1 光照与着色

1.1 材质与着色

我们已经学习了如何在计算机中储存和表示三维物体. 然而, 现实中的物体除几何属性外, 还存在**材质 (Material)** 属性. 材质决定了物体表面对光的反射和吸收特性, 从而影响我们最终看到的物体的颜色, 纹理等视觉信息. 在计算机图形学中, 为几何体的表面加上材质的过程称为**着色 (Shading)**.

定义 1.1 材质与着色
材质 (Material) 是物体表面的光学属性, 决定物体如何与光相互作用, 也即决定物体呈现的视觉效果.
着色 (Shading) 是绘制几何体表面的颜色使得它的视觉效果符合设定材质的过程.

1.2 光照

1.2.1 光源

按照光源发出光线的不同, 光源可以分为以下几类.

(1) 点光源 (Point Light Source): 点光源在空间中可以看作一个无大小的点, 从该点向各个方向均匀发射光线. 点光源的光强度 I 只与距离有关, 并且正比于 $1/r^2$, 而与方向无关.

(2) 平行光源 (Directional Light Source): 平行光源发出的光线在空间中是平行的. 平行光源的光强度 I 与距离无关, 但与接受平面与光线方向的夹角 θ 有关, 即 $I_{\text{eff}} = I\cos\theta$.

(3) 环境光 (Ambient Light Source): 环境光源表示来自环境中各个方向的漫反射光线的总和, 可以认为在空间中均匀分布, 因此其光强度 I 与距离和方向均无关.

1.2.2 反射

与研究光照类似, 我们先考虑简单情况下的反射规律.

漫反射 当物体表面粗糙不平时, 入射到表面的光线会被反射到各个方向, 称为**漫反射 (Diffuse Reflection)**.

定义 1.2 漫反射与朗博体
漫反射 (Diffuse Reflection) 是指入射光线被物体表面反射到各个方向的现象.
朗博体 (Lambertian Surface) 是表面只有完全随机的漫反射的物体.

对于朗博体的漫反射而言, 我们有如下简单的公式以计算其反射的光强:

$$L_d = k_d\left(I_a + I_d\max(\cos\theta, 0) + \frac{I_p}{r^2}\cos\theta\right)$$

镜面反射 当物体表面光滑时, 入射到表面的光线会按照一定的规律被反射, 称为**镜面反射 (Specular Reflection)**.

定义 1.3 镜面反射 **镜面反射 (Specular Reflection)** 是指入射光线按照反射定律被反射的现象.

一般而言, 由于物体表面并不能完全光滑, 镜面反射并不总是完全符合反射定律, 而是分布于理想反射光线 $\boldsymbol{r}$ 附近.

1.3 渲染的光栅化

与二维图形一样, 我们可以通过**光栅化**将虚拟的几何与材质转化为像素. 这主要包含两个问题: 确定像素对应的三维位置 (也即正确处理遮挡关系), 以及确定该像素的颜色.

1.3.1 深度缓存

画家算法 为了确定物体的前后关系, 最初想到的办法是给每个形状/物体一个额外的深度属性 z, 然后对所有形状的深度排序, 按照 z 从大到小 (即从远到近) 的顺序进行绘制. 这和画家的绘画方法类似.

定义 1.4 画家算法 上述过程称作**画家算法 (Painter's Algorithm)**.

然而, 真实世界中物体各处的深度值可能并不相同, 因此可能出现复杂的遮挡关系, 使得画家算法难以实现.

深度缓存 为了解决这个问题, 我们不妨转换思路, 由记录每个物体的深度转为记录每个像素的深度值, 所有像素的深度值构成**深度缓存 (Depth Buffer)**. 每个图形各处的深度值也不是固定的. 在绘制时, 我们对每个像素独立检测, 如果发现等待绘制的图形上的深度小于当前像素的深度, 则覆盖当前像素的颜色和深度值, 否则表示图形在此处被遮挡, 不更新像素.

定义 1.5 上述过程称作**深度缓存 (Depth Buffer)** 算法.

深度缓存技术可以处理更复杂的情况, 并且只需要维护所有像素的缓存, 是一种更高效的办法.

然而, 深度缓存技术对于半透明物体的处理并不理想, 因为半透明物体叠加后的颜色仍密切取决于此处各物体的前后关系. 因此, 通常在这些地方仍然需要使用画家算法.

1.3.2 着色模型

我们现在考虑第二个问题: 确定像素的颜色. 这可以用各种着色模型来实现. 下面给出几种常见的着色模型.

平面着色 最简单而直接的办法就是使用网格的每个面的法向量进行光照计算, 并为整个面赋予同样的颜色.

定义 1.6 平面着色 上述过程称作**平面着色 (Flat Shading)**.

在平面着色中, 每个面只有一个法向, 因而只有一个颜色, 于是我们能在渲染结果里看到很多独立而不连续的面片, 但是我们还是可以直观看到整个形状上明暗的过渡以及高光. 如果要得到比较平滑的结果, 就需要更精细的网格. 为了解决这一问题, 人们提出了许多其它的着色模型.

Gouraud 着色 与平面着色不同, 我们可以根据每个顶点的法向量进行光照计算, 并为顶点赋予颜色. 然后在渲染时, 通过重心插值等插值方法得出三角形面片内部各点的颜色.

定义 1.7 Gouraud 着色 上述过程称作 **Gouraud 着色 (Gouraud Shading)**.

Gouraud 着色在曲面的大部分区域里都可以得到光滑的颜色过渡, 但是在高光区域我们能明显看到三角形插值的痕迹. 这是因为在高光区域附近, 颜色变化随着法向量变化比较明显, 并且这不是线性关系 (例如, 如果高光中心在三角形面片的中心, 而顶点处颜色较暗, 显然不能通过插值表现出中心更亮的颜色).

Phong 着色 [1]为了克服 Gouraud 着色在高光区域的不足, 我们可以直接在三角形面片的**每个像素**处计算法向量, 并根据该法向量计算颜色.

定义 1.8 Phong 着色 上述过程称作 **Phong 着色 (Phong Shading)**.

1.4 风格化渲染

应用于不同场景的渲染并不一定追求真实感, 有时希望突出/隐去一些细节或展现特定的艺术风格. 这类渲染称为**风格化渲染 (Stylized Rendering)** 或**非真实感渲染 (Non-photorealistic Rendering, NPR)**.

简单而言, 风格化渲染中比较重要的两个特征分别是**线**和**风格化的着色模型**. 我们现在分别介绍这两种特征相关的处理方法.

1.4.1 轮廓线提取

物体的轮廓线对描述物体的信息非常重要. 在风格化渲染中通常会对轮廓线进行特殊处理.

定理 1.9 轮廓线的提取方法 I 物体的边缘位置的发现 $\boldsymbol{n}$ 应当与视线方向 $\boldsymbol{v}$ 近似地垂直, 因此我们可以判断物体上所有满足 $|\boldsymbol{n}\cdot\boldsymbol{v}|<\varepsilon$ 的点, 这些点就可以视作物体的轮廓线.

上述办法依赖于 ε 的设定, 并且得到的边缘并不一定粗细均匀. 由此, 我们还有另一种办法.

[1]注意与前面的 Blinn-Phong 反射模型加以区分. 两者分别是反射模型和着色模型, 并没有直接联系.

定理 1.10 轮廓线的提取办法 II 我们绘制两遍几何体. 第一遍只绘制背面, 并且绘制的颜色为轮廓线的颜色, 同时将几何体向外扩展一些; 第二遍再在背面上绘制正面的几何体, 如此, 没有被遮挡的部分就是轮廓线. 这种办法又被称作**程序化几何法**.

使用上述办法时, 需要注意向外扩展的实现办法, 也就是将顶点沿它的法向移动一定距离, 并且这段距离在屏幕上最终呈现的长度是固定的, 因此这需要考虑投影变换带来的影响; 此外, 绘制正面的几何体时, 需要使用深度缓存, 否则可能丢失部分轮廓线.

1.4.2 风格化着色模型示例——Gooch 着色

除去用光影表现物体的形状, 我们还可以使用颜色的冷暖变化来表现物体的形状. 一种常见的风格化着色模型是 **Gooch 着色 (Gooch Shading)**.

定义 1.11 Gooch 着色 取定冷色 k_c 和暖色 k_w, 物体上各点的颜色通过插值得到:

$$k = \left(\frac{1+\boldsymbol{l}\cdot\boldsymbol{n}}{2}\right)k_c + \left(\frac{1-\boldsymbol{l}\cdot\boldsymbol{n}}{2}\right)k_w$$

其中 $\boldsymbol{l}$ 是物体上的点指向光源的方向, $\boldsymbol{n}$ 是该点表面发现的方向.