Java3D考点

第一章：绪论

1. **线架模型**：以线段、圆弧和一些简单的曲线来表示一个三维模型
2. **Brep**：边界面表示（Boundary Representation），Brep边界面可以是平面多边形，也可以是曲面，如Bezier曲面、B样条曲面、NURBS（非均匀有理B样条）曲面。
3. **Solid实体**：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
4. **SCG**（构造实体几何）方法：通过简单实体之间的正则布尔运算生成比较复杂的体。
5. **体素**（VOXEL）是三维图形显示的最小单元
6. **深度元素**（DEXEL），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，与体素模型相比，可以大大压缩存储空间
7. **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。
8. **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，STL文件标准是美国3D System公司于1987年制定的一个接口协议

第二章：Java 3D基本概念

1. Java 3D高分辨率大尺度坐标系采用**256**个二进制位的定点数来表示一个数
2. 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。
3. Virtual Universe 包含 Locale 包含 BranchGroup（P27图）
4. **Simple Universe**类可快速地设置一个最小的用户环境，并且很容易使一个Java 3D应用程序运行起来。该实用程序类创建了场景图中与观察相关的所有必须对象。
5. **Bounds**类用来限定特定操作的作用范围，Bounds类对象常用来确定某种动作或行为的范围。
6. SimpleUniverse 包含 Locale 包含 BranchGroup 包含 TransformGroup 包含ViewPlatform（P34图）
7. **View**类能让Java 3D观察模型编写的应用程序在不修改场景图的情况下能够将可视化后的图像显示到各种不同的显示设备上。
8. **ViewPlatform**类在虚拟世界中定义了一个坐标系和一个具有相关原点或参考点的参考框架。ViewPlatform作为观察对象的依附点，并且其作为一个可视化器观察的基点。
9. **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。
10. **透视投影**：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。

第三章：Java 3D基本图形功能

1. **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。在一个Shape3D的几何对象列表红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。（例如所有的点对象等价，所有的线对象等价，所有的多边形对象等价）
2. **Appearance**类对象定义所有与显示相关的外观状态。
3. **BranchGroup**类对象是唯一能插入到一个Locale对象中的对象，一个BranchGroup作为一个场景图分支的根。
4. **TransformGroup**类节点通过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。
5. **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。
6. **PointArray**类是组织与定义点的数组类。
7. **PointAttributes**类的对象用来定义点的各种属性
8. **IndexedPointArray**类能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。
9. **LineArray**类能够生产不连续的线段，线段的连接方式为：0-1，2-3，4-5等。
10. **LineAttributes**类定义所有与线的显示相关的属性与状态。
11. **IndexedLineArray**类能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。
12. **LineStripArray**类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，各条带之间互不连接。StripVertexCounts是一个一维整形数组，数组的长度为带的条数，数组元素的值为每条带中点的数量。
13. **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、三角扇、四边形。多边形的显示模式包括：①POLYGON\_POINT：多边形以顶点方式显示，则只显示所有的点。②POLYGON\_LINE：多边形以线方式显示。③POLYGON\_FILL：以填充的多边形面的方式显示，这种模式是默认显示模式。
14. **多边形面及其法向量**：向量P1P2=(x2-x1，y2-y1，z2-z1), P1P3=(x3-x1，y3-y1，z3-z1)，则法向量N=|P1P2|·|P1P3|·sinθ = | i j k |

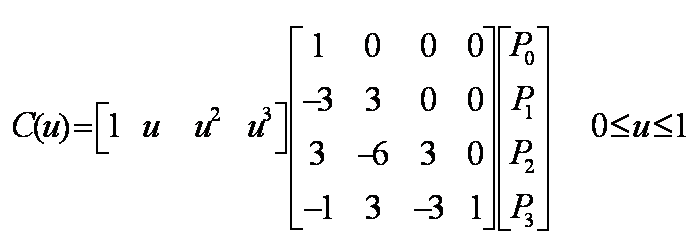
|(x2-x1) (y2-y1) (z2-z1) |

|(x3-x1) (y3-y1) (z3-z1) |

1. **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，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，如果要填充的像素点在多边形的边界之内（外），则（不）进行填充；如果要填充的像素点在边界上，则保留边界像素点颜色。
2. **扫描线填色算法**：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。通过判断对属于多边形内部的线段，以需要填充的颜色画水平直线。而对属于多边形外的线段，不用进行填充。
3. **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个顶点形成一个角形，并且上一个三角形与下一个三角形之间没有公共顶点。
4. **TriangleStripArray**类：将定义好的数据点分组，然后分别连成几个三角形，分成几个组，就连成几条三角带，各条带之间互不连接。StripVertexCounts是一个一维整形数组，数组的长度为三角带的条数，数组元素的值为每条三带中点的数量。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，其余以此类推。在每一个三角形Strip内，所有的三角形都是相互连接在一起的。
5. **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。所有三角形具有指向同侧的法向量。
6. **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。
7. **ColoringAttributes**类：定义所选择的颜色与光照模型
8. **Material**类：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。
9. **TransparencyAttributes**类：定义所有几何体透明度的属性。
10. **Light**类：定义光源
11. **Java语言多维数组**P109
12. **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。

第四章：复杂曲线、曲面设计

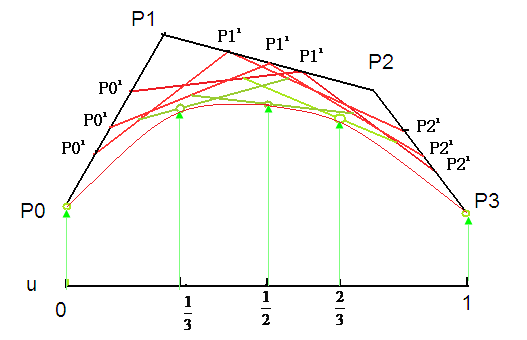
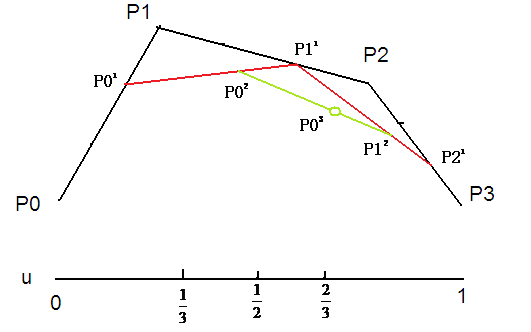
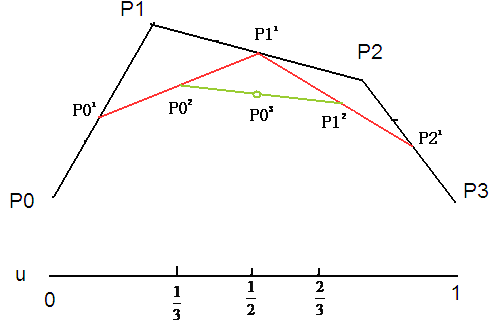
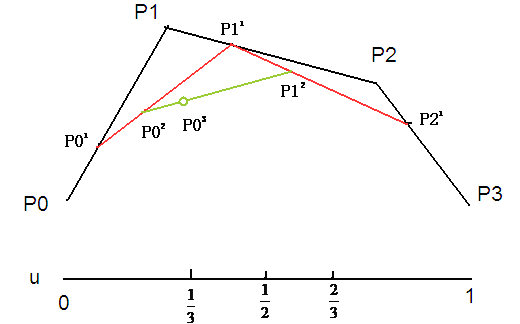
1. **三次Bezier曲线**的矩阵表示：



对参数u∈[0，1]区间的等分越多，得到的Bezier曲线越光滑、精确。

1. Bezier曲线的**De Casteljau算法**：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，并通过下面对控制顶点依次进行的线性插值算法,可求出Bezier曲线上的点。它也是一种手工绘制平面Bezier曲线的方法。





1. **三次Bezier曲面**的矩阵表示形式：

①一般Bezier曲面的定义：



②双三次Bezier曲面的矩阵表示



1. **Bezier曲线、曲面拼接的连续性**：

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。

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

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

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。

G1连续（一阶几何连续）：两相拼接的曲线段在拼接点处重合，切线方向相同，但大小不等，则这种连续称为G1连续。

G2连续（二阶几何连续）：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，将这种连续称为G2连续。

1. **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。下面以三次Bezier曲面为例说明。

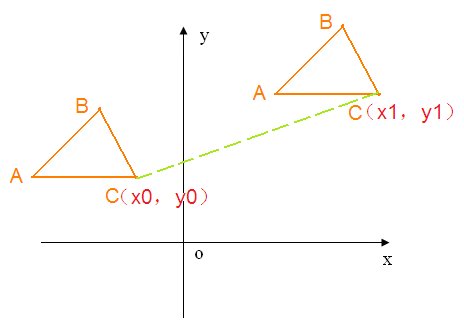
则法向量为：

1. Bezier曲面的**端点性质**：Bezier曲面的四个角点与控制多边形网格的四个角控制顶点重合。

第五章：Java 3D的坐标变换与基本形体

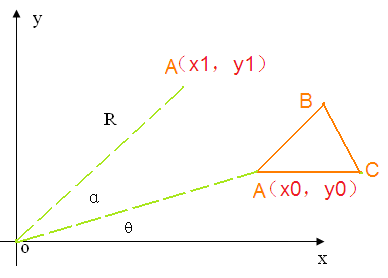
1. 二维空间图形**坐标变换**

①**平移变换**：图形在二维空间的平移是指将一个图形从一个位置(x0, y0)移到另一个位置(x1，y1)所进行的坐标变换。此时，Ｔx=x1-x0，Ｔy=y1-y0



②**旋转变换**：以二维坐标原点(0, 0)为圆心，将图形上的点A(x0, y0)围绕该园心逆时针转动一个角度α，变换到（x1，y1）位置，x1=x0·cosα- y0·sinα，y1=y0·cosα+ x0·sinα，其中，x0=R·cosθ，y0=R·sinθ

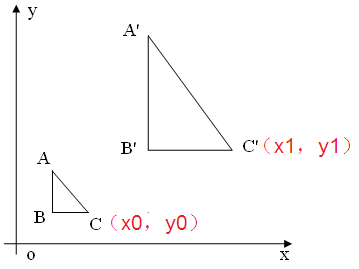
逆时针：



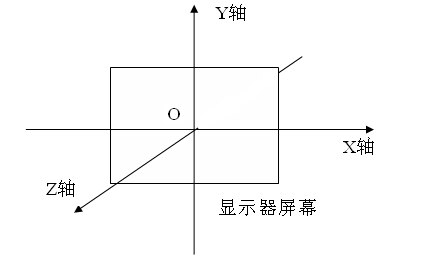
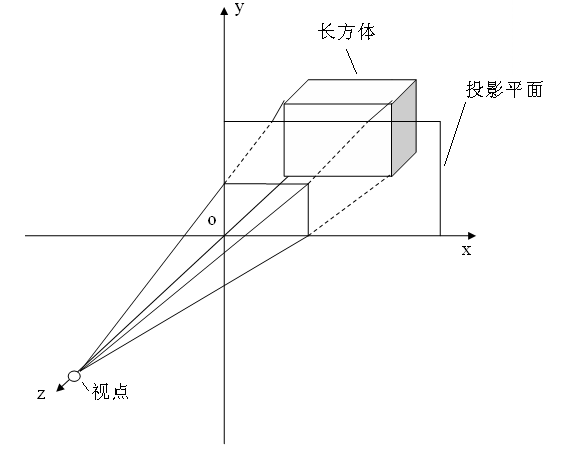
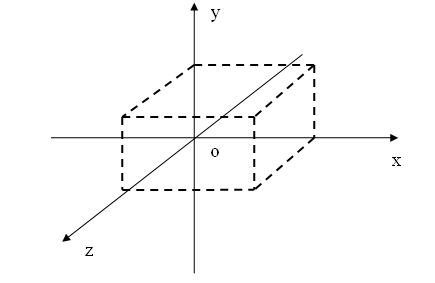
顺时针：



③**变比变换**：在二维空间，将多边形按比例因子(Sx, Sy)以坐标原点为中心进行放大或缩小的变换，x1=x0∙Sx，y1=y0∙Sy。

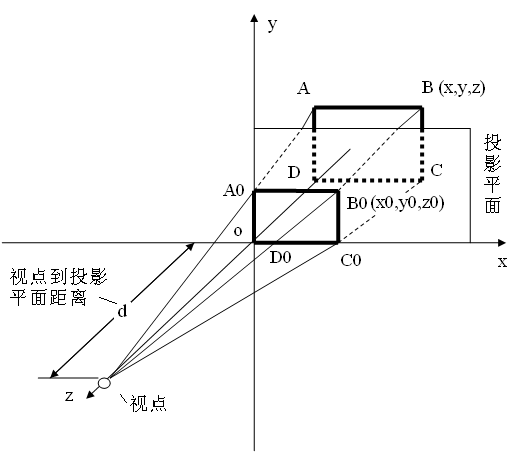


1. **Transform3D**类：用来执行平移、旋转、变比例等坐标变换。
2. **TransformGroup**类：所有表示各种几何变换的Transform3D类的对象必须包含在一个TransformGroup类的对象中，表示对在该TransformGroup类的对象中的三维图形所进行的坐标变换。
3. **Transform3D**类与TransformGroup类之间的关系：①如果表示一种几何变换的Transform3D对象没有加入到该TransformGroup类的对象中，则该变换对在该TransformGroup类的对象中的三维图形不起作用。②也可以一次定义一个总的坐标变换矩阵，并以此创建一个Transform3D对象，将该Transform3D对象加入到TransformGroup类的对象中。③也可以分步定义多个不同类型的坐标变换矩阵，然后依次创建多个Transform3D对象，每个Transform3D对象表示一种坐标变换，将这些Transform3D对象依次加入到一个TransformGroup类的对象中，则以Transform3D对象的加入次序，依次对三维图形进行变换，Java 3D系统形成最后总的坐标变换。④如果在一个TransformGroup节点中没有定义任何的坐标变换，也就是没有加入任何的Transform3D对象，则该TransformGroup节点拥有一个缺省的4\*4坐标变换矩阵，但该矩阵为单位矩阵，表示对几何体不进行任何几何变换。
4. **世界坐标系、观察坐标系与Java 3D显示器坐标系**：

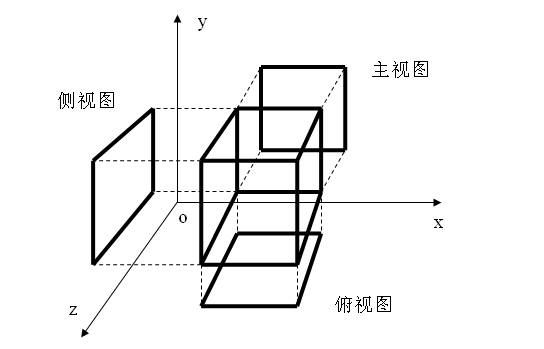


观察坐标系是在世界坐标系的基础上变换来的，显示器坐标系中显示的物体是在观察坐标系的投影平面上三维几何体的投影。

1. **透视投影**：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。



1. **平行投影**：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。



1. **隐藏线消除**：被前面的可见面隐藏的所有边都是不可见的，不用将这些边画出来，或者画成虚线形式。
2. **隐藏面消除**：只需显示可见的面，无需显示隐藏的面
3. **法向量计算**：



1. **Z Buffer隐藏面消除算法**：P228
2. **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 ②夹角之和检验法 ③两向量夹角计算

1. **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，不考虑场景中其它几何体的影响，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。

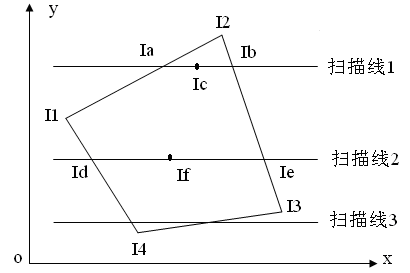
局部光照明计算模型只考虑物体表面对直射光、环境光的反射、散射与透射。

1. **Gouraud**算法：

Gouraud算法基本过程如下：

（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。

1. 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。
2. 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。
3. 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象素点的光颜色、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。



Gouraud算法光强度双线性插值算法示意图



1. **Phong**算法：

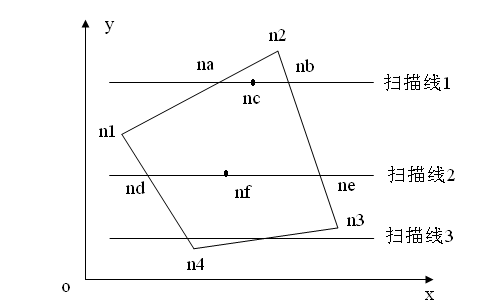
Phong算法计算过程如下：

（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。



Phone算法法向量双线性插值示意图







1. **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。
2. **Switch**类：Switch节点主要控制哪些节点将被显示。
3. **OrderedGroup**类：可确保在一个场景图中的三维形体的显示顺序。
4. **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。
5. **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。