文章编号:1671-6833(2009)04-0103-05

基于场景图的并行渲染系统研究与实现

谭同德,秦 鑫,赵新灿,张关锋

(郑州大学 信息工程学院,河南 郑州 450001)

摘 要:为满足大规模虚拟现实应用在渲染速度和显示分辨率等方面的要求,提出基于场景图技术的并行渲染,使用PC集群构建了高性价比的分布式图形系统.利用场景图在视景体内的快速裁剪技术,有效提高图元的归属判断速度.研究了OpenFlight格式与sort-first任务粒度划分的关系,基于场景图的层次结构来划分图元组.在保留模式下实现了一个基于场景图的sort-first原型系统,并用软件方式实现了多显示通道的无缝拼接.该系统同时考虑了CPU的功能并行和GPU的数据并行,提升了大规模场景漫游时的运行速度.

关键词:场景图;sort-first;归属判断;粒度

中图分类号:TP 391.9 **文献标识码**:A

0 引言

大规模场景漫游系统等沉浸式虚拟现实应用,在几何数据的产生、图形计算、几何指令发射和显示分辨率等方面要求很高,而孤立的图形系统受到限制[1].专业的图形工作站由于价格过于昂贵,一般用户无法承担.因此,选用廉价的PC集群来构建并行渲染系统,具有较高的性价比和良好的扩展性.

按照并行渲染操作介入到渲染流水线的位置,并行渲染系统的体系结构分为sort-first、sort-middle和sort-last 3种[2].其中,sort-first结构的并行渲染在几何处理阶段通过预变换判断图元对象在屏幕上的对应位置,然后将图元对象发送到相应流水线进行渲染,每条渲染流水线负责屏幕上的一块区域,最后将每条流水线的渲染结果拼合为最终图像.Sort-first结构最大程度的保持了图形渲染流水线的完整性,易于使用软件实现,也适用于多投影仪的大屏幕显示,很多集群渲染系统都采用了这种结构.其中代表性的有:WireGL是第一个sort-first[1]结构并独立于硬件平台的完备的集群渲染系统,但是它通过对OpenGL几何指令的分析获得图元数据,无法利用整个场景的空间结构特性;Princeton大学的Scalable Display Wall[3]采用多屏幕拼接技术实现分辨率的可扩展性;浙江大学CAD&CG国家重点实验室的Ant-Force[4]是一个数据分布型的sort-first原型系统,它采用cell结构来控制粒度,但为了支持大规模的场景定义和并行,必须建立更高级的数据结构.

对此,笔者利用场景图的空间层次结构研究了sort-first并行渲染系统,重点研究了图元归属判断和任务粒度划分两个问题.笔者还实现了一个保留模式原型系统,使用软件的方式实现了大屏幕多投影仪的无缝拼接.

1 场景图技术与sort-first图元归属判断

1.1 场景图技术简介

场景图(scene graph)是一种描述三维场景的树状数据结构,是一个有向非循环图(Directed Acyclic Graph, DAG),如图1所示.场景图的节点分为组节点和叶子节点,其中组节点对应于树结构的内部节点,如图1中的建筑1和房间2都是组节点,走廊和房间则为叶子节点.对场景的更新、裁剪和绘制等操作,都通过对场景图的遍历来完成.

OSG(Open Scene Graph)是一款日益流行的优秀开源场景图管理工具.它使用标准C++语言和OpenGL编写,能在多种操作系统平台上运行,广泛应用于可视化仿真、游戏、虚拟现实、科学可视化、地理信息系统以及建模等领域.OSG还是

收稿日期:2009-04-16;修订日期:2009-07-27

基金项目:河南省自然科学基金资助项目(0611051900)

作者简介:谭同德(1950-),男,河南郑州人,郑州大学教授,博士,主要研究方向为计算机图形学、虚拟现实、CAD.

一个高性能的图形库,它支持视景体剔除、隐藏面剔除、细节层次(LOD)、状态排序、顶点缓冲对象、OpenGL着色语言等技术.

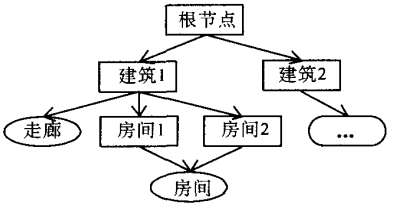

图1 场景图示例

Fig.1 Example of scene graph

1.2 图元归属判断

场景图为优化图形性能主要提供了裁剪和状态分类两项技术;其中裁剪的主要功能是剔除不在屏幕上出现的图形对象.场景图的层次结构,使得其裁剪操作非常高效;比如图1所示的场景图中,父节点建筑物下面有一些房间子节点,在访问到房间之前,整幢建筑物可能已经被裁剪掉,这样就节省了时间提高了效率.

结合场景图的快速裁剪技术,我们通过讨论屏幕空间与视景体的对应关系,提出基于场景图技术的图元归属判断,在3D空间里快速地确定图元在2D屏幕空间上的归属.如图2所示,屏幕空间与一个视景体,即平截头体相对应;对屏幕的划分,就可以理解为对平截头体的划分.图2中,O为视点,ABCD和A″B″C″D″分别是视景体的远近裁剪面.那么如果要将屏幕空间即投影平面A′B′C′D′划分为A′B′F′E′和C′D′E′F′,就可以理解为将视景体划分为ABFE-A″B″F″E″和CDEF-C″D″E″F″.落在某一视景体内的图元,最终将会在其对应的屏幕区域内出现.这样,

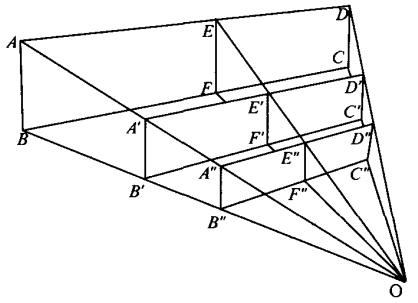

图2 屏幕空间与视景体

Fig.2 Screen space and view frustum

求出图元或图元组的包围球[5]后,利用场景图在视景体内的快速裁剪技术,就可以快速地判断图元归属.

同样,在sort-first系统中,将屏幕划分成N个子区域,就相当于把视景体划分为N个子视景体.划分视景体其实是改变了每个显示通道的投影矩阵,从而实现超大屏幕的分屏[6].

传统上常用的图元归属判断策略有两种.第一种策略先将图元变换到二维屏幕空间,然后根据屏幕划分发送到目标区域,Display Wall[3]采用的是这种策略.很显然这种策略多了一些没有意义的操作,它把一些最终不在屏幕上出现的图元也进行了2D变换.为了避免这种情况,可以在变换到屏幕空间前对场景进行裁剪.另外的一种策略是对一组图元直接求包围盒,将该包围盒变换到视景体内进行归属判断,WireGL采用的是这种方法[1].但是WireGL将操作指令和几何指令分离处理,对几何指令包内的几何数据的外包围盒作归属判断.显然从几何指令包内的几何数据看不到整个场景的空间逻辑性,基于场景图的图元归属判断则通过对场景图的层次裁剪提高效率.

2 OpenFlight与Sort-first任务粒度划分

2.1 OpenFlight数据格式

OpenFlight是MultiGen-Paradigm公司内置的3D文件格式,是世界上占主导地位的虚拟数据库标准,事实上已成为业界的标准格式.OpenFlight也为OSG等场景图工具所支持.

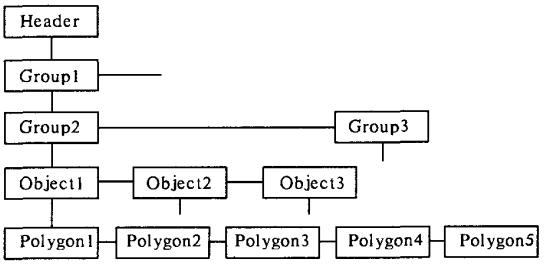

图3 OpenFlight数据格式

Fig.3 OpenFight data format

OpenFlight是一种典型的层次型数据存储格式,如图3.多个相关的面或多边形可以组成一个体节点(Object),若干体节点又被组织成一个组(group).这样OpenFlight把整个模型按照空间的位置分布和相关性,组成逻辑上的有层次结构的组,从而把空间有序地组织起来.

2.2 Sort-first的任务粒度划分

在sort-first结构的并行渲染系统中,由于大规模场景中包含的基本图元个数巨大,如果把每

个三角形面片作为任务拆分和调度的粒度显然是不现实的，那样只会形成很大的调度开销. 我们采取一种相对粗粒度的调度方式，把 OpenFlight 中的一个体节点作为一个图元组或划分为多个图元组，并将其编号，作为调度的粒度. 在 OSG 中，这种调度的粒度是 osg::Geode 类的节点，也是场景图的叶子节点.

OSG 读取 OpenFlight 模型文件时，保留了体节点上层的所有场景结构，即保留了所有组节点. 然后 OSG 对 OpenFlight 的体节点重新优化，组成一个或多个 osg::Geode 节点；每个 osg::Geode 节点下面包含若干个多边形面片.

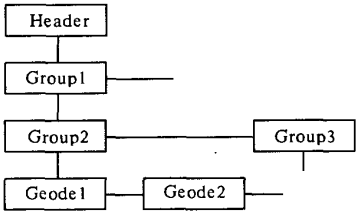

图 4　OSG 场景图数据格式

Fig. 4　Open scenegraph data format

与图 3 所示的 OpenFlight 格式数据对应的 OSG 格式数据如图 4 所示. 图 3 中的所有组节点及它们所蕴含的空间层次信息都保留下来. 但图 3 中的体节点依据面片之间的顶点共用性和空间紧凑性被重新组织.

这样由于场景图的空间逻辑性，处在同一粒度下的图元在空间上具有很强的紧凑性. 在判断图元归属时，只需判断图元组，即场景图叶子节点的归属. 对于同时落在多个子视景体内的图元组，则需要向对应的每个渲染进程调度该粒度. 调度开销很小，因为我们调度的只是图元组的编号，不会因此而形成网络负担.

3　基于场景图的 sort-first 并行渲染系统

3.1　系统架构设计

基于上面对图元归属判断和任务粒度划分的分析和研究，本系统中的 PC 节点设计为两类，一类是控制节点，一类是渲染节点.

(1)控制节点. 控制节点的主要功能是判断图元组的归属，向对应的渲染节点发送图元组的编号. 控制节点还要保证各个渲染节点的同步.

控制节点中只保留模型的场景图结构和每个图元组的包围盒等信息，而不保存具体的场景图数据，故占用很少的内存. 控制节点没有具体的渲染任务，对图形卡性能没有要求.

(2)渲染节点. 渲染节点的主要功能是负责某一屏幕区域的图元组的渲染工作.

在多通道的投影系统中必须解决边缘融合问题，以有效消除多投影仪之间的拼缝，使拼接后的画面自然统一. 所以渲染节点还需要在渲染工作完成后和渲染结果输出前，对帧缓存中的数据进行融合边缘部分图像的亮度的处理. 渲染节点的边缘融合处理，需要采用一定的亮度渐变函数，通常采用二次曲线函数.

3.2　系统工作流程

系统的工作流程如图 5 所示. 渲染流水线过程的并行包括功能并行、数据并行、时间并行及它们的混合并行[7]. 在本系统中，裁剪过程与绘制过程分离，在控制端利用场景图的快速裁剪功能计算图元的归属，减轻了渲染节点 CPU 负担，在渲染节点和控制节点之间实现 CPU 的功能并行化. 将渲染任务分布到各个渲染节点，有效减轻渲染节点 GPU 的负担，在各个渲染节点之间实现 GPU 的数据并行化.

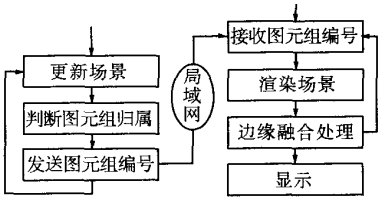

图 5　基于场景图的 sort-first 系统的工作流程

Fig. 5　Workflow of scenegraph based sort-first system

由于控制节点向渲染节点传输的是图元组的编号和其他一些控制信息等少量数据，所以网络传输采用可靠的 TCP 协议.

3.3　系统性能分析

在一般的场景渲染程序中，渲染出每一帧图像都要经过对场景图的 3 个遍历：更新遍历、裁剪遍历(时间开销计作 t_{cull})和绘制遍历(时间开销计作 t_{draw}). 其中更新遍历主要修改场景图的参数，实现动态场景，在运行中耗时比例很小，可以忽略. 所以渲染出每帧图像所需时间 t 为

$$t = t_{cull} + t_{draw} \tag{1}$$

在本系统中，我们把控制节点中判断图元归属的时间计作 t_{belong}，发送图元组编号和同步信息等数据所用时间计作 t_{send}；把渲染节点中渲染时间计作 t_{render}，边缘融合处理所用时间计作 t_{merge}，接收控制节点发送数据的时间计作 $t_{receive}$. 那么，本系统运行过程的时间分配情况如图 6 所示.

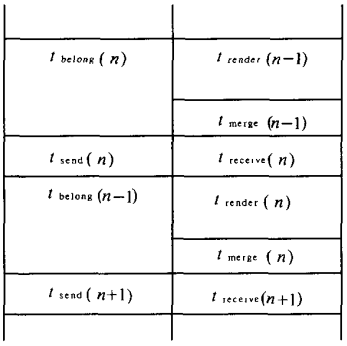

图6 系统运行过程的时间开销

Fig. 6 Time spent of system

图6的左侧是控制节点的时间开销,右侧为渲染节点,括号中的数字代表当前运行的是第几帧.控制节点与渲染节点之间并行工作,当渲染节点进行第 n 帧的渲染和边缘融合处理的时候,控制节点执行的是第 $n+1$ 帧的图元组归属判断操作.由于采用TCP协议同步传输数据,故:

$$t_{send} = t_{receive} = t_{net} \tag{2}$$

所以系统渲染出一帧图像所需要的时间为:

$$t' = \max\{t_{belong}, (t_{render} + t_{merge})\} + t_{net} \tag{3}$$

我们将式3与式1对比.式3中的 t_{belong} 相当于式1的 t_{cull},因为两个操作的内容基本相同.而 t_{render} 相当于 t_{draw} 的一部分.在整体屏幕空间相等且渲染节点大于1的情况下:$t_{render} < t_{draw}$.

由于控制节点与各渲染节点之间的并行性,控制节点的 t_{belong} 与渲染节点的 $t_{render} + t_{merge}$ 只取其较大者.

因为控制节点向渲染节点发送的数据量很小,所以式3多出的 t_{net} 在总体时间中比例很小,更不会构成系统的瓶颈.而且 t_{merge} 用于大屏幕的拼接当中,拼接的结果是获得了更高的分辨率.

4 实验结果分析

本系统运行的郑州大学新校区虚拟场景模型中眉湖一景的效果图如图7所示.运行时采用两个显示通道,图7的左图是系统的两个渲染端,右图是两个投影仪拼接后的效果.其中,两个渲染端的输出分辨率为800×600,图像融合宽度为194像素,融合后图像分辨率为1 406×600,投影面积为5.90 m×2.55 m,投影幕上的融合宽度为0.81 m.融合时采用的融合函数为余弦函数,拼接后的图像过渡平滑,给人很强的沉浸感.

由于微机性能的不断提高,以及测试场景规模等方面的不同,不同时期建立的sort-first系统的性能不具有可比性.对于系统的并行加速,我们使用单机与使用本系统分别对同样的场景作了一系列的对比测试.使用单机测试时的系统配置是:Pentium(R) D 2.80GHz CPU、2G内存、Radeon X1950 Pro显卡;使用本系统测试时,控制节点的显卡配置是:Radeon X550其它配置与单机测试时的配置相同.测试时实现的总分辨率为1 280×1 024.试验数据如表1所示.

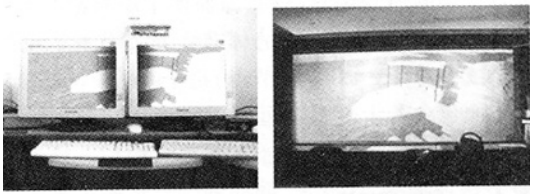

图7 双通道显示

Fig. 7 Two display channel

AntFore原型系统在中等规模的场景中获得最高的性能(双机的加速比为2.0左右)[4],此时本系统表现一般;而在单机很难实时漫游的大规模虚拟场景中,基于场景图技术的本系统则表现出较好的实时性能,获得很高的加速比.

表1 并行加速对比实验

Tab. 1 Speedup of sort-first parallel system

三角形面片数	图元组数	单机帧速/fps	sort-first帧速/fps	加速比
60 359	1 979	239	262	1.10
395 851	18 724	198	258	1.30
538 819	14 227	187	251	1.34
1 062 162	38 741	48	82	1.71
1 122 521	40 720	4	75	18.75
1 182 880	42 699	0	72	Very large
1 518 372	59 444	0	68	Very large
1 661 340	54 947	0	10	Very large

5 结束语

基于场景图的sort-first系统,利用场景图的快速裁剪功能计算图元的归属,减轻了渲染节点CPU负担;同时,把渲染任务分布到各个渲染节点,还有效地减轻了渲染节点GPU的负担,从而提高系统的整体性能.采用场景图的叶子节点作为任务调度的粒度,不仅保持了场景图的空间逻辑性,还减小了调度开销,在保留模式下有效降低了网络负担.另外,基于开源的场景图API开发并行渲染系统软件,可以显著缩短开发周期,降低开发成本.所以基于场景图技术的sort-first系统具有很强的研究意义和现实意义.由于本系统要拼

接多台投影仪设备来构建投影墙,所以须静态分割屏幕空间,难以实现系统的动态负载均衡.下一步将要根据图元在屏幕上的分布情况动态地分割屏幕空间.

参考文献:

[1] HUMPHREYS G, ELDRIDGE M, BUCK I, et al. A Scalable Graphics System for Cluster [C] // Proceedings of Siggraph 2001. 2001:129 - 140.

[2] MOLNAR S, COX M, ELLSWORTH D, et al. A Sorting classification of parallel rendering [J]. IEEE Computer Graphics and Applications, 1994, (7):23 - 32.

[3] LI K, CHEN H, CHEN Y, et al. Building and using a scalable display wall system [J]. Computer Graphics and Applications, 2000, (12):29 - 37.

[4] 金哲凡,林　海,石教英. 数据分布型 sort-first 并行图形绘制系统的研究与实现[J]. 计算机研究与发展,2004,(2):376 - 382.

[5] 黄晓生,顾景文. 一种 Sort - irst 架构的基于包围球的归属判断策略[J]. 计算机应用与软件,2007,(10):67 - 69.

[6] 孙益辉,陈福民. 超大屏幕实时交互分布式渲染平台关键技术[J]. 计算机应用,2008,(6):230 - 234.

[7] CROCHETT W. Parallel Rendering [R]. Technical Report TR95-31, Institute for Computer Application in Science and Engineering, NASA Langley Research Center, April 1995.

Research and Implementation on Scene Graph based Parallel Rendering System

TAN Tong - de, QIN Xin, ZHAO Xin - can, ZHANG Guan - feng

(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: Scene graph - based parallel rendering is proposed to build cost - effective distributed graphics system based on PC cluster in order to meet the need of large scale virtual reality applications. The procedure of primitives belongingness determining is speeded up using the fast view frustum culling technology of scene graph. The relationship between OpenFlight scene data format and the task granularity of sort-first is also researched and primitive groups are built based on the hierarchy of scene graph. We have realized a scene graph based sort-first prototype system which achieved seamless tile using the software way in retained mode. Both the function parallel of CPUs and the data parallel of GPUs is considered in this system, and it could improve frame rate while We ran large scale virtual scenes.

Key words: scene graph; sort-first; belongingness determining; granularity