文章编号: 1006-2475(2010)03-0015-04

基于多核处理器的三维场景并行化探讨

吴玮欣, 陆 达

(厦门大学计算机科学系, 福建 厦门 361005)

摘要: 提高三维场景的运行速度一直以来都是程序开发人员需要面临的一大难题, 随着面向主流应用的多核处理器的出现与普及, 利用处理器提供的多个内核而不通过编写多线程的方法来提高程序的并行性成为了一种可能。本文介绍虚拟现实开发工具 OpenGL和共享存储系统并行编程接口 OpenMP; 分析 OpenGL绘制三维场景的一般过程; 并以纹理映射为例着重探讨在 OpenGL程序中使用 OpenMP来提高程序并行性的方法。

关键词: 虚拟现实; 多核; OpenGL; OpenMP; 并行化

中图分类号: TP391.9 文献标识码: A **doi**: 10.3969/j.issn.1006-2475.2010.03.005

Research on Parallelization of Three-dimensional Scene Based on Multi-core Processors

WU Wei-xin, LU Da

(Dept. of Computer Science, Xiamen University, Xiamen 361005, China)

Abstract: Improving the speed of three-dimensional scene has always been a major challenge which programmers need to face. With the appearance and popularization of the multi-core processor, using the cores on processor instead multi-threading methods to improve parallelism of the program has become a possibility. This article introduces the development of virtual reality tools OpenGL and parallel programming interface OpenMP which is a shared-memory systems; analyzes the general process of using OpenGL to draw up the three-dimensional scene; and takes the texture-mapping as an example to discuss how to used OpenMP in the OpenGL program to enhance the parallelism.

Key words: virtual reality; multi-core; OpenGL; OpenMP; parallelization

0 引 言

在当今信息时代下, 计算机技术发展无突飞猛进, 三维空间模拟技术已逐步应用于各种行业研究领域。同时, 3D虚拟环境现实模拟及多媒体技术在个人电脑(PC)上的实现, 使得虚拟现实(Virtual Reality)这一技术得到了更普遍的应用。虚拟现实技术就是采用以计算机技术为核心的现代高科技生成逼真的视、听、触觉一体化的特定范围的虚拟环境。而许多三维空间场景往往又具有很高的复杂性, 如何在现有的计算机处理能力下更有效地提高三维空间场景的性能, 使得人们在虚拟世界中的漫游和事务处理更加流畅已经是每个程序开发人员都必须要思考的问题。

Co-Design Automation Inc.的创始人 Simon Davidmann在 2005年秋天对《EE Times》表示: “所有的芯片都将成为多处理器, 我们必须学习如何给它们编程”。可以预见, 今后数年内各个领域的计算机都将发展成为并行计算机, 包括服务器、个人电脑、手机等。充分利用好并行计算技术来提升软件性能将是一项重要的竞争优势。

在三维空间场景的绘制过程中引入并行机制可以有效地提高场景的真实感和运行速度, 目前可选择的多线程开发工具有 OpenMP、Win32[1-2]线程库以及 pThread库。Win32线程库和 pThread库虽然操作灵活但要求开发者控制线程操作的细节, 编程较为复杂, 且在多核平台上推广应用有一定的难度。OpenMP是 Intel极力推荐的多线程开发工具, 具有简单、可移植的特点, Intel C++编译器 9.1、Microsoft Visual Studio 2005等都提供了对 OpenMP的支持。

收稿日期: 2009-10-27

作者简介: 吴玮欣(1985-), 男, 福建厦门人, 厦门大学计算机科学系硕士研究生, 研究方向: 电力系统仿真技术, 虚拟现实; 陆达(1954-), 男, 吉林吉林人, 教授, 硕士生导师, 研究方向: 电力电子, 电力系统仿真。

1 OpenGL及其工作流程

OpenGL[3]即开放性图形库(Open Graphic Library),是由SGI公司为其图形工作站开发的IRIS演变而来的,现在它已成为三维图形的标准。OpenGL作为一种图形硬件的软件接口,具有完全开放的图形开发环境。它提供了相应的图形变换函数、光源处理函数、纹理映射函数及特殊效果处理函数等,很容易实现模型的各种变换、着色、光照、纹理、交互和动画,因此在虚拟现实、创建三维逼真场景方面,它是其它软件无法比拟的。

1.1 OpenGL图形操作步骤

作为图形硬件的软件接口,OpenGL最主要的工作就是将二维及三维物体绘制到帧缓存。这些物体由一系列的描述物体几何性质的顶点或描述图像的像素组成。OpenGL执行一系列操作把这些数据最终转化成像素数据并在帧缓存中形成最后的结构。

在微机上运用OpenGL进行主要图形操作,并最终在显示器上绘制三维场景的基本步骤是:

(1)设置像素格式。设置OpenGL的绘制风格、颜色模式、颜色位数、深度位数等重要信息。

(2)建立模型。建立所需的虚拟环境中的模型。

(3)舞台布置。将模型放置于虚拟场景中的适当位置。

(4)效果处理。设置物体的材质(颜色、光学性能和纹理等),加入光照等条件。

(5)光栅化。将景物及其他颜色信息转化为可在计算机屏幕上显示的像素信息。

1.2 OpenGL图形绘制过程

OpenGL图形绘制过程如图1所示。

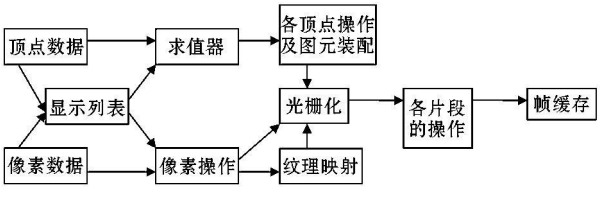

图1 OpenGL绘制原理

2 OpenMP——多核下的并行工具

2.1 OpenMP简介

OpenMP[4]是一款为共享内存计算机而编写并行应用程序的行业标准API,具有简单、移植性好和可扩展等优点。OpenMP的主要目的是使高性能计算中常见的循环导向型程序更加易于编写。OpenMP并不是一门新的语言,它是对基本语言(如Fortran77、Fortran90、C、C++等)的扩展。OpenMP规范中定义的制导指令、运行库和环境变量,能够使用户在保证程序的可移植性的前提下,按照标准将已有的串行程序逐步并行化。制导指令是对程序设计语言的扩展,进一步提供了对并行区域、工作共享、同步构造的支持,并且支持数据的共享和私有化。这样,用户对串行程序添加制导指令的过程,就类似于进行显式并行程序设计。运行库和环境变量,使得用户可以调整并行程序的执行环境。目前已经有许多硬件和软件供应商提供支持OpenMP的编译器,如DEC、Intel、IBM、HP、Sun、SGI及U.S DOE ASCI program等,并且包括了Unix和NT两种操作系统平台[5]。

2.2 OpenMP并行编程模型

OpenMP基于fork-join[6-7]的编程模式而设计。OpenMP程序起初以一条单线程的形式开始运行。如果编程人员希望在程序中利用并行,那么就需将额外的线程进行分支,以创建线程组。这些线程在称为"并行区域"的代码区域内并行执行。在并行区域末尾,将等待所有线程全部完成工作,并将其重新结合在一起。那时,最初线程或"主"线程将继续执行,直至遇到下一个并行区域(或程序结束)[8],其过程如图2所示。

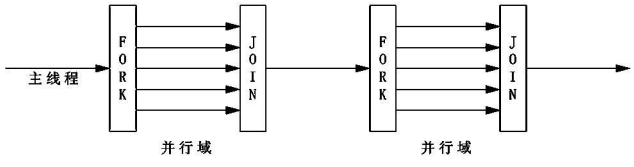

图2 OpenMP并行模型

2.3 OpenMP指令

为一个应用程序增加OpenMP并行能力只需要增加几个编译器指令或者在需要的地方调用OpenMP函数[9-11]。这些编译器指令的格式如下:#pragma omp directive-name [clause[[,] clause]...] new-line,其中"directive-name"为OpenMP制导指令,规定了编程人员希望执行的并行动作,而"clauses"(子句)则对该动作进行修改,或对线程所见的数据环境进行控制,制导指令包含如下几种:

(1)并行域结构:parallel;

(2)共享任务结构:for、sections、single;

(3)组合并行共享任务结构:parallel for、parallel sections;

(4)同步结构:master、critical、barrier、atomic、flush和ordered;

对于制导指令而言clause子句是可选的,但子句可以影响到指令的行为。每一个指令有一系列适合

它的子句, 但有 5个指令 (master, critical, flush, ordered和 atomic)不能使用子句[4]。

3 纹理映射

纹理映射又称纹理贴图[12], 广泛地应用于描述具有真实感的物体。采用纹理映射的方法可以大大地简化建模的过程。比如绘制一面国旗, 就可以用一幅真实的国旗图像或照片作为纹理贴到一个矩形上, 这样, 一面逼真的国旗就画好了。如果不用纹理映射的方法, 则国旗上的每个图案都必须作为一个独立的多边形来画。相比之下, 使用纹理映射方法的工作量要小得多。

3.1 OpenGL中纹理映射的一般过程

如图 3所示, 在 OpenGL中使用纹理映射的一般过程是:

(1)从外存中读入图像文件, 载入贴图。

(2)创建纹理, 包括创建纹理存储空间、创建纹理对象和生成纹理。

(3)设置纹理的滤波和映射方式。

(4)设置纹理坐标和物体几何坐标, 使用纹理绘制场景。

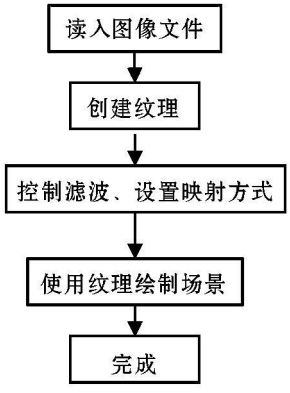

图 3　OpenGL中纹理映射的一般过程

3.2 纹理映射实现

在 Microsoft Visual Studio 2005环境中, 将一块二维纹理映射到矩形平面的具体过程如下:

首先, 读入图像文件:

```
HBITMAP hBMP; //位图句柄
BITMAP BMP; //位图变量
hBMP= (HBITMAP) LoadImage(GetModuleHandle(NULL), MAKEINTRESOURCE(bmp_id), IMAGE_BITMAP, 0,0, LR_CREATEDIBSECTION); //加载编号为 bmp_id的位图文件
```

然后, 创建纹理:

```
if(hBMP) { //如果文件存在?
GetObject(hBMP, sizeof(BMP), &BMP); //获取位图对象
GLuint texture[1]; //创建一个纹理的存储空间
GLGenTextures(1, texture); //创建纹理
GLBindTexture(GL_TEXTURE_2D, texture); //使用位图数据生成纹理
GLUBuild2DMipmaps(GL_TEXTURE_2D, 3, BMP.bmWidth, BMP.bmHeight, GL_BGR_EXT, GL_UNSIGNED_BYTE, BMP.bmBits);
DeleteObject(hBMP); } //删除位图对象
```

接着, 设置滤波和映射方式:

```
GLTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR); //控制滤波
GLTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
GLTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL); //说明映射方式
```

最后, 使用纹理绘制场景:

```
GLBindTexture(GL_TEXTURE_2D, texture[0]); //选择纹理
GLBegin(GL_QUADS); //绘制四边形
GLTexCoord2f(0.0,0.0); GLVertex3f(-2.0,-1.0,0.0); //设置纹理坐标和物体几何坐标
GLTexCoord2f(0.0,1.0); GLVertex3f(-2.0,1.0,0.0);
GLTexCoord2f(1.0,1.0); GLVertex3f(0.0,1.0,0.0);
GLTexCoord2f(1.0,0.0); GLVertex3f(0.0,-1.0,0.0);
GLEnd();
```

3.3 纹理映射的并行化处理

在绘制具有真实感的大规模三维场景中, 需要用到大量的图像文件, 特别是在每次运行场景和进行场景变换时, 由于需要进行大量的纹理映射来绘制场景, 往往给人一种画面的停滞、运行不够流畅的感觉。

实现一个完整的纹理映射过程可以分为: 读入图像文件, 创建纹理, 控制滤波、设置映射方式, 使用纹理绘制场景 4个部分。分析这 4个部分可以发现, 最后一个过程使用纹理绘制场景相对独立于前面 3个过程。整个纹理映射中最耗费时间的是第一个过程: 读取图像文件。

经过以上分析, 在绘制三维场景时, 如果每次生成新的场景都需进行大量的纹理映射, 为了提高纹理映射的速度, 可以引入 OpenMP对纹理映射过程进行并行化处理。另一方面, 由于 OpenMP的并行化主要是针对循环级的细粒度并行化[13], 所以需要对纹理映射过程进行调整, 以最大限度地发挥多核的优势, 调整过程如下:

(1)从功能上对纹理映射的 4个部分进行重新划分, 关系较为密切的: 读入图像文件, 创建纹理, 控制滤波、设置映射方式 3个部分合并在一起; 使用纹理绘制场景部分单独独立。

(2)将合并完的 3个部分封装成一个函数: Build

_bmpTexture( ) (创建纹理)。

(3)由于每一次纹理映射过程都是相互独立的,不存在数据相关性,所以,采用这样一种并行方法,即:每次需要进行纹理映射时,先通过 OpenMP的组合并行共享任务结构 parallel for一次性创建完所有纹理,当需要绘制具体场景时再调用所需要的纹理。

3.4 纹理映射的并行化实现

在所有工作开始之前,需要在 Microsoft Visual Studio 2005中开启对 OpenMP的支持:在项目属性对话框"配置属性"中的"C/C++"语言页里,将 OpenMP支持选项改为"是/(OpenMP)"就能够支持 OpenMP[14]。

首先,通过 Build_bmpTexture( )函数创建纹理,Build_bmpTexture( )函数接受两个参数:加载位图的编号和纹理的存储空间,最后在内存中生成纹理,函数实现如下:

```
Void Build_bmpTexture(USHORT bmp_id, gluint& texture)
{ //(位图编号,纹理存储空间)
    HBITMAP hBMP; //位图句柄
    BITMAP BMP; //位图变量
    hBMP= (HBITMAP) LoadImage(GetModuleHandle(NULL),
MAKEINTRESOURCE(bmp_id), IMAGE_BITMAP, 0,0, LR_CRE-
ATEDIBSECTION); //加载编号为 bMP_id的位图文件
    if(hBMP) { //如果文件存在?
    GetObject(hBMP, sizeof(BMP), &BMP); //获取位图对象
    GLBindTexture(GL_TEXTURE_2D, texture[ i]); //使用位
图数据生成纹理
    GLuBuild2DMipmaps(GL_TEXTURE_2D, 3, BMP.bm-
Width, BMP.bmHeight, GL_BGR_EXT, GL_UNSIGNED_BYTE,
BMP.bmBits);
    DeleteObject(hBMP); //删除位图对象 }
    GLTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_
FILTER, GL_LINEAR); //控制滤波
    GLTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG
_FILTER, GL_LINEAR);
    GLTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_
MODE, GL_DECAL); //说明映射方式 }
```

然后,使用 parallel for,将一个 for循环分配到多个线程中执行对所有需要创建的纹理进行并行化:

```
# ifdef_OpenMP
# pragma omp parallel for
# end if
 for( int i= 0; i< 100; i++ )
 Build_bmpTexture( ,i texture[ i]);
```

在一次创建完所有纹理后,再根据需要,具体地使用纹理进行场景绘制,其过程同 3.2。

4 结束语

在三维场景中引入并行机制可以有效地提高虚拟场景的运行速度,使得人们在虚拟世界中的漫游和事务处理更加的流畅。随着多核技术的蓬勃发展,在未来的几年,多核处理器势必将取代单核处理器成为应用的主流。本文着重对如何有效地利用多核处理器的硬件优势充分挖掘三维虚拟空间的并行潜力这一问题进行了探讨,将纹理映射与 OpenMP结合进而提高了 OpenGL纹理映射的速度,展现了 OpenMP在三维虚拟空间并行化应用中的广阔前景。

参考文献:

[1] [美] Jeffrey Richter. Windows核心编程(第5版)[M]. 黄陇,李虎译. 北京:清华大学出版社,2008.

[2] Jim Beveridge, Robert Wiener. Win32多线程程序设计[M]. 侯捷译. 武汉:华中科技大学出版社,2002.

[3] Dava Shreiner OpenGL编程指南(第6版)[M]. 徐波译. 北京:机械工业出版社,2008.

[4] OpenMP Architecture Review Board OpenMP应用编程接口2.5版[EB/OL]. http://www.openMP.org/,2009-07-12.

[5] 周伟明. 并行计算_OpenMP编程指南[EB/OL]. http://blog.csdn.net/drzhouweiming/archive/2008/05/23/2472454.aspx,2008-05-23.

[6] Shameem Akhter, Jason Roberts. 多核程序设计技术[M]. 李宝峰,等译. 北京:电子工业出版社,2007.

[7] Barry Wilkinson, Michael Allen. 并行程序设计(第2版)[M]. 陆鑫达,等译. 北京:机械工业出版社,2005.

[8] Tim Mattson. 编写并行程序:多语言指南介绍[EB/OL]. http://software.intel.com/zh-cn/articles/writing-parallel-programs-a-multi-language-tutorial-introduction/,2008-01-09.

[9] Quinn M J. Parallel Programming in C with MPI and OpenMP[M]. 北京:清华大学出版社,2005.

[10] Timothy G Mattson, Beverly A Sanders, Berna L Massingill. 并行编程模式[M]. 敖富江译. 北京:清华大学出版社,2005.

[11] Kang Su Gatlin, Pete Isensee. Reap the Benefits of Multithreading without All the Work[EB/OL]. http://msdn.microsoft.com/en-us/magazine/cc163717.aspx,2005-10-30.

[12] 巴斯. 3D计算机图形学(OpenGL版)[M]. 北京:清华大学出版社,2006.

[13] 郑锋,李名世,蔡佳佳. 基于 OpenMP的并行遗传算法探讨[J]. 心智与计算,2007(4):396-402.

[14] [美] Andrew Parsons, Nick Randolph. Visual Studio 2005高级编程[M]. 吴雷译. 北京:清华大学出版社,2008.