基于 OpenGL 的物体反射效果模拟

姜玲燕

(南京师范大学教育科学学院,江苏 南京 210097)

[摘要] 物体反射效果的模拟可以大大增强虚拟世界的真实感,目前已广泛应用于计算机仿真系统、三维动画以及计算机游戏中. 现有的一部分模拟物体反射效果算法中存在实时性较差、真实感较低、算法较复杂等弊端. 本文基于 OpenGL 的模板技术和裁剪平面技术等,模拟了静态和动态模型的反射效果. 实验证明该方法具有实时性好、真实感较高且算法较易实现等特点.

[关键词] OpenGL,模板技术,裁剪平面,反射效果

[中图分类号]TP391.9 [文献标志码]A [文章编号]1672-1292(2013)01-0045-05

Simulating Objects Reflection Based on OpenGL

Jiang Lingyan

(School of Education Science, Nanjing Normal University, Nanjing 210097, China)

Abstract: The simulation of the effects of objects reflection can dramatically enhance reality of virtual world and related technologies have been widely used in various computer applications, such as simulation system, 3D animation, video game, and so on. At present, some simulation methods are confronted with some problems, for example, bad real-time performance, low reality in image, and complex calculation in their implementations. In this paper, we propose a new OpenGL-based method for simulating objects reflection. Some advanced programming skills are involved in our method, including template buffers and clopping plane. The experimental results demonstrate that our method has advantages of good real-time performance, high reality in image, and facility to implement.

Key words: OpenGL, template technique, clopping plane, reflection effect

反射是日常生活中随处可见的自然现象,例如,在平静的湖面上可以看到山峰的倒影,镜子可以反射出人和物的形象,高楼的玻璃幕墙可以反射出街道的景象,光亮的不锈钢器具可以反映出周围的场景,等等,正是由于这些反射效果使现实世界的景象多样化. 随着计算机图形技术的发展,人们对场景绘制的真实感要求也越来越高,计算机生成场景时仿真出逼真的反射效果可以增加绘制画面的真实感. 目前在利用计算机模拟现实场景时,往往较少考虑物体反射效果的实现,导致虚拟世界真实感不强. 因此高效地绘制出物体反射效果,对于提高数字化虚拟世界的真实感具有重要的意义.

模拟物体反射效果的最佳方式是 Turner Whitted 在 1979 提出的光线跟踪算法[1],该算法通过模拟光线的传播得到反射物体的形象,但该方法计算量巨大且费时,因此很难实现实时计算. 目前常见的绘制反射效果的方式有如下几种:对于具有金属质感的物体而言,表现其表面反射效果的直接方法是采用纹理映射[2],该类方法利用纹理贴图[3,4]来模拟表面对任意环境的反射,但该方式需要创建适当的纹理贴图,并让 OpenGL 生成纹理坐标,其反射效果很大程度上取决于纹理与纹理坐标的生成方式,且实现方法较为复杂. 另一种简单的方法是绘制反射物的镜像图,然后对反射面做透明处理,此方法最大的不足是无法有效地控制反射镜像的显示位置,使图像产生严重失真.

本文主要利用 OpenGL 的模板技术[5,6]将绘制的反射物限制在某个区域内,并利用平面裁剪技术[6]绘

收稿日期:2012-05-20.

基金项目:国家自然科学基金(60873175)、江苏省高校自然科学基础研究项目(07KJD460108)、江苏省普通高校研究生物科研创新计划(CXLX12_0408).

通讯联系人:姜玲燕,硕士研究生,研究方向:数字媒体计算技术. E-mail:jiang070801308@126.com

制将反射区域之外的部分剪切掉来实现反射效果. 本文方法克服了以上方法的不足,且反射效果不会受到纹理等因素的影响.

1　模拟算法

本文的物体反射模拟是基于 OpenGL 实现的,其主要思想是:先设计场景中的物体,再绘制一面“镜子”,这里所谓的“镜子”是呈现反射物体的区域. 客观世界中反射物不会出现在反射平面之外,为模拟这一效果,需要应用 OpenGL 中的模板技术以及裁剪平面技术. 具体分为如下几个步骤:

(1)构建场景中物体的几何模型;为场景设置光照和材质属性[7];

(2)关闭模板缓存,绘制模型;开启模板测试并将模板缓冲区的值设置为 0;绘制反射平面,并将反射平面的模板值设置为 1;

(3)绘制反射物,设置模板值为 1 的区域(反射平面)可被反射物的像素值所取代,其他区域像素值保持不变;利用裁剪平面技术将超出反射平面的反射物裁剪掉.

以下将着重阐述步骤(2)和(3)中的模板和裁剪平面使用方法.

1.1　绘制反射平面

绘制场景之前首先将整个场景中模板缓冲区的值设置为 0. 反射平面是在场景中设定的一个区域,为区别反射区域和其他区域,需利用模板测试将该区域模板缓冲区的值设置为 1,其他区域模板缓冲区的值仍保持 0.

模板技术是使用混色来控制深度在模板之后的物体是否在模板区域的某个位置显示. 采用模板技术可透过深度较靠前的物体(反射平面)看到深度较靠后物体(反射物)的一个部分并限制反射物显示的区域,得到较为真实的半透明场景效果. 图 1 为模板技术原理流程图.

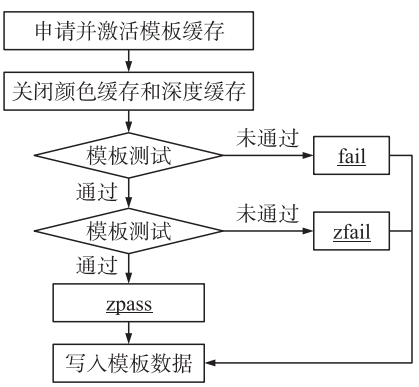

图 1　模板技术原理流程图

Fig. 1　The flow chart of template technology principle

模板缓冲区是存储每个像素附加信息的缓冲区. 采用模板技术实现反射效果时,首先应建立模板缓存,通常情况下建立模板缓冲区的方法与所使用的窗口系统有关. 在使用 GLUT 工具包时,申请模板缓冲区可采用 glutInitDisplayMode( ) 函数. 为简单起见,本文采用 GLUT 作为绘图的窗口系统,代码如下:

```
glutInitDisplayMode ( GLUT _ RGB | GLUT _ DOUBLE | GLUT _ DEPTH|GLUT_STENCIL);
```

在对模板缓存进行操作之前,首先需向 glEnable( ) 函数传递 GLUT_STENCIL_TEST 参数以激活模板缓存,否则无法对模板缓冲区进行操作. OpenGL 通过改写模板缓冲区中的数据来指定绘图区域,为防止写入的数据改写深度缓存和颜色缓存中的数据,需用 glColorMask( ) 函数指定所有的颜色掩码为 GL_FALSE,以关闭帧缓存颜色分量的写操作并用 glDisable( ) 函数关闭深度缓存[4]. 代码如下:

```
glColorMask(GL_FALSE,GL_FALSE,GL_FALSE,GL_FALSE);
glDisable(GL_LIGHTING);
```

完成以上操作之后,对模板缓冲区进行操作,将反射平面区域的模板值设置为 1:

```
glStencilFunc(GL_ALWAYS,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_REPLACE);
Draw the plane…//绘制反射平面.
```

函数 glStencilFunc(Glenum func,Glint ref,Glnint mask)为模板测试设置函数(func)、参考值(ref)和掩码(mask). OpenGL 在模板缓冲区为每一个像素都设置了“模板值”,当像素需要进行模板测试时,将设定的参考值与该像素的“模板值”进行比较,若某一像素点符合条件(func)则将其绘制在屏幕上,否则将其屏蔽.

函数 glStencilOp(GLenum fail,GLenum zfai,GLenum zpass)根据模板测试的情况(具体执行方式如图 1 所示)决定如何修改模板缓冲区的数据.

1.2 绘制反射物

绘制反射物是模拟物体反射效果的重点,首先要确保反射镜像在反射平面内,其次需要利用裁剪平面将显示在反射区域之外的物体裁剪掉. 由于已将反射平面模板缓冲区的值设置为 1,因此将值为 1 的模板缓冲区设置为可被反射物像素取代,而其他区域保持原值,同时启用深度测试使绘制的反射物镜像可以透过反射平面. 确保反射区域中呈现出反射物镜像. 其代码如下:

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_EQUAL,1,0x01);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
Draw the reflection object…//绘制反射物.
```

经过以上操作后,反射平面将会呈现出具有半透明效果的反射镜像. 若有部分反射物超出反射平面,则需要利用裁剪平面技术将超出的部分裁剪掉. OpenGL 中有左、右、远、近、底、顶 6 个视景体裁剪平面,除了这 6 个裁剪平面外,还可以定义 6 个裁剪平面,通过定义这 6 个裁剪平面可以对视景体施加进一步的限制. 其原理为:

平面的数学表达式为:$Ax+By+Cz+D=0$,可表示为:$p\ln[\ ]=\{A,B,C,D\}$.

平面的可见域为:$[A,B,C,D]\cdot\boldsymbol{M}^{-1}[x_e,y_e,z_e,w_e]^{-1}\geqslant 0$. 其中,$[x_e,y_e,z_e,w_e]$为视点的坐标;$\boldsymbol{M}$ 为模型视景矩阵.

进行裁剪平面操作之前需要用 glEnable( )函数启用每个被定义的裁剪平面. 平面数学表达式的 4 个系数 A,B,C,D 决定裁剪掉平面的哪一部分,以下代码表示裁剪前半平面,即屏蔽出现在反射平面之前的反射物:

```
GLdouble mirror_plane[4]={0,0,-1,0};
glClipPlane(GL_CLIP_PLANE0,mirror_plane);
glEnable(GL_CLIP_PLANE0);
```

2 光照和材质

真实感是计算机模拟客观世界的首要标准. 在 OpenGL 中,光照和材质属性是计算机真实感图形绘制非常重要的内容之一,没有光照,虚拟世界就会缺乏立体感、层次感和丰富的色彩;没有材质属性,物体就不会反射光线. 光照和材质属性[7,8]的渲染管线如图 2 所示.

2.1 设置光照属性

(1)为模型顶点设置法向:函数 glNormal*( )定义物体顶点的法向. 正确的法向是设置合适光照的前提:模型的法向决定了它相对于光源的方向. OpenGL 通过法向确定模型各个顶点所接收的光照.

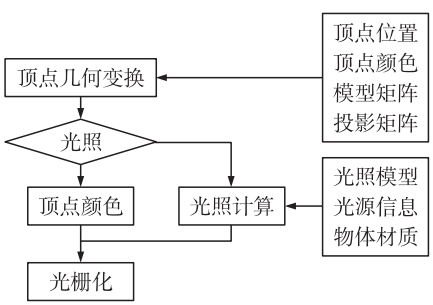

图 2 光照和材质的渲染管线

Fig. 2 The rendering pipeline of light and materials

(2)创建光源:函数 glLight*( )创建光源. 在 OpenGL 中光源具有颜色、位置和方向 3 个属性,以下代码为物体模型设置了光的环境强度、散射强度和镜面强度:

```
static float light_ambient[4]={0.1,0.1,0.1,1.0};
static float light_diffuse[4]={1.0,1.0,1.0,1.0};
static float light_specular[4]={1.0,1.0,1.0,1.0};
glLightfv(GL_LIGHT0,GL_AMBIENT,light_ambient);
glLightfv(GL_LIGHT0,GL_DIFFUSE,light_diffuse);
glLightfv(GL_LIGHT0,GL_SPECULAR,light_specular);
```

(3)选择光照模型:创建光源之后,函数 glLightModel*( )为场景设置光照模型. OpenGL 中光照模型包含 3 个部分:设定全局环境光 RGB 值、视点位置和形体正反面光照模式.

2.2 设置材质属性

现实世界中,物体形态除了和光照有关,还取决于物体本身的材质属性. OpenGL 中材质属性包括环

境、散射、镜面颜色、光泽度以及所有发射光的颜色,它们决定材料对环境光、散射光和镜面反射光的反射程度. OpenGL 通过材料对光线 RGB 的近似反光率来定义材料颜色. glMaterial * ( )函数定义材质属性,调用方式与 glLight * ( )相似.

定义物体的散射颜色的代码为:

static float mirror_diffuse[4] = {0.6,0.6,0.8,0.8};

glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,mirror_diffuse);

图 3 为设置了材质属性的环形和没有设置材质属性的环形,显然前者更加具有立体感和层次感.

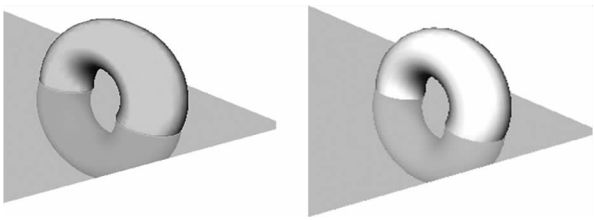

图 3　设置了材质属性的模型(左)和没有设置材质属性的模型(右)

Fig. 3　Setting material properties(left) and without setting material properties(right)

3　结果与分析

本文在普通 PC 机(Intel Pentium 处理器,1 G 内存,Intel(R)G33/G31 显卡)上利用 OpenGL 图形库实现物体反射效果模拟,效果良好,且具有一定的实时性,画面中物体旋转时不会出现失真或者延迟的情况,刷新率为 66 帧/s.

本文结合模板技术和剪裁平面技术,很容易地实现物体在另一个平面上的反射效果. 绘制反射物时可通过剪裁平面将反射物位于反射平面之上的部分剪裁掉,同时利用模板技术限制反射物的显示位置. 图 4(左)显示了未使用模板测试的效果图,反射物在平面之后的物体也绘制了出来;图 4(右)是未使用裁剪平面的效果图,反射物超出平面之前的部分被绘制了出来,以上两种情况都出现了严重的失真. 而图 5 利用模板测试屏蔽了位于平面之后的物体,同时利用裁剪平面将超出平面之前的部分裁剪掉,更加符合客观事实.

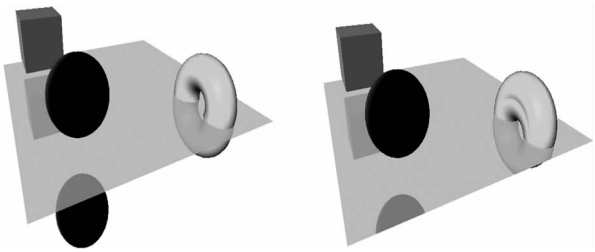

图 4　未使用模板的测试效果图(左)与未使用裁剪平面技术的效果图(右)

Fig. 4　Without using template testing(left) and using template testing(right)

目前模拟物体反射效果时,物体模型大部分是静态的. 本文应用模板缓冲区,可模拟物体在旋转[9,10]时的反射镜像,且不会出现延迟或者失真现象,增强了真实感. 同时模板技术可限制物体显示的区域,因此无论反射平面如何变换,都不会影响反射效果.

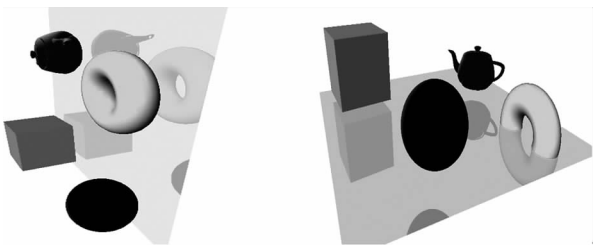

图 5　垂直反射平面(左)与水平反射平面(右)

Fig. 5　Vertical reflection plane(left) and level reflection plane(right)

4　结语

基于 OpenGL 的物体反射效果模拟是计算机图形学和虚拟世界模拟的热点. 本文实现的物体反射效果是从简单、实用、实时性、模拟效果真实度以及算法的简易程度出发设计的一种算法,今后还可以进一步研究模板技术与其他技术的结合,模拟大自然中其他复杂的景象,如模拟自然光照下物体倒映在水波中的景象[11],并进一步考虑物体在反射之后略微变形的情况等.

5　致谢

本文在写作过程中得到了庞明勇教授的悉心指导,特此致谢!

[参考文献](References)

[1] 陈家新,周纬杰. 动态光线跟踪算法与实现[J]. 系统仿真学报,2001,2(13):1-2.
Chen Jiaxin,Zhou Weijie. Dynamic ray tracing algorithm and realization[J]. Journal of System Simulation,2001,2(13):1-2. (in Chinese)

[2] 杨刚. 基于 OpenGL 的 2D 纹理映射研究[J]. 吉林建筑工程学院学报,2011(28):87-89.
Yang Gang. The study of two-dimension texture mapping based on OpenGL[J]. Journal of Jilin Institute of Architecture & Civil Engineering,2011(28):87-89. (in Chinese)

[3] Mark Segal,Carl Korobkin,Rolf van Widenfelt,et al. Fast shadows and lighting effects using texture mapping[J]. Computer Graphics,1992,26(2):249-252.

[4] 马骏,朱衡君,龚建华. 基于 Cg 和 OpenGL 的实时水面环境模拟[J]. 系统仿真学报,2006(18):395-400.
Ma Jun,Zhu Hengjun,Gong Jianhua. Simulation of real-time water surface based on Cg and OpenGL[J]. Journal of System Simulation,2006(18):395-400. (in Chinese)

[5] 薛守良,苏鸿根. 模板应用综述[J]. 计算机工程与设计,2004,(8):1 320-1 322.
Xue Shouliang,Su Honggen. Summarization of stencil buffer[J]. Computer Engineering and Design,2004,(8):1 320-1 322. (in Chinese)

[6] Dave Shreiner,The Khronos OpenGL ARB Working Group. OpenGL 编程指南[M]. 李军,徐波,译. 北京:机械工业出版社,2011.
Dave Shreiner,The Khronos OpenGL ARB Working Group. OpenGL Programming Guide[M]. Li Jun,Xu Bo,Translated. Beijing:China Machine Press,2011. (in Chinese)

[7] 杨健,张敏. OpenGL 中的光照技术研究[J]. 软件导刊,2011(4):84-86.
Yang Jian,Zhang Min. The research of illumination based on OpenGL[J]. Software Guide,2011(4):84-86. (in Chinese)

[8] Wolfgang Heidrich, Hans-Peter Seidel. Realistic, hardware accelerated shading and lighting [C]//Computer Graphics Proceedings,Annual Conference Series,ACM SIGGRAPH. Los Angeles,1999:171-178.

[9] 刘慧杰,靳海亮. 基于 VC++的 OpenGL 三维图形开发设计[J]. 计算机与数学工程,2009(7):123-125.
Liu Huijie,Jin Hailiang. Development and design of three-dimensional graphics based on VC++ and OpenGL[J]. Computer & Digital Engineering,2009(7):123-125. (in Chinese)

[10] 王莹莹. 在 VC 中利用 OpenGL 实现动态效果图像的技巧[J]. 微型电脑应用,2002(6):50-54.
Wang Yingying. Using OpenGL to realize dynamic image techniques in the VC[J]. Microcomputer Applications,2002(6):50-54. (in Chinese)

[11] Kei Iwasaki,Yoshinori Dobashi,Tomoyuki Nishita. A fast rendering method for refractive and reflective caustics due to water surfaces[J]. Computer Graphics Forum,2003,22(3):601-609.

[责任编辑:严海琳]