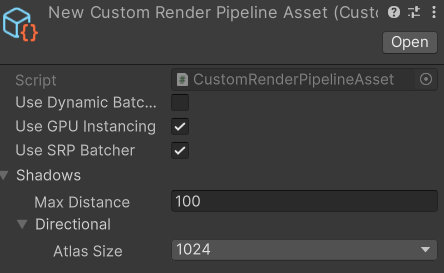
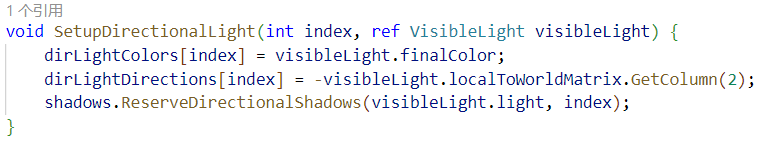
本部分包含方向光源的阴影实现，并支持了级联阴影方案。

使用RP Asset配置阴影参数，并在构造时传给RP。



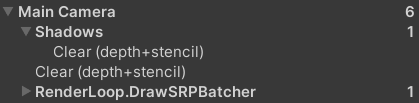
对应的参数进一步传给Renderer，先在剔除的时候使用maxShadowDistance，其次在管理光源的结构体中，创建管理阴影的结构体实例(Shadows)。

然后在设置平行光的时候，存储可以投射阴影的光源index。



接着，在设置好光源后，调用Shadows类的Render方法，在方法中，先申请阴影贴图，再将该帖图设置为RenderTarget。

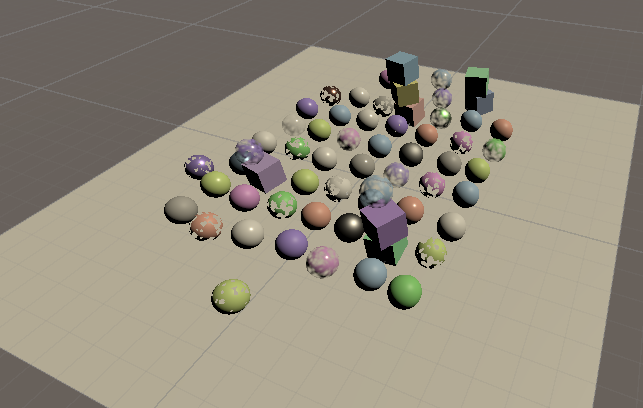
这时，我们需要调整Pipeline的顺序，把设置光照的部分与设置相机的部分对调，目的是在绘制正常的场景前，就完成阴影贴图的绘制工作。



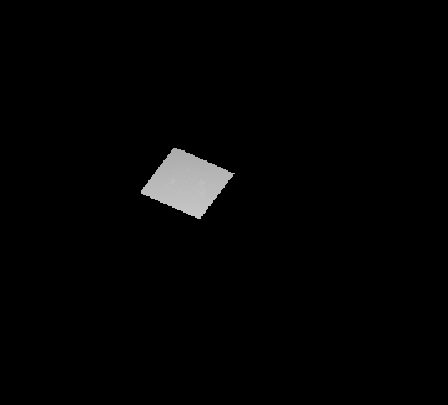
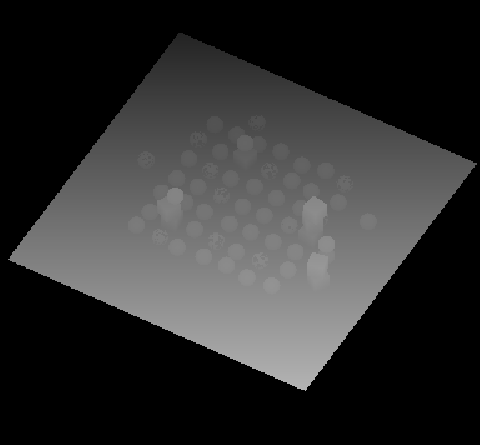
配置好shadowSettings（具体见代码），就可以通过Context调用DrawShadows了。

但是，该方法只会对包含ShadowCaster这个Pass的材质进行绘制，所以要为着色器添加对应Pass。并且，该Pass只会写入深度。对于Lit材质对应的ShadowCaster，只需要拷贝它着色部分的代码，并删除绘制颜色的部分即可。

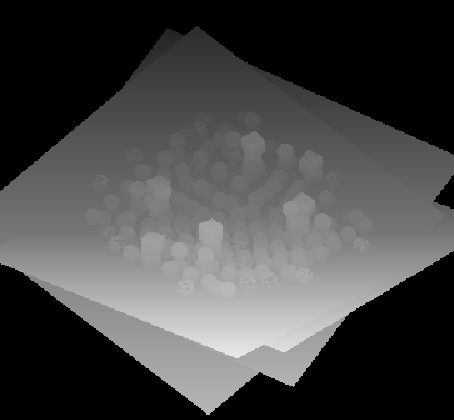
测试场景，一个方向光：



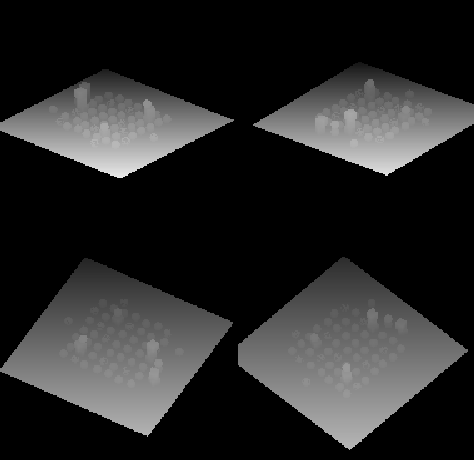
改变最大阴影距离，查看阴影贴图绘制结果：

但是当使用多个光源时，它们的阴影贴图会叠加到一起：



修改Viewport的设置，让它们不会重合：



为了能进行阴影深度的比较，将变换矩阵存储起来，并且需要对矩阵进行预处理，分别是：1.消除reverse\_z的影响; 2.将裁剪空间坐标变换为屏幕空间坐标（并且需要把深度范围也变到0-1）; 3.将viewport变换也考虑进来

以上是思路，下面是对于矩阵变换公式的具体解释：

<https://zhuanlan.zhihu.com/p/683661359>

我们需要让光源记录对应的Shadow数据，把光源的ShadowStrength和在Shadows里的index打包作为Vector数组上传。

对于前面创建的上传到GPU的数据和贴图，编写对应的hlsl文件来声明接口。

注意，阴影贴图的采样器设置为：sampler\_linear\_clamp\_compare。

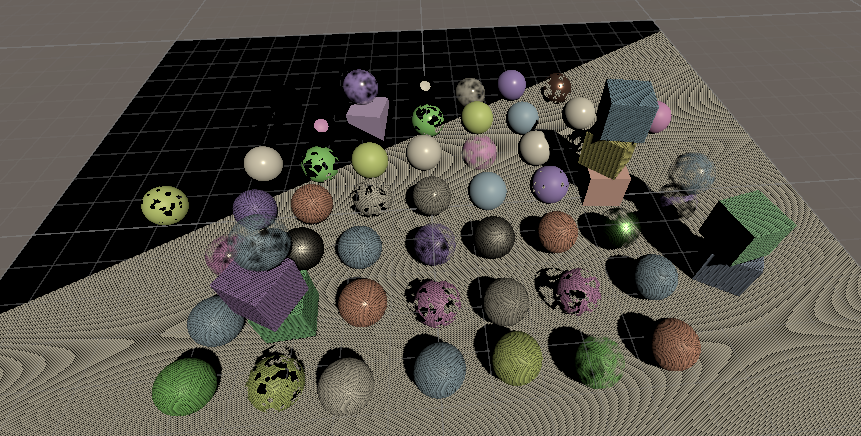
对阴影贴图采样时，通过SAMPLE\_TEXTURE2D\_SHADOW实现深度值的比较，得到结果。

在对阴影贴图进行采样时，结果会根据阴影强度进行衰减：



我们把这个结果记录在光源的结构体里（attenuation）。

会得到范围外有问题，且存在自遮挡的阴影结果，我们之后处理它们。



两个光源的结果：

