三维GIS渲染引擎的关键技术

曹国峰，张立立，钟耳顺
(中国科学院地理科学与资源研究所，北京 100101)

摘要：随着计算机图形技术的不断发展和人们对视觉要求的不断提高，三维GIS场景的复杂度也在增大。本文在分析三维GIS场景特点的基础上，提出了三维GIS渲染引擎的概念，并分析了渲染引擎在降低三维GIS开发复杂度和提高软件可重用性中的作用。重点探讨了3DGIS中，针对大数据量复杂实时场景进行快速、高效渲染的关键技术，如场景图、层次细节、视锥体剔除、遮挡面剔除、渲染通道等，并对技术原理进行了详细的分析，最后，通过具体的实验验证了这些技术的可用性和有效性。

关键词：三维GIS；三维渲染引擎；层次细节；剔除技术；渲染通道
中图分类号：P208

1 引言

近年来，传统的二维地理信息系统(二维GIS)得到了长足的发展，但二维GIS将现实世界简化为平面上二维投影的概念模型注定了它在描述三维空间现象上的无能为力。克服这一缺陷迫切需要深入研究三维空间的GIS[1]。

随着数据采集能力的不断提高，可供GIS处理的数据不断膨胀，覆盖全球的1km分辨率的数据集已经可从Internet上免费获取，能够覆盖全球大部分地区的30m分辨率的数据集也在准备中(http://www.jpl.nasa.gov/srtm/)[6]。如此庞大的数据量大大的提高了场景的复杂度。要保证渲染的实时性(每秒钟大约产生20帧左右的图像[4])和真实感，图形硬件性能的提高，无疑起着重要的作用。但是目前图形硬件的处理能力提高的水平远远赶不上数据量增大的幅度[2]。这要求我们必须通过软件的方法，设计和实现高效的算法，在不影响场景渲染真实感的条件下，尽可能的降低场景的复杂度。

为了提高软件的复用性，减少重复劳动，屏蔽实现细节，方便三维系统的建立，我们提出了三维GIS渲染引擎的概念。作为三维GIS系统的核心部分，引擎必须提供统一、方便、全面、高效的渲染接口，从而对GIS中的各种三维场景进行渲染、组织和管理。

三维GIS场景是比较复杂的，可包含自然界中一切可渲染的具体的或抽象的物体，如房屋、楼盘、道路、树木、地形、天空、云、雨，以及应用于各种科学计算的抽象模型等。

伴随数据量的增大和场景复杂度的不断提高，对三维GIS渲染引擎的研制要求对计算机图形学中各种算法进行深入研究，探讨各种场景简化技术在三维GIS中的应用。

2 三维渲染引擎的关键技术分析

(1) 三维渲染引擎是三维GIS的核心组成部分。场景图是解决这个问题的有效手段。它一般是树形数据结构，以树的方式对复杂的场景进行有序组织。场景组织者可以根据自己的需要设计自己的具体结构，这种结构特别适用于对层次化结构场景的管理。在Java3D和VRML中已经得到成功的应用[7]。

我们把场景按其可渲染物体的关系抽象成可重复划分的空间，每一个可渲染的物体都对应着一块空间。每个空间通过SceneNode来管理，SceneNode处理空间的移动、旋转、缩放和空间相关的行为，最后把要渲染的物体连接到SceneNode对应的空间

收稿日期：2004-04-19.
作者简介：曹国峰(1980-)，男，中国科学院地理科学与资源研究所硕士研究生，主要研究方向：地学可视化，计算几何在GIS中的应用，空间分析等。

上。每个 SceneNode 对应着树结构里面的一个节点保存与父节点的相对位置和指向子节点的指针。伪代码如下：

```
class CSmSceneNode
{
  CSmSceneNode *m_pChildNode; //子节点的指针
  CVector m_vPos;          //相对于父节点的位置
  CVector m_vOrientation;  //相对于父节点的方向
  CVector m_vScale;        //节点的缩放因子
  Void    SetPosition(CVector vPos);//设置节点位置
  Void    Translate(CVector vTraslate); //平移
  Void    Rotate(CVector vAxis,Real fDegree);
                           //绕 vAxis 旋转
  Void    Roll(fDegree); //绕 Z 轴旋转
  Void    Yaw(fDegree); //绕 Y 轴旋转
  Void    Pitch(fDegree); //绕 X 轴旋转
  ……
}
```

结构如图 1：

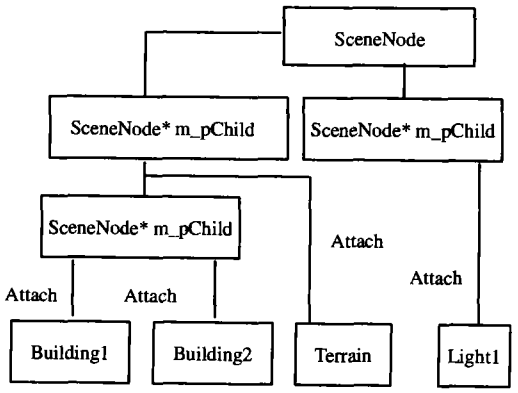

图 1 场景图组织结构

Fig.1 Structure of a scenegraph

经过这样的组织,在渲染的过程中,就可以按照场景图的组织，按照深度优先的顺序对各个物体进行渲染,从而保证了渲染的有序性。

另外,每一个节点中记录着相对于父节点的位置变化，这样父节点和它对应的子节点之间就有层次关系,父节点的各种变换都会有相应的影响子节点。

(2) 层次细节(LoD)。地形是一个三维场景的重要组成部分。大地形场景的数据量一般较大,能否对其进行高效显示是 3DGIS 渲染引擎成功与否的一个重要标志。

对于大地形数据一般通过多分辨率技术进行地形化简,再进行渲染。地形化简作为三维网格简化技术在地学中的应用的算法不断地提出。但这些算法都是基于这样一个事实:距离视点远和相对比较平坦，起伏不大的部分，采用相对比较低的分辨率,而对于距离视点比较近而起伏比较大的部分则采用比较高的分辨率。这样就大大避免了不必要的计算,从而在保证帧速率的条件下,也保证了渲染的质量。

GIS 中的地形一般采用规则网格(GRID/DEM)和不规则网格(TIN)表示,地形简化也根据不同的表示方式而采取不同的简化算法。但是由于 TIN 的数据结构比较复杂，并且每调整一次分辨率都要涉及到大量的三角剖分工作[11],计算量较大,所以在实际的应用中多采用 GRID/DEM 的规则网格表示。

基于规则网格的方法多数是在四叉树的基础上发展起来的,应用最广泛的有 Peter Lindstrom 的连续层次细节方法 [8],Mark Duchaineauy 的 ROAM 方法[9],以及 Stefan Röttger[10]的 QuccdTree 方法。这些方法各有利弊,对此，根据 ROAM 的缺点,对其进行了改进。这种方法采用二叉三角树作为基本的数据结构来组织地形，并为三角形提供了分劈(Split)和合并(Merge)两种操作来控制场景中各个位置的分辨率(如图 2)。

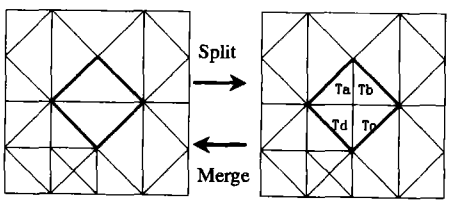

图 2 三角形的 Split/Merge 操作

Fig.2 Split/Merge operation of a triangle

如前所述,有 2 个因素决定着显示需要的分辨率,也就是对三角形进行 Split 还是 Merge 操作:

①三角形和视点之间的距离 S;

②采用三角形底边中心的高程值与底边两顶点高程的平均值之 δv 差。如图 3 所示：

$$\delta v = v - \bar{v} \tag{1}$$

其中 $$\bar{v} = (v_l + v_r)/2 \tag{2}$$

$$\text{设 } f = \delta_v \times C/S \tag{3}$$

其中:C 为常数,给定阈值 t,对每个要显示的三角形计算 f, 当 $f>t$ 时,则需要对三角形进行 Split 直到

每个子三角形的 $f \leqslant t$,再进行显示。

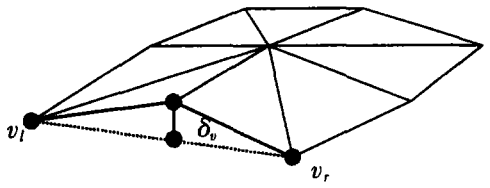

图 3　三角形的高程差(文献[9])

Fig.3　Height difference in a triangle

在不断的 Split 和 Merge 操作过程中,最关键的是在控制三角形显示与否的同时能保证没有如图 4 所示的裂缝的产生。

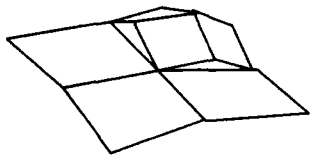

图 4　由于层次不同产生的裂缝

Fig.4　Cracks caused by different levels

问题的解决需要我们在不断得 Split 的过程中,制定一定的规则保证 Split 的顺序,防止裂缝的产生。除了地形以外,对于场景中的复杂模型也可以进行网格简化从而进一步减少显卡的负载,提高渲染的速度。

(3)剔除技术。在一个复杂度较高的三维场景中,要对每个物体进行渲染对于现在的图形渲染硬件来讲是比较困难的,也是没有必要的。在渲染过程中,真正有意义的是处于视域范围内真正可以看到的物体,看不到的物体则完全可以剔除。为此,图形学引入视锥体的概念,用来表征视点对应的视域范围。视锥体是一个六面体,如果物体的任何一部分在视锥体内则认为物体可见,反之,则认为不可见,如图 5 所示:物体 B,C 可见,而 A 则不可见。

对于 OpenGL，可以通过投影矩阵和模型矩阵来得到视锥体的方程,求得视锥体伪代码如下:

```
CalculateViewFrustum()
{
    glGetFloatv (GL_PROJECTION_MATRIX, Projection Matrix); // 得到投影矩阵
    glGetFloatv(GL_MODELVIEW_MATRIX, Model Matrix);  // 得到模型矩阵
    ResultMatix = MatrixMulti (Projection Matrix, Model Matrix); // 将两个矩阵相乘,再从得到
                                                    // 结果矩阵中得到视锥体各面方
                                                    // 程的系数
}
```

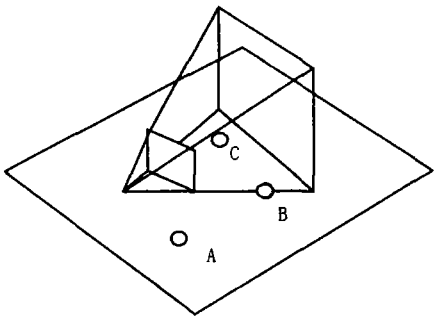

图 5　视点 V 对应的视锥体

Fig.5　View frustum corresponding to the viewpoint V

场景中的所有物体,地形或者房屋建筑在渲染之前首先进行视锥体剔除,剔除的方法是按物体的包围盒(BoundBox)进行。对包围盒的六个面分别进行判断,如果所有的面都在视锥体的外面,则该物体就不必进行渲染。

对于遮挡面剔除在观察者面前,体积较大的物体可以完全遮挡住很多体积小的物体。这时候那些完全被遮挡的物体在渲染过程中可以完全不被考虑。如图 6 所示:

V 为视点,物体 F 将物体 A,B 完全遮挡。

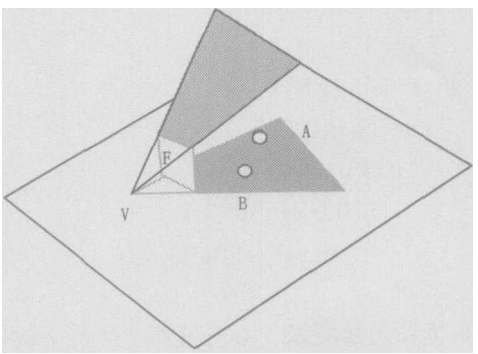

图 6　遮挡面 F 对应的遮挡区域

Fig.6　The region occluded by plane F

给定视点 V 和遮挡多边形 P,把三维物体投影到平面上的图形如图 7 所示:多边形 ABCD 就是 V 和 P 形成的遮挡范围,完全落在 ABCD 范围里的物体在渲染过程中则由于遮挡都可以剔除。

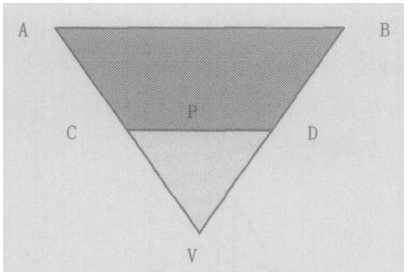

图 7 ABCD 为遮挡区域在平面上的投影

Fig.7 ABCD is the projective result of the occluded region

一般来讲,随着场景复杂度的提高,相互遮挡的频率也会提高,所以遮挡面剔除特别适合于对大型复杂场景的绘制。

遮挡面剔除的具体算法一般先建立一个与实际地形对应的平面网格,网格的每一个单元都相对应一块相应的实际地形, 在对每一帧进行渲染时,选择遮挡面,并对这些遮挡面进行绘制,把在帧缓存中对应的深度值读出存于建立的网格中,场景其他物体的绘制与否则决定于其高程与对应网格位置的高程关系,如果小于物体高程,则无需绘制。

综合视锥体剔除和遮挡面剔除,绘制过程的伪代码如下:

```
void DrawBuildings()
{
    T = CTile;
    VF = CviewFrustum;
    Building = Cbuilding;
    OS = CoccludeShadow;
    For Each T
      If  T  is in VF
        For Each B in T
          If B is in VF
            If(B is not in OS)
              DrawEachBuilding();
}
```

(4)渲染通道技术。由于场景中的每个物体,都有不同的材质,纹理,对于采用的场景图结构,就有一个很大的缺点: 在对不同的物体进行渲染时,需要在不同的渲染状态之间进行切换。而这对于 3DAPI 来说是非常费时、低效的。一个较好的解决方案是按照纹理来组织渲染物体。并按照一定的规则,如贴图纹理的数量,进行排序,这样在渲染的过程中, 就可以尽可能的减少 3DAPI 渲染状态的切换,从而提高渲染的效率。

伪代码如下:

```
Typedef  std::vector <Renerable* >  Renderable List; //某个文理对应的渲染物体的数组
Typedef  std::map  <Material*,RenderableList >  Material List Map; //纹理与其对应的渲染物体
```

3 应用实验结果

笔者在 3DGIS 软件 Super3D 中对上述的各项关键技术进行了实验, 并取得了较好的效果。

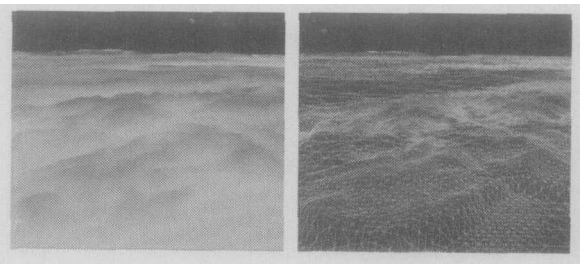

图 8 广东某地区 6000×6000 的 DEM LoD 显示效果的实体模型和线框模型(从右图中可以看出线框密度随着距离的增加而减小)

Fig.8 A scene of terrain visualization using LoD technology. The 6000×6000 DEM data represent someplace in Guangdong province.From the wireframe model,we can see that the triangular density decrease with the increase of distance

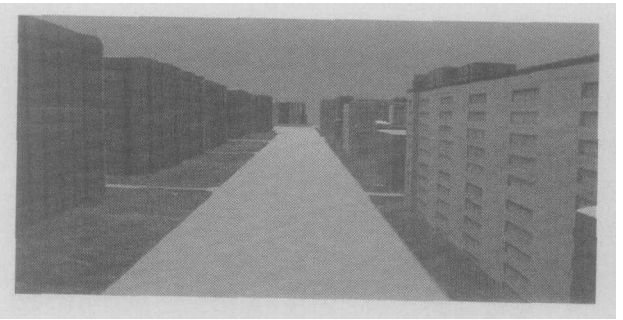

图 9 某城市街道三维场景

Fig.9 A 3D scene in one city

图形显示设备性能的不断提高和 3DAPI 的不断升级使得我们可以更加有效的控制整个渲染过程的每个细节(vertex shader,pixel shader),并且可以对显存进行操作(vertex buffer),这样就大大的提高了显示的真实感和渲染的效率。如何充分的利用这些功能来提高 GIS 场景的渲染效果,完善我们的渲染引擎有待深入研究。

4 结语

渲染引擎作为3DGIS的核心，把人们从复杂的底层技术细节中解脱出来，降低了3DGIS的复杂度，并且提高了系统的可重用性。随着计算机图形学的不断发展，各种新算法，新思路不断提出，并在3DGIS中得到应用。笔者结合Super 3D渲染引擎的设计实践，通过对面向大数据量，复杂场景的三维渲染引擎中的几项关键技术应用实验取得了较好的渲染效果。

参考文献

[1] 肖乐斌，钟耳顺. 三维GIS的基本问题探讨.中国图象图形学报,2001,6(9).

[2] 王永明.地形可视化. 中国图象图形学报,2000,5(6).

[3] 肖金城，李英成. 大规模地形场景三维实时漫游显示技术研究.遥感信息,2002.

[4] 蒋丁华，谭 兵. VR技术中地形场景的实时显示.测绘学院学报,2001, 18(z1).

[5] 陈 刚，夏 青，万 刚. 地形RSG模型的实时动态构网算法的设计与实现.测绘学报, 2002, 31(1).

[6] Aasgaard R, T Sevaldrud. Distributed handling of level of detail surfaces with binary triangle trees. Proc. ScanGIS, 2001, 45~58.

[7] Steve Cunningham. Lessons from scene graphs: Using scene graphs to teach hierarchical modeling. Computers & Graphics, 2001,25(1).

[8] P Lindstrom, D Koller, W Ribarsky, L F Hodges, N Faust, G A Turner. Real-time, continuous level of detail rendering of height fields. In Proceedings SIGGRAPH 96, ACM SIGGRAPH, 1996: 109~118.

[9] M Duchaineau, M Wolinsky, D E Sigeti, M C Miller, C Aldrich, M B Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In Proceedings Visualization 97, IEEE, Computer Society Press, Los Alamitos, California, 1997: 81~88.

[10] Stefan Röttger,Wolfgang Heidrich, Philipp Slasallek, Hans-Peter Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. ftp://faui90.informatik.uni-erlangen.de/pub/Publications/1998/Publ.1998,2. ps. gz, 1998,

[11] Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In Proceedings Visualization 98. IEEE Computer Society Press, 1998: 19~26, 515.

A Discussioin on Key Techniques in 3D GIS Rendering Engine

CAO Guofeng, ZHANG Lili, ZHONG Ershun

(Institute of Geographic Sciences and Natural Resources Research ,CAS, Beijing 100101, China)

Abstract: With the fast development of Computer Graphics and the increase of visual need of the people,3DGIS has become the main trend in GIS. But the way people can get spatial data has been improved greatly which made the explosion of the spatial data that GIS can handle and increased the complexity of the 3DGIS scenes. Besides, to be real-time is also a must in 3D system,all of these challenge the research and development of 3DGIS software.In this situation, based on the analysis of the key features of the 3DGIS, this article presented the definition of the 3DGIS Rendering Engine, and offered some key techniques to render complex, large scale 3DGIS scenes, such as SceneGraph, Level of Detail, View Frustum Culling, Occulding Culling, Rendering Pipe, etc. The author also explained clearly the theory behind them and proved the efficiencies of these techniques through experiment. Besides that, this arcticle presented that 3DGIS Rendering Engine, as the core of 3DGIS, can also decrease the difficulties of the development of 3DGIS and increase the reusability of the software.

Key words: 3DGIS; 3D rendering engine; level of detail;occluding technique;render pipeline