文章编号:1001 - 9081(2009)01 - 0245 - 03

基于 Cell 多核的光线投射并行算法

冯高锋[1,2]
(1. 中国科学院 计算技术研究所,北京 100190; 2. 中国科学院 研究生院,北京 100049)
(gf. feng@ gmail. com)

摘 要:如何充分利用多核异构系统的性能优势为实际应用服务正在成为新的研究热点。以图形算法为例,通过对光线投射体绘制算法的过程进行功能分解,并采用按行分块的静态分配策略,对光线投射体绘制算法研究实现了在 Cell B. E. 多核异构系统上的并行算法设计和开发,并使用 SIMD 等方法进行了性能优化。性能评测显示其具有较好的加速比和可扩展性,优化后性能提升明显。

关键词:Cell;SIMD;光线投射体绘制算法;静态分配;并行算法
中图分类号: TP301 **文献标志码**:A

Parallel ray casting algorithm based on muti-core cell processors

FENG Gao-feng[1,2]
(1. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080, *China*;
2. *Graduate University of Chinese Academy of Sciences, Beijing* 100039, *China*)

Abstract: How to make full use of high performance of multi-core heterogeneous platform to support practical application is becoming a new research hotspot. Take graphic algorithms for example, depending on balance allocation study of Ray Casting, the paper proposed a designing method to develop a parallel ray casting application in Cell B. E multi-core heterogeneous system with function decomposition and the row-block static allocation. SMID method was used to optimize the performance. Testing results show that the speedup and scalability is fine, and the performance in Cell is 14 times better than SMP server.

Key words: Cell; SIMD; ray casting; static allocation; parallel algorithm

0 引言

随着多核系统成为一种趋势,多核系统上的并行算法和程序设计方法研究成为新的热点。特别是 Cell B. E. 异构多核系统,其异构多核体系结构、强大的计算能力和高速通信带宽等特点,为众多的多媒体、图形学等应用带来了机遇,同时也为并行算法和程序设计和实现带来了挑战[1,4]。

光线投射算法[3](Ray Casting)是一种基于图像空间的经典体绘制算法[2,6],其基本原理是根据视觉成像原理,构造出理想化的物理视觉模型,即将每个体素都看成为能够透射、发射和反射光线的粒子,然后根据光照模型或明暗模型,依据体素的介质特性得到它们的颜色和不透明度,并沿着视线观察方向积分,最后在象平面上形成具有半透明效果的图像。光线投射体绘制算法采用梯度表示等值面的法向能较好地反映物质边界的变化,使用 Phone 等光照模型,引入镜面反射、漫反射和环境反射能得到较好的光照效果,达到较好的逼真效果。此外,采样步长可调,因而能反映局部细节信息,因而图像质量比较高,特别适用于绘制区域特征模糊、体素特征相关性高的三维图像。但是需要巨大的计算量,并且计算量与体数据大小和图像分辨率等呈正比例;因为此算法需要对屏幕上的每一个像素进行操作,而且,当观察方向发生变化时,数据场中的采样点之间的前后关系也发生变化,这样就要进行重新采样,因此计算量极为庞大。

由于光线投射体绘制算法具有海量数据处理和高密度计算等代表性难点,本文以光线投射体绘制算法为例,研究在 Cell B. E. 多核上的并行算法和程序设计方法。

1 Cell 体系结构

Cell BE(Cell Broadband Engine)[5]处理器包括一个基于 PowerPC 架构的控制处理单元(Power Processing Element, PPE),8 个基于 SIMD 的协处理器单元(Synergistic Processing Elements, SPE),以及用于连接 PPE、SPE 和输入输出单元的高速环形数据总线(Element Interconnect BUS, EIB)。

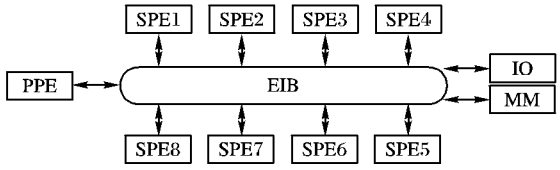


图 1 CELL 体系结构

PPU 一般负责运行操作系统,创建、管理和维护 SPE 线程,而 SPE 一般负责主要的计算任务。PPE 与 SPE 的一个关键的区别在于它们访问内存的方式。PPE 通过 load 和 store 指令来直接访问主存和寄存器。而 SPE 直接访问其本地存储器(Local Storage,LS),通过 DMA 方式访问主存,由 DMA 控制器负责将指令和数据由内存读取到 LS 或由 LS 写入到内存。这种方式能够有效地减小内存读写延迟对处理器利用率的影响,因为 SPE 的计算和 DMA 传输可以并发地进行。每

收稿日期:2008 - 07 - 29;**修回日期**:2008 - 10 - 05。
作者简介:冯高锋(1978 -),男,河南济源人,硕士研究生,主要研究方向:高性能计算、生物信息、网格技术、流计算。

一个 Cell BE 处理器包含 8 个协同处理单元(SPE)。SPE 每周期发散两条指令,有各自独立的 128 位向量处理单元,包含 128 个 128 位寄存器,达到 32 GFlops 的流水浮点单元以及 32 Gops 的 4 流水整数单元。为了解决 Cache 设计带来的复杂性并提高性能,每个 SPE 一个 256 kB 的不同于 Cache 结构的本地存储器。SPE 通过寄存器操作来读写本地存储器。本地存储器在 MFC 控制下,以 1024 位的块来访问主存,但是 SPE 并不能直接访问主存。

2 光线投射体绘制并行算法

目前,主流的光线投射体绘制大致可以分为三个阶段:1) 三维体数据的预处理;2) 利用视觉成像原理,对屏幕图像上每个像素的颜色计算;3) 结果图像的显示。其中的第二阶段是整个算法的关键,主要过程是:根据视线方向,从屏幕图像空间中的每一个像素点出发,投射出一条穿过整个三维体数据场的光线;沿这条光线进行采样,并经过插值、分类、明暗计算等过程,求出采样点的不透明度和颜色值;最后将这条光线上的所有采样点的颜色值和不透明度进行合成,得到屏幕图像空间上该像素点的最终颜色值——这是从反方向模拟光线穿过物体的全过程。

为了实现基于 Cell BE 体系结构的光线投射体绘制,我们首先对光线投射体绘制算法的过程进行功能分解,由 PPE 负责初始化、预处理以及 SPE 线程的管理;由 SPE 负责主要的绘制计算。接着,探讨具体的 SPE 间的任务划分和分配策略,以及相应的性能。最后,采用 SIMD 等方法进行相应优化。

2.1 任务分配策略

本文对光线投射体绘制算法的过程进行功能分解,由 PPE 负责体数据的读入、物体相机灯光等场景设置、创建维护 SPE 线程和结果图片的集中显示等功能;由 SPE 负责最主要的绘制流程,包括相交测试、坐标变换、数据插值、数值映射、光照效应计算和颜色组合等(如图 2 所示)。

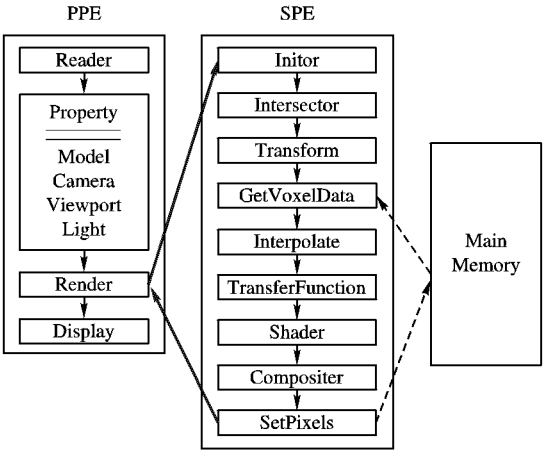


图 2 PPE 和 SPE 的功能示意图

通过对动态和静态创建 SPE 线程的性能进行对比,本文采取静态的 SPE 创建策略,由 PPE 根据物理机器和实际运行情况,创建固定的 SPE 线程,这样能有效地减少 SPE 线程频繁创建和撤销的开销。

以前的光线投射体绘制往往采取一种类似于“任务池”的任务分配策略,即对屏幕图像空间进行分块,每一个为一个单独的任务,被称为“任务块”。这种分配方式的优势是有利于负载平衡,劣势是增加了额外开销。这些开销包括 PPE 把相应任务块的控制信息发送给 SPE,供 SPE 进行任务更新本地存储空间。由于需要 PPE 和 SPE 之间的同步以及 SPE 的本地存储空间和内存的数据传输,这类额外开销在实时计算中是不可忽略的,严重地影响绘制速度。

本文通过与“任务池”的任务分配策略进行比较,最终采取按行分块的静态分配策略。这种分配方式有如下优势:1) 同步开销较少。只需要初始化时任务块的控制信息,在运行中不需要同步等待新的任务块控制信息。2) 额外的数据传输开销较少。减少了 SPE 本地存储空间更新的次数,从而减少了与内存之间的数据传输。3) 每个 SPE 的像素绘制结果按次序存放,有利于进一步的结果输出的优化。

2.2 SIMD 优化

每次对投射光线进行采样前,都要对此射线和体数据的包围盒进行相交测试。若不相交,则体数据对此像素点没有影响,此像素点呈背景颜色;若相交,则需要求出最近交点和最远交点,在交点之间按需要进行采样,以这些采样点的颜色贡献来代表整条线段上的点对像素点的贡献。

在重心坐标法的相交测试中,一旦射线与面片相交,则需要 4 次内积运算、2 次外积运算和 3 次分支判断。其中最费时的就是外积运算,因此外积运算的向量化是相交测试向量化的关键。V_1 和 V_2 的外积向量化大致可以分为 6 步:1) 把三维向量放入四维的单浮点向量;2) 把单浮点向量的第一个元素复制到第四个元素中,得到新的向量 V_1' 和 V_2';3) 对 V_1' 和 V_2' 向左循环移动一个元素,得到新的 V_1' 和 V_2';4) V_1 与 V_2'、V_1' 与 V_2 分别相乘得到 V_1'' 和 V_2'';5) V_1'' 与 V_2'' 相减得到新的向量 Vr';6) 利用“洗牌” 函数对 Vr' 重新排位,得到最终的结果 Vr。具体过程如图 3 所示。

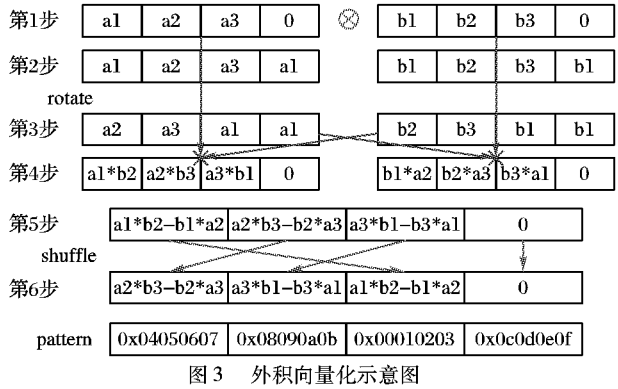


图 3 外积向量化示意图

光线投射体绘制中根据世界坐标或视窗坐标进行采样,当需要进行插值或读取体元数据时,需要先进行坐标变换,在物体坐标中进行插值和访存。坐标变换的次数正比于采样、插值和访存的次数,因此也是光线投射体绘制的瓶颈之一。

齐次坐标变换向量化大致可以分为 6 步:1) 按齐次坐标值和矩阵的行分别初始化单浮点向量 Va、Vr_1、Vr_2、Vr_3 和 Vr_4;2) Vr_1 和 Vr_3 分别与 Va 相乘得到新的向量 Vr_1' 和 Vr_3',被分为第一组,Vr_2 和 Vr_4 分别与 Va 相乘得到新的向量 Vr_2' 和 Vr_4',被分为第二组;3) 利用两个“洗牌” 函数分别对两组的元素重新选择和排位,分别得到 Vr_1'' 和 Vr_3''、Vr_2'' 和

Vr_4''(具体变换如图4第3步所示);4) Vr_1'' 与 Vr_3''、Vr_2'' 与 Vr_4'' 分别相加,得到 Vr_1 和 Vr_2;5) 对 Vr_1 和 Vr_2 进行如第3步的"洗牌"操作,得到新的向量 Vr_1' 和 Vr_2';6) Vr_1' 与 Vr_2' 相加,即得到最终的结果 Vr。具体过程如图 4 所示。

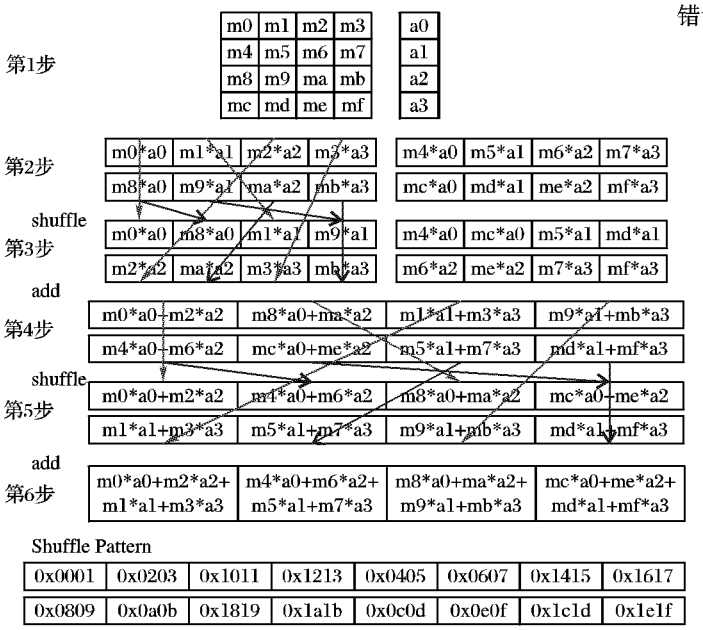


图 4　齐次坐标变换向量化示意图

3　性能评价

本文的实验平台主要是 IBM BladeCenter QS20 刀片服务器和 PlayStation 3 游戏机。QS20 刀片服务器拥有两颗 3.2 GHz Cell 处理器芯片,内存 1 GB,硬盘 40 GB,与双千兆以太网相连。本文用于实验的体数据主要有两个:

CT_Bone_1_250:587 × 341 × 250,47.7 MB,内容为头部骨骼的 CT 数据。

CT_Bone_1_341:587 × 1878 × 341,358 MB,内容为全身骨骼的 CT 数据。

SIMD 和分支停顿消除等 SPE 程序计算优化技术对 SPE 程序的性能有明显的提高。如图 5、6 所示,SIMD 和分支消除等计算优化技术提升了程序 25% 左右的性能。特别是 SIMD 技术,经过对相交测试、齐次坐标变换、光照效应计算和颜色合成等关键步骤的向量化,能极大地提高程序性能,对部分应用可以提高 20% 左右的性能,如图 5 所示。SIMD 技术不仅减少了多余的标量转化为向量的费时操作,还充分地利用了众多的向量寄存器(128 个 128 位寄存器)。

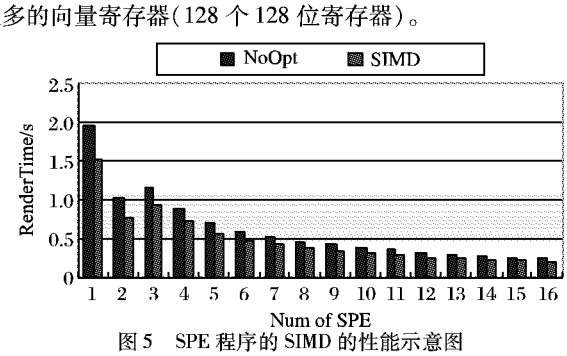


图 5　SPE 程序的 SIMD 的性能示意图

此外,经过软件分支预测、内联函数、循环展开等分支停顿消除技术的优化,SPE 程序能获得 10% 左右的性能提升(图 7)。这主要是因为 SPE 不提供对分支预测的硬件支持,在默认的情况下,SPE 遇上的每个分支都不会执行。如果从错误的分支中恢复需要 18 到 19 个周期——相对于 SPU 指令一般只需要 2 ~ 7 周期而言,惩罚是相当大,将会严重影响 SPE 程序的性能。而分支停顿消除技术能够有效地降低分支错误惩罚的影响。

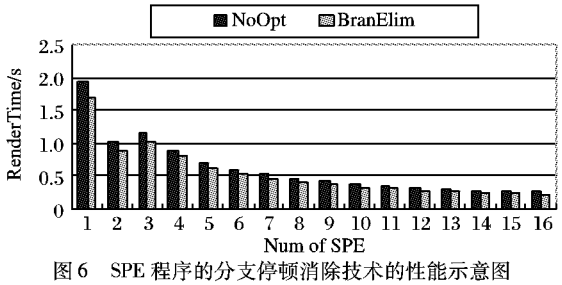


图 6　SPE 程序的分支停顿消除技术的性能示意图

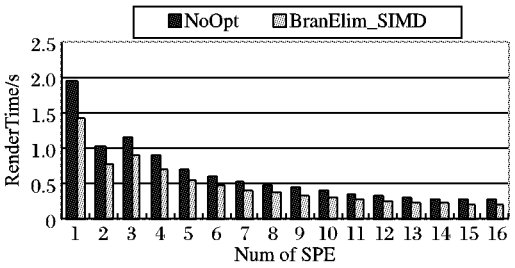


图 7　SPE 程序的计算优化性能示意图

此外,基于 Cell B. E. 的光线投射并行体绘制具有良好的加速比和可扩展性。如图 8 所示,开始增加 SPE 个数时,程序具有几乎线性的加速比;随着 SPE 个数的增加,任务分配和结果收集的开销降低了并行带来的性能提升,加速比急剧下降;最终,SPE 增多带来的并行性优势最终占主导地位,可扩展性良好。

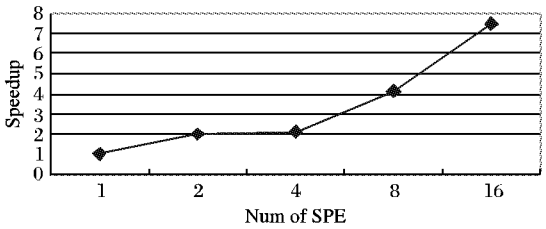


图 8　SPE 程序的加速比

本文对基于 Cell B. E. 的 QS20 上的绘制速度与 PC、SMP 上的绘制速度进行了对比。其中,PC 的 CPU 为 Intel Pentium 4 2.4 GHz,内存 512 MB;SMP 拥有两个 AMD Opteron 1.6 GHz 的处理器,内存 1 GB。如图 9 所示,从测试结果可以看出,基于 Cell B. E. 的光线投射并行体绘制在 QS20 上能获得 0.20 秒/帧的速度——体数据大小为 587 × 341 × 250,屏幕图像分辨率为 256 × 256,图像空间采样率为 2,物体空间采样率为 10。其速度分别是 PC 上速度的 20 倍,是 SMP 上速度的 14 倍。

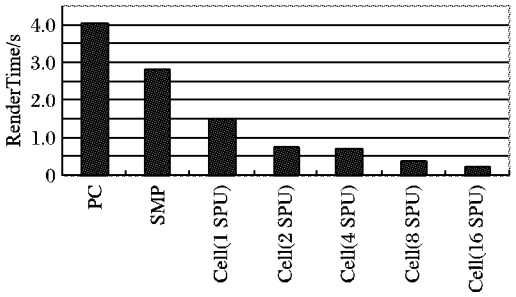


图 9　PC、SMP 和 QS20 上的性能对比图

(下转第 260 页)

处理器,NVIDIA GeForce 8600 GT 显卡,1 GB 内存的PC机上,利用VC++7.1 和 OpenGL 图形库实现。当雷达扫描挡板半径 *Radius* 为 40, k_4 为 0.75,不同 k_1,k_2 值得出的绘制帧率见表1。表中:k_1 为 0.8,k_2 为 30 时的平均绘制速度为 290.450 fps,比 k_1 为 1.0,k_2 为 35 的平均绘制速度 212.633 fps 提高了 36.6%。

图4(a)由于 k_1,k_2 取值过小,曲线不够连续,效果不够逼真;(b)因 k_1,k_2 取值过大,显得曲线不够光滑而失真;并影响了绘制速度。由上可以得出,当雷达扫描挡板半径 *Radius* 为 40,k_4 为 0.75 时,k_1 取值为 0.8,k_2 取值为 30 效果最佳。

表1 *Radius* = 40, k_4 = 0.75 不同 k_1,k_2 值的帧率表

k_1	k_2	*Particle_num*	k_3	帧率/fps
0.5	25	20000	0.00133	352.733
0.5	30	24000	0.00111	344.033
0.5	35	28000	0.00095	332.150
0.8	25	32000	0.00133	316.667
0.8	30	38400	0.00111	290.450
0.8	35	44800	0.00095	256.450
1.0	25	40000	0.00133	280.183
1.0	30	48000	0.00111	242.750
1.0	35	56000	0.00095	212.633

5 结语

本文所实现的雷达扫描系统满足了系统实时性和逼真性的要求,取得了一定的效果。以后的工作中可以进一步的考虑该系统的移植性和适应性。另外在实时性方面,可以考虑三角函数查找表及粒子空间位置信息的重复利用等方面。

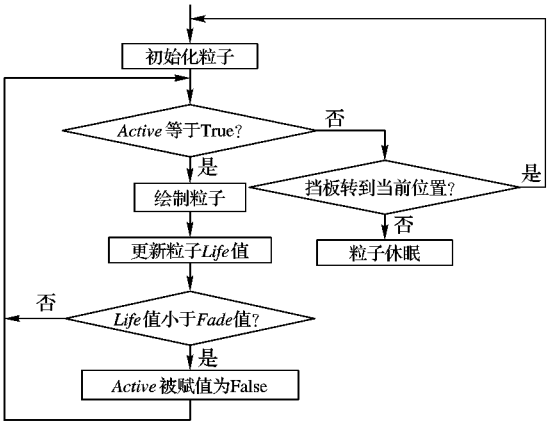
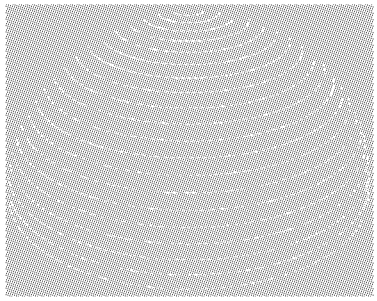
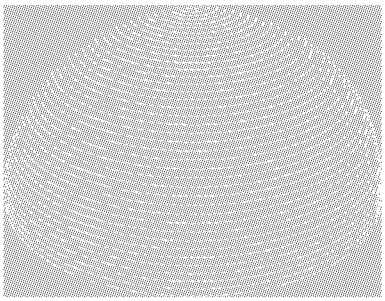


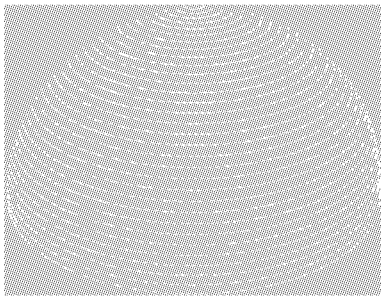
图3 余辉粒子更新流程



(a)k_1=0.5,k_2=25



(b)k_1=1.0,k_2=35



(c)k_1=0.8,k_2=30

图4 不同 k_1,k_2 雷达扫描效果

参考文献:

[1] 熊壮, 方兴, 李松维, 等. 基于 OpenGVS 的雷达仿真[C]// 第五届全国仿真器学术会论文集. 长春:[s.n.], 2004.

[2] 樊世友, 杨作宾, 孙书鹰, 等. 基于余辉模型的 P 型雷达显示器计算机仿真[J]. 计算机仿真, 2003,20(4):6-8.

[3] 刘翠海, 温东. 光栅扫描显示器上实现 PPI 雷达长余辉仿真[J]. 计算机仿真, 2002,19(2):25-27.

[4] REEVES W T. Particle systems - A technique for modeling a class of fuzzy objects [J]. Computer Graphics, 1983, 17(3): 359-376.

[5] 詹荣开, 罗世彬, 贺汉根. 用粒子系统理论模拟虚拟场景中的火焰和爆炸过程[J]. 计算机工程与应用,2001,37(5):91-92.

[6] WRIGHT S R, LIPCHAK B. OpenGL 超级宝典 [M].3 版.北京: 人民邮电出版社, 2005.

(上接第 247 页)

参考文献:

[1] BENTHIN C, WALD I, SCHERBAUM M, *et al*. Ray tracing on the cell processor[C]// IEEE Symposium on Interactive Ray Tracing. [S.l.]: IEEE Press, 2006:15-23.

[2] ELVINS T T. A survey of algorithms for volume visualization[J]. ACM Siggraph Computer Graphics, 1992, 26(3):194-201.

[3] LEVOY M. Display of surfaces from volume data [J]. IEEE CG & A, 1988, 8(3): 29-37.

[4] LOHR C, CHAPMAN D. Ray tracing on the cell broadband engine, CMSC635[R]. 2007.

[5] KAHLE J A, DAY M N, HOFSTEE H P, *et al*. Introduction to the Cell multiprocessor [J]. IBM Journal of Research and Development, 2005, 49(4/5): 589-604.

[6] KAUFMAN A, MUELLER K. Overview of volume rendering[M]. JOHNSON C, HANSEN C. Boston: Elsevier Academic Press, 2005.

[7] BADER D A, AGARWAL V, MADDURI K, *et al*. High performance combinatorial algorithm design on the cell broadband engine processor[J]. Parallel Computing, 2007, 33(10/11):720-740.

[8] CHEN T, RAGHAVAN R, DALE J, *et al*. Cell broadband engine architecture and its first implementation[J]. IBM developerWorks, 2005,49(4/5):589-604.