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
Department of Mathematics, Beihang University

September 14, 2011

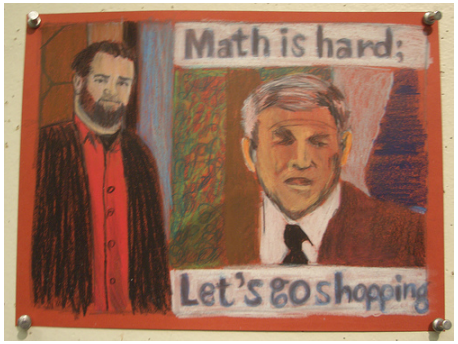
Chapter 1: 计算机图形学简介

1. 什么是计算机图形学？
   - What is Computer graphics?
   - 课程大纲
   - 计算机图形学的应用及例子
2. 交互式计算机图形学的基础
   - 图形学硬件介绍:
   - 图形学软件介绍：openGL

引言

自我介绍

- 张思容: Ph.D. 几何分析，医学图像分析;
- 办公时间：周二(12pm-2pm)或预约。 图书馆西配楼501
- 联系方式：134-3920-1025. zhangsirong@buaa.edu.cn
- 欢迎大家学期中提建议和问题!

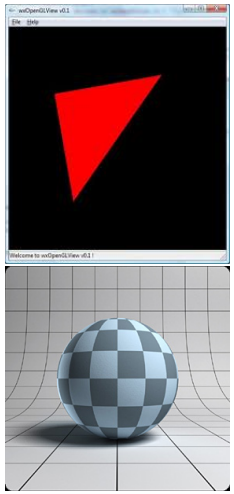


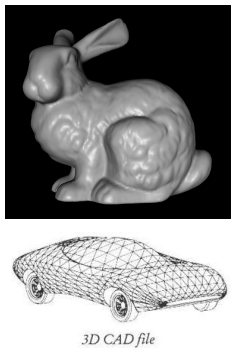
关于课程：
选修,限选？
数学课程还是计算机课程？
难易？

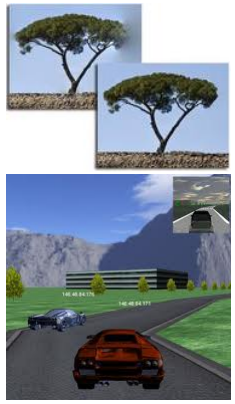
What is Computer graphics?

- 字面解释： 计算机+图形学
  "计算机画图"
- 图形即几何：平面几何，射影几何，
  解析几何, 微分几何...
- 组成：硬件+软件(设备, 编程语言)
- 内容：简单图形, 复杂几何对象，真实图像；(科学数据，计算机艺术，虚拟现实...)

What is Computer graphics? 例子







计算机图形学相关的学科和概念

图形 vs 图像

- 图形：有几何学模型(和物理模型), 可以有不同表示形式(图像) 主要用于计算机辅助几何设计(制造)
- 图像：是位图bitmap 数据(矩阵). 其数学模型未知(随机模型？). 主要用于图像处理，模式识别。

相关学科：

- 计算机图形学衍生的相关分支：
  计算机辅助几何设计CAGD, 计算机绘图drawing, 虚拟现实, 计算机动画，计算机艺术。。。
  交互式系统设计；科学计算可视化；
- 图像科学：图像处理与分析, 模式识别, 人工智能；
- 计算机视觉 Computer Vision:
- 计算几何：Computational Geometry;

课程内容

预备要求：
解析几何？线性代数, C语言, 耐心!
主要目标：
使用OPENGL, 了解相关算法***,
学习几何模型，做点有趣的应用；
教学参考书:

- (教材)计算机图形学基础教程(第二版)：孙家广,胡事民. 清华大学出版社.
- (参考教材)计算机图形学（openGL版)，Hearn. 电子工业出版社。
- (推荐)计算机图形学(C版)：原理与实践，Foley.
  入门版：计算机图形学导论：Foley 等。
- Graphics Gems I-V: 图形学宝藏： 非常适合课程设计；
- openGL 编程指南：红宝书。

课程安排

教学日历：上课16周 (9/6-12/27)

- 第一章：介绍：2周
- 第二章：光栅图形学：4周
- 第三章：几何模型：曲线与曲面 4周
- 第四章：真实感图形学 4周
- 第五章：其他主题和课程设计答辩 2周

考核：

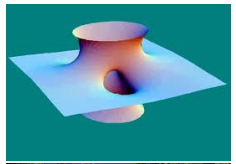
- 作业3-4次(包含上机作业)
- 课程设计：1-2人
- 成绩：作业60+课程设计30+课堂参与10=100分
- 要求：提问参与!!!

插曲：

十年前我的图形学课程设计：
万花筒
flowers

历史和应用

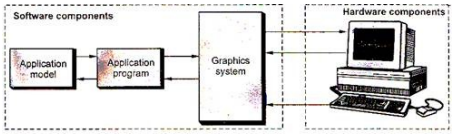
- 起源： 1962，MIT实验室 Sutherland 博士论文：交互式SketchPad.
  法国雷诺汽车公司：Bezier曲面；
- 技术发展： CGI, GKS, PHIGS;
  openGL：VHML，SIGGRAPH.
- 计算机更新： 八十年代： APPLE II, 九十年代：DOS到WINDOWS; 二十一世纪：Ipod到Ipad(Iphone);
- 应用技术：医学图像；



应用领域

- 生活： UI:人机界面
- 工业：CAGD: 工业设计（Bezier 曲线曲面);
- 娱乐：计算机艺术和动画。
- 科学研究：可视化, 医学图像;

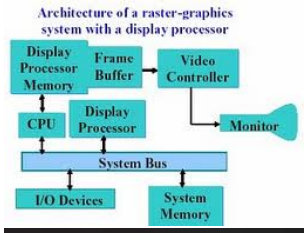
简单交互式图形系统



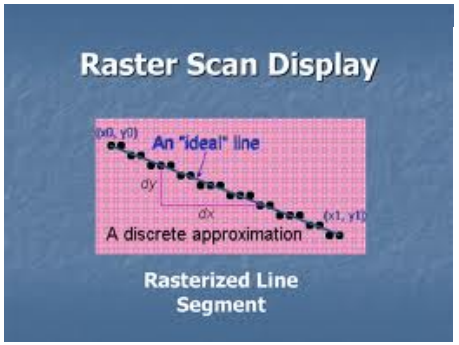
交互图形学的系统结构

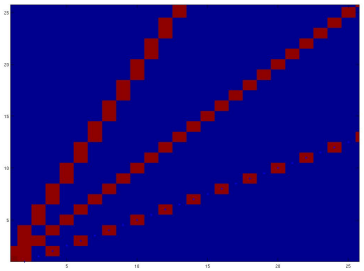
- 应用模型： 几何建模+物理模型（颜色与光）
- 应用程序：人机交互+投影变换；
- 图像系统： 扫描光栅显示；

光栅显示系统



例子：怎样画直线?





理想数学直线： 没有宽度，无限长!
MATLAB模拟例子：$y = x, y = 2x, y = x/2$

硬拷贝技术

重要术语：

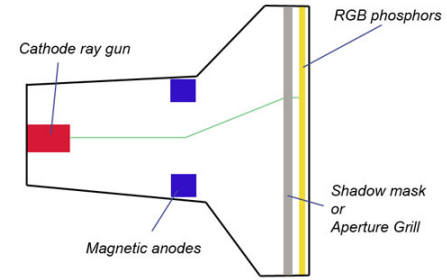
- 寻址能力addressability(每英寸的点个数 dpi), 点的尺寸,点间距离。
- DPI, 分辨率(resolution), PPI:
  分辨率小于DPI; 显示分辨率：VGA(640),XGA(1024),HDTV(1920);
  PPI: Windows 96 ppi,Apple: 72 ppi
- 彩色等级：bpp 1 bit:黑白； 8 bit: 256彩色;
  16 bit: 65536 彩色；18bit: 26万； 24bit 真彩$(16,777,216) > 1$千万。32bit?

常见设备：

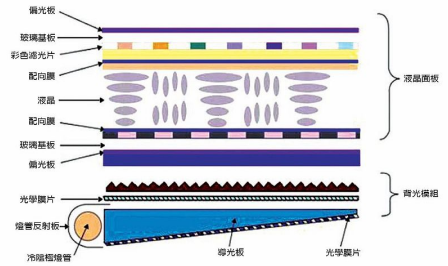
- 打印机：点阵；喷墨, 激光, 热传导;
- 绘图仪：
- 扫描仪，鼠标：

显示设备

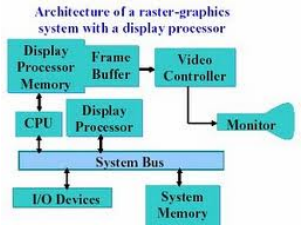
CRT:阴极射线管



LCD:液晶显示器(TFT: LED)



光栅显示系统



- CPU+视频控制器（video controller):
- 视频控制器：读帧内存(frame buffer) → 光栅扫描发生器 → 显示器
  常见扫描术语：交错或不交错60Hz； NTSC,PAL:
- 其他：视频查找表(彩色或灰度); 独立显示处理器GPU(扫描等硬件加速);

交互设备

- 定位设备： 输入板(手写),鼠标,跟踪球,游戏杆； 触摸板；
- 定值设备：键盘
- 选择设备：键盘+功能键;
- 输入设备：键盘或扫描仪;

3D设备？

图形学软件

常见软件包：

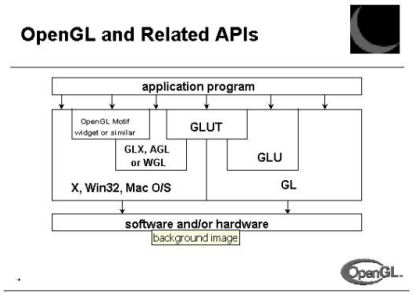
- 程序设计软件包: Openg GL, DirextX, VRML,Java2D,Java3D;
- 应用软件包：AutoCad, Flash, 3DsMax, Maya,...
- 其他通用软件: MATLAB, Mathematics,
  图像专用软件：

openg GL:

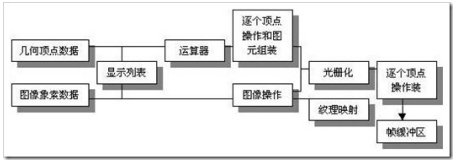
- 历史： 1992 1.0; 2.0,3.0
- 优点：OPEN： 工业标准；跨平台, 通用,(编程规范,文档详细);
- 缺点：入门难, 图形学知识+数学知识;

OpenGL 的结构

API接口：



Pipeline:绘制流程



OpenGL 编程

- 组成：核心库(gl函数 115),应用库(glu函数 43),工具库(glut函数 30）辅助库(aux 31)；
  专用库：Windows: wgl函数，Linux: glX函数；
- 语法：C,C++,Fortran,Java 不同版本。
- 函数与变量：函数名前缀：gl,glu,glut等;
  常量名前缀：*GL*:
  变量类型：*GLint*等；参数调用：支持不同类型参数;
- 编辑器：Windows *VC* + +,linux: 任何编辑器如gedit;
  头文件 *stdio.h*, *stdlib.h*, *math.h*
  openGL: *GL*/*gl.h*, *GL*/*glu.h* 或*GL*/*glut.h*.
- 编译： $gcc * .c - o * * * - lglut - lGL - lGLU - lm$

openg GL的运行机制：状态机制!!!
需要初始化glColor；需要打开或关闭某些功能(glEable);

简单openGL例子

openg GL的程序基本结构：

- 初始化：设置背景颜色；打开(光照，纹理);
- 显示设置：窗口，相机，变换;
- 建立模型(点，线,面);
- 主程序驱动：显示与交互;

代码：

$int\ main(intargc, char * *argv)\ \{$
$glutInit(\&argc, argv);$
$glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);$
$glutInitWindowSize(250, 250);$
$glutInitWindowPosition(100, 100);$
$glutCreateWindow("hello");$
$init();$
$glutDisplayFunc(display);$
$glutMainLoop();$
$return0; / * ANSICrequiresmaintoreturnint. * / \}$

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
Department of Mathematics, Beihang University

October 26, 2011

Chapter 2: 光栅图形学

OpenGL的基本画图

从建模到设备的显示流程 pipeline

- 模型坐标系：$p_m = (x_m, y_m, z_m)$
- 三维世界坐标$p_w = (x, y, z)$
- 观察与投射坐标：$p_p = Projection * p_w$
- 单位坐标(规范化) $p_n = p_p/||p_p||$
- 设备坐标：$p_d = p_n * (width, height)$

OpengGl 相关函数

- 投射矩阵$glMatrixMode(GL - PROJECTION)$, $glLoadIdentity()$
- 二维坐标$gluOrtho2D(xmin, xmax, ymin, ymax)$
- 画点坐标函数 $glVertex()$; 选项 2，3，4；i,s,f,d;v
- 画点函数$glBegin(), glEnd()$,常量$GL - POINTS$
- 画线函数$glBegin(), glEnd()$, 常量$GL - LINES$，$GL - LINES - STRIP$，$GL - LINES - LOOP$
- 其他：

直线扫描算法I：DDA

一般直线段：$y = kx + b$, 给定$(x_0, y_0), (x_1, y_1)$.

- 模拟直线扫描：(x_0, y_0),计算$\delta(x), \delta(y)$,向量扫描算法
- 数字直线扫描：(x_0, y_0),计算$x++, y = k * x + b$,坐标取整, 离散扫描算法;
- DDA：数字微分扫描 (x_0, y_0),计算$x++, y+ = k$,坐标取整。增量扫描算法;
  注意：$|k| \leq 1$.
- 主要运算问题：浮点运算慢，有误差和坐标取整慢。

直线扫描算法II：中点算法和Bresenham算法

一般直线段：$y = kx + b$, 给定$(x_0, y_0), (x_1, y_1)$.
定义:水平线函数$F(x, y) = ax + by + c$，区域划分为$F > 0, F = 0, F < 0$
扫描算法的基本运算：找下一个点可以用$F(x + 1, y + 0.5)$做选择！
设$0 < k < 1, d = F(x + 1, y + 0.5), bd = F(x + 1, y + 1)$.

- 中点算法：$(x_0, y_0), d(0)$,p判断$d > 0, y = y; d \le 0, y++$, 更新$d+ = \delta d$,循环。
  注：$d(0) = a + b/2$,可以用$2 * d$得到整数算法;
- Bresenham算法：$(x_0, y_0), bd(0)$,计算$bd > 0.5, y = y; bd \le 0.5, y++$,更新$bd+ = \delta bd$,循环。
  注：可以定义$e = bd - 0.5, e(0) = -0.5$,可以用$2 * e * \delta(x)$得到整数算法;

Remark

中点算法和*Bresenham*算法对于直线和整数半径的圆扫描结果是一样的。

寻找更快的扫描算法？ 多步扫描和并行扫描.

OpenGL的曲线画图

曲线的类别

- 直线→多项式曲线
  特别：二次曲线(conic section) $ax^2 + by^2 + cxy + dx + ey + f = 0$
- 样条曲线：若干多项式曲线段连接组成;
  特别：Bezier曲线，B样条曲线;
- 有理样条曲线（面) Nurbs,
- 任意参数曲线：（比如极坐标表示）;

OpengGl 画曲线方法

- GL库函数仅有Bezier曲线;
  GLU有B样条，Nurbs(有理B样条)，球面，锥面等;
- 利用直线逼近曲线：找到若干点，直线连接;
- 自己构造算法实现:

圆扫描算法I：直接扫描

一般方程：$y - y_c = \pm\sqrt{R^2 - (x - x_c^2}$.给定$(x_c, y_c), R$.

- 圆的8对称性：$(\pm x, \pm y), (\pm y, \pm x)$;
- 数字圆扫描：(x_0, y_0),计算$x++, y$,坐标取整，离散扫描算法;
  像素不均匀！
- 参数圆扫描：(x_0, y_0),计算$\theta+ = \delta\theta, x, y$,坐标取整;
  三角函数慢！
- 主要运算问题：浮点运算慢，有误差和坐标取整慢。

圆扫描算法II：中点算法即Bresenham算法

定义:水平线函数$F(x, y) = x^2 + y^2 - R^2$，区域划分为$F > 0, F = 0, F < 0$
扫描算法的基本运算：找下一个点可以用$F(x + 1, y - 0.5)$做选择！
设$0 < x < R/\sqrt{2}, d = F(x + 1, y - 0.5)$.设半径为整数，算法：

1. 给定园心和半径R，设$(x_0, y_0) = (0, R)$
2. 计算$d(0) = 5/4 - R$或$1 - R$;
3. $x++$, $d(k) < 0, y(k + 1) = y(k), d(k + 1) = d(k) + 2x(k + 1) + 1$;
   $d(k) > 0, y(k + 1) = y(k) + 1, d(k + 1) = d(k) + 2x(k + 1) + 1 - 2y(k + 1)$;
   特别：$x(k + 1) = x(k) + 1; y(k + 1) = y(k) - 1$;
4. 利用对称性得到其他七个点;
5. 循环直到$x \ge y$;

其他曲线和相关问题

其他曲线

- 椭圆：中点算法
- 二次曲线：参数扫描或中点扫描算法
- 多项式和样条曲线：

实现问题：

- 扫描算法和内存存储：线性地址$address(x+1,y+1) = address(x,y) + xwidth + 2$
- 扫描线的方向问题：可以规定一个方向但有特征的线段不行；
- 像素大小问题：

二维图形的opengGL画图

二维图形的表示

- 规则图形：曲线（多边形)边界；区域表示（水平集）
- 不规则图形：区域表示（颜色？）
- 特殊图形： bitmap 块状图像；字符；

opengGL

- 矩形：$glRect()$,圆?
- 多边形：$glBegin(GL-PLOYGON); glVertex(); glEnd();$ $GL-TRIANGLES, GL-TRIANGLE-STRIP, GL-TRIANGLE-FAN,$ $GL-QUAD, GL-QUAD-STRIP$
- 图像块：$glBitmap(), glDrawPixels()$
- 字符： $glutBitmapCharacter(); glutStrokeCharacter()$

多边形的扫描算法

给定多边形按逆时针顺序的顶点集$v_i = (x_i, y_i)$，边为$e_i = v_i v_{i+1}$；
基本的扫描算法：

- 求交：计算每条扫描线和多边形每条边的交点
- 排序：对交点按x坐标排序；
- 配对：确定交点间线段是否再多边形内部；
- 画像素：

活性边表AET的扫描算法：

- 初始化每条扫描线的起点边表
- for 每一条扫描线
  加入起点边表
- 排序插入AET：
- 对每条边配对，画像素：（增量扫描）；
- 删除AET中扫描完边；
- end

多边形的区域填充算法

给定多边形的边界的颜色和区域内一点（种子），假设区域是连通的，
常见区域填充算法：

- 连通性： 4连通，8连通；
- 递归算法：从种子出发，按连通性递归画像素；
- 边界填充boundary：边界颜色唯一确定；
- floodfill：内部颜色确定，用新颜色替换；

主要运算问题：递归太多，费时和内存；建议使用扫描线填充；

字符

字符的分类

- 标准字符：ASCII字符（128），UTF8，UTF16，GB18030，GB1832；
- 字样typeface： courier 字体font：
  大小10pt，$1pt = 1/72$英寸；斜体，黑体；
  有无衬线（末端加粗) serif,sans serif;
  宽度：固定monospace，可变proportional；
- 点阵字体和矢量字体：
  点阵字体是矩形内逼近画出；一般要压缩，适合整数倍放大；
  矢量字体即多边形，可以任意变换；但画得慢；

openGL的基本属性

openGL的基本属性

- 颜色和灰度：参见后面内容；
- 点的属性：大小和颜色；
  $glPointSize(); glColor*();$
- 线的属性：
  线宽$glLineWidth()$
  线型$glLineStripple(repeat, pattern)$，$pattern$是16位整数；
  $glEnable(GL - LINE - STRIPPLE)$;
  颜色插值：$glShadeModel(GL - SMOOTH)$;

openGL的二维图形属性

openGL的多边形属性（凸多边形）

- 显示模式：
  正面和反面：$glFrontFace(order)$; $GL - CCW$
  线框图：$glPloygonMode(face, displayMode)$;,$GL - FRONT$,
  $GL - POINT, GL - LINE, GL - FILL$
- 填充模式：$glpolygonStripple(fillpattern)$，$fillpattern$是$32 \times 32$矩阵；
  $glEnable(GL - PLOYGON - STRIPPLE)$;
- 纹理texture和颜色插值：参见后面内容；

openGL的字符属性：一般由字体决定；矢量字体可以设置线的属性和变换；

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！

OpenGL的几何变换和观察

从建模到设备的显示流程 pipeline

- 模型坐标系：$p_m = (x_m, y_m, z_m)$
- 三维世界坐标$p_w = Transform * p_m$
- 观察变换与投射变换：
  $p_p = Projection * View * p_w$
- 单位坐标(规范化)
  $p_n = p_p/||p_p||$
- 设备坐标：
  $p_d =$
  $p_n * (width, height)$

OpengGl 相关函数

- 矩阵模式： 建模观察$glMatrixMode(GL - MODELVIEW)$,
  投射$glMatrixMode(GL - PROJECTION)$
- 二维观察: 正交投影$gluOrtho2D(xwmin, xwmax, ywmin, ywmax)$
  窗口：可选$glViewport(xvmin, xvmax, yvmin, yvmax)$
- 三维观察：观察点：$gluLookat(x0, y0, z0, xr, yr, zr, Vx, Vy, Vz)$;
  正交投影
  $glOrtho(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$
  通用透视投影：$glFrustum(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$
  或$gluPerspective(theta, aspect, dnear, dfar)$.

几何变换

给定几何模型(顶点集), 坐标变换到世界坐标系。 基本变换:

- 平移： $glTranslate * (tx, ty, tz)$
- 旋转：$glRotate * (\theta, vx, vy, vz)$
- 缩放：$glScale * (sx, sy, sz)$

相关说明:

- 刚体变换：平移,旋转
- 仿射变换：包含缩放,反射，错切(shear)
- 变换的组合和逆变换得到变换群，不同变换群的不变量对应不同几何。
  欧氏几何，仿射几何，还有投射几何;

矩阵表示和齐次坐标

给定$P = (x, y, z)$,得到变换矩阵$P_w = M * P$.

- 基本变换：$P_w = M_1 * P + M_2$
- 齐次坐标：$P_h = (x, y, z, w) = (x/w, y/w, z/w, 1)$ 一般取$w = 1$
  $P_{wh} = M_h * P_h$
- 基本变换的复合矩阵：w坐标不变!

openGL中的矩阵:

- 使用矩阵：$glLoadMatrix * (m), glMultMatrix * (m), glLoadIdentity()$
- m是4×4矩阵，特别是列优先的16个值的数组;
  注：C语言为行优先排序。可以定义$m[16]$, 或者用$glLoadTransposeMatrix * (m), glMultTransposeMatrix * (m)$;

观察变换

给定观测点（照相机）的新坐标系，得到观察坐标$P_v = M_v * P_w$

- 坐标系的变换矩阵M_v是正交矩阵；由三个向量决定。
  一般uvn坐标系：N是观察平面法向量，v是向上向量，$u = v \times N$.
- OpenGL观察函数：$gluLookat(x0, y0, z0, xr, yr, zr, Vx, Vy, Vz)$;
  缺省值：$(0, 0, 0), (0, 0, -100), (0, 1, 0)$
- 一般的坐标变换包含平移变换。

openGL中的$MODEL - VIEW$矩阵:

- 建模观察$glMatrixMode(GL - MODELVIEW)$
  因为几何变换和观察变换可以得到相同的效果；（移动物体和移动相机)
- 矩阵复合运算的顺序：$P_v = M * N * L * P_w$
  对应于程序中调用的顺序：从左往右。
  两种解释(代码一样)：全局坐标系的变换，局部坐标系的变换;
- 可以构造不同的观察矩阵。

投影变换：平行投影

给定观察坐标$P_v = M_v * P_w$，$M_V = R \times T$.
怎样得到二维观测平面规范坐标$P_n = M_{3\times4}P_v$?
一般还是要得到有限的3维规范化观察体$[-1,1]^3$.
规范化便于剪裁，保留3维信息(深度信息)便于消隐。
平行投影:
有限观察体$(xwmin, xwmax, ywmin, ywmax, znear, zfar)$

- 正交投影: $(x,y,z) \to (x_p, y_p, z)$;
  规范化观察体: $M_{ortho,norm} = S(x,y,z) * T(x,y,z)$
- 斜平行投影: 投影与观测平面不垂直; 包含一个z方向错切变换，可以自己实现沿投影方向V_p的变换;
  $x_p = x + (z_v - z)V_x/V_z, y_p = y + (z_v - z)V_y/V_z, z = z$
  规范化观察体: $M_{oblique,norm} = M_{ortho,norm}Shear(z)$

openGL: 正交投影函数(包含裁剪)
$glOrtho(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$
默认值: $(-1,1,-1,1,-1,1)$; 特别有$gluOrtho2d()$;
无斜平行投影。

投影变换: 透视投影

设投影中心(或观测参考点)$P_r = (x_r, y_r, z_r)$,观察平面(投影平面)$z = z_p$,
则任意点P的投射点$P_p = P - (P - P_r)t$,计算有$t = (z_p - z)/(z_r - z)$.
得到$x_p = x - (x - x_r)t, y_p = y - (y - y_r)t, z = z$,不是线性变换！引入齐次坐标才有矩阵变换。

- 齐次透视矩阵变换: $(x_p, y_p, z_p, 1) \to (x_h, y_h, z_h, h)$
  $h = z_r - z$, $P_h = M_{pers}P$, 最后结果 $x_p = x_h/h, y_p = y_h/h$,包含z的规范化。 一般情形: 观察平面$z_p = 0$,投影中心在z轴上。
- 有限的视觉棱锥体pyramid of vision: 选择近平面和远平面;
  一般情形: $(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$
  对称情形:$(theta, aspect, dnear, dfar)$, 视场角$\theta$,纵横比$h/w$
- 规范化矩阵: $M_{normpers} = S(s_x, s_y)M_{pers}Shear(z)$
  特别对称视觉棱锥体$M_{normpers} = S(s_x, s_y)M_{pers}$

openGL: 透视投影函数(包含裁剪)
$gluPerspective(theta, aspect, dnear, dfar)$.
$glFrustum(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$

裁剪 clipping

给定规范化坐标系，去掉位于坐标系外面的几何物体，即裁剪。

- 点的裁剪: 直接坐标判断。
- 线段裁剪: 可能部分可见;
- 区域裁剪: 边界裁剪加上填充;
- 曲线裁剪: 类似直线，复杂判别方程;
- 字符裁剪: 可以选择全部，或部分。

openGL中的clipping
缺省: 规范化投射加裁剪: 6个平面$x = \pm1, y = \pm1, z = \pm1$.
可以指定任意平面$glClipPlane(id, paramaters)$; $glEnable(id)$; 参数为(a,b,c,d).
注: 任意平面裁剪在观察坐标系中实现，程序变慢！

2D线段裁剪算法

简单算法: 直接求交，乘法除法多，容易bug。 通常增加测试，减少求交。
Cohen-Sutherland算法:

- 矩形窗口编码: 九个区域;
- 每个线段端点判断; 全内，全外，可能交;
- 可能交情形再求交。

梁友栋-Barsky 算法

- 建立线段参数化模型;
- 每个线段端点判断; 全内，全外，相交;
- 仅仅相交情形再求交。

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
Department of Mathematics, Beihang University

November 25, 2011

Chapter 3: 几何造型

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;清华GEMS5；
- 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：

三维空间曲线与曲面

曲线与曲面的表示

- 非参数表示：显示表示和隐式表示
- 参数表示：
- 几何：切线，法线，曲率等；
- 参数表示的优点：几何变换，易于控制，计算容易；

曲线与曲面的几何

- 曲线的几何：局部坐标系；曲率，挠率；长度，弧长参数；
- 曲面的几何：基本形式；曲率；面积，法向量；

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：$GLUquadricObj * sphere1;$
  $sphere1 = gluNewQuadric();$
  $gluQuadricDrawStyle(sphere1, GLU - LINE);;$
  $gluSphere(sphere1, r, nlog, nlat);$
  $**gluDeleteQuadric(sphere1);$

样条曲线与曲面

样条：通过一组给定点集而生成的平滑的blobby object(spline curve或surface), 给定点称为控制点 control points。
样条曲线：数学上一般用分段三次曲线；计算机图形学：多项式曲线段连接而成；
相关概念：

- 插值样条：线性插值，二次插值（抛物线）；Hermite插值（三次）
- 逼近样条：Bezier曲线等；或称拟合。
- 光顺fairing: 二阶几何连续性；曲率变化小；无多余拐点等；
- 参数化：对控制点取参数的方法： 均匀；弧长；
- 参数连续和几何连续：
  参数连续：交点处导向量相同；
  几何连续：交点处导向量成比例；

一般曲线或曲面的表示

一般光滑曲线或曲面$P(t), P(u, v) \in R^3$

- 函数空间与逼近：
  Weierstrass 定理：光滑函数$f(x)$存在多项式函数逼近。
  即$f(x) \sim \sum a_i P_i(x)$
  泛函分析语言：光滑函数空间存在一组多项式函数组成的(可数个)基.三角函数基(傅立叶分析)
  特别存在有限支撑集的小波基。分段多项式函数或B样条是其中的特殊基。
- 多项式逼近 vs 样条逼近：
  Runge现象：高阶多项式的逼近在端点处有较大的振荡误差。类似Gibbs现象。
  样条逼近可以用较低的分段多项式逼近，容意控制，误差并不大。
- 常见样条函数：cubic spline：三次B样条，三次Bezier样条
  实用样条：有理样条(可以表示二次曲线), 非均匀有理样条NURBS(工业标准)。

Bernstein基

Theorem (Bernstein基)

$B_{i,n}(t) = C_n^i t^i (1-t)^{n-i}$是$n$次多项式空间的一组基。

性质:

- 恒正性;
- 端点性;
- 单位分解性 $\sum B_{i,n}(t) = 1$
- 对称性:
- 递推性: $B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t)$
- 微分积分: $\int_0^1 B_{i,n}(t) = 1/(n+1)$, 最大值在$t = i/n$处。

Bezier曲线和Bernstein基

Definition (Bezier曲线)

$P(t) = \sum_{i=0}^n P_i BZ_{i,n}(t)$是由$n+1$个控制点得到的$n$次*Bezier*曲线。 特别$P_i$组成一个特征"多边形".

性质:

- 端点性; Bezier曲线过两个控制点。
- 切向量: 端点切向量与特征"多边形"的边重合;
- 单位分解性 $\sum BZ_{i,n}(t) = 1$
- 对称性:
- 递推性: 可以递推计算;
- 凸包性: 曲线位于特征"多边形"的凸包内;
- 变差最小性: 任意直线和曲线的交点数少于和特征"多边形"的交点数;
- 仿射不变性。

Bezier曲线的例子和算法

递推得到

- 线性:
- 二次:
- 三次:

de Casteliau算法:
$P_i^k = (1-t)P_i^{k-1} + tP_{i+1}^{k-1}$.

B样条曲线和B样条基

Definition (B样条曲线)

$P(t) = \sum_{i=0}^n P_i B_{i,d}(t)$是由$n+1$个控制点得到的$d-1$次*B*样条曲线。 特别$P_i$组成一个特征"多边形".$B_{i,d}(t)$是定义在$[t_i, t_{i+d}]$上的$d-1$次多项式。

性质:

- B样条基:$B_{i,1}(t)$是Box函数;
  Cox deboor 递推公式:
  $B_{i,d}(t) =$
  $(t-t_i)/(t_{i+d-1}-t_i)B_{i,d-1}(t) + (t_{i+d}-t)/(t_{i+d}-t_{i+1})B_{i+1,d-1}(t)$
- B样条基性质: 局部支撑集, 单位分解;
- 节点向量:$[t_0, t_1, \ldots, t_n + d]$ 是支撑区间的并;
  分类:均匀, 准均匀, 不均匀, 对应与不同类别的样条曲线;
- B样条曲线的性质: 局部性, 光滑性$(d-1-rep)$,导数计算;
- B样条曲线的几何性质:变差最小性, 凸包性, 仿射不变性;
- B样条曲线的灵活控制性: 特征"多边形"是直线得到直线; 多重节

B样条曲线的例子I: 均匀与准均匀样条

均匀样条: 节点向量: $[t_0, t_1, \ldots, t_n + d]$ 为均匀分布，通常记为$[0, 1, 2, \ldots, n + d]$
对应有均匀周期B样条曲线: B样条基是平移得到的。

- $B_{i,d}(t) = B_{i+1,d}(t + 1)$
- 基数样条 cardinal spline $B_k = B_{k-1} * B_0$,$B_0 = B_{0,1}$是BOX函数或$[0, 1]$上特征函数。
  验证: B_1是hat函数。类似与 deboor 递推。
- 均匀二次B样条: $B_{0,3}(t) = 1/2t^2, 0 \le t < 1$
  $B_{0,3}(t) = 1/2t(2 - t) + 1/2(t - 1)(3 - t), 1 \le t < 2$
  $B_{0,3}(t) = 1/2(3 - t)^2, 2 \le t < 3$

准均匀样条: 仅仅端点处有重复节点d次，其他是均匀的;
例子: 节点向量为$[0, 0, \ldots, 0, 1, \ldots, 1]$, 端点重复数为$d = n + 1$时得到Bezier样条。
特别: 准均匀二次B样条:
一般非均匀样条: Non Uniform B 样条。

B样条曲线的例子II: 有理与非有理样条

- 有理B样条曲线: $R(t) = \frac{\sum_{i=0}^{n} w_i P_i Bi,d(t)}{\sum_{i=0}^{n} w_i Bi,d(t)}$,其中$w_i$是加权。
- 例子: 二次曲线 $d = 3$,节点向量$[0, 0, 0, 1, 1, 1]$,加权$w_0 = w_2 = 1, w_1 = r/(1 - r)$;
- 有理样条的齐次表示:(x, y, z, w) 可以看成四维非有理样条在三维空间的投影。
  控制点(wx, wy, wz, w).

不同样条表示可以互相转换，常用Bezier表示(可以用矩阵)!
de Boor算法:

OPENGL 中的曲线与曲面

样条曲线和曲面的显示:

- 多边形逼近显示: 计算参数值，连接多边形;
  多项式的求值: Horner 法则，向前差分法,
- 控制点逼近: 细分法(bezier样条)
  等分曲线，递归得到新控制点，直到距离充分小，连接控制点!

OPENGL 中的曲线与曲面:

- GL库: bezier样条的求值器; (一维和二维)
- GLU库: Nurbs包含一般B样条等。通过参数调用求值器画图;
- GLU库: 曲面修剪函数;

bezier 曲线与曲面

bezier 样条曲线:
$glMap1 * (GL - MAP1 - VERTEX - 3, t0, t1, stride, npts, *ctrlpoints)$;
$glEnable(GL - MAP1 - VERTEX - 3)$;

- GL-MAP1-VERTEX-3: 坐标维数; 可用齐次坐标(4).
- $t0, t1$参数范围;
- ctrlpoints 控制点数组，大小npts，间隔: stride;
- 画图: 画点$glEvalCoord1 * (t)$;
  或者$glMapGrid1 * (n, t1, t2), glEvalMesh1(mode, n1, n2)$;

bezier 样条曲面:
$glMap2 * (GL - MAP2 - VERTEX -$
$3, u0, u1, ustride, nupts, v0, v1, vstride, nvpts, *ctrlpoints)$;
$glEnable(GL - MAP2 - VERTEX - 3)$;
$glEvalCoord2 * (u, v)$
$glMapGrid2 * (), glEvalMesh2()$;

B样条曲线与曲面：NURBS

B样条曲线:
$GLUnurbsObj * mycurve$
$mycurve = gluNewNurbsRenderer();$
$gluBeginCurve(mycurve);$
$gluNurbsCurve(mycurve, nknots, *knots, stride, *ctrlpts, deg, GL - MAP1 - VERTEX - 3);$
$gluEndCurve(mycurve);$
$glEnable(GL - MAP1 - VERTEX - 3);$

- 节点向量：knots，大小 nknots；
- deg：样条参数d,多项式$d-1$次；
- GL-MAP1-VERTEX-3：求值器得到顶点；
  有理样条：GL-MAP1-VERTEX-4

B样条曲面：类似，
附加：法向量 $glEnable(GL - AUTO - NORMAL);$

NURBS的其他函数

- 属性函数：$gluNurbsProperty(theNurb, property, value)$
  9种属性；
- 事件函数：$gluNurbsCallback(theNurb, GLU - ERROR, nurbsError);$
- 释放内存：$gluDeletaNurbsRenderer(theNurb);$

曲面修剪：
$gluBeginTrim(theNurb);$
$gluPwlCurve(theNurb, 5, *edgePt, 2, GLU - MAP1 - TRIM_2);$
$gluEndTrim(theNurb);$

三角网格化

实体模型的表示：空间分解表示，构造表示，边界表示（曲面表示）
离散曲面的最常用表示方法：三角网格。

- 优点：容易得到，结构简单，计算快(光照等)；缺点：不精确(相对于NURBS)。
- 简单数据结构：V, F 参见Wavefront OBJ 文件 或 BYU曲面
- 半边数据结构：双向链接边表 E 包含顶点，面，邻近边，反向边；
- 网格处理：细分和简化；调整和光顺；

其他非几何造型

几何造型：规则刚性对象，CAD,CAM.
其他造型：不规则对象，自然景观，气体，液体 及柔性物体。 动画，虚拟现实等。

- 分形模型：树，山，云等 $\rightarrow$ 分形几何
- 粒子系统：模糊对象：雨，雪等；$\rightarrow$ 随机过程
- 物理模型：非刚性物体的运动 $\rightarrow$ 力学
- 可视化数据：颜色，等高线，向量场等；

分形 fractal geometry

历史：1967(Science) Mandelbrot：英国的海岸线有多长?
依赖于测量单位!!!

Definition (分形*)

集合具有无限的细节和局部与整体的自相似性称为分形。 一般它的分形维数大于拓扑维数，用迭代方法得到。

- 分形的生成：迭代函数$F : R^3 \to R^3$, 初始形状或点集P_0,
  得到集合$F(P_0), F^2(P_0), \ldots, F^n(P_0), \ldots$.
- 分形的分类:
  自相似: (存在收缩系数s)+统计自相似;
  自仿射：存在收缩系数s_x, s_y, s_z+
  不变集：非线性变换的不变集：如复平分变换 Julia集，Mandelbrot集;
- 分形的维数：描述分形的细节变化的数字D

分形的维数与自相似分形

拓扑维数理论：标准球形邻域覆盖得到欧氏维数，非标准形状得到hausdoff 维数。

EXAMPLE (Koch曲线)

代替三角形的每一边用四条等分线段，$D = \ln 4/\ln 3 = 1.2619$

- 欧氏维数与自相似维数D: $ns^D = 1, D = \ln n/\ln(1/s)$.
  线段L,二等分，系数$s = 1/2$， 方程:$ns^1 = 1$
  正方形P,四等分，系数$s = 1/2$，方程:$ns^2 = 1$
- 一般自相似分形维数: $\sum s_k^D = 1$
  通常用方框覆盖数，收缩系数等来估计。
- 拓扑维数：可以用若干个参数决定的对象(点，曲线，曲面);
  通常：分形维数大于拓扑维数。 例子：Peano曲线$D = 2$,

常见应用: 树。需要加入随机变化。
可以得到山，云等。参见IFS迭代函数系统。

自平分分形

起源：复平面的解析映射的迭代集 Fatou，Julia 1918，但没有图!

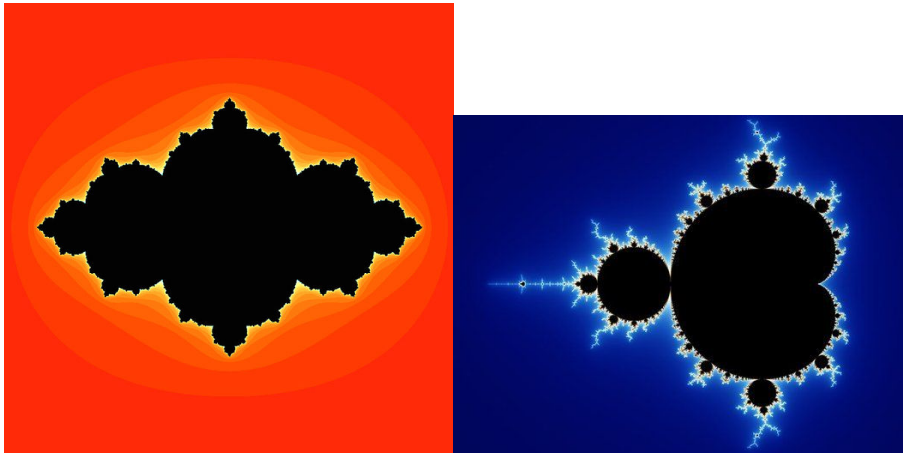
EXAMPLE (平方映射)

$f(z) = z^2$的迭代结果: 两个吸引子之间一个边界:单位园。 概念: 不动点，周期点，吸引子。

一般考察:$f(z) = z^2 + c$

- Julia 集是两个吸引子之间一个边界。
  定义：f的周期点集的闭包，满足$(f^p)'(z) > 1$,p是周期。
  例子：$c = 0.1 - 0.1i \to$ 类似园
  例子：$c = 1 - 0.05i \to$ 周期2轨道;
- Mandelbrot 集： 固定$z_0 = 0$, 使迭代有界的c值构成的集合称为Mandelbrot 集。
  或者 使得Julia 集连通的c值构成的集合称为Mandelbrot 集。

Mandelbrot 集和Julia 集



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
Department of Mathematics, Beihang University

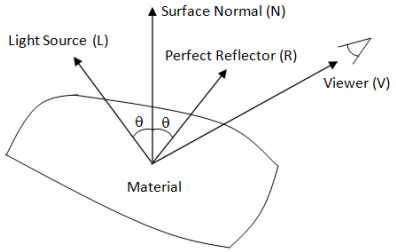
December 14, 2011

Chapter 4: 真实感图形学

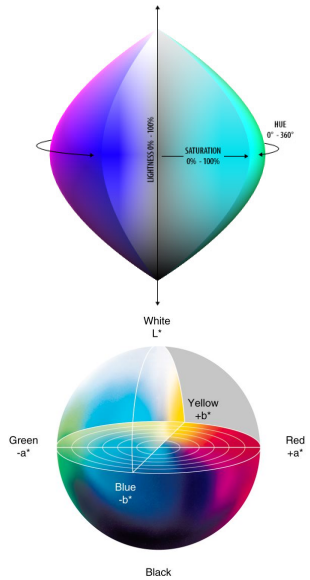
真实感图形学的基础

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。
  消除隐藏面(画家算法，Z缓冲区算法)
- 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，光透射模型；局部和整体光照模型，
- 感觉逼真：颜色和纹理的心理学基础；
  颜色模型，纹理映射；



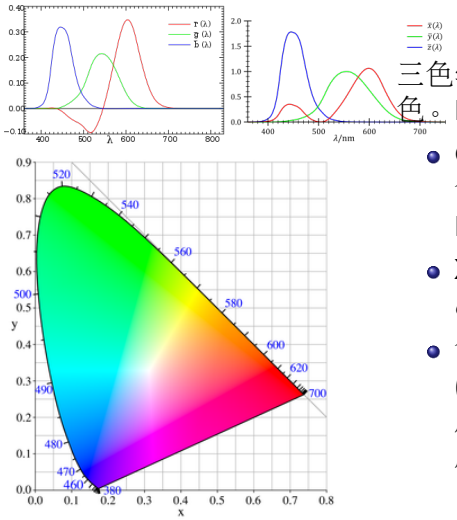
颜色的视觉基础



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。

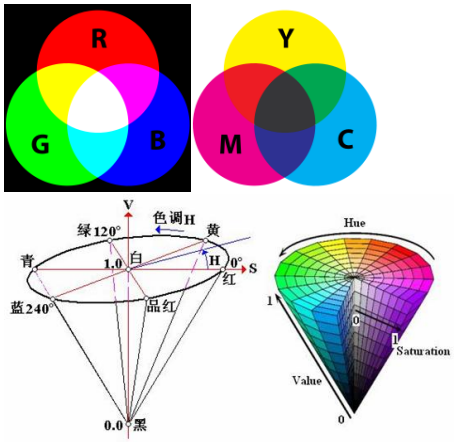
颜色空间的三色模型



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.
  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);

OPENGL中的颜色

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
$glEnable(GL-BLEND)$;
$glBlendFunc(sdactor, dfactor)$;

OPENGL中的交互技术

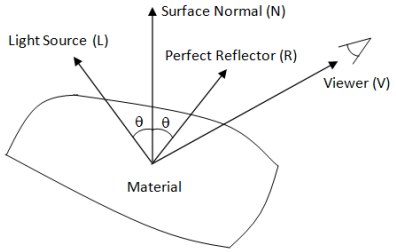
GLUT：对各种输入设备指定回调函数；通过重画得到交互效果。
glutDisplayFunc (dispFcn);
****
glutReshapeFunc (reshapeFcn);
glutMainLoop ( );

- 键盘： $glutKeyboardFunc(enlargeSquare)$;
  特殊功能键： $glutSpecialFunc(reduceSquare)$;
- 光标： $glutMouseFunc(fillSquare)$;
- 菜单：
  glutCreateMenu (mainMenu);
  glutAddMenuEntry ("Solid-Color Fill", 1);
  glutAddMenuEntry ("Color-Interpolation Fill", 2);
  glutAddSubMenu ("Color", subMenu);
  glutAttachMenu (GLUT-RIGHT-BUTTON);

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 $I_d = k_d I_i \cos\theta, \cos\theta = N\dot{L}$
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.
  $n = 1 \to 1000$
- 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$H = (L + V)/|(L + V)|$,
  $\cos\phi = V \cdot R \sim N \cdot H$

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。

OPENGGL 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);
******
glLight*(i,f,v)(GL-LIGHT0, GL-POSITION, light-position);
glEnable(GL-LIGHT0);
******
glEnable(GL-LIGHTING);

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，其光照由顶点的简单光照模型直接计算得到。

- 恒定强度: 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;
- Gouraud： 双线性光照插值;
  计算：顶点法向量N，光照I，依y, x坐标插值I;
  优缺点: 简单；可与扫描算法合并，存在Mach带;
- 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；光照N，
  优缺点: 更准确；太慢，需要加速算法(插值N);

整体模型I: 光线追踪

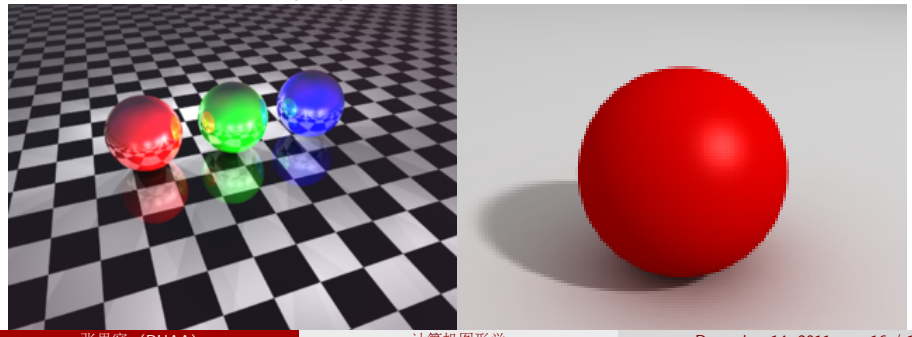
Ray tracing：对投射平面上每一个像素通过光线(逆)追踪，得到光照I的方法。

- 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;

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。



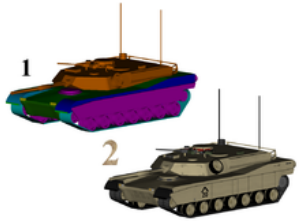
纹理与纹理映射

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*();



EXERCISE ONE

*作业和上机作业在11月1日前交,带***可不做。*

1. 说明以下不同领域的主要内容, 简述其关系和区别：计算机图形学，计算机绘图，计算机视觉，计算几何；计算机辅助几何设计， 计算机虚拟现实，数字图像处理，计算机艺术。

2. 给出交互式计算机图形系统的基本框架图, 说明一个典型的openGL程序包含的对应功能块。

3. 分别用中点扫描和Bresenham扫描算法给出线段$(1,0)$到$(5,3)$的每一点的坐标和判别函数值。

4. 计算沿给定方向(a,b,c),伸缩因子为s的三维几何变换矩阵。

5. 设观察点在$(2,4,-1)$,照相机在$(4,2,1)$,向上向量是$(2,2,-1)$,计算其对应的观察变换矩阵。 ***可以用openGL程序验证。

6. 设视角为120度，纵横比为1，近远裁剪距离为$3,10$的对称透视体，计算其投影变换矩阵。 ***可以用openGL程序验证。

上机作业：上交源程序和一个readme.txt文件到邮箱

要求：说明编译命令和执行方法. 可以组成2人小组一起完成作业

1. 画点和球.

   A:编程画任一大小的点和球面。

   B:考察反走样技术的差别：

   C：考察填充方法的差异：

2. 画字符和裁剪

   A: 编程写出一句话。*** 汉字更好。

   B: 通过平移，放大等几何变换，看裁剪：

   C: 改变视点看透视变换；

3. *** 三维相交视图

   A: 编程画一个大三角形和一个正方体；

   B: 通过平移，放大等几何变换，观察三角形和正方体的交点；

   C: 发现交点最多的多边形形状。

EXERCISE TWO

作业和上机作业在*12*月*13*日前交,带***可不做。

1. 说明几何模型的主要种类，实体模型的不同构造方法，各给出一个例子。特别给出正方体的一个边界表示(至少包含顶点，面表，边表).

2. 给出三次的Bernstein函数基，给定任一个三次多项式，计算其表示为Bernstein函数基线性组合的系数。

   (可以写成基的变换矩阵，即Bezier表示矩阵) 原始基为$1, t, t^2, t^3$.

3. 计算基数样条的卷积公式: $B_k = B_{k-1} * B_0$,$B_0 = B_{0,1}$是BOX函数或$[0, 1]$上特征函数。B_1是hat函数。 计算B_2.

   ***验证其满足Deboor递推公式。

4. 准均匀B样条：节点向量为$[0, 0, 1, 1]$, 验证其为一次Bezier样条。

   *** 节点向量为$[0, 0, 0$fi1fi$1, 1]$, 验证其为二次Bezier样条。

5. ***构造圆锥二次曲线：给定有理B样条$d = 3$,节点向量$[0, 0, 0, 1, 1, 1]$,加权$w_0 = w_2 = 1, w_1 = \cos\theta$; 设控制点为$P_0 = (0, 1), P_1 = (1, 1), P_2 = (1, 0)$. 说明曲线为一个四分之一圆。

6. 证明：Koch三角形的边长趋于无穷大，面积趋于一个常数。

上机作业：上交源程序和一个readme.txt文件到邮箱

要求：说明编译命令和执行方法. 可以组成2人小组一起完成作业

1. 画多面体。用OPENGL画一个四面体。建议使用顶点数组。

2. 画样条曲线，以正方形的四个顶点为控制点； (可以画在一个窗口，适当平移下面的曲线).

   A: 画一个Bezier曲线;

   B: 设节点向量$[0, 0, 0, 1, 1, 1]$,画一个B样条曲线;

   C: ***如何得到以上A,B情形的封闭曲线?

3. 画圆锥曲线，给定有理B样条$d = 3$,节点向量$[0, 0, 0, 1, 1, 1]$,加权$w_0 = w_2 = 1, w_1 = r/(1 - r)$; 设控制点为$P_0 = (0, 1), P_1 = (1, 1), P_2 = (1, 0)$. (注：用四维齐次坐标)

   A: 画$r = 0.5, r = 0$对应两条曲线;

   B: 画$r = 0.8, 0.2$对应两条曲线;

   C: 画四分之一圆。验证习题5.

4. *** 画出$f(z) = z^2 + 1$对应的分形。

EXERCISE THREE

作业和上机作业在*2012/1/15*前交.纸版送图书馆西配楼*501*

1. 给出常见七种颜色的RGB坐标表示和CMY的坐标表示;
2. 镜面反射Phong模型中半角向量$H = (L + V)/|(L + V)|$,证明半角向量与法向量的夹角是反射光R与观察方向V的夹角的一半。 设所有向量都是共平面的单位向量。
3. 给出简单光线追踪的伪代码算法过程。
4. 设给定纹理图像$F(s, t)$在$[0, 1]\times[0, 1]$上，给定圆柱面方程$H(u, v) = (\cos 2\pi u, \sin 2\pi u, v)$，构造一个映射，给出圆柱面每一点$(x, y, z)$对应的纹理坐标。

   ***如果是球面呢?

上机作业：上交源程序和一个readme.txt文件到邮箱

要求：说明编译命令和执行方法. 可以组成2人小组一起完成作业

画一个正方体;

1. 给不同面涂上不同颜色;
2. 给一个面上方加点光源；比较两种shademodel：$GL-FLAT, GL-SMOOTH$;
3. 给另一个面上加一个方向光源，取材料的SHINENESS(镜面反射的n)不同系数，观察其对图像的影响;
4. 利用键盘函数，给出一个键c使得程序可以自动结束。