基于图形硬件的辐照度环境纹理图的计算方法

艾祖亮[1)] 彭 耿[1)] 张立民[1,2)]

[1)](海军航空工程学院电子与信息工程系,烟台,264001) [2)](天津大学电子与信息工程学院,天津,300072)

摘 要 辐照度环境纹理图是绘制任意光照环境下漫反射物体表面的一种有效的方法。为了实现动态光照环境下辐照度环境纹理图的实时绘制,基于当前的通用图形硬件,提出了一种采用顶点着色器快速计算辐照度环境纹理图的方法。该方法从分析球面调和函数入手,首先得出环境贴图的二次多项式表达形式;然后用顶点着色程序对多项式系数和球面调和函数进行加速计算,以快速生成辐照度环境纹理图;最后对于动态光照环境,则通过对环境纹理图的分级细化来加快光照系数的计算,进而实现了动态光照环境下辐照度环境纹理图的重新绘制。实验表明,在动态光照条件下,采用该方法在获取真实感光照效果的同时,其运算速度也能满足交互系统的需求。

关键词 辐照度 环境纹理图 球面调和函数 顶点着色 分级细化纹理

中图法分类号:TP391.41 **文献标识码**:A **文章编号**:1006-8961(2008)05-0972-05

A Technique to Evaluate Irradiance Environment Maps Using Graphics Hardware

AI Zu-liang[1)], PENG Geng[1)], ZHANG Li-min[1,2)]

[1)](*Department of Electronics and Information Engineering, Naval Aeronautical Engineering Institute, Yantai* 264001)

[2)](*School of Electronic and Information Engineering, Tianjin University, Tianjin* 300072)

Abstract Irradiance environmental maps are an effective technique to render diffuse objects in arbitrary lighting environments. Based on current computer graphics hardware a technique is proposed for calculating irradiance environmental maps using vertex shader. The approach starts from an analysis of spherical harmonics and finds out a quadratic polynomial form of environment mapping. Using vertex shader instructions the polynomial coefficients and spherical harmonics functions can be fast calculated, and then the corresponding irradiance environment maps are also generated. When the lighting environment changes, the lighting coefficients are fast computed by generating a mipmap of the source environment map, so that real-time recalculating irradiance environment is obtained. The experimental result shows it not only renders realistic lighting but also satisfies the interactive requirements in dynamic lighting environments.

Keywords irradiance, environmental maps, spherical harmonics, vertex shader, mipmap

1 引 言

真实感图形的实时绘制是虚拟现实技术和计算机游戏软件的一个重要组成部分,并支配、影响着画面的最终视觉效果。光照模型是生成真实感图形的基础,它是根据几何光学原理计算出的由场景中的物体表面上任一点投向观察者视点中的亮度大小和色彩的组成[1]。随着现代图形硬件技术的不断进步,传统的固定功能绘制流水线(fixed function rendering pipeline)正在向可编程绘制流水线(programmable rendering pipeline)转变,在新一代 GeForce 和 ATI 显卡中均提供了以 GPU 为核心的可编程顶点着色器(vertex shader)和像素着色器(pixel

基金项目:中国博士后科学基金资助项目(2005038469)

收稿日期:2006-05-29;**改回日期**:2006-10-13

第一作者简介:艾祖亮(1980~),男。2007 年获海军航空工程学院硕士学位,现为天津大学博士研究生。主要研究方向为真实感图形、纹理映射和实时绘制技术。E-mail: aizuliang@sina.com

shader),它们为实现复杂光照模型下的实时绘制提供了硬件加速支持。

环境贴图技术[2~4]的一个重要应用就是生成物体表面复杂的光照效果,它通过将光照效果表示在纹理中来实现复杂的光照模型。辐照度环境纹理图(irradiance environment maps)是一种有效的绘制漫反射物体表面的方法,而辐射照度贴图的优点则是从管线中去掉了每一个顶点的光线计算,其不仅可以使用任意多的光源,而且漫反射和环境也比较真实。通常,创建一幅辐照度环境纹理图需要很高的代价,且生成的每个纹理元素都是通过原始纹理图中近半数的纹理元素计算得到的,Ramamoorthi 和 Hanrahan 基于球面调和函数提出了环境纹理图的表示方法[5],它可以快速地将环境贴图转化为辐照度函数,其虽然大大提高了算法的执行效率,然而由于其绘制算法往往要执行复杂的矩阵和与向量运算,因此对于没有优化矩阵和与向量操作的通用图形硬件来说,并不能有效地执行,并且由于其算法需要耗时的预滤波计算,因此也不能实时处理动态的光照环境。针对上述问题,本文基于图形硬件,提出一种采用顶点着色器来实现辐照度环境纹理图快速计算的方法,并对环境纹理图的快速预滤波计算进行了研究。实验表明,该方法在动态光照条件下,既有较好的真实感光照效果,又保证了场景渲染实时性。

2　图形硬件可编程机制

以 GeForceFX 为代表的现代图形显示硬件集成了以 GPU 为处理核心的可编程图形处理系统,其可编程性主要体现在以下两个方面:一是以向量机为基础的可编程顶点着色器,它通过用户编制的面向向量运算的顶点处理程序(vertex program),实现了顶点由物空间(model space)到屏幕空间(screen space)的转换;二是以纹理并行处理为基础的 4 级层次化的纹理绘制机制和 8 级层次化的寄存器、混合器处理机制,即可编程像素着色器。

本文采用顶点着色器来对辐照度纹理图的计算进行加速,图 1 给出了 DirectX 8.1 环境下的第 1 代顶点着色器体系结构框图[6]。

图形显示硬件为顶点着色器提供了 128 个向量运算单元,并将顶点着色器作为 SIMD(single instruction multiple data)并行处理单元来运行,顶点着色器可以使用 4 种类型的存储器,它们分别是每个顶点的输入数据寄存器、输出数据寄存器、临时数据寄存器、常量内存。每个内存位置保存的是一个 4 元素向量,其中每一个元素都是一个有符号的 32bits,浮点型数据,向量通常指位置信息、法线、矩阵行、颜色或者纹理坐标。

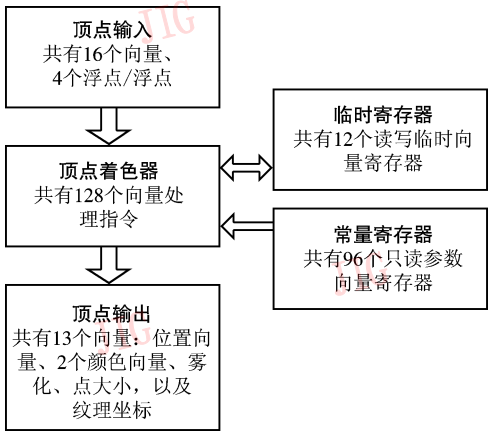


图 1　顶点着色器体系结构

Fig. 1　Vertex shader architecture

3　辐照度环境纹理图的计算

3.1　辐照度环境纹理图的表示

辐照度环境纹理图体现了周围环境的辐射率,即对于物体表面给定位置 $\boldsymbol{x}$,辐照度环境纹理图保存了位置点 $\boldsymbol{x}$ 向所有视点方向的漫反射出射辐射率,其表达式为

$$L_{\text{diffuse}}(\boldsymbol{x};\boldsymbol{n}) = \frac{k_{\text{d}}}{\pi}\int_{\Omega} L_{\text{in}}(\boldsymbol{x};\boldsymbol{l})(\boldsymbol{n}\cdot\boldsymbol{l})\,\text{d}\boldsymbol{l} \tag{1}$$

其中,$k_{\text{d}}\in[0,1]$ 表示漫反射系数,L_{in} 为周围环境的入射辐射率,相当于初始的环境纹理图。$\boldsymbol{l}$ 为光线方向,$\boldsymbol{n}$ 为物体表面法向量。由于采用直接积分计算来创建漫反射纹理图的方法需要比较昂贵的计算开销,因此 Ramamoorthi 和 Hanrahan 基于球面调和函数提出了环境纹理图的表示方法,它可以快速地将环境贴图转化为辐射照度函数。

球面调和函数[7]是应用在球面上的一种正交函数基,记为 $Y_l^m(\theta,\phi)$,它是拉普拉斯方程在球面坐标上的解,其中 θ 表示纬度坐标,$\theta\in(0,\pi)$,ϕ 表示经度坐标,$\phi\in(0,2\pi)$。完整的球面调和函数表达式如下:

$$Y_l^m(\theta,\varphi) = K_l^m \text{e}^{\text{i}m\varphi} P_l^{|m|}(\cos\theta),\ l\in\boldsymbol{N},\ -1\leqslant m\leqslant l \tag{2}$$

其中,$P_l^{|m|}$ 为相关的勒让德多项式,K_l^m 是归一化系

数,其表达式如下:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{3}$$

由于由式(2)定义的球面调和基函数比较复杂,因此为了方便表示光照,实际使用的基函数为

$$y_l^m = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi) P_l^m(\cos\theta), m>0 \\ \sqrt{2}K_l^m \sin(-m\varphi) P_l^{-m}(\cos\theta), m<0 \\ K_l^0 P_l^0(\cos\theta), m=0 \end{cases} \tag{4}$$

l 表示频率波段指数,其低的值表示低频球面调和基函数,实验证明,由于光照环境的高频项对环境贴图贡献很小,当 $l\leqslant 2$,即用前 9 项来近似表示环境纹理图时,平均误差大约为 1%,因此只需计算前 9 项球面调和基函数即可:

$$\begin{cases} y_0^0 = \dfrac{1}{\sqrt{4\pi}} \\ (y_1^1; y_1^0; y_1^{-1}) = \sqrt{\dfrac{3}{4\pi}}(x;z;y) \\ (y_2^1; y_2^{-1}; y_2^{-2}) = \sqrt{\dfrac{15}{4\pi}}(xz;yz;xy) \\ y_2^0 = \sqrt{\dfrac{5}{16\pi}}(3z^2-1) \\ y_2^2 = \sqrt{\dfrac{15}{16\pi}}(x^2-y^2) \end{cases} \tag{5}$$

用上述球面调和基函数表示的辐照度环境纹理图如下式所示:

$$\begin{cases} L_{\text{diffuse}}(\boldsymbol{n}) = \sum_{l,m} A_l L_l^m y_l^m(\boldsymbol{n}), l\leqslant 2 \\ \boldsymbol{n} = (x,y,z,1) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta, 1) \\ L_l^m = \int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} L_{\text{in}}(\theta,\varphi) y_l^m(\theta,\varphi)\sin\theta \mathrm{d}\theta\mathrm{d}\varphi \end{cases} \tag{6}$$

其中,$A_0=\pi$,$A_1=2\pi/3$,$A_2=\pi/4$[8],上式表明,环境贴图最终可通过球面调和函数用 9 个光照系数 L_l^m 表示为简单的多项式来进行计算。

3.2 计算方法

式(6)为辐照度环境纹理图的计算公式,为了提高算法的执行效率,计算的方法有以下两种:一是执行特定的图形 API 专为矩阵与向量运算优化,以加速纹理图的计算,如 RenderMan 图形渲染系统[8]及美国斯坦福大学的实时着色系统[9]等;二是直接分解为多项式形式进行计算的方法,其主要是针对没有矩阵与向量操作优化的通用图形系统。目前直接分解计算的方法大多是依靠计算机 CPU(中央处理器),随着图形硬件的发展,也可利用 GPU(可编程图形硬件)来加速计算,以便将 CPU 解放出来,也可用来从事其他应用方面的计算工作(例如碰撞检测、物理模拟等),并且硬件着色要比软件着色速度快得多。本文基于当前通用图形硬件,采用顶点着色器提出了一种更为有效的直接计算方法,并通过对初始环境纹理图分级细化来加快光照系数 L_l^m 的计算,进而实现了动态光照的环境的实时模拟。该计算方法包括以下 3 个主要步骤:

(1)获取环境纹理图 L_{in}。本文之所以选择立方体纹理图作为环境纹理图的表示方法,主要是因为立方体纹理图相对于球面、抛物面纹理图来说,不仅具有最高的质量,而且环境纹理图的获取也相当简单,因此对于静态的光照环境可以直接加载预计算好的立方体环境纹理图,而对于动态光照环境则需要在画面之间的空闲时间重新捕捉得到。

(2)计算光照环境系数 L_l^m。对于辐射照度纹理图来说,光照环境的变化意味着每一帧都需要重新计算光照系数,如果环境纹理图分辨率较高,则预计算这 9 个系数将耗费很大的时间开销,这将不能满足交互式应用实时性的要求。为了解决这个问题,在保证真实感的情况下,本文提出了一种对环境纹理图进行分级细化的方法,即首先对球面调和函数进行采样,并用 2×2 窗口滤波器产生分级细化纹理图(mipmap)[10],这个过程只需开始执行一次,然后对初始的环境纹理图进行分级细化即可产生纹理图的 mipmap,最后通过对上述两种纹理图元素进行乘积求和操作来得到最终光照系数的值。为了加快计算速度,并不需要对原始纹理进行操作,而是选择 mipmap 的低层纹理图进行计算,以一幅 256×256 分辨率大小的环境纹理图为例,直接计算生成一幅辐照度纹理图的时间复杂度为 $O(9\cdot 256\cdot 256)$,而采用 mipmap 一般选择第 3 层(64×64)纹理图就可以获得很好的视觉效果,其计算的时间复杂度仅为 $O(9\cdot 64\cdot 64)$,显然本文的方法在速度上有明显的优势。上述方法的计算过程如图 2 所示。

该方法执行的过程也相当简单,由于图形硬件已经支持分级细化纹理,因此计算的速度非常快,只需要选择相应的纹理层绘制即可。

(3)计算辐照度环境纹理图 $L_{\text{diffuse}}(\boldsymbol{n})$。为了方便计算,对环境纹理图的每个颜色通道分别进行处

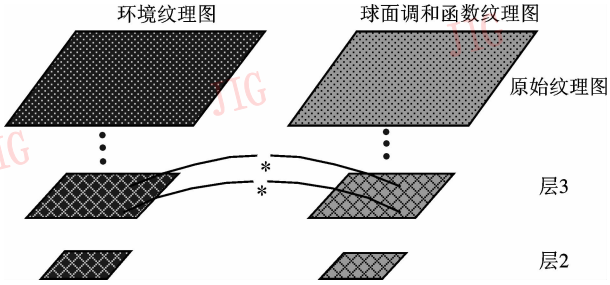


图 2　光照系数计算过程

Fig. 2　Calculation of lighting coefficients

理,上一步骤计算的光照系数 L_l^m 为 R_l^m、G_l^m、B_l^m,它们分别表示光照环境的红色、绿色、蓝色通道光照系数。现以红色通道为例,令

$$n_0=\frac{1}{2\sqrt{\pi}},n_1=\frac{\sqrt{3}}{2\sqrt{\pi}},n_2=\frac{\sqrt{15}}{2\sqrt{\pi}},n_3=\frac{\sqrt{5}}{4\sqrt{\pi}},n_4=\frac{\sqrt{15}}{4\sqrt{\pi}}$$

$$c_0=A_0n_0,c_1=A_1n_1,c_2=A_2n_2,c_3=A_2n_3,c_4=A_2n_4$$

其中,A_0、A_1、A_2 为式(6)中的常量,而对于红色通道,则 $L_{\text{diffuse}}(\boldsymbol{n})$ 可表示为

$$\begin{aligned}L_{\text{diffuse}}(\boldsymbol{n})=&c_0R_0^0-c_3R_2^0+c_1R_1^{-1}y+c_1R_1^0z+\\&c_1R_1^1x+c_2R_2^{-2}xy+c_2R_2^{-1}yz+\\&3c_3R_2^0z^2+c_2R_2^1xz+c_4R_2^2(x^2-y^2)\end{aligned}\tag{7}$$

从上式可以看出,计算一个颜色通道需要计算 9 个二次多项式系数和 9 个球面调和函数,为了加快辐照度环境纹理图的计算,本文采用顶点着色器来进行加速,即首先用 7 个常量寄存器(*cAr*、*cAg*、*cAb*、*cBr*、*cBg*、*cBb*、*cC*)来存储 3 个颜色通道多项式的系数,由于每个常量寄存器保存的是一个 4 元素向量(x,y,z,w),其中每一个元素都是一个有符号的 32bits 浮点型数据,因此可以保存 28 个常量系数,当光照环境变化导致光照系数 L_l^m 改变时,则需要重新计算改写这 7 个常量寄存器的值,常量寄存器定义如表 1 所示。

表 1　常量寄存器定义

Tab. 1　Definition of constant registers

寄存器	*cAr*	*cAg*	*cAb*	*cBr*	*cBg*	*cBb*	*cC*
x	$c_1R_1^1$	$c_1G_1^1$	$c_1B_1^1$	$c_2R_2^{-2}$	$c_2G_2^{-2}$	$c_2B_2^{-2}$	$c_4R_2^2$
y	$c_1R_1^{-1}$	$c_1G_1^{-1}$	$c_1B_1^{-1}$	$c_2R_2^{-1}$	$c_2G_2^{-1}$	$c_2B_2^{-1}$	$c_4G_2^2$
z	$c_1R_1^0$	$c_1G_1^0$	$c_1B_1^0$	$3c_3R_2^0$	$3c_3G_2^0$	$3c_3B_2^0$	$c_4B_2^2$
w	$c_0R_0^0-c_3R_2^0$	$c_0G_0^0-c_3G_2^0$	$c_0B_0^0-c_3B_2^0$	$c_2R_2^1$	$c_2G_2^1$	$c_2B_2^1$	无定义

然后通过顶点着色器程序计算辐照度环境纹理图,r_0、r_1、r_2 为临时寄存器,r_0 初始为已经归一化的物体表面法向向量 $\boldsymbol{n}$。下面以 DirectX 顶点着色器 1.1 版本来详述其着色过程:

(1)计算式(7)前 4 项;

dp4 $r_1.r,r_0,cAr$;

dp4 $r_1.g,r_0,cAg$;

dp4 $r_1.b,r_0,cAb$;

(2)计算下 4 项;

mul $r_2,r_0.xyzz,r_0.yzzx$;

dp4 $r_3.r,r_2,cBr$;

dp4 $r_3.g,r_2,cBg$;

dp4 $r_3.b,r_2,cBb$;

(3)计算最后一项;

mul $r_0.xy,r_0.xy,r_0.xy$;

add $r_0.x,r_0.x,-r_0.y$;

mad $r_1.rgb,cC.rgb,r_0.x,r_3.rgb$;

add $r_0,r_1.rgb,r_2.rgb$;

执行完毕后的 r_0 为根据该法向所计算得到的辐照度环境纹理图的纹理元素 RGB 值,由此可以看出,本文方法最终用 7 个常量寄存器和 11 条着色指令就完成了辐照度环境纹理图的计算。

3.3　绘制过程及结果

采用上述计算方法绘制辐照度环境纹理图将是非常简单的,下面给出其详细绘制过程:

(1)采样球面调和函数生成 mipmap(仅执行一次);

(2)获取立方体环境纹理图(cube map),并生成环境纹理图的 mipmap;

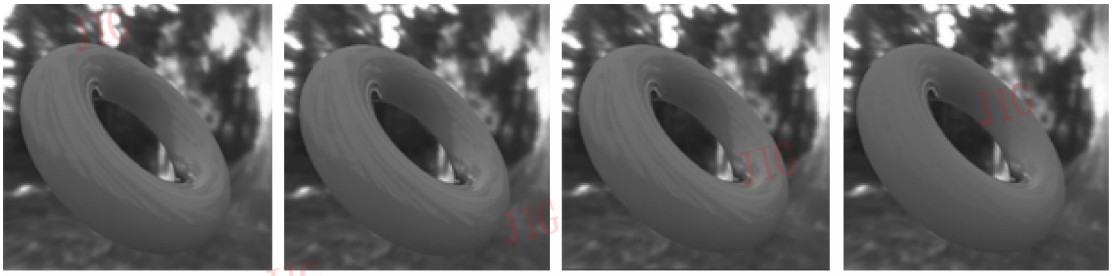
(3)选择 mipmap 纹理层,计算环境纹理图红、绿、蓝通道光照系数 R_l^m、G_l^m、B_l^m;

(4)如果光照环境变化,则改写 7 个常量寄存器;

(5)执行顶点着色程序,计算辐照度环境纹理图;

(6)用辐照度环境纹理图绘制反射物体表面。

本文所使用的实验平台是 3.0G CPU 的 PC 机，内存为 512M，Geforce 5200 显卡，显存为 128M。为了进行比较，还分别采用传统依赖于 CPU（中央处理器）计算的直接运算方法和本文基于图形硬件的计算方法来实现辐照度纹理图的绘制，光照环境纹理图大小为 256 × 256，并且光照环境是动态变化的，其绘制结果如图 3 所示。图 3（a）为用传统方法的绘制结果，其绘制速度为 21fps；图 3（b）～图 3（d）为采用本文方法的绘制结果，其绘制速度分别为 34fps、45fps、56fps，其中图 3（b）计算光照系数时采用的为原始环境纹理图，而图 3（c）、图 3（d）则分别采用环境纹理图 mipmap 的第 3 层（64 × 64）和第 2 层（16 × 16）。从绘制的结果来看，对于动态光照环境，选择环境纹理图 mipmap 的第 3 层（64 × 64）的绘制结果就可以取得很好的视觉效果，但本文的方法在速度上更具有明显的优势，这表明利用图形硬件是提高真实感图形绘制速度的有效手段。



（a）传统计算方法　（b）本文方法　（c）本文方法（层3）　（d）本文方法（层2）

图 3　辐射照度环境纹理图绘制结果

Fig. 3　Results of rendering irradiance environment maps

4　结　论

随着现代可编程图形显示硬件技术的飞速发展，以往费时的复杂光照模型已能够在硬件支持之下达到实时绘制的要求，特别是基于环境贴图计算的技术还能将复杂的光照模型计算通过纹理形式表示，这实际上为构造个性化的特殊光照效果提供了便利。本文主要集中解决在动态光照环境下如何有效地对物体进行照明的问题，基于当前图形硬件，提出了一种采用顶点着色器进行辐照度环境纹理图快速计算的方法，并通过对环境纹理图分级细化加速了光照系数的计算，取得了动态光照环境下实时的运行速度。实验结果证明，这种方法在效率和真实感上取得了令人满意的效果，可以应用于游戏、动画、视景仿真等交互式应用场合。

参考文献（References）

1　Peng Qun-sheng, Bao Hu-jun, Jin Xiao-gang. Procedural Elements for Realistic Image Synthesis[M]. Beijing: Science Press, 1999: 27～52.［彭群生，鲍虎军，金小刚. 计算机真实感图形的算法基础［M］. 北京：科学出版社，1999：27～52.］

2　Blinn J F, Newell M E. Texture and reflection in computer generated images［J］. Communications of the ACM, 1976, **19**（10）: 542～546.

3　Miller G S, Hoffman C R. Illumination and reflection maps: simulated objects in simulated and real environments［A］. In: Proceedings SIGGRAPH' 84 Advanced Computer Graphics Animation Seminar Notes［C］, New York, USA, 1984: 20～32.

4　Greene N. Environment mapping and other applications of world projections［J］. IEEE Computer Graphics & Applications, 1986, **6**（11）: 21～29.

5　Ramamoorthi R, Hanrahan P. An efficient representation for irradiance environment maps［A］. In: Proceedings SIGGRAPH' 2001［C］, Los Angeles, California, USA, 2001: 497～500.

6　Tomas Akenine-Möller, Eric Haines. Real-Time Rendering（Second Edition）［M］. Wellesley, MA, USA: A K Peters Ltd, 2002: 122～123.

7　Sloan P, Kautz J, Snyder J. Precomputed radiance transfer for real rime rendering in dynamic, low-frequency lighting environments［C］. In: Proceedings SIGGRAPH' 2002［C］, San Antonio, Texas, USA, 2002: 527～536.

8　Apodaca A A, Gritz L. Advanced RenderMan: Creating CGI for Motion Pictures［M］. San Francisco, CA, USA: Morgan Kaufmann, 2000.

9　Proudfoot K, Mark W, Tzvetkov S, *et al*. A real-time procedural shading system for programmable graphics hardware［A］. In: Proceedings SIGGRAPH' 2001［C］, Los Angeles, California, USA, 2001: 159～170.

10　Williams L. Pyramidal parametrics［J］. Computer Graphics, 1983, **7**（3）: 1～11.