基于限制四叉树的大规模地形可视化及其实现*

殷　宏[1,2]，许继恒[1]，周良伟[1]，刘世彬[1]，黄大节[1]

(1.解放军理工大学 工程兵工程学院，江苏 南京 210007；2.南京理工大学 计算机系，江苏 南京 210094)

摘　要：论述了一种基于限制四叉树的大规模地形实时动态构网算法,实现了在模型误差控制下视点相关的多分辨网格的实时正确构网。实验证明,该方法对于实时控制大规模地形模型的细节层次,增强大规模地形模型的绘制效率是非常有效的。

关键词：限制四叉树；节点；视点相关

中图法分类号：TP391　　**文献标识码**：A　　**文章编号**：1001-3695(2006)05-0151-03

Technology of Progressive Meshes for Large Scale Terrain Based on Restricted Quadtree

YIN Hong[1,2], XU Ji-heng[1], ZHOU Liang-wei[1], LIU Shi-bing[1], HUANG Da-jie[1]

(1. Engineering Institute of Engineering Crops, the PLA University of Science & Technology, Nanjing Jiangsu 210007, China; 2. Dept. of Computer Science, Nanjing University of Science & Technology, Nanjing Jiangsu 210094, China)

Abstract: Discusse an algorithm of real-time dynamic triangulation based on restricted quadtree for large scale terrain visualization. The real-time dynamic triangulation controlled by giving view dependent error come true. The experiments show that this algorithm is very good of the real-time LOD controlling for large scale terrain visualization and also for the real-time rendering.

Key words: Restricted Quadtree; Node; View-dependent

随着计算机图形图像软硬件技术的发展,人们认识周围环境从传统的二维思维方式转向立体空间的思维方式,开始构建三维的、实时交互的、可“进入”的虚拟地理环境,相继提出3DGIS, VRGIS以及相关三维 GIS 的概念。在这些三维的虚拟环境中,地形数据有着非常广泛的应用,同时也是虚拟环境的基础。

为了逼真地反映地形地貌,要对地形表面进行大量的采样,采样模型主要分为规则格网模型(Regular Square Grid, RSG)和不规则三角网模型(Triangulated Irregular Networks, TIN),通过对这些采样数据的三角化就可以获得地形的表面模型。由于地形 RSG 模型中(如 DEM)顶点分布的规则性,使得这种模型比 TIN 模型具有更大的应用价值。但 DEM 采样数据量庞大和高度的数据冗余给地形的实时绘制与处理带来了巨大困难。

要提高模型的绘制效率,就要在一定的误差控制下对模型进行简化,多层次细节技术(Level Of Detail, LOD)是当前解决这一问题的最好办法,可以有效地应用到地形数据的简化中。根据视点位置和视线方向的变化,通过删除重要性较小的顶点和三角形或者加入重要性较大的顶点和三角形来与原形逼近,从而实现网格的动态简化。

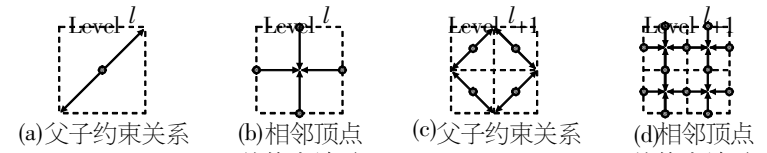
1　限制四叉树

四叉树(Quadtree)结构用于描绘地形碎片,非常简单但很有效。一个四叉树递归地把一个地形分割成一个个小块(Tessellates)并建立一个近似的高度图,通过将地形模型中的顶点数指定为$(2^n+1)\times(2^n+1)$并划分成不同的层次而形成四叉树,使树中的每一节点对应着由四块格网单元组成的面片,形成不同细节的层次模型。为了避免相邻两个不同的分辨率节点在接边处的裂缝现象,采用限制四叉树方法[1]进行动态构网。图 1 表示顶点之间的相互约束关系。

(1) 对于四叉树层次结构中第 l 层中的中心类顶点,与中心顶点所在四叉树块的对角线顶点具有约束关系。对角线的方向是中心顶点与上一层中心顶点连线的垂直方向。

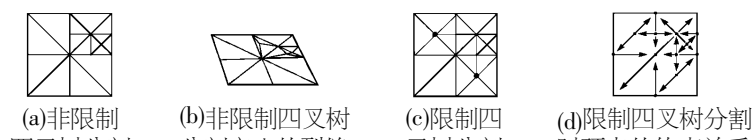
(2) 对于四叉树层次结构中第 l 层中的边界类顶点,与该层中的垂直或水平方向上的相临中心类顶点具有约束关系。

图 2 表示四叉树分割不同的分辨率节点在接边处产生的裂缝以及利用限制四叉树消除裂缝的方法,即如果选定某一细节层次中的某一顶点,那么与其有约束关系的顶点也应该被选定,这些与其有约束关系的顶点可能在较粗糙的上一层甚至更多。



(a)父子约束关系　(b)相邻顶点的约束关系　(c)父子约束关系　(d)相邻顶点的约束关系

图 1　四叉树中顶点间的约束关系



(a)非限制四叉树分割　(b)非限制四叉树分割产生的裂缝　(c)限制四叉树分割　(d)限制四叉树分割时顶点的约束关系

图 2　限制四叉树分割

收稿日期：2005-01-23；**修返日期**：2005-06-21

基金项目：国家自然科学基金资助项目(70371039)

2 视点相关误差判断标准

对象在投影到屏幕后,有的所占面积很小,产生大量微小三角形,对这些三角形进行简化,可以显著提高系统的效率。有些对象虽然比较大,由于视点和视线方向的关系,投影到屏幕后,所占的面积却很小,因此细化程度要求并不高,可以进行简化。给定屏幕容许误差 τ可以删除许多投影面积足够小的三角形。

设视点为$\vec{E}$,左方向为$\vec{L}=(1,0,0)$,向上方向为$\vec{U}=(0,1,0)$,视线方向为$\vec{D}=(0,0,1)$。考虑四叉树中节点 V_1(图 3),$\vec{P}$为边$\langle\vec{V}_0,\vec{V}_2\rangle$的中点,$\delta=\|\vec{V}_1-\vec{P}\|$,$\vec{V}$为$\langle\vec{V}_1,\vec{P}\rangle$的中点[3]。对于四叉树的中点 V_4,可以取[4]:

$$\delta=\mathrm{Max}\left\|V_4-\left(\frac{V_8-V_0}{2}\right)\right\|,\ \left\|V_4-\left(\frac{V_2-V_6}{2}\right)\right\|$$

考虑到 $\delta<<\|\vec{V}-\vec{E}\|$,所以视线方向单位矢量$\frac{\vec{V}-\vec{E}}{|V-E|}$,可以得到 δ在屏幕坐标系的投影长度 δ_s 的计算公式

$$\delta_s=\frac{\lambda^2 n^2\delta^2[(V_x-E_x)^2+(V_y-E_y)^2]}{[(V_x-E_x)^2+(V_y-E_y)^2+(V_z-E_z)^2]^2}$$

这里,视域四棱锥体的近平面距视点的距离为 n,视区为 $2b\times 2h$;屏幕分辨率为(S_x,S_y),$\lambda=\frac{S_x}{b}=\frac{S_y}{h}$。

引入容许误差 τ当 $\delta_s\le\tau$时顶点可以被简化,即

$$\delta_s=\frac{\lambda^2 n^2\delta^2[(V_x-E_x)^2+(V_y-E_y)^2]}{[(V_x-E_x)^2+(V_y-E_y)^2+(V_z-E_z)^2]^2}\le\tau^2$$

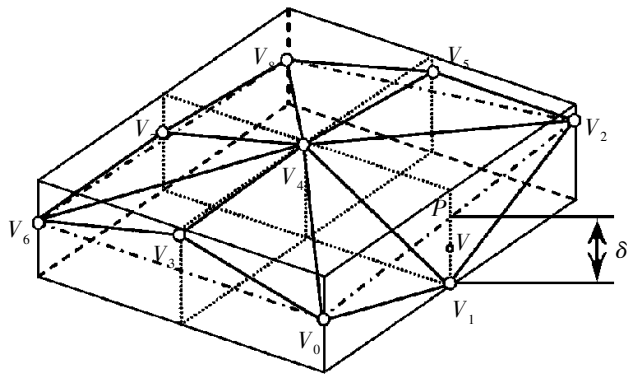


图 3 四叉树

3 算法实现

采用自顶而下的构图(Top-down)方法,考虑到 Intel 的 CPU 的内存页大小是 4KB,块的大小采用了 32 ×32,然后采用四叉树进行分割。这样的存储结构在一定层度上能提高存储效率,降低内存缺页的次数,有效地降低内存缺页带来的性能影响[6]。

3.1 数据结构

我们把全分辨率的地形数据存储在一个二维数组中,同时将预先计算的各顶点的误差 δ存放在四叉树中,这是静态的。四叉树中同时存放操作数据,用来存放地形四叉树顶点状态和网格递进情况,是动态的,四叉树节点 C# 数据结构如下:

```
//四叉树数据结构
Struct NodeData
    int Level              //当前节点层次,0 表示根节点
    string Index           //当前节点编号(标志符)
    ErrorData error        //四叉树误差数据结构
    OperateData opdata     //四叉树操作数据结构
//四叉树误差数据结构
Struct ErrorData           //定义地形四叉树节点
    double Error[5]        //δ值。0: Center; 1: West;
                           //2: South; 3: East; 4: North
//四叉树操作数据结构
Struct OperateData         //定义操作四叉树节点
    bool vertexactive[5]   //顶点数据。0: Center; 1: West;
                           //2: South; 3: East; 4: North
    bool IsNodeActive      //节点激活状态
```

利用树节点数据结构中的 Index,可以迅速判定当前节点的父节点、子节点以及相邻的节点(邻域)[5]。在四叉树中,我们采用四进制编码(图 4)较好地符合了四叉树的分割形式。在当前节点的 Index 中,删除最后一位即得父节点的 Index,在末尾增加一位(0,1,2 或 3),可得子节点的 Index。图 4(a)所示的四叉树 Index 编码方法划分为以下四个集合:南 $S=\{0,1\}$;北 $N=\{2,3\}$;西 $W=\{0,2\}$;东 $E=\{1,3\}$。边邻域称为 S 邻域、N 邻域、W 邻域、E 邻域;角邻域称为 NW 邻域、NE 邻域、SW 邻域、SE 邻域,如 3 为 0 的 NE 邻域等。

NodeData 用于存放四叉树节点各顶点的误差数据,顶点的(x,y)坐标和 DEM 高程数据根据节点的 Index 从地形数据的二维数组中获得。在顶点的数据结构中,只存放五个顶点的数据,分别为 Center, West, South, East 和 North[6],如图 5 所示。0、1、2、3 表示四叉树的节点。这样,每个节点的九个顶点数据,除本身储存五个外,其他四个角顶点数据可以从其父节点获得(根节点除外)。

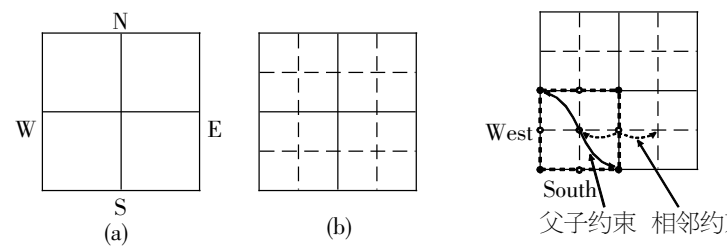


图 4 四叉树 Index 编码方式 图 5 四叉树节点各顶点的数

对于四叉树的每一层,网络结构只能为图 6 中所列六种基本网格递进的基本方式,其他方式均为基本方式逆时针旋转 n 倍 90°。

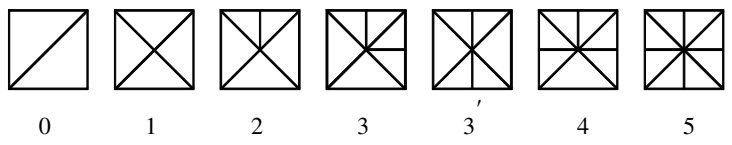


图 6 四叉树中网格基本方式

另外,还要建立一个视点相关链表,链表的元素为地形四叉树各顶点的视点相关系数,在视点变化过程中使用,并与操作四叉树结合,从而确定哪些顶点要激活或者删除。

3.2 四叉树的建立

在系统初始化阶段,利用 QuadTree. CreateNode 先建立四叉树,并利用 calculateDelta 计算四叉树各节点的五个顶点静态误差,同时根据初始视点,由 SetOperateData 计算四叉树各节点的操作数据,建立四叉树的伪代码如下:

```
Void CreateQuadtree(level){  //建立四叉树
    if(level<=maxlevel)
    for each subnode∈(0…3){  //四个子节点
        //建立四叉树节点
        Node = QuadTree.CreateNode(Error);
        for each vertex //计算五个顶点的误差
            Node.ErrorData.Error = calculateDelta;
        //根据最初设定视点初始化操作四叉树节点数据
```

```
        SetOperateData ( Node, tolerance, viewpoint) ;
        next level;
        CreateQuadtree( level) ; //递归
}       }
```

3.3　动态操作数据

在系统运行过程中,随着视点的不断变化,操作数据也在动态发生变化。此时只需要更新视野范围内四叉树节点的动态操作数据,由 SetOperateData 来实现。SetOperateData 中,由 setVertextState 根据容许误差以及限制关系确定节点中五个顶点的状态,并依据父子限制关系更新父节点各顶点的状态,最后根据顶点的状态 setVertextState 确定网格的递进方式,动态数据更新的伪代码如下:

```
Void DynamicOperate( Node, tolerance, viewpoint) { //操作四叉树
    if ( NodeInFrustum)
    for each subnode∈(0…3) { //四个子节点
        SubNode = TreeNode. SubNode;
        if ( SubNode! = null) {
            //计算操作四叉树节点数据
            SetOperateData( SubNode, tolerance, viewpoint) ;
            DynamicOperate( SubNode, tolerance, viewpoint) ;
    }       }
    //不在视野范围内
    else Node. OperateData. IsNodeActive = false;
}
SetOperateData ( Node, tolerance, viewpoint) { //节点操作数据
    for each vertex∈(0…4){ //五个顶点
        //比较基于视点的顶点误差和容许误差
        getVertextState( tolerance, viewpoint, Error) ;
        //根据限制关系和顶点误差设置顶点状态
        setVertextState( Node) ;
    }
}
```

3.4　限制关系

限制四叉树的限制关系分为父子限制关系和相邻限制关系两类。父子限制关系是由于子节点的中心顶点处于激活状态而导致父节点的相应顶点(Center, West, East, South 或 North 中的两个) 因受限制而被激活, ParentRestricted 实现了该功能,父子限制关系伪代码如下:

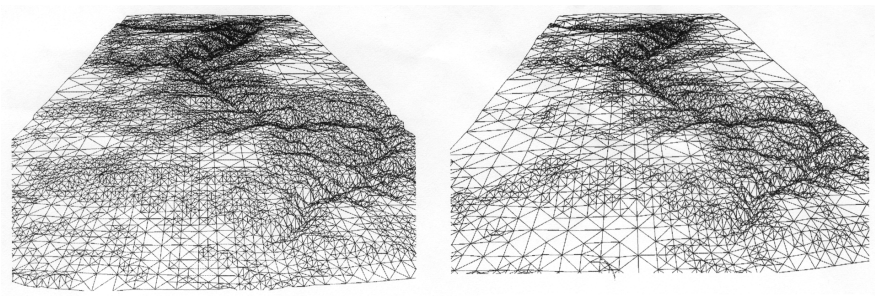
```
//父子限制关系
Void ParentRestricted( Node) {
    if ( Node! = null) { //存在父节点或父节点相应限制顶点未激活
        getVertextState; //获得相应限制顶点的状态
        if (! ActiveState) { //父节点相应限制顶点未激活
            setVertextState; //设置相应限制顶点的状态
            NeighborRestricted( Node) //相邻限制关系
            Node = Node. parent
        }
        else Node = null //父节点相应限制顶点已激活
        ParentRestricted( Node. parent) //递归
}       }
```

在相邻限制关系的处理中(NeighborRestricted) , GetNeighborNode 把相邻限制关系分为同一四叉树间的兄弟限制关系和相邻四叉树间的邻居限制关系。在此类限制关系中,只需处理中心类顶点,相邻限制关系伪代码如下:

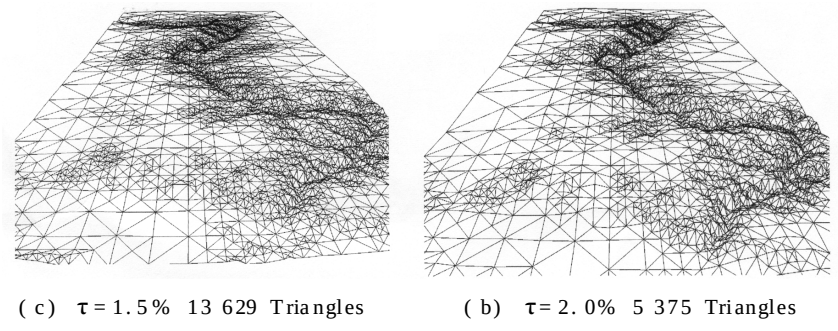
```
//相邻限制关系
Void NeighborRestricted( Node) {
    NeighborNode = GetNeighborNode( Node) //获得相邻节点
    //中心顶点未激活
    if(! NeighborNode. NodeData. OperateData. vertexactive[0]) {
        //激活中心顶点
        NeighborNode. NodeData. OperateData. vertexactive[0] = true;
        ParentRestricted( NeighborNode) ; //父子限制关系
}       }
```

4　实验结果

基于以上算法,使用 Hoppe 的大峡谷 DEM 数据进行实验,其大小为 4097 ×2049, 采样间距为 30m, 地形实际大小为 120km ×60km, 高度数值由 255 级灰度值表示, 一个高度单位对应 10m 计算。在普通 PC 机平台, 128MB 显卡上使用 C#基于 OpenGL 实现了一个地形漫游的原型系统。绘制窗口大小为 1024 ×768(图 7) 。实验表明, 该算法免去了许多数据结构和操作, 开销较小, 对于实时控制大规模地形模型的细节层次, 增强大规模地形模型的绘制效率是行之有效的, 可以作为战场可视化的基础。由于大规模地形数据很多, 下一步注重开发地形压缩数据的地形漫游系统。



(a) τ=0.5%　52 733 Triangles　　(b) τ=1.0%　23 108 Triangles



(c) τ=1.5%　13 629 Triangles　　(b) τ=2.0%　5 375 Triangles

图 7　Hoppe 的 4097 ×2049 大峡谷不同分辨率网格表示

参考文献:

[1] Pajarola. Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation[C]. Proceedings of IEEE Visualization '98, 1998. 19-26.

[2] P Lindstrom, V Pascucci. Visualization of Large Terrains Made Easy [C]. Proceedings of IEEE Visualization, 2001. 363-370.

[3] P Lindstrom, D Koller, W Ribarsky, *et al*. Real-time, Continuous Level of Detail Rendering of Height Fields[C]. Proceedings of SIGGRAPH '96, ACM SIGGRAPH, 1996. 109-118.

[4] Thomas Gerstner. Top-down View-dependent Terrain Triangulation Using the Octagon Metric[C]. Eurographics Symposium on Geometry Processing, 2003. 1-11.

[5] W Evans, D Kirkpatrick, G Townsend. Right-triangulated Irregular Networks[J]. Algorithmica, 2001, 30(2): 264-286.

[6] Xiaohong Bao, Renato Pajarola. LOD-based Clustering Techniques for Efficient Large-scale Terrain Storage and Visualization[C]. Proceedings SPIE Conference on Visualization and Data Analysis, 2003. 225-235.

作者简介:

殷宏(1967-),男,安徽黄山人,副教授,博士研究生,主要研究方向为军事仿真与可视化战场;许继恒(1976-),男,陕西咸阳人,硕士,主要研究方向为虚拟现实与虚拟战场环境。