计算机图形学

Computer Graphics

张思容
zhangsirong@buaa.edu.cn

数学与系统科学学院，北京航空航天大学
School of Mathematics and Systems Science, Beihang University

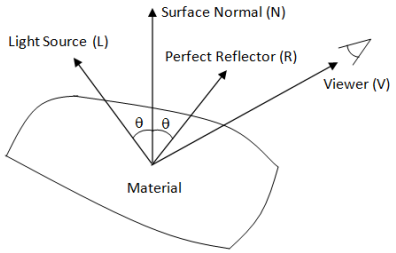
November 22, 2012

Chapter 3: 真实感图形学

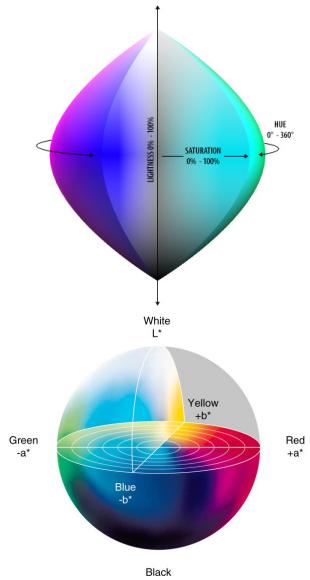
真实感图形学的基础

真实感图形学：几何准确+物理近似+感觉逼真

- 几何准确：透视模型，符合几何关系。
  计算算法：阴影+消除隐藏面
- 物理近似：近似计算环境的光照和场景的布置；
  简单光照模型；整体光照模型(光线追踪+辐射度）；局部实现算法；
- 感觉逼真：颜色和纹理的心理学基础；
  颜色模型，纹理映射；



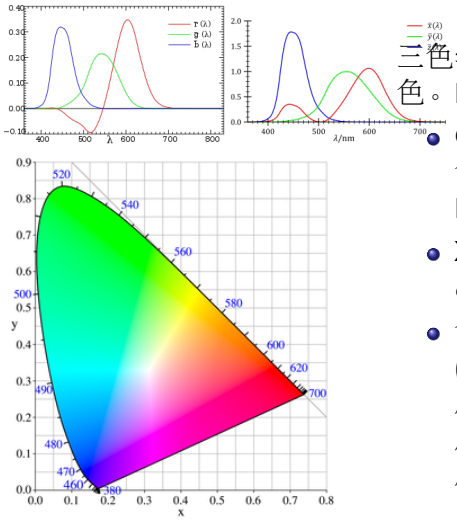
颜色的视觉基础



颜色的光学和视觉特征：

- 可见光:波长
  380nm→780nm,$nm = 10^{-7}m$.
  $c = \lambda f$
- 物理特征: 光是一个能量谱分布。无法严格定义颜色(多对一关系)
  主要特征:主波长(color)，亮度（brightness), 纯度purity（saturation），
- 视觉基础: 人眼的三种锥状细胞(颜色)和杆状细胞（亮度)
  颜色受大脑影响: 环境，心理，文化等。

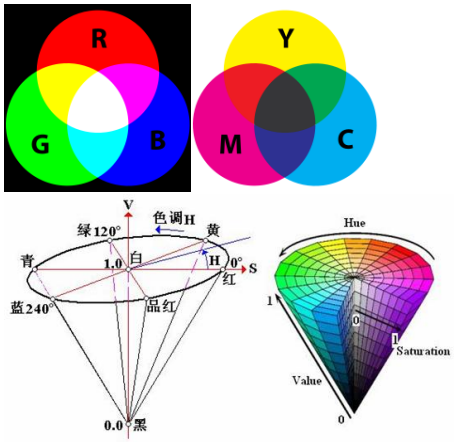
颜色空间的三色模型



三色学说:红+黄+蓝可以得到所有颜色。RGB

- CIE：国际照明委员会 1931 标准三原色: $c = rR + gG + bB$
  RGB空间有负值!
- XYZ空间：数学三原色，第一象限。$c = rX + gY + bZ$
- 色度图:chromaticity
  $(x, y, z) = c/|c|$,取x, y.$Y = |c|$是亮度;
  用于比较不同颜色空间范围，得到色度(波长和纯度).得到互补色;

常见颜色模型



常见颜色模型：位于CIE颜色空间的一个子集

- RGB：加法模型,常见于监视器:
  电视:NTSC(YIQ); PAL(YUV)
- CMY：减法模型：印刷业；CMYK加入黑色
  $[C, M, Y] = [1, 1, 1] - [R, G, B]$
- HSV: 面向用户模型
  (hue,saturation,value);

应用程序API中的颜色

OPENgl:颜色模型:RGBA

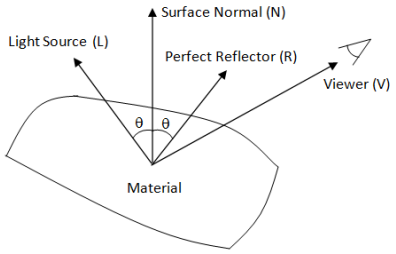
- 初始化 $glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)$;
  或者RGBA
- 颜色函数： $glColor3/4*()$ 或$glIndex*(colorIndex)$
- 清屏:$glClearColor(R, G, B, A)$; 缓存$glClear(GL-COLOR-BUFFER_BIT)$
  $glClearIndex(index)$
- 实用技术：颜色混合 $glEnable(GL-BLEND)$;
  $glBlendFunc(sdactor, dfactor)$;

MATLAB: RGA向量或字母
[1 1 0] y yellow ,[1 0 1] m magenta ,[0 1 1] c cyan
[1 0 0] r red,[0 1 0] g green,[0 0 1] b blue
[1 1 1] w white,[0 0 0] k black

物理光学的基础I:光源

光源 $I(\lambda, p)$

- 点光源: $p = (x, y, z)$, 强度衰减$f_r(d) = 1/(a_0 + a_1 d + a_2 d^2)$
- 无穷远光源: $v = (vx, vy, vz)$,强度衰减$f_a(d) = 1$
- 聚光灯光源： p, v,角强度衰减$f(\phi) = \cos^a \phi$



物理光学的基础II:光照

表面光照：$I = I_v + I_r + I_t$ 吸收，反射,折射;
基本光照模型：设入射光线I_i

- 环境光:I_a
- 吸收：与表面有关;
- 透射：两次折射；Snell定律 相当于微小平移
  简单模型:$I_i = (1 - k_t)I_r + k_t I_t$, 颜色的$alpha$混合;
- 反射=漫反射+镜面反射:$I_r = I_d + I_s$

漫反射模型: Lambert 余弦定律：强度与夹角的余弦成正比。
点光源 $I_d = k_d I_i \cos\theta, \cos\theta = dot(N, L)$
环境光:$I_d = k_a I_a$

物理光学的基础III:镜面反射Phong模型

镜面反射：沿理想反射方向R附近的高光反射。

- 镜面反射Phong模型：强度与$\cos^n\phi$成正比。
  ϕ是观察方向V和理想反射方向R的夹角.
  $n = 1 \to 100$
- 镜面反射系数$W(\theta)$:与入射角度,颜色，材料相关。
  $I_s = W(\theta)I_i \cos^n\phi$.
- 简单模型:定义$W(\theta) = k_s$,
  半角向量$H = (L + V)/|(L + V)|$,
  $\cos\phi = dot(V, R) \sim dot(N, H)$

一般模型: $I = I_o + I_a + \sum f_r f_a(I_d + I_s)$

真实显示技术

应用考虑:

- 颜色： 可以I对每个颜色指定计算，或I_B, I_G, I_R;
  或相同系数不同反射分量;
- 亮度:$lum = \int p(f)I(f)df$,一般 $lum = -.299R + 0.587G + 0.114B$
  或$lum = Y$
- 雾气: $f_g(d) = e^{-\rho d}$或$f_g(d) = e^{-\rho^2 d}$
  有颜色的雾：$I = f_g I_o + (1 - f_g)I_a$
- 阴影：隐藏面算法:
  glShadeModel(GL-SMOOTH); glEnable(GL-DEPTH-TEST);

真实世界光的显示：人眼对光的强度感觉按对数变化!!!
$I = 0 \to 1$ 对应n个强度等级$I_k = (1/I_0)^{1/n}, I_0 > 0$
电视：Gamma校正 $I = aV^\gamma, V = (I/a)^{1/\gamma}$， NTSC:$\gamma = 2.2$
印刷：半色调技术$halftone$ 利用网格实现更多强度。

应用API中的光照

openGL

- 光源: 至多8个
- 基本设置： glutInitDisplayMode (GLUT-SINGLE — GLUT-RGB — GLUT-DEPTH);
  glShadeModel(GL-SMOOTH); glEnable(GL-DEPTH-TEST);
- 光源:
  glLight*(i,f,v)(GL-LIGHT0, GL-POSITION, light-position);
  glEnable(GL-LIGHT0);
  glEnable(GL-LIGHTING);

MATLAB: 光仅仅对曲面片有效，还要设置相关曲面的光学系数;
h = surf(peaks);
set(h,'FaceLighting','phong','FaceColor','interp',... 'AmbientStrength',0.5)
light('Position',[1 0 0],'Style','infinite');

整体模型I: 光线追踪

Ray tracing：对投射平面上每一个像素通过光线(逆)追踪，得到光照I的方法。

- 基本设置: 投影观测点V在z轴，观察平面$XY: z = 0$,光线VP;
- 追踪过程: 每一个平面发生反射(可能折射)，继续反射，或加上新的从属光线，
  直到: 光线不与任何物体相交；或者光线到达一个非反射光源，或者反射与折射次数达到最大值。
- 计算：建立二叉光线跟踪树，每一次相交的交点是树的节点；同时得到反射和折射两个分支；
  光强计算:从叶子出发，每个节点按简单模型计算光强；
  依次向上累积计算光强(考虑衰减因素);直到根节点。
- 优缺点: 物理近似，阴影效果比较真实，计算量极大(大量求交运算)；
- 改进: 快速计算光线与曲面求交；分割空间减少求交；
  复杂光线追踪: 过采样，自适应采样；分布式随机光线；

光线追踪的基本算法:

总光线亮度$I = I_o + \sum_i k_s I_s^i + k_t I_r^i$,
该点的直接光线亮度：$I_o = I_a + k_d I_d + \sum k_s I_s$
该点的间接亮度(反射和折射)：I_s', I_r'.

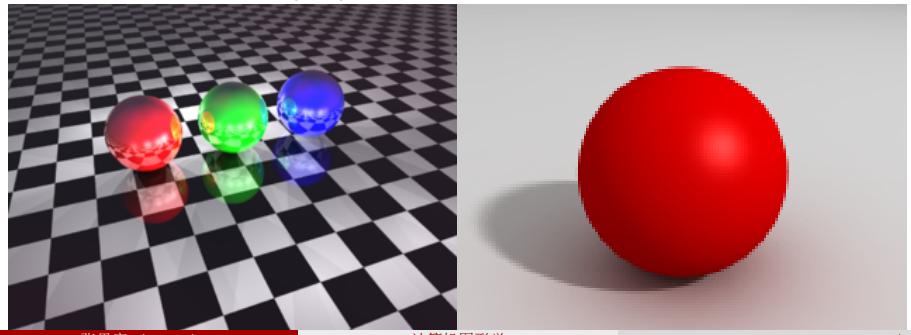
- I_o计算:
  阴影算法：该点到光源的(阴影)虚拟光线，判断求交。
  有交：透明系数：$I_o = k_o I_o$;
- I_s计算：反射光线：$R(t) = P + tW, W = 2dot(V, N)N - V$;
  光线不交物体：$I_s' = 0$,交光源$I_s' = d_p I_p$,
  交某个物体：继续递归新的虚拟光线;
- I_r计算：折射光线：$R(t) = P + tW$,
  $W = (c_2/c_1 dot(V, N) - \cos\theta_2)N - c_2/c_1 V$;
  $\cos\theta_2 = \sqrt{1 - (c_2/c_1\Psi^2(1 - dot(V, N))^2}$
  光线不交物体：$I_r' = 0$,交光源$I_r' = d_P I_p$,
  交某个物体：继续递归新的虚拟光线;

一般情形: 还应该有间接的漫反射亮度，阴影不柔和!

整体模型II: 辐射度模型

Radiosity Model（1984）：考察光照能量在不同物体表面的分布，计算求出全局漫反射的解;

- 基本术语: 能量$E = hf$；光通量$\Phi = dE/dt$, 单位是瓦;
  辐射度$B = d\Phi/dA$,A是面积;
- 基本模型: 不同平面k有 $B(k) = E(k) + \rho(k)\sum_j B(j)F(j, k)$
  其中$E(k)$是发射光，$\rho(k)$是反射因子，$F(j, k)$是形状因子;
- 计算：可以计算$F(j, k)$,迭代求出方程组的最佳解。



图形的反走样技术***

Definition (aliasing 走样)

从连续对象提出离散样本的过程中造成信息失真的现象称为走样。

抽样定理:抽样频率大于对象最高频率的两倍不走样(Nyquist频率)。
一般情形要用反走样技术。

- openGL函数：$glEnable(primitiveType)$(或颜色$GL - BLEND$)
  $GL - POINT - SMOOTH, GL - LINE - SMOOTH, GL - POLYGON - SMOOTH$
  注；openGL有约20个属性组，可以$glGet()$或属性堆栈查询处理;
- 反走样技术：提高物理分辨率；利用图像的亮度或颜色光滑化;
  常见技术：过抽样（supersampling），区域抽样（area）
- 例子：直线的反走样；postfiltering(过抽样再求和),
  prefiltering(区域求和),或其他滤波器技术;

直线的亮度调整：$y = x$比$y = 0$要暗!

几何造型的常见表示

模型和表示

- 几何模型：线框模型；曲面模型，实体模型
- 实体模型的表示：空间分解表示，构造表示，边界表示
  例子：8叉树；构造实体几何CSG树，特征表示；B-reps;

边界表示模型：

- 内容：几何信息：顶点坐标。
  拓扑信息：顶点，边，环（loop），面，体；
- 数据结构： wing-edge，radial-edge；
- 几何运算：欧拉操作；集合运算；求交

实体造型系统： Parasolid， ACIS；

OPENGL中的多边形和多面体***

多边形和多面体

- 多边形： $GL-PLOYGON, GL-TRIANGLES, GL-QUAD$
- 多面体:glut*cube,tetrahedon,octahedron,dodecahedon,icosahedron;
- 凹多边形分解***：gluTess;
- 例子：构造球面的多边形逼近：细分法。

实用技术：顶点数组vertex array

- 例子：正方体的画法（六个面）
- 顶点数组：$glEnableClientState(GL-VERTEX-ARRAY)$
  $glVertexPointer(3, GL-INT, pt)$
  $glDrawElements(GL-QUADS, 24, 0, verIndex)$;

OPENGL还有其他相关函数：

三维空间的曲面

一般曲面的表示

- 参数表示： $S(u,v)=(x(u,v),y(u,v),z(u,v))$
- 非参数表示之隐式表示
  水平集： $F(x,y,z)=0$
- 非参数表示之显示表示:
  样条曲面，分形等计算机生成曲面;

曲面与仿射变换： M

- 参数表示:$S=M*S$
- 非参数表示之隐式表示: $F^*(M^{-1}P^*)=0$
- 非参数表示之显示表示：顶点或特征点变换；再程序生成;

注：以上称为变形曲面:

三维空间的曲面计算

法向量:

- 参数表示： $S(u,v)=(x(u,v),y(u,v),z(u,v))$
  $N=\frac{\partial S}{\partial u}\times\frac{\partial S}{\partial v}$
- 非参数表示之隐式表示: 水平集： $F(x,y,z)=0$
  $N=\nabla F$
- 变形曲面： $N=M^{-\tau}*N$;
  注：利用$dot(v,N)=0, v=M*v$

直线求交： $L(t)=P+vt$ （一般情形）

- 参数表示:求解方程组$S(u,v)=L(t)$
- 非参数表示之隐式表示: $F(L(t))=0$
- 变形曲面：求变形曲面的原像和变形直线原像的交$M^{-1}*L(t)$,
  再求像或利用t求交点；t不变！！！

注：其他几何平均曲率，高斯曲率等参考微分几何教材;

简单曲面的计算

基本思路:变形曲面到简单曲面,利用几何求法向量,交点等,再用逆变换得到可用几何量. 特别:交点可以用射影变换(去掉虚拟交点)

- 多边形：平面和直线方程$dot(N, P-Q)=0, L(t)=P+vt$
  交点:$t_0 = dot(N, Q-P)/dot(N, v)$;再判断交点是否在内部;
- 球面：$|Q-C|=r$, 简化到平面上与直线求交;
  1)求圆心到直线距离：判断是否交;
  2)求到直线最近点：$Q = C + w, w = (P-C) - dot(P-C, v)v$
  3)求交点$A = Q - dv, B = q + dv$.和距离$d = r^2 - |w|^2$
  4)求对应参数t：线性$t = dot(Q-P, v)/dot(v, v)$
- 其他曲面：利用几何求： 圆柱面，环面等;
- 二次曲面:写成二次形$P^t Q(x,y,z)P$,变成二次方程求根!

特别：可用于透视投影得到变形曲面求与直线的交；但要用质点模型，求出直线的有理参数（以上步骤4):
L由点(m_0P_0, m_0)和(m_1P_1, m_1)决定;
$t = (m_0 * dot(Q-P_0, P_1-Q))/(m_1 * dot(P_1-Q, P_1-Q))$

OPENGL的二次曲面***

常见二次曲面：球面，椭球面，环面;
*超二次曲面：例子：$x^{2/s} + y^{2/s} = 1$
OPENGL的二次曲面

- GLUT：sphere，cone，torus，teapot;
- GLU：sphere,disk,partialdisk;
- 例子： $GLUquadricObj * sphere1$;
  $sphere1 = gluNewQuadric()$;
  $gluQuadricDrawStyle(sphere1, GLU-LINE)$;;
  $gluSphere(sphere1, r, nlog, nlat)$;
  **$gluDeleteQuadric(sphere1)$;

多边形绘制模型：局部模型

简单光照模型:$I = I_a K_a + I_r k_d (N \cdot L) + I_r K_s (N \cdot H)^n$
给定多边形，其光照由顶点或重心的简单光照模型直接计算得到。
面的法向量：$N = \sum P_k \times P_{k+1}$ Newell 公式;

- 恒定强度(flat): 多边形面上光照有一点(比如重心)决定。
  优缺点：快速；假设物体是真实平面，光源和视点都足够远；但是有边界不光滑;
- Gouraud： 双线性光照插值;
  计算：顶点法向量$N(p) = \sum N_i / |\sum N_i|$，光照$I$，依$y, x$坐标插值$I(x, y)$;
  优点: 简单；可与扫描算法合并,
  缺点：扫描与坐标位置有关(可用三角形代替，重心坐标唯一！);模糊了镜面反射;
- Phong:双线性法向量插值;
  计算：顶点法向量N，依y, x坐标插值$N(x, y)$；每点计算光照$I(x, y)$,
  优缺点: 更准确；太慢，需要加速算法(线性插值N或球面插值);

隐藏面算法

隐藏面算法：判断某些曲面或曲面片需要画。 一般算法：n个多边形；N个像素;

- 图像空间算法: $O(nN)$:
  对每一点，找到最近多边形，画像素。
- 对象空间算法: $O(n^2)$:
  对每个多边形，找出可见的多边形片;

基本思路：画家算法，依次画所有多边形即可； （可能有错误!)
图像空间算法: $z-buffer$算法（内存保留深度信息，依次画）;
扫描线算法(画一次，利用边表比较多边形)
对象空间算法: 深度排序算法（依深度给多边形排序，再考察不能排序情形,必要时得到曲面片);
BSP算法：(一次排序多边形，适当多边形分块),适用与任何视点!

纹理与纹理映射

物体表面的真实度依赖与细节或纹理:
常见纹理：几何纹理；表面纹理(简单纹理，图像纹理):

- 几何纹理:
  bump映射：改变表面的法向量 $P = P + Nb(u, v)$
  frame 映射：改变表面的法向量和坐标系； $P = P + Nb(u, v) + Tc(u, v)$
- 表面纹理：一维，二维，三维纹理;
  简单纹理：$b(s, t)$ 函数纹理(比如棋盘);
  图像纹理：$I(s, t)$,$P(s, t, w)$
- 纹理映射:纹理空间$I(s, t), [0, 1] \times [0, 1]$
  纹理扫描映射：$F : (s, t) \to (u, v) \to (x, y)$ 注：边界不能匹配!
  图像扫描映射：$G : (x, y) \to (u, v) \to (s, t)$ 注：常用，滤波处理;

OPENGGL 光照模型和纹理***

OPENGGL 光照模型:
glLightModel*()： 环境光，观察方向，镜面反射颜色；前后面;
glmaterial（）： 各种反射系数；；
OPENGL 表面绘制模型:
glShadeModel(): flat, Smooth(Gouraud);
glNormal*();
OPENGL 纹理映射模型:
glTexImage(1,2,3)D();glEnable();
glTexParameter*();glTexCoordi*();