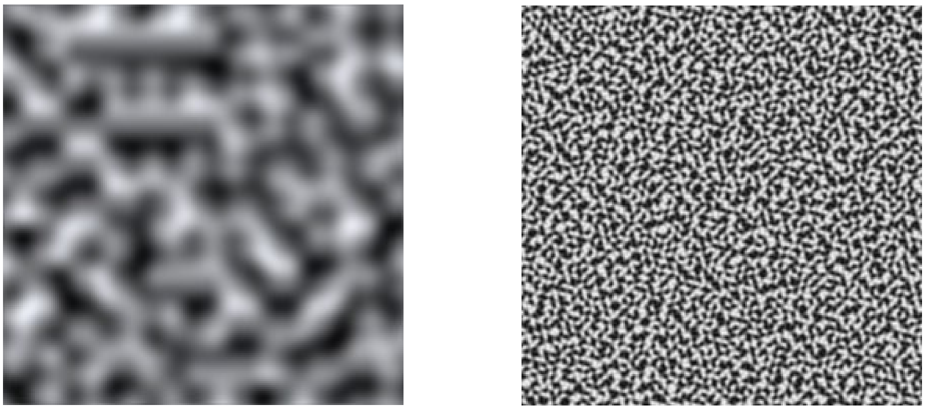
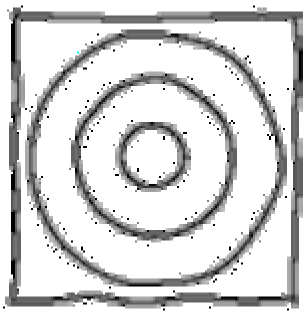
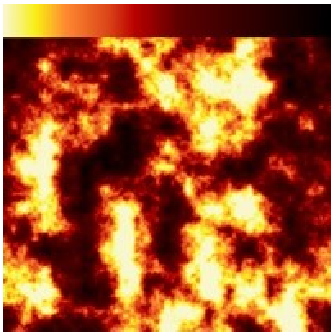
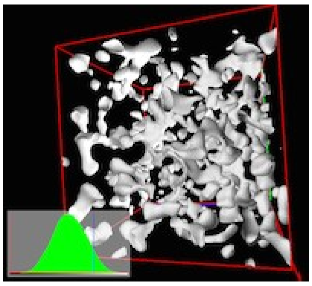


Figure 1 **左边是柏林噪声的效果，右边是普通噪声的效果**

上图是柏林噪声和普通噪声的灰度图像的对比图，我们可以看到，柏林噪声的图像更接近现实情况，看上去更加自然。当然，柏林噪声不止二维，实际上，有一维、二维、三维甚至四维，每一个维度的柏林噪声都有许多奇妙的应用。

| 噪声维度 | 图形学表示 | 应用 |
|---|---|---|
| 1 |  |  *上图的每一条线都是用一维柏林噪声画出来的，通过在直线上设置随机起伏来达到这种手绘的效果，这种起伏就是柏林噪声做出来的。* |

| | | |
|---|---|---|
| 2 |  | <br>*把平面上的由柏林噪声产生的随机值转化为火焰的颜色深度，就可以产生这种逼真的火焰效果。* |
| 3 |  | <br>*三维的例子就不用多说了，著名的游戏 Minecraft 的地形就是利用三维柏林噪声生成的。* |

那么，柏林噪声究竟是如何给出任意一个点的随机值的？为什么柏林噪声比随机噪声更接近真实情况，看起来更顺眼？

# 从 value 噪声开始

在讲柏林噪声开始，我们先讲一讲比较简单的 value 噪声，这种噪声比柏林噪声稍微简单一点。我们这里有一个网格：

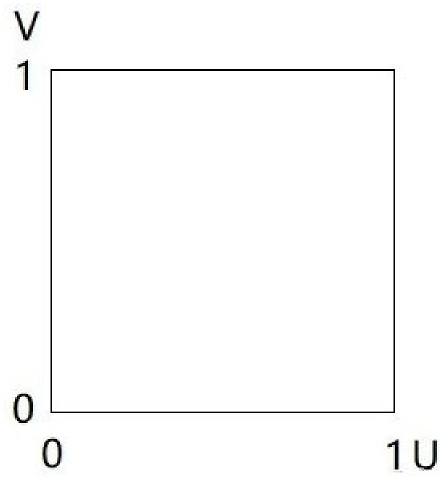



Figure 2 **基本单位：一个网格**

显然它是二维的，因为它有两个坐标轴，V 轴和 U 轴，我们的随机值就需要在这些网格中产生。这些网格只是起到协助作用，不会在后面的图像中显示，至于什么样的作用，你可以看下去。

只有一个网格可能有些单调，我们把网格画得多一点。

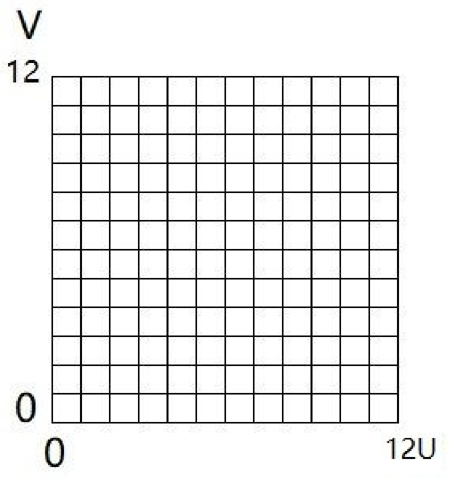



Figure 3 **更多的网格**

你可能会问，这些网格有什么用呢？别急，我们先在这些网格的交叉点上设定一些随机值，至于如何生成随机值，我还没学习，估计是一种符合一定分布函数的随机值吧，还得说一句的是，这种随机值其实是伪随机，因为当位置确定时，它们的值也就确定了。

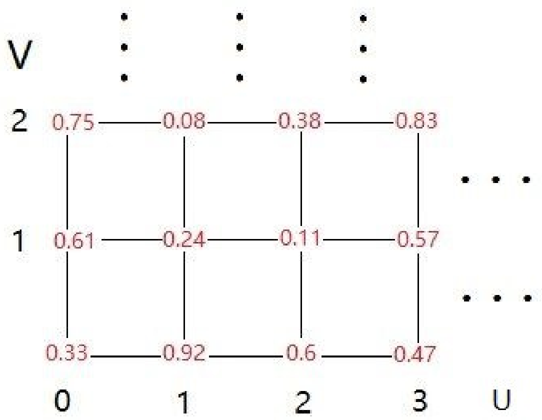



Figure 4 加上交点值的网格

接下来我们取一个点来举例子，这个点可以是

$$U = 2.3 \ V = 1.6$$

我们可以知道，它在[2,2] [3,1]这个网格中。但是我们现在不知道 P 点的数值，只知道 P 点的位置，我们只知道的是 P 所在网格的四个顶点的位置和数值，现在要求 P 的数值，想到什么了？嗯哼？数值分析里的插值法！

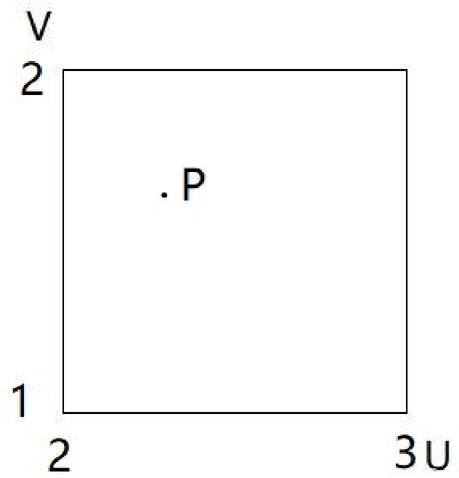



Figure 5 网格中的一个点，我们要求出它的值

那么为什么要插值呢？什么是插值呢？

i. 许多实际问题都可用函数来表示某种内在规律的数量关系；

ii. 但函数表达式无法给出，只有通过实验或观测得到的数据表；

iii. 如何根据这些数据推测或估计其它点的函数值？

对于上面的问题，答案是插值，插值是在找不到确定关系函数的时候的一种估算方法，那什么是插值呢？

定义 2.1 已知函数 y = f(x) 在区间 [a, b] 上有定义，且已经测得其在点

$$a \leq x0 < x1 < \cdots < xn \leq b \tag{1}$$

处的值为

$$y0 = f(x0),\cdots,yn = f(xn).$$

如果存在一个简单易算的函数 p(x), 使得

$$p(xi) = yi, i = 0,1,\ldots,n, \tag{2}$$

则称 p(x) 为 f(x) 的插值函数. 区间 [a, b] 称为插值区间, xi (i = 0, 1, . . . , n) 称为插值节点, 条件 2 称为插值条件。

所以我们现在的任务就是选取一个合适的插值函数，我们先从简单的开始，我们选线性函数来作为插值函数，我们要从二维开始，那么对于直线上的任意一点，它的值都可以表示为：

$$Value(C) = Value(L) * \frac{X(C) - X(L)}{X(R) - X(L)} + Value(R) * \frac{X(R) - X(C)}{X(R) - X(L)} \tag{3}$$

通俗地来讲，就是 L，R 之间的 C 点的值，是 L 和 R 的值加权相加的结果，这个权就是 C 点距离 L 点或者 R 点的距离。当然，你可以设定离某一点越近，该点的权值就越大，这比较符合逻辑，只需要调换一下权值即可：

$$Value(C) = Value(L) * \frac{X(R) - X(C)}{X(R) - X(L)} + Value(R) * \frac{X(C) - X(L)}{X(R) - X(L)} \tag{4}$$

在这个公式的右侧，只有 X(C)的值是不确定的，而且最高次数为 1，说明这是个线性函数。我们以这个函数作为插值函数，借助（1，2）和（3，1）点的值来补全结果 P 点的垂直线与它的交点的值，以此类推，估算出与（2，2），（3，2）的值，只需要代值就可以了，如图所示：

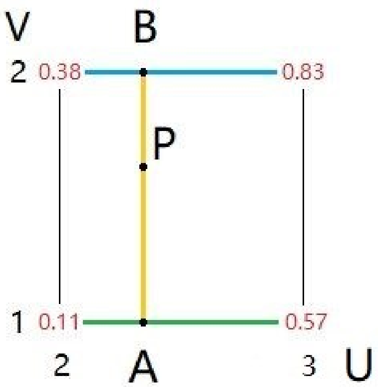



Figure 6 沿 y 轴的线性插值

我们可以把值代入到公式(4)中，就可以计算出 A 点和 B 点的插值，这样我们又得到了一条直线，我们又可以开心地作插值了。

$$Value(A) = 0.11 * 0.7 + 0.57 * 0.3 = 0.248$$

$$Value(B) = 0.38 * 0.7 + 0.83 * 0.3 = 0.551$$

我们把 A 点和 B 点作为端点，对 P 点进行插值：

$$Value(P) = 0.551 * 0.6 + 0.248 * 0.4 = 0.4082$$

这样我们就能得出 P 点的插值，也就是 P 的估计值，注意，这个值当然不是位置，你可以把它看作是高度。我们可以把这个值转化为灰阶，也就是颜色发黑的程度，这样你就可以感受到它的大小了。

现在 P 点的值是(2.3, 1.6)，但是实际上，我们可以得出任意点的插值，只要它周围的网格的随机值存在，这样我们其实就能得到一定分辨率的灰阶图像了，像这样：

Figure 7 **线性插值的灰度图像**

但是看起来还是很不自然，你看，不同网格之间的衔接很明显，你甚至可以尝试摘掉眼镜（如果你没有眼镜那就站远点），你甚至可以想象出原来的网格。这是为什么呢？问题出现在我们的插值函数上，我们为了图简单，选择了线性的插值函数，这种函数有很大的确定，我们来分析分析。

我们回到一维的插值的过程中来，下面是一条线段，我们把公式(4)作为插值函数，成功地估计出了线段中任意点地值：

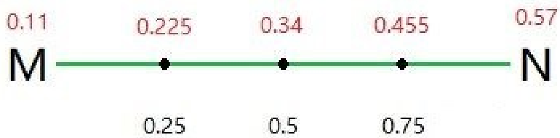



Figure 8 **一条线段的插值**

现在我们多加一条线段：

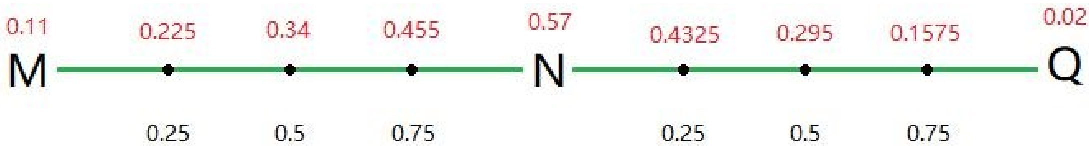



Figure 9 **两条线段的插值**

这样我们得到了更多的插值，还得到了一个转折点 N 点，接下来我们把这些值表示在另外

一个坐标轴中：

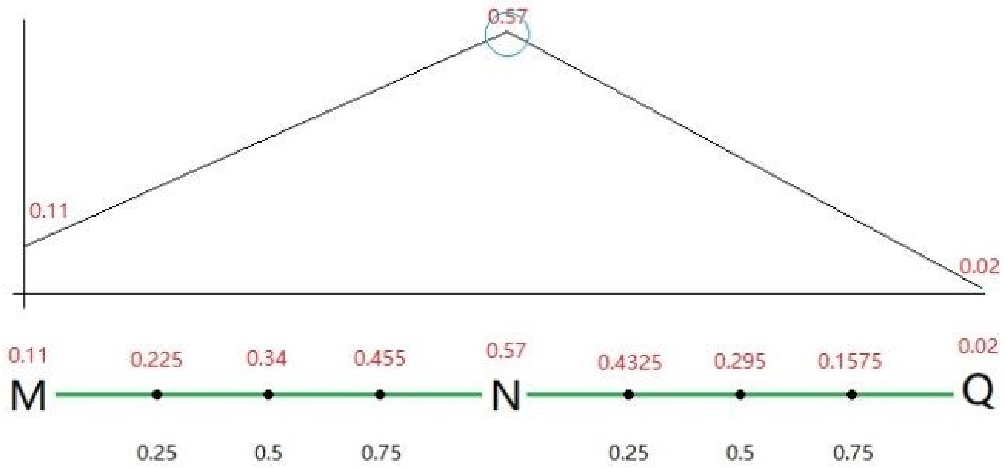



Figure 10 **线性插值函数在一维上的变化**

芜湖，发现 N 点对应的图形了吗，N 点作为转折点过于生硬了，你不看坐标轴，一看就能看出这个点是转折点，另外，其它的线段变化得太过生硬，看起来太假了。

那我们如何改进呢？答案当然是寻找更好的插值函数啊！接下来就轮到柏林噪声登场了

# 柏林噪声的改进

柏林噪声之父 ken Perlin 给出了一个比较好的插值函数：

$$y = 6x^5 - 15x^4 + 10x^3 \tag{5}$$

我们只需要对这个函数在 Y 轴进行一些缩放和平移，让它刚好能嵌入线段，也就是左右端点值和函数取 0 和取 1 的值一样而形状不发生改变，我们就能得到比较平滑的过渡，完美地解决上面的两个问题。

下面这个是函数(5)在 0 到 1 区间的函数图形：

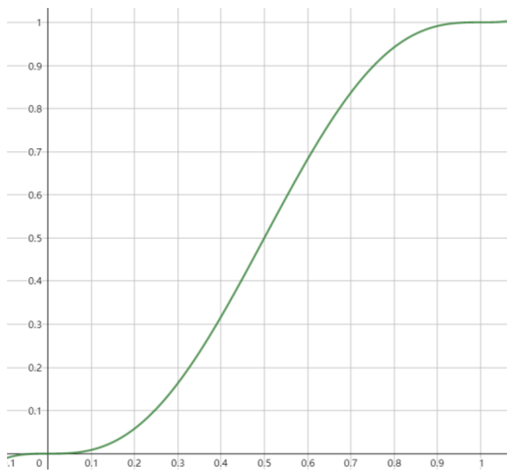


Figure 11 **我们的目标：更加平滑的插值函数**

我们再这个插值函数应用到到一维的坐标轴中，我们可以画上起伏图形看看效果，我们可以看到，现在的起伏就自然多了。

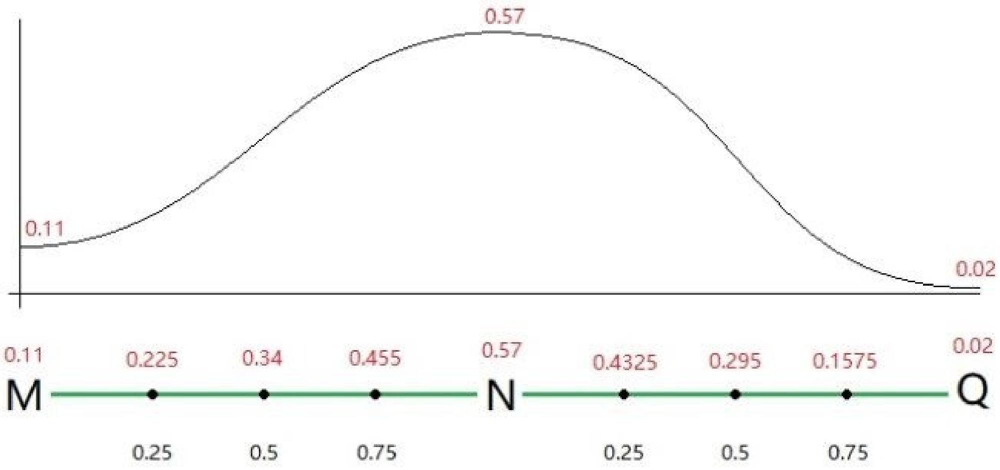



Figure 12 Perlin 的插值函数在 一维上的效果

让我们生成灰阶图片看看效果吧，毕竟图线还不是很直观：

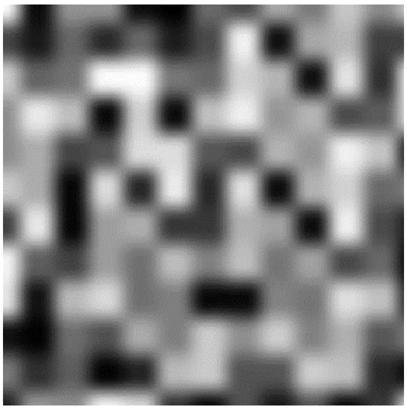


Figure 13 新的插值函数

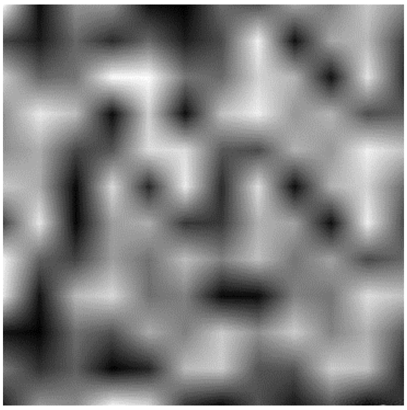


Figure 14 旧的插值函数

嗯，看起来好多了，左边是新插值函数，右边是
线性的插值函数。那，为什么是函数(4)呢？换句话说，函数(4)是如何作为插值函数被确定的呢？我们可以观察这个函数，我们发现，它有一下特征

i. 当 x=0 时，y=0；当 x=1 时，y=1

ii. 当 x=0 和 1 时，图线是水平的，也就是导数为 0

iii. 在中间，图线的上升速度是先变快后变慢，所以导数的值是先小后大再小

第一条的特性是为了这个函数能更好地被收缩移动，毕竟网格地单位长度一般有虽然不是 1，需要的估计值可能也不是 0 到 1，但是我给的函数是 0 到 1 的定义域和 0 到 1 的值域，你可以随便调整，很方便。

第二条的意义是让线段/网格之间的过渡更加平缓，不会有明显的界限，而第三条的作用是

让线段之间变化得更加自然。

所以这个函数会如此的自然，那这个函数是如何确定的呢？我们讨论它本身不太方便，我们可以讨论它的导数，只要我们有了它的导数，我们就可以通过不定积分积出原函数，也就能得到这个插值函数了。

根据原函数的性质，我们可以得出导数函数的性质：

i. 对于导数的积分函数（右移 0.5 后的），当 x=1 时，y=1 ，当 x=0 时，y=0
ii. 当 x=正负 0.5 时，值为 0
iii. y 是先小后大再小

我们先从简单的做起，我们先得到一个对称的图形，也就是说，以 x=0.5 为界限，y 的值虽然不是对称的，但是 y 的值的变化情况是对称的，也就是说，原函数的导数是关于 x=0.5 对称的，为了方便起见，我们把图形向左移动 0.5，这样原函数的导数就关于 x=0 对称了，也就是关于 Y 轴对称，也就是说，导数函数是一个偶函数，我们知道，一个偶函数可以通过多个偶函数相加得到，我们不需要太高的系数，4 次幂就最够了，所以我们得到一个带未知参数的导数函数：

$$y = ax^4 + bx^2 + c \tag{6}$$

接下来我们就可以利用它的特性来求出未知的值，有的同学说，我那最高次幂是 2 次的行不行？也就是：

$$y = ax^2 + b \tag{7}$$

答案是不可以的，因为

我们可以代入 x=0.5 或 x=-0.5，得出：

$$a = -4b$$

然后我们把导数右移 0.5，得到：

$$y = a\left(x - \frac{1}{2}\right)^2 + b$$

化简，代入 a 与 b 的关系之后：

$$y = -4bx^2 + 4bx$$

对这个函数求积分，得到最原始的函数：

$$y = -\frac{4}{3}bx^3 + 2bx^2 + c$$

代入(0,0)可得：

$$c = 0$$

$$y = -\frac{4}{3}bx^3 + 2bx^2$$

代入(1,1)可得:

$$b = \frac{3}{2}$$

$$y = -2x^3 + 3x^2 \tag{8}$$

它的图像是这样的:

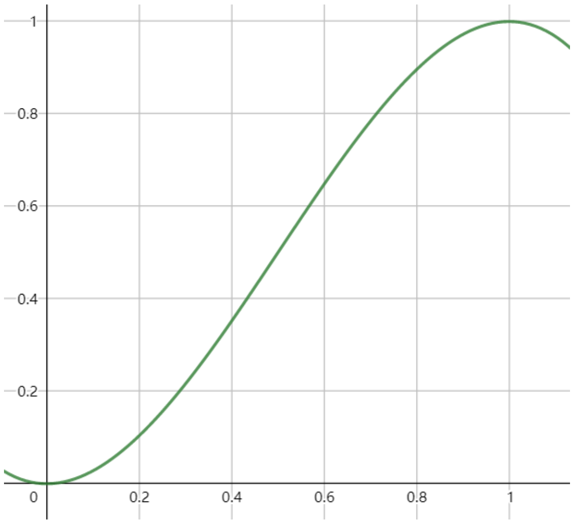



Figure 15 **三阶插值函数**

这个函数完美地符合我们的要求，但是，实际的图片告诉我们，由该插值函数做出来的灰度图任有缺陷:

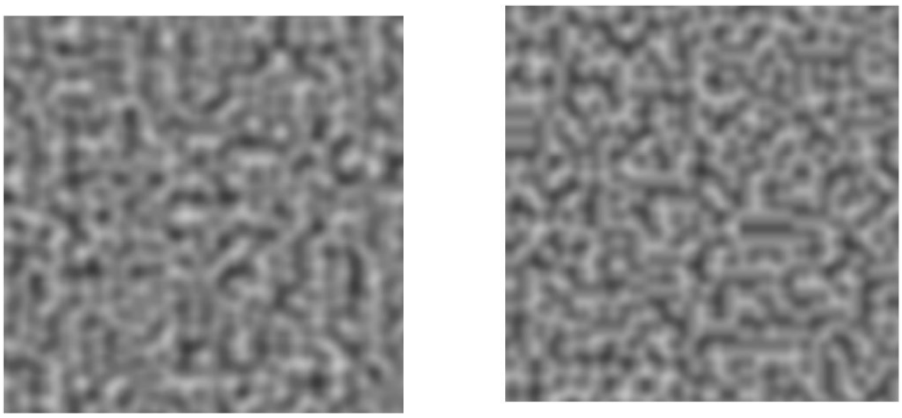


Figure 16 **左边是三阶插值函数生成的灰度图像，右边是五阶插值函数的效果**

新的函数的插值函数我们将稍后推导，造成这个结果的原因是函数(8)的导数具有线性分量（也就是 x 的一次项）。Ken Perlin 自己是这么说的:

*The above deficiencies are addressed as follows. 3t2 -2t3 is replaced by 6t5 -15t4 +10t3 , which has zero first and second derivatives at both t=0 and t=1. The absence of artifacts can be seen in Figure 1b.*

[Improving Noise-Ken Perlin]

所以我们要求一下另外一个更高阶的函数作为我们的插值函数。我们先从导数开始:

$$y = ax^4 + bx^2 + c$$

还是一样的条件，我们把（0.5，0）代入这个函数:

$$\frac{1}{16}a + \frac{1}{4}b + c = 0$$

也就是:

$$a + 4b = -16c \tag{9}$$

接下来我们把这个函数向右移动 0.5:

$$y = a\left(x - \frac{1}{2}\right)^4 + b\left(x - \frac{1}{2}\right)^2 + c$$

化简，代入(9)式子:

$$y = ax^4 - 2ax^3 + \left(\frac{3}{2}a + b\right)x^2 - \left(\frac{1}{2}a + b\right)x + \frac{1}{16}a + \frac{1}{4}b + c$$

$$y = ax^4 - 2ax^3 + \left(\frac{5}{4}a - 4c\right)x^2 - \left(\frac{1}{4}a - 4c\right)x + 0$$

$$y = ax^4 - 2ax^3 + \left(\frac{5}{4}a - 4c\right)x^2 - \left(\frac{1}{4}a - 4c\right)x \tag{10}$$

接下来通过不定积分求公式(10)的积分:

$$y = \frac{1}{5}ax^5 - \frac{1}{2}ax^4 + \left(\frac{5}{12}a - \frac{4}{3}c\right)x^3 - \left(\frac{1}{8}a - 2c\right)x^2 + c \tag{11}$$

接下来我们把(0,0)代入公式 11，得到:

$$y = \frac{1}{5}ax^5 - \frac{1}{2}ax^4 + \left(\frac{5}{12}a - \frac{4}{3}c\right)x^3 - \left(\frac{1}{8}a - 2c\right)x^2 \tag{12}$$

我们再把(1,1)代入，得到:

$$\frac{1}{5}a - \frac{1}{2}a + \left(\frac{5}{12}a - \frac{4}{3}c\right) - \left(\frac{1}{8}a - 2c\right) = 1$$

化简得:

$$-a + 80c = 120 \tag{13}$$

为了在减少损失的情况下，减少计算量我们令公式(12)中二次项为 0：

$$\frac{1}{8}a - 2c = 0$$

$$a - 16c = 0 \tag{14}$$

我们把公式 13 和公式 14 联立：

$$c = \frac{15}{8}$$

$$a = 30$$

我们再代回公式 12：

$$y = 6x^5 - 15x^4 + 10x^3 \tag{15}$$

我们就得到了新的插值函数。

把这个函数当作插值函数，做出来的灰度图片更加自然，这也是我们目前广泛使用的插值函数。

接下来，我们需要改进随机值，我们之前说过，网格顶点的值是随机产生的，我们需要引入垂直向量来改进它，我们可以把随机变量和垂直变量结合起来。

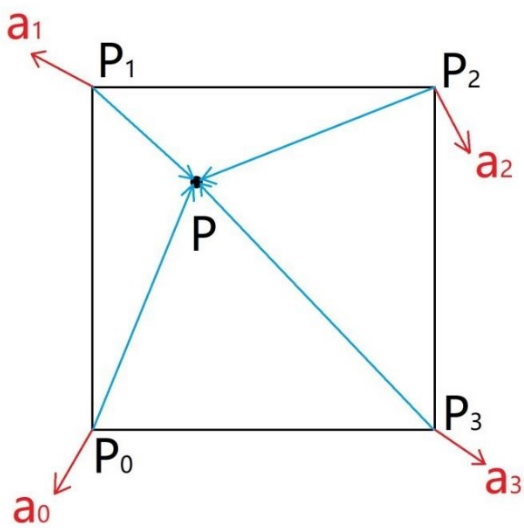



Figure 17 **引入向量来设置顶点数值**

在上图的网格中，红色的向量是我们随机生成的向量，称作梯度向量，蓝色的向量是四个顶点指向所求点的向量，称为垂直向量，我们通过点成两者来求出一个新的值，来作为四个顶点的值：

$$Value(P_n) = a_n \cdot \overrightarrow{P_nP} \qquad (n = 1,2,3,4)$$

改进插值和顶点值之后，我们就能像 Value 噪声一样做出 Perlin 噪声了。

# 疑问

1.在公式 13 到公式 14 的过程中，为什么可以这样简化？

2.为什么要以垂直向量点乘随机向量的方法代替原来的单纯随机值？有什么具体的好处？

3.随机数如何生成？有什么要求？

# 参考


如何生成一张 Value Noise 算法图片（包括 Perlin Noise） - N.ci 的文章 - 知乎

Improving Noise-Ken Perlin

Understanding Perlin Noise-flafla2

柏林噪声-fade 函数究竟是什么

数值分析讲义* 潘建瑜