基于 Java 3D 的地球空间环境可视化研究

张健康，杨宜康，李　雪，刘　磊
（电子科技大学 空天科学技术研究院，成都 611731）

摘　要：根据空间科学研究对交互性和 Web 化的发展需求，将 Java 3D 应用于该领域。深入剖析了 Java 3D 的主要几何类和实现细节；改进并实现了一种矩形网格等值线生成算法，解决了二义性和等值点在网格顶点的问题；介绍了一种生成规则网格的简单算法，并选用合适的 Java 3D 类加以实现；介绍了一种数值——颜色映射算法，并将空间环境数据映射为 RGB 颜色信息。综合运用这些算法，对地球空间环境中若干物理要素模型进行了二维和三维可视化。可视化结果与国内外空间环境模型研究成果基本一致，说明上述算法正确，有较高的参考价值。

关键词：空间环境；Java 3D；等值线；网格；数值颜色映射；可视化
中图分类号：TP39　　**文献标志码**：A　　**文章编号**：1001-3695(2013)01-0211-04
doi:10.3969/j.issn.1001-3695.2013.01.054

Research on geospace environment visualization based on Java 3D

ZHANG Jian-kang, YANG Yi-kang, LI Xue, LIU Lei
(*Institute of Astronautics & Aeronautic, University of Electronic Science & Technology of China, Chengdu* 611731, *China*)

Abstract: According to space science research's development and requirement on interaction and Web, this paper applied Java 3D to this field. It dived into several common Java 3D geometry classes and details. It improved and implemented one rectangular net isoline tracing algorithm, solved the ambiguity problem and the case when isoline points appeared at grid vertexes. It introduced one simple method to generate regular net and implemented it with proper Java 3D class. It introduced one algorithm to map numeric value to color and mapped space environment data into RGB information. It also visualized some geospace environment physical elements models both in two-dimension and three-dimension based on these algorithms. Results are in line with those research achievements both at home and abroad. This verifies these algorithms' correctness and high reference value.

Key words: space environment; Java 3D; isoline; grid; value-color mapping; visualization

Java 3D 提供了一个基于 OpenGL 或 DirectX 的交互式三维图形 API，是 Java 在三维图形领域的扩展。它已广泛应用于多个行业和领域。淮永建等人[1]将 Java 3D 应用于三维地形可视化研究，用场景图管理、更新和渲染数据。张平等人[2]构造了基于 Java 3D 的空间机器人运动仿真系统，实现了空间机器人的碰撞检测。然而，目前 Java 3D 的应用研究大多只介绍其主要数据结构，即场景图(scene graph)和程序框架的创建过程，忽略了所用几何类的细节。另外，目前地球空间环境科学领域主要关注物理要素模型本身的研究[3,4]和可视化技术及工具的研究[5,6]。其中，MATLAB、IDL、OPENDX 等可视化工具均有其长处，但尚不能较好地胜任交互性和基于 Web 的可视化要求。相比之下，凭借 Java 成熟的网络技术和框架，Java 3D 能够较出色地完成基于 Web 的 3D 可视化和交互任务。因此，研究基于 Java 3D 的空间环境可视化技术不仅拓展了 Java 3D 的应用领域，对空间科学的研究和发展也有较好的借鉴意义。

1　常用几何类详解

用 Java 3D 可视化时一般通过几种几何类构造对象：点、线段、三角形、四边形、圆锥、圆柱和球等。如需绘制一个正方体，应先构造六个正方形，再为它们指定合适的位置以使其恰好构成一个正方体。GeometryArray 抽象类用来构造几何对象，其子类的对象均由顶点构成。如线段由两个顶点构成，三角形由三个顶点构成等。这些顶点包含位置信息(坐标)和颜色信息(RGB 索引值)。Java 3D 能自动填充顶点间和多边形内部的颜色。这种内插机制使颜色平滑过渡，呈现较好的渲染效果。GeometryArray 类的部分层次关系[7]如图 1 所示。

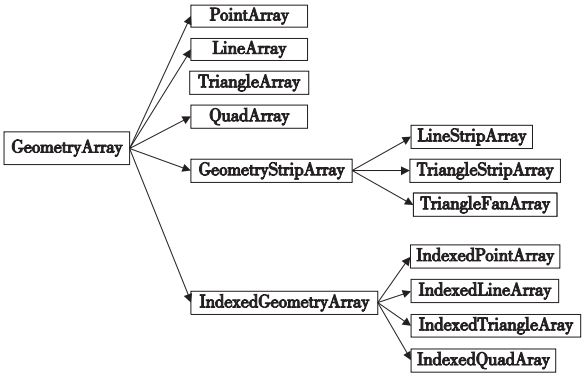

图1　GeometryArray类层次

收稿日期：2012-05-10；**修回日期**：2012-06-23

作者简介：张健康(1987-)，男，河南安阳人，硕士研究生，主要研究方向为计算机三维可视化(zjk0501@163.com)；杨宜康(1974-)，男，教授，博士，主要研究方向为测控通信与导航技术、编队卫星星间精密测量与授时理论及算法；李雪(1981-)，女，副教授，博士，主要研究方向为测控通信与导航技术、编队卫星星间精密测量与授时理论及算法；刘磊(1985-)，男，讲师，博士研究生，主要研究方向为卫星导航、星间精密测量与授时.

GeometryArray 子类分为三种：

a) PointArray、LineArray、TrangleArray 和 QuadArray。这些类的对象之间相互独立，无任何关联，构造图[7]如图 2 第一行所示。例如，构造独立的三角形应选取 TriangleArray 类，调用构造方法：TriangleArray(int vertexCount, int vertexFormat)。其中：vertexCount 是所要构造对象的顶点总数，这里要构造三角形，所以它必须是 3 的整数倍，若构造一个三角形则其值为 3，若构造两个三角形则其值为 6，且这两个三角形相互独立，无任何公共顶点或边；vertexFormat 一般选择 COORDINATES 和 COLOR_3，前者表示定义顶点之后需要调用 setCoordinate( )等方法设定顶点坐标，后者表示需要调用 setColor( )等方法向该点填充颜色且无透明度设置。

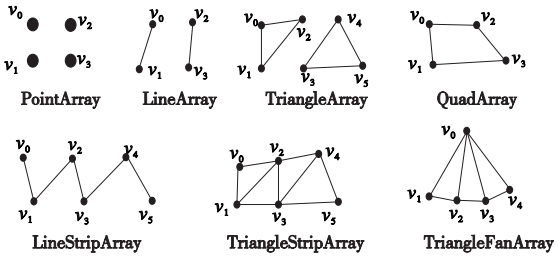

图2 部分几何类构造示意图

b) GeometryStripArray 抽象类及其子类。该类的对象之间可共用顶点。因此其三个子类 LineStripArray、TriangleStripArray 和 TriangleFanArray 相邻对象间可共用顶点或边，构造图如图 2 第二行所示。当构造有公共边的三角形时应选取含有 strip 的类 TriangleStripArray，调用构造方法：TriangleStripArray (int vertexCount, int vertexFormat, int stripVertexCounts[ ])。前两个参数与 a) 中相同，第三个参数表示传入一个整型数组，该数组的元素依次表示所建立的每个图形中的顶点个数。例如，若构造有一条公共边的两个三角形，vertexCount 为 6，vertexFormat 值同上，stripVertexCounts[ ]则为{3,3}，表示所建立的两个图形(这里是两个三角形)均有 3 个顶点。

c) IndexedGeometryArray 抽象类及其子类。由 b) 可知 strip 类的对象要想使用公共边或顶点，必须对这些顶点重复定义。该类解决了重复定义的问题。仍以构造两个共用一边的三角形为例。调用构造方法 IndexedTriangleArray(int vertexCount, int vertexFormat, int indexCount)。两个相邻的三角形在几何上共有四个顶点，所以这里的 vertexCount 值为 4。这种几何类把重复定义顶点的工作交给了 indexCount 参数。该参数是所有顶点的总和，每个重复使用的顶点都应计算在内。此例中公共边上的两个顶点均被使用两次，vertexCount 是所有顶点的和，值应为 6。

有四个问题需要特别注意：

a) Java 3D 默认使用 x、y、z 直角坐标系。其中 x 轴正方向指向显示屏右侧，y 轴正方向指向显示屏上方，z 轴正方向垂直于显示屏朝外。

b) Java 3D 不提供 QuadStripArray 类，即现有的类不能直接构造出共用一边的两个四边形。

c) 如需构造带有 strip 的对象，vertexCount 必须等于 stripVertexCounts 数组中所有元素之和。这意味着，即使两个对象共用一个顶点或一条边，那么被共用的顶点仍需重复定义，只不过两个顶点是在同一个位置，它们的坐标和颜色信息完全相同。这就是为什么上述 TriangleStripArray 例子中 vertexCount 为 6 而不是 4。可见，虽然 Java 3D 提供了可共用顶点的类(含 strip 或 Indexed)，但所定义的顶点数必须达到实际所用的数量。Strip 类中该数量由 vertexCount 参数指定，stripVertexCounts 数组分配给每个对象图形；Indexed 类中由 indexCount 参数指定。

d) Java 3D 在绘制三角形、四边形时按照顶点定义的先后顺序，以逆时针方向进行，即右手准则。右手大拇指指向 z 轴正方向，其余四指以逆时针方向依次经过各个顶点形成一个凸多边形，其内部为渲染区域。

2 算法分析

2.1 等值线生成算法

孙桂茹等人[8]提出了四点矩形区域划分法，蒋瑜等人[9]提出了基于 Delaunay 三角网的算法。显然，对地球空间环境中常以经纬线划分网格而言，四点矩形划分法更易操作。孙桂茹等人给出的网格内八种等值线连接方法很好地总结了网格单元内等值线的分布情况，但并未说明等值线进入和离开网格的具体方向。于嘉等人[10]给出一种实用的等值线数据结构和算法，用二进制标志等值线进入情况和等值点存在情况有较高的参考价值，但实现上不够简练。这里提出更简单、更易实现的数据结构和算法。

1) 数据结构　网格节点、等值点、网格的数据结构分别定义为

```
class GridPoint{int x,int y,float value}
class ContourPoint{int x,int y,
boolean isEdge}
class Grid{GridPoint gridPoint[4],
ContourPoint contourPoint[4],
float contourvalue,
int in,int out,
boolean isTraced}
```

其中：gridPoint 数组表示某网格的四个节点，gridPoint[ ].x 和 gridPoint[ ].y 指定节点位置，gridPoint[ ].value 指定节点数值；contourPoint 数组表示该网格四条边上的等值点(如果存在)，contourPoint[ ].x 和 contourPoint[ ].y 指定等值点位置( )，contourPoint[ ].isEdge 指示等值点是否处于边界；contourvalue 为经过该网格的等值线的值；in 和 out 分别表示经过该网格的等值线由哪边进哪边出，取值是{1,2,3,4}；isTraced 表示该网格是否历经。

2) 实现算法　上述八种连接方法按等值线进出方向不同各有两种情况，因此单个网格中等值线追踪一共有十六种情况。定义单网格对象追踪方法 int trace(int in, float contourvalue)。追踪步骤为：

a) 根据 in 值指定的入方向和上述十六种情况判断出方向，设置 out 值；

b) 用线性插值法求出该出方向所在边上等值点的坐标；

c) 判断并设置 isEdge 和 isTraced 的值；

d) 返回 out 作为下一网格的输入；

e) 按指定的追踪顺序依次遍历各个网格，当遇到某网格 isEdge 或 isTraced 为真时停止。

等值线生成有两个问题较难解决，即二义性问题[10~12]和当等值点位于网格节点时的问题。前者可用四个顶点均值判断解决[10,12]。对于第二个问题，目前等值线追踪算法研究中较少提及，这里采用一种折中办法。当等值点位于网格某个节点时，等值线应该穿过该节点，这会影响后面网格的追踪判断。因此，在不影响整体的前提下，不妨对该节点值做微小改动，即加上或减去一个很小的数使其偏离等值线。此时追踪算法就

可以按普通情况处理该网格。这种做法有悖于科学研究的准确性,但大大利于工程实践操作,而且能自由调整偏离程度,保证误差在合理范围内。

2.2　数值—颜色映射算法

地球空间环境可视化是将物理要素值以相应颜色展现出来。Java 3D 不能识别这样的数值,因此需要一种算法将数值映射为颜色信息。Java 3D 用三个浮点型或字节型值确定颜色信息,它们分别控制对应的 R、G、B 值,由 Color3f(float r,float g,float b)或 Color3f(byte r, byte g,byte b)类的对象完成。当 r、g、b 是浮点型时,取值为 0 ~ 1;当它们是字节型时,取值为 0 ~ 255。每种颜色分量都有 256 个值的字节型比小数更易操控,所以这里选用字节型。

通常,地球空间磁场或辐射环境可视化时,强度弱的地方使用蓝色,强的地方使用红色。因此,强度值数组中的最小值应映射为蓝色,最大值应映射为红色。由计算机图形学 RGB 颜色模型[13]可知(0,0,255)表示蓝色,(255,0,0)表示红色,中间的颜色为绿色,值为(0,255,0)。特别注意,蓝色和绿色的中间颜色为青色(0,255,255),绿色和红色的中间颜色为品红(255,255,0)。因此,强度值从小到大按如下顺序分阶段均匀映射:(0,0,255)—(0,255,255)—(0,255,0)—(255,255,0)—(255,0,0)。算法伪代码为:

```
step = (max - min)/4
microstep = step/255
c1 = min + step
c2 = c1 + step
c3 = c2 + step
if value > = min and value < c1
   then r = 0;g = (value - min)/microstep;b = 255;
if value > = c1 and value < c2
   then r = 0;g = 255;b = 255 - (value - c1)/microstep;
if value > = c2 and value < c3
   then r = (value - c2)/microstep;g = 255;b = 0;
if value > = c3 and value < = max
   then r = 255;g = 255 - (value - c3)/microstep;b = 0;
```

3　应用剖析

3.1　曲线可视化

3.1.1　一维曲线

地球空间环境研究中,用曲线统计分析物理要素的数据非常普遍。一般地,选取 LineStripArray 类最合适。只需设定好要绘制的点坐标,Java 3D 能将所定义的各顶点自动连成曲线。以最常见的地磁场强度可视化为例进行说明。这里的数据由国际地磁参考场模型(IGRF 模型)[3~5,14]计算得到。

图 3 中横轴表示经度,-180 表示西经 180°,179 表示东经 179°;纵轴表示磁场强度值(单位 nT)。横坐标为 -180 时,自上而下依次是南纬 30°、北纬 30°和赤道上磁场强度随经度的变化曲线,分别用绿色、红色和蓝色线画出(见电子版)。图 3 按照经度步径为 1 取得,如果步径更小或磁场强度值变化剧烈,则能生成折线图。

3.1.2　等值线生成

借助 Java 的 ArrayList 或 LinkedList 类,用 LineStripArray 也可以实现如图 4 所示的等值线或等高线。现以 ArrayList 类为例说明等值线生成步骤:

a)根据等值线生成算法查找等值点,将追踪到的等值点坐标添加至 ArrayList 类的对象 list。

b)根据 list 的长度与 LineStripArray 类的参数要求,构造 LineStripArray 对象 line。如 list 长度为 size,则调用构造方法 LineStripArray(int vertexCount, int vertexFormat, int stripVertexCounts[ ])构造对象 line。如图 2 所示的中构造图,list 两个端顶点只需定义一次,其他顶点均需定义两次,所以顶点数 vertexCount 值应为 size + size - 2,vertexFormat 值为 LineStripArray. COORDINATES | LineStripArray. COLOR_3, stripVertexCounts 数组大小为 size - 1,每个元素值均为 2,表示每个图形(这里是线段)中含有两个顶点。

c)用 list 的元素为 line 对象的 size + size - 2 个顶点设置坐标。

d)把参考该 line 的 Shape3D 对象加入场景图进行渲染。渲染结果如图 4 所示。

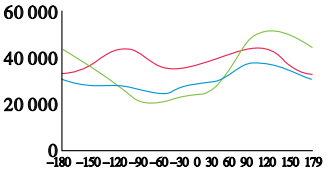

图3　不同纬度下地磁场强度值随经度变化曲线(时间:2000.1.1,高度:距地心6 400 km)

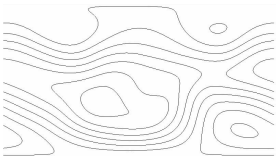

图4　地磁场强度值等值线(时间:2000.1.1,高度:距地心6 400 km)

3.2　二维可视化

空间环境研究中常需查看物理要素的全球或地区分布情况。二维平面图虽不能完全反映要素的各个特征,但因其简单高效而应用广泛。气象预报中的动态云图就是一个典例。二维可视化实质是渲染若干相邻的小三角形、小矩形或小多边形来反映整体。以 IGRF 模型的全球分布可视化为例分析 Java 3D 在二维可视化中的应用。

首先沿经纬线建立规则的矩形网格。经线从西经 180 度到东经 179°,纬线从南纬 90°到北纬 90°,步径均为 1。这样可建立一个 181 × 360 个节点的网格阵列,如图 5 所示。

每个网格节点含有的数据信息为:经度、纬度、地磁场强度值。Java 3D 中可以方便建立矩形网格的几何类有 QuadArray 和 IndexedQuadArray。因两者的对象均不能共用顶点或边,故需重复定义网格阵列中大部分节点,用 QuadArray 实现较复杂,所以选用 IndexedQuadArray。该类实际只需定义与网格节点同等数量的顶点即可,重复使用顶点的操作工作留给参考点[7]完成。上述例子的可视化步骤如下:

a)定义三个 181 × 360 的二维数组分别存放经度、纬度、地磁场强度值,经纬度分别用于设置 x 和 y 的值,z 值取 0,表示这些顶点都在 xoy 平面上。

b)调用方法 IndexedQuadArray(int vertexCount, int vertexFormat, int indexCount)构造对象 quad。其中:vertexCount 是网格阵列中实际出现的顶点数,上述网格阵列中有 181 × 360 个节点,所以 vertexCount 值为 181 × 360;vertexFormat 与前面类似,其值为 IndexedQuadArray. COORDINATES | IndexedQuadArray. COLOR_3;indexCount 为参考点数,是渲染过程使用到的顶点总数。渲染该网格阵列需绘制 180 × 359 个矩形,每个矩形均要使用四个顶点。分析可知,该网格阵列中,位于四个角的节点渲染时只使用一次。四条边界中,除四个角节点外,其他节点因被两个矩形共用,渲染时要使用两次。剩下所有节点均被四个矩形共用,渲染时需使用四次。综上可知,indexCount 值为 181 × 360 + 2 × (181 - 2) + 2 × (360 - 2) + 3 × (181 - 2) × (360 - 2)。

c)用 setCoordinate 和 setColor 方法为 vertexCount 个顶点设

置坐标和颜色信息。

d)完成坐标和颜色设置后的关键步骤是设置参考点。不妨称由 vertexCount 指定的网格阵列中的 181 ×360 个节点为原点,indexCount 指定的那部分点为参考点。参考点与原点一样含有坐标和颜色信息,这些信息通过 setCoordinateIndex(int index,int coordinateIndex)和 setColorIndex(int index, int colorIndex)设置。其中,index 是要设置的顶点在全部 indexCount 个参考点中的索引值,coordinateIndex 和 colorIndex 是该顶点所参考原点的相应索引值。如图 5 所示的四个较大顶点构成的网格,该网格左下角的节点在第 i 行、第 j 列,它们在原点中的索引值如图 5 中的表达式所示。假设现在要渲染的某个矩形中四个顶点的 index 值分别为 1000、1001、1002、1003,coordinateIndex 值分别为 $i\times360+j$、$i\times360+j+1$、$i\times360+j+360$、$i\times360+j+360+1$。这四个值的先后顺序可变,只要保证在图 5 中是逆时针即可。设置坐标和颜色信息如下:

setCoordinate(1000, $i\times360+j$)
setColor(1000, $i\times360+j$)
setCoordinate(1001, $i\times360+j+1$)
setColor(1001, $i\times360+j+1$)
setCoordinate(1002, $i\times360+j+360$)
setColor(1002, $i\times360+j+360$)
setCoordinate(1003, $i\times360+j+360+1$)
setColor(1003, $i\times360+j+360+1$)

e)把参考该 quad 的 Shape3D 对象加入场景图进行渲染,渲染结果如图 6 所示。

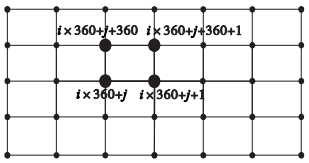

图5 网格阵列示意图

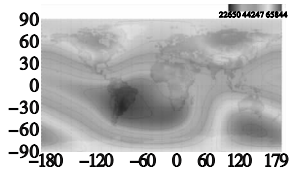

图6 地磁场强度值二维可视化结果

3.3 三维可视化

有时为了从不同角度观察空间环境,需将物理要素模型以立体方式呈现,即三维可视化。三维可视化与二维可视化类似,区别在于后者将渲染结果以平面形式呈现,而前者则以曲面或立体几何形体展现。现以 NASA 辐射带模型(Ae8Ap8 模型)[15~18]计算所得的地球空间辐射强度的全球分布可视化为例分析 Java 3D 在三维可视化中的应用。

Java 3D 中在赋予顶点相同颜色信息的前提下,它被置于哪个位置就会在相应位置显现。2.2 节中统一将顶点置于 xoy 平面内,则呈现一个如图 6 所示的平面。据此,如果能将顶点统一置于一个球面,那它们就会以球面的形式展现该物理要素。故三维可视化与二维基本相同,区别仅在于给顶点赋球面坐标值。注意这里所说的球面坐标有别于球坐标系。Java 3D 不直接支持 r、φ、θ 球坐标系,它本身是以三维空间直角坐标系为参考来绘制对象的。此处仍以 x、y、z 系为参考,只不过点 (x,y,z) 在一个球面上而已。因此,为顶点设置坐标前应进行坐标转换:

$$\begin{cases} x = h\times\cos\alpha\times\sin\theta \\ y = h\times\sin\alpha \\ z = h\times\cos\alpha\times\cos\theta \end{cases}$$

其中:θ、α、h 分别表示经度、纬度、距地心高度;x、y、z 是转换后的直角坐标。

原来位于平面的四个顶点现被置于球面,但它们之间相对位置不变,仍构成一个小矩形且内部颜色平滑过渡。因此,渲染一个球面和渲染一个平面的实质相同。渲染 180 ×359 个小矩形的分辨率足够呈现一个较光滑的球体。可视化结果如图 7 所示。

图7 Ae8Ap8三维可视化结果

4 结束语

将本文地球空间环境物理要素可视化仿真结果与领域内公认的结论[14,16~18]进行比较,结果表明本文对 Java 3D 主要几何类的深入剖析和相关算法运用是正确的,且可靠易实现。这为相关科研和工程人员提供了较高的参考价值。本文只介绍了物理要素模型的 Java 3D 本地实现,下一步工作重点将是实现基于 Web 的 3D 可视化。而且,鉴于程序执行速度较慢,相关数据结构和算法等也需要改进。

参考文献:

[1] 淮永建,于鹏,张倩倩. 基于 Java 3D 多分辨率 LOD 地形可视化研究[J]. 计算机仿真,2009,26(11):255-259.
[2] 张平,杨时杰,梁斌. 基于 Java 3D 的空间机器人运动仿真系统[J]. 计算机应用研究,2007,24(9):19-21.
[3] 白春华,徐文耀,康国发. 地球主磁场模型[J]. 地球物理学进展,2008,23(4):1045-1057.
[4] 徐文耀. 地球电磁现象物理学[M]. 合肥:中国科学技术大学出版社,2009:87-141.
[5] 王丹,彭丰林,马麦宁,等. IGRF 国际地磁参考场模型可视化研究[J]. 地震地磁观测与研究,2009,30(4):7-11.
[6] 李大林,李秀冰,李玉英,等. 地球空间环境要素可视化技术研究[J]. 计算机与数字工程,2008,36(8):31-34.
[7] SUN Microsystems. Java 3D API turorial[EB/OL]. (2000-09)[2012-03]. http://java.sun.com/developer/onlineTraining/java3d/index.html.
[8] 孙桂茹,马亮,路登平,等. 等值线生成与图形填充算法[J]. 天津大学学报,2000,33(6):816-818.
[9] 蒋瑜,杜斌,卢军,等. 基于 Delaunay 三角网的等值线绘制算法[J]. 计算机应用研究,2010,27(1):101-103.
[10] 于嘉,吴旭. 一种改进的矩形网格等值线追踪算法[J]. 河南师范大学学报:自然科学版,2008,36(6):34-36.
[11] 李楠,肖克炎,陈学工,等. 基于格网数据自动生成等值线程序的设计与开发[C]//2009 年全国数学地球科学与地学信息学术会议. 2009:413-419.
[12] 刘永奎,范柏乃. 基于构型表的等值线绘制算法及程序实现[J]. 南京大学学报:自然科学版,2008,44(4):371-378.
[13] HEARN D,BAKER M P. 计算机图形学[M]. 蔡士杰,宋继强,蔡敏,译. 3 版. 北京:电子工业出版社,2010:531-534.
[14] International Association of Geomagnetism and Aeronomy(IAGA), Working Group V-MOD. International geomegnatic reference field[CP/OL]. (2010-01)[2010-11]. http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html.
[15] 邹鸿,陈鸿飞,邹积清,等. 星内粒子探测器观测结果与辐射带模型的比较[J]. 地球物理学报,2007,50(3):678-683.
[16] VETTE J I. The AE-8 trapped electron model environment,NSSDC/WDC-A-R&S Report 91-24[R].[S.l.]:NASA-GSFC,1991.
[17] SAWYER D M,VETTE J I. AP8 trapped proton environment for solar maximum and solar minimum,NSSDC WDC-A-R&S 76-06[R].[S.l.]:NASA-GSFC,1976.
[18] BILITZA D. AE-8/AP-8 RADBELT[CP/OL]. [2010-11]. http://ccmc.gsfc.nasa.gov/models/modelinfo.php? model = AE-8/AP-8 RADBELT.