2012

Infamous2 背景:

PS3 独占

开放世界，城市环境

延迟渲染

支持顶点预烘焙AO和SSAO

静态烘焙的AO可以正常工作,但是它有一些缺点.为了在AO中获得更小比例的细节,与逐个烘焙AO相比,您可能需要更多的细分网格；或者如果您将其存储在纹理中，则它们需要大量内存才能获得足够的分辨率以获取精细的细节，尤其是在大型开放世界环境中。当然，采用任何烘焙方法，您都无法实时移动或更改任何内容。因此，烘焙的AO最适合源和目标都可能是静态的超大规模遮挡，例如从建筑物到街道，小巷和周围的其他建筑物。它不适用于较小规模的遮挡或可能会移动的物体。

另一方面，SSAO完全是动态的，因此可以适应任何移动或变化的情况。但是出于性能方面的考虑，它通常在屏幕空间中具有有限的半径，因此，如果您靠近某个物体，则阴影似乎会收缩，因为阴影不能超过一定数量的像素。而且您没有任何关于屏幕外或其他事物的信息。由于这两种效果，SSAO可以在不同的相机位置产生看起来不同的阴影。因此，SSAO非常适合用于环境光遮挡的非常精细的细节，但不适用于较大的比例。

烘焙的AO和SSAO之间存在间隙，这两种方法都不适合中等规模的遮挡，它大于SSAO半径，但小于网格细分。因此，在我们的引擎中，我们添加了一种混合方法，可以通过中等程度的遮挡来补充烘焙的AO和SSAO。基本思想是预先计算对象投射到其周围空间的AO表示，并将该数据存储在纹理中。这是在世界空间中完成的，因此在所有相机位置上都具有一致的外观。

而且，预计算仅基于源几何体，而不基于目标，因此可以实时移动。它不是完全动态的； 它确实要求光源的几何形状是刚性的。 它在延迟着色中的应用非常像灯光：我们在对象周围绘制一个框，并使用像素着色器评估框内每个阴影点的AO。有两种变体，我们称为AO场和AO贴花，我将依次讨论它们。

让我们从AO 场开始。这是一个演示该技术的视频。（视频位于http://reedbeta.com/gdc）该视频中禁用了SSAO，因此您在这些对象周围看到的接触阴影都是由于AO 场引起的。 我们将其用于许多较小的对象，例如邮箱和盆栽植物，还用于一些较大的对象，例如汽车。 如您所见，它为运动中的对象提供了非常合理的结果。

* + Kontkanen and Laine, “Ambient Occlusion Fields”, SIGGRAPH ’05
  + Malmer et al. “Fast Precomputed Ambient Occlusion for Proximity Shadows”, Journal of Graphics Tools, vol. 12 no. 2 (2007)
  + Hill, “Rendering with Conviction”, GDC ’10

那么这是如何预生成的呢？首先，我们在汽车周围放置一个盒子，并在盒子中放置一个体积纹理。 该纹理中的每个体素都存储一个遮挡锥，该遮挡锥表示从该点观察到整个汽车的方向。RGB分量是平均遮挡方向上的单位矢量，而alpha分量存储该圆锥的宽度，作为遮挡的半球的一部分。

这是汽车周围的遮挡样品的示意图。每个圆锥体代表纹理的一个体素，如您所见，圆锥体指向汽车，它们越靠近则越宽。

AO场：预计算

遍历所有体积纹理的体素

以每个体素为中心, 渲染几何体到32x32 cubemap中

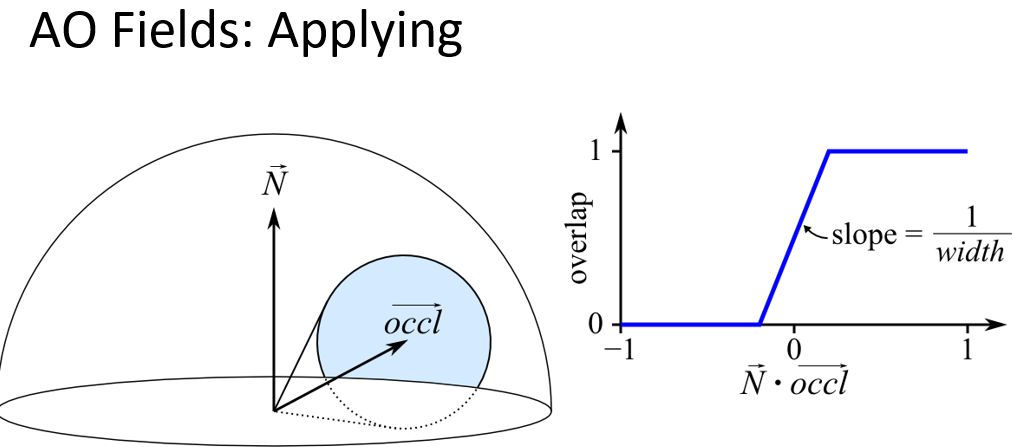
读取cubemap纹理，计算在上一步中被覆盖的纹理的立体角加权的平均方向

计算（以平均方向为中心的）半球方向上被遮挡的纹理比例。

现在，我们需要实时应用此功能。就像是延迟着色中的光源：绘制AO场的包围盒，然后在像素着色器中，对G缓冲区进行采样，以获得着色点的世界空间位置和法向矢量。所有常用的延迟着色优化都可以使用，例如模板遮罩或深度边界测试。有了世界位置后，我们将其转换为场的局部空间，对体积纹理进行采样以获取遮挡矢量和圆锥宽度，然后将遮挡矢量转换回世界空间。

使用下列方程计算AO:

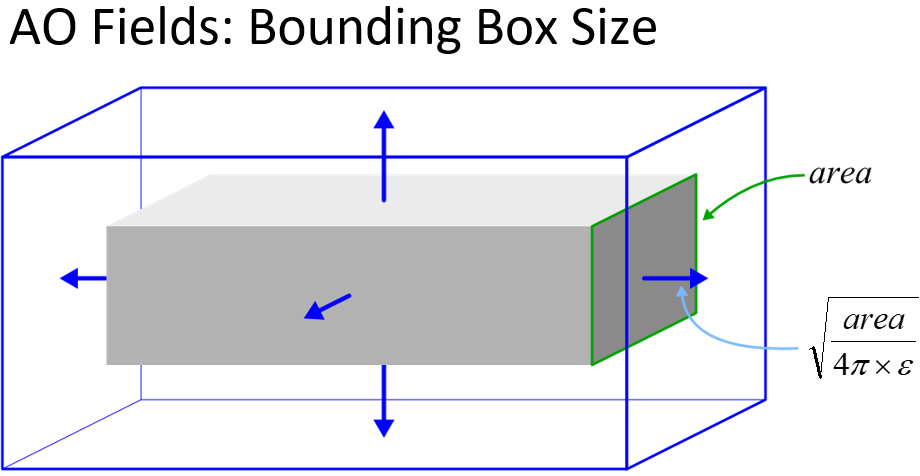
其中是着色点的法线,是存储在体积纹理中的平均法线,是存储在体积纹理中的w值,是控制AO强度的参数.

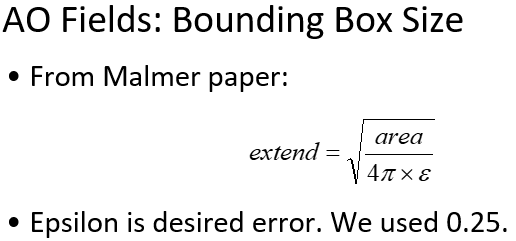


这是该术语的功能示意图。它近似是遮挡锥和正常半球之间重叠的比例。锥可能不完全位于半球内，在这种情况下，我们不应该应用整个遮挡值。先前的方法在这里使用了更复杂的函数或查找表，但是我们只是通过基于法线和遮挡矢量的点积的此钳位线性斜坡对其进行近似，而斜率基于圆锥宽度。 这是一个很粗略的近似值，但以我的经验来看，效果很好。

有了AO值后，我们只需将其乘以融合到G缓冲区的AO通道中即可。对于解决双重融合问题，我们没有做任何特殊的事情-在我们的用例中，通常我们的AO场没有重叠太多，因此这将是一个问题。

现在讨论技术细节。第一个问题是我们应该将包围盒做成多大？我们使用了参考文献之一建议的方法，Malmer论文。在这里，灰色框是我们的汽车或任何源对象，蓝色框是AO场。为了获得AO场的大小，我们从源对象的边界框开始，然后通过根据该面的面积将每个面推出一个距离来展开它。epsilon是一个理想的误差–即，由于在有限距离处切断AO场而导致的误差（因为理想情况下它将永远持续下去）。





AO场：纹理细节

**Texture size: chosen by artist, typically 8–16 voxels along each axis**

Car: 32×16×8 (= 16 KB)

Park bench: 16×8×8 (= 4 KB)

Trash can: 8×8×8 (= 2 KB)

**Format: 8-bit RGBA, no DXT**

Density so low, DXT artifacts look really bad

No mipmaps necessary

不幸的是，所有这些都显示了一些artifaces，我将讨论我们如何解决它们。首先，您会注意到AO场是由于场在有限距离处截止，因此遮挡在其边缘不会一直为零，因此您可以看到汽车周围非常明显的盒形阴影.

我们通过强制所有的alpha值（即遮挡锥宽度）在边界处为零，以最简单的方式解决该问题。我们遍历边缘体素并找到最大的alpha，然后线性重新映射所有alpha以将最大值发送为零。这是执行此操作的方程式。

我们看到的另一个artiface是偶尔在对象表面上出现不正确的自闭塞斑点。根本原因是，当您靠近表面时，遮挡会迅速变化，而低的体素密度无法很好地捕捉到这一点。这是车上的AO。每个带圆圈的区域都包含一个错误的自我遮挡的深色斑点。

理想情况下，我要修复此问题的方法是检测几何体内部的体素并对它们进行某种修复。但是，识别体素的内外在3D中并不是一件容易的事。我们不能依靠我们的几何形状是2流形或类似的东西，因此定义内部和外部不是一件容易的事。因此，我选择了一种可以简单解决最严重问题的技术。我们只是实时地使采样点偏离接收表面。对体积纹理进行采样时，将采样点沿接收器的法线推动固定距离，该距离是体素大小的一半。这有助于使样本点远离问题区域。

AO Decals

AO场的平面版本

使用情况：细物体嵌入到或来源于平坦的表面投影（墙面，地板，窗户等等）

AO Decals：预计算

存储在2D纹理，与墙面/地面平行

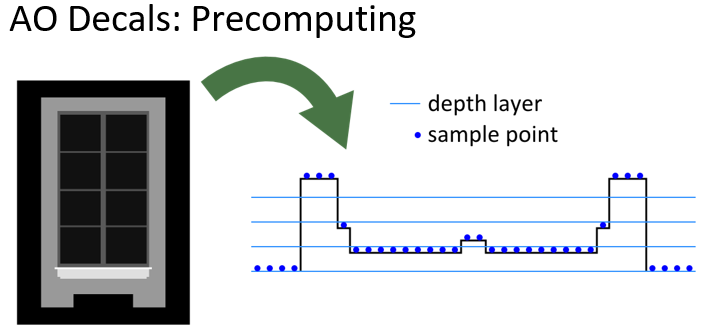
在RGBA通道中存储四层深度：不需要方向，仅仅存储墙面上半球的遮挡系数

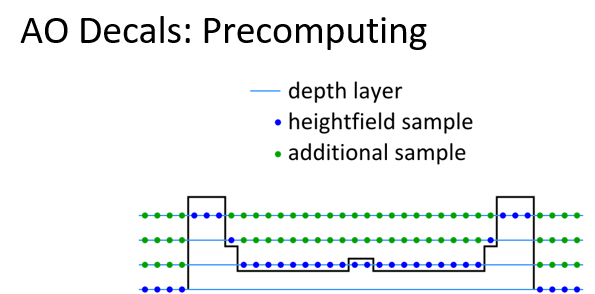
**渲染原始几何的高度贴图**

平行投影正对着墙壁/地面

以灰度绘制几何体，颜色从黑到白由远及近的变化

遍历所有纹素，取正上方高度贴图所对应的AO样点，如下图所示





**将样点分配到最近的深度切片**

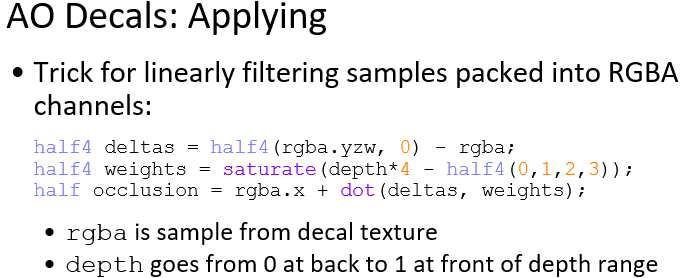
深度切片在深度范围的位置:

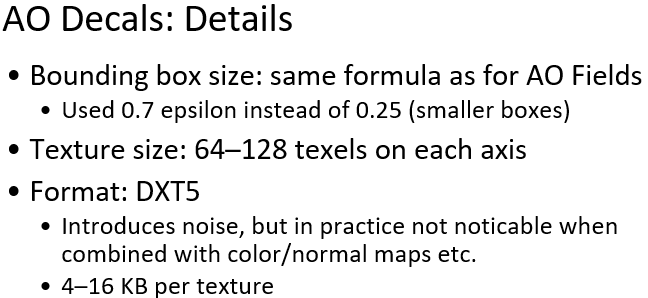
最前面的深度范围(i=4)总是0，总是被遮挡

**在高度图上方至深度范围的顶部进行其他采样**

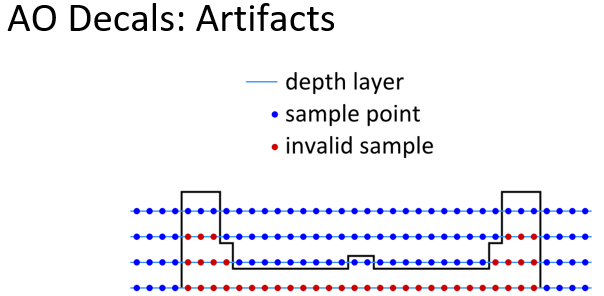
AO Decals:应用

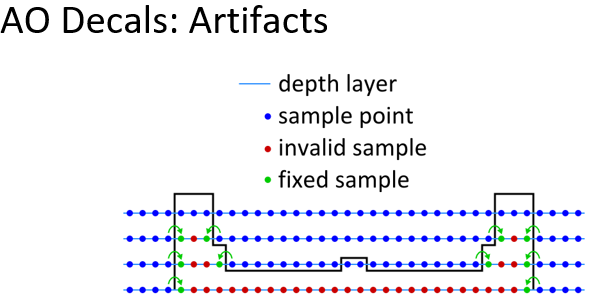
现在，当我们实时应用AO贴花时，与AO Fields几乎相同；我们再一次绘制一个框并执行像素着色器以评估每个着色点的遮挡。但是，我们现在没有方向信息，因此AO值仅为。





用AO贴图显示的伪像是我们在源几何体的高度变化周围得到光晕。这是其中一个伪像的屏幕截图；您会看到窗框与墙相交处的白线，以及图像中其他地方的各种相似白线。这与我们对AO域的不正确的自闭塞问题非常相似。这是由双线性过滤引起的，因为高度变化可能落在两个纹理像素之间，因此我们最终将高度图上方的遮挡值与高度图下方的遮挡值混合。

我之前提到过，我们对高度图下的采样点有特殊处理。在预计算步骤中，我们将所有这些点标记为无效； 然后，一旦我们获得了所有其他样本，我们将执行一个扩展步骤，将有效样本传播到相邻的无效样本上。



单位矢量D是贴花面的方向

将此乘以源几何体上的顶点AO

有关性能的部分每有记录到该文档中。