**VirtualUniverse类**定义的对象是包含所有场景图的最高级别的容器。

**SimpleUniverse类**创建场景图中与观察有关的所有必需对象。

ViewingPlatform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

**Appearance类**定义所有与显示相关的外观状态

**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。

**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。

**GeometryArray类**的对象中包含有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。

**点PointArray类**是组织与定义点的数组类。

**PointAttributes类**的对象用来定义点的各种属性。

**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。

**线LineArray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。

**LineAttributes类**该类定义所有与线的显示相关的属性与状态。

**IndexedLineArray类**从定义好的所有线的坐标与颜色数组中选择出一部分线进行显示。

**LineStripArray类：**Strip的英文含义是带、条的意思。该构造函数将定义好的数据点分组，然后分别连成几段连续的线段，分成几个组，就连成几条带（几个Strip）。各条带之间互不连接。

**PolygonAttributes类**用来定义多边形显示时的相关属性。这些基本多边形包括：三角形，三角形带，三角形扇，四边形。

**三角面TriangleArray类**以一维顶点数组给出顶点的坐标值，从前向后依次以三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。

三角带TriangleStripArray类

**三角扇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。其中给定的顶点数最少为3个。用这种方法生成的所有三角形面具有指向同侧的法向量。

**IndexedTriangleArray类：**用该类时，首先需要根据给定顶点的一维数组的序号，建立要用到的顶点的索引数组index，该数组是一个一维数组，每个元素保存选择出的一个顶点的序号。然后再用继承自IndexedGeometryArray方法分别设置颜色数组、坐标数组与索引数组之间的对应关系。

**四边面QuadArray类**以顶点坐标数组中给出的一维顶点数组，从前向后依次以四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。

**IndexedQuadArray类：**用该类时，首先需要根据给定顶点的一维数组的序号，建立要用到的顶点的索引数组index，该数组是一个一维数组，每个元素保存选择出的一个顶点的序号。然后再用继承自IndexedGeometryArray方法分别设置颜色数组、坐标数组与索引数组之间的对应关系。

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

**Material类**定义三维型体在光照条件下外观，包括环境光颜色（ambient color）、散射光颜色（diffuse color）、反射光颜色（specular color）、发射光颜色（emissive color）、光亮度（shininess）。这些属性都定义在Material类对象中

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

**平行光源DirectionalLight类**定义了一种光源在无限远处有方向的光。

**环境光源AmbientLight类**是指来自于所有方向的一种光源。

**Vector3f类**是一个包含有3个单精度浮点元素x、y、z的向量。如果该向量表示一个法向量，则该向量应该单位化为一个单位向量。

**Transform3D类**的对象是一种内部表达为4x4的双精度类型的矩阵，该矩阵是以行的方式存放的。一个Transform3D类的对象用来执行平移、旋转、变比例等坐标变换。

TransformGroup类

**Switch节点**主要控制哪些节点将被显示。

**OrderedGroup类**可确保在一个场景图中的三维形体的显示顺序

**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。

**Alpha类：**Alpha节点组件对象对将一个时间值转换为[0,1]范围内的一个alpha值提供一种公用方法。Alpha对象实际上是一个以时间为变量的函数，该函数能够根据给定的时间产生[0,1]范围内的alpha值。Alpha对象最基本的应用就是对插值行为提供alpha值。

**Behavior叶子节点**对象场景图中增加用户定义的行为提供一种框架。

**Interpolator类**是一个抽象类，扩展自Behavior类，为不同的插值提供公用的方法，包括将一个时间值装换成一个alpha值，以及初始化一个行为类所用的方法。

**LOD类**可使三维形体根据距离视点的距离决定其显示的细节。引入该细节程度功能主要就是为了在三维形体距离视点较远时，将各种细节程度减小，以减少计算量，提高程序的运行效率。

**Billboard类**行为节点对目标TransformGroup节点进行操作，使目标TransformGroup节点中局部坐标系的z轴正向始终指向观察者的眼睛位置，而不管目标TransformGroup节点中transform坐标变换如何设置。

**Morph类**是一个叶子节点，允许一个应用程序在多个集合数组之间进行变换。

**Texture类**在纹理映射是定义纹理的属性。

线架模型：以线段、圆弧和一些简单的曲线来表示一个三位模型。

Brep边界面模型：用几何体边界面的顶点、边、面及其关系来描述一个几何体。

NURBS曲面：

半边数据结构：是一种五级层次数据结构，它是有5种节点Solid、Face、Loop、HalfEdge和Vertex组成的链接结构。

Solid实体：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。

VOXEL体素模型：规则体素模型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。

DEXEL深度元素模型：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。

VOXEL体素模型与多属性体图形学：VOXEL体素模型的最小几何单元是一个三维小体素。对体素显示的技术主要包括空间物体模型顺序、图像顺序和域方法等。

逆向工程：逆向工程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维数据点云，通过计算机对数据点云进行处理，生成计算机内三维几何模型，这个过程正好与一般的工程设计过程相反，因此称为逆向工程。

STL三角网格数据文件：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。由于这种文件格式简单、容易读取与处理，所以现在也普遍用于不同CAD系统之间的数据交换。

快速原型系统：快速原型系统是将CAD产生的实体模型转变为以三角面表示的STL文件格式，然后再将该三角网格面模型从底向上逐渐切片转换为层状数据，在每一层内属于实体内部的部分，确定激光头的移动轨迹，生成一层的模型，再逐层以层层堆叠的方式快速、自动地制作出整个零件。

Java3D高分辨率大尺度坐标系：Java3D高分辨率大尺度坐标系采用256个二进制的定点数来表示一个数。Java3D高分辨率大尺度坐标系用256位带正负符号的、二进制补码、固定点数来表示。固定点在128位。固定点之右有128个二进制位，固定点之左有128个二进制位。这种坐标系足以描述大到几千亿光年的距离，小到可以描述比质子还小的物体。

透视投影：模拟人眼睛看东西与相机照相的原理，以这种投影方式显示的图像与人眼睛看到的，或者与相机照相照到的图像相近。这种透视投影的特点是离视点近的形体显得大，离视点远的显得小。

平行投影：将空间的三位形体分别向3个坐标平面进行平行头型，分别形成各种视图。

多边形的法向量计算：给定3个点及其坐标*P*1（*x*1，*y*1，*z*1）、*P*2（*x*2，*y*2，*z*2）、*P*3（*x*3，*y*3，*z*3），计算这3个点形成的平面的法向量。

向量***P1P2***与***P1P3***的叉积为

***P1P2*** =（*x*2 － *x*1，*y*2— *y*1，*z*2 — *z*1）

***P1P3*** =（*x*3 － *x*1，*y*3— *y*1，*z*3 — *z*1）

***N***= ***P1P2***  ***P1P3*** = | ***P1P2***|·| ***P1P3***|·=

向量***P1P2***与***P1P3***的点积为

***P1P2*** =（*x*2 － *x*1，*y*2— *y*1，*z*2 — *z*1）

***P1P3*** =（*x*3 － *x*1，*y*3— *y*1，*z*3 — *z*1）

***S***=***P1P2***  ***P1P3*** = | ***P1P2***|·| ***P1P3***|·=（*x*2 － *x*1）（*x*3 － *x*1）+（*y*2— *y*1）（*y*3— *y*1）+（*z*2 — *z*1）（*z*3 — *z*1）

**种子填充算法**：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象素点相比较进行填色

种子填色算法主要有四邻法、八邻法。

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)是一种运算效率比较高的多边形填充算法。这类算法是建立在多边形边界的基础上。

算法的基本思想：多边形以*n*、x\_array、y\_array的形式给出，其中，x\_array、y\_array为两个一维数组，x\_array、y\_array中分别存放着多边形的*n*个顶点的*x*、*y*坐标。用水平扫描线从上到下扫描这个由点与线段定义的多边形。每根扫描线与多边形各边产生一系列交点。这些交点按照*x*坐标值从小到大的顺序进行分类排列。将分类后的交点成对取出，作为两个端点。通过拍段，对属于多边形内部的线段，以需要填的颜色画水平直线。而对属于多边形外的线段，不用进行填充。多边形被扫描完毕后，填色也就完成。

**双三次Bezier曲面的矩阵表示形式**

****

**Bezier曲面法向量计算**

**对Bezier曲面S（u，v），当给定参数点（u，v）时，则可分别求S（u，v）对参数u、v的偏导数矢量。再由这两个偏导数矢量的叉积求得曲面在该点的法向量。**

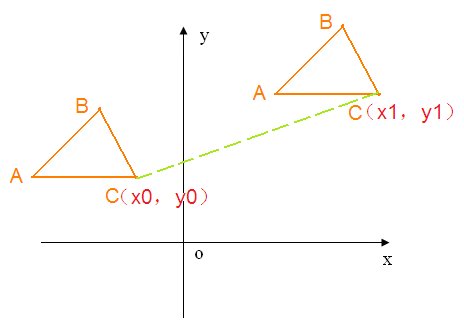
****

**则法向量为：**

****

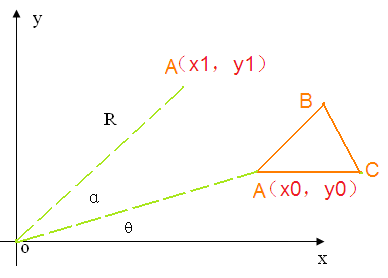
**二维空间图形坐标变换**

**平移变换(Translation)**



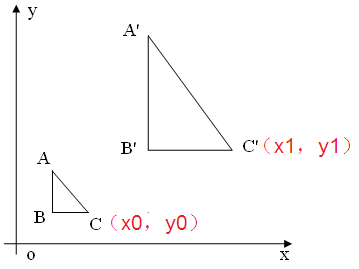


**旋转变换（Rotation）**





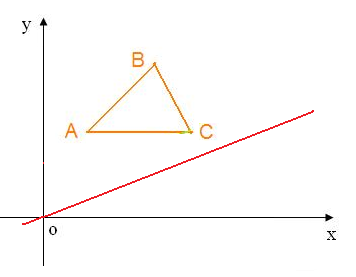
**变比变换(Scaling)**

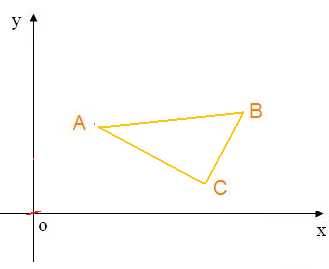


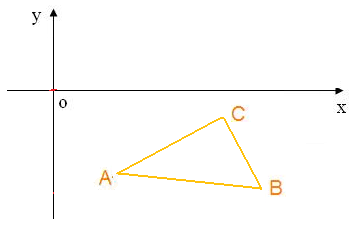


**组合变换**

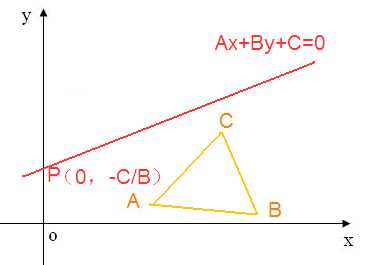
已知直线的方程为Ax+By+C=0，该直线与y轴的交点为P（0，-C/B），直线旁边有一个三角形ABC，如图5.7，现在要求将该三角形对该直线做对称变换，现在介绍该变换步骤，以及每一步的变换与变换矩阵。







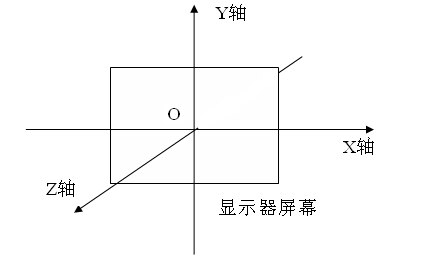
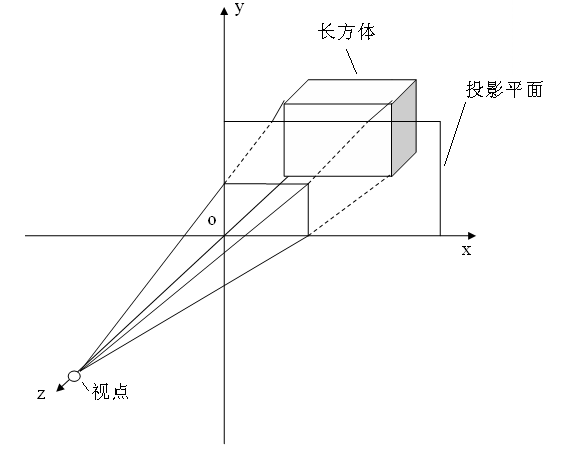
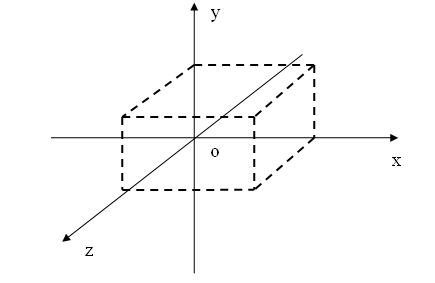






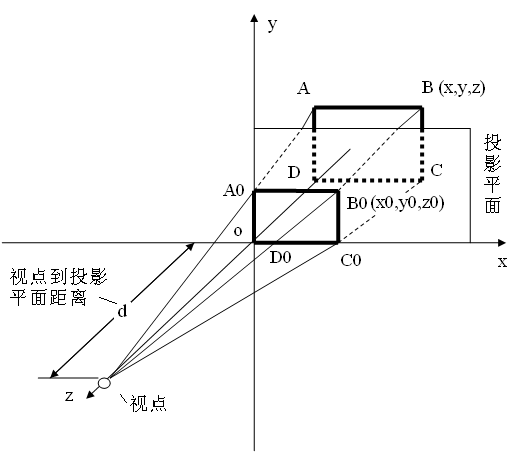


**世界坐标系、观察坐标系、Java 3D所用的显示器坐标系**



**透视投影计算**

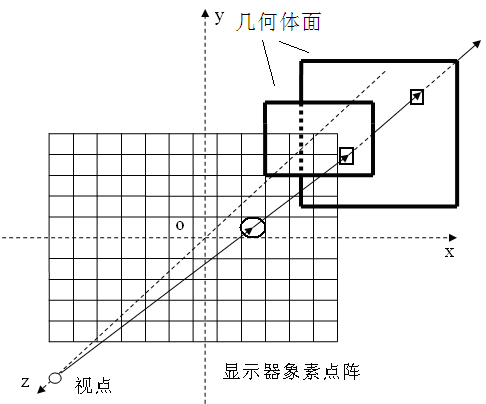
在世界坐标系中，定义三维型体。当给定视点、观察方向与投影平面后，将世界坐标系转换为观察坐标系，对观察坐标系中的三维物体通过比例变换向投影平面投影，将这种投影称为**透视投影（Perspective Projection）。**

**平行投影也称为垂直平行投影，**就是将世界坐标系中三维几何体分别向XOY、YOZ、ZOX坐标平面做垂直投影，分别形成主视图、侧视图、俯视图。在这种投影中，投影平面分别和三个坐标平面重合，从空间点到坐标平面上的点的投影不进行比例投影变换。

将这种消除所有隐藏边的算法称为**隐藏线消除算法**

将这种消除隐藏面的算法称为**隐藏面消除算法**

**Z Buffer隐藏面消除算法原理**如下。首先针对显示器的象素点阵建立两种缓冲存储单元阵列：Z Buffer与C Buffer阵列。Z Buffer与C Buffer存储单元阵列中的每一个存储单元与显示器象素点阵中的一个象素唯一地一一对应，如图5.28。

**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，最后将所有的可见多边形显示出来，就为几何体消隐后的图像。

**扫描线消隐算法**

见课本230

**光线跟踪消隐算法**

**最小包围矩形**就是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：求所有几何数据点的最小*x*、*y*值与最大*x*、*y*值，则有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**

就是用三对分别平行于坐标平面的最小六面体包围一个三维几何体。

求法：对一个三维几何体所有顶点的*x*、*y*、*z*坐标进行比较，分别求出最小与最大的*x*、*y*、*z*坐标，即Xmin、Xmax、Ymin、Ymax、Zmin、Zmax，则最小坐标点（Xmin，Ymin，Zmin）与最大坐标点（Xmax，Ymax，Zmax）所形成的平行于坐标平面的长方体就是最小包围盒

**点与平面多边形区域的包含判定**

给定一个点P（*x*，*y*，*z*），多边形所在平面方程*ax+by+cz+d=*0，

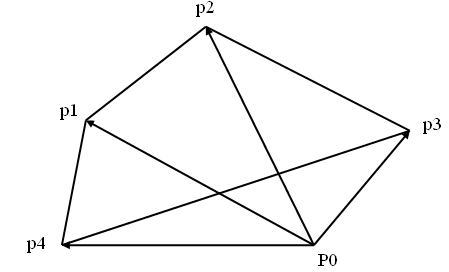
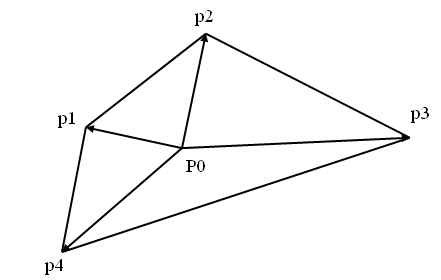
点到平面的距离为*d* =

给定一个误差，若*d* < ,则认为点在该平面上。

**同一平面内点与多边形的包含判定**

**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作向量。然后，按顶点顺序逐个计算每相邻两向量之间的叉积。如果是按逆时针方向计算的叉积，其值符号则为正；如果是按顺时针方向计算的叉积，其值符号则为负。



**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。按顶点顺序逐个计算每相邻两向量之间的夹角，并加上相应的正、负号。

向量***P1P2***与***P1P3***的叉积为

***P1P2*** =（*x*2 － *x*1，*y*2— *y*1，*z*2 — *z*1）

***P1P3*** =（*x*3 － *x*1，*y*3— *y*1，*z*3 — *z*1）

***N***= ***P1P2***  ***P1P3*** = | ***P1P2***|·| ***P1P3***|·=

向量***P1P2***与***P1P3***的点积为

***P1P2*** =（*x*2 － *x*1，*y*2— *y*1，*z*2 — *z*1）

***P1P3*** =（*x*3 － *x*1，*y*3— *y*1，*z*3 — *z*1）

***S***=***P1P2***  ***P1P3*** = | ***P1P2***|·| ***P1P3***|· =（*x*2 － *x*1）（*x*3 － *x*1）+（*y*2— *y*1）（*y*3— *y*1）+（*z*2 — *z*1）（*z*3 — *z*1）

= = =

**交点计数检验法**

当要判断的平面多边形的形状比较复杂时，如复杂凹多边形，或者带孔的多边形，无法使用前面两种方法判断，可采用交点计数法判断点是否在多边形内。

具体做法是，从要判断的点P0(x0,y0)向右作一条水平射线，射线的方程如下。求该水平射线与多边形边的交点个数。若交点个数为奇数，则点在多边形内；若交点个数为偶数，点在多边形外，如图5.35。

**局部光照明计算模型**

见课本P237

**Gouraud算法和Phong算法**

**Gouraud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，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，称为马赫带（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整体光照明模型算法** *I=Ic+ksIs+ktIt*

见课本P243

**光线追踪算法**

见课本P244