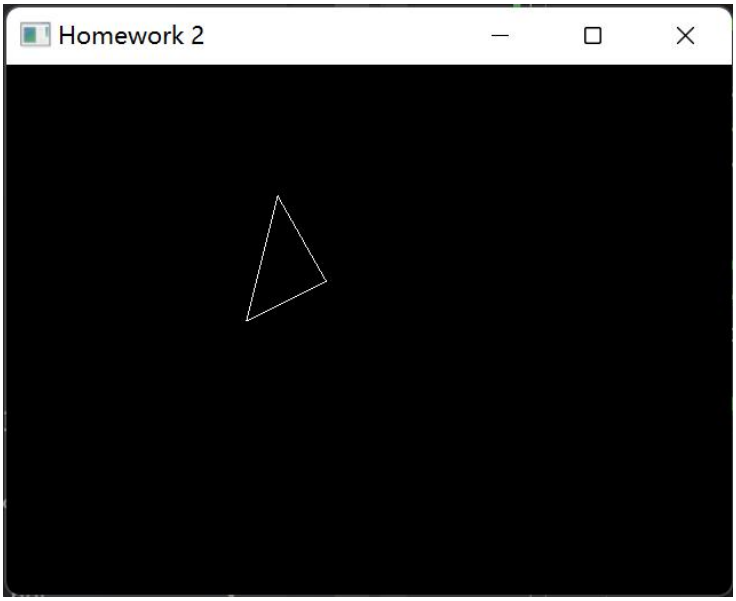
计算机图形学 作业二

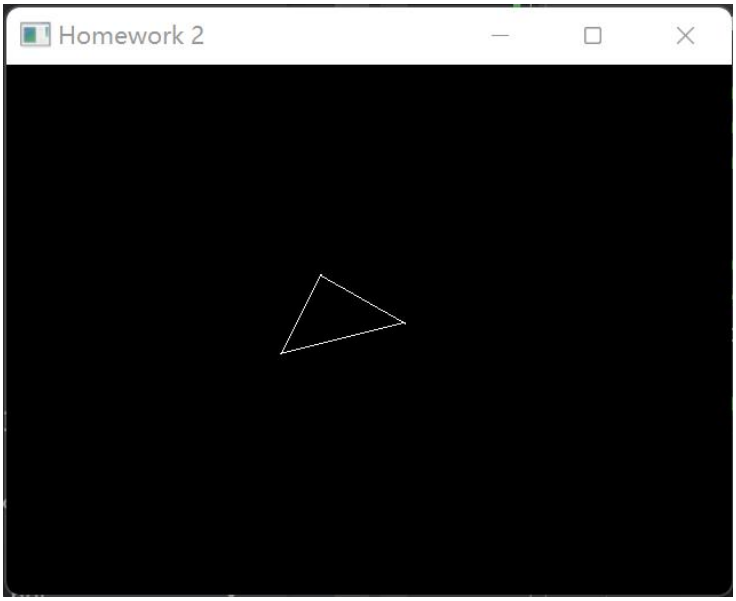
1. 实现三角形的光栅化算法

实验要求：

1.1：用 DDA 实现三角形边的绘制

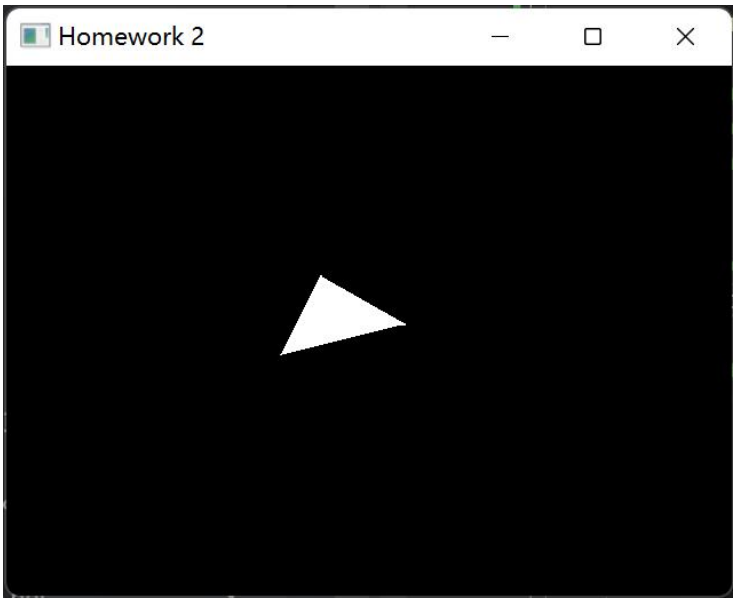


1.2：用 bresenham 实现三角形边的绘制



↓

1.3：用 edge-walking 填充三角形内部颜色



讨论要求：

1.4：针对不同面数的模型，从实际运行时间角度讨论 DDA、bresenham 的绘制效率。

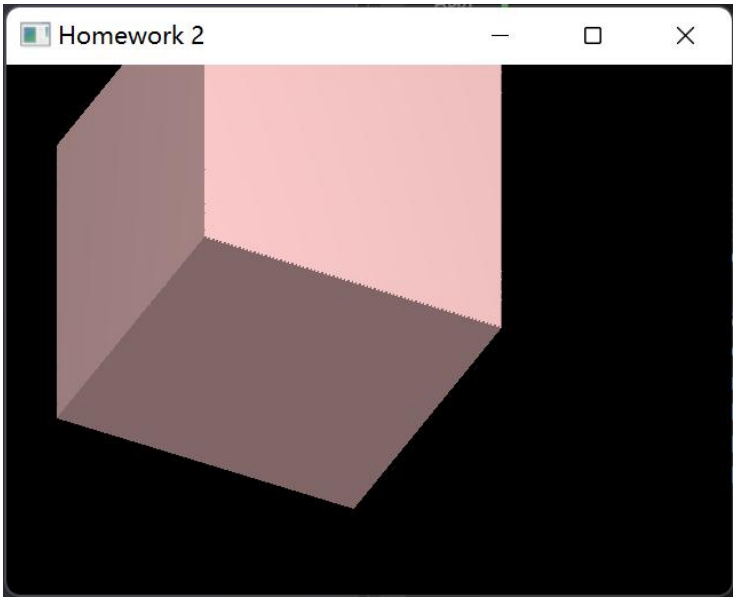
	DDA	bresenham
teapot_600	5014ms	4789ms
rock	2326ms	2221ms
cube	158ms	147ms

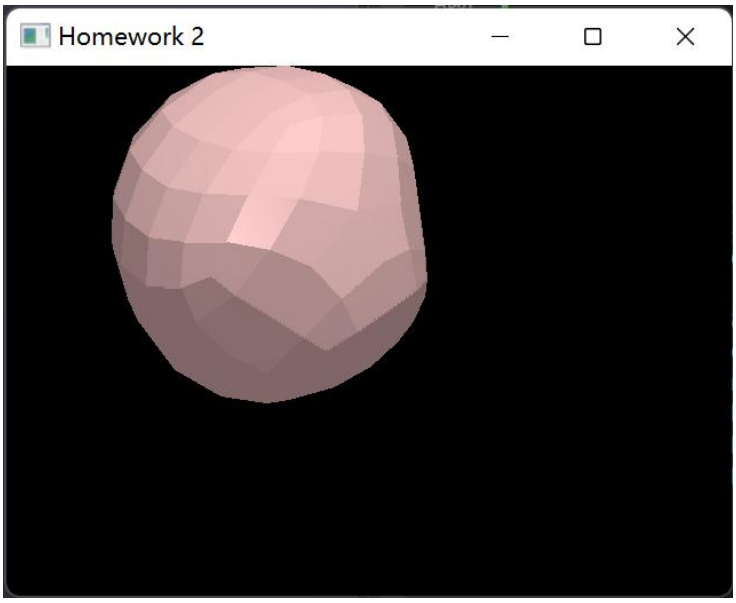
可以看到 Bresenham 的效率比 DDA 更高，从理论上来说，Bresenham 算法的时间复杂度相对较低，因为它使用整数运算和避免了浮点运算，不过由于模型中还有大量的面内填色等其他工作，且模型的三角形数量并不是非常多，所以 Bresenham 算法带来的时间改善并不显著。

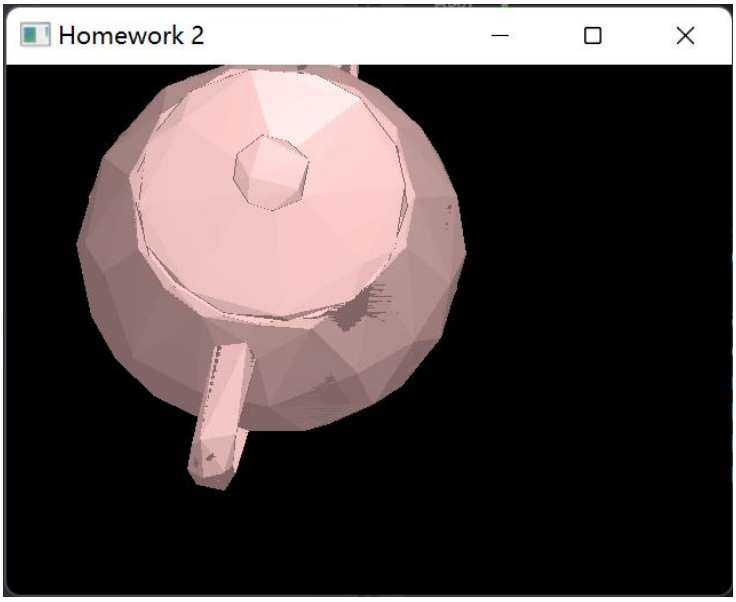
2. 实现光照、着色

实验要求：

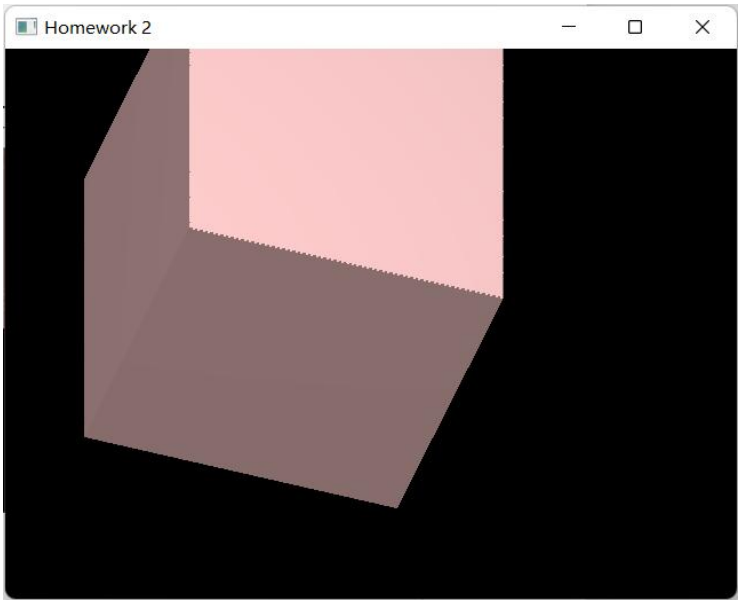
2.1：用 Gouraud 实现三角形内部的着色

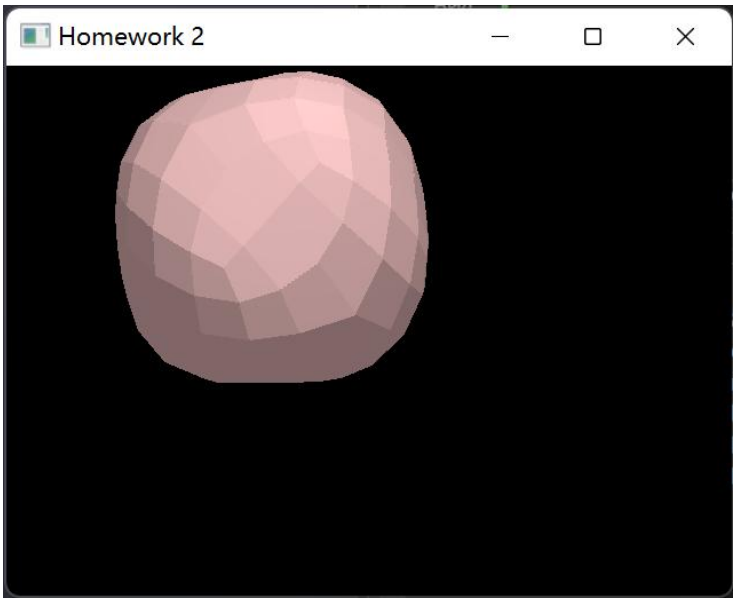


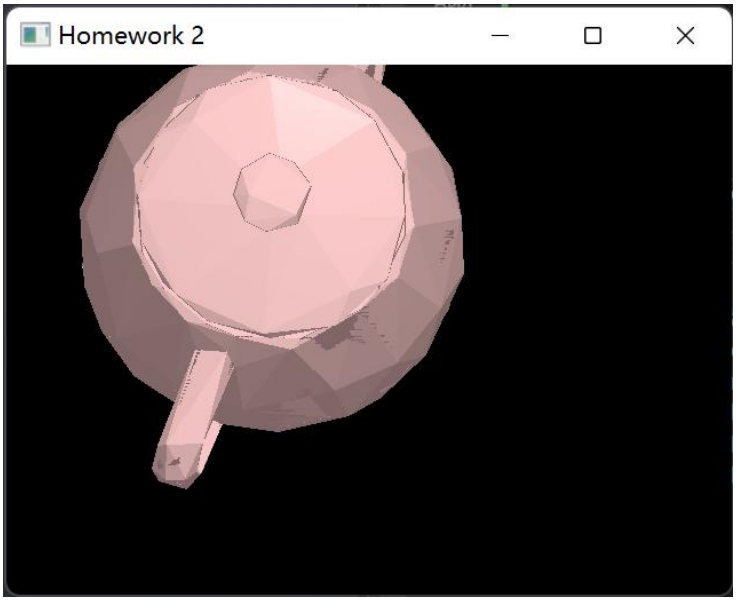




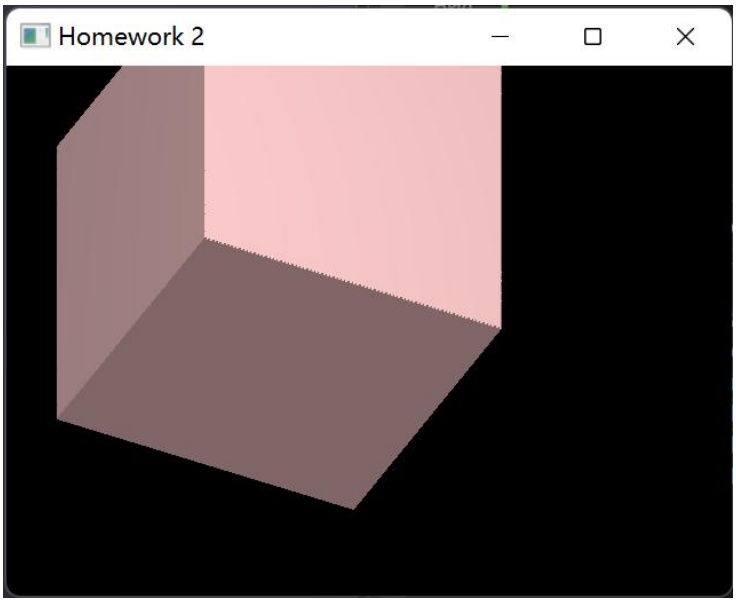
2.2：用 Phong 模型实现三角形内部的着色

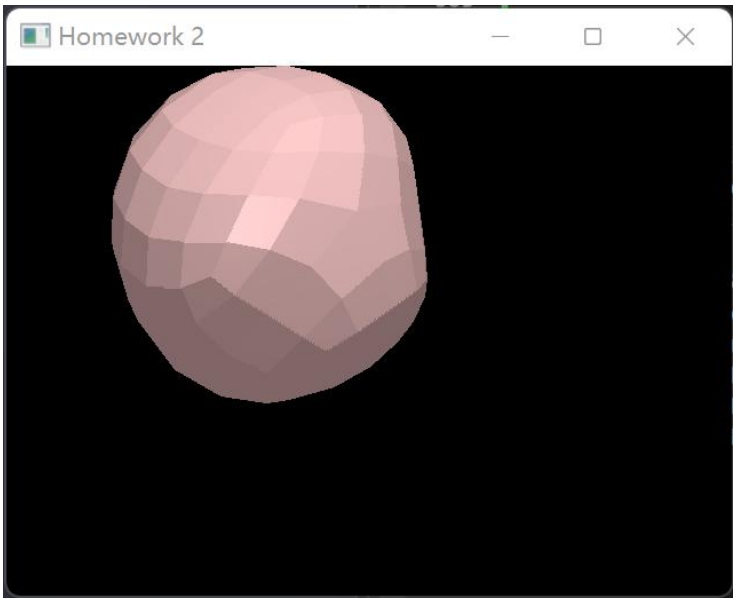


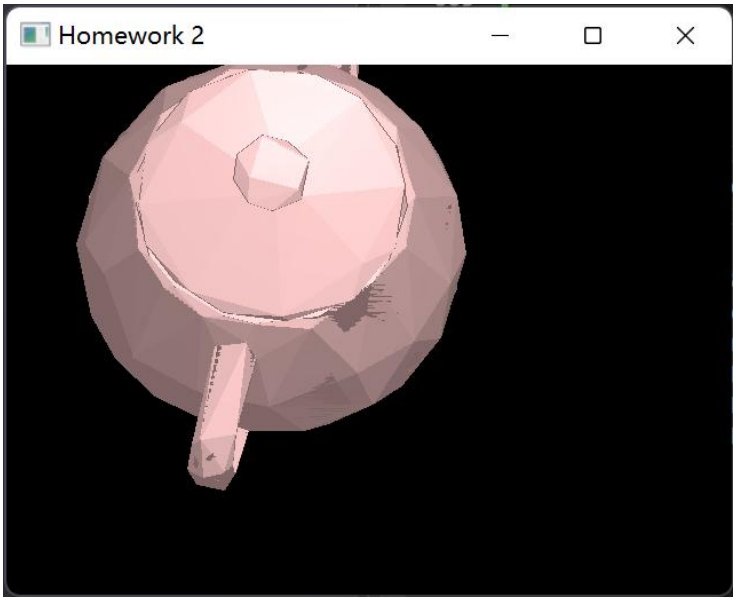




2.3：用 Blinn-Phong 实现三角形内部的着色



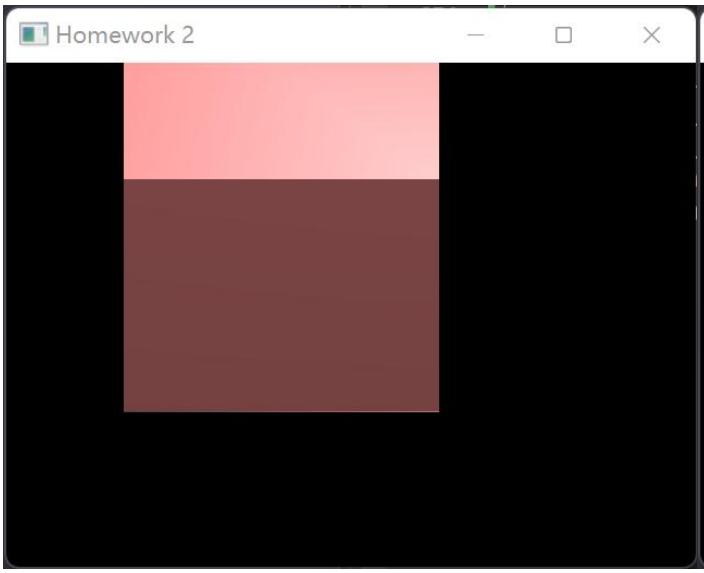

Homework 2


Homework 2

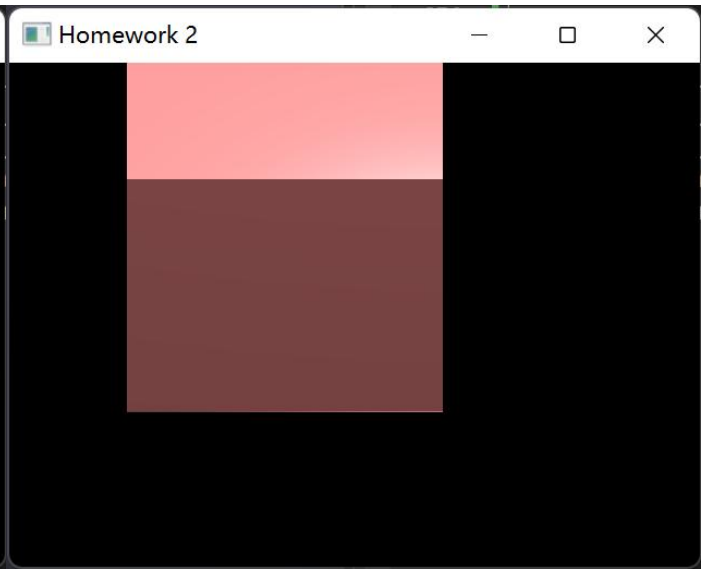
↓
↓

讨论要求：
2.4：结合实际运行时间讨论三种不同着色方法的效果、着色效率。

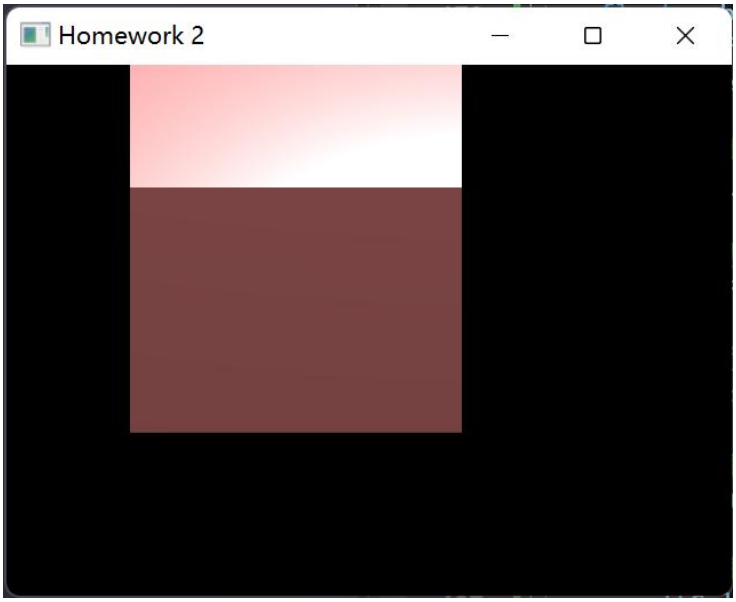
Gouraud 算法：



Phong 算法：

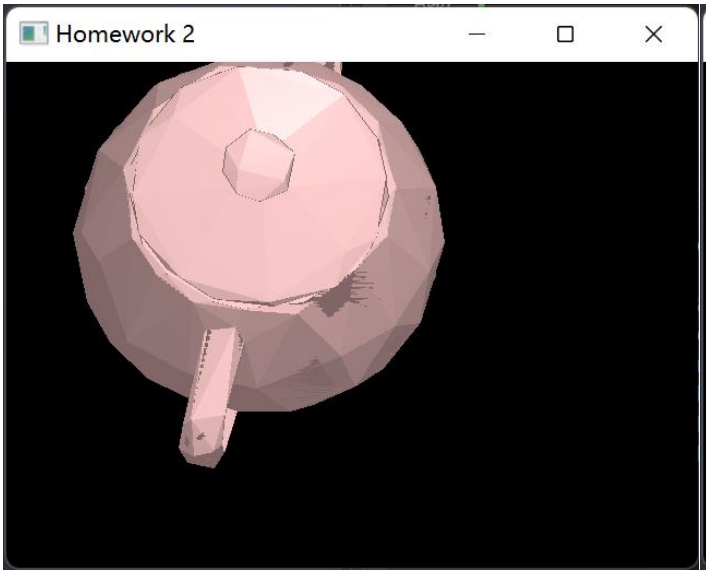


Blinn-Phong 算法：

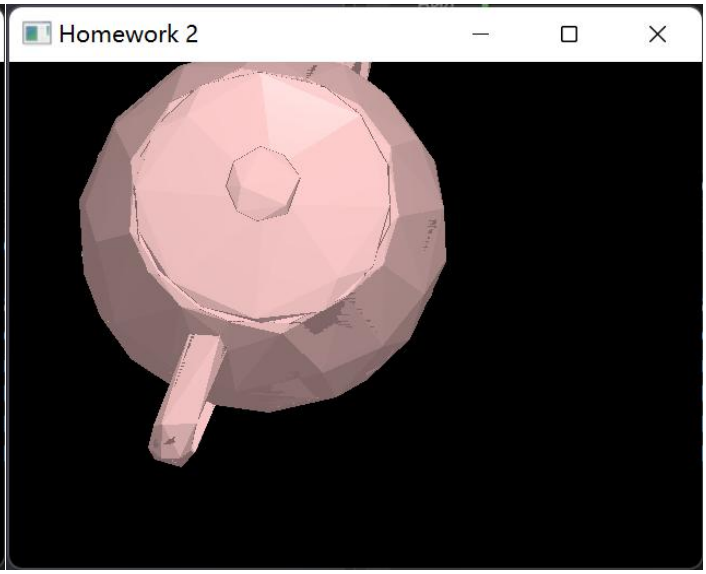


上面三幅图依次是 Gouraud 算法、Phong 算法、Blinn-Phong 算法。
可以从立方体的上方面看出，Blinn-Phong 反射模型的高光最强，这是由于 Blinn-Phong 算法使用了半程向量来近似镜面高光的计算，Blinn-Phong 模型中限制了高光的范围，确保只有在法线和半程向量之间的夹角较小时才有明显的高光。这种设计使得 Blinn-Phong 高光相对较强，尤其在较小的夹角范围内，产生了明显的高光效果。
Gouraud 算法着色的效果相对较平滑，在远离光源时没有很好的将颜色变淡，这是因为 Gouraud 是在顶点着色阶段进行光照计算的，然后通过插值在三角形内部得到每个像素的颜色，所以其光线的减弱只会以线性降低。
Phong 算法在这个图中效果仍然相对较平滑，但是能比较明显的看到上面右下角是有光照的，与 Blinn-Phong 算法的高光位置对应，而 Gouraud 算法看起来似乎灯光在立方体的正上方，Phong 算法能够在不过度曝光的情况下较好的展示光线信息。

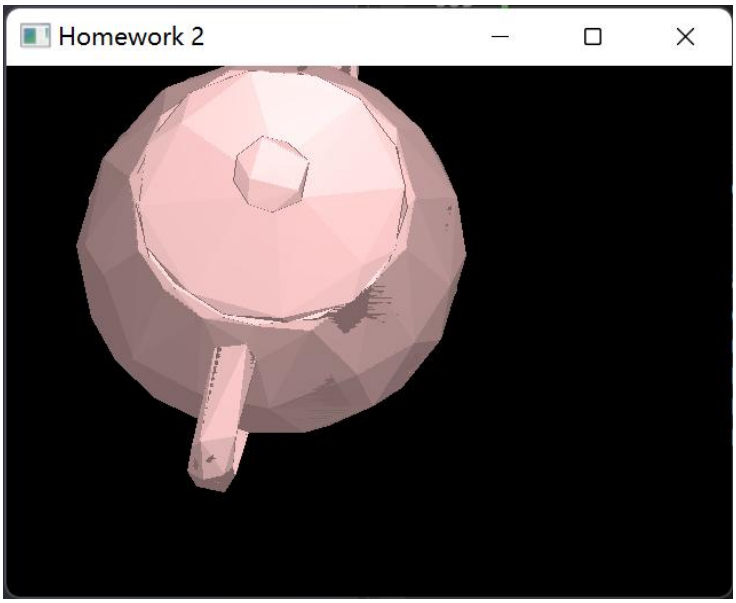
Gouraud 算法：



Phong 算法：



Blinn-Phong 算法：



在这三幅图中也能比较好的看清楚，Blinn-Phong 算法壶顶高光比较强，Gouraud 算法茶壶盖子相邻面之间的光线差异过大，而 Phong 算法能够在其中效果会较好一些。

着色耗时：

	Gouraud	Phong	Blinn-Phong
teapot_600	5658ms	5976ms	5861ms
rock	2645ms3	3153ms	2871ms
cube	131ms	235ms	193ms

可以看到 Gouraud 用时最少，Phong 用时最多，Blinn-Phong 位之中间。这与理论是相符的，Phong 每个像素都要进行较为复杂的光照计算，性能开销较大。Blinn-Phong 通过使用半程向量来简化计算，减少了计算量。Gouraud 在整个三角形内部是线性插值的，计算量最小。不过 cube 模型这个差异比较显著，teapot_600 这个用时差异反而没那么显著，经过分析发现主要原因可能是因为本人电脑配置较低，导致绘制 teapot_600 三种方法都是满负载运行，性能瓶颈掩盖了算法瓶颈。