第 33 卷 第 6 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.6 Computer Engineering March 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)06—0275—03 文献标识码：A 中图分类号：TP311

一种体绘制专用体系结构存储器的设计

乌晓峰，孙济洲，魏继增
(天津大学电子信息工程学院，天津 300072)

摘 要：提出了一种适合 Ray Casting 算法的体绘制专用体系结构的存储模型。根据处理器数目的不同，体数据被划分为不同的子体。子体依据其空间坐标位置的不同被分配到不同处理单元，子体内的体素被分配到相应处理单元的存储器的对应位置。说明了子体在处理器间的分配方式以及体素在存储器内的编址和寻址方式。

关键词：体绘制；体系结构；存储模型；光线投射

Memory Design of Dedicated Architecture for Volume Rendering

WU Xiaofeng, SUN Jizhou ,WEI Jizeng
(School of Electronic and Information Engineering, Tianjin University, Tianjin 300072)

【Abstract】 A memory model suitable for dedicated architecture of volume rendering based on Ray Casting algorithm is proposed. Volume data is divided into subcubes based on the number of processors. A subcube is distributed to the corresponding processing unit according to the space coordinates. Each voxel of a subcube is stored in a given memory unit. The way of the assignment of subcube to different processors is introduced and the addressing scheme of the voxels is described.

【Key words】 Volume rendering; Architecture; Memory model; Ray casting

体绘制技术是三维数据场重建的关键技术之一，在医学三维重建、流体力学计算、地震地质等方面具有广泛的应用。Ray Casting 算法是直接体绘制算法中的一种，其特点是绘制图像质量优但同时计算复杂度高，绘制速度慢，不能满足实时绘制的要求，针对这个问题有 3 种可能解决方案：

(1)软件算法的改进。使用一些加速技术，比如冗余数据剔除、空间数据结构优化和提前不透明度计算截止等。

(2)使用商用大型机或者并行计算，比如机群计算技术。

(3)使用专用的体系结构来实现绘制。

3 种方案比较而言，软件加速能做到的空间有限，很难达到实时的绘制效果，特别是当体数据的数据量比较大的时候，满足不了实时绘制的需要；大型机和集群技术资金消耗较大，不利于推广使用；因为计算复杂度和实时绘制之间存在的矛盾需要进行硬件加速，而直接体绘制并不使用传统的面绘制方法，所以传统的图形加速设备并不能满足需求。因此设计专用的体系结构是一个相对合理的解决方案。

1 Ray Casting 算法及并行化对存储器设计的要求

光线投射算法(Ray Casting)是直接体绘制算法中像序算法的典型代表，从像平面的每一个像素点出发，沿视线方向发出一条射线进入体素空间。在这条射线[a,b]上等间距选取若干重采样点，计算每个重采样点在体空间中的位置，然后用所选定的重构函数通过插值计算该重采样点的体数据值，从而既进一步通过传递函数确定其不透明度及颜色值，最后对各个重采样点按照从后向前或者从前向后进行图像合成。

Ray Casting 算法有很多使用了加速技术的改进版本，本文所提出的存储模型并不专门针对某一种 Ray Casting 算法的特例，而是探索适合于采用 Ray Casting 算法以及 Ray Casting 算法的一些加速技术所实现的这样一系列的体系结构上。这种存储结构支持多处理器对体数据的并行处理，加快了处理速度，对于实现实时的体绘制有很大的帮助。同时该存储结构经过简化可以成为八路交叉访问的存储器的一种解决方案，同样适用于单处理器的情况。

光线投射算法具有空间上的连续性的特点。如果对体数据进行分割而在不同的处理器间分配，在光线从一个子体传入另一个子体的过程中要进行信息的传递，传递的这些信息包括：计算的中间结果，上一个重采样点的位置信息，以及光线投射的步长等。

直接体绘制由于要进行重采样点体数据值和梯度的三次线性插值运算,为了提高运算速度，希望将体数据分配给不同的处理器去进行并行处理。每个处理器处理一部分体数据。除了处理器个数等硬件因素外，如何更有效地提高运算速度同两个因素具有直接的关系：(1)体数据相对于处理器的分配方式；(2)体数据在存储器中的存储方式。

对于第 1 点，做法是将在几何空间上位置相近的那一部分体数据交给一个处理器进行处理。这种分配方式是由 Ray Casting 算法的特点决定的：在光线投射过程中，下一次计算要依靠于上一个重采样点的位置和截至到光线投射位置的采样点的中间结果。因此，(特别是对于平行投影)将位置邻近的体数据交给相同的处理器便于中间结果的重复利用而有效减少了在不同的处理器之间的数据交换。

体数据在存储器中的存储方式却不同，使用多个存储器

基金项目：国家自然科学基金资助项目“实时体绘制关键技术及支撑体系结构研究”(60373061)

作者简介：乌晓峰(1981－)，男，硕士生，主研方向：体绘制体系结构，并行计算；孙济洲，教授、博导；魏继增，硕士生

收稿日期：2006-03-28 **E-mail：**wufengxiao@hotmail.com

希望将体数据进行分步存储，以便做到在一个存储周期内能够将相应的数据取出，这不是将空间位置相近的体数据存放在相同的存储器内所能满足的。与这种密集存储相反，应该尽可能分散在空间上相邻的体数据以分配更多的存储器，便于达到用较少的存储周期取出计算所需全部数据。

2 存储体系结构

一个体绘制的专用并行体系结构由几个功能部件组成，这些功能部件包括多个处理器单元，以及为每个处理器分配的存储单元。各个处理运算单元之间使用双向数据线进行数据交互(图 1)。处理运算单元包括其核心处理单元——运算插值器(IU)，梯度运算的运算单元(GU)以及地址产生器(AG)等部件。其中三次线性运算单元是最基本的运算单元，在梯度计算过程，和重采样点的体素值的就算过程中会不断被重复使用。

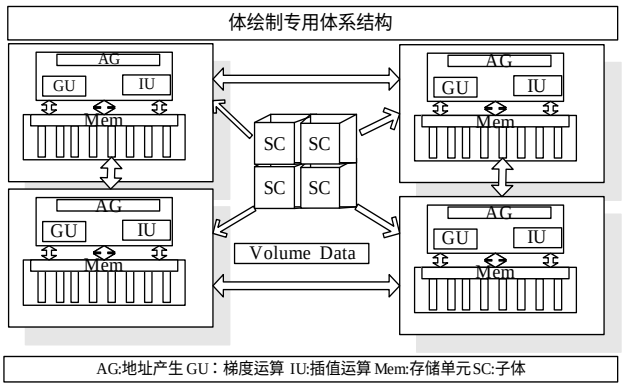

图 1 体绘制专用体系结构

将整个存储模型设计过程分为：处理单元级别的地址分配(外层分配)和存储器级别的地址分配(内分配)两个阶段。此外对于一部分处于两个子体交界位置的重采样点的数据值的插值运算，涉及分配到不同子体的体数据，这种例外处理方式在后面进行专门描述。

(1)处理单元级别的地址分配决定每个体素属于哪个处理器相对应的存储单元，这种外层分配相对简单。对于体素坐标为(x,y,z)的体素，要决定它属于哪个的处理器对应的存储单元，首先要确定(x,y,z)对应于哪个子体，进而决定子体被分配到的处理器。

假定处理器的个数为 P，划分出的子体的边长为 slen，对于边长为 vlen 体数据内部坐标为(x,y,z)的体素点，其所在子体的坐标可以使用以下方式计算：

$$(x_s, y_s, z_s) = (x/slen, y/slen, z/slen)$$

如果按照 Z->Y->X 的顺序遍历在同一体数据内的各个子体，坐标为 (x_s, y_s, z_s) 的子体的遍历序号 s_l 可以由以下公式给出：

$$s_l = x_s + y_s(vlen/slen) + z_s(vlen/slen)^2$$

因此，将子体依照序号顺序分配给处理器，序号为 s_l 的子体被分配到序号(s_l % p)的处理器，这样的分配方式也是由 Ray casting 算法特点决定的，交错的分配方式有利于各个处理器之间同时开始进行计算，而尽量减少等待其他处理器的运行结果的几率，更加有利于达到负载平衡。

(2)处理器内部存储单元级别的地址分配处理分配决定每个体素在所属存储结构内的分配位置。在同一处理单元内部，存储结构被分为 8 个存储器，以便进行八路交叉访问。存储单元级别的地址分配也分为两步进行：1)确定每个体素属于哪个存储器，可以用三位二进制来标识 8 个存储器，可以使用(x%2, y%2, z%2)来表示坐标为(x,y,z)的体素点所属的存储器；2)确定每个存储器内部的地址分配方式即如何确定体素(x,y,z)在其所属的存储器内的顺序。

在此之前先引入一个八元体素组的概念。一个重采样点周围的 8 个体数据点(假定坐标分别是(x,y,z)，(x,y+1,z)，(x,y,z+1)，(x,y+1,z+1)，(x+1,y,z),(x+1,y+1,z)，(x+1,y,z+1)，(x+1,y+1,z+1))为一个八元体素组(voxel octet)。一个八元体素组里 3 个方向都具有最小坐标值的那个坐标(这里是(x,y,z))为这个八元体素坐标组的基准坐标(Base Coordinates)。如果一个八元体素组的基准坐标 3 个方向上的坐标值都是偶数，那么称这个八元体素组为一个超级八元体素组。可以证明，每一个体素坐标必然对应着唯一一个超级八元体素坐标组。

坐标为(x,y,z)的体素在所属的存储器内的分配顺序可以使用一个四元组 $(s_l, z_{ls}, y_{ls}, x_{ls})$ 来表示。其中 s_l 表示子体在体数据内部的遍历排序，而(z_{ls}, y_{ls}, x_{ls})表示体素(x,y,z)所在的超级八元组在第 s_l 个子体内的相对坐标。(试图证明进行这样的存储分配不会导致两个不同的体素去占用相同的内存单元。事实上，两个属于不同的超级八元体素组的坐标不会占用相同的存储单元。而属于同一个超级八元体素组的八个坐标必然分属于 8 个不同的存储器。)(z_{ls}, y_{ls}, x_{ls})可以由下面的公式给出：

$$(z_{ls}, y_{ls}, x_{ls}) = ((z\%slen)/2, (y\%slen)/2, (x\%slen)/2))$$

这里，$(z\%slen), (y\%slen), (x\%slen))$ 是坐标为(x,y,z)的体素在所在子体内的相对坐标。各个方向上除以 2 即得到所在的超级八元体素坐标组的基准相对坐标。

经过上述变换。每一个体素坐标(x,y,z)都映射到了 8 个存储器组某一个内的一个存储区域，进行后续的插值计算了。在插值计算过程中，给定一个重采样点的坐标位置(a,b,c)，计算重采样点体数据值的时候需要通过对其周围的 8 个点进行线性插值运算来求得。对于坐标值(a,b,c)进行取整截取，即可获得其所在八位体素组的基准点坐标，进而获得其余 7 个点的坐标，从 8 个采样点取数据，在计算存储器地址的时候，没有必要针对每个坐标都计算一次，事实上，因为体素存放在哪个处理器是由 3 个方向的坐标对 2 取余得到的。所以，组成一个八元体素组的相邻的 8 个体数据必然分别属于不同的 8 个处理器。具体操作如下：

(1)如果这个八元体素组是超级八元体素组，那么这 8 个体数据分别在其所属的存储器内部具有相同的编址。也就是说拥有相同的四元组 $(s_l, z_{ls}, y_{ls}, x_{ls})$，因为四元组的第一维表示子体，后面三维表示所属于的超级八元体素组。

(2)如果所属的八元体素组不是超级八元体素组，那么这 8 个体素在各个存储器内的地址并不相同。这时的计算方法是对于坐标 (x_b, y_b, z_b)，若 x_b 是偶数，则 x_b 所对应的 x_{ls} 作为 8 个点的第一维坐标。如果 x_b 是奇数，那么将 x_{ls} 分配第一维为 x_b 的点，而将 $x_{ls}+1$ 分配给第一维为 x_b+1 的点。对于第二维，第三维采用相同的处理策略。

(3)光线投射过程中如果重采样点落在了相邻的两个子体之间，那么按照上述分配方法任何一个子体的处理器都无法得到全部的重采样点相邻的 8 个体素点的体素数据值。因此需要进行调整。有 3 种方式进行这种处理：1)对于图像要求精度不高的应用，可以在融合计算的时候为这个采样点补

充空值。而直接将这个重采样点的坐标向下传递，因为对于一般的体绘制应用中，有 70%的采样点的数据值是空值，空数据剔除本来就是在直接体绘制过程中的一个加速手段；2)在需要计算的时候从其他存储器向处理这个子体的处理器进行数据传递；3)对数据进行复制存储，在本子体内预先存放这些需要传递的数据。这种做法在需要更大的存储空间的同时有效地节省了计算时间，提高了计算精度，从而保证了实时绘制的质量。本文采用第 3 种方案，首先确定哪些数据需要存储，然后介绍对于需要重复存储的这些数据怎样进行地址分配和寻址。

首先确定需要存储的数据范围。对于一个长度为 slen 的子体，那么在这个子体周围与之相邻的共有 26 个子体，将这些相邻的子体按照相邻方式分为 3 类：(1)共面的，共有 6 个；(2)共边的，共有 12 个；(3)共顶点的，共有 8 个(图 2)。

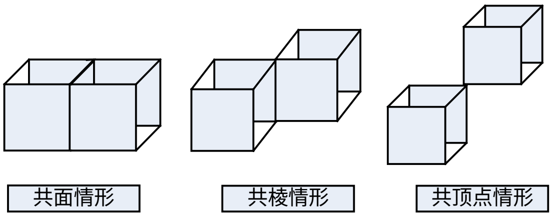

图 2 子体间相邻方式示意图

将属于同一子体，与同一子体相邻，且属于同一相邻方式的那部分体数据称为一个共邻体素簇。这样每个子体周围存在 26 个共邻体素簇，存储器结构进行如下划分：第 1 维由子体序列决定；第 2 维由 2 位二进制来决定，表示需要被存储的体素簇的类型；第 3 维表示体素簇在同一类型内部的遍历序号；第 4 维可以使用体数据在当前子体内部的超级八元体素组的序号来标识。这样完成重复数据的存储和访问。

3 总结

在软件加速提升空间不大，商用大型机资金消耗大的现实情况下，体绘制的专用体系结构是实现实时体绘制的一个有效的解决方案。本文所提出的并行多路存储方式适合于多处理器的体绘制体系结构，同时在单处理器情形下简化为八路交叉存储器的设计方案，能够在一个存取周期内取得一个重采样点的周围 8 个采样点的数据值，从而加快体绘制的处理速度。在提高速度的同时，尽量节省了重复数据的存储。这种数据存储设计思想同样适合在并行机群上实现体绘制中用来进行数据分配，从而减少集群节点之间的通信，提高加速比。

参考文献

1 Bertil S. Design of a Parallel Accelerator for Volume Rendering[C]//Proceedings of the 6th International Euro-par Conference on Parallel Processing. 2000: 1095-1104.

2 Barthold L. Design of a High Performance Volume Visualization System[C]//Proceedings of the ACM Siggraph/Eurographics Workshop on Graphics Hardware. 1997: 111-119.

3 Michael D, Michael M. A Memory Addressing and Access Design for Real Time Volume Rendering[C]//Proceedings of the IEEE Circuits and Systems. 1999.

4 Frank D, Kevin K, Baoquan C. High-quality Volume Rendering Using Texture Mapping Hardware[C]//Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware. 2001.

(上接第 274 页)

程中调用 ENCODEPTR，处理接收缓冲区线程中调用 DECODEPTR，在 EPA 应用实体中判断是否满足状态转换条件，满足则调用 HANDLEPTR。当收到请求报文，HANDLEPTR 实际完成 INDENTIFY 原语；当收到响应报文，它完成 CONFIRMED 原语。

```
typedef SYS_STATE (*DECODEPTR)(char* SockRecvBuf, SigEvent** pDatagram);
typedef void (*HANDLEPTR)(SigEvent* pDatagram);
typedef void (*ENCODEPTR)(SigEvent* pDatagram, char* SendBuf, Unsigned16* rLength);
typedef struct
{ DECODEPTR Decode;
  HANDLEPTR Handle;
  ENCODEPTR Encode;
}HandleTable;
```

4 测试结果

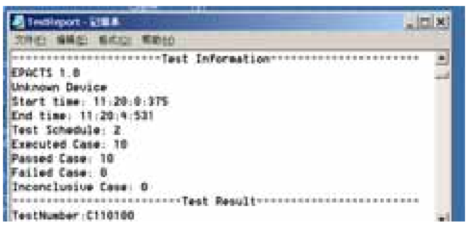

图 3 一致性测试报告示意图

实验室开发了基于 ARM9s3c2410 Cpu、DM9000 以太网控制芯片和 LQ035Q7DB02 触摸屏的 EPA 接入模块硬件平台。协议已稳定运行于此平台并通过了国家“ 863/CIMS ”专家组委托的专家测试，测试结果如图 3 所示。

测试表明在嵌入式 Linux 上实现的 EPA 通信协议完全达到了“ 用于工业测量与控制系统的 EPA 系统结构与通信规范 ”所规定的所有要求，各项测试均达到了相应的性能指标，且工作稳定可靠。

图 3 中 Test Information 栏中 EPACTS 1.0，表明测试平台软件为 1.0 版本；Test schedule：2 表明进行第 2 个测试计划；Execute Case：10 表明预测试 10 个服务；Passed Case：10 表明测试通过了 10 个服务；Failed Case：0 表明没有服务测试失败。

参考文献

1 Samek M. 嵌入式系统的微模块化设计——实时状态图 C/C++实现[M]. 北京: 航空航天出版社, 2004-08.

2 国家质量技术监督局. 中华人民共和国国家标准“ 用于工业测量与控制系统的 EPA 系统结构与通信规范 ”(报批稿)[Z]. 2005-08.

3 Noble J, Weir C. 内存受限系统之软件开发[M]. 武汉: 华中科技出版社, 2003-01.

4 Li Qing, Yao Carolyn. 嵌入式系统的实时概念[M]. 北京: 航空航天出版社, 2004.