**2022-Java3D**

*不考编程*

*不考类方法*

*不考不常用类*

一、填空题 5x2’

二、选择题 5x2’

三、判断题 5x2’

四、常用类问答题 20x1’

这个类是干什么的？

某个操作需要用到哪个类？

这个类包含哪些类？

这个类可以加到哪些类里去？

五、简答题 2x10’

六、计算题 2x15’

# 第一章

**线架模型**(Wire Frame)：以线段、圆弧和一些简单的曲线来表示一个三维模型。

**Brep边界面模型**(Boundary Representation)：将一个封闭的几何体模型所使用的多边形面称为该几何体的Brep边界面。所有的多面体模型都是一种Brep边界面模型。

**非均匀有理B样条曲面**Non Uniform Rational B-Spline(NURBS)

曲面几何模型的特点是光滑的曲面表示，现在多数曲面几何模型以非均匀有理B样条曲面为基础

在Brep边界面表示中最常用的数据结构：翼边数据结构、半边数据结构

**Solid实体**

Solid实体几何模型主要通过组成该几何体的边界面所形成的半空间来表示一个物体。物体内部的所有属性都是相同的。

**CSG构造实体几何(**Constructive Solid Geometry**)**

通过简单实体(如立方体﹑圆柱体、球体﹑圆锥体、扫描表示法产生的体等)之间的正则布尔运算生成比较复杂的体。其用一种二叉树结构来表示相关实体以及相关的并、交、差布尔运算操作。

CSG树只反映了物体的构造过程与方式，并不反映实体的面、边、顶点等几何信息及其之间关系。因此，这种表示又称为实体的隐式模型或过程模型

**VOXEL体素模型(**Volume Element**)**

三维图形显示的最小单元

规则体素空间是将一个立方体分别沿x、y、z轴进行等间距均匀分割所形成的，每一个体素都是一个小立方体。小立方体的中心点称为立方体的格点。

体素与体素之间的连接关系有3种：6连通、18连通、26连通

**DEXEL深度元素模型(**Depth Element**)**

DEXEL模型就是用一射线与一个几何体的Brep边界面模型求交，两交点之间属于几何体内部的这段线段称为DEXEL。对于DEXEL模型,通常用一组群到三组群的射线组与几何体的交点来表示一个几何体。

**优点**：只需要存储射线组群的交点坐标，可以大大压缩存储空间，并且布尔运算是DEXEL线段间的线性运算，运算速度快、精度高。

**缺点**：失去了每个体素的所拥有的属性信息。

因此DEXEL模型和VOXEL模型之间可相互转换。

# 第二章

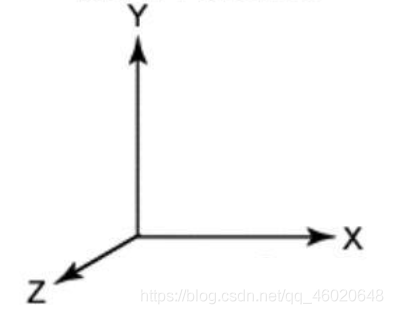
## Java3D高分辨率大尺度坐标系

8个字节，即**256**个二进制位的定点数来表示一个数

## 默认情况下Java3D的坐标系

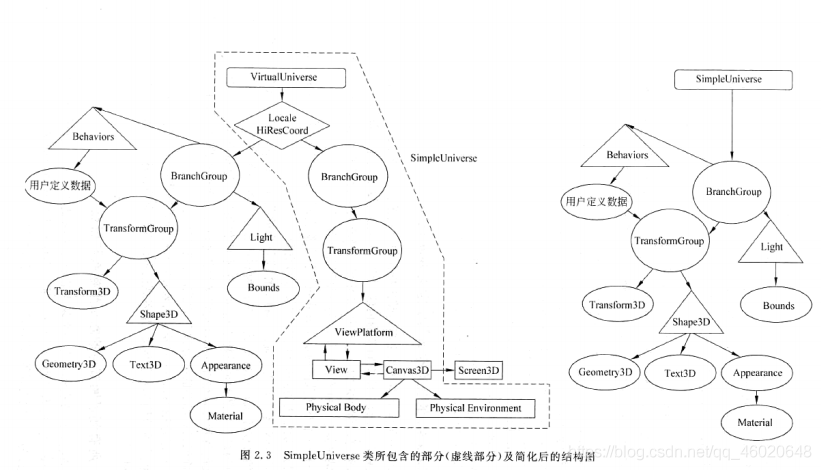
默认情况下，Java 3D坐标系采用右手坐标系统，坐标系原点在显示器的中心，x轴水平向右，y轴垂直向上，z轴指向观察者。默认情况下，坐标单位为米。

如图所示：



## P34页的图

包含谁，谁能加到谁里面？



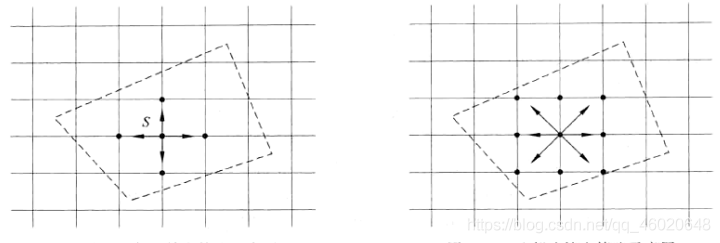
# 第三章

## 多边形可视化填充算法

\*种子填色算法

种子填色算法首先在要填充的多边形区域内选择一个填色种子点( Seed Point)，然后以该种子点为基础，通过与多边形边界的像素点相比较进行填色。

种子填色算法主要有四邻法（如左图）和八邻法（如右图）：



四邻法所⾛的填充路线为：上、下、左、右。在填充时要判断当前填充象素点与

边界象素点之间的关系。如果要填充的象素点在多边形的边界之内，则进⾏填充；

如果要填充的象素点在多边形的边界之外，则不填充；如果要填充的象素点正好

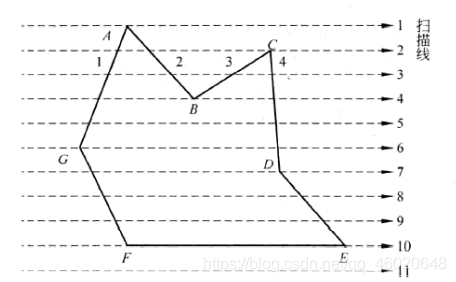
在多边形的边界上，则保留边界象素点颜⾊。

⼋邻法所⾛的填充路线为：上、下、左、右、左上、左下、右上、右下。在填充

时要判断当前填充象素点与边界象素点之间的关系

\*扫描线填色算法(Scan-Line Filling)

每一条扫描线从上向下以一个像素单位为间隔逐次向下扫描。对每一条扫描线与多边形求交。将交点按x值从小到大排序。用**奇偶计数法**判断多边形内部的点对。



出现点重合时采取的处理方法：

1首先按照多边形顶点顺时针方向标明：A、B、C、D、E、F、G

2扫描线6：扫描到3个顶点，G点处两点重合，按奇偶计数法会出现错误。于是将入线FG的G点的y轴坐标向下移动1个像素单位（我理解为逆时针方向）。

3扫描线7：扫描到3个顶点，D点处两点重合，按奇偶计数法会出现错误。于是将入线CD的D点的y轴坐标向上移动1个像素单位（我理解为逆时针方向）。

如果判断出多边形中有水平边，则将该水平边首先画出，不用求该边与扫描线的交点。

## Java语言的多维数组与计算机几何数据定义

线性数据点与二维数组表示

P110

平面型数据、三维数组

P110

## Java语言多维数组与Java3D一维数组之间的转换

P112

# 第四章

## 三次Bezier曲线及其性质

1）端点性质

C(0)=P0, C(1)=Pn

该性质说明一段Bezier曲线通过其首与末控制顶点

2）端点的切线矢量

3）Bezier曲线的端点曲率

4）对称性

对于一段Bezier曲线及其控制顶点，若保持原全部控制顶点的位置不变，只把控制顶点次序编号颠倒过来，则新生成的这段Bezier曲线形状不变，但参数的增长方向相反。

5）凸包性

一段Bezier曲线严格地位于其控制多边形形成的凸包之内。

6）变差缩减性

对于一段平面Bezier曲线C(u)，平面内任意一条直线与其交点的个数不多于该直线与其控制多边形的交点个数。

## Bezier曲线、曲面拼接的连续性

（1）C0连续：0阶参数连续，第一段曲线的终点与第二段曲线的起点位置重合。

（2）C1连续：一阶参数连续，两相拼接的曲线在拼接点处重合，拼接处有相同的一阶导数。

（3）C2连续：二阶参数连续，两相拼接的曲线在拼接点处重合，拼接处有相同的一阶导数和二阶导数。

（4）G0连续：0阶几何连续，第一段曲线的终点与第二段曲线的起点位置重合，称为G0连续，同时也是C0连续。

（5）G1连续：1阶几何连续，两相拼接的曲线在拼接点处重合，切线方向相同，但大小不等。

（6）G2连续：2阶几何连续，两相拼接的曲线段在拼接点处重合，在拼接点处C0连续、C1连续，在拼接点处二阶导数方向相同，但大小不等。

曲线、曲面的G0（或C0）拼接可保证曲线、曲面在拼接点处是连接在一起的。

曲线、曲面的G1（或C1）拼接可保证曲线、曲面在拼接点处是光顺的。

## Bezier曲面的性质-F

（1）端点性质

Bezier曲面的4个角点与控制多边形网格的4个角控制顶点重合。

（2）边界线

Bezier曲面S(u,v)的4条边界线S(0,v)、S(1,v)、S(u,0)、S(u,1)分别由对应的控制网格的4条边界控制多边形形成。

（3）曲面端点的切平面

由通过该端点的两个控制多边形的边所形成的平面和Bezier曲面在该端点的切平面重合。例如，对于曲面角点S(0,0)，切平面为控制顶点P0，0、P0，1、P1，0所形成的平面。

（4）凸包性

Bezier曲面位于其控制网格所形成的凸包之内。

（5）变差递减性

空间任意条直线与Bezier曲面交点的个数不多于该直线与其控制多边形网格的交点个数。

# 第五章

## 隐藏线、隐藏面消除算法

通常面向视点的所有边都是可见的，将这些边画出来。被前面的可见面隐藏的所有边都是不可见的，不用将这些画出来，或者画成虚线形式。以这种方式显示 的形体和用眼睛看到的三维物品相同。将这种消除所有隐藏边的算法称为隐藏线消除算法。

当给定视点与观察方向后，对视点来讲，并不是所有的面都可见，可见面是由视点与观察方向及面的法向量决定。只需显示可见的面，无须显示隐藏的面，这种算法叫做隐藏面消除算法。

隐藏面的消除与隐藏线消除有时是相互关联的。由隐藏线所围成的面是隐藏面，两个隐藏面所共有的线为隐藏线。

## Z Buffer隐藏面消除算法

在显示器的像素点阵建立两种缓冲存储单元阵列：Z Buffer 和C Buffer阵列。

从视点出发通过每个像素点发出一条射线，Z Buffer阵列存储距离视点最近的多边形与射线交点z坐标，C Buffer阵列存储对应的交点所在面颜色。

## 区域排序消隐算法

区域排序消隐算法就是在图像空间中，将组成几何体的所有多边形按照z坐标深度值从小到大排序，然后用前面的可见多边形去切割后面的多边形，将后面的多边形切割为可见与不可见两部分，最后将所有可见多边形显示出来，就为几何体消隐后的图像。

## 最小包围矩形、最小包围圆

用对边分别平行于两个坐标轴的边形成一个最小包围矩形

求法如下：求所有几何数据点的最小x、y值与最大x、y值，则由这两点定义的平行于两坐标轴的矩形就是最小包围矩形

## 最小包围盒、最小包围球

最小包围盒指用三对分别平行于坐标平面的最小六面体（长方体）包围一个三维几何体。最小包围盒是指该六面体正好包围该三维几何体，在所有包围该三维几何体的六面体中，它是最小的。

## 同一平面内点与多边形的包含判定算法-F

（1）凸多边形的叉积判断法

首先假定按逆时针方向计算的叉积为正，则按顺时针⽅向计算的叉积为负。从给定的一个点向多边形的各个顶点做向量。然后，按顶点顺序逐个计算每相邻两向量之间的叉积。如果所有的叉积符号相同，说明点在多边形内；反之，则在多边形外。

补充：

叉积的一个重要性质是可以通过它的符号判断两矢量相互之间的顺逆时针关系：

若 P × Q > 0 , 则P在Q的顺时针方向

若 P × Q < 0 , 则P在Q的逆时针方向

若 P × Q = 0 , 则P与Q共线，但可能同向也可能反向）

（2）夹角之和检验法

首先将要判断的点与多边形的各个顶点依次连接形成一系列向量，然后按照顶点 顺序求相邻两向量之间的夹角。假定按逆时针方向计算的角度为正，按顺时针⽅ 向计算的角度为负。如果所有的夹角符号相同，说明点在多边形内；反之，则在多边形外。

（3）交点计数检验法

从要判断的点向右做一条射线，求出射线与多边形交点的个数。如果交点的个数是奇数，则点在多边形内；反之，则在多边形外。

## 局部光照明计算模型

公式、参数 P238

**环境光部分计算**

Ed表示P点所反射的环境光的强度

R为P点的漫反射系数（介于0和1之间）

Id为照射在P点的入射环境光的强度

**直射光强度计算**

Es表示直射光反射的总光强

Is为直射光的入射光强

R为P点的漫反射系数（介于0和1之间）

W为P点的镜面反射系数，介于0和1之间

i为直射光在P点与法向量的夹角

n为控制高光聚散的系数，它与P点的材料有关

**透射光强度计算**

Et为从几何体内部经几何体表面P点处透射出的光强度

T为P点的透射系数（取值介于0到1之间）

Ib为从物体内部到达几何体表面P点处的入射光强度

**P点处总的光强度计算**

## Gouraud算法和Phong算法及区别-F

Gouraud算法：⾸先计算各个平⾯多边形的法向量，由该法向量确定各个多边形顶点处的单位法向量，再由每个顶点的单位法向量通过向量合成⽅式计算各公共顶点的单位法向量，然后应⽤局部光照明模型计算各多边形顶点处的光颜⾊、光强度值，通过对各多边形顶点的光颜⾊、光强度值的线性插值求出多边形边及内部所有点的光颜⾊、光强度值。

Gouraud 算法的优点是计算量⼩。Gouraud 算法的缺点：⾼光区域有时会出现

异常；当对曲⾯⽤不同的多边形进⾏分割时会产⽣不同的显示效果；Gouraud

明暗处理会造成表⾯上出现过亮或过暗的条纹，称为⻢赫带（Mach\_band）效应。

Phong 算法基本思想：⾸先计算各个平⾯多边形的法向量，由该法向量确定各个

多边形顶点处的单位法向量，再由每个顶点的单位法向量通过向量合成⽅式计算

各公共顶点的单位法向量；通过对平⾯多边形各顶点的法向量进⾏双线性插值，

计算出平⾯多边形内部各点的法向量；再由所有这些点的法向量,通过局部光照模型计算平⾯多边形内部所有点的光颜⾊与光强度值。

Phone 算法的优缺点：Phong 算法显示的图形⽐ Gouraud 算法更真实，能够产

⽣正确的⾼光区域；由于是先计算出平⾯上各点的法向量，然后再计算各点的光

强度，所以 Phong 算法计算量远⼤于 Gouraud 算法；针对某些多边形分割的曲⾯，Phong 算法不如 Gouraud 算法好

## Whitted整体光照明模型

计算公式及各参数的含义

Ic：通过局部光照明模型计算出的光强度，也可采⽤ Phong 模型计算。

Is：为来⾃于其它⼏何体的镜⾯反射⽅向的⼊射光强度。

kS：为镜⾯反射系数，为 0～1 之间⼀个常数。

It：为来⾃于其它⼏何体的折射⽅向光强。

kt：为透射系数，是 0～1 之间的常数。

Whitted 光照模型是⼀个递归的计算模型。

## 光线追踪算法

光线追踪算法是一种在由多光源与多个几何体组成的场景中，对几何体进行消隐与整体光强度计算的算法。光线追踪算法沿着到达视点的光线的反方向进行跟踪，经过屏幕上每一个像素点，找出所跟踪的光线与几何体的交点，在该交点处分别沿反射方向与折射方向再进行跟踪，找出影响该点光强度的所有光源，通过迭代、累加计算跟踪点的光颜色与光强度。