1 总览

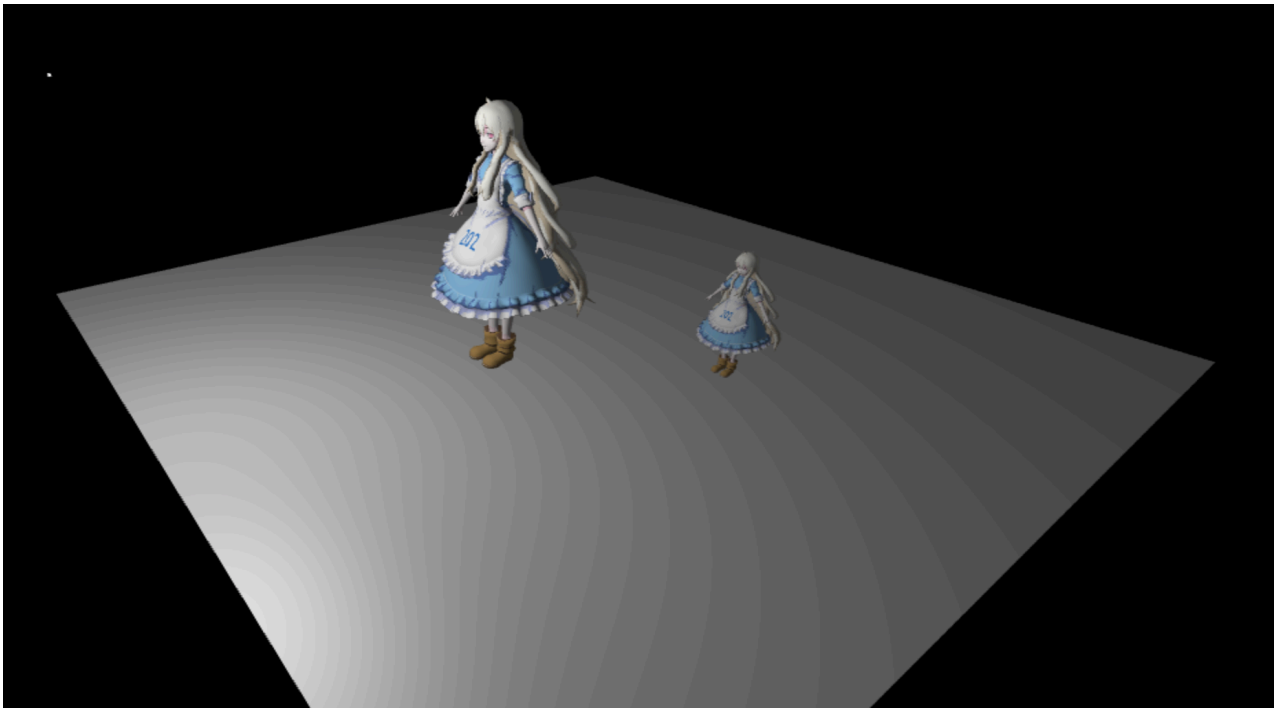
正确的阴影对于渲染质量的提升起到着举足轻重的作用。本次作业的内容为实时阴影,对于硬阴影要求实现经典的 Two Pass Shadow Map 方法,对于软阴影则要求实现 PCF(Percentage Closer Filter) 和 PCSS(Percentage Closer Soft Shadow)

本轮工作的主要内容将集中在 `phongFragment.glsl` 内函数的完善上。

- 对于场景中的每个物体都会默认附带一个 `ShadowMaterial` 材质，并会在调用 `loadOBJ` 时添加到 `WebGLRenderer` 的 `shadowMeshes[]` 中。当光源参数 `hasShadowMap` 为 true 时，作业框架将开启 Shadow Map 功能，在正式渲染场景之前会以 `ShadowMaterial` 材质先渲染一遍 `shadowMeshes[]` 中的物体，从而生成我们需要的 ShadowMap。
- `ShadowMaterial` 将使用 `Shader/shadowShader` 文件夹下的顶点和片元着色器。实现该材质的重点是要向 `ShadowVertex.glsl` 传递正确的 `uLightMVP` 变量。
- 在正确实现 Shadow Map 之后，实现 PCF、PCSS 的主要工作将集中在 glsl 的编写上，详见开发说明中的相应小节。

2 开发说明

初始的代码框架是没有阴影的 Blinn-Phong 场景。



$$L_o(p,\omega_o) \approx \frac{\int_{\Omega^+} V(p,\omega_i)d\omega_i}{\int_{\Omega^+} d\omega_i} \cdot \int_{\Omega^+} L_i(p,\omega_i) f_r(p,\omega_i,\omega_o)\cos\theta_i d\omega_i$$

如上式所示，后半部分的积分是我们作业 0 中完成的 Blinn-Phong 模型，前半部分是我们要实现的阴影方法。这里我们提供了 `blinnPhong()` 函数，在实现了阴影算法，得到可见性信息后，将两部分相乘得到最终的着色结果。

我们推荐按照硬阴影，PCF，PCSS 的顺序来完成。

2.1 Shadow Map

对于 ShadowMap 的实现有两个任务点需要完成：

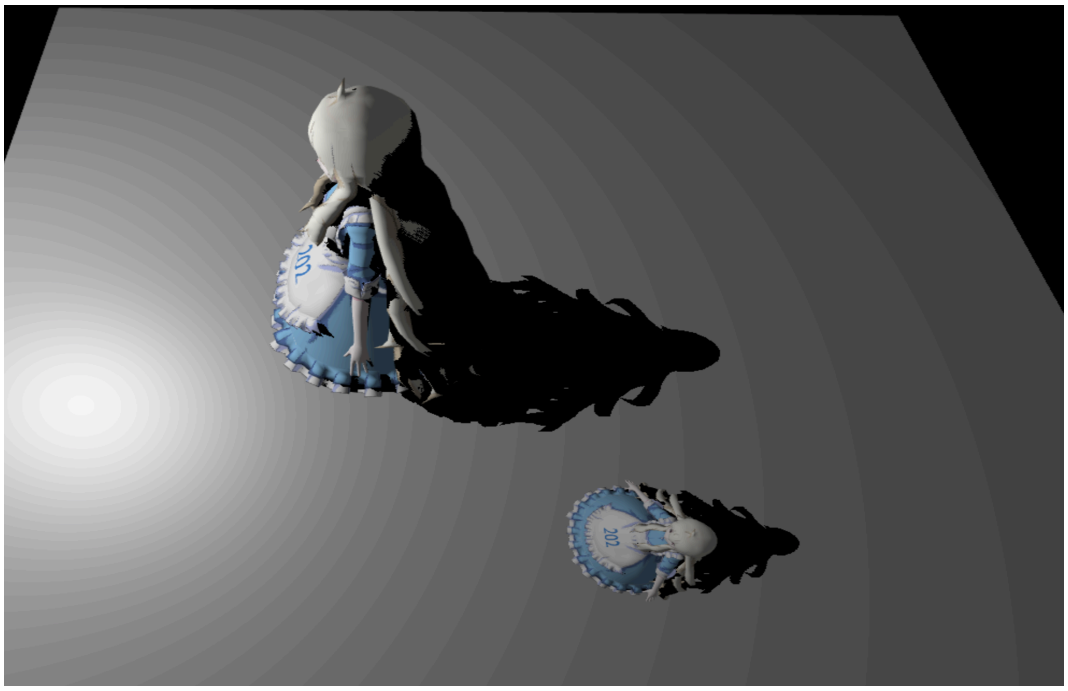
1. 在 `ShadowMaterial.js` 中需要向 `Shader` 传递正确的 `uLightMVP` 矩阵，该矩阵参与了第一步从光源处渲染场景从而构造 ShadowMap 的过程。你需要完成 `DirectionalLight` 中的 `CalcLightMVP(translate, scale)` 函数，它会在 `ShadowMaterial` 中被调用，并将返回光源处的 MVP 矩阵绑定从而完成参数传递过程。

   这里我们相当于在光源处建立一个虚拟的摄像机：

- 你需要使用 `lightPos`, `focalPoint`, `lightUp` 来构造摄像机的 LookAt 矩阵。
- 推荐在使用正交投影，这可以保证场景深度信息在坐标系转换中保持线性从而便于之后使用。正交投影的参数决定了 shadow map 所覆盖的范围。

2. 需要完善 `phongFragment.glsl` 中的 `useShadowMap(sampler2D shadowMap, vec4 shadowCoord)` 函数。该函数负责查询当前着色点在 ShadowMap 上记录的深度值，并与转换到 light space 的深度值比较后返回 visibility 项（请注意，使用的查询坐标需要先转换到 NDC 标准空间 [0,1]）。

   如果一切顺利，我们的模型会带有合理的硬阴影，效果如下图所示：



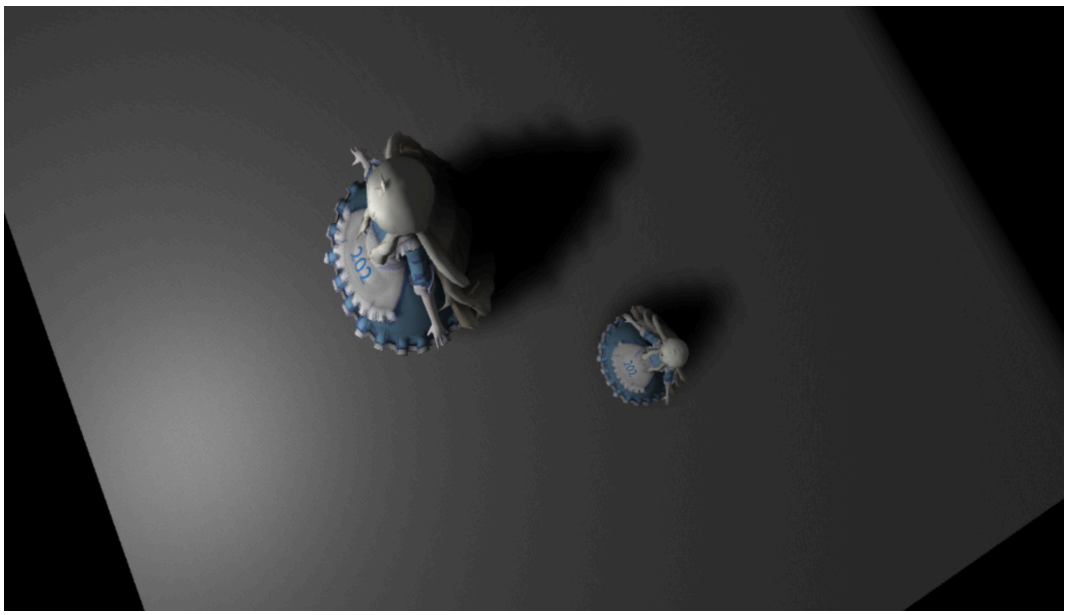
2.2 PCF(Percentage Closer Filter)

需要完善 `phongFragment.glsl` 中的 `PCF(sampler2D shadowMap, vec4 shadowCoord, float filterSize)` 函数。我们推荐在一个圆盘滤波核中进行随机采样，采用这种方案的原因是可以简化后续 PCSS Shader 的编写同时可以使软

阴影上模糊的部分更显圆润自然，计算出来的伴影直径可与单位圆盘上的采样点相乘生成 ShadowMap 上的采样坐标（值得注意的是随机采样函数的质量将与最终渲染效果的好坏息息相关，我们在框架中提供了泊松圆盘采样和均匀圆盘采样两种采样函数，替换使用对比一下两种采样函数的细微区别，我们也鼓励使用其他的采样方法）。

假如你对随机采样函数的了解暂时没有那么多，我们提供了框架中采样函数代码的可视化展示——`https://codepen.io/arkhamwjz/pen/MWbqJNG?editors=1010`，应该可以帮助你更快的对随机采样函数有一个比较形象的认识。

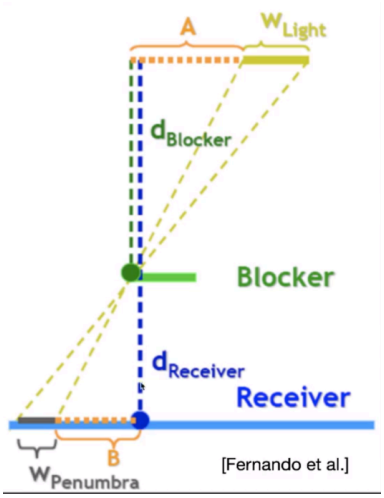
如果一切顺利，得到的阴影在边缘处会有模糊，之前 ShadowMap 阴影中边缘的锯齿会得到大大的改善，效果如下图所示：



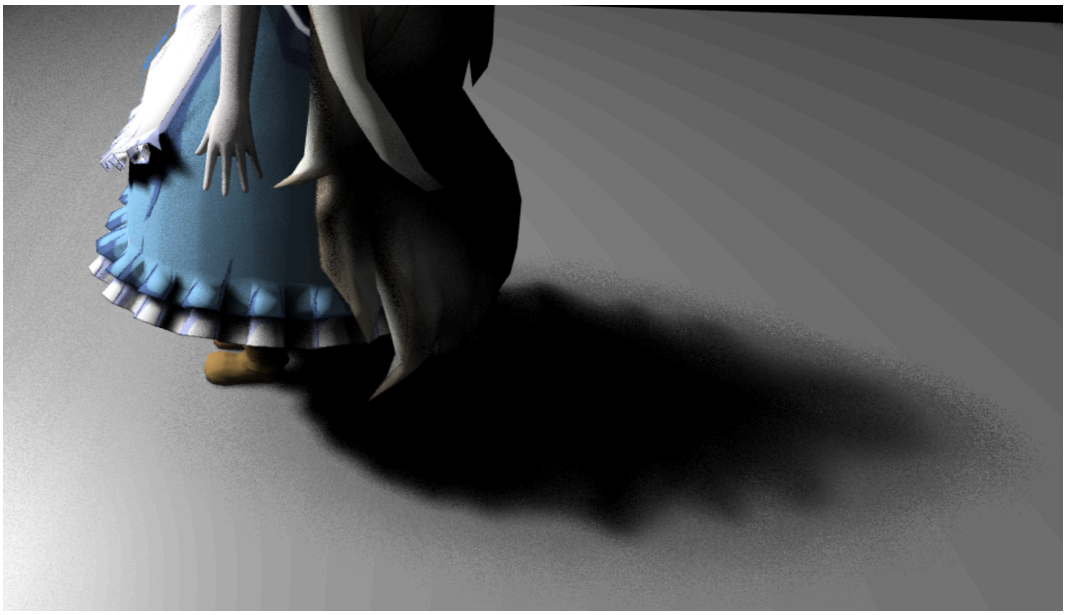
2.3 PCSS(Percentage Closer Soft Shadow)

需要完善 `phongFragment.glsl` 中的 `findBlocker(sampler2D shadowMap, vec2 uv, float zReceiver)` 和 `PCSS(sampler2D shadowMap, vec4 shadowCoord)` 函数。`findBlocker` 函数中需要完成对遮挡物平均深度的计算。接着就可以在 `PCSS` 函数中调用 `findBlocker`，利用相似三角形计算伴影直径并传递给 `PCF` 函数以调整其滤波核大小。（请注意，为了使本轮工作的重心放在算法的核心实现

上，诸如光源宽度、采样数之类的参数可以通过 `#define` 直接定义使用。同时请保证使用的数据具有统一标架，而不是简单的套用公式，要保证使用公式的具有物理意义否则将酌情扣分)。



如果一切顺利，得到的阴影在和遮挡物距离近的地方偏向于硬阴影，在和遮挡物距离远的地方会偏向于更为模糊的软阴影，效果如下图所示：



2.4 提示

1. 可以在 camera pass 时将 shadow map 的范围以及深度值可视化出来。

2. 可以在 PCSS 中将计算得到的遮挡物的平均深度以及软阴影范围可视化出来。

3 提交

提交时需要删除 /lib, /assets 文件夹，并在建立的 /images 文件夹中保存运行截图。截图需要展示硬阴影、PCF、PCSS 的阴影效果，分别以 SM_xxx.png、PCF_xxx.png、PCSS_xxx.png 格式保存，xxx 的内容可自行决定。