计算机图形学

一、光栅化成像

1.1 三角形网格性质

1. 最简单的多边形
2. 其他多边形可以拆分成三角形
3. 三角形一定是平面的
4. 内外定义清晰
5. 对内部的任意点方便做插值（重心坐标）

1.2 采样、走样

走样的例子

- 锯齿：空间采样
- 摩尔纹：欠采样的图像
- 车轮错觉：时间采样

走样的原因：信号变换的太快，错误识别信号频率

改善走样

- 提高采样率
- 反走样

1.3 反走样

原理：采样前先过滤高频信号

超级采样：通过在一个像素中多次采样，计算他们的平均值作为平均像素值

	超级采样	点采样
采样率	一个像素中一个采样点	一个像素中 `N*N` 个采样点
颜色	内红外白	计算每个像素中 `N*N` 个采样点的平均值

1.4 遮挡/可见性

画家算法

- 受画家由远至近作画方式的启发，近处物体覆盖遮挡远处物体
- 在帧缓冲器中重写

画家算法的问题

- 对场景中的多边形按深度排序（复杂度 $\mathrm{O}(n\log(n))$）

- 存在无解的深度顺序（在交点处分割成多个子多边形）

Z-buffer 算法

- 思想：对每个采样点（像素）记录当前最小的 Z 值
- 时间复杂度：$\mathrm{O}(n)$，易于硬件实现

Z-buffer 伪代码

- 帧缓冲器：framebuffer 记录颜色值
- 深度缓冲器：zbuffer 记录深度值

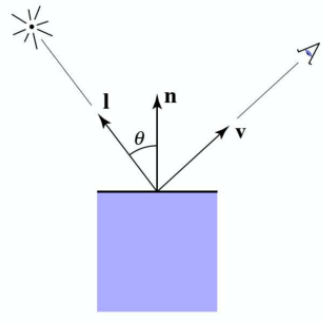
```
for (each sample(x, y))
    zbuffer[x, y] = infty

for (each triangle T)
    for (each sample(x, y, z) in T)
        if (z < zbuffer[x, y])
            framebuffer[x, y] = rgb
            zbuffer[x, y] = z
```

1.5 Blinn-Phong 反射模型

漫反射：

- 着色与观察方向无关

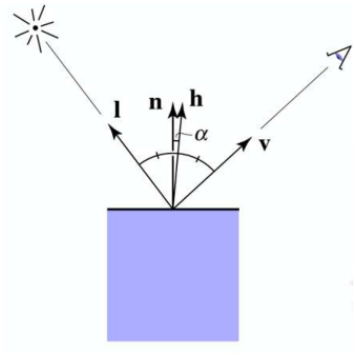
$$L_d = k_d \cdot \frac{I}{r^2} \cdot \max(0, \mathbf{n} \cdot \mathbf{l}) \tag{1}$$



镜面高光：

- 强度取决于观察方向：近镜面反射方向更加明亮
- v 接近镜面反射方向等价于半程向量接近法线方向

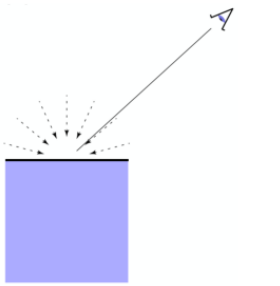
$$\begin{aligned} \mathbf{h} &= \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|} \\ L_s &= k_s \cdot \frac{I}{r^2} \max(0, \mathbf{n} \cdot \mathbf{h})^p \end{aligned} \tag{2}$$



环境光：

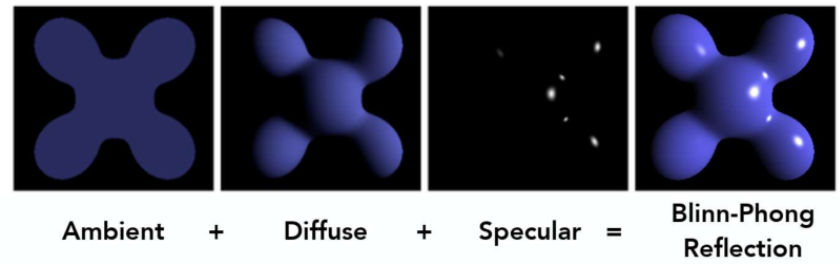
- 着色不取决于任何其他因素
- 添加常量颜色
- 这是一种近似，并不真实

$$L_a = k_a I_a \tag{3}$$



Blinn-Phone 反射模型：

$$\begin{aligned} L &= L_a &&+L_d &&+L_s \\ &= k_a I_a &&+k_d \cdot \frac{I}{r^2} \cdot \max(0, \mathbf{n} \cdot \mathbf{l}) &&+k_s \cdot \frac{I}{r^2} \max(0, \mathbf{n} \cdot \mathbf{h})^p \end{aligned} \tag{4}$$



1.6 着色频率

着色类型	面着色	点着色	像素着色
着色类型	Flat shading	Gouraud shading	Phong shading
说明	（1）三角形是平面的：得到一个面的法向量 （2）不适合光滑的表面	（1）三角形的顶点携带颜色信息 （2）每一个顶点上有一个法向量	（1）插值得到法向量 （2）在每个像素上计算着色模型 （3）区别 Blinn-Phong 反射模型

法向量的计算

- 面法向量：面上两个向量的叉乘（注意方向）再归一化
- 顶点法向量：最好是从想要表示的几何体获取顶点法向，否则利用环绕顶点的面法向取平均再归一化
- 像素法向量：利用顶点法向的重心插值，并归一化

1.7 重心坐标

定义：利用三角形的三个顶点建立的一套坐标系

$$
\begin{aligned}
(x, y) &= \alpha A + \beta B + \gamma C \\
\alpha + \beta + \gamma &= 1
\end{aligned}
\tag{5}
$$

求解 (α, β, γ)：

$$
\begin{aligned}
\alpha &= \frac{A_A}{A_A + A_B + A_C} \\
\beta &= \frac{A_B}{A_A + A_B + A_C} \\
\gamma &= \frac{A_C}{A_A + A_B + A_C}
\end{aligned}
$$

举例

- A 点的重心坐标：$(1, 0, 0)$
- 重心的重心坐标：$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

应用：

- 点是否在三角形内
- 光线与三角形求交

在投影变换（透视投影）下，重心坐标无法保持不变！

1.8 纹理的应用

纹理过小的问题 → 纹理放大

- 最近邻方法：直接取离采样点最近的纹理值，作为采样点的值
- 双线性插值：取最近的 4 个点，做两次线性插值

  lerp(x, u, v) = u + x(v - u)

纹理过大的问题 → 纹理缩小

- 超级采样，但代价很高
- Mipmap（多级贴图）：三线性插值，在 D 层做双线性插值再在 D + 1 层做双线性插值，对两个结果在做一次线性插值
  - 存在过度模糊
  - 存储空间多占用约 1/3

凹凸贴图：添加表面细节但不改变任何几何信息

- 方法
  - 在每个像素上扰动表面法线
  - 利用纹理定义每个纹素上的“高度偏移”
  - 计算扰动后的法向量
    - 原始表面法线：$n(p) = (0, 1)$
    - P 点出的导数：$dp = c * [h(p + 1) - h(p)]$
    - 扰动后的法向量：$n(p) = (-dp, 1).\text{normalized}()$
- 不足
  - 在模型边缘不能很好的模拟凹凸效果
  - 阴影效果还是由真实的几何信息计算得出

位移贴图：真正移动了三角形的顶点

- 不足
  - 对三角形数目的要求（可以利用动态曲面细分来解决）

二、几何表示

2.1 几何的表示形式

	隐式表示	显式表示
定义	定义几何体上的点满足的关系，并不给出实际的点	直接给出几何体上的所有点，或者通过参数映射给出
举例	代数曲线、水平集、距离函数、构造实体几何（布尔运算）	点云、多边形网格、细分、NURBS、Bezier 曲面、细分曲面

隐式表示：

- 优点
  - 简洁的描述（利用函数）
  - 某些查询很容易（点在不在内部，点到表面的距离）

    - 适合做光线与表面的求交
    - 对于简单形状，描述准确，不存在采样误差
    - 易于处理拓扑变化
- 缺点
    - 复杂形状建模困难

显式表示：

- 点云
    - 最简单的表示形式：点（x, y, z）的列表
    - 可以方便的表示各种几何类型
    - 适用于大数据集
    - 通常需要转换成多边形网格
    - 在欠采样区域难以绘制
- 多边形网格（Polygon Mesh）
    - 存储顶点和多边形
    - 易于进行处理/模拟、自适应采样
    - 相较于点云，需要更复杂的数据结构
    - 图形学中最常用的表示形式

2.2 贝塞尔曲线

公式、性质、优缺点、计算

2.2.1 de Casteljau 算法

二次

$$
\begin{aligned}
b_0^1(t) &= (1-t)b_0 + tb_1 \\
b_1^1(t) &= (1-t)b_1 + tb_2 \\
b_0^2(t) &= (1-t)b_0^1 + tb_1^1
\end{aligned} \tag{6}
$$

三次略了

2.2.2 代数公式

Bernstein 多项式

$$
\begin{aligned}
B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i} \\
b^n(t) &= \sum_{i=0}^{n} B_i^n(t) b_i
\end{aligned} \tag{7}
$$

二次公式

$$b^2(t) = (1-t)^2 b_0 + 2t(1-t)b_1 + t^2 b_2 \tag{8}$$

三次公式

$$b^3(t) = (1-t)^3 b_0 + 3t(1-t)^2 b_1 + 3t^2(1-t)b_2 + t^3 b_3 \tag{9}$$

性质

- 端点位置
  - $P(t)|_{t=0} = P_0$
  - $P(t)|_{t=1} = P_n$
- 端点切向量
  - $P'(t)|_{t=0} = n(P_1 - P_0)$
  - $P'(t)|_{t=1} = n(P_n - P_{n-1})$
- 仿射变换性质：通过变换控制点达到变换曲线的目的
- 凸包性质：曲线位于控制点的凸包内

2.2.3 分段 Bezier 曲线连续性判断

C^0 连续：$a_n = b_0$

C^1 连续：$a_n = b_0 = \frac{1}{2}(a_{n-1} + b_1)$

2.2.4 贝塞尔曲线优缺点

优点

- 形状控制直观，设计灵活，应用较为广泛

缺点

- 控制顶点数较多时，多边形对曲线的控制能力减弱
- 控制顶点数增多时，生成曲线的阶数也增高
- 曲线拼接需要附加条件，不太灵活
- 局部控制能力弱：因为 Bezier 基函数的值在（0,1）开区间内均不为零，曲线上任意一点都是所有给定顶点的加权平均

2.3 几何处理

	Loop 细分	CatMull-Clark 细分
如何加点	每个三角形插入新顶点，连接形成 4 个小三角形	面中心加点和边中点加点，连接形成新面
适用范围	针对三角形网格	针对多边形网格，多用于四边形网格
原因	简单高效，每个三角形独立处理，并行性好	通用性强，收敛性好

CatMull-Clark 细分用于三角形网格时会导致大量奇异点

三、光的传播理论

3.1 基本的光线追踪算法

3.1.1 光线投射算法

基本思想

1. 从视点或像素出发，仅对穿过像素的光线反向跟踪
2. 当光线路径到达一个离视点最近的可见的不透明物体的表面（即为屏幕上该像素对应的可见面）时，停止追踪

3.1.2 Whitted-Style 光线追踪算法

基本思想

1. 沿着到达视点的光线的相反方向追踪
2. 经过屏幕上一像素点找出与视线所交的物体表面点 P0
3. 继续追踪，找出影响 P0 点光强的所有的光源
4. 算出 P0 点上精确的光照强度

结束条件

- 光线与光源相交
- 光线与漫反射表面相交
- 被追踪的光线对第一个交点处的贡献趋近于 0

3.1.3 二者异同

光线投射算法：被追踪的光线仅从每个像素到离它最近的景物为止

Whitted-Style 光线追踪算法：通过追踪多条光线在场景中的路径，以得到多个景物表面所产生的反射和折射影响

Whitted-Style 光线追踪算法是光线投射算法的延伸

3.2 光线-物体求交

3.2.1 光线与三角形求交

考虑三角形所在的平面

- 光线与平面求交：$(\mathbf{o}+t\mathbf{d}-\mathbf{p})\cdot\mathbf{N}=0$ 检查 $0\leq t<\infty$
- 检查交点是否在三角形内

Möller Trumbore 算法

- 原理：$\mathbf{O}+t\mathbf{D}=(1-b_1-b_2)\mathbf{P_0}+b_1\mathbf{P_1}+b_2\mathbf{P_2}$
- 结果：

$$\begin{pmatrix} t \\ b_1 \\ b_2 \end{pmatrix} = \frac{1}{\mathbf{S_1} \cdot \mathbf{E_1}} \begin{pmatrix} \mathbf{S_2} \cdot \mathbf{E_2} \\ \mathbf{S_1} \cdot \mathbf{S} \\ \mathbf{S_2} \cdot \mathbf{D} \end{pmatrix} \tag{10}$$

- 其中：

$$\begin{aligned} \mathbf{E_1} &= \mathbf{P_1} - \mathbf{P_0} \\ \mathbf{E_2} &= \mathbf{P_2} - \mathbf{P_0} \\ \mathbf{S} &= \mathbf{O} - \mathbf{P_0} \\ \mathbf{S_1} &= \mathbf{D} \times \mathbf{E_2} \\ \mathbf{S_2} &= \mathbf{S} \times \mathbf{E_1} \end{aligned} \tag{11}$$

3.2.2 光线与轴对齐包围盒求交

主要思想：

- 仅当光线进入到所有成对平板时，光线进入到包围盒内
- 光线离开任意一对平板时，即离开了包围盒

步骤：

- 对每一对平板，计算 $t_{\min}$ 和 $t_{\max}$
- 对包围盒，$t_{\text{enter}} = \max\{t_{\min}\}$，$t_{\text{exit}} = \min\{t_{\max}\}$

有无交点需满足的条件：

- $t_{\text{enter}} < t_{\text{exit}}$
- $t_{\text{exit}} \geq 0$

3.3 利用 AABB 加速光线追踪

BVH 算法步骤

- 确定包围盒
- 递归地将物体划分为两部分并重新计算包围盒
    - 选择最长轴方向做划分
    - 选择中间物体位置来划分
- 停止后将物体存储在每一个叶子节点中

	空间划分（KD 树）	物体划分（BVH）
划分角度	把空间划分成不重叠的区域	把物体集合划分成不相交的子集
物体所在区域	空间中的一个物体可能包含在不同的区域中	空间中的一个物体可能包含在不同的区域中
中间节点	按照沿 x 轴、y 轴、Z 轴的划分平面来划分 指向孩子节点的指针 不存储空间中的物体	存储包围盒 指向孩子节点的指针
叶子节点	存储空间中的物体	存储包围盒 存储空间中的物体

四、动画与模拟

4.1 正向运动学

动画师提供角度，计算机确定末端位置 p，动画被描述为关于时间的角度参数值的函数

优点

- 直接控制，非常方便
- 很直观，容易实现

缺点

- 动画可能与物理规律不一致
- 艺术家需要耗费大量时间

4.2 逆向运动学

动画师提供末端位置 p，计算机必须给出满足约束条件的关节角度.

可能存在多个解，也可能无解

4.3 动画绑定

动画绑定是一组更高级别的对动画角色的控制操作，允许更快速和直观的修改姿势、变形、表情等。

缺点：应用起来代价较高

- 人力成本高
- 对艺术造诣和技术能力都有较高要求

4.4 动作捕捉

动画捕捉是利用数据驱动的方式来创建动画序列：

- 记录真实世界中人的行为
- 从收集到的数据中提取姿态，生成关于时间的函数

优点

- 能够快速捕获大量真实数据
- 生成的动画真实度高

缺点

- 需要复杂和昂贵的前期配置
- 捕获的动画可能不符合艺术需求，还需要后期修改