# **基于GPU的快速烘焙AO贴图**

作者：[韩宇飞](http://gad.qq.com/user/index?id=2044697)

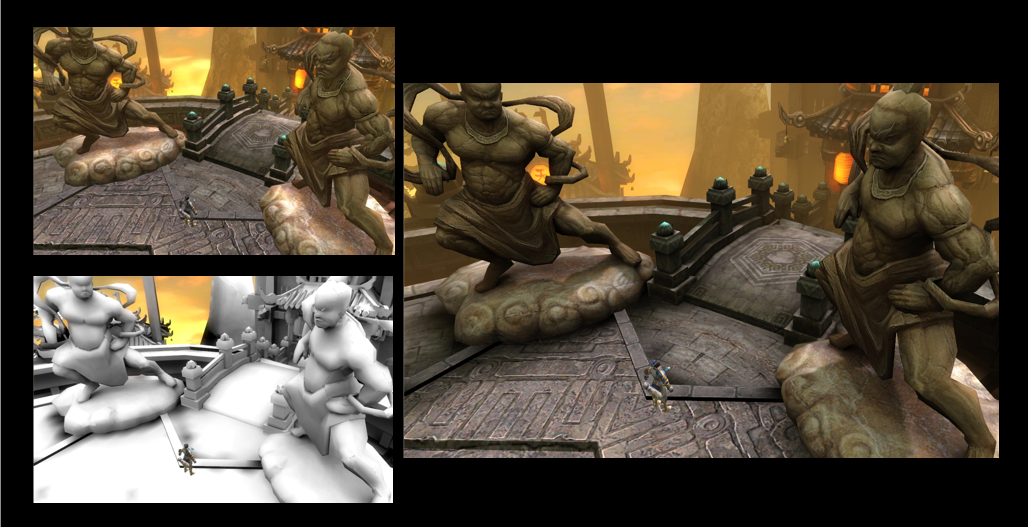
链接：<http://gad.qq.com/article/detail/42107>

**本文首发于知乎专栏：[MACK的游戏开发笔记](https://zhuanlan.zhihu.com/c_157429371)，欢迎各位关注。**

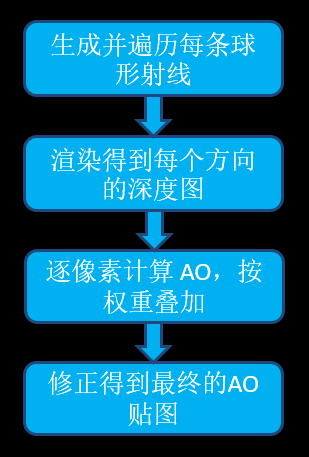
这是之前专利文档的详细介绍。在12年的时候为了进一步提升游戏品质增加场景的立体感，美术希望引入AO效果。因为自研场景编辑器本身就包含了烘焙Lightmap和Shadowmap的功能，速度也非常快，因此我们希望进一步扩展增加烘焙AO的流程。

首先我们尝试集成了Beast来进行烘焙，但是集成之后烘焙一张场景需要二十几个小时的时间，即使只烘焙AO速度也非常慢需要接近一小时（当时Beast并不支持联合烘焙）。因为游戏有上百张地图，美术同学的期望是所见即所得，能在几分钟内看到效果并且可以实时调整AO贴图的浓度效果，因此还是决定使用自研的烘焙方式并在增加一种自研的快速烘焙AO的功能。

首先我们尝试了在CPU中通过发射射线的方式计算AO，效率也是极低。然后又尝试了利用GPU，使用SSAO的方式烘焙静态AO，实现后发现速度还可以但是精度效果无法满足。  
最终我们自己想了一种使用阴影算法来烘焙AO的近似算法。大致流程就是对烘焙物体产生一组球形分布的射线，每个射线单独对物体进行阴影的烘焙（使用游戏内实时阴影的算法，非常快），然后将各个阴影贴图进行权重叠加得到一张混合阴影图模拟AO效果，而这个任务也落到了我身上。比较有意思的是这个方式当时并不是参考其他论文方案实现的，而是自己尝试发现的。但是在实现做过分享之后，过了两个月其他公司就拿相似的文档去申请专利，连一些描述参数都是一模一样的…..实在很难说是巧合，好在最后还是成功申请了专利否则真的是尴尬了。  
  
**什么是AO**  
AO，Ambient Occlusion，一般称作“环境光遮蔽”。百度百科的描述，AO是来描绘物体和物体相交或靠近的时候遮挡周围漫反射光线的效果，可以解决或改善漏光、飘和阴影不实等问题，解决或改善场景中缝隙、褶皱与墙角、角线以及细小物体等的表现不清晰问题，综合改善细节尤其是暗部阴影，增强空间的层次感、真实感，同时加强和改善画面明暗对比，增强画面的艺术性。  
简单来说就是通过简化算法模拟全局光中的部分光照，可以提升场景的立体感，纵深感，看上去更真实。  
  
下图就是不使用AO，单独显示AO，最终效果的三张截图：



**算法原理：**  
常规的AO是通过每个点向周围（全球或半球）发射n条射线，判断点到交点的距离作为权重并累加作为AO值。而我们的做法是对每个物件生成一组球形分布的射线，可以理解为球形光源，每个光源产生一个阴影值，这个阴影值就作为这个点是否被遮挡的结果，然后叠加各个方向的结果产生出一张近似的AO贴图。因为采用的是实时阴影的算法，整个计算非常快，只要几百ms即可得到结果。  
  
**计算流程：**



**实现细节：**

* 生成球形射线。依据球谐光照中生成均匀分布的球形射线的方法，生成900个（实际测试得到的性价比最高的值）射线。首先为了批量烘焙，针对每个可能的分辨率创建AOMap的RT，为了保证精度采用FLOAT\_COLOR\_128
* 渲染深度图。遍历每条射线进行烘焙，传入模型根节点，包围盒，贴图，宽度，权重（即光源数分之一），射线方向。依据方向和固定位置计算出摄像机的位置，得到深度相机，设置View矩阵，设置视口。为了保证物体所有点的AO摄像方向一致，需要依据物体的包围盒生成平行投影矩阵，将物件包围盒转到View空间计算。在VS里输出屏幕坐标和UV，在PS里将屏幕坐标归一化，通过Pack压缩得到深度图，需要采用基础贴图得到Alpha
* 计算AO。遍历每条射线，设置垂直向下的平行投影摄像机，六面体正好照到物件。设置深度贴图，设置RT（第一次需要ClearAll），设置Camera和Shader进行烘焙。VS里直接将UV转换成投影坐标，-1到1（其实就是去渲染UV贴图上像素对应的3D点），输出世界坐标和世界法线。在PS中，计算光的方向和该点的法线的点积作为权重之一，计算出点光的投影空间的坐标，归一化得到-1到1，然后再转换成0到1，注意y需要取反用1减（贴图坐标方向和坐标轴方向的不同的原因）最后得到该点在贴图上的坐标，然后解压深度贴图，求的该点光空间的深度与自己的深度相比，如果光空间的深度值大说明该点被遮挡中，AO值为1，再乘以方向的权重，光数目的权重，系数，最后输入到rg通道中，ba通道设置为1作为mask
* 修正。因为像素溢出的问题，得到的AO贴图会有黑边。因此还需要进行修正，修正方法就是将AO贴图再渲染到一个相同的大小的RT中，在PS中通过Mask进行区分，有效像素直接写阴影值，无效像素则采用周围8个像素，得到有效像素取平均值，相当于扩充了一个像素。

**工作流集成和优化：**  
完成了AO的算法只是万里长征走完了第一步，而真正要投入使用，做到产品化还要需要做大量的后期工作，所消耗的时间远超AO算法本身的时间。

* 在编辑器中加入批量烘焙AO，单独对某个地形，某个物体，以及某些选中物体的功能，美术同学既可以对整张地图进行一次性烘焙，也可以针对修改过的物件进行单个烘焙  
  加入了对半透明物件烘焙AO的功能，需要计算半透明度
* 加入了烘焙AO的进度预测，时间统计信息等功能，便于测试性能和给美术同学一个精准的预期
* 在编辑器和客户端中增加了AO快关，只显示AO等调试功能，便于对比效果和检查错误
* 烘焙出来的AO效果可能并不满足美术同学对场景风格的预期，因此又增加了Gamma矫正，纯黑指定，整体加亮加暗，颜色过滤等功能，并提供了相应的数值和曲线编辑面板。美术可以实时修改AO贴图的颜色曲线，调整到一个比较满意的后期处理效果
* AO贴图分辨率优化，实际使用过程中会发现，如果贴图的分辨率太小UV不合理就会出现采用错误，如果分辨率太高又会占用大量的硬盘和内存空间，因此又加入了依据物件的包围盒，场景大小，人工系数自动计算AO贴图大小的算法，同时也允许美术单独或批量指定AO贴图的分辨率
* 实际测试中发现对于大场景烘焙物件的速度会呈指数下降，主要瓶颈在裁剪上，因此又引入了四叉树来快速剔除，对地形的AO贴图不进行过滤去黑边的操作等优化方式，大幅提升烘焙速度
* 实际使用中发现烘焙速度很快，但是从编辑器导出AO贴图到地图数据的过程非常慢，测试发现是因为dds压缩的功能呢，因此使用了nvidia的新的dds压缩库nvtt，并使用了cuda加速。但发布版本给美术使用之后发现优化过的编辑器导出的AO贴图经常会出现诡异的错乱问题。最后确认是Cuda加速的bug，如果美术使用的显卡不是Nvidia的图形显卡（ATI的显卡和Nvidia的民用显卡都有概率会出问题），或者在烘焙的过程中锁屏了就会出现错乱，最终放弃了Cuda加速
* 实际使用中发现客户端的AO效果会有白斑问题，又修改了烘焙AOShader的贴图采用方式，消除了白斑
* 还加入了生成AO时进行高斯模糊过滤，可以得到更加平滑的AO贴图
* 为了降低客户端容量，我将Ligmap分成两张，一张贴图的G通道存AO，R通道存阴影，另一张的存点光的光照和阴影，两张贴图都使用Dxt1的压缩方式，极大的节省了安装包和内存的大小。

**最终效果：**  
最终完成了GPU烘焙AO的功能，并集成到了美术的工作流中。同一张地图使用Beast烘焙AO需要45分钟，而使用自己开发烘焙方式只需要2分钟，单个物件不到1秒，极大的提升了美术同学的开发效率。

