

基于 OpenGL 技术的多幅电子地图数据快速组织与表达

周小军, 王光霞, 夏青, 王富强

(信息工程大学地理空间信息学院, 郑州 450052)

摘要: 电子地图符号化的质量和绘制速度一直是地图工作者关注的重点。目前, 地图符号化的质量基本能满足各类用户的需求, 但是, 电子地图绘制速度较慢的问题, 特别是在涉及较大数据量的多幅电子地图显示时, 系统响应用户操作时间比较长, 影响用户操作体验。本研究分析比较了 GDI 与 OpenGL 的特点, 提出使用 OpenGL 相关技术实现地图要素的符号化方法, 并用内存池技术和共享显示列表的多线程调度技术实现多幅电子地图数据的快速组织与表达。实验表明, 利用文中的方法能够实现地图点、线、面要素的符号化, 并在地图数据的漫游、缩放操作的屏幕绘制速度基本保持在 17 帧/s 以上, 表明利用本文的方法和技术提高了系统的显示速度, 保证了用户与电子地图良好的交互效果。

关键词: 电子地图; 符号化; 三维 API; 内存池; 多线程

DOI: 10.3724/SP.J.1047.2013.00491

1 引言

电子地图具有无级缩放、无缝显示、动态载负量调整等优点^[1]。一直以来, 电子地图符号化的质量和效率是地图工作者追求的两个目标^[2]。目前, 电子地图符号化的质量基本能满足各类用户群体的需求, 但是, 电子地图绘制速度的问题一直影响地图用户的用图感受, 特别是在大范围, 涉及大量不同尺度、多幅地图数据时, 系统响应用户操作的迟滞时间比较长, 导致用户因等待而产生焦躁和不满, 难以给地图使用者带来较为流畅的操作体验, 从而降低了电子地图与用户适人化的交互操作效果。此外, 现已进入了 Web 地图的时代, 电子地图数据需要异地实时的传输和显示, 而且在应用中往往需要加载各类专题数据, 因此, 更需要考虑在网络环境下提高电子地图绘制速度的问题, 很多学者提出了各种解决方法, 归纳起来主要是对地图进行分块、分层的处理并建立高效的索引模型^[3-7], 通过减少显示区域内的地图的数据量和提高索引速度达到电子地图高显示效率的目的。

三维虚拟地理环境采用具有图形加速功能的三维图形接口及其技术对场景进行渲染, 渲染平均

帧数基本达到了用户实时操作的要求, 因而它在给用户带来更符合人类视觉特点的直观三维场景的同时也带来了更加快速流畅的操作。

本文利用三维接口 OpenGL 以实现地图数据的符号化方法和多幅地图数据的快速组织和调度策略。

2 基于 OpenGL 的电子地图绘制方法

三维图形接口 OpenGL 是一种流线型、独立于硬件的具有强大图形绘制功能的接口, 它能利用硬件加速方法绘制图元, 具有的硬件协助图形绘制性能远远胜过单纯的软件实现。与 GDI(Graphics Device Interface)一样, OpenGL 也可以处理二维的点、线、面等图元。GDI 和 OpenGL 都可实现对二维图元的绘制, 在图元数量较少时, 二者绘制的速度差距不大, 但是, 在图元数量比较大且以静态图元为主时, 利用显示列表技术、纹理映射技术、顶点数据技术的 OpenGL 渲染速度要远快于 GDI。此外, OpenGL 的渲染方法与系统无关, 可扩展强, 因此, 本文利用 OpenGL 实现的地图要素符号化模块可在其他操作系统上方便地进行移植。

收稿日期: 2013-02-18; 修回日期: 2013-03-29.

基金项目: 国家“863”计划资助项目(2013AA12A202); 国家自然科学基金项目(41271393)。

作者简介: 周小军(1986-), 男, 汉族, 江西上饶人, 博士生, 主要研究方向为 GIS 与数字高程模型。E-mail: zxbmzsy@sina.com

根据空间几何特征可将地物分为点状、线状和面状要素^[8]。利用 GDI 或 GDI+ 实现这 3 种要素符号化的方法及技术已经相当成熟, 本文利用 OpenGL 相关技术针对不同要素设计了不同的绘制方法。

2.1 点状要素符号化

点状要素符号一般可分为简单符号和复杂符号。简单符号在利用 OpenGL 绘图函数直接绘制点、圆、三角形等基本图元的基础上, 通过对基本图元的组合就可完成点状要素的符号化, 例如, 村庄、县级行政等。对于比较复杂的符号, 本文利用纹理映射的方法, 先将点状符号预先处理成符号纹理, 而后在点状要素的位置上构造一个多边形面元, 最后将符号纹理映射到点状要素所在位置的多边形面元上, 例如水塔、无线电站等。

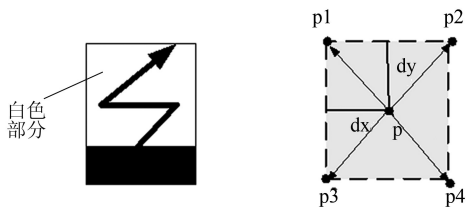


图1 点要素的符号化方法

Fig.1 The symbolization of point feature

以无线电站为例, 如图1所示, 其包含一个定位点 p , 依据这个点计算出以该点为中心的矩形4个角点 $p1, p2, p3, p4$ 。其中, 关键问题就是 dx 和 dy 这两个值的确定, 其值与显示比例尺密切相关, 可通过空间坐标转换的方法, 将符号在屏幕上的显示距离换算成空间坐标系中的数值。由于符号底色为白色, 如果简单的将该符号映射到矩形上, 必然会出现符号中白色部分遮挡其他地图要素的情况。解决这个问题方式就是利用 OpenGL 的透明纹理技术, 将白色部分进行混合透明处理。对于带有方向的点状要素, 只需根据两点计算出的偏转角在 OpenGL 进行旋转即可。此外, 还可以采用 OpenGL 中点块纹理 (Point Sprite) 技术提高点状纹理映射的效率。该技术支持通过绘制一个三维点, 把一个二维纹理对象显示在屏幕上, 而且具有自动调整视点与纹理的位置功能, 不需要应用矩阵逻辑来维护, 其所需要处理的数据也只为4个顶点的多边形的 $1/4$, 因而, 可以大大提高点状纹理映射的效率。

2.2 线状要素符号化

线状要素的符号化配置方法本质上是一种伦移变换的方法^[9]。

其定义为: 设 $f_0, f_1: X \rightarrow Y$ 是拓扑空间 X 与 Y 之间的映射, 如果存在一个从 X 上的柱形 $X \times I$ (I 是实数轴上的单位闭区间 $0 \leq t \leq 1$) 到 Y 的映射 $F: X \times I \rightarrow Y$, 使得 $F(x, 0) = f_0(x)$, $F(x, 1) = f_1(x)$, 对于任意 $x \in X$, 则 f_0 与 f_1 同伦, 这样的 F 就叫作 f_0 到 f_1 的一个同伦或伦移。

因此, 线状要素的符号化配置可以通过伦移变换的原理构建线状符号单元与线状要素实体间配置的函数关系, 使得线状符号单元在线状要素定位线上正确地配置。本文对线状要素的符号化采用两种方式, 一种是针对简单的线状符号, 如小路、大车路、乡路等, 直接利用 OpenGL 函数进行绘制, 通过对线画的宽度、颜色、样式等的设置表示不同线状要素。

对于堤岸、长城、铁路等复杂的线状要素符号, 利用纹理循环配置的方法进行绘制, 如图2所示, 该方法以线状要素的定位线为基准, 将线状要素扩展成一定宽度的闭多边形, 然后在该多边形上循环映射不同的线状符号图元组合来表示不同的线状要素。本文采用平行线推移算法实现以线状要素坐标点连线为定位线的多边形生成, 其原理如图3所示。

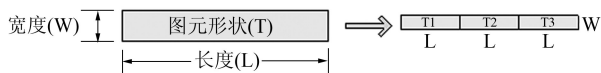


图2 复杂线状要素抽象表达模型

Fig.2 The abstract representation model of complex line feature

如图3(a)所示, 假设线段 ABC 的3个点坐标分别为 (x_A, y_A) , (x_B, y_B) , (x_C, y_C) , 对其向上进行平移垂直距离 t , 得到其平行推移后的线段为 GB_1E , 以 B 点为中心建立直角坐标系, 设线段 CB 与 x 正轴的夹角为 α , 线段 AB 与 x 负轴的夹角为 β , 线段 B_1B 与线段 BC 的夹角的为 θ , 则

$$\theta = (180 - \alpha - \beta) / 2, d = t / \sin \theta,$$

$$\alpha = \arctan(y_C - y_B / x_C - x_B),$$

$$\beta = -\arctan(y_B - y_A / x_B - x_A),$$

根据其几何关系推理可推算出

$$x_{B1} = x_B + d \cos(\alpha + \theta);$$

$$y_{B1} = y_B + d \sin(\alpha + \theta);$$

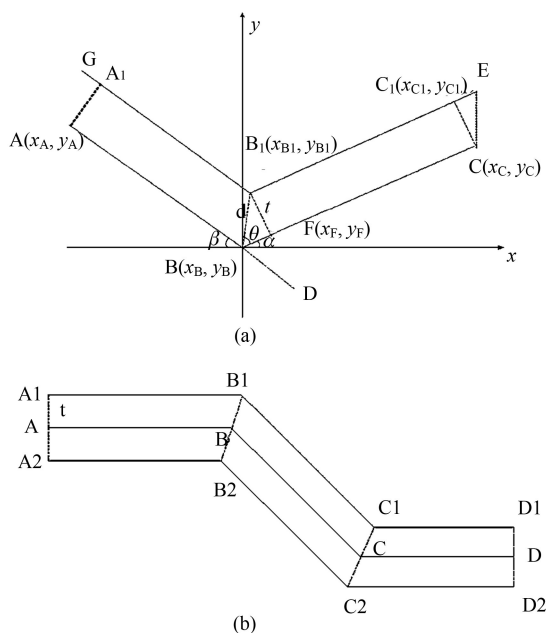


图3 平行线推移算法示意图

Fig.3 The parallel line shift algorithm

图3(a)中,如果C点为线段的最后一个点,则其平行推移后C1点位置坐标的计算公式为:

$$x_{C1} = x_C + t \left(\frac{\cos(\alpha + \theta)}{\sin \theta} - \frac{\cos \alpha}{\tan \theta} \right)$$

$$y_{C1} = y_C + t \left(\frac{\sin(\alpha + \theta)}{\sin \theta} - \frac{\sin \alpha}{\tan \theta} \right)$$

在以上的论述中,参数 t 值可以为正,也可以为负。 t 值的正负可决定平行推移出的平行线是在定位线的左侧还是右侧,推移距离 t 需要根据地图比例尺和屏幕分辨率的不同来设定。

如图3(b)所示,线段ABCD为线状要素的定位线,根据平行线推移算法,分别向左右两侧各推移垂直距离 t ,得到了其左右两侧的平行线,再将线段A1B1C1D1和线段A2B2C2D2两端进行闭合处理,就得到了一个多边形,最后,在多边形上映射上相应的符号纹理。考虑到线状符号在拐角处会产生严重变形以及出现断裂、基本图元自相交等问题^[10],采用符号自适应性及双仿射变换方法解决^[11]。对于像铁路类要素,只需要将单位纹理符号按照配置规则在平行四边形里循环贴图即可;对于像电力线、拦水坝类的线状要素,还需要使用透明纹理技术,将单位符号纹理中的符号以外的像元进行透明纹理处理。

2.3 面状要素符号化

面状符号一般由边界线和填充图形构成,其边

界轮廓线可利用线状要素符号化方法进行处理,填充图形可以用晕线、点状符号、位图等来描述^[12]。在面域内填充底色及填充符号时,如果该面域的多边形是复杂多边形,则需要对该多边形进行剖分处理。OpenGL只能直接操作凸多边形,如果需要处理凹多边形、中间带孔的多边形或具有相交边的多边形,就需要将不规则的多边形剖分成简单三角形。如图4所示,是对于复杂多边形的一种分解剖分方案,剖分完成的多边形可直接进行纹理映射,完成面状要素的符号化。

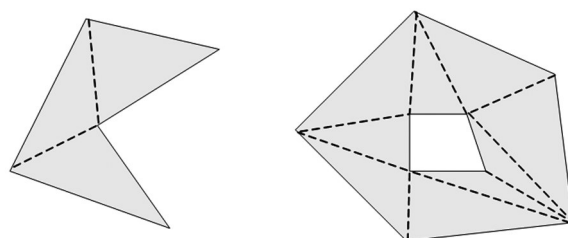


图4 复杂多边形的剖分

Fig.4 The split of complex polygon

对于复杂多边形的剖分,本文使用Delaunay法则带约束的递归生长算法,剖分后相邻的Delaunay三角形互不重叠,各三角形的外接圆也不包含其他三角形中的点,其剖分流程如图5所示。

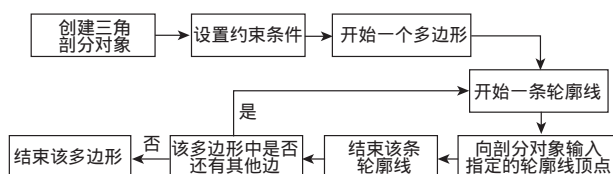


图5 复杂多边形三角剖分流程图

Fig.5 The triangulation flowchart of complex polygon

3 多幅电子地图数据快速组织与表达

3.1 内存池技术的数据文件缓存机制

数据缓存与调度是提高系统效率的关键技术之一,许多学者对此进行了研究^[13-14]。由于多幅图涉及的数据量大,因此,需要对其设计合理高效的调度机制。本文提出采用内存池缓存的技术,并根据视点位置和视域的大小制定相应的数据调度规则,实现多比例尺矢量空间数据的快速调度。

视点的快速变化,需要系统能够及时快速地据当前视点的位置和显示范围将相关比例尺的地图数据进行预处理并更新到内存中,最后进行可视化

表达。以漫游操作为例,按照一般的内存管理方式,视点位置的变化要求对窗口中显示的地图数据进行及时更新,需要计算机不断分配和释放内存。如果有新的数据需要加载,则向系统发出内存申请,并将不在视域范围内的数据释放。因此,视点的不断变化必然导致计算机内存碎片的大量增加,影响系统的运行效率。为改善这种情况,本文采用建立内存池的方法减少内存不必要的重新申请和释放。内存池需要根据地图数据量的大小,进行预先分配适当的内存缓冲区,并对缓冲区构建唯一的索引表。已有研究表明^[15-18],不同的索引方法对于同一类型的地理目标索引效率差别甚大,常见的建立索引方式有网格索引、四叉树索引和R树索引。本文为便于管理,将地图图幅编号作为页面索引值,采用四叉树索引方式,建立其与地图数据的映射关系。当有地图数据申请载入时,先通过索引表判定该块数据是否已存在于内存池中,然后决定是否在内存池中开辟空间。一旦内存池中地图数据超过其预定大小时,需通过规则和算法将相应的页面数据置换出去。文献[19-20]对空间数据库的页面替换策略进行了分析,页面置换方法主要有最优页面置换算法、二次机会算法、老化算法和FIFO策略^[21],考虑到矢量数据组织的方便,本文采用视点相关的老化算法。具体来讲,就是构建页面内地图数据块中心点与当前视点位置的偏移量,以此偏移量来设定页面的重要性参数,并设置一个计时器对其进行记录,当内存池中的页面数据需要被置换时,则将内存池中计时最长的页面置换出去。内存分配和释放的主要流程如图6所示。

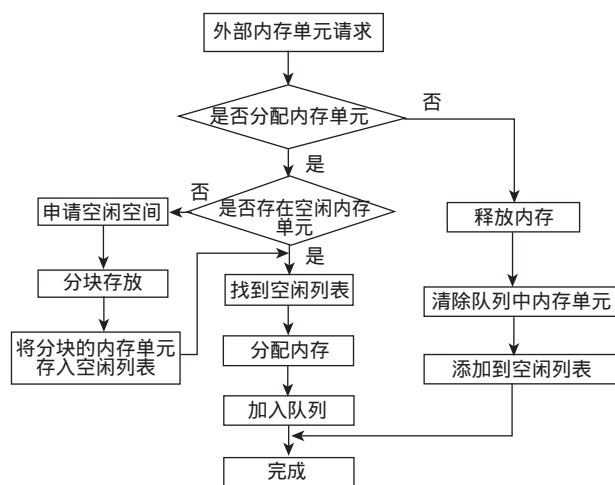


图6 内存分配和释放的流程

Fig.6 Flowchart of memory allocation and release

3.2 视点相关的地图数据实时调度

在视点相关的地图数据实时调度中,提出了利用共享显示列表的多线程数据调度方法,首先创建两个线程,主线程和后台线程,主线程负责响应用户的操作、与视点相关图幅的计算,后台线程负责对内存池的操作和数据的各种处理工作,二者通过共享显示列表的方式实现通信。

设定2个存储数据的容器,分别为V1和V2。如图7所示,根据当前视点的位置信息及当前比例尺地图数据的经差和纬差可计算当前地图周围的8幅地图图幅号,将全部地图图幅号存入到数据容器V1中,这时需对V1中进行一次预读取判断,以此来确定V1中存储的地图数据哪些图幅是存在于地图数据库中的,然后将存在的图幅号存入到容器V2中。

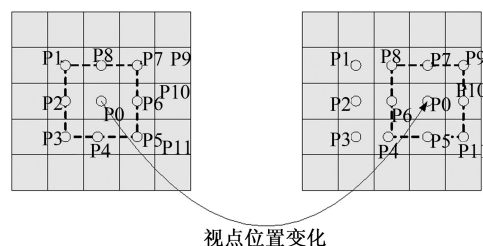


图7 与视点相关的图幅调度

Fig.7 The mapsheet scheduling depending on the point of sight

V2中一旦有了新数据,后台线程启动,如果以下2个条件满足,则立即对V2中数据进行处理。

条件1:将V2中存放的图幅号与内存池中的地图数据图幅号进行比对,V2中存在新地图数据。

条件2:后台线程中的上一任务已经完成,即后台线程是空闲的。其目的是为了控制主线程与后台线程的同步。

如果后台监控线程还在进行上一个任务,这时包含新图幅的V2被后台线程监控到,这时不会对V2中的数据进行处理。

当图中的视点由P0点漫游到P6点时,这时V2中的图幅将是P0,P4至P11各点所在的图幅。在后台线程对V2进行处理时,先将其与内存池中的数据的图幅号进行匹配,将内存池中不存在的数据进行处理,加载到内存池,数据调度流程图绘制如图8所示。

4 实验系统验证及分析

实验验证主要利用OpenGL实现电子地图符号

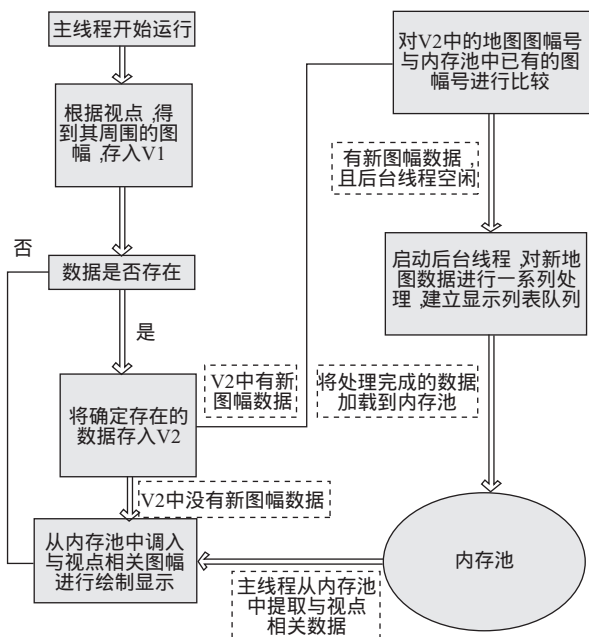


图8 显示列表的多线程数据调度流程图

Fig.8 The scheduling flowchart based on multithreading of display list

化的效果,提高在多幅地图数据中进行漫游、缩放等操作时的绘制效率。

4.1 地图符号化的效果实验

该实验利用文中提及的方法对地图要素进行符号化表达,效果如图9所示。实验结果表明,该方法对点状、线状和面状要素能进行合理的绘制,符号与色彩的设计基本上能满足地图符号化的要求。

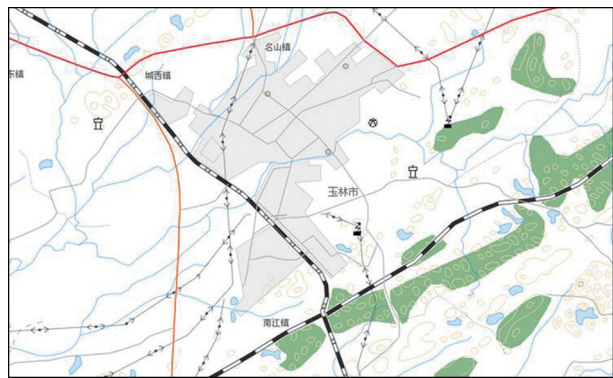


图9 基于OpenGL的地图要素符号化效果

Fig.9 The symbolizing effect of map elements based on OpenGL

4.2 地图绘制效率测试

实验目的是测试在多幅电子地图数据中进行

漫游、缩放操作时,系统的绘制速度。

实验数据为25幅1:25万地图数据。

实验方法为,在地图要素都显示的条件下,利用键盘和鼠标对地图数据进行漫游和缩放的操作。测试计算机CPU使用情况和屏幕绘制帧率。

系统共使用了6种不同硬件配置的环境,分别是:

(1) Intel Core2 双核 CPU 2.4GHz,1GB 内存, NVIDIA 140M 显卡(笔记本)。

(2) Intel Core2 双核 CPU 1.8GHz,1GB 内存, ATI X1400 显卡(笔记本)。

(3) Intel Core2 双核 CPU 2.0GHz,1GB 内存, NVIDIA 7700 显卡(笔记本)。

(4) Intel Core2 双核 CPU 2.5GHz,2GB 内存, NVIDIA GT240M(笔记本)。

(5) Intel Core2 单核 CPU 2.4GHz,512MB 内存, NVIDIA 7300GT(台式机)。

(6) Intel Core2 双核 CPU 1.8GHz,1GB 内存, ATI HD 2600PRO(台式机)。

表1 不同硬件环境下测试结果

Tab.1 The test results in different hardware environments

硬件环境序号	后台线程未处理数据时CPU使用率(%)	后台线程正处理数据时CPU使用率(%)	使用文中方法后绘制帧率(帧/s)	使用文中方法前绘制帧率(帧/s)
1	15~25	61~75	22~30	11~15
2	25~30	46~63	22~25	9~13
3	18~24	63~78	24~32	10~16
4	12~20	40~60	28~36	13~18
5	20~35	60~90	18~23	8~12
6	15~22	50~70	26~33	12~15

实验结果如表1所示,从表中数据可知,后台线程启动并处理数据时,计算机CPU使用率明显增高,表明OpenGL多线程利用了多核处理器的优势,提高了CPU的使用率,实现了各线程任务的分工。使用文中的方法,系统绘制帧率保持在17帧/s以上,表明利用文中的方法和技术提高了系统的渲染速度,从而保证了用户与地图良好的交互效果,提高了用户操作地图时的舒适感。

5 结束语

本文从人机交互绘制地图效率的角度出发,使

用三维接口 OpenGL 实现地图要素的符号化,以及用内存池技术和共享显示列表的多线程调度技术,实现多幅电子地图数据的快速组织与表达。通过实验表明,本文方法可实现地图的符号化,并提高多幅电子地图数据的绘制速度,从而给地图用户带来更舒适的电子地图使用和操作体验。此外,由于本文采用的是在三维接口中实现矢量数据的符号表达,在三维虚拟地形环境中还可以将符号化的矢量数据通过多重纹理的方法实时映射在三维地形上,实现矢量数据与 DEM、影像数据的综合表达。

参考文献:

- [1] 蔡孟裔,毛赞猷,田德森,等.新编地图学教程[M].北京:高等教育出版社,2000,286-287.
- [2] Mustafa N, Krishnan S, Varadhan G, *et al.* Dynamic simplification and visualization of large maps[J]. *International Journal of Geographical Information Science*, 2006,20(3): 273-302.
- [3] 王俊,张文诗,王建涛.多图幅海量数据电子地图快速显示的研究与实现[J].*测绘工程*,2003,12(3):19-20.
- [4] 杨春成,谢鹏,何列松,等.地图数据读取过程中的数据调度[J].*武汉大学学报·信息科学版*,2009,34(2):166-168.
- [5] 肖计划,孙群,刘海砚.多源多尺度地图数据的组织与管理[J].*测绘科学技术学报*,2009,26(1):24-28.
- [6] 江南.基础电子地图显示的关键技术[J].*测绘科学技术学报*,2008,25(4):241-244.
- [7] 郑束蕾,陈毓芬.多媒体电子地图集响应用户操作速度的研究[J].*测绘科学*,2005,30(6):61-63.
- [8] 张保钢,罗晓燕.超大城市地形图数据建库分库设计[J].*测绘通报*,2007(8):8-9.
- [9] 游涟,胡鹏.地图代数的符号化方法[J].*测绘学报*,1994,22(5):136-137.
- [10] 郭庆胜,郑春燕.地图线状符号图案单元的优化配置方法[J].*武汉大学学报·信息科学版*,2002,27(5):499-504.
- [11] 吴小芳,杜清运,徐智勇,等.复杂线状符号的设计及优化算法研究[J].*武汉大学学报·信息科学版*,2006,31(7): 632-635.
- [12] 吴立新,刘纯波.地图符号库的面向对象技术与引用借口设计[J].*矿山测量*,1999(1):32-35.
- [13] Chou H T, DeWitt D J. An evaluation of buffer management strategies for relational database systems[C]. *The 11th VLDB Conference*, Sweden, 1985,127-141.
- [14] Chen CM, Roussopoulos N. Adaptive database buffer allocation using query feedback[C]. *The 19th VLDB Conference*, Ireland, 1993,342-353.
- [15] 阎超德,赵学胜.GIS空间索引方法述评[J].*地理与地理信息科学*,2004,20(4):23-26.
- [16] 郭建忠.系列比例尺条件下海量数据的快速显示[J].*测绘学院学报*,2005,22(2):136-138.
- [17] 张江水.基于 ReWorks 的嵌入式地理信息系统(EGIS)的设计与实现[D].郑州:信息工程大学测绘学院,2005.
- [18] 王轩,李鹤元.基于分区索引方法的地图快速显示的设计与实现[J].*测绘通报*,2002(10):56-57.
- [19] Thomas B. A robust and self-tuning page-replacement strategy for spatial database systems[C]. *The 8th International Conference on Extending Database Technology*, Prague, 2002.
- [20] 涂小明,汪林林.分布式空间数据库中基于事务的客户端高速缓存技术研究[J].*计算机科学*,2004,31(6):76-78.
- [21] 徐甲同.操作系统教程[M].西安:西安电子科技大学出版社,1994.

Fast Organization and Expression of Multi-sheet Electronic Map Based on OpenGL Technology

ZHOU Xiaojun*, WANG Guangxia, XIA Qing and WANG Fuqiang

(*Institute of Geography Spatial Information, Information Engineering University, Zhengzhou 450052, China*)

Abstract: The quality of map symbolization and the display speed of the electronic map are always the hot issues for the cartographers. The availability of the electronic map mainly includes the effectiveness, efficiency and user subjective satisfaction. Currently, the map symbolization quality can meet to the basic needs of various user groups, but electronic map rendering speed has affected the feelings of map users. Since the electronic map involving large volumes of data, it needs a relatively long period of time to respond to user's operation, causing the user to generate anxiety and dissatisfaction because of waiting, so it is difficult to give map users more

smooth operating experience. For the problem of improving the rendering speed, current studies mainly focused on data block processing, hierarchical processing and the establishment of efficient indexing model to improve the efficiency of electronic map display by reducing the amount of map data. Based on analysis and summary on the improved multi-sheet electronic map rendering speed, this research compares the characteristics between GDI and OpenGL. When we use GDI and OpenGL to draw 2D entities at the same time, if the number of primitive is small, the difference is not large. But if the number of primitive is large and primitives are static, the rendering speed using OpenGL is far faster than the GDI, because OpenGL can use display list technology, texture mapping and vertex data technology. In this paper we presented and implemented a symbolic method in three-dimensional interfaces for the different types of features. It proposes the way of expression and organization of multi-scale data in the three-dimensional space through multithreading technology and memory pool. The experimental results show that it is able to achieve the symbolization of point feature, line feature and area feature. And roaming, zooming in multi-sheet electronic map data, screen rendering speed remain at more than 17 frames per second. So, using methods and techniques mentioned in this paper can improve rendering speed and ensure the good interaction effect between user and map.

Key words: electronic map; symbolization; 3D API; memory pool; multithreading

***Corresponding author:** ZHOU Xiaojun, E-mail: zxjbmzsy@sina.com