# Notes For Material Nodes in Unreal Engine 4.

—Digital Tutor Material Nodes Reference Library.

—Witten by Reforia.

## Notes For Practical Material Nodes

---

### 1.TextureSample.（贴图式样）

这个节点代表了一个贴图的式样，在材质编辑器的右侧中可以找到，隶属于Texture类。

TextureSample节点允许输出五个通道，分别为整体，红（Red），绿（Green），蓝（Blue），透明度（Alpha），双击Content Browser（内容浏览器）中的贴图，在左上角的View选单中可以下拉选择显示不同通道中的贴图信息。

右键单击节点并且选择Start Previewing Node可以在不连接节点到材质面板的情况下预览该节点。

在TextureSample节点左侧Details面板的Material Expression Texture Sample（贴图式样材质表达式）栏内，有MipValueMode（减值模式）参数，包含了None（无），MipLevel（减值层次）和MipBais（减值偏差）三类。其中，MipLevel和MipBais均代表了降低贴图分辨率的方式。当选择其一时，会对应在TextureSample节点的输入侧出现Level或Bais的通道。

在Content Browser中，贴图的前缀为T_，根据贴图的类型（Texture Type）不同，后缀也不同，常用的有漫反射贴图DiffuseColor（_D），法线贴图（_N）等。根据不同的命名方式，虚幻可以自动识别贴图类型，Sampler Type（贴图类型）参数位于Details面板Material Expression Texture Base（材质基本贴图表达式）一栏中。

通过快捷键T并左键单击材质编辑窗口可以快速调用节点，当在Content Browser（内容浏览器）中选中贴图时，按快捷键T并左键单击材质编辑窗口可以快速将贴图作为TextureSample调用。

在Details面板，Material Expression栏内的Desc参数中可以改变节点注释。

---

### 2.TextureCoordinate.（贴图坐标）

这个节点能够以双通道形式输出uv表达式，快捷键是U。

通过这个节点，我们能够控制材质所使用的uv。

在detail面板中，第一个是Coordinate Index代表了材质所使用的模型通道，通常是0（贴图uv），1通道通常是为光照贴图所创建的uv。

U/V tiling：当数值变大时，uv将变得更密集，当数值为负时，uv将被翻转，当数值介于0～1之间时，uv将变得更分散。

UnMirror U/V：当贴图由于本身存在镜像而不能完好的对齐时，这两个参数可以快速修正由于镜像导致的贴图错误。

---

## 3.Constant3Vector（3维向量）

用于创建三维的常数。快捷键分别对应3。

Detail面板中的Material Expression Constant 3Vector（3维向量常数表达式）中包含了RGBAHSV的值，单击颜色块可以打开颜色编辑器，其中，RGBSV值只介于0到1之间，H的值决定了位于彩色板边缘的颜色，S的值决定了目标颜色与纯白之间的偏差程度，V的值决定了目标颜色与纯黑之间的偏差程度。sRGB模式默认勾选，用于统一颜色显示方式。

---

## 4.Multiply（乘积）

在palette中可以找到，隶属于Math类，快捷键为M，允许输入A，B两个参数并做乘法运算。

AB两个参数可以在Detail面板中用一维常数指定，也可以输入一个TextureSample或任意维度的常数。

---

## 5.Add（叠加）

在Palette中可以找到，隶属于Math类，快捷键为A，允许输入A，B两个参数并做加法运算。

当输入的常数超过1时，输出的图像会做clamp运算（限定区间，将超过1的部分做1处理，因RGB通道的值介于0～1之间。）但输出的值不会因此而被约束为1，这意味着当叠加的常数值为2时，输入BaseColor通道的图像被clamp成1，而Emissive Color通道的值依然是2。

与Multiply运算相同，A，B的值同样接受TextureSample输入。

与Multiply运算不同，当A，B的某个值为0时，Multiply节点的输出值会成为黑色（0值），而Add节点会输出非0的那一个参数的值。

---

## 6.Constant（常数）

可以在palette面板中找到，隶属于Constant类，快捷键为1，储存了一个浮点型常数。

Constant节点中超过1的值在输出图像时会被做Clamp处理，但在代表值而非图像时不会被做Clamp处理。

Constant可以作为Math类运算的输入值。

Constant节点可以被作为一个暴露的参数（Parameter），快捷键S，以用于在材质示例中进行实时变动操作。

创建parameter之后，只需要在内容浏览器中右键材质并选择create material instance即可创建该材质的实例，并在实例化材质面板中实时操作被暴露的参数（Parameter）而不用重新编译着色器。

---

## 7.Linear Interpolate（线性插值）

在Palette面板中，同时隶属于Math类和Utility类（实用类），快捷键L，缩写为Lerp。允许输入三个值，A，B和Alpha值。

Lerp节点会基于Alpha的值输出一个基于A和B之间的数值，例如A为0，B为1，Alpha为0.5时，输出的值为0.5。

输入的A，B，Alpha值均可以接受TextureSample节点。

Lerp节点可以被认为控制了A，B两者被Multiply的程度。

例如，当A通道输入了一张泥土的TextureSample，B通道输入了一张砖块的TextureSample时，只需要在Alpha通道中输入一张随机噪点贴图即可输出一张覆盖了泥土的石砖贴图。

---

## 8.Vertex Color（顶点着色）

位于Palette中的Constant类，无快捷键。

将Vertex Color节点作为Alpha节点输入Lerp节点时，输出的贴图将会被顶点化，此时回到引擎的编辑器窗口，在Modes（模式）面板内选择Paint模式后，可以在下方的参数栏中调整笔刷大小，同时选择顶点颜色和翻转材质颜色，随后在模型上绘制材质时会根据Vertex Color节点进行分部分的绘制。

---

## 9.Panner & Time（UV动画和时间）

在Palette中隶属于Coordinate类，快捷键P，用于创建动态UV。允许输入一个Coordinate参数和一个Time参数，可以输出信息给TextureSample。

在details面板中，SpeedX/Y定义了UV的移动方向，X代表水平方向，Y代表垂直方向。当数值为负时，UV将会反向移动。

输入的Time值是用于定义UV移动的规范，例如，当输入一个Time（世界时间）节点时，UV的移动将会平滑的按照时间的流逝进行移动。

Coordinate通道允许输入TexCoordinate节点，通过调整Coordinate Index的值，Panner输出的UV将会参照给定的Coordinate节点中的信息进行移动。

---

## 10.Rotate（旋转）

在palette中隶属于Coordinates类，无快捷键。允许输入Coordinate和Time参数，可以输出信息给TextureSample。

在Details面板中包含了Center X/Y参数，定义了贴图将围绕什么位置旋转，X为1，Y为0时，贴图以左上角为中心逆时针旋转。当X为1，Y为1时，贴图以右下角为中心逆时针旋转。Speed参数定义了旋转的速度。

同样可以输入Time节点到Time通道中。

当TexCoord节点输入到Coordinate通道中时，通过调整U/V tiling参数的值可以放大或缩小旋转的图像的UV。

---

## 11.One-Minus（1减节点）

在Palette中隶属于Math类，快捷键O，可以用1减去所输入的值，用于将输入的图像信息取反。（黑白颠倒。）

---

## 12.Noise（噪点）

在Palette中隶属于Utility类，无快捷键。用于创建一些随机噪点。允许输入Position通道和FilterField通道，其中FilterField通道影响了输出噪点的模糊程度。

在Details面板中，Scale参数用于改变噪点的大小，位于0～1的数字会增大UV，超过1的数值会减小数值。

Quality参数定义了图像的品质，越大的数值会导致更好的效果，但会降低处理速度。

Function参数控制了节点被计算的方式，Simplex的方式拥有快速的计算速度，Perlin会更快，但是品质会降低。Gradient会拥有很高的品质，但是处理速度会慢，Fast Gradient会降质，但比Gradient的运算要快。

Turbulence决定了是否运算更多层次的噪点细节。

Levels节点定义了多少层次的噪点将被混合运算。

Output Min和Output Max决定了输出的图像的对比度。

Level Scale定义了噪点层次的运算，越低的值会让噪点的离散程度更低。

---

## 13.Desaturation（去饱和）

在Palette中隶属于Color和Utility类。无快捷键，允许输入Fraction通道和一个通用通道，该节点用于将输入的图像去饱和，输出一张黑白的贴图。

Details面板中的RGBA值用于定义对不同通道去饱和的程度。

Fraction节点用于输入一个常数，用于定义整体去饱和程度，当常数为1时，完全去饱和，当常数为0时，不去饱和。

---

## 14.Component Node & Append Vector（组件节点和附加向量）

Component Mask节点在Palette中位于Math类，无快捷键，用于将输入的贴图的通道选择性输出。

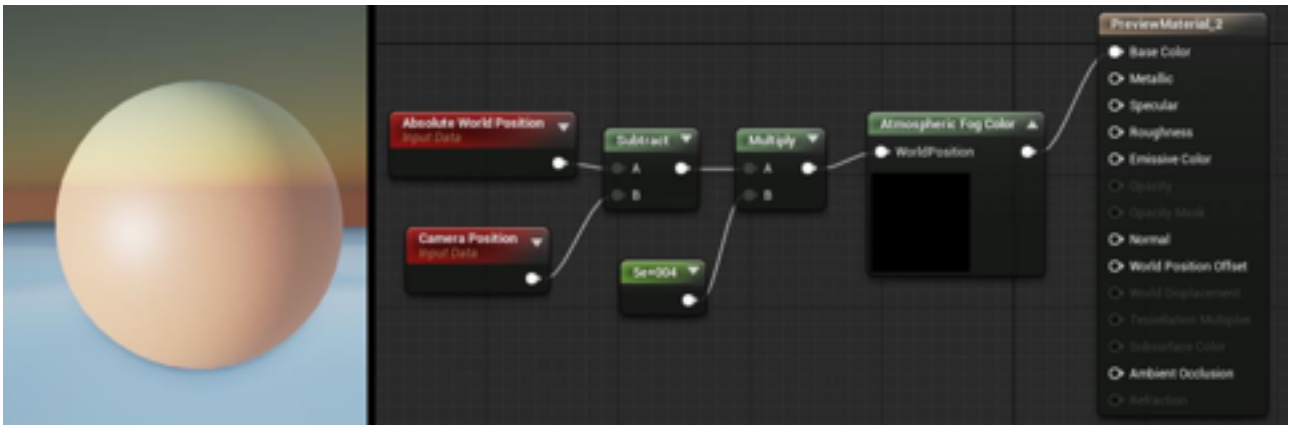
AppendVector节点用于将通道或向量做叠加运算，只用同样维度的向量可以运算，因此当需要将贴图与Mask节点提出的双通道向量相加时，需要先用AppendVector节点加入一个值为0的常数作为Blue通道，并将输出的值与TextureSample做Add运算。

## Atmosphere Expressions（大气表达式）

### 1.Atmosphere Fog Color（大气层雾效颜色）

大气层雾效颜色材质表达式允许你访问当前位于场景中任何一个世界坐标的大气层雾效颜色，如果没有接入任何世界坐标的话，将会使用未被确定的像素的世界坐标，这在你需要将材质从远处的雾气中淡入淡出时是非常有用的。

在以下例子里，基本色被设置为使用了一个大气层雾效节点，并且接受了一个被简单连接的节点网络。让材质总是显示物体背后50000个虚幻单位处的大气层雾效颜色。




## Constant Expressions（常量表达式）

### 1.Distant Cull Fade（限定距离淡入）

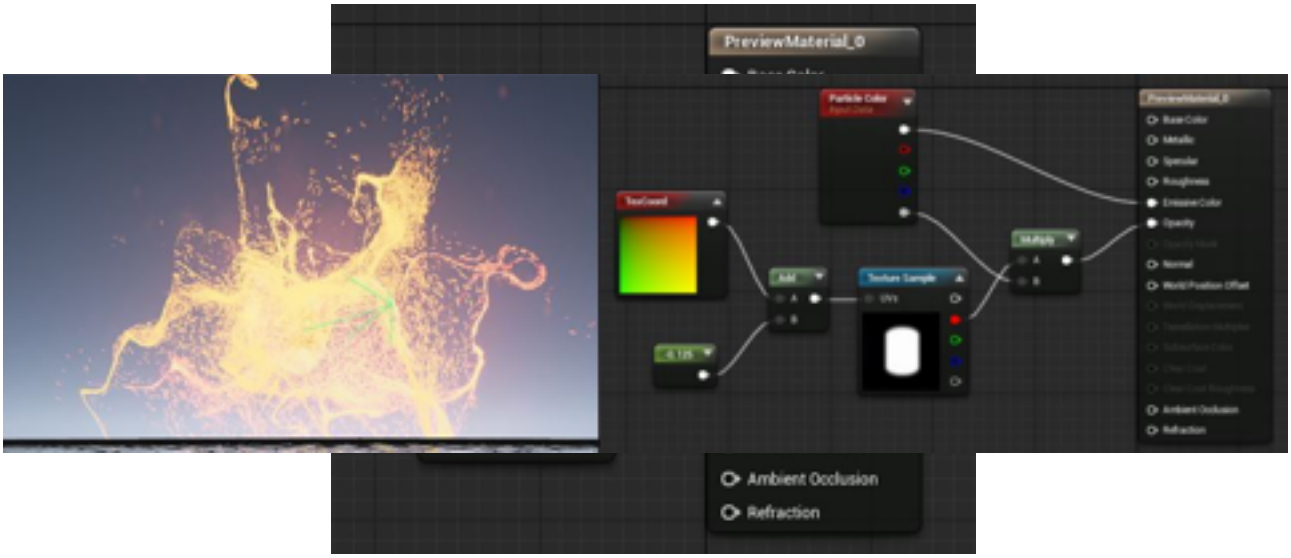
限定距离淡入表达式输出了一个无向的值，使得物体在限定距离内时从黑到白逐渐淡入场景，值得注意的是，这个节点并不能将物体淡出。

这个节点网络会在摄像机进入限定距离时淡入材质而不是突然出现。

### 2.Particle Color（粒子颜色）

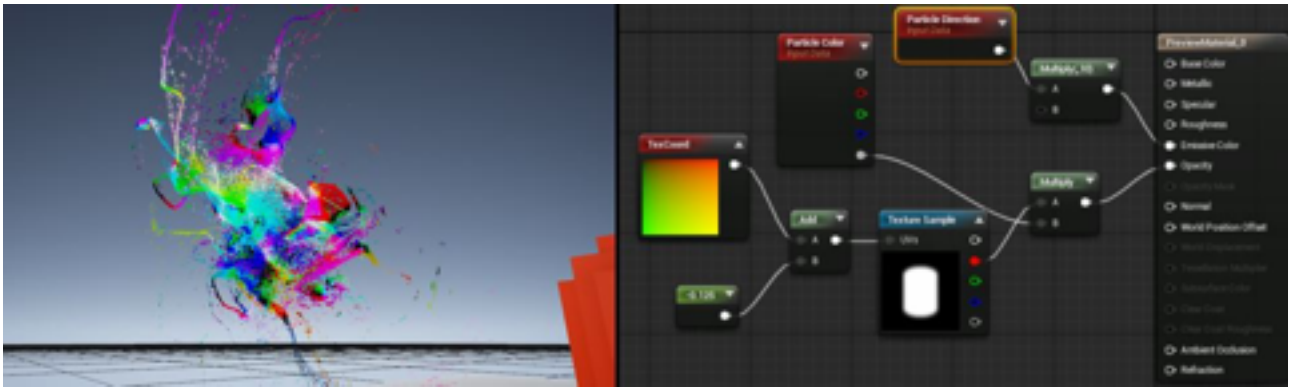
粒子颜色表达式能够基于任何在Cascade中定义的单粒子颜色数据，将其捆绑在当前粒子的颜色上。该节点必须被连接到合适的通道里（自发光通道）。

在这个例子中，你可以看到粒子颜色表达式在哪里为定义好的粒子提供颜色数据。




## 3.Particle Direction（粒子方向）

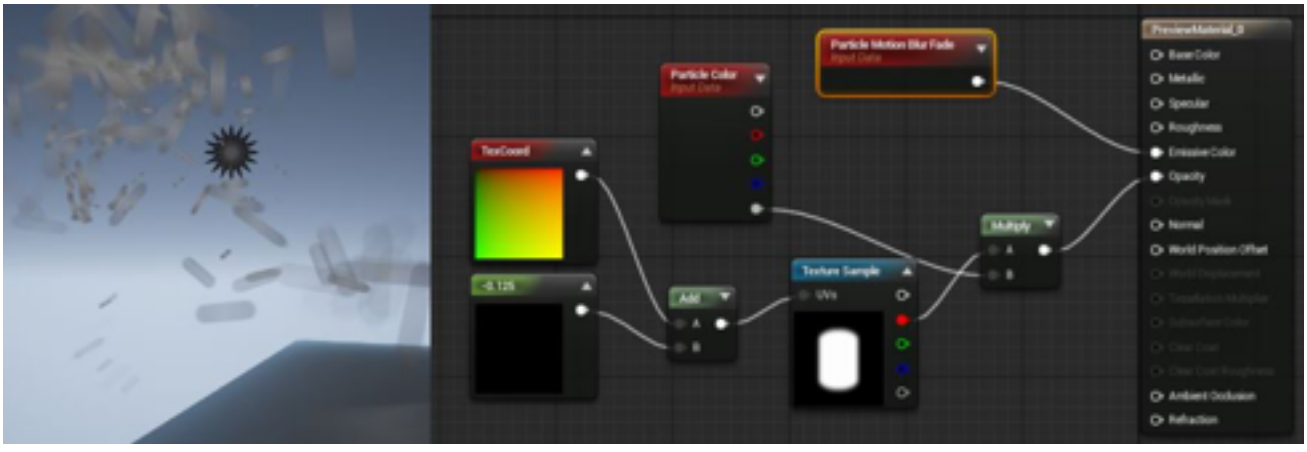
粒子方向表达式输出了一个基于单粒子的3维向量（RGB），代表了给定的粒子正在运动的方向。



在这张图片中你可以看到根据每个粒子的运动轨迹不同，粒子的颜色是如何改变的。

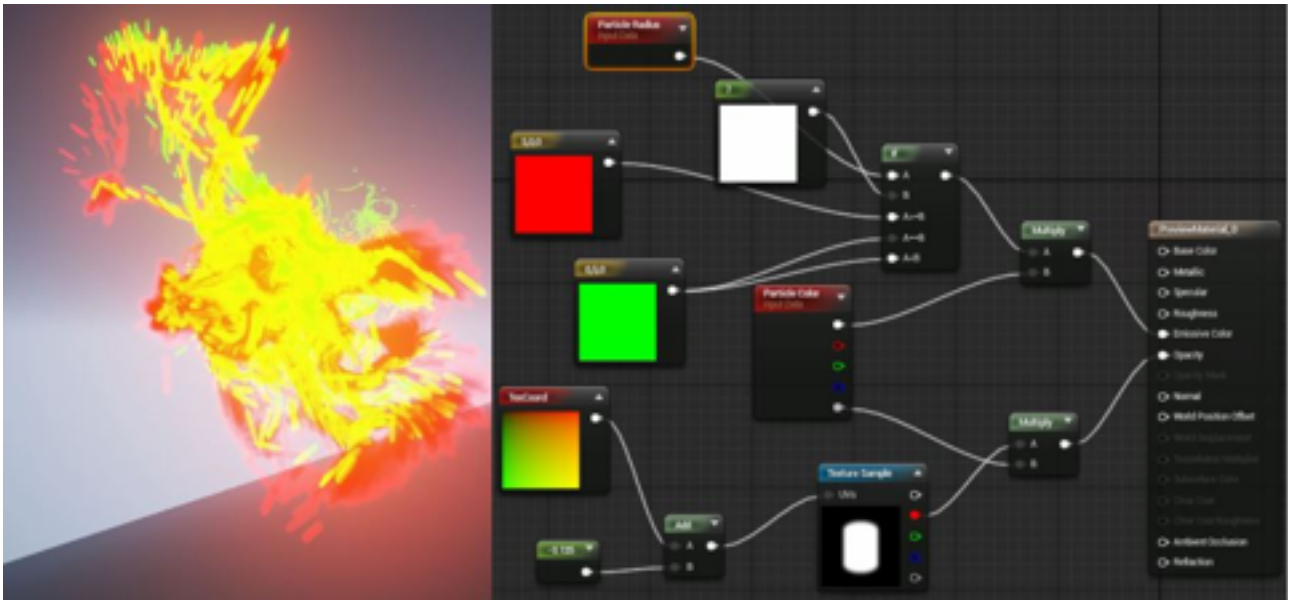
## 4.Particle Motion Blur Fade（粒子动态模糊淡入淡出）

粒子动态模糊淡入表达式输出了一个代表粒子动态模糊淡入程度的值。当值为1时代表无模糊，黑色（值为0）代表完全模糊。



## 5.Particle Radius（粒子半径）

粒子半径表达式为每一个单独的粒子输出以虚幻单位为标准单位的粒子半径。这可以用来创造一些基于粒子半径的颜色变化。



在这张图片中，当粒子半径超过7时，粒子颜色会由绿色变为红色。

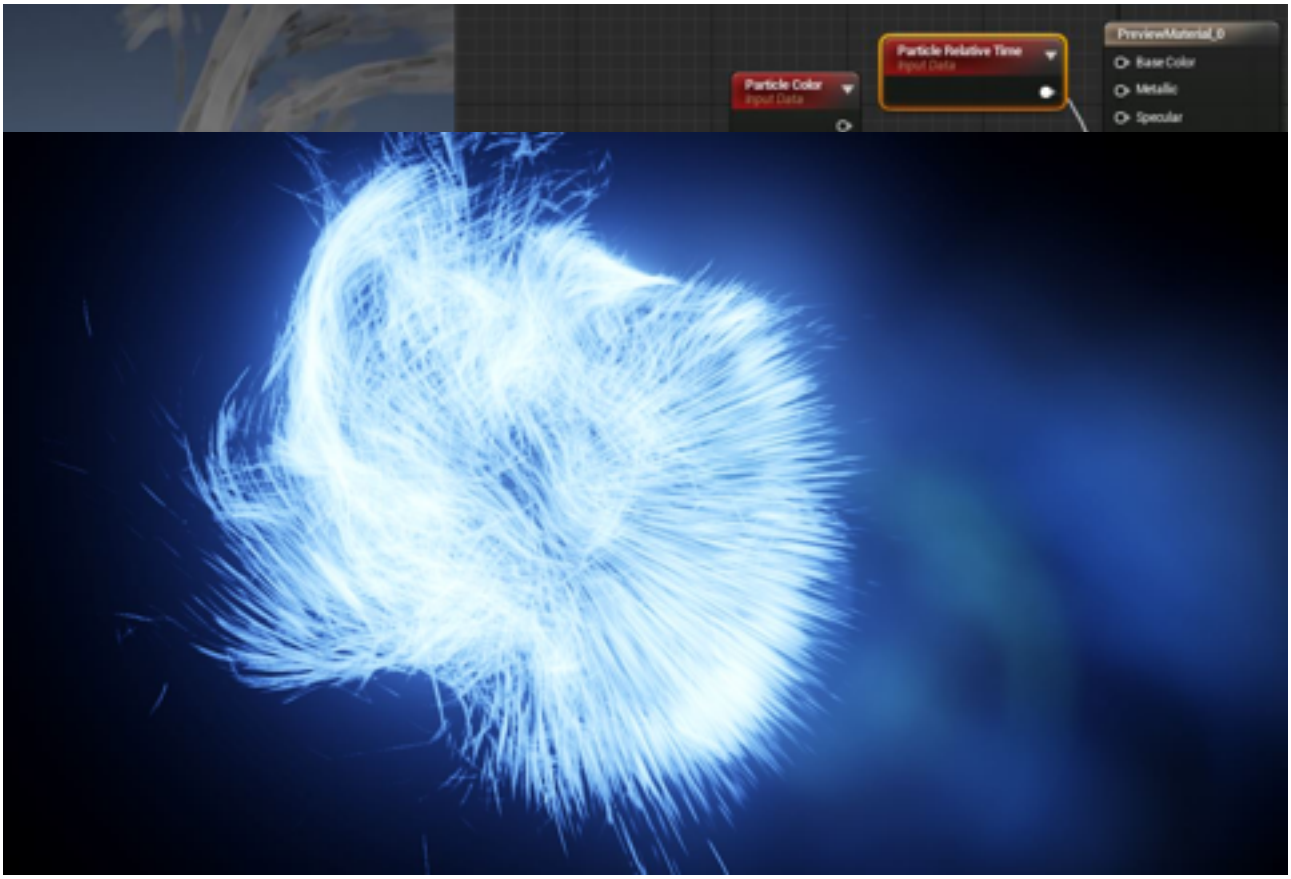
## 6.Particle Relative Time（粒子周期）

粒子周期表达式输出了一个介于0到1的值，0代表粒子产生，1代表粒子消亡。

在这张图片中，你可以看到粒子周期表达式被接入了自发光通道。于是刚产生的粒子是黑色的（值为0），而即将消亡的粒子是白色的（值为1）。

## 7.Particle Size（材质尺寸）

材质尺寸表达式输出了粒子贴图的XY尺寸。这可以被用来驱动材质的某些方面。

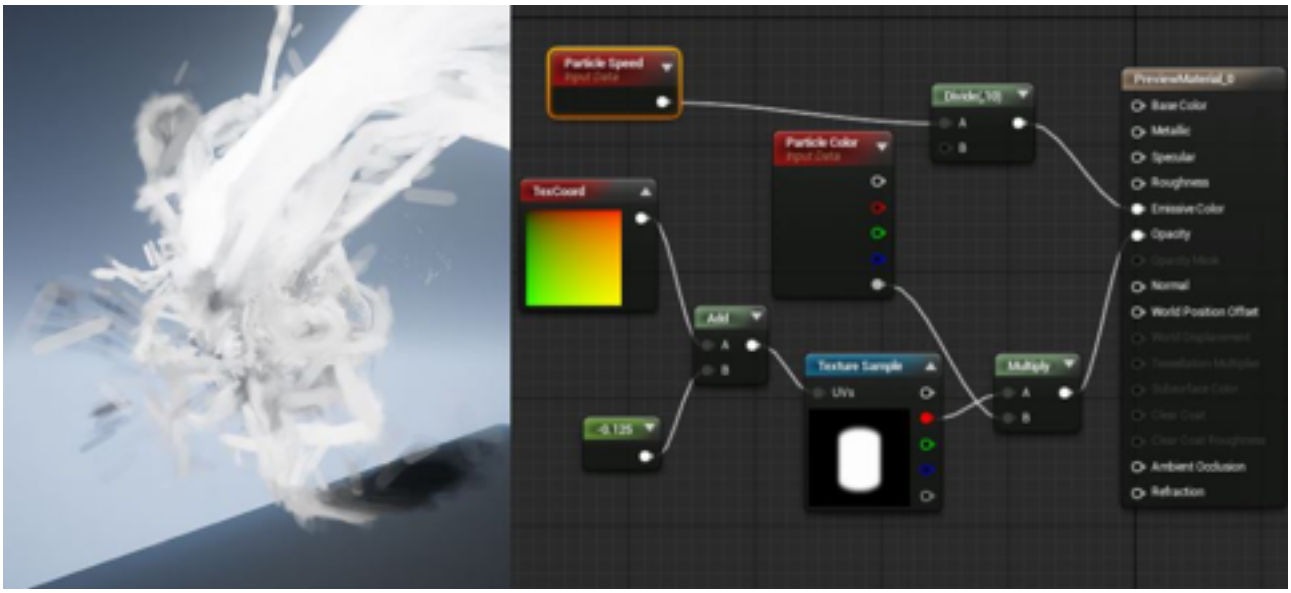



在以上例子中，粒子尺寸表达式被与粒子颜色表达式相乘，注意到我们只提取了绿色通道，代表了Y轴的值，也就是说，当粒子被拉伸的时候，粒子颜色会变的更加明亮，而当粒子被压缩的时候，颜色会变的更为暗淡。

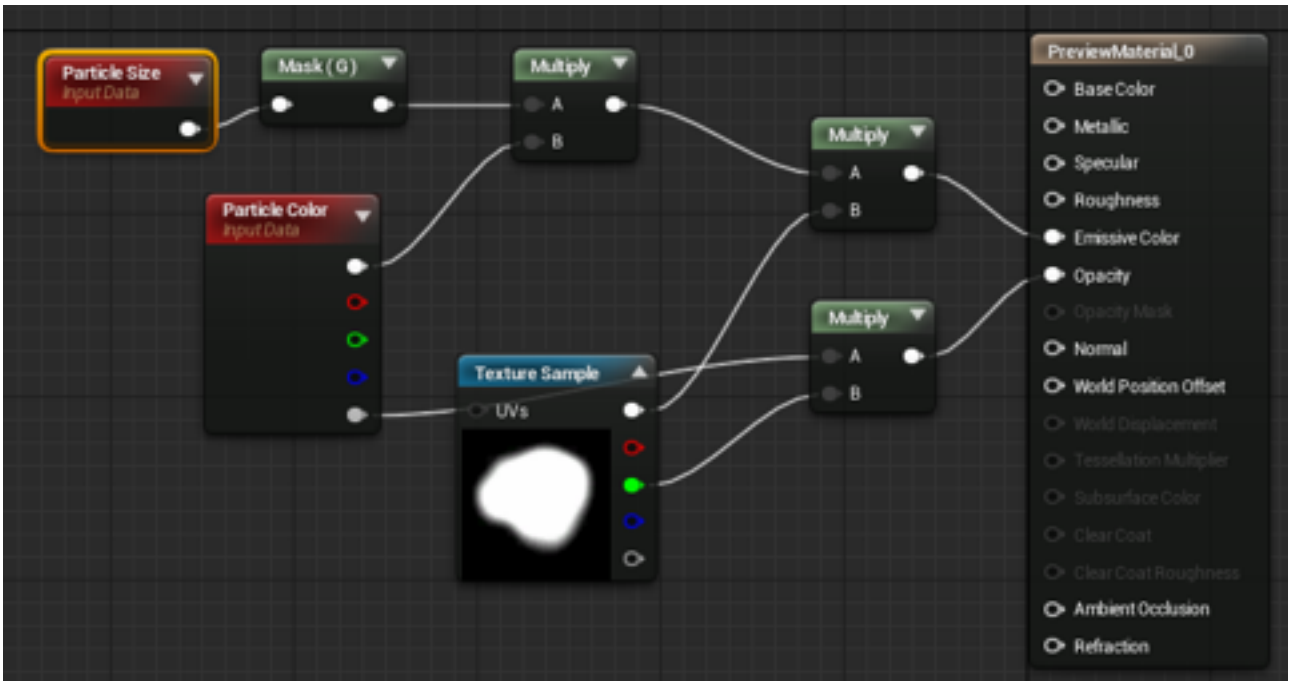
---

## 8.Particle Speed（粒子速度）

这个节点输出了每个粒子的运动速度，单位是虚幻单位每秒。




这个例子中，粒子速度被除以10以得到更有意义的结果，可以看到当粒子速度减慢的时候，他们的颜色更趋向于黑色。




## 9.PerInstance Fade Amount（单实例淡入淡出数）

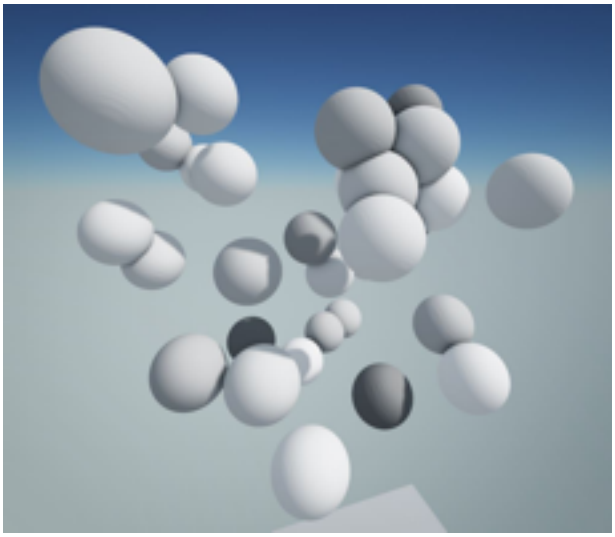
单实例淡入淡出数表达式输出了一个浮点型数值，决定了实例化静态网格物体的淡入淡出数量（像是植物等），虽然它是一个常数，但是可以给每个不同的实例化静态网格物体赋不同的值。

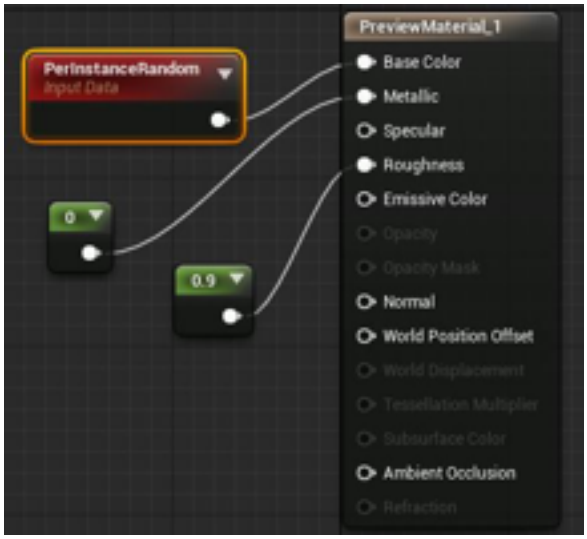
（这个节点只能适用于实例化静态网格物体或包含有实例化静态网格组件的对象。）

## 10.PerInstance Random（单实例随机数）

单实例随机数表达式为赋予这个材质的每一个实例化静态网格物体随机生成一个不同的波动数值。实例化静态网格组件为实例设立了一个随机波动，并且被暴露出来，因此可以根据需求去改变它。（例如窗后面随机的灯光层次），这是一个常量，但是每个实例都会不同。

输出的数值是一个介于0到目标平台所接受的最大随机数之间的整数。

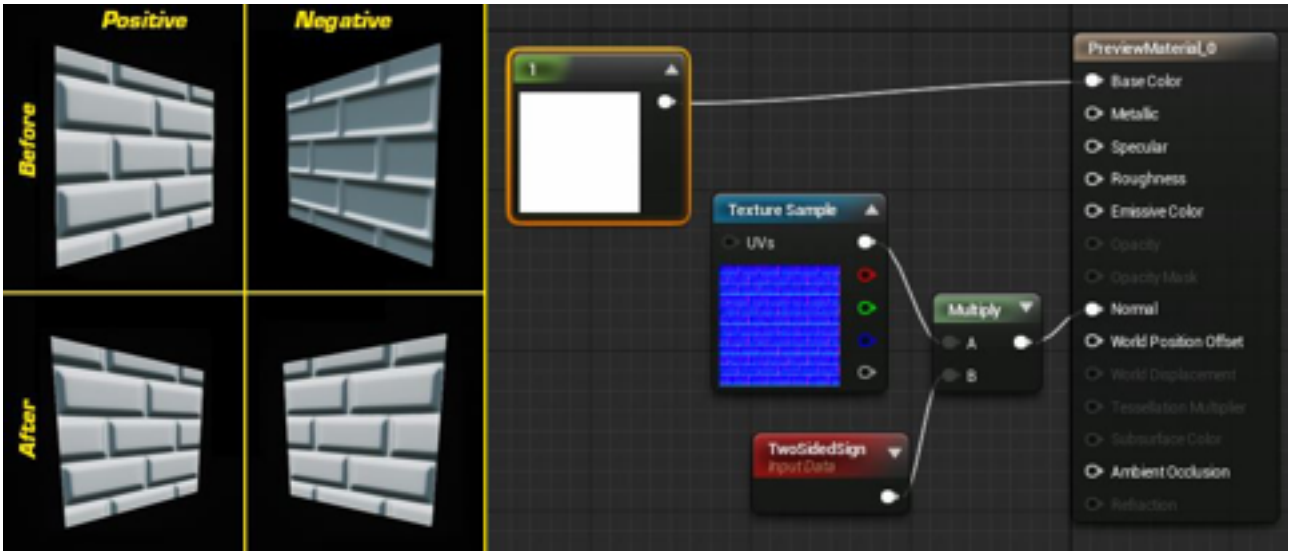





（这个节点只能适用于实例化静态网格物体或包含有实例化静态网格组件的对象。）

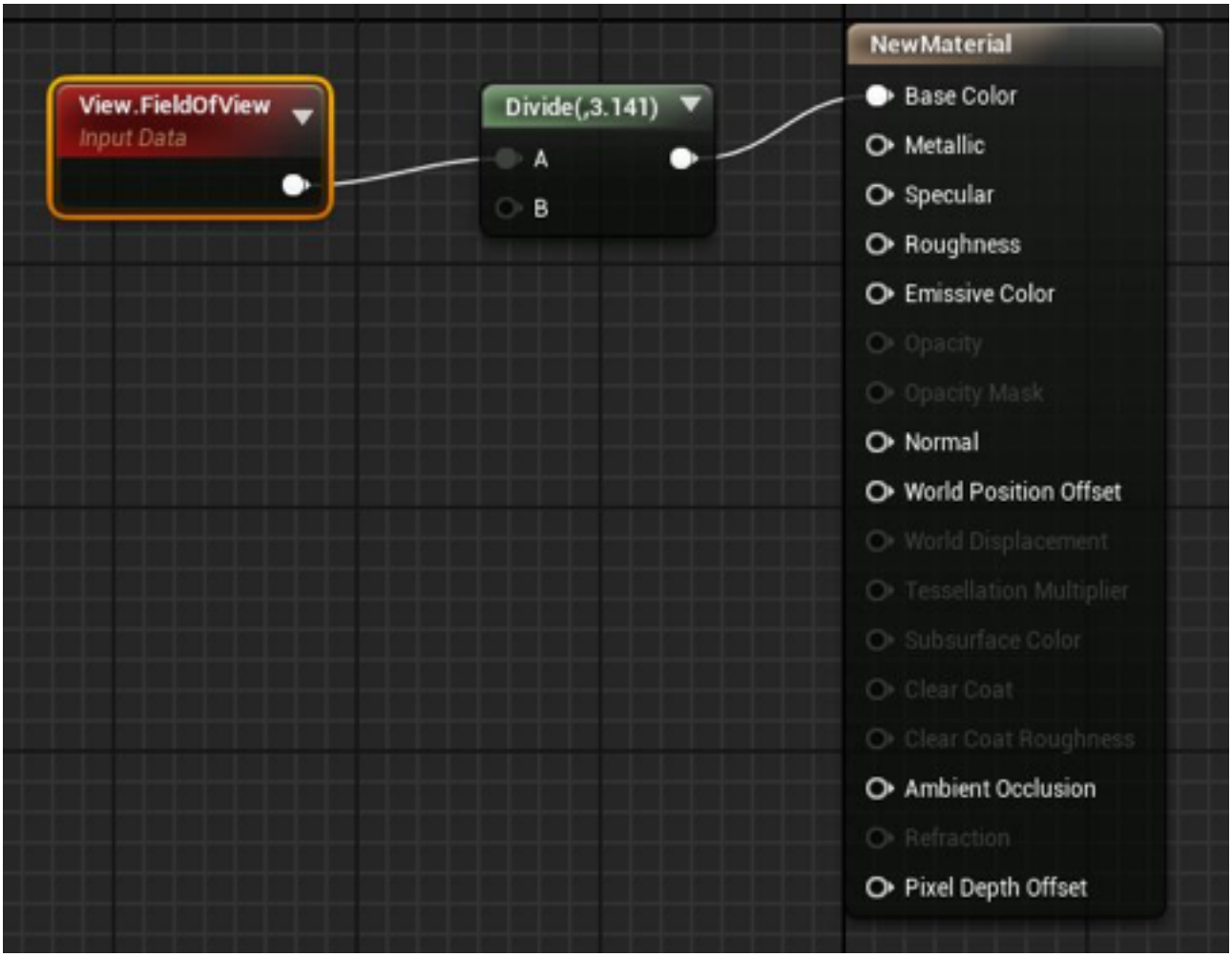
## 11.TwoSidedSign（双面符号）

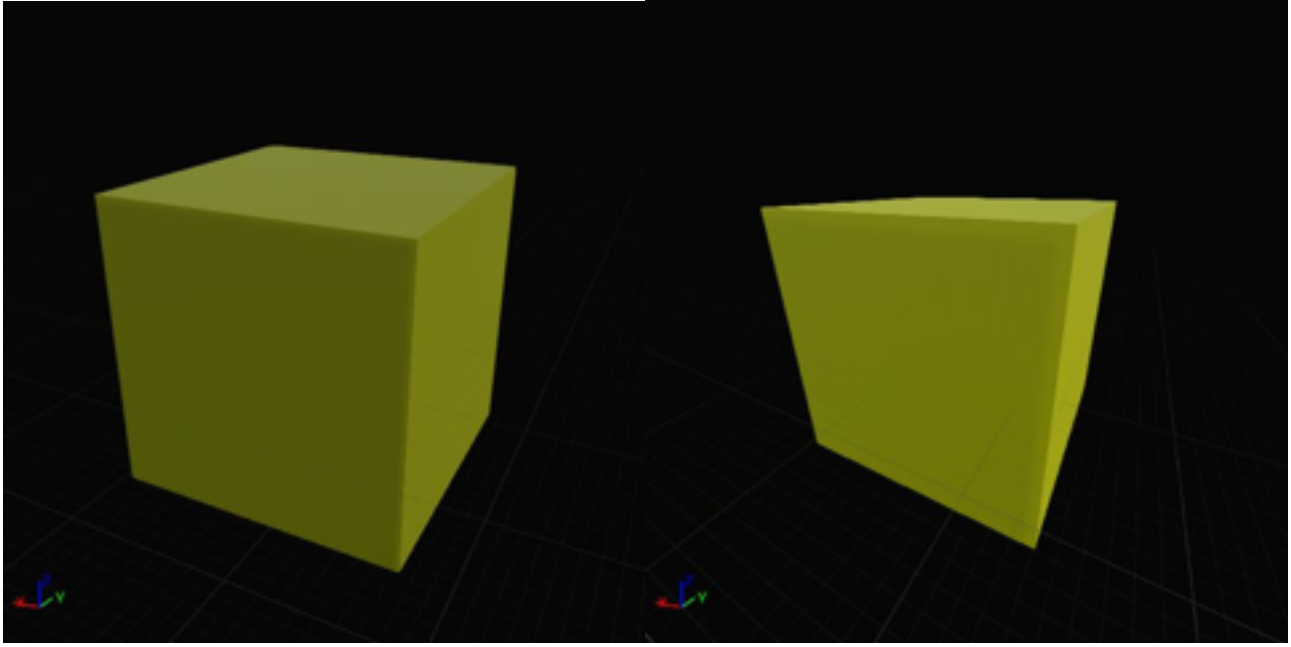
双面符号表达式在翻转法线或者自定义双面光照材质的背面来适配Phong着色（一种着色方式，越南裔美籍人裴祥凤发明）的功能时是非常有用的。对双面材质而言，+1代表正面，-1代表反面。




## 12.View Property（审查参数）

审查参数节点输出了一个从属的常数参数，比如FOV或者渲染目标尺寸。访问的参数是可被配置的，而且输出的类型也基于接入的可配置参数类型而定。





上图分别是45度FOV和90度FOV时的图像。

---

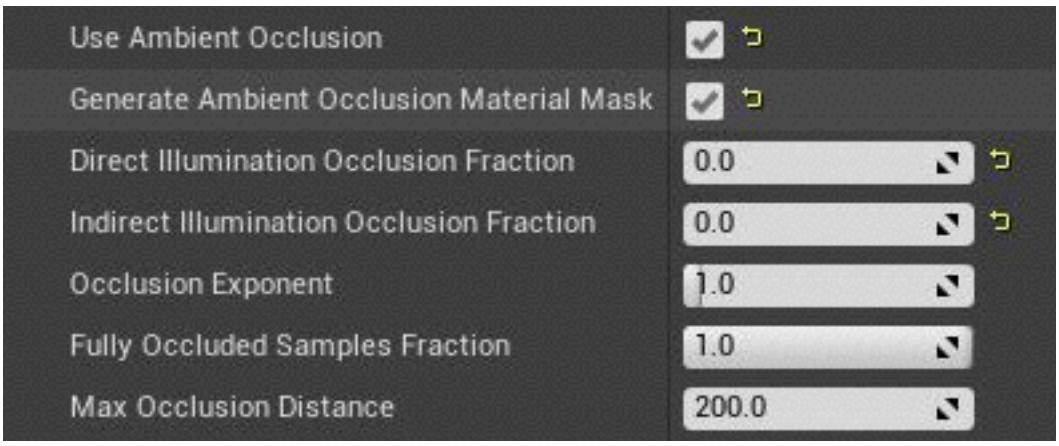
## 13.Precomputed AO Mask（预计算环境光遮蔽图层）

预计算环境光遮蔽图层节点让你能够在材质中获取通过Lightmass计算的环境光遮蔽信息，这在处理贴图或者在需要跟随时间的变动而产生缓慢风化效果的区域中添加拥有时间质感的效果和灰尘时是非常有用的。

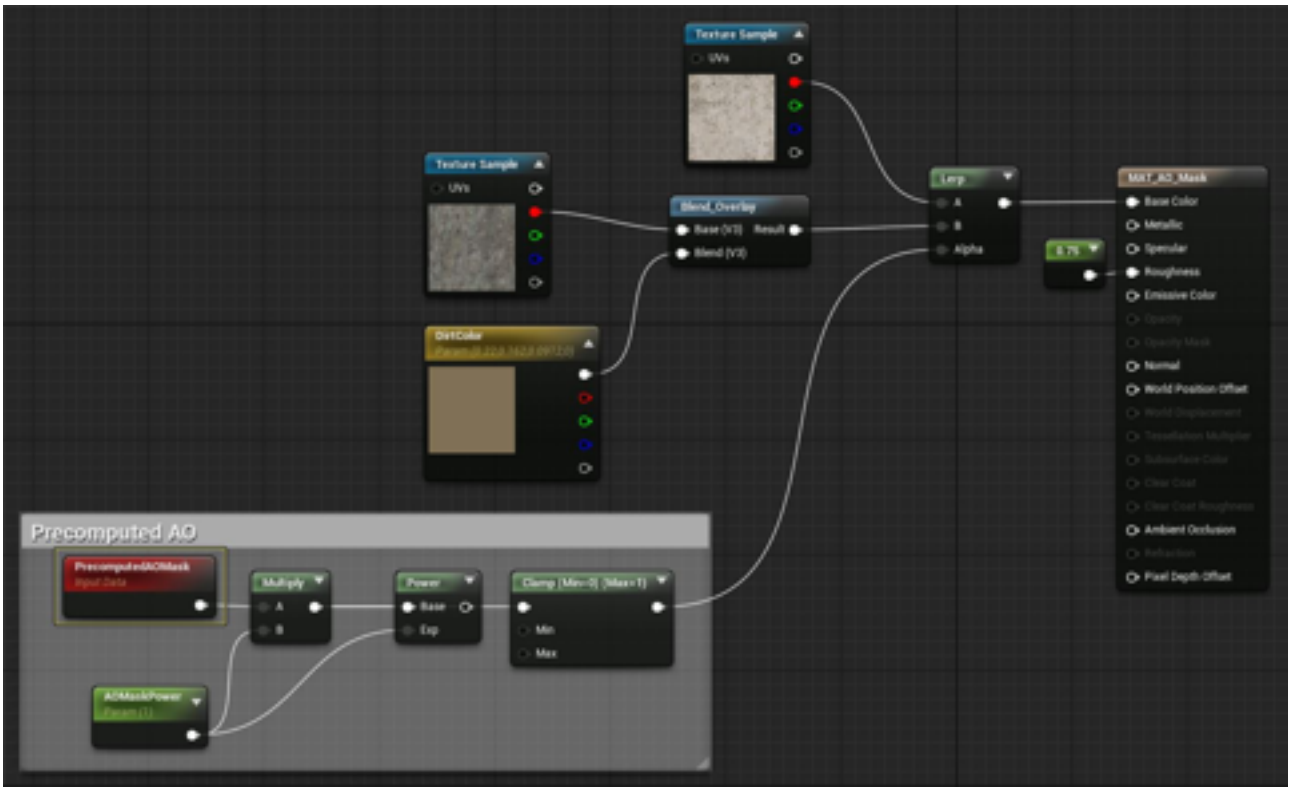
（在你能看到预计算环境光遮蔽图层之前，你需要先用Lightmass为整个关卡构建光照信息。）

上面这张截图用了环境光遮蔽图层来自动在环境的角落里面混合泥土层。为了使用AO层，你要在世界设置－Lightmass设置里同时允许Use Ambient Occulusion（使用环境光遮蔽）参数和Generate Ambient Occulusion Material Mask（生成环境光遮蔽材质图层）参数。其他的AO控制参

数，例如Max Occlusion Distance（最大遮蔽距离）在微调AO的表现时是很有用的。同时，请确保直接和间接遮蔽分数（Direct/Indirect Occlusion Fraction）被设置为0，这样AO层就不会被应用到实际关卡光照中。



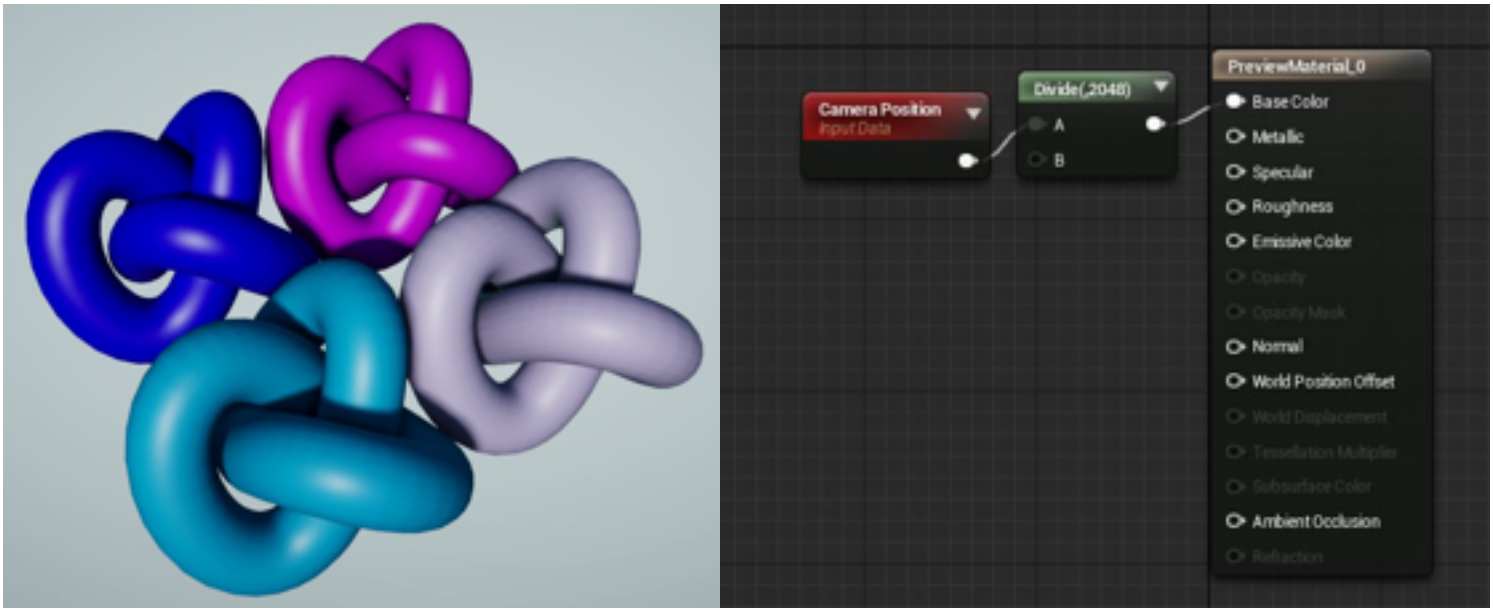

通过使用预计算环境光遮蔽图层材质表达式节点，你可以在任何的材质中访问到AO层，这个图层类似于一个0到1的图层，1代表环境被AO层影响，而0代表不影响，在以下图片中你可以看到如何利用预计算AO图层来建立一个材质。




## Coordinate Expression（坐标表达式）

## 1.Actor Position WS（对象坐标WS）

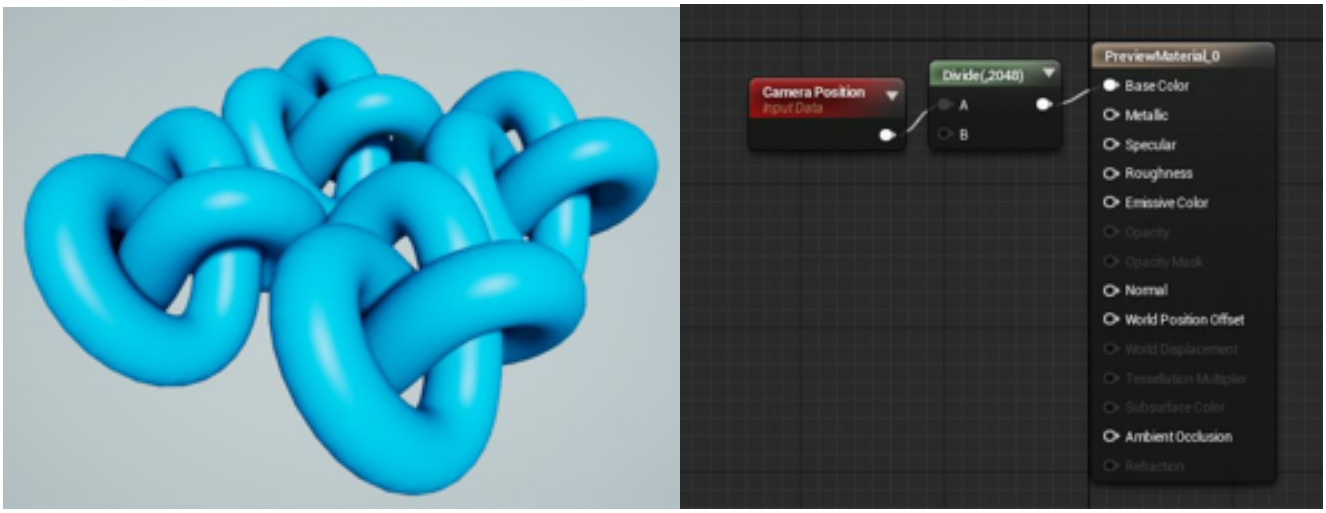
对象坐标WS输出了一个三维向量，代表了利用这个材质的物体的世界坐标。




在这个例子中，可以看到ActorPositionWS被直接接入了基本色通道，因此所赋予材质的物体在不同位置移动时表现出不同的颜色，这个数值被除以1600来获得渐变的效果而不是突然的出现。

## 2.Camera Position WS（摄像机坐标WS）

摄像机世界坐标表达式输出了一个三通道的向量，代表了摄像机的世界位置，预览图中的物体将随着摄像机的旋转改变颜色。




## 3.LightmapUVs（光照贴图uv）

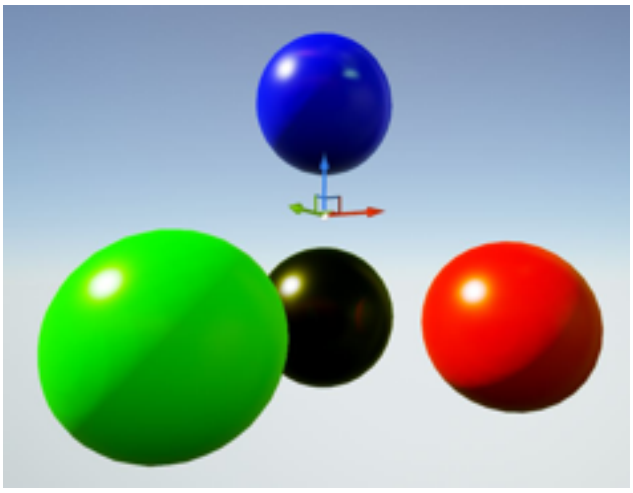
光照贴图uv表达式输出了光照贴图的uv坐标式样（双通道贴图），如果光照贴图不可用，则.会输出一个值为（0，0）的双通道向量。

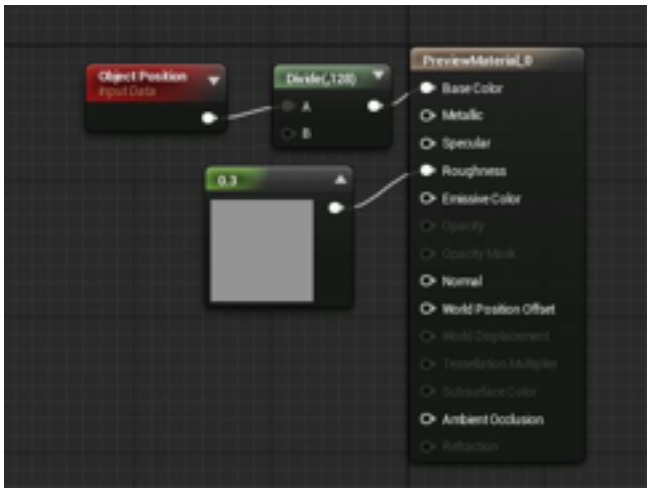
## 4.Object Orientation（物体朝向）

物体朝向表达式输出了一个世界空间中的向上的向量，换句话说，物体的Z轴被表示了出来。

## 5.Object Position WS（物体位置WS）

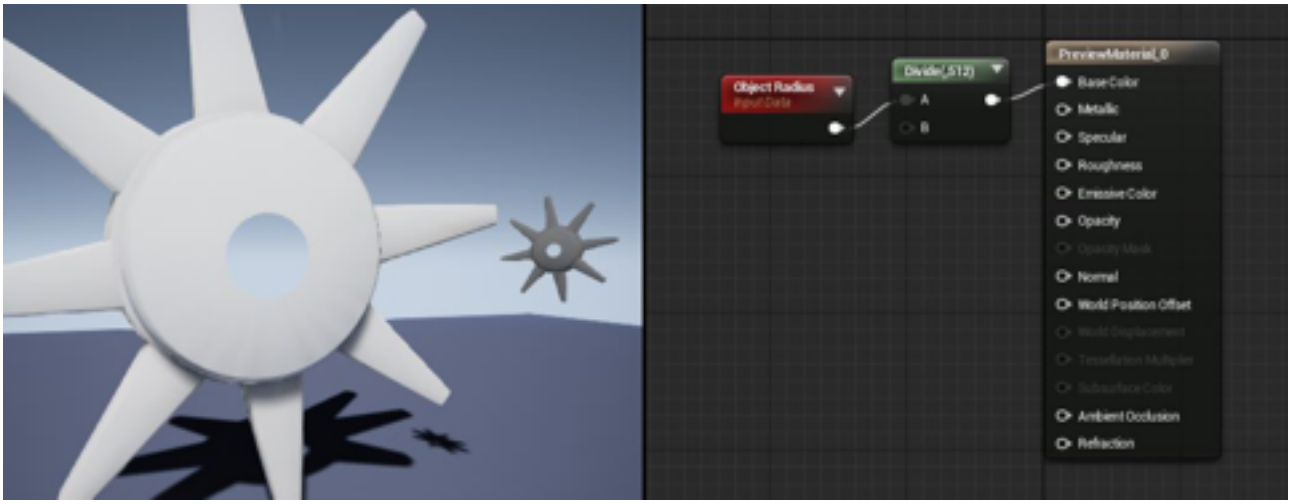
物体位置WS表达式输出了物体的世界坐标位置，比如用于为植物创建球状光照。






## 6.Object Radius（物体半径）

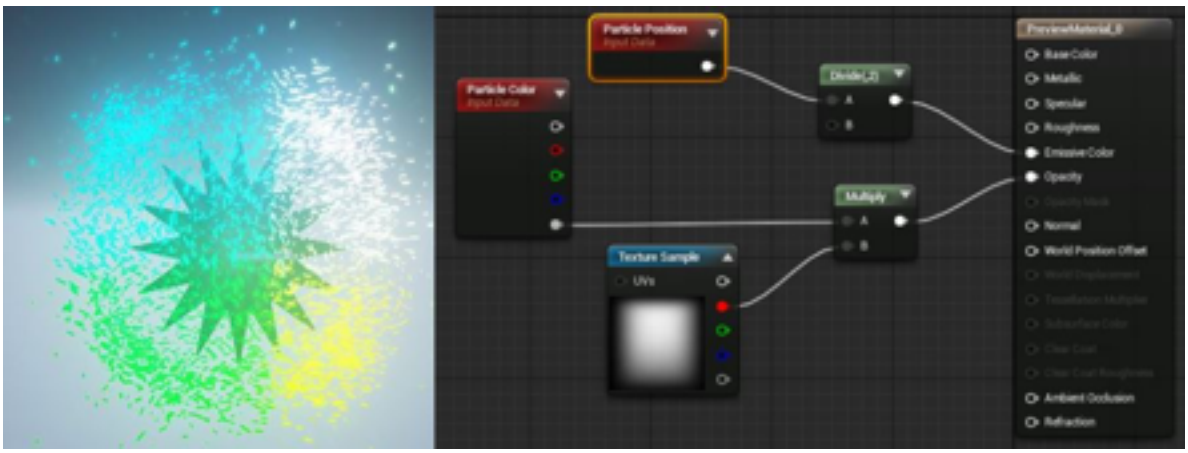
物体半径表达式输出了一个与物体半径相等的以虚幻单位为单位的值，缩放值也会被计算进去，同时每个物体都可以有不同的值。




在这个例子中，两个物体都被赋上了相同的材质，在漫反射通道中接入了物体半径表达式，并且把结果除以512以得到一个更有意义的结果。

## 7.Particle Position WS（粒子位置WS）

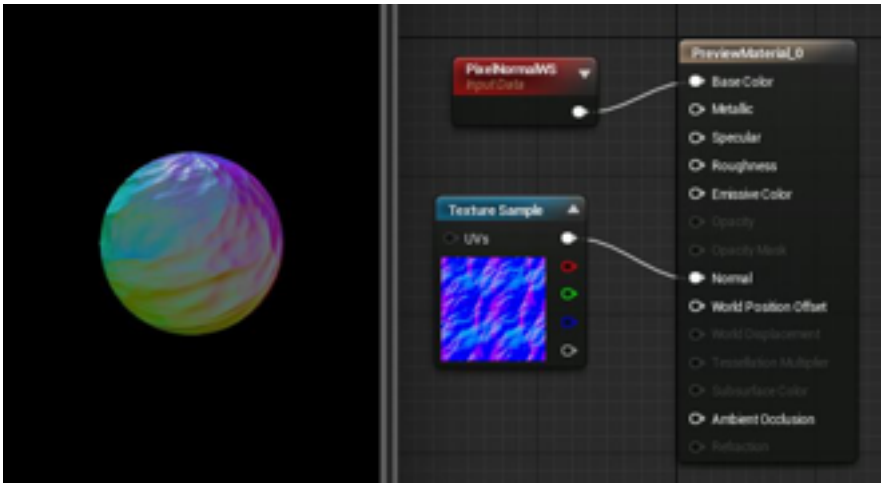
粒子位置WS表达式输出了一个三通道的向量（RGB），代表了每一个粒子的世界位置。




在这张图中，粒子位置WS被接入了自发光通道，粒子被放大以用于显示基于位置的颜色。

## 8.Pixel Normal WS（像素法线WS）

像素法线WS表达式输出了一个向量，代表了基于现有的法线贴图，每一个像素所面对的方向。

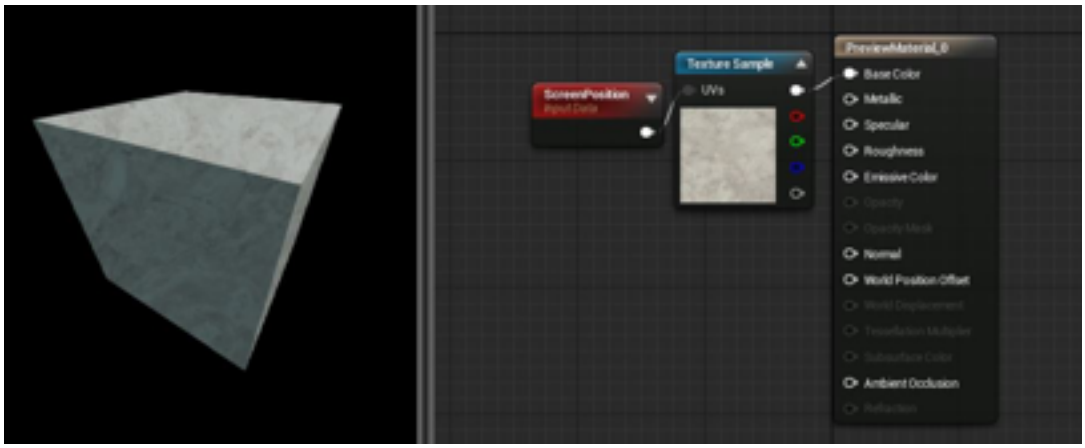



## 9.Scene Texel Size（场景贴图尺寸）

场景贴图尺寸表达式允许你偏移贴图尺寸，当你使Scene Color或者Scene Depth表达式的时候可能会用到，它可以用来在多分辨率的系统中做边缘检测，否则的话你必须使用更小的静态值，可能会导致在低分辨率时出现不统一的情况。

## 10.Screen Position（屏幕位置）

屏幕位置表达式输出了屏幕空间中被渲染的像素的位置。

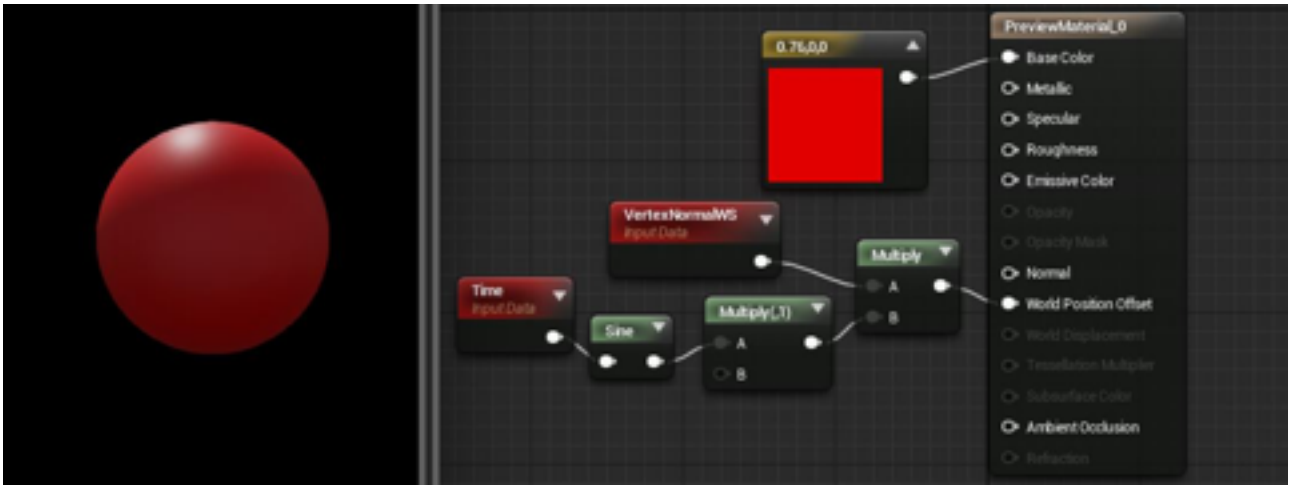



## 11.Texture Coordinate（贴图坐标）

贴图坐标表达式以双通道的形式输出了贴图的UV坐标信息，允许材质使用不同的uUV通道，调整尺寸，而不是改变物体的UV。

## 12.Vertex Normal WS（顶点法线WS）

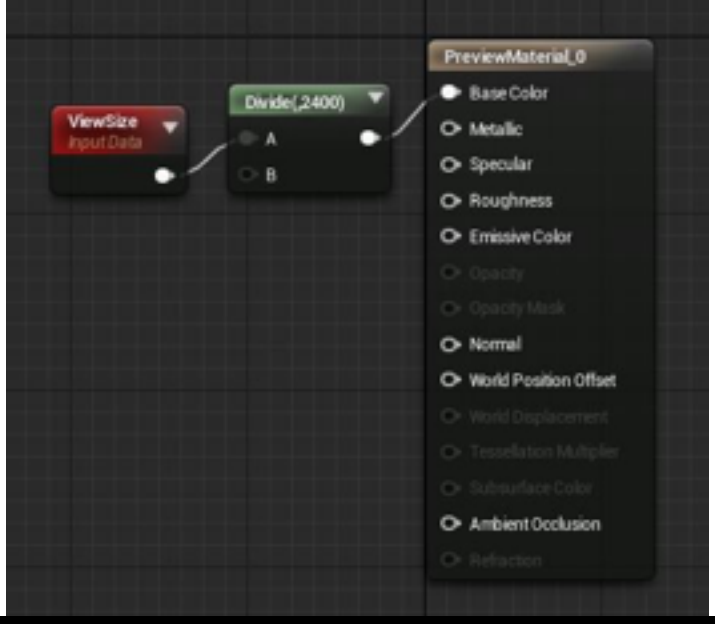
顶点法线WS表达式输出了世界空间的顶点发现，它只能被连接到能够执行顶点阴影的通道里，例如WorldPositionOffset（世界位置偏移），这在创造一个可以变大或变小的模型上是非常有用的。注意，沿着法线的位置偏移会导致几何体在uv的接缝处被切断。

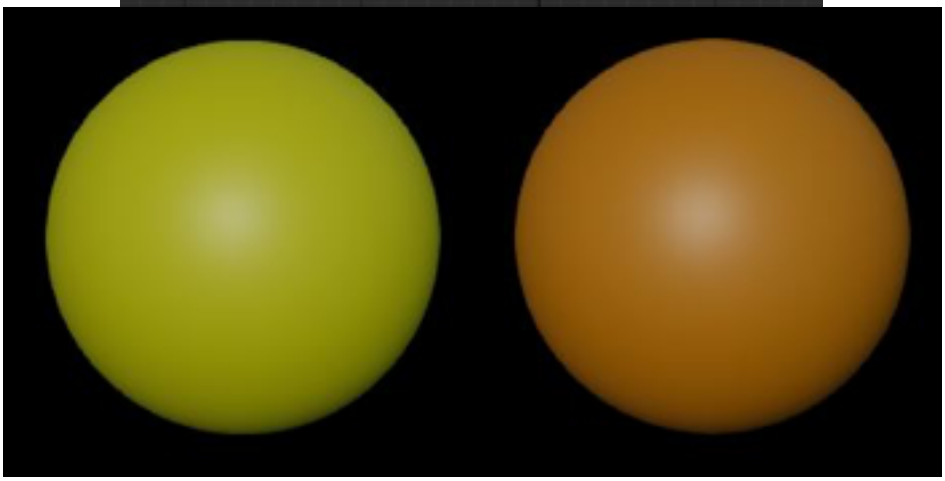



在这个例子中，这个球体会随着时间变大变小，因为基于世界时间做了Sin运算，因此每一个像素都会朝着各自的法线方向移动。

## 13.View Size（视角尺寸）

视角尺寸表达式将输出一个以像素为单位的代表了当前视角尺寸的2D向量，这在做一些基于当前分辨率的不同而进行变化的材质是很实用的。



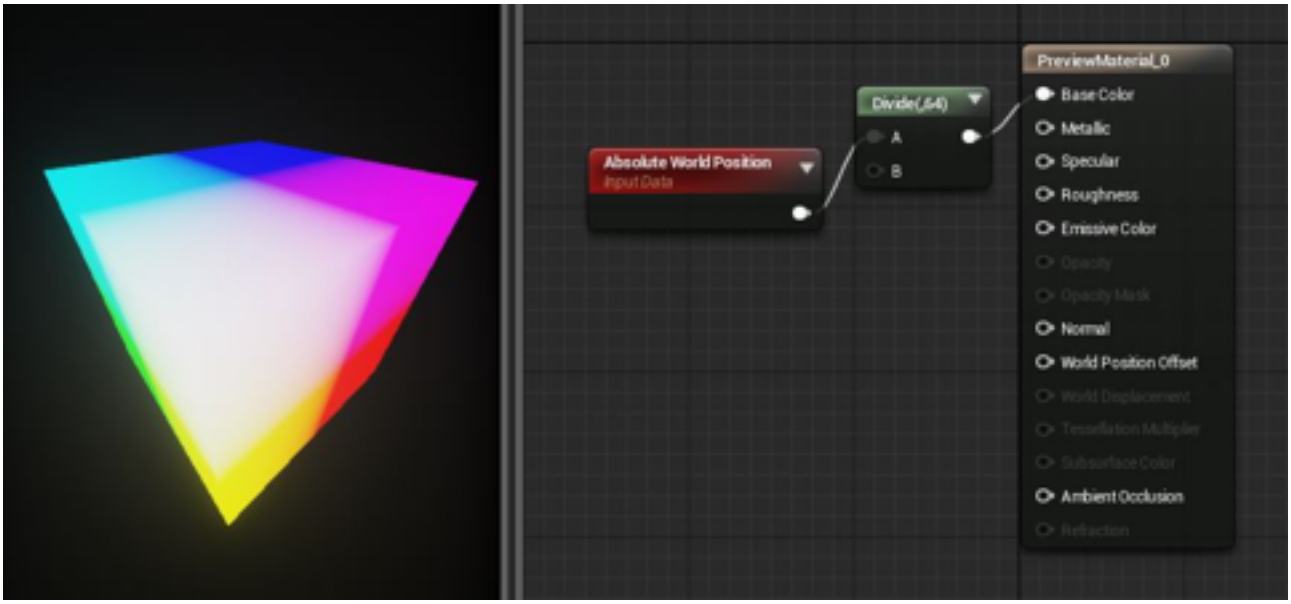



以上分别是720*700分辨率时和740*280分辨率时的图像。表达式被除以2400后接入了基础色通道中。

---

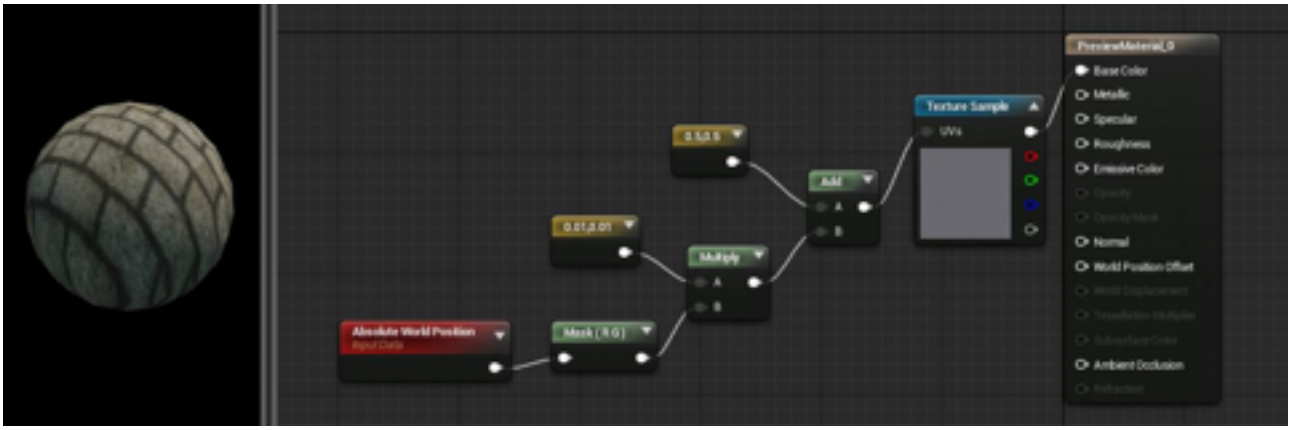
## 14.Absolute World Position（纯世界位置）

纯世界位置表达式输出了世界空间中每一个像素的位置，想要直观一点的话就直接把这个表达式接入自发光通道。



通常的用法是获取摄像机到像素的辐射距离（像是相对于PixelDepth的垂直距离）。

这个节点也被用来在贴图坐标和无关模型相近时适配。这里有个例子就是用AbsoluteWorldPosition.xy来平坦地铺张贴图。




## Custom Expressions（自定义表达式）［进阶］

---

### 1.Custom（自定义）

自定义表达式允许你用HLSL语言编写自定义的着色器代码。自定义任意数量的输入通道和可操作的输出通道。

自定义表达式包含了：

代码：包含了表达式将会执行的着色器编码。（参考下面的“警告”）。

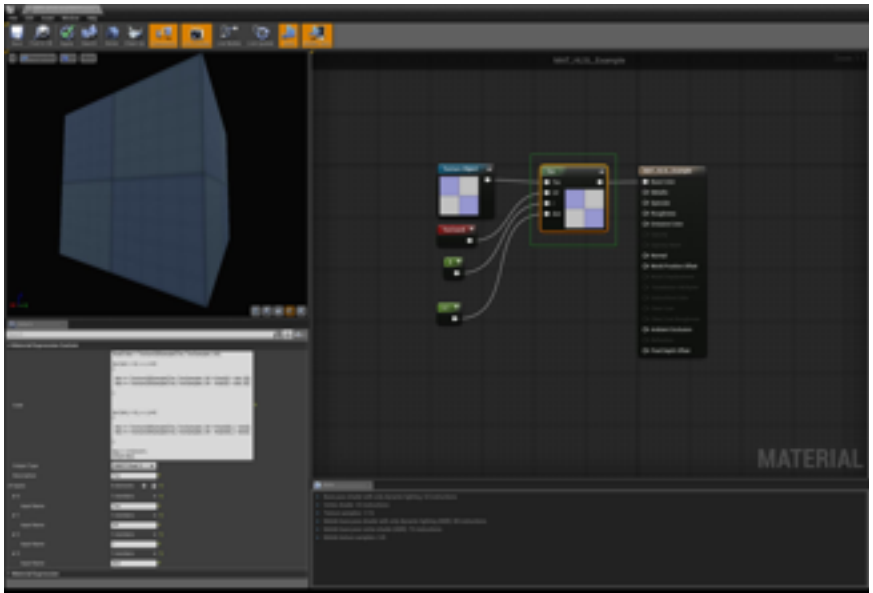
输出类型：定义表达式输出的值的类型。

描述：在标题栏中描述这个表达式。

输入：表达式使用的输入值的数组。

输入名称：定义输入的值的名称，这个名称将会被现实在材质编辑器中的表达式里。也会用于在HLSL代码中引用输入的值。

在输入数组中添加任意数量的输入通道，并且命名他们。你能够在代码模块中编写代码。你既可以写一段完整的函数代码，就像例子中那样包含了返回句，也可以只是一个简单的描述，类似“Input.bgr”。你必须在输出类型中定义输出类型。



以下是上图中自定义节点里面使用的代码，你也可以自己试着写一下。

```
float3 blur = Texture2DSample(Tex, TexSampler, UV);

for (int i = 0; i < r; i++)
{

blur += Texture2DSample(Tex, TexSampler, UV + float2(i * dist, 0));
blur += Texture2DSample(Tex, TexSampler, UV - float2(i * dist, 0));

}

for (int j = 0; j < r; j++)
{

blur += Texture2DSample(Tex, TexSampler, UV + float2(0, j * dist));
 blur += Texture2DSample(Tex, TexSampler, UV - float2(0, j * dist));

}

blur /= 2*(2*r)+1;
return blur;
```

可能出现的警告：

Using the custom node prevents constant folding and may use significantly more instructions than an equivalent version done with built in nodes!

使用的自定义节点阻止了常数折叠，可能会比同样的版本在构建节点是使用相当多的指令。

所谓的常数折叠是虚幻4在引擎的基础上做的优化来在必要的时候减少着色指令数量，比如，一个Time>Sin>Multiplay by Parameter>Add to Something表达式链中最后的Add可能会在单指令时让虚幻4将其合并起来。这是有可能的，因为所有的表达式在所有的Draw Call中都隶属于常量，它们不会随着每个像素而改变。虚幻4不能合并任何的自定义节点，也就是说会导致比同样的版本更低效的着色。（已有的节点）因此，请只有在已有的节点不能满足你的需求时才使用自定义节点。

Shader code written in a custom node is compiled 'as is' for the target platform.

写在自定义节点中的着色器代码被编译成了适合目标平台的模式。

这意味着如果着色器被编译成了PC，那么它将使用HLSL语言的，如果是为PS3平台做的编译，那么它将使用Cg语言。