# java3D与计算机图形学期末复习 第一章 绪论

图片与定义摘自《Java 3D 与计算机三维动态图形网络编程设计》

重点复习：

1、线架模型

英文：Wire Frame

以线段、圆弧和一些简单的曲面来表示一个三维模型。

2、Brep

英文：Boundary Representation

中文含义：边界面表示

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。

最常用的数据结构是翼边结构与半边结构

3、NURBS曲面

英文：Non Uniform Rational B-Spline

中文含义：非均匀有理B样条曲面

4、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。

平面或曲面的半空间是指一个空间平面或曲面将空间分为两部分，如果一部分位于物体的内部则另一部分位于物体的外部。

5、CSG方法

英文： Constructive Solid Geometry

中文含义：构造实体几何

实体造型（Solid Modeling）就是通过各种实体之间的并、交、差、布尔运算生成一个封闭实体的过程。

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。其中运用了二叉树来记录构造过程，这种表示也被称为实体的隐式模型(Unevaluted Model)或过程模型(Procedure Model)。

6、VOXEL（体素）

含义：三维图形显示的最小单元

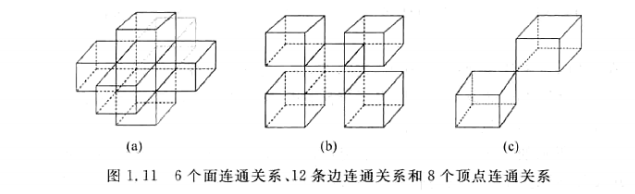
规则体素空间是将一个立方体分别沿x、y、z轴进行等间距均匀分割所形成的，每一个体素都是一个小立方体。

其中每一个小体素所具有属性的定义分两种情况。

(1）用小体素8个角点的不同属性来定义,通过三次线性插值,可求出该体素内任一点的属性值,也可求出体素中心点的属性值,该中心点属性值,在体素足够小的情况下可用来代表整个体素的属性。

(2）直接定义小体素中心点的属性值﹐用该中心点属性值代表该体素的属性。

体素模型空间中，体素和体素之间的连接方式有三种：分别是6连通（面连接）、18连通（边面连通）、26连通（点边面连通），如图：



7、DEXEL模型

中文含义：深度元素模型

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。

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。

8、逆向工程（RE）

英文：Reverse Engineering

正向工程：一般的工业设计，先在计算机中设计三维几何模型，再将计算机中的三维几何模型经过机械制造的方式转变成真实的三维模型。

逆向工程：与正向工程相反，先有真实的三维几何模型，再通过三维扫描将真实的三维模型转变成计算机内的三位数据点云，通过计算机处理生成计算机内的三维几何模型。

在逆向工程中产生的三维模型都是以STL三角网格数据文件格式给定的,因此,这种模型是三角网格模型。

9、STL三角网格数据文件

STL全称：Stereo Lithography

STL格式文件是用三维几何体表面的一系列三角形的顶点坐标与法向量来表示一个实体。

采用的半边数据结构，对3D扫描设备所得到的体的三维数据点云进行检查，清除多余面、孔洞或缝隙等问题，对其体、面、边、点的拓扑结构进行重建。

STL文件格式有两种：二进制格式与ASCII码格式。（后者最为常用）

# java3D与计算机图形学期末复习 第二章 java3D基本概念

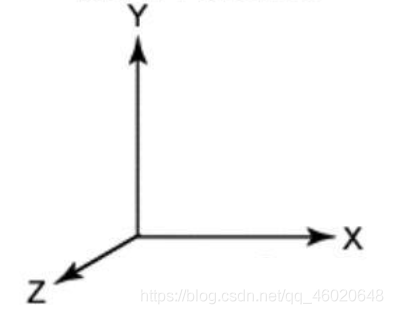
1、java3D高分辨率大尺度坐标系

java3D高分辨率大尺度坐标采用256个二进制位的定点数来表示一个数。

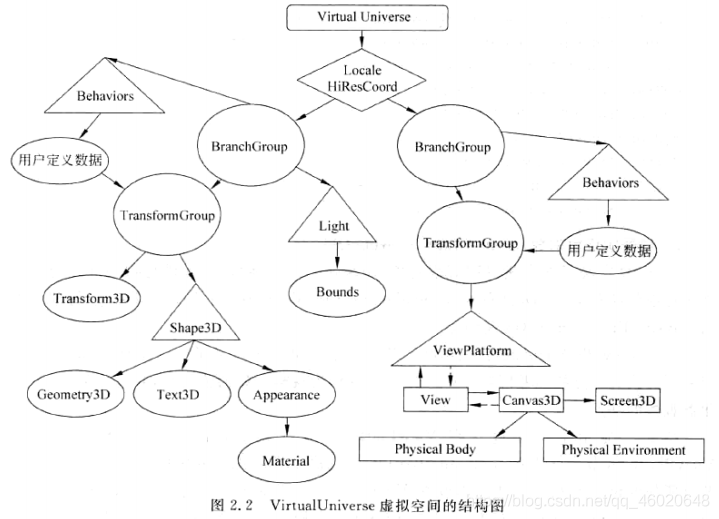
2、默认情况下java3D坐标系

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：



3、VirtualUniverse虚拟空间的结构图



4、SimpleUniverse类

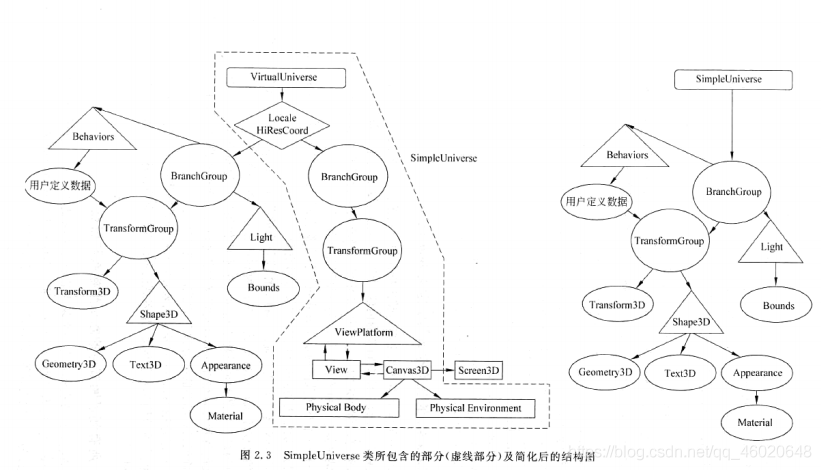
可以快速的设置一个最小的用户环境，并且很容易使一个Java3D应用程序运行起来。

5、Java3D的三维空间范围定义Bounds类

Bounds类对象常用来确定某种动作或行为的范围。行为和声音只有在它们离观察者足够近的时候才执行或者播放。通过正常地应用Bounds三维范围，程序员可确保只有相关的行为和声音被执行或者播放。

三维范围Bounds对象也用来确定某种全景操作的应用范围，这种全景操作包括背景、剪裁、音响范围选择。例如，距离观察者最近的应用程序的背景节点将选择为给定的观察对象使用。

6、SimpleUniverse类所包含的部分(虚线部分)及简化后的结构图



7、java3D的View类观察模型

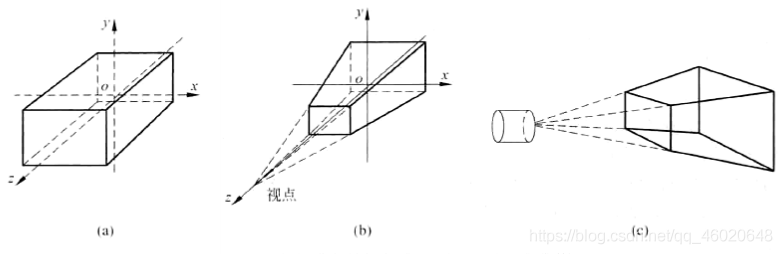
View(观察)：主要的观察对象包含了很多观察的状态。

8、在虚拟世界中的ViewPlatform观察平台

ViewPlatform(观察平台)：一个view用一个叶子结点来在场景图为自己定位。观察平台的起始结点指定了它的位置、方向和在虚拟世界中的比例尺。

ViewPlatform观察平台坐标系是一种附加有一个View观察的ViewPlatform叶子节点的局部坐标系。

9、平行投影（a）与透视投影（b）及基于相机的观察模型（c）



# java3D与计算机图形学期末复习 第三章 java3D基本图形功能

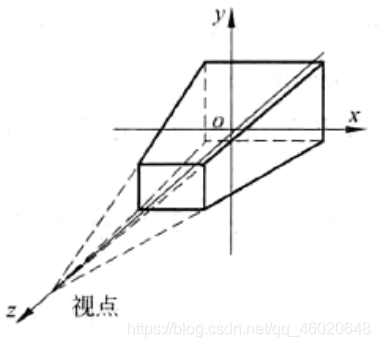
1、SimpleUniverse类

该类可以快速的设置一个最小的用户环境，并且很容易使一个Java3D应用程序运行起来。

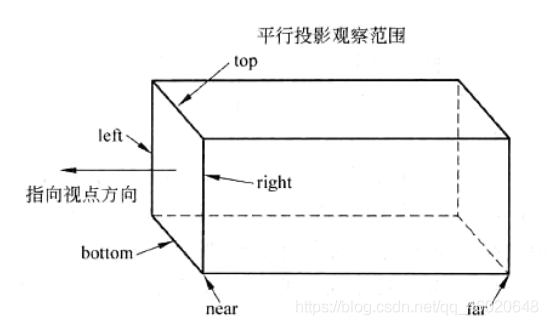
该类创建一个Locale(场景)、一个单独的ViewingPlatform(观察平台)和一个Viewer(观察者对象)，其中,该ViewingPlatform和Viewer观察者都是用其所有默认参数创建的。

2、ViewingPlatform类

透视投影(Perspective Projection)：模拟人眼睛看东西和相机照相的原理，特点是离视点近的形体显得大，离视点远的形体显得小。其中，在投影平面上的投影通过相似三角形的比例关系进行计算。



平行投影(Parallel Projection)：将空间的三维形体分别向三个坐标平面进行平行投影，形成三视图。最常用的就是工程制图中的主视图、俯视图和侧视图。



3、Shape3D类

Shape3D类定义所有的几何体。它包含几何体与该几何体的外观属性。几何体部分定义三维空间体的几何形状,外观属性部分定义颜色、材质等属性。

在一个Shape3D的几何对象列表中的所有元素必须属于同一个等价类，也就是属于同一种基本的几何类型。

等价类：对Geometry Array类的所有子类，所有的点对象是等价的，所有线对象是等价的，所有多边形对象是等价的。

4、Appearance类

Appearance类对象定义所有与显示相关的外观状态，这些状态可设置为一个Shape3D节点的组件对象(Component Object)。

这些显示状态包括下列几个方面：

颜色属性、点的属性、线属性、多边形属性、可视化模式、透明度属性、材质、纹理、纹理属性、纹理坐标的产生。

5、BranchGroup类

一个 BranchGroup作为一个场景图分支的根。BranchGroup对象是唯一能插入到一个Locale对象中的对象。以 BranchGroup节点为根的场景图子图是一个编辑单元。

注意：如果一个 BranchGroup节点包含在另一个子图中，作为其他Group的子节点，则该BranchGroup节点不能插入到一个Locale 节点。

6、TransformGroup类

TransformGroup节点通过Transform3D对象定义了一个唯一的3D空间坐标变换,该变换可对其子节点进行位置、方向及比例变换。（仿射变换）

7、GeometryArray类

GeometryArray类是 PointArray类,、LineArray类、TriangleArray类、QuadArray类、GeometryStripArray类与IndexedGeometryArray类的直接父类。

GeometryArray类的对象中包含独立的位置坐标数组、颜色数组、法向量数组、纹理坐标数组和顶点属性。

8、PointArray类

PointArray类是组织与定义点的数组类。

9、PointAttribute类

PointAttributes类的对象用来定义点的各种属性。

点的属性包括：

（1）Size，点的大小(默认为1个像素)；

（2）Anitialiasing，反走样(当一个点的大小大于一个像素单位时,为了使点在显示时外形接近于一个圆形,则需要设置该参数)。

10、IndexedPointArray类

从定义好的所有点的坐标与颜色数组中选择出一部分点进行显示。

1、LineArray类

LineArray类是组织与定义线的数组类。

该类生成的线段是不连续的,线段的连接方式:0-1，2-3，4-5等。其中,1、2之间与3、4之间不连接。

2、LineAttributes类

LineAttributes类定义所有与线的显示相关的属性与状态。

3、IndexedLineArray类

IndexedLineArray类从定义好的所有点的坐标中选择一部分连成线段显示。

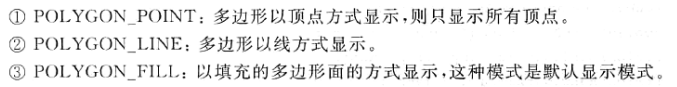
4、LineStripArray类

LineStripArray类将定义好的数据点分组，分别连成几段连续的线段，分成几个组，组间线段连续，各组之间互不连接。

5、多边形属性类PolygonAttributes类

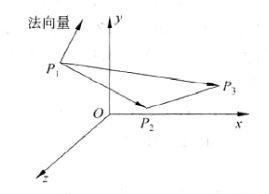
PolygonAttributes类用来定义多边形显示时的相关属性。

三种显示模式：

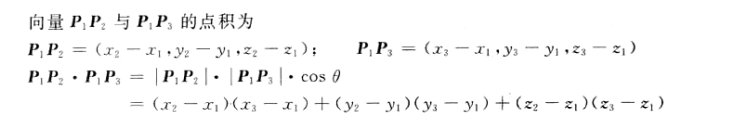


6、多边形面及其法向量的计算

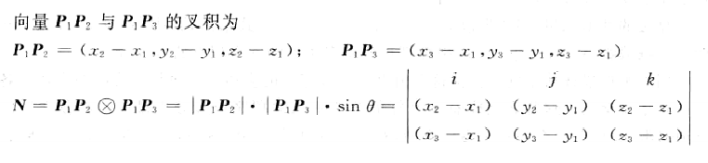
如图所示，求法向量？



先求出点乘，计算出两向量的夹角：



通过叉乘，计算出法向量N：

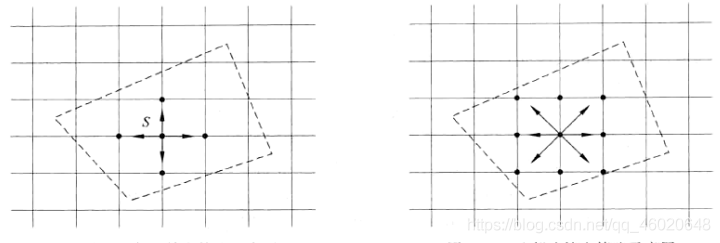


7、多边形可视化的填充算法

（1）种子填充算法

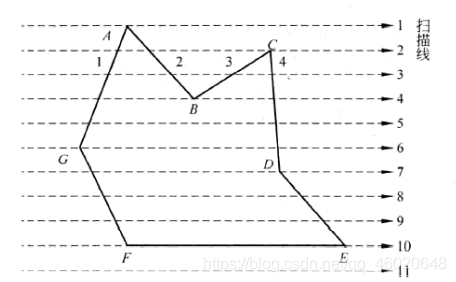
种子填色算法首先在要填充的多边形区域内选择一个填色种子点( Seed Point)，然后以该种子点为基础，通过与多边形边界的像素点相比较进行填色。

种子填色算法主要有四邻法（如左图）和八邻法（如右图）：



（2）扫描线填充算法

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，采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：

首先按照多边形顶点顺时针方向标明：A、B、C、D、E、F、G

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）。

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）。

8、三角面TriangleArray类

TriangleArray类以一维顶点数组给出顶点的坐标值，从前向后依次以3个顶点形成一个三角形，并且上一个三角形与下一个三角形之间没有公用顶点。

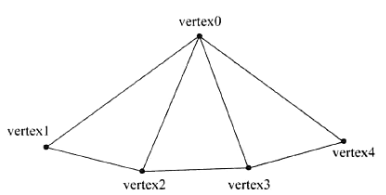
9、三角带TriangleStripArray类

在同一个三角形Strip内，生成三角形时，第一个三角形由前3个顶点生成，第一个三角形的后两个顶点与第四个顶点形成第二个三角形，其余依次类推。

两个子Strip之间没有交点。

10、三角扇TriangleFanArray类

TriangleFanArray类在组织三角形时，以第一个顶点为公用顶点，依次与其余顶点分别连接形成三角形。用这种方法生成的一系列三角形公用第一个顶点。如图：



1、四边面QuadArray类

QuadArray类以顶点坐标数组中给出的一维顶点数组，从前向后依次以4个顶点形成一个四边形面，并且相邻两个四边形面之间没有公用顶点。给定的总的顶点数必须是4的倍数。

2、颜色属性ColoringAttributes类

ColoringAttributes类定义所选择的颜色与光照模型(也称为阴影模型)。

3、材质Material类

Material类的对象定义一个三维物体在光照情况下的外观。

如果在一个Appearance类的对象中的Material类的对象一项为null，则光照效果对所有包含该Appearance类的对象的节点不起作用。

4、透明属性TransparencyAttributes类

TransparencyAttributes 类的对象定义所有几何体透明度的属性。

5、光源Light类

Light类叶子节点是一个抽象类，其中定义的一组参数（光颜色、光照开关标志和一个光的作用范围）属于所有类型的光。

光源的类型包括：

（1）点光源（Point Light）

（2）平行光源（Directional Light）

（3）环境光源（Ambient Light）

6、Vetor3f类

Vector3f类是一个包含有3个单精度浮点元素x、y、z的向量。

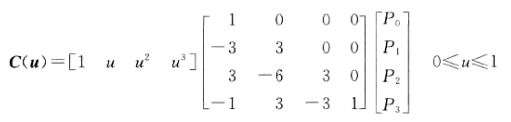
7、Java语言多维数组表示

没啥好说的，懂的都懂，不懂的我也不想说了（你搁这搁这呢？）

# java3D与计算机图形学期末复习 第四章 复杂曲线、曲面设计

重点复习：Bezier曲线、曲面

1、三次Bezier曲线的矩阵表示



其中P0、P1、P2、P3是四个控制顶点，u为三次Bezier曲线的参数

2、Bezier曲线的De Casteljau算法

如果给定空间n＋1个点P0、P1、P2…Pn，则可生成一条n次的Bezier曲线。根据参数u的值（(0,1]区间内从小到大取值）,并通过对控制顶点依次进行线性插值算法,可求出 Bezier曲线上的点。这种算法被称为Bezier曲线的De Casteljau算法。

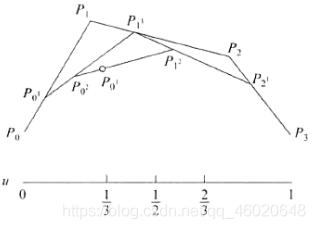
实现步骤（个人理解）：

（1）将所有控制顶点依次连接，取参数u的值；

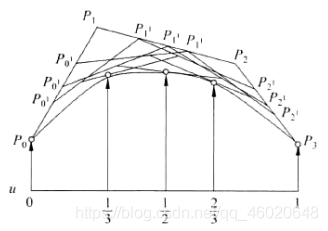
（2）每个连接线取1/3处的点，然后依次连接取得的1/3点（这里以u=1/3为例）；

（3）若最后还剩下两个点及其以上，重复（2）步骤。每重复一次，顶点数会减少一个；

（4）最后剩余一个点。这个点就是Bezier曲线在1/3的点。



（5）重复步骤（1）-（4），取不同的u值，可求出不同位置的Bezier曲线的点。



3、Bezier曲线、曲面拼接时的连续性

（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。

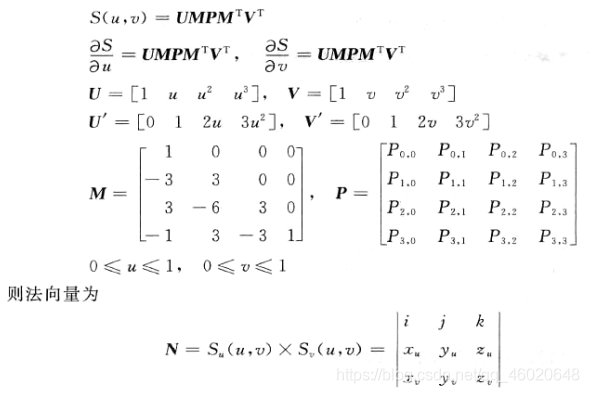
（4）G0连续：0阶几何连续，性质同C0连续。

（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。

4、Bezier曲面法向量计算

以三次Bezier曲面为例：



5、Bezier曲面的性质

（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(o,v)、S(1,v)， S(u,0)，S(u,1)分别由对应的控制网格的4条边界控制多边形生成。

（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。

（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。

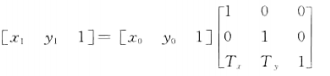
# java3D与计算机图形学期末复习 第五章 坐标变换与基本形体

重点复习：

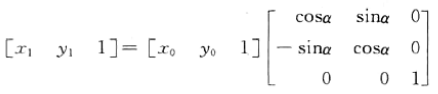
1、图形坐标变换

初始点A坐标为（x0,y0），变换后点A'坐标为（x1,y1）

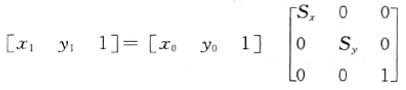
（1）平移变换

矩阵表示：，其中Tx=（x1-x0）,Ty=（y1-y0）。

（2）旋转变换

矩阵表示：，其中α为点A围绕圆心逆时针旋转的角度。（如果是顺时针旋转α角，将上面的α全部替换成-α即可）

（3）等比变换

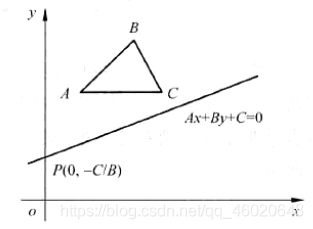
矩阵表示：，其中Sx= x1/x0，Sy= y1/y0。

（4）轴对称变换

轴对称变换是特殊的等比变换，Sx=1，Sy=-1，是对于x轴对称；Sx=-1，Sy=1，是对于y轴对称。

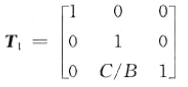
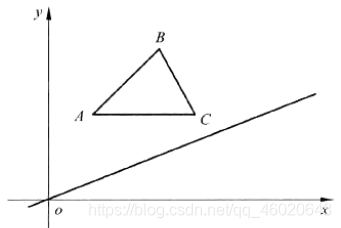
2、组合变换

将上述四种变换，综合起来使用就可以求出复杂的变换。接下来，以求三角形ABC关于直线Ax+By+C=0对称的三角形A'B'C'为例：

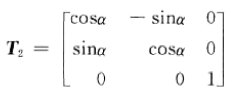
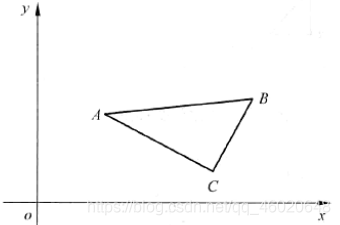


具体步骤：

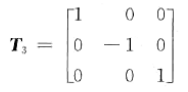
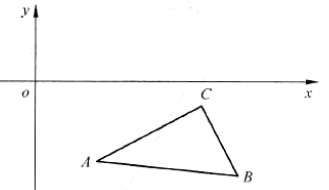
（1）求出直线与y轴的交点坐标（0，-C/B），将三角形向下平移C/B，得出第一个矩阵T1。

，变换后结果为

（2）标记直线与x轴的夹角为α角，将三角形顺时针旋转α角，得出第二个矩阵T2。

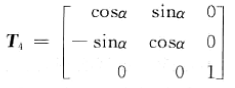
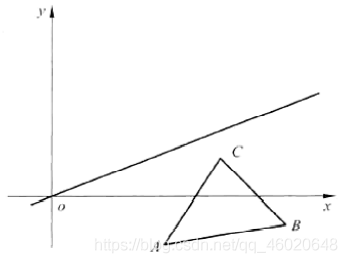
，变换后结果为

（3）对三角形求关于x轴对称，得出第三个矩阵T3。

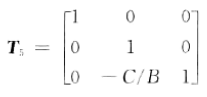
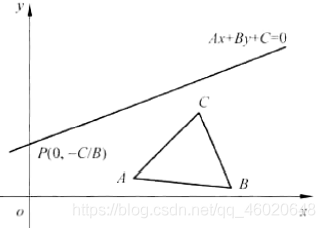
，变换后结果为

当前，直线Ax+By+C=0是与x轴重合的，我们接下来需要将它变换到原来的位置。

（4）逆时针旋转α角，得出第四个矩阵T4。

，变换后结果为

（5）最后向上平移C/B，得出第五个矩阵T5。

，变换后结果为

（6）最后将所有矩阵相乘，可以求出组合变换矩阵。



3、Transform3D类

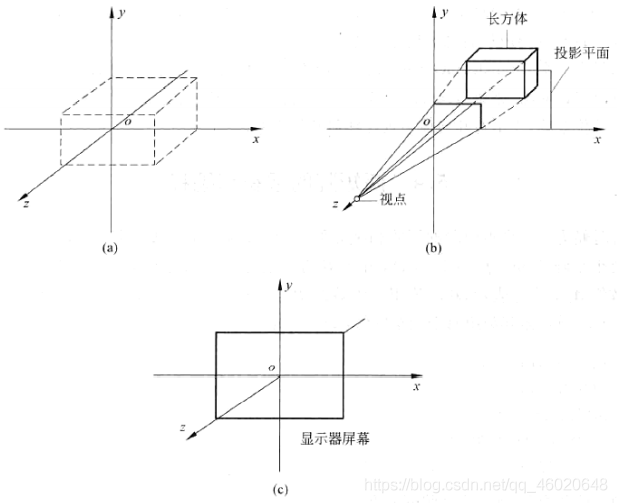
通常一个Transform3D类的对象是一种内部表达为4×4的双精度类型的矩阵。一个Transform3D类的对象用来执行平移、旋转、变比例等坐标变换。

4、TransformGroup类

所有表示各种几何变换的Transform3D类的对象必须包含在一个TransformGroup类的对象中，表示对在该TransformGroup类的对象中的三维图形所进行的坐标变换。

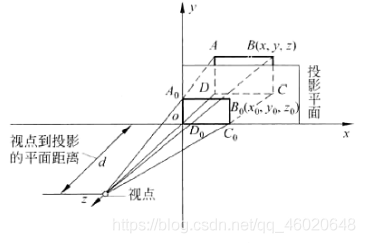
如果表示一种几何变换的Transform3D对象没有加入到该TransformGroup类的对象中,则该变换对在该TransformGroup类的对象中的三维图形不起作用。

5、世界坐标系、观察坐标系与Java3D显示器坐标系

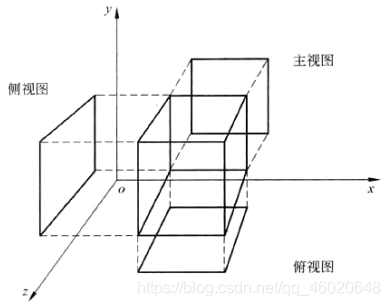


6、透视投影、平行投影及其计算

透视投影利用相似三角形原理计算。



平行投影——就是求三视图



7、隐藏线、隐藏面消除算法

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。

8、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。

简单地说：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。

java3D与计算机图形学期末复习 第五章 坐标变换与基本形体

重点复习：

1、同一平面内点与多边形的包含判定

（1）凸多边形的叉积判断法

（补充一个知识点：

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）

从给定的一个点向多边形的各个顶点做向量。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。

（2）夹角之和检验法

从给定的一个点向多边形的各个顶点做向量。按顶点顺序逐个计算每相邻两向量之间的夹角。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。

2、局部光照明计算模型

在给定光源（包括一个或多个光源）光源的类型、颜色以及几何体表面的各种特性参数后，不考虑场景中其他几何体的影响，通过某种计算算法计算几何体表面上各点的光颜色与强度，将这种算法称为局部光照明计算模型。

局部光照明计算模型只考虑物体表面对直射光、环境光的反射、散射与透射。

3、Gouraud算法

基本过程：

（1）先计算出各个小多边形及其顶点的单位法线向量，公共顶点处用合成向量。

（2）在给定光源与几何体表面各特性参数基础上，用局部光照计算模型计算每个顶点的光颜色、光强度值。

（3）由两顶点的光颜色、光强度值,通过线性插值计算出两个顶点之间的边上的各点的光颜色、光强度值。

（4）在扫描线填色算法中，每条水平扫描线与多边形的边相交，都有一对交点,通过线性插值求出这些交点的光颜色与光强度值，再由这些交点的光颜色、光强度值，通过线性插值计算出该水平扫描线上位于多边形内部的各像素点的光颜色、光强度值。通过水平扫描线从上向下的移动，也就相应地求出了小多边形平面上所有点的光颜色、光强度值。

优点：计算量小

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，称为马赫带(Mach \_band)效应。

4、Phong算法

基本过程：

（1）计算平面多边形的单位法矢量。

（2）计算多边形各顶点处的单位法矢量,进而计算各公共顶点的合成向量。

（3）通过对多边形各顶点的单位法向量进行双线性插值,计算多边形内部点的单位法向量。

（4）通过平面多边形内部各点处的单位法向量,利用局部光照明模型计算多边形内部各点光颜色,光强度值。

5、Whitted整体关照模型

为了增加三维图形显示的真实性，除了考虑各种类型光源对几何体表面的直接照射外，还必须考虑当前场景中来自于其他几何体的漫射光、镜面反射光和透射光对该几何体表面的照射，这样就会产生比较真实的照明效果，如阴影与透明效果等。将这种光照明计算模型称为整体光照明模型。

Whitted光照明模型是一种整体光照明模型,，它能很好地模拟光能在光滑几何体表面之间的镜面反射和通过透明体产生的规则透射。

6、光线追踪算法（Raytracing）

光线跟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。

光线跟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。光强度计算包括直接照射到几何体表面上一点的漫射光源、直射光源、透射光源,还包括来自反射跟踪方向的其他几何体传来的光强与沿着透射跟踪方向的其他几何体传来的光强。

7、Switch类

意为切换，Switch类可以控制哪些节点将被显示。

8、OrderedGroup类

一般情况下，在一个场景图中三维形体的显示顺序是不确定的，这种显示顺序依赖于当时程序的执行效率。但是，OrderedGroup 类可确保在一个场景图中的三维形体的显示顺序。

9、SharedGroup类

SharedGroup类提供了一种可对场景图中的子图进行链接(Link)复制的能力。这样，在生成复杂场景时，就可以通过Link叶子节点来共享与复制场景中的子图。SharedGroup节点允许叶子节点对其子图通过链接(由Link叶子节点实现)方式共享。

10、Link类

Link 叶子节点允许一个应用程序引用一个以SharedGroup为根的共享子图。任何数量的Link 叶子节点都能引用同一个SharedGroup节点。