中图法分类号: TP391 **文献标志码**: A **文章编号**: 1006-8961(2011)06-1112-09
论文索引信息:杨猛, 吴恩华. 层次结构树木物理运动的实时仿真 [J]. 中国图象图形学报,2011,16(6):1112-1120

层次结构树木物理运动的实时仿真

杨猛[1),3)], 吴恩华[1),2)]
[1)](中国科学院软件研究所 计算机科学国家重点实验室, 北京 100190)
[2)](澳门大学科学技术学院电脑与资讯科学系,澳门) [3)](中国科学院研究生院, 北京 100049)

摘 要: 提出一种在 GPU 上实现基于力学运动原理的层次结构树木运动的并行仿真技术。该技术通过分析物理运动原理与多层次的矩阵结构(HMSM)算法的并行性,来达到将树木动画在图形硬件 CUDA 平台上进行加速的目的。首先介绍层次结构树木在外力诸如风力等作用下的物理运动原理;然后,针对树木的物理运动以及层次结构叠加算法详细地进行并行性分析;之后着重阐述 CUDA 框架下树木运动的并行结构设计过程与并行算法的详细设计方法;最后在 GPU 上执行树木物理运动仿真。实验结果表明,该技术不但能够生成真实感较强的树木动画序列,还能够实时模拟基于物理的树木运动。同时,该技术给计算机动画的加速算法提供了很好的思想。

关键词: 树木动画;基于物理;层级结构;CUDA 加速;实时

Simulation of physically-based tree animation in realtime

Yang Meng[1),3)], Wu Enhua[1),2)]
[1)](*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190 *China*)
[2)](*Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau China*)
[3)](*Graduate University of Chinese Academy of Sciences, Beijing* 100049 *China*)

Abstract: This paper presents a novel physically-based parallel approach to simulate tree motion using GPU in real-time. The tree's kinematic principle and the hierarchical matrix structure model (namely HMSM)'s parallel mechanism are analyzed to realize the speedup of the trees animation on CUDA. Firstly, we briefly introduce a method of physically-based tree motion called hierarchical matrix structure model (namely HMSM) driven by the external forces such as wind; then we analyze this model in a parallel management theoretically in detail; thereafter, describe the design of parallel structure and parallel algorithm of the tree's motion, which are suitable to implement on CUDA; finally, realize the physically based motion simulation on GPU. Experimental results show that the presented approach can animate trees realistically and naturally, and simulate physically based trees motion in real-time. Moreover, the approach give good clues for speeding up the algorithm for the computer animation.

Keywords: tree animation; physically-based; hierarchical structure; CUDA accelerate; real-time

0 引 言

计算机动画技术是当前计算机图形学中的研究重点[1],广泛应用于广告、动画片、计算机辅助设计、教育、军事等诸多领域。而自然场景的动画模拟与绘制一直是计算机图形学中的重中之重,由于自然界的错综复杂,以及人们认知的局限性和计算机发展的速度等因素,因此,自然界的仿真模拟一直是极富挑战性的工作。

收稿日期:2010-10-20;**修回日期**:2010-12-29
基金项目:国家自然科学基金项目(60833007, 60773030, 60973066);国家高技术研究发展计划(863)项目(2008AA01Z301);澳门科研基金项目。
第一作者简介:杨猛(1982—),男。中国科学院软件研究所计算机应用技术专业博士生,主要研究领域为计算机图形学,计算机动画。E-mail:yangm@ios.ac.cn。

众所周知,植物是自然界的重要组成元素。而植物的结构形态复杂,运动千变万化,例如随风轻轻摇摆的柳枝,狂风摇曳的树枝,微风轻抚的树叶,大雨滂沱中的小树,……,因而,如何能真实而快速地模拟植物是很多科研人员追求的目标之一。植物,特别是树木,有丰富的外部结构特征,由多种不同的器官和组织组成,并且因为受到光照、水源、病虫害等外界环境的影响,表现出千变万化的形态,而这些都是人们不能准确预知的;同时,浩瀚的大自然创造了许许多多令人惊奇的景象,如风、雨、雷、电等,人们对此亦是不能完全掌握其规律,因而仿真自然条件下树木的运动给研究人员带来了前所未有的难题。

虽然基于物理的动画在计算机图形学中已经有了很长的研究历史,但是植物仿真中却很少采用基于物理的方法。其主要的原因在于人们对于树木运动的机理的认知有限,导致在模拟中很多运动效果带有人工制造的效果;目前基于物理的方法需要大量的计算过程来真实地模拟事物的每一步变化,需要大量的时间消耗,并且一般都是串行的模拟过程,很难满足对其运动实时仿真的要求。因而,提出一种适合于在图形硬件运行的并行树木运动算法:通过对基于层次结构的树木在风等外力作用下的运动原理进行分析以及典型运动方式的并行性分析,真实地再现树木在外力下的实时动画效果。该算法给虚拟现实和计算机图形学等领域注入了新的活力,无论是对树木的模型模拟,还是对其运动仿真以及绘制,都给人们带来了强烈的视觉冲击。

GPU 的使用,由最开始的图形加速到目前的通用计算(GPGPU),已经有了革命性的发展[2-3]。它的并行海量数据处理能力,使很多串行算法进行了并行加速。采用当前 NVIDIA 公司设计 GPGPU 模型——CUDA 框架,很大程度上提高了程序运行的效率,节省了时间开销。在此基础上,提出树木动画的并行算法,设计树木运动适合并行计算的数据结构,真实地模拟了树木在自然风力下摇曳等实时动画效果,提高了树木动画的效率。

1　相关工作

植物的建模和动画仿真是图形学中的热点问题之一,目前已经有对于树木进行建模和在风等外力作用下的运动仿真算法,这些已有的方法很难快速地生成比较真实的动态仿真效果,难以满足实际应用的要求。为此,提出一种基于图形硬件的并行加速算法。

1.1　树木运动仿真

外力作用下的树木运动仿真,目前大致有基于物理、过程和数据驱动 3 种方法。Alagi[4]、Wu[5]、冯金辉等人[6]都提出了基于物理的风中树木运动仿真方法,生成的树木真实感较强;Ralf 等人[7]提出了一种基于物理指导的树木运动仿真方法,其并不是纯粹的基于物理,当模拟参数发生变化时,需要重新计算新的条件;Julien 等人[8]提出了一种预计算的方法,存储物理参数,当环境改变时,同样需要重新预计算后才能适应新的环境;Ota 等人[9-10]用一种基于 $1/f^{\beta}$ 噪声的纯过程性模型方法,该方法原理简单,运算量小,操作容易;Zioma 等人[11]应用一种基于 GPU 的过程性树木运动仿真方法,树木模型结构和运动原理简单;Dinener 等人[12]从视频中提取模型的运动数据进而重新合成模型的运动,而 James 等人[13]通过把预计算模拟数据作为输入,来得到模型对外力的真实感反应。

对于树木运动仿真,已有的物理方法原理复杂,计算量大,不适合实时模拟,而过程方法很难表述树木的真实物理运动,基于数据驱动的方法很难满足环境的变化,应用不够灵活。为此,提出一种基于物理的层次结构子树枝的并行运动仿真算法,结合过程性运动合成方法,极大地简化了树木的运动原理,减少了模拟过程的运算量,使操作更加简化,真实地再现了树木在风力等作用下的实时摇曳模拟。

1.2　通用 GPU 计算

目前,通用计算 GPU(GPGPU)[1-2]的并行流处理和可编程性能的快速发展给人们利用 GPU 提供了便捷的平台,使海量数据的并行计算成为现实,而将 GPU 和 CPU 结合起来使用更加发挥了 GPU 的高度并行性和 CPU 的良好控制性的优点[14]。因而,越来越多的人开始用其做一些非图形学方面的计算,这些计算涉及的范围很广,从图形输出流水线以外的非绘制处理,到几何计算、碰撞检测、运动规划、代数运算、优化计算、偏微分方程 PDEs(partial differential equations)数值求解等。自从 NVidia 公司发布计算统一设备架构(compute unified device architecture:CUDA)架构[15]以来,因为 CUDA 通用性和基于 C 语言的简便性,迅速得到普及,目前研究人员利用 CUDA 已经解决了很多领域的大量计算加速问题,诸如流体模拟、光线跟踪、体绘制等传

统图形学经典问题,以及 *N*-body 模拟、选举模拟、应用快速傅里叶变换技术的算法实现等非图形学的数学计算。

下面对 CUDA 并行计算的硬件基础进行简单的介绍。

在显卡的通用计算编程框架 CUDA[15] 中,GPU 芯片执行程序的基本单位是线程(thread),数个线程可以组成一个块(block)。执行相同程序的块可以组成一个网格(grid)(图 1)。以 NVidia G80 系列的 GPU 为例,拥有 16 个多处理单元(multiprocessor),共计 128 个线程处理单元(thread processor);每个处理单元中包含共享存储器(shared memory)、常数存储器(constant memory)和纹理处理器(texture memory),这些存储空间为多处理单元提供计算场所和数据通信等服务;同时,每个处理单元(processor)都含有数个寄存器(registers),为线程执行程序提供数据存储空间,当每个线程需要的寄存器数目多于可以分配的数目时,核(kernel)就会启动失败而退出;每个多处理器单元可以和具有很大容量的全局处理器(global memory)进行通信和计算。

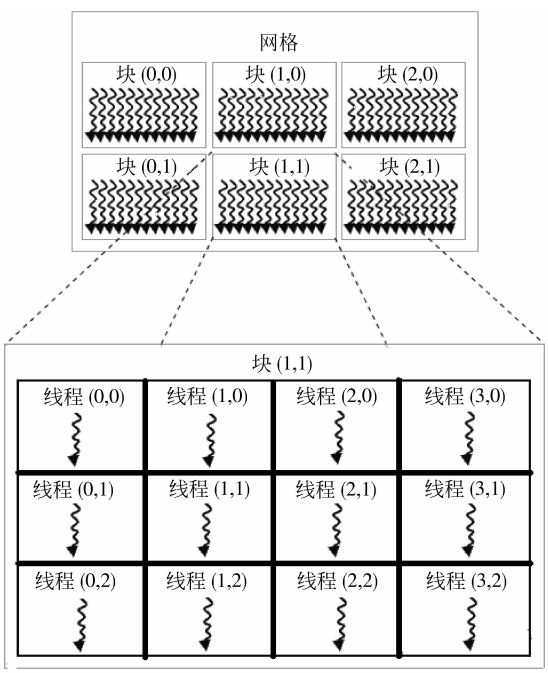


图 1 计算统一设备架构框架:线程、块、网格

Fig. 1 CUDA: grid of thread blocks

当前,一个核必须启用一个 GPU 设备(device),当不同的核要想同时运行时,只能考虑应用多设备(multi-GPU)来执行。核中对于线程和块的维度以及个数的选择,需要根据问题的大小以及数据的结构来做决定,不同的选择,效率相差可能会很大。

提出一种应用 CUDA 加速树木动画的基于物理的并行计算算法(图 2),重新设计了树木运动的数据结构,能够达到树木运动的实时仿真,提高了运行效率。

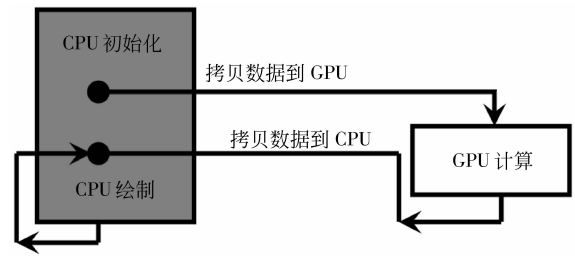


图 2 GPU-CPU 程序执行顺序

Fig. 2 GPU-CPU implementation

2 树木层次结构及其动力学运动原理

树木的结构复杂,不同的结构模型有不同的运动机制,即动力学原理;不同的模型和原理又决定着运动并行程度。本节主要描述树木的层次结构模型以及动力学原理,在此基础上,分析其运动并行性并提出并行仿真算法。

对于树木采用层次结构,其模型主要由多级的树干组织和树叶两部分组成。树木模型是由结点信息和段骨架信息来表示树枝的重心和树枝间的拓扑结构。图 3 展示了一棵完整树木的方向优先层次结构模型,不同的颜色代表不同的层次,同时也反映了各个层次之间的连接关系。在实际计算中,为了满足真实感的精度需要,通常树木模型采用 5 级层级结构树枝。如图 4 所示,大写字母 *O*、*A*、*B*、*C*、*D*、*E* 表示树枝的结点,线段 *OA*、*AB*、*AC*、*CD*、*CE* 分别表示每段树枝(段骨架),同时,也反映出树枝之间的拓扑连接。为了实现树木的动画,树干模型采用多级的层次模型结构,依据分层方法的不同,树干组织的层次模式可以分为宽度优先层次结构和方向优先层次结构两种。所谓的宽度优先层次结构是将某一个段骨架的所连接的所有段骨架中与此骨架宽度最接近的那个段骨架划为同一个层次;方向优先层次结构是将某一个段骨架的所连接的所有段骨架中与此骨架方向最接近的那个段骨架划为同一个层次。不失一般性,我们以方向优先层次结构为例,如图 4 中

的骨架段 OA,与 OA 连接的段骨架有 AB、AC,很容易通过计算知道,AB 的方向较 AC 的方向更接近于 OA,这样,我们将 OA 与 AB 视为同一层次。图 4 显示了 3 个层次的方向优先层次结构模型,为 *Level*0、*Level*1、*Level*2,其中 *Level*0 由结点 O、A、B 以及段骨架 OA、AB 组成,*Level*1 由结点 A、C、D 以及段骨架 AC、CD 组成,*Level*2 由结点 C、E 以及段骨架 CE 组成。

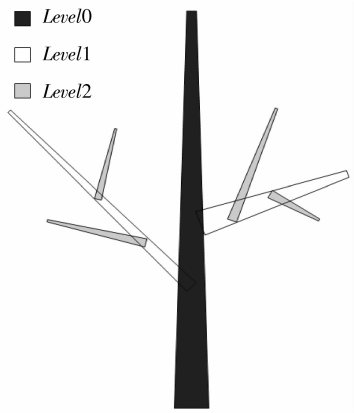


图 3　方向优先层次结构树木模型

Fig. 3　Hierarchical tree model by direction

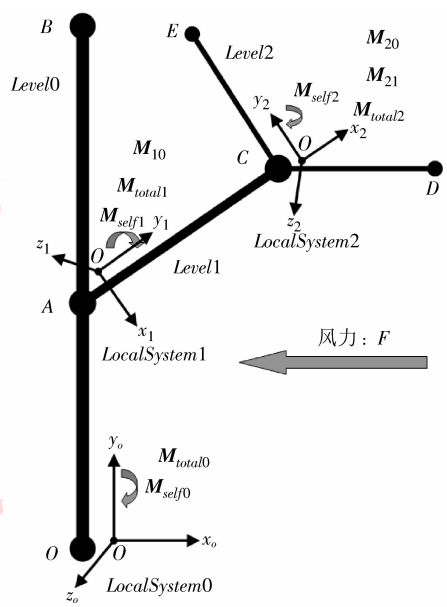


图 4　树木动画层级结构

Fig. 4　Hierarchical tree structure model

有了分层结构,我们可以计算不同层次间的相互联系。如图 4,在每个层次的根结点处定义局部坐标系,如 *Level*0 的根结点为 O,局部坐标系 *LocalSystem*0 定义为 y_o 方向与矢量 OA 一致,x_o 与 z_o 与 OA 垂直,同理可以定义 *LocalSystem*1、*LocalSystem*2;定义各个结点的局部坐标分别为 Lo、La、Lb、Lc、Ld、Le,世界坐标分别为 Wo、Wa、Wb、Wc、Wd、We;定义 $\boldsymbol{M}_{ij}$ 为由 *Level*i 到 *Level*j 的转化矩阵,$\boldsymbol{M}_{self}i$ 为第 i 级树枝在外力作用下绕自身局部坐标系 *LocalSystem*i 的旋转矩阵,$\boldsymbol{M}_{total}i$ 为第 i 级树枝相对于世界坐标系运动的总转换矩阵,其中 i,j 分别表示第 i 级和第 j 级层次。

树木的模型是树木动画的结构基础,本文采用对每一层次结构分别求解,之后通过层次间的级联关系进行累计积分的方法作为动画的具体实现策略。以下是对树木动画的详细分析。

自然界中,树木的运动受到诸多因素影响,比如重力、风吹、雨打等。风作为自然界中驱动树木运动的常见外力之一,风力数学模型对于树木运动仿真的真实感模拟起着至关重要的作用。目前已经存在了一些有关风模型的研究工作,比如空气动力学风模型[16],随机风模型[17],基于物理分析的风模型[18]以及风的数学模型[11]等。本文着重模拟树木因外力(如风力等)所产生的摇曳现象模拟,风模型定义为 2 维空间上的速率场,即以时间为自变量,风的大小为因变量的 4 种类型带噪声和周期的数学函数,可以真实模拟现实世界中的 4 种风速度场[11]:带紊乱的风 1(Y_1),带轻微噪声的光滑风(Y_2),带紊乱的风 2(Y_3) 以及带噪声的周期风(Y_4),其数学函数表达式分别为

$$Y_1 = \cos(\pi x) \times \cos(3\pi x) \times \cos(5\pi x) \times \cos(7\pi x) - 0.1 \times \sin(25\pi x)$$

$$Y_2 = (\sin(\pi x) + \sin(3\pi x) + \sin(5\pi x) + \sin(7\pi x))/4$$

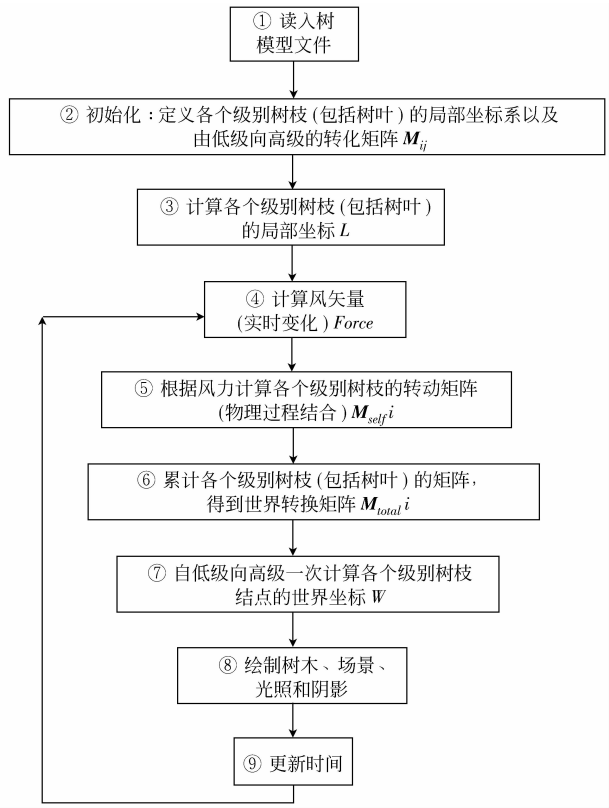
$$Y_3 = \cos^2(\pi x) \times \cos(3\pi x) \times \cos(5\pi x) - 0.02 \times \sin(25\pi x)$$

$$Y_4 = \cos^2(\pi x) \times \cos(3\pi x) \times \cos(5\pi x) - 0.1 \times \sin(25\pi x)$$

其中,πn 为相位(n 为常正数),调节风的频率;x 为时间;Y 为风力大小 *Force*(如图 4),其方向可由用户来确定。

应用一种基于物理和过程的相混合的树木动画方法,大致可以分为以下几步:首先初始化各局部坐标系下层次树枝的局部信息,包括局部坐标等;之后将 xy 平面内的 2 维风场映射到 3 维几何空间后,再映射到各层次结构的局部坐标空间;然后应用牛顿运动定律在局部坐标系下,求解各层次运动的角位

移矢量 $\boldsymbol{S}$;有了局部位移角 $\boldsymbol{S}$ 后,我们通过转化算法,将所有局部角位移累加,得到相应的世界坐标系下的角位移;最后,我们可以根据世界角位移,更新树枝每个结点的坐标,完成树木动画序列的生成。该树木动画算法的具体流程如下:



根据流程图,我们可以求得模型中每个节点在每一时刻的世界坐标 W,首先,根据牛顿第二运动定律 、运动学公式,以及欧拉迭代法算法,得到:

$$F = m \times a \tag{1}$$

$$V = V_0 + a \times t \tag{2}$$

$$S = S_0 + V \times t \tag{3}$$

初始条件:$V_0 = 0$;$S_0 = 0$;

离散化后:

$$V_n = V_{n-1} + a_{n-1} \times delta(t) \tag{4}$$

$$S_n = S_{n-1} + V_{n-1} \times delta(t) \tag{5}$$

式中,n 为迭代次数,$delta(t)$ 为时间步长。

有 $V_n - V_{n-1} = a_{n-1} \times delta(t)$,$\cdots$,$V_1 - V_0 = a_0 \times delta(t)$,依次相加

$\Rightarrow V_n = V_0 + \sum_0^{n-1}(a_n \times delta(t))$,$V_0 = 0$;

$\Rightarrow V_n = \sum_0^{n-1} f_n \times delta(t)/m$,记 $C_1 = delta(t)/m$,m 为质点质量;

$$\Rightarrow V_n = C_1 \times \sum_0^{n-1} f_n; \tag{6}$$

同理,由(3)得到:

$$S_n = \sum_0^{n-1} V_n \times delta(t); \tag{7}$$

将式(6)带入式(7)中可以得到

$S_n = C_1 \times ((f_0) + (f_0 + f_1) + \cdots + (f_0 + f_1 + \cdots + f_{n-1})) \times delta(t)$

$\Rightarrow S_n = C_2 \times ((n-1) \times f_0 + (n-2) \times f_1 + \cdots + 1 \times f_{n-1})$,记 $C_2 = delta^2(t)/m$

$\Rightarrow S_n = S_{n-1} + C_2 \times \sum_0^{n-1} f_n$,记 $F_{n-1} = \sum_0^{n-1} f_n$

$$\Rightarrow S_n = S_{n-1} + C_2 \times F_{n-1}; \tag{8}$$

且 $$F_{n-1} = F_{n-2} + f_{n-1} \tag{9}$$

我们知道 f_n 为子树枝在第 n 步迭代时所受的合外力,此处认为主要受到 3 个力的作用:外力 $f_{n-1}^{(Wind)}$,回复力 $f_{n-1}^{(Back)}$,以及子树枝在运动过程中受到的阻尼力 $f_{n-1}^{(Damp)}$。即

$$f_n = f_{n-1}^{(Wind)} + f_{n-1}^{(Back)} + f_{n-1}^{(Damp)} \tag{10}$$

根据式(8)—(10)进行叠加,我们很容易用第 $n-1$ 步的结果 $\boldsymbol{S}_{n-1}$、$\boldsymbol{F}_{n-1}$ 等计算出第 n 步的值 $\boldsymbol{S}_n$、$\boldsymbol{F}_n$。最终迭代计算公式依次为

$$\begin{aligned} f_{n-1}^{(Back)} &= K \times S_{n-1} \\ f_n &= f_{n-1}^{(Wind)} + f_{n-1}^{(Back)} + f_{n-1}^{(Damp)} \\ F_{n-1} &= F_{n-2} + f_{n-1} \\ S_n &= S_{n-1} + C_2 \times F_{n-1} \end{aligned} \tag{11}$$

式中,S_n 为角位移,$f_{n-1}^{(Wind)} = Force$(如图 4)。其次,求得了每一层次子树枝相对于其根节点的角位移后,通过对坐标轴的旋转,得到相应的相对转换矩阵,该方法称为 HMSM,可以表示为

$$\boldsymbol{M}_{total}i = (\boldsymbol{M}_{total}i - 1) \times (\boldsymbol{M}_i i - 1)^{-1} \times (\boldsymbol{M}_{self}i)^{-1}$$

式中 $\boldsymbol{M}_{total}i$ 为第 i 级相对于第 0 级的转换矩阵。最后,根据运动合成原理对所有的树枝进行,形成一棵完整的树。随着时间的推移,树木受到的外力 $Force$ 在不断地变化,致使每个子树枝的角位移不断变化,最终合成的树木模型就是一个随时间而变化的序列,即动画。

3 基于 CUDA 的树木运动并行仿真算法

对树木运动仿真的并行算法的关键在于提取可以树木运动过程中的可并行计算的过程,另外对于

不可并行的算法模块要通过多数据单元来做并行处理,从而极大程度地提高程序的并行度。

根据树木运动的算法可知:每个节点的受力 f 求解互不影响,是相互独立的;每个子树枝进行欧拉求解其自转换矩阵 $\boldsymbol{M}_{self}i$ 也是相互独立的,它们都适合并行处理;而在累积所有节点的世界坐标 W 的步骤中,第 i 级子树枝的各节点的位置 W^i 依赖于第 $i-1$ 级子树枝的根节点位置 W_{root}^{i-1},所以子树枝间的世界坐标 W 的累积过程是一个串行的过程,不适合并行处理,对此,通过增加临时存储空间来使计算并行化,提高运行仿真的效率。

在 CPU 执行中,树木模型是以 C++ 语言的对象形式进行存储的,包括结点、树枝段骨架、树枝等属性,以及相应的操作,因而相同的数据在内存中的存储与操作都是不连续的;同时,CPU 中计算主要是以串行为主,执行过程以大量分枝结构进行控制,因而此数据结构不合适在 GPU 上并行执行。所以,如果想要在 GPU 上并行执行动画过程,首先需要重新设计所有数据,使之适合在 GPU 上并行执行。根据树木的模型的特点、运动原理,以及 CUDA 相适应的数据结构,对所有数据采用 1 维数组的方式进行存储。以图 2 为例,按照层次由低到高的顺序存储树枝骨架节点的局部坐标为 *LocalPosition*,同样,我们将结点的世界坐标存为 *WorldPosition*,自转换矩阵 *ConvertMatrix_self*,总的转换矩阵 *ConvertMatrix_total* 等。

定义了适合 CUDA 并行执行的树木数据结构以后,便可以进行树木并行动画仿真算法(算法 1)。

算法 1

1)计算每个子树枝在局部坐标系下的外力;

2)计算每个子树枝在局部坐标系下的自转换矩阵;

3)计算每个子树枝在世界坐标系下总的转换矩阵;

4)计算每个子树枝节点世界坐标系下的相对坐标;

5)计算每个子树枝的根节点的世界坐标;

6)计算每个子树枝的节点的世界坐标;

7)计算并调整树木轮廓;

8)重复 1)—7),直到所有树枝计算完成。

根据并行算法(算法 1),以线程为单位,每个线程都执行相同的代码,通过索引表对不同存储空间的树木模型 1 维线性数组数据结构进行索引和并行计算,完善树木的运动仿真。

以图 2 动画为例,首先执行算法 1 中的第 1)—3)步,并行计算每个子树枝的受力情况。如图 5 所示,thd_i 表示第 i 号线程,$\boldsymbol{M}_{inverse}i$ 表示 $Level_i$ 层次树枝的逆转换矩阵,由初始化得到。图 5 每个线程 thd_i 首先根据式(8)—(11)计算外力作用下的局部角位移 $\boldsymbol{S}_i$,然后计算对应的自转化矩阵 $\boldsymbol{M}_{self}i$,进而以 $\boldsymbol{M}_{total}i$ 为中间变量;此时将转换矩阵的积累赋予临时变量 $\boldsymbol{M}_{temp}i$ 中,然后将 $\boldsymbol{M}_{temp}i$ 中的值拷贝到 $\boldsymbol{M}_{total}i$ 中。如图 5 的计算中没有引入 $\boldsymbol{M}_{temp}i$ 变量,那么求解 $\boldsymbol{M}_{total}i$ 过程可能会出现读写冲突和写写冲突的问题,因为对 $\boldsymbol{M}_{total}i$ 的累计过程中既要读取 $\boldsymbol{M}_{total}i$ 又要写入 $\boldsymbol{M}_{total}i$,所以在图 5 具体求解的过程中引入了 *ConvertMatrix_temp* 临时变量存储中间结果,可以消除存储器的读写、写写冲突问题。所以我们看到,通过增加一定的存储空间不但解决了存储器访问冲突问题,同时将串行计算过程变成并行计算,可以充分利用 CUDA 进行加速的执行。

图 6 演示了算法 1 中步骤 4)5)的模拟过程。和上述类似,步骤 4)中的局部坐标 L 和世界坐标系下的相对坐标 W 的求解都是相对独立的,适合并行执行;而步骤 5)类似于步骤 3),对于世界坐标的累积同样会出现读写、写写等访问冲突,因而引入了 *Root Positions* 变量存储中间结果 R,进而配合图 7,将线程 thd_i 所对应的结点世界坐标 W 进行增值操作,增加对应的上一级树枝的根节点的世界坐标 R,有效完成步骤 6)的并行执行。

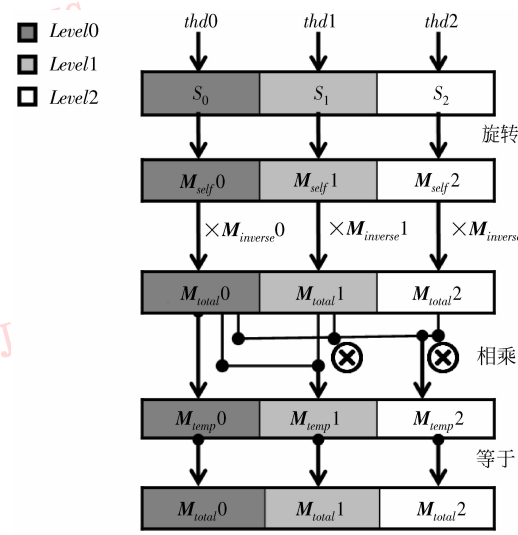


图 5　转换矩阵求解算法的并行执行

Fig. 5　Parallel implementation of transform matrix algorithm

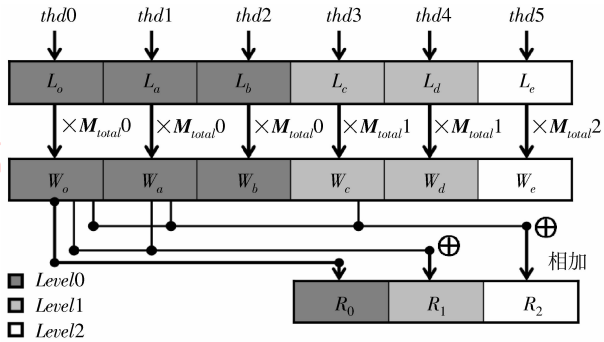


图 6 根节点世界坐标求解算法的并行执行

Fig. 6 Parallel implementation of root nodes' world positions algorithm

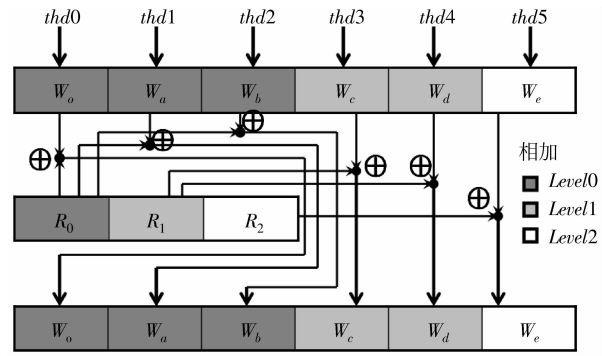


图 7 树木节点世界坐标求解算法的并行执行

Fig. 7 Parallel implementation of all nodes' world positions algorithm

图 8 演示了算法 1 中步骤 7)的模拟过程。因为每个骨架段相对应的轮廓结点的坐标只与该骨架段的坐标有关系,因而每个骨架段对应的轮廓之间的求解具有相互独立性,并且具有相同的处理过程,所以适合并行处理,同时结合方向优先层次结构的段链接索引数组,可以方便地完成轮廓结点并行求解(图 8 第 1 步);相似地,轮廓结点的修改算法也是相互独立的,同时动态的搜索相连段之间的结点的连接索引,通过线性插值的方法并行修饰轮廓相同结点的坐标求解。

到此为止,我们已经实现了树木动画仿真的 CUDA 并行算法求解,但是我们的并行仿真算法的加速能力还有提高的潜力。

树木模型的结点数目一般是 5 万～6 万,因而考虑 CUDA 的计算能力,我们取块和线程的维度均为 256。这样最多有 256×256 个线程并行执行,对于一般比较小的模型这样的取值是适用的;当模型的结点数目大于 256×256 时,并行执行的线程最大

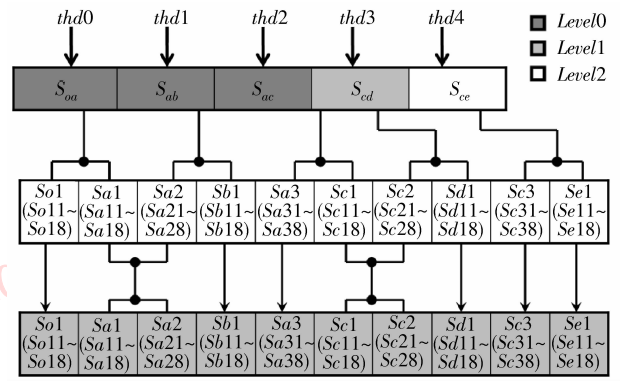


图 8 树木轮廓节点求解及其修正算法的并行执行

Fig. 8 Parallel implementation of tree contour algorithm

个数小于 256×256,因而不能提供足够的线程数进行并行计算,同时伴随着绘制图像跳变现象的出现,为此我们或者增加二者之一到 512 设置更高,或者通过程序将多出的数据应用串行的方式进行计算,所以块和线程的维度大小的选取应该综合考虑 GPU 计算能力限制和程序的灵活应用。例如以代码中某一核(记为核 1)为例,每个块选择线程数目为 256,每个线程需要寄存器(register)数目为 23,每个块利用共享存储器大小为 68 B,那么每个多处理器单元中活动的线程数目为 512,活动的偏差(warp)为 16,活动的线程块为 2,每个多处理器单元的利用率为 50%。

同时,为了提高 GPU 中数据的访问速度,我们将运动运算过程中不变的数据,比如骨架结点局部坐标数组、轮廓段索引数组、树叶的局部坐标数组等放在相对访问速度更快一些的常数存储器或纹理存储器中,同时将一些临时变量定义在共享存储器或者寄存器中,降低大量数据同时运行过程时的访问延迟,提高 GPU 数据的访问的最优化速度。要注意 CUDA 中线程的同步执行,避免出现动画的跳变现象,否则会出现意想不到的结果,同时尽量优化核程序,减少流程中不必要的分枝转移结构等控制,以充分发挥 GPU 的并行功能。

4 实验结果与分析

根据本文中的基于物理的树木动画仿真模拟系统,我们构造了多种树木在不同风力条件下的运动仿真效果。图 9 展示了树木动画仿真效果图,描述了两种不同结构树木在静止和风力作用下发生运动

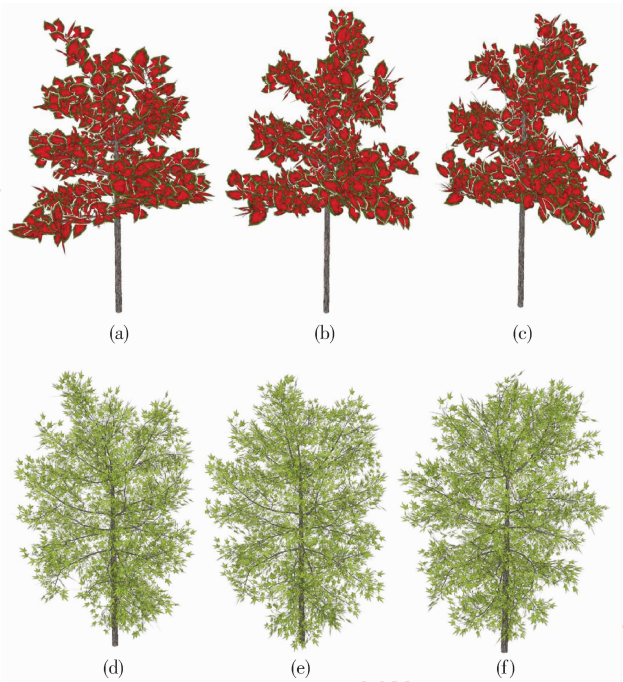


图 9　风中树木动画效果图

Fig. 9　Results of tree animation under wind

的动画仿真,其中(a)—(c)演示了某一种多树枝结构的树模型的运动过程,(b)是该树木模型在初始时刻(无外力作用时)的状态,(a)(c)分别是该模型在风里作用下向不同方向运动的仿真效果;(d)—(f)演示了分叉树枝结构的树木模型运动的仿真,同样,(e)是在初始时刻的状态,(d)(f)分别表示该模型在风里作用下向不同方向运动的仿真效果。通过图 9 我们可以看到本文提出的并行算法对于不同的树木模型是有效的。

本文所有仿真过程运行在 Intel Core(TM) 2.6 GHz CPU、nVidia GeForce 8800 GPU 以及 3 G 内存环境下。表 1 是对 5 种不同的树木模型的动画数据统计及其分析,其中"GPU"代表树木模型在 GPU 上运行的平均帧率,"CPU" 代表树木模型在 CPU 上运行的平均帧率,"顶点数"表示不同模型的顶点的个数。由表 1 可知,对于不同结构、不同数据规模的树木模型,运动仿真的加效果各不相同,GPU 运行相对于 CPU 加速比由1.5 ～8.9倍,基本表现出树木模型顶点数越多,GPU资源占用率越高、其并行

表 1　树木动画加速数据统计

Tab. 1　Statistics of accelerating data of tree animation

模型	顶点数/个	运行环境	平均帧速/(帧/s)	图像预览	加速倍数
1	78 563	GPU	9.483		8.9
		CPU	1.071		1.0
2	24 902	GPU	60.001		2.4
		CPU	25.472		1.0
3	37 823	GPU	18.927		4.5
		CPU	4.223		1.0
4	26 792	GPU	23.284		3.5
		CPU	6.700		1.0
5	15 731	GPU	65.539		1.5
		CPU	43.377		1.0

程度也越高,加速效果越明显。同时,我们可以看到本文对树木动画的物理模拟在模型比较小的情况下,已经达到了实时仿真。

5　结　论

提出一种在风力作用下基于物理的树木运动仿真的并行算法,应用 CUDA 架构的并行通用计算进行加速,达到实时树木运动加速的效果,生成的树木动画有很强的真实感。本文新算法解决了树木动画难以实时运算的瓶颈问题。

我们将进一步调整和挖掘 CUDA 代码中可以并行和调高程序效率的部分,例如用共享存储器、寄存器和程序并行性的重组等;将树木最终的绘制直接用 CUDA 来实现,这样就可以免去 GPU 向 CPU 传递数据,耗费过多的数据传输时间;实现大规模树

木的实时动画,比如一片森林的动画[19]等。今后,我们进一步探索,将本文的 CUDA 并行算法的设计思想和实现方法应用在普通植物的动画模拟中,充分利用 CUDA 的通用计算能力使植物运动的仿真更加简便快速,同时也给图形学注入一股新的力量。

参考文献(References)

[1] Liu Youquan. Study on Acceleration Techniques of Physically Based Computer Animation [D]. Beijing: ISCAS, 2005. [柳有权. 基于物理的计算机动画及其加速技术的研究[D]. 北京: 中科院软件所,2005.]

[2] Wu Enhua. State of the art and future challenge on general purpose computation by graphics processing unit [J]. Journal of Softwar, 2004,15(10): 1493-1504. [吴恩华. 图形处理器用于通用计算的技术现状及其挑战[J]. 软件学报,2004, 15(10): 1493-1504.]

[3] Wiki World. GPU Technological Development [EB/OL]. (2008-06-03)[2010-10-14]. http://www.allwiki.com/wiki/GPU%E6%8A%80%E6%9C%AF%E5%8F%91%E5%B1%95. [天下维客. GPU 技术发展 [EB/OL]. (2008-06-03)[2010-10-14]. http://www.allwiki.com/wiki/GPU%E6%8A%80%E6%9C%AF%E5%8F%91%E5%B1%95.

[4] Akagi Y, Kitajima K. Computer animation of swaying trees based on physical simulation [J]. Computers & Graphics, 2006, 30(4):529-539.

[5] Wu Enhua, Chen Yanyun, Yan Tao, et al. Reconstruction and physically-based animation of trees from static images [C]// Computer Animation and Simulation'99. Heidelberg: Springer Verlag, 1999: 157-166.

[6] Feng Jinhui, Chen Yanyun, Yan Tao, et al. Going with wind - physically-based animation of trees [J]. Journal of Softwar, 1998, 21(9): 769-773. [冯金辉,陈彦云,严涛,等. 树在风中的摇曳-基于物理的计算机动画[J]. 计算机学报,1998, 21(9): 769-773.]

[7] Ralf H, Alexander K, Michael W. Physically guided animation of trees [J]. Comput. Graph. Forum, 2009, 28(2): 523-532.

[8] Julien D, Mathieu R, Lionel B, et al. Wind projection basis for real-time animation of trees [J]. Comput. Graph. Forum, 2009, 28(2): 533-540.

[9] Ota S, Fujimoto T, Tamura M, et al. $1/f^{\beta}$ Noise-based real-time animation of trees swaying in wind fields [C]//Proceedings of Computer Graphics International 2003. Los Alamitos, CA, USA: IEEE Computer Society, 2003: 52-60.

[10] Ota S, Tamura M, Fujimoto T, et al. A hybrid method for real-time animation of trees swaying in wind fields [J]. The Visual Computer, 2004, 20(11): 613-623.

[11] Zioma R. GPU gems 3, chapter 6: Gpu-generated Procedural Wind Animations for Trees [EB/OL]. (2007-12-03)[2010-10-14]. http://http.developer.nvidia.com/GPUGems3/gpugems3_ch06.html.

[12] Diener J, Reveret L, Fiume E. Hierarchical retargeting of 2D motion fields to the animation of 3D plant models [C]// Symposium on Computer Animation, SCA 06. Vienne, Autriche: ACM-Siggraph/Eurographics, 2006: 187-195.

[13] James D L, Barbic J, Pai D K. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources [J]. ACM Transactions on Graphics (SIGGRAPH 2006), 2006, 25(3): 987-995.

[14] Geys I, Van G L. View synthesis by the parallel use of GPU and CPU [J]. Image and Vision Computing, 2007, 25(7): 1154-1164.

[15] nVidia Compony. CUDA Programming Guide: CUDA Toolkit [EB/OL]. (2010-8-23)[2010-10-14]. http://www.nvidia.com/object/cuda_get.html.

[16] Wejchert J, David H. Animation aerodynamics [J]. ACM SIGGRAPH Computer Graphics, 1991, 25(4):19-21.

[17] Shinya M, Fournier A. Stochastic motion-motion under the influence of wind [J]. Comput. Graph. Forum, 1992, 11(3): 119-128.

[18] Seller D, Fourcaud T, Lac P. A finite element model for investigating effects of aerial architecture on tree oscillations [J]. Tree Physiology, 2006, 26(6): 799-806.

[19] Thomas D G, Stéphane C, François F. An interactive forest [C]//Proceedings of the Eurographic Workshop on Computer Animation and Simulation. Heidelberg: Springer, 2001: 65-74.