我们先支持聚光灯的实时阴影。

首先，重构GetOtherShadowAttenuation和ReserveOtherShadows的代码，像方向光一样变成混合模式。

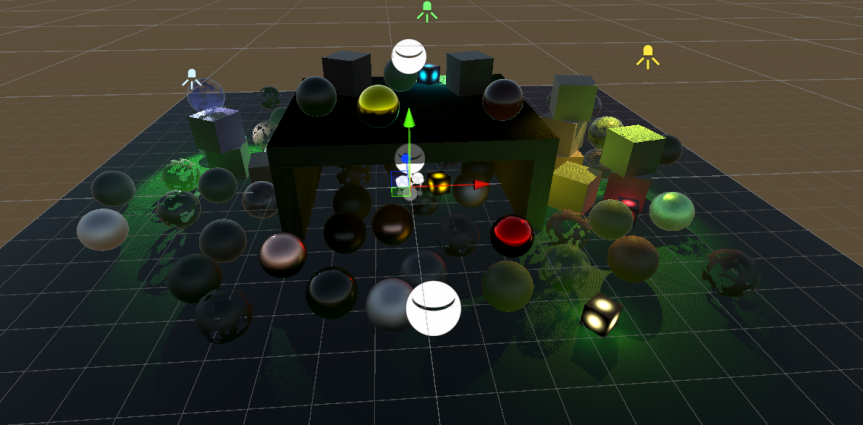
接着，对于实时阴影，只保留AtlasSize和Filter的配置，像方向光一样准备阴影贴图的代码（但是不需要cullingSphere和cascade，最多支持16个灯光）。

先绘制聚光灯的阴影贴图，调用比方向光更简单，具体见代码。



对于聚光灯，我们不需要Pancaking（把物体压到阴影贴图渲染时的近平面）。

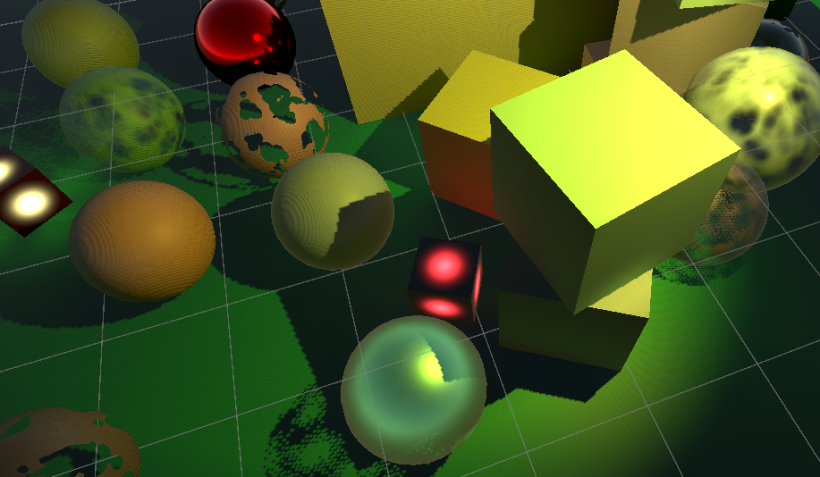
阴影贴图采样过程和方向光类似，区别在于它是透视投影的贴图，因此采样位置的xyz分量需要除以w分量归一化。



上面的结果还没有normalBias。

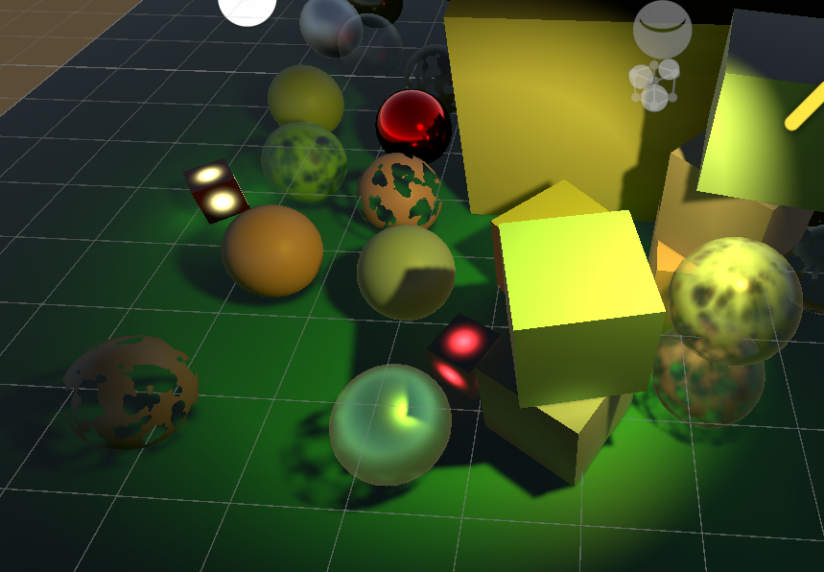
因为是透视投影，所以像素大小在世界空间是不确定的，但是会与到光平面的距离成正比。我们计算出距离为1的像素大小，再进行缩放。

该大小由spotAngle决定，计算并传给采样函数，测试结果：

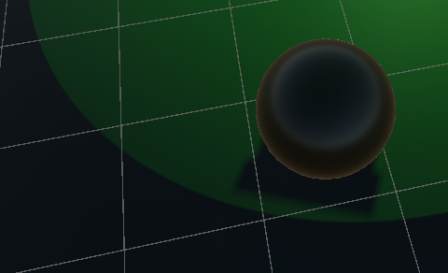


为了计算缩放倍数，需要知道世界空间的聚光灯位置和光源方向。

缩放后得到正确结果：

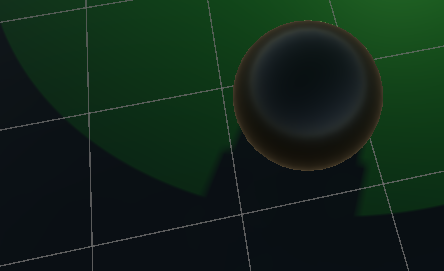


此时，因为bias和filter的存在，我们会采样到贴图外的区域，在边缘会出现问题：



因此要手动clamp采样点。

在存储bias的地方额外存储tile的边界。

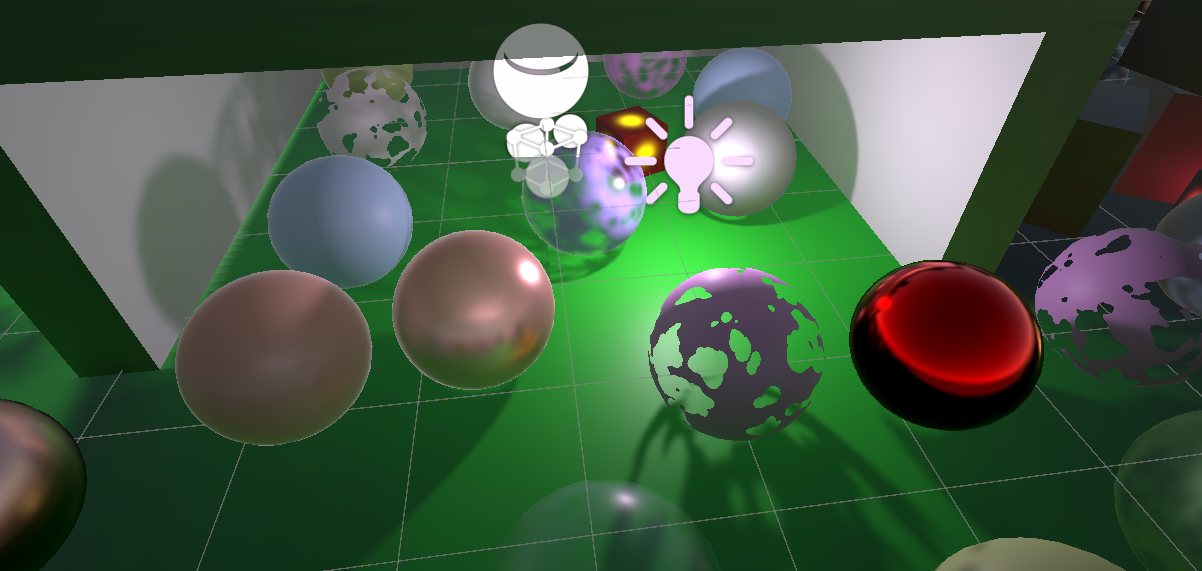


下面实现点光源的阴影。因为点光源的阴影贴图是一个CubeMap，所以需要占用六张贴图的空间。（总共16张贴图的空间）两个点光源的结果：

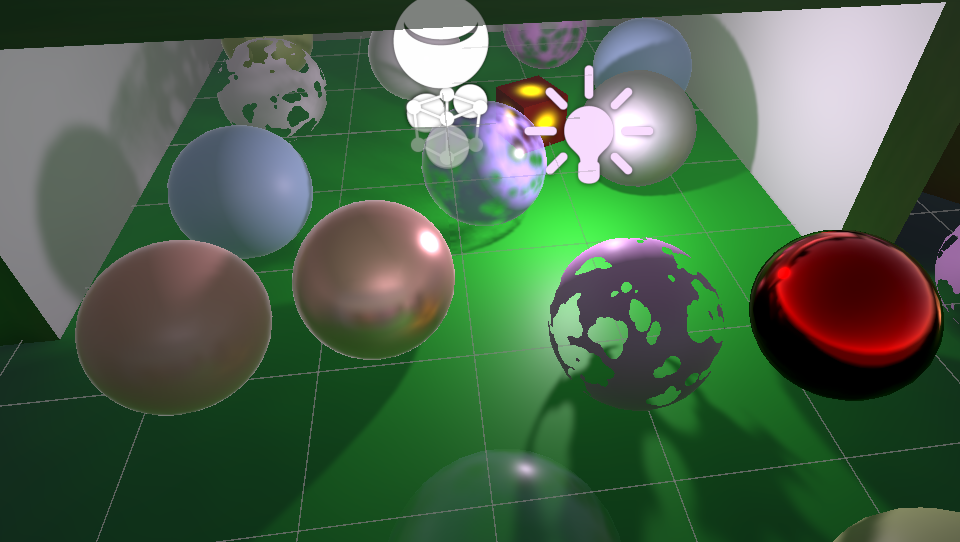


此外，因为立方体贴图的视角始终为90度，所以bias计算可以统一起来。

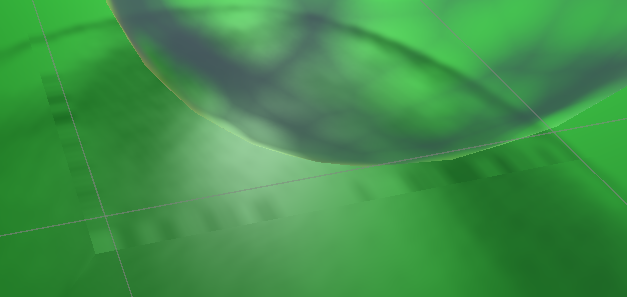
在采样时，通过CubeMapFaceID获取当前采样的贴图。并且需要更具对应的方向修改光平面的方向（后者用于计算bias）。



但是现在因为bias出现了漏光的现象，我们修改view Mat，修改了阴影贴图的绘制视角。因为Unity原本是绘制的背面，现在我们要绘制正面。

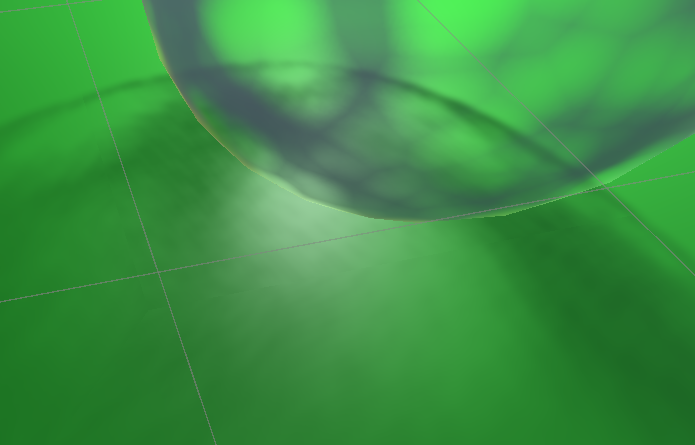


接下来，因为我们是直接采样cubeMap的，所以在过渡处会采样到超出边界的部分：

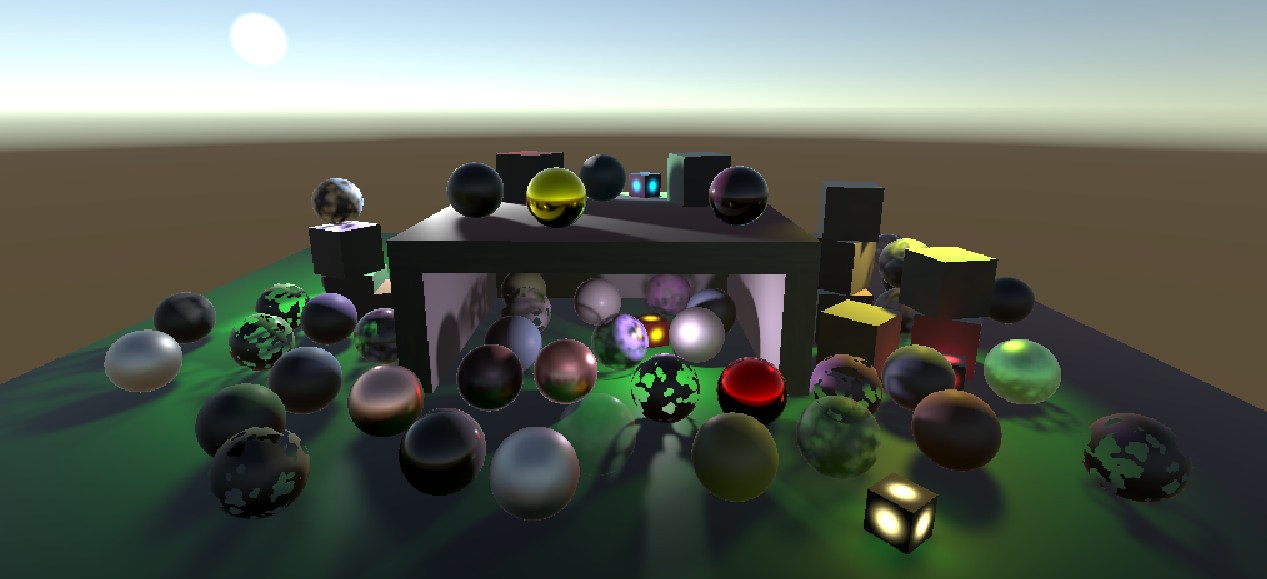


因此，增加绘制每张贴图的fov即可，对应接口传入fovbias。

尽管增加fov还会进一步修改像素大小，导致我们需要进一步调整，但是这里误差已经足够小了：



最终结果（全使用实时阴影）：



如果是Mixed模式，那么超出MaxShadowDiatance的使用的是shadowMask，而且会限制最大个数为4：

