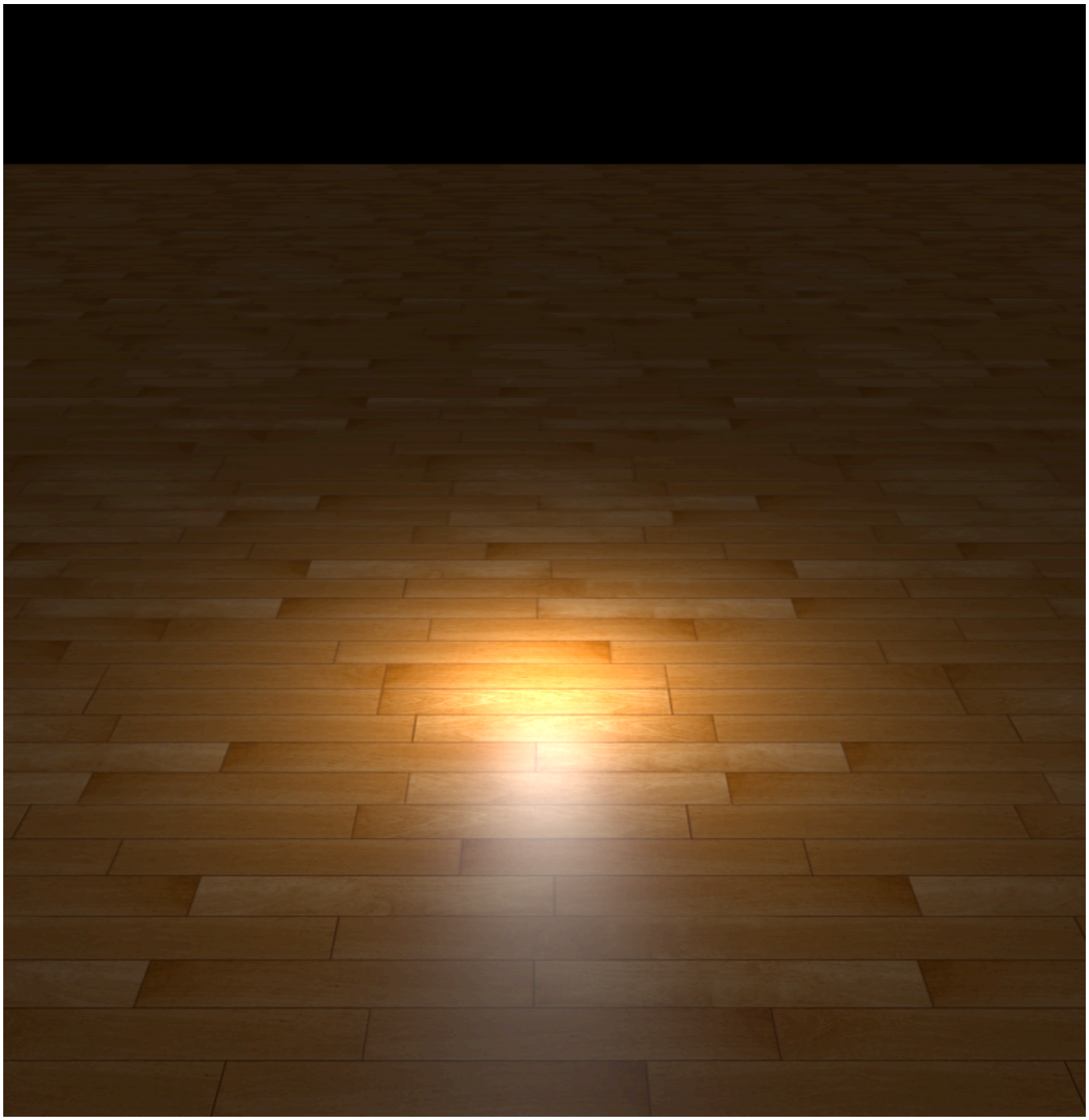
Task1

根据 `phong` 光照和 `blinn phong` 光照的公式即可完成渲染。

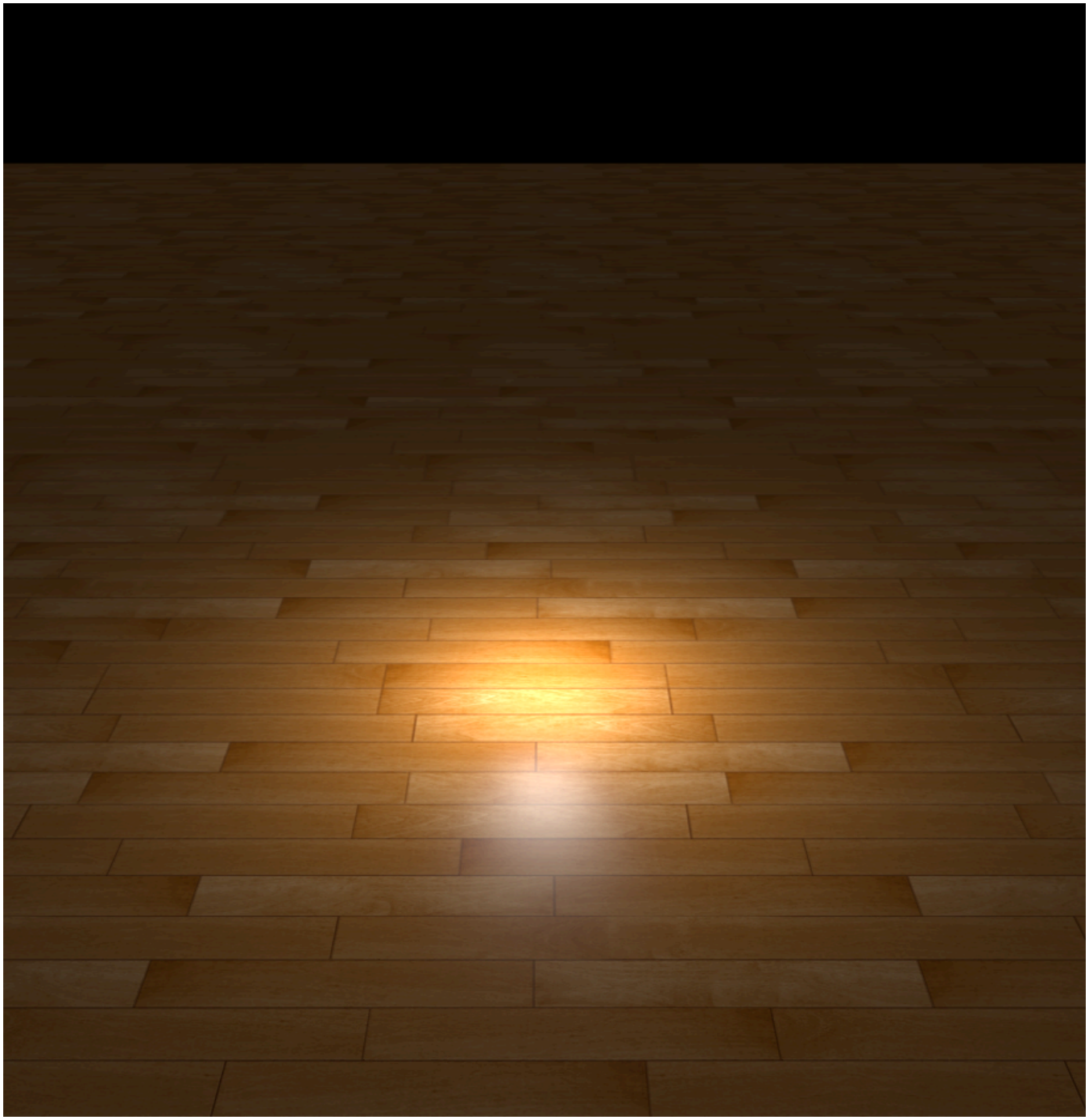
`phong` 为 $k_d(I_a + I_d \max(0, n\dot{l})) + k_s I_d \max(0, dir \cdot r)^p$ 其中 r 为反射光线， dir 为视角方向。

`blinn phong` 为 $k_d(I_a + I_d \max(0, n\dot{l})) + k_s I_d \max(0, n \cdot h)^p$ 其中 h 为半程向量。

blinn phong 效果



phong 效果



1. 顶点着色器和片段着色器的关系是什么样的？顶点着色器中的输出变量是如何传递到片段着色器当中的？

顶点着色器将顶点属性处理之后传递给片段着色器，将顶点着色器的输出作为片段着色器的输入。顶点着色器处理顶点后将数据传递给片段着色器通过插值来渲染整个图形。

再顶点着色器声明一个与片段着色器 `in` 属性的变量名字相同的 `out` 变量，再链接的时候就能

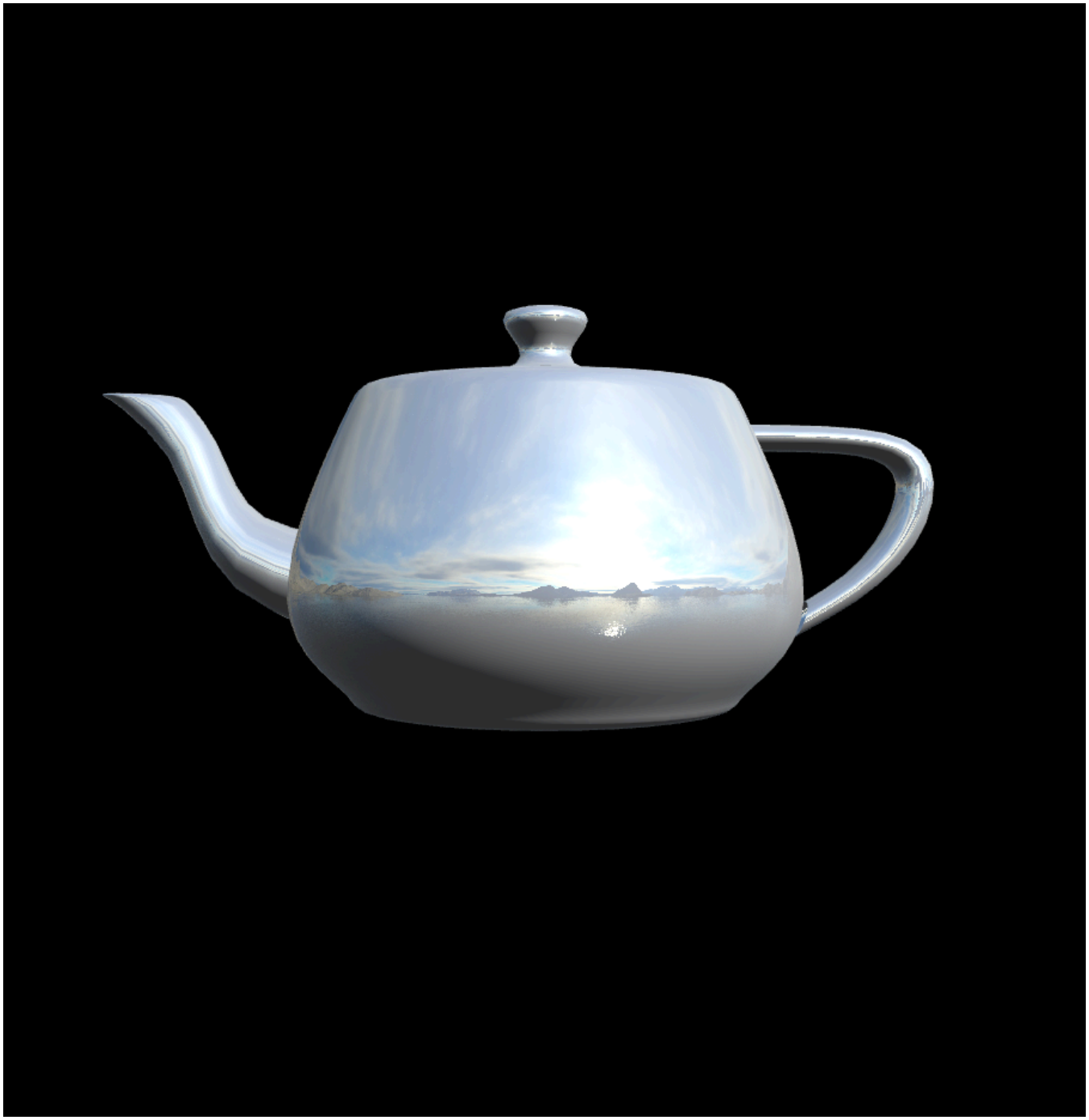
进行传递。

2. 代码中的 `if (diffuseFactor.a < .2) discard;` 这行语句，作用是什么？为什么不能用 `if (diffuseFactor.a == 0.) discard;` 代替？

作用是当这个片段的 `alpha` 值 < 0.2 就不进行绘制，因为 `alpha` < 0.2 就几乎透明了，不然很透明的地方会被渲染得很暗。

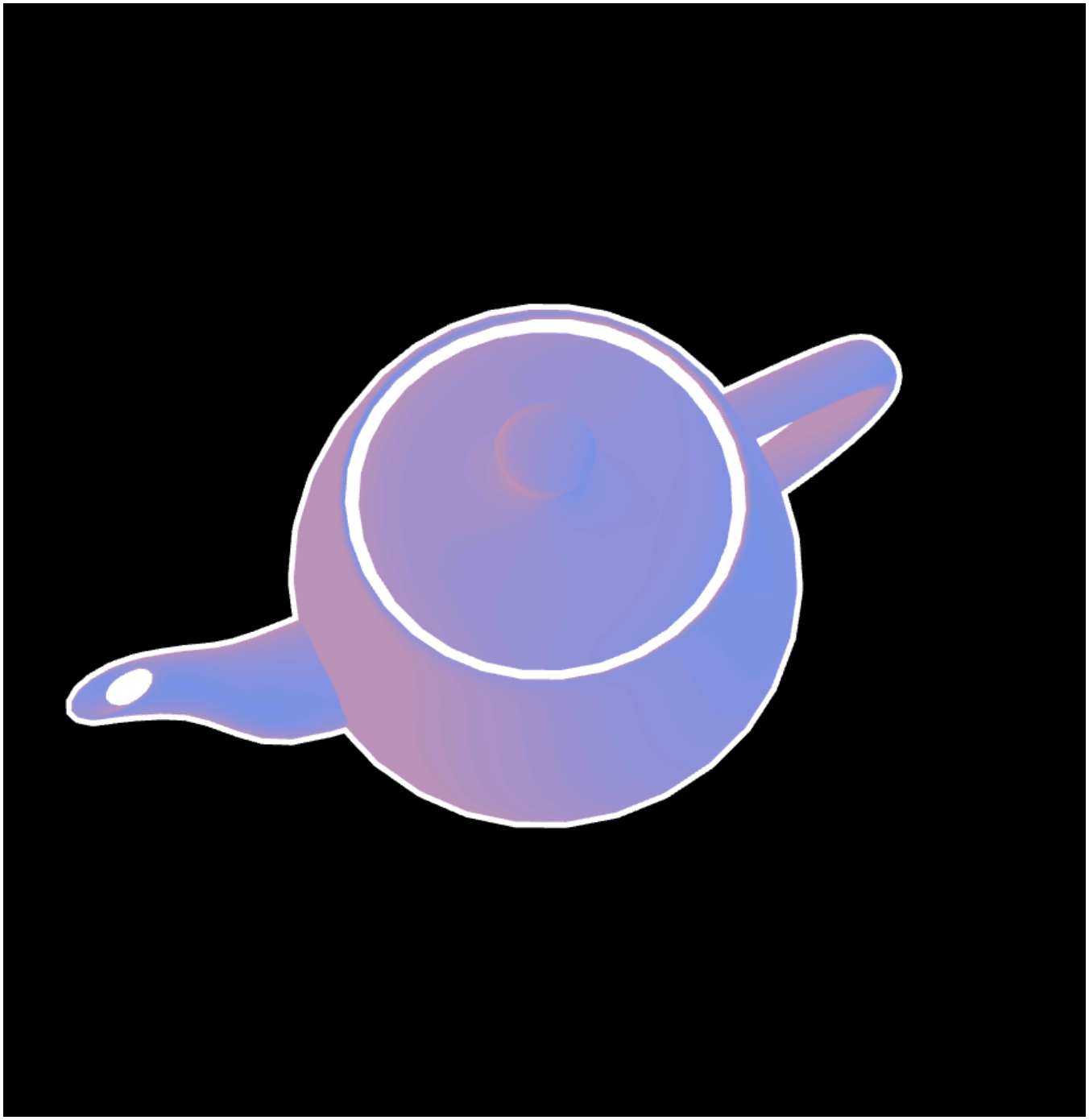
Task2

通过计算反射光，根据反射光线确定纹理位置即可，最后取出 `texture` 。



Task3

直接运用冷暖色插值公式即可！

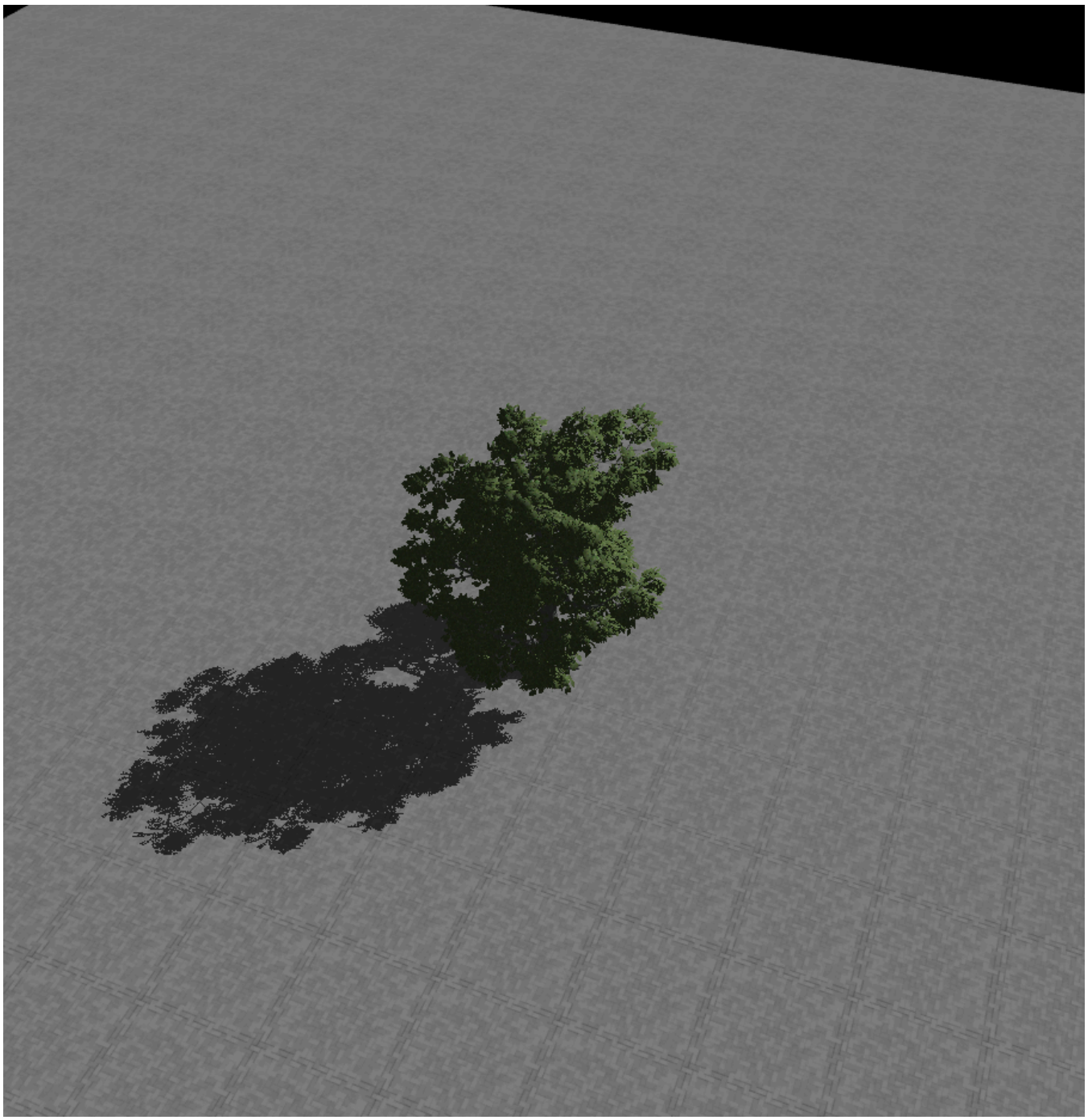


1.使用 glCullFace 函数，可以剔除几何体的正面或背面。
正面剔除（GL_FRONT）：渲染背面。
背面剔除（GL_BACK）：渲染正面。

2.线的宽度难以控制。

Task4

通过投影坐标，从深度贴图中找到 $[0,1]$ 的结果，通过坐标变换，我们可以找到最近的点，就能知道这个点是不是在阴影中了。



1.有向光源:透视投影矩阵

平行光源:正交矩阵

2.因为坐标已经到了 $[-1,1]$ ，通过 `pos=pos * 0.5 + 0.5` 可以算出对应深度。

Task5

计算出直线三角形相交的点，然后可以判断这个点的颜色，以及是反射还是折射，求出这个光的最终状态。