本部分将实现FXAA抗锯齿算法。

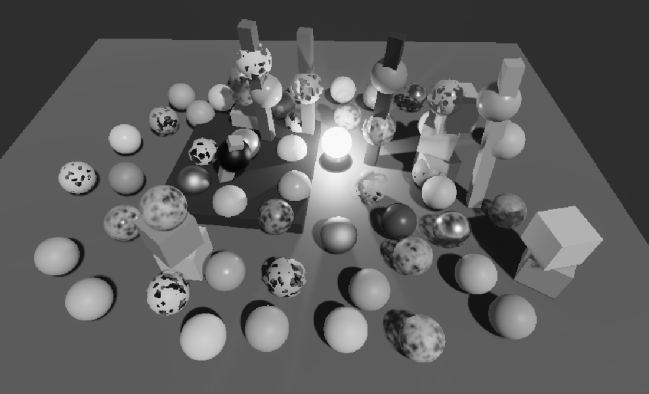
“帧缓冲区的有限分辨率会给最终图像带来视觉锯齿伪影。这些通常称为jaggies或stairsteps，我们可以在与像素网格不对齐的线条上观察到。除此之外，对于小于像素点的特征，当它们处于运动状态时，会随时间产生闪烁伪影。”

16中改变分辨率的做法实际上是SSAA抗锯齿的方法，但是该方法开销较大。第一个后处理的AA方法叫MLAA，它分析图像以检测视觉特征的边缘并有选择地模糊它们，FXAA受此启发实现。与MLAA相比，FXAA以图像质量为代价换取了处理速度。

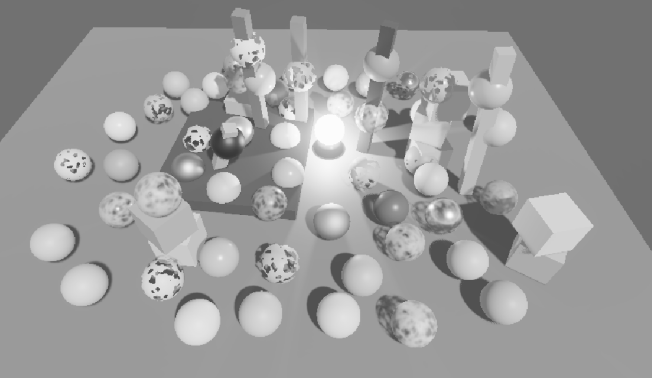
（MSAA只会应用在三角形的边缘，对于clip的效果支持不好）

FXAA的工作原理是选择性地降低图像对比度，依据是像素的伽马矫正亮度（luma）。FXAA处理的阶段是颜色分级和色调映射之后，分辨率缩放之前。

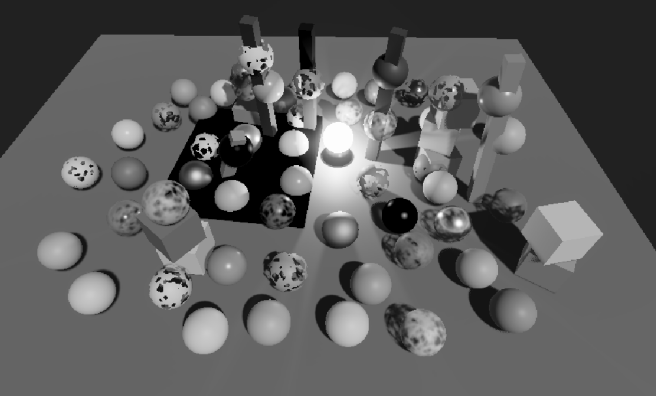
输出亮度：



因为我们对深色的变化比浅色的更敏感，所以我们对亮度取平方根，以实现灰度系数调整：



不过，计算亮度开销较大。因为我们在视觉上对绿色更敏感，所以可以用绿色通道作为替代方法：

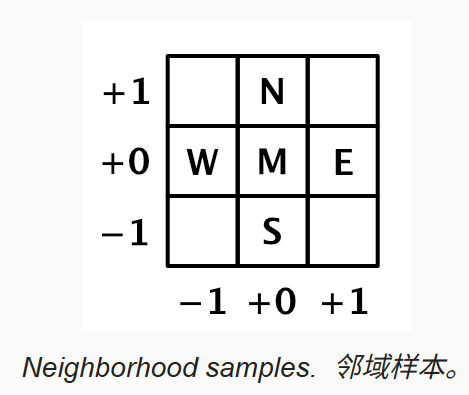


不过，我们也可以在颜色分级时，将luma存储到alpha通道里，调整ApplyColorGradingPass。

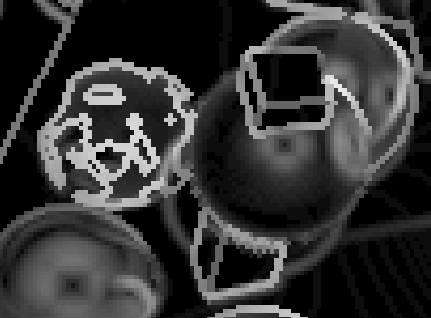
当该通道不可用时，我们再去选择用g通道的方法。对每个相机进行设置：



“FXAA 的工作原理是混合具有高对比度的相邻像素。所以这不是图像的简单均匀模糊。首先，必须围绕源像素计算局部对比度（从最低到最高亮度的范围）。其次，如果有足够的对比度，则必须根据对比度选择混合因子。第三，必须研究局部对比度梯度以确定混合方向。最后，在原始像素与其相应的相邻像素之间执行混合。”



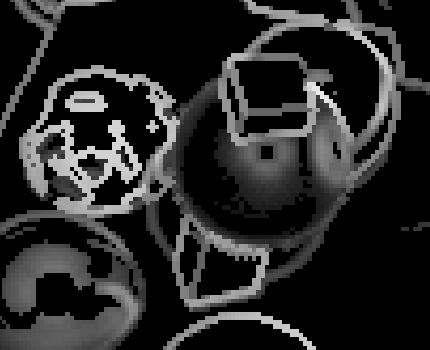
采样领域亮度，获取最高值和最低值，输出二者之差，得到：



我们只对对比度足够大的区域进行混合，因此增加一个固定阈值的配置参数。只有超过阈值的才被处理：



再增加一个相对阈值，当对比度超过相对阈值乘最高亮度时，才进行处理：



我们取二者最大值作为用于比较的阈值。

“提高边缘视觉质量的唯一正确方法是提高图像的分辨率。但是，FXAA 只有原始图像数据可供使用。因此需要猜测缺少的子像素数据。它通过将中间像素与其相邻像素之一混合来实现此目的。在最极端的情况下，这将是两个像素的简单平均值，但具体的混合因子是filter的结果，这取决于像素的对比度及其相邻像素的平均值。我们将分步骤进行可视化。”

首先，取领域的平均值，作为低通滤波：



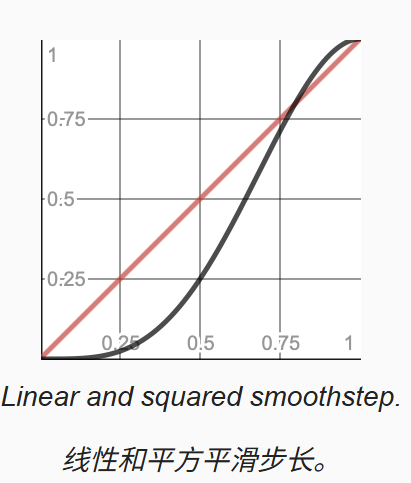
将平均值与原像素的值作差，取绝对值为高通滤波：



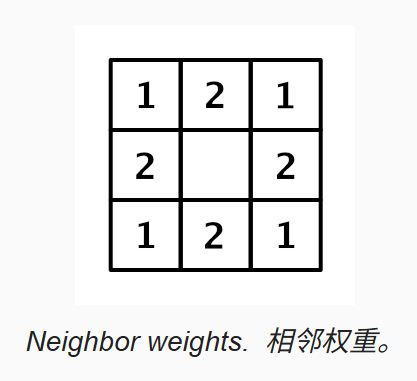
然后除以亮度范围进行归一化：



结果过强了，FXAA应用平方平滑步长调整结果：

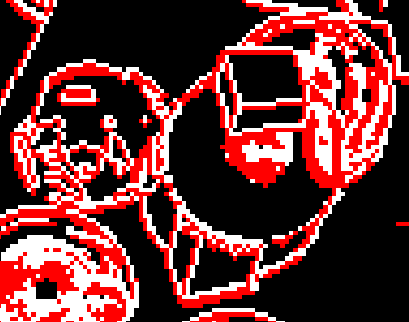
改进原来的平均值操作，考虑对角线的样本，权重如下：



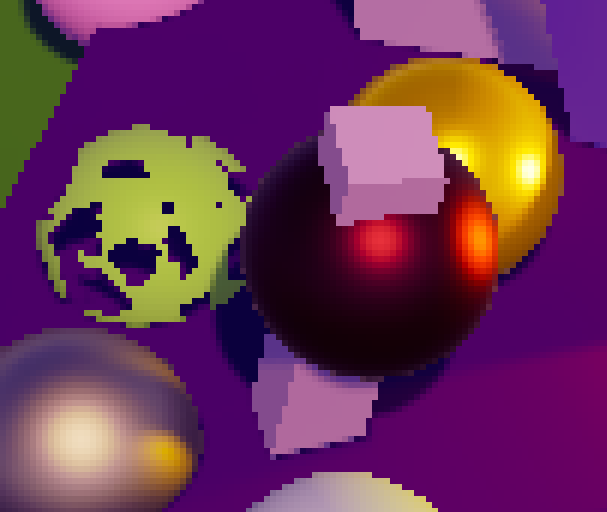
下一步是决定要混合的两个像素，从东南西北里选择。我们比较领域的水平和垂直对比度来寻找水平边缘或垂直边缘。水平边缘为红色：



然后，我们选择差值更大的方向作为混合方向，可视化结果，正边为红色：

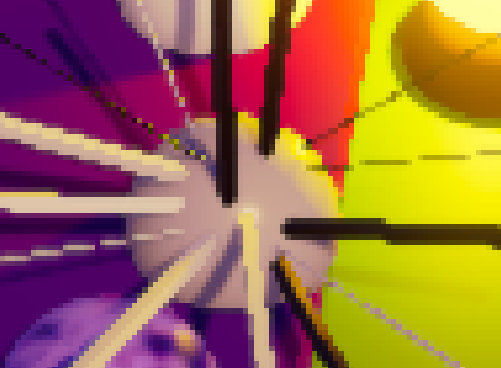


现在，我们知道了混合方向和系数，那么利用该结果改变采样点进行采样即可实现插值：

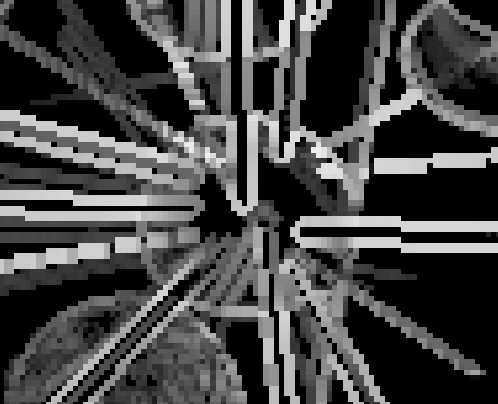
 （原图）

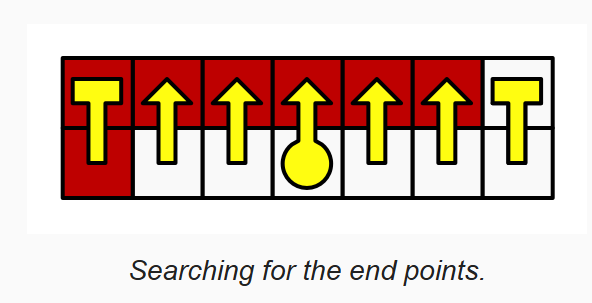
但是，利用对比度混合，可能会把材质的细节也模糊掉，比如之前教程中的Detail。为此，在混合因子的计算时，可以乘上子像素混合强度。

但是，上述对边缘的考虑只在3x3的情况下实现，并且只考虑水平和垂直边缘，实际上的边缘可能很长并带有一定角度：

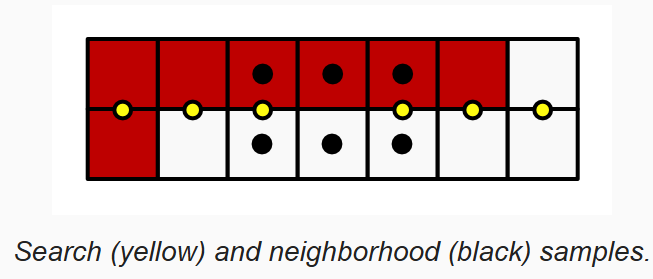


我们将使用新的混合因子计算函数，先把之前计算得到的梯度作为结果返回：



我们需要知道当前点相对于所在边端点的位置，因此我们可以沿着边进行采样：

进一步简化的话，我们可以只采样下图中的黄色点，因为线性插值是自动完成的：

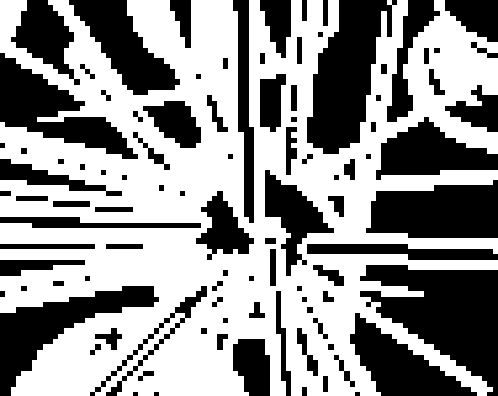


起始位置是当前uv向所在边偏移半个像素长度的位置。

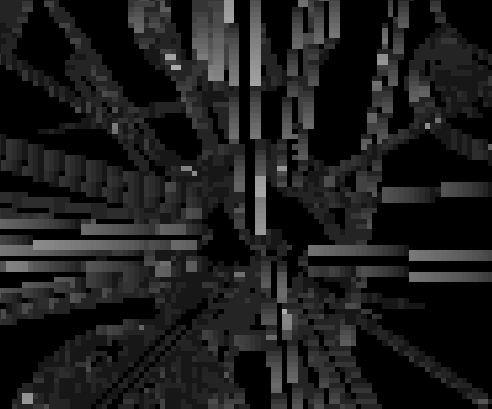
我们先向正方向移动一步，判断是否超过阈值（这里的阈值为0.25乘采样点的梯度大小）：



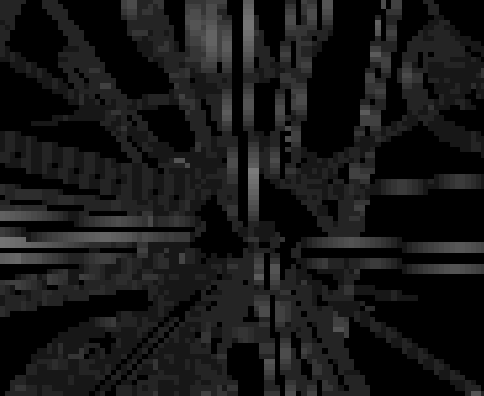
进一步的，我们沿正方向采样，直到找到端点，或超过100步：



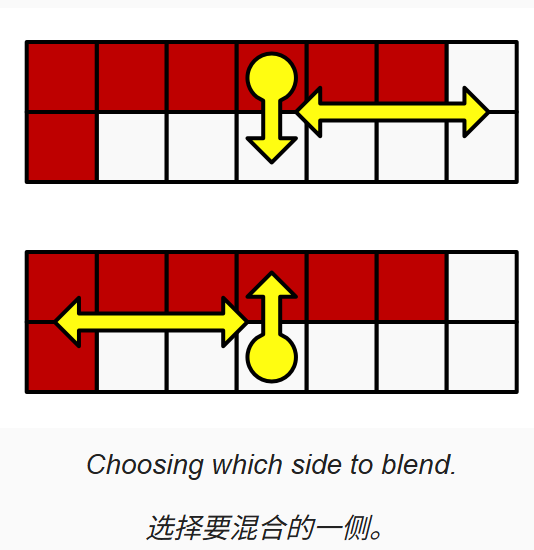
将得到的端点与当前位置作差，得到距离：



然后对反方向也做同样操作，进而可以找到当前点到所在边最近端点的距离：



然后，我们判断终点的梯度方向，以决定需要混合的一侧：



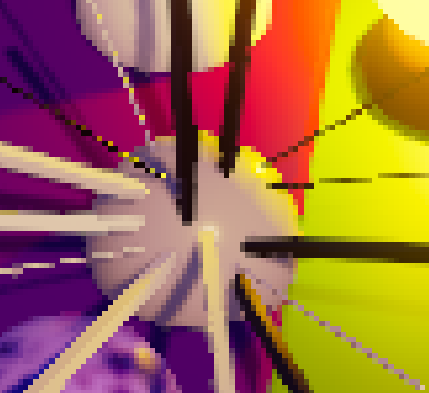
如果最后端点的梯度方向与采样点的相同，表示当前点是远离边缘的（否则所在边的中点肯定落在采样点到端点之间），对于远离边缘的情况，我们直接返回0：



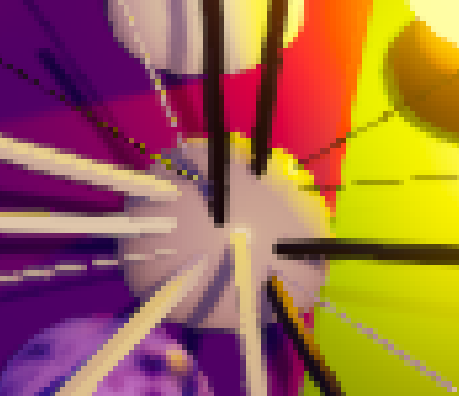
混合因子为0.5减去采样点在所在边的相对位置，当相对位置靠近端点时，我们混合地更多，靠近中点时，混合因子逐渐变为0：



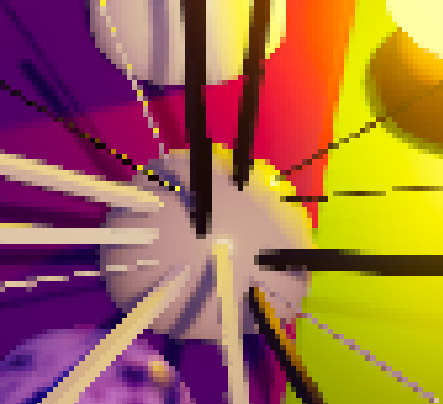
仅使用边缘混合因子：



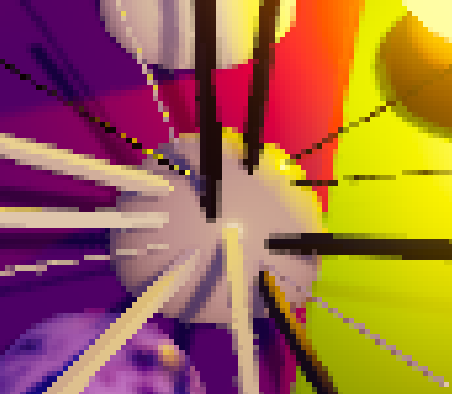
将该混合因子与子像素的混合因子选最大值：



接下来，限制搜索长度为4个像素，如果最后没有找到，那么我们猜测端点为第5个像素：

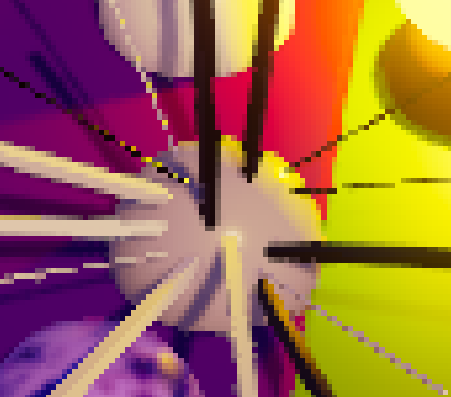


因为这种操作是质量和时间的trade-off，所以对于采样步数、每步步长和超出后的端点位置猜测，进行宏定义。此外，使用较大的步长可以处理更长的边，但容易导致边缘抖动。

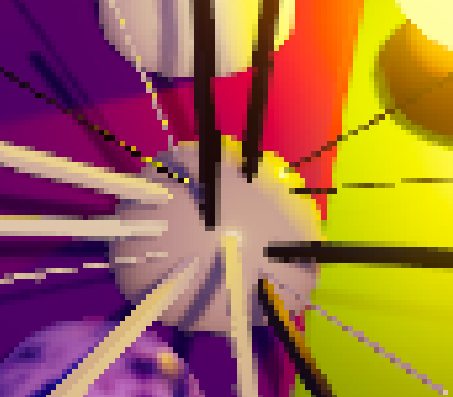


注意，我们让第一次的步长为1.5，可以在采样时对相邻的四个像素点插值。

进一步的，我们可以配置中等质量的结果：



高质量的结果：



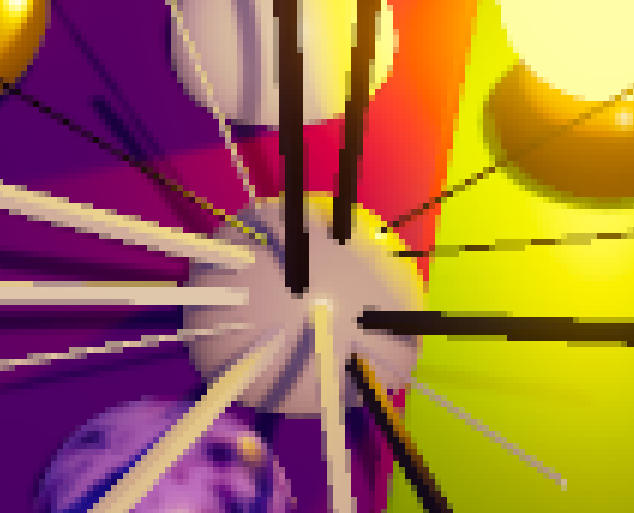
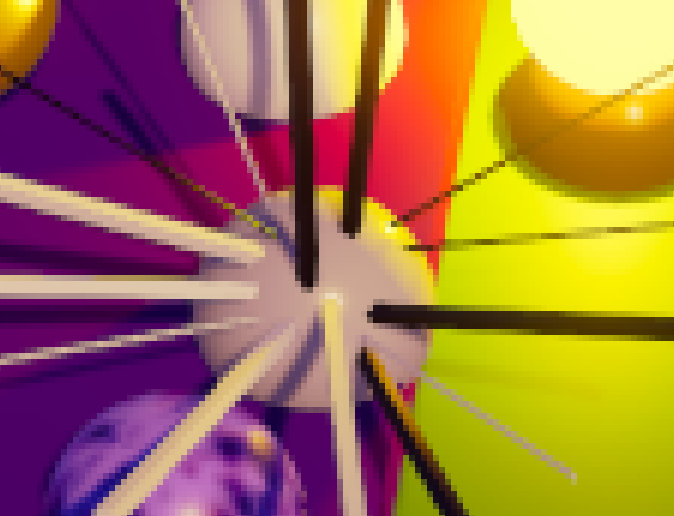
我们通过multi\_compile定义关键字选择质量级别。

知识点：





你可以将SSAA和FXAA结合，缩放比例为2，左为无FXAA：

缩放比例1.333，使用双三次下采样，加上FXAA：



还可以使用FXAA改善低渲染比例的结果，左为无FXAA：

此外，我们可以在循环前添加 UNITY\_UNROLL 来让着色器展开循环。

至此，基础系列的教程完结。