ISSN 1009-3044
Computer Knowledge And Technology 电脑知识与技术
Vol.4,No.7,December 2008, pp.1749-1751

E-mail: eduf@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

openGL 几何变换与 Windows 屏幕坐标转换的研究

翁伟[1],黄翠兰[1],蔡建立[2]

(1.厦门理工学院 计算机科学与技术系,福建 厦门 361024;2.厦门大学 自动化系,福建 厦门 361005)

摘要: 分析 openGL 三维图形显示流程,讨论其坐标变换过程,推导了 openGL 坐标与 Windows 屏幕坐标的转换关系,给出了在 MFC 下实现 openGL 图形编程的方法。

关键词: openGL;几何变换;坐标转换

中图分类号: TP391 **文献标识码:** A **文章编号:** 1009-3044(2008)34-1749-03

The Research of OpenGL Geometric Transformation and Conversion of the Windows Screen Coordinates

WENG Wei[1], HUANG Cui-lan[1], CAI Jian-li[2]

(1.Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China; 2. Department of Automation, Xiamen University, Xiamen 361005, China)

Abstract: By analysing the processes of openGL 3D graphics and discussng it´s coordinate alternation process, the thises derives the transformation relationship between openGL coordinate and the Windows screen coordinate. Meanwhile it gives the openGL graphical programming approach under the platform of MFC.

Key words: openGL;geometric transform;coordinate transformations

1 引言

openGL(open graphics library,开放性图形库)是以 SGI 的 GL 三维图形库为基础制定的一个开放式三维图形库,目前得到了许多软硬件厂商的支持,是世界上最流行的图形 API 之一,已经成为一个工业标准。openGL 独立于硬件设备,窗口系统和操作系统,可移植性高,主要有绘制模型、几何变换、光照处理、纹理映射、图像操作、动画等功能[1],已经在军事、航天、医学、虚拟现实等领域得到了广泛的应用[2]。

利用 openGL 进行三维图形绘制时,世界坐标系的三维物体位置坐标信息经过几何变换、投影变换、裁剪变换和视口变换这几个空间坐标变换过程变为屏幕坐标系的坐标。论文[3]分析了 openGL 常见坐标使用中的问题;论文[4]讨论了坐标变换相关的 openGL API,但没有涉及几何变换的具体公式,亦没有研究 openGL 坐标和屏幕坐标的转换问题,不便于进行交互图形设计;论文[5]讨论了正投影和透视投影下 openGL 坐标和屏幕坐标的转换关系,但设定了判断条件,不具有一致性。

本文首先在世界坐标系下讨论了 openGL 的几何变换过程,然后推导了 openGL 坐标与 Windows 屏幕坐标的转换关系,最后给出了在 MFC 下实现 openGL 图形编程的方法。

2 openGL 三维图形显示流程

绘制三维图形是一项复杂的任务,对于一个特定的图形,光照、材质、纹理等因素都会影响它的绘制,openGL 采用状态机模型来控制这些非位置因素[6],由一系列的点所组成的图元受状态机控制以便形成真实感图形。openGL 要将现实三维图形显示在二维屏幕上,关键是一系列的几何变换,包括平移、旋转、缩放、投影、裁剪和视口变换,基本过程如图 1 所示[7]。

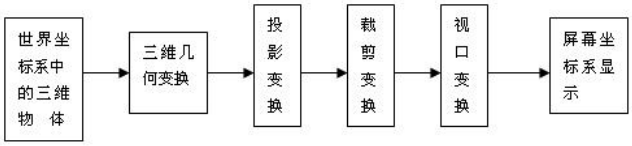

图 1 三维图形的显示流程

2.1 三维几何变换

openGL 中的世界坐标系 x 轴正方向水平向左,y 轴正方向水平向上,z 轴正方向垂直屏幕向外。三维几何变换均在世界坐标系中完成,openGL 提供了几个 API 实现这些变换,相关函数原型如下:

1) 平移

glTranslate{fd}(TYPE x,TYPE y,TYPE z);

x,y,z 为 x 轴、y 轴和 z 轴方向上的平移量。

2) 旋转

glRotate{fd}(TYPE angle,TYPE x,TYPE y,TYPE z);

angle 为旋转角度,后三个变量与原点的连线构成旋转轴。

3) 缩放

glScale{fd}(TYPE x,TYPE y,TYPE z)

该函数可以对物体沿 x 轴、y 轴和 z 轴分别放大或缩小。

三维几何变换中还有一个视点变换,函数原型如下:

收稿日期:2008-08-28
基金项目:福建省教育厅 B 类课题(JB0527)
作者简介:翁伟(1979-),男,湖南衡阳人,讲师,硕士,研究方向:图形图像、数据挖掘;黄翠兰(1969-),女,福建厦门人,副教授,研究方向:程序设计和数据库技术;蔡建立(1950-),男,福建泉州人,副教授,研究方向:图形图像自动控制理论。

Void gluLookAT(GLdouble ex,GLdouble ey, GLdouble ez, Gldouble cx, Gldouble cy, Gldouble cz, Gldouble ux, Gldouble uy, Gldouble uz)

openGL 成像原理与摄像机拍摄类似,这个函数功能就是设定摄像机的位置,点(ex,ey,ez)三个参数指定摄像机位置,点(ex,ey,ez)到点(cx,cy,cz)方向为摄像机拍摄方向,(ux,uy,uz)指定了向上向量的方向,所有参数均相对于世界坐标系,相当与在世界坐标系中建立了一个以摄像机位置为坐标原点的视坐标系,如图 2 所示。通常,摄像机置于世界坐标系上的 z 轴上,拍摄方向为 z 的负轴,类似于人眼观察屏幕。

事实上,openGL 对顶点坐标位置的一系列变换为矩阵运算。图形在世界坐标系中进行一系列的平移、旋转和缩放,然后变换到视坐标系中。这就相当于图形在世界坐标系中进行平移、旋转和缩放后,再进行一次世界坐标系的平移和旋转变换,这个变换的效果与世界坐标系到视坐标系的变换等效就可以了。

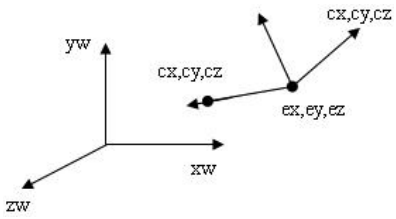

图 2　世界坐标系和视坐标系

2.2 投影变换

投影变换是模仿人类视觉效果的重要手段,离我们远的物体看起来小,而进的物体看起来大。投影函数如下:

void glFrustum (Gldouble left,Gldouble rihgt,Gldouble bottom,Gldouble top, Gldouble near,Gldouble far)。

如图 3 所示,该函数建立透视投影模型,并设定了裁剪范围(称为视景体),只有在棱锥体中 near 和 far 所截取的棱台里面的物体才是可视部分,物体位于这个棱台之外的部分被裁剪掉了。该图中相机位于坐标系原点,相机所对的方向为 z 轴负方向,投影面是 z=-near 所指的平面(near 和 far 只取正值)。投影面中由 left、right、bottom 和 top 四个参数所围成的矩形可以看成是视口变换中的窗口(事实上,openGL 中有一系列的标准化变换过程)。设视景体中有点(x,y,z),投影到窗口上对应的点为(x_w,y_w,near),那么应满足如下关系式:

$$\frac{x}{x_w}=\frac{z}{near} \tag{1}$$

$$\frac{y}{y_w}=\frac{z}{near} \tag{2}$$

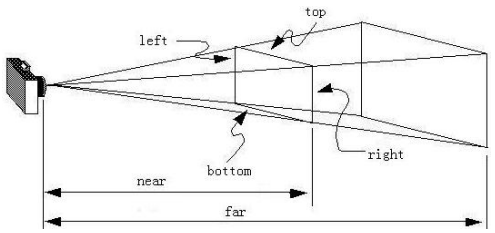

图 3　透视投影模型

另一类投影为平行投影,这种投影无论物体距离相机多远投影后物体的大小都不变,虽不是人类视觉效果的模拟,但在实践中经常使用,其投影函数如下:

void glOrtho(Gldouble left,Gldouble rihgt,Gldouble bottom,Gldouble top, Gldouble near,Gldouble far)。

该函数对应的模型如图 4 所示,位于长方体中的物体参与投影运算,长方体之外的部分被裁剪掉,投影面就是 z=-near 所指的平面。视景体中的点投影到窗口上 x 坐标与 y 坐标都不变。

还有其他方法实现投影,但就效果上来说,与这两类投影类似。

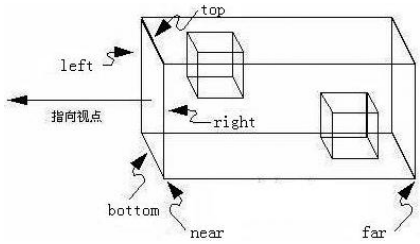

图 4　平行投影模型

2.3 裁剪变换

在投影变换模型中已经包含了裁剪变换,openGL 还可以在定义裁剪平面,该平面将视景体中的物体截成两部分,平面之上的部分不参与投影,平面之下的部分才参与投影:

Void glClipPlane(Glenm plane,Const Gldouble *equation)

参数 plane 指定了裁剪面号,equation 指向 4 个系数数组,这 4 个系数为平面方程 Ax+By+Cz+D=0 的系数。

2.4 视口变换

最后一步为窗口到视口的变换,视口对于屏幕上的显示区域,该区域一般充满整个屏幕,可以由以下函数设定:

glViewport(Glint vx,Glint vy,Glsizei width,Glsizei height)。

窗口变换到视口变换的过程如图 5 所示。

根据比例关系,窗口内的点(x_w,y_w)映射到视口内的点(x_v,y_v)应满足:

$$\frac{x_w-left}{right-left}=\frac{x_v-vx}{width} \tag{3}$$

$$\frac{y_w-bottom}{top-bottom}=\frac{y_v-vy}{height} \tag{4}$$

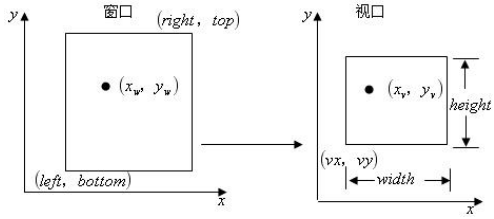

图 5　窗口到视口的转换

3 openGL 与 Windows 坐标转换

设视景体中的点(x,y,z)经过透视投影到窗体上的点为(x_w,y_w,near),继而点(x_w,y_w,near)对应到视口上的点为(x_v,y_v,0),那么根据(1)和(3)有:

$$x=\frac{(x_v-vx)\times(right-left)\times z}{width\times near}+\frac{left\times z}{near} \tag{5}$$

据(2)和(4)有:

$$y=\frac{(y_v-vy)\times(top-bottom)\times z}{height\times near}+\frac{bottom\times z}{near} \tag{6}$$

一般情况下,Windows 屏幕坐标左上角为坐标原点,与图 2 中视口坐标原点在左下角不一致;另外,视口也不一定充满整个屏幕,为了一般化公式,在进行一次 Windows 屏幕坐标到视口坐标的转换,如图 6 所示。

图 6 中点(point x,point y)为 Windows 屏幕坐标,为了转换为视口坐标,需要知道 Windows 窗口的宽度和高度。Windows 窗口坐标有很多中单位,一般采用像素坐标,其宽度和高度可用 Windows API 得知,设保存在结构体 m_rect 中,m_rect.width 保存 Windows

屏幕宽度,m_rect.height 保存 Windows 高度。那么,对应的视口坐标为:

$$x_v=\text{point x} \tag{7}$$

$$y_v=\text{retc.bottom}-\text{point y} \tag{8}$$

将(7)式代入(5)式中,有:

$$x=\frac{(point.x-vx)\times(right-left)\times z}{width\times near}+\frac{left\times z}{near} \tag{9}$$

将(8)式代入(6)式,有:

$$y=\frac{(m\_rect.bottom-point.y-vy)\times(top-bottom)\times z}{height\times near}+\frac{bottom\times}{near} \tag{10}$$

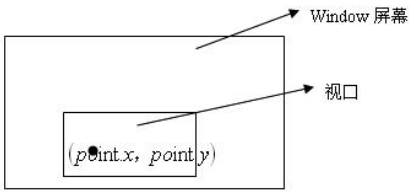

图 6 Windows 窗口到视口的转换

式(9)和(10)即是 Windows 屏幕坐标(point x,point y)到视景体中坐标(x,y)的转换公式。若是采用平行投影,此时视景体中的点投影到窗口上 x 坐标和 y 坐标都没有变,直接根据式(3)和(4),有:

$$x=\frac{(x_v-vx)\times(right-left)}{width}+left \tag{11}$$

$$y=\frac{(y_v-vy)\times(top-bottom)}{height}+bottom \tag{12}$$

在 Windows 编程中,屏幕上任何一点的坐标可以方便地用鼠标事件等方法获得。该公式能够通过获得屏幕坐标逆向算出视景体中的坐标,在交换式图形图像编程中有广泛的应用。

4 基于 MFC 的实现

MFC(Microsoft Foundation Class)提供了完善的基础类库,其框架下的消息相应机制和多态性给程序设计带来了极大的方便。openGL 是一个与平台无关的三维图形接口,采用客户/服务器机制,服务器解释绘图命令。那么像素格式管理和渲染环境管理必须通过由其他语言和平台来设置。oepenGL 图形库被封装在一个动态链接库 OpenGL32.DLL 中,客户应用程序调用的 openGL 函数均在此动态链接库中处理,然后传给服务器 WINSRV.DLL,处理后再传给 Win32 的设备驱动接口,这样视频显示驱动就能处理显示了。openGL 采用渲染描述表(Render Context)的绘图方式,与 Windows 的 GDI 绘图方式不同。一种实现方式如下:

1) 加入 openGL 头文件;设置全局变量。

2) 在视图类的 PreCreateWindow 方法中设置窗口类型风格支持 openGL 绘图:cs.style=WS_CHILD|WS_VISIBLE|WS_CLIPCHILDREN|WS_CLIPSIBLINGS。

3) 在视图类的 OnCreate 方法中设置像素格式,明确绘制风格、颜色模式、颜色深度等重要信息;设置 openGL 操作所必须的渲染描述表 RC 与 DC 的关联;还可以设置投影变换和视口变换。

4) 改写 OnSize 方法。该方法在视图尺寸发生变化的时候激发,此时要通知 openGL 进行改变图形大小等操作,主要是设置 glViewport 函数。

5) 改造 OnDraw 函数。在这个函数中完成每次的屏幕绘制,包括三维几何变换和颜色、光照和纹理信息的处理。

6) 若要获取鼠标的 Windows 坐标,进行交互式绘图,可以改写 OnLButtonDown 函数,将鼠标位置参数转换为视景体中的坐标位置,然后调用 OnDraw 函数绘制。

7) 在 OnDestroy 函数中释放渲染描述表 RC 等资源。

5 结束语

本文重坐标变换的角度探讨了 openGL 绘制三维图形的过程,推导了 Windows 坐标和 openGL 坐标相互转换关系,并提供了一种 MFC 绘制 openGL 图形的方法。这些方法在笔者进行的人脸建模中得到了应用,方便了坐标拾取,实现了鼠标点选和微调特征点。总之,本文提出的方法提供了 openGL 坐标反馈机制,拓宽了 openGL 绘图应用,方便了交互图形编程。

参考文献:

[1] 杨钦,徐永安,瞿红英.计算机图形学[M].北京:清华大学出版社,2005.
[2] 僧德文,李仲学,李春明,等.基于 openGL 的真实感图形绘制技术及应用[J].计算机应用研究,2005(3):173-175.
[3] 林夏菲,陈磊,熊辉.openGL 中关于坐标使用常见问题的分析[J].电脑知识与技术,2008,2(1):130-131,137.
[4] 高美真,黄娇青.openGL 中的图形变换[J].焦作师范高等专科学校学报,2006,22(2):51-53.
[5] 蒲玮.openGL 与 Windows 坐标转换技术的研究[J].电脑开发与应用,2008,21(1):8-10.
[6] Richard S W,Jr Benjamin L.OpenGL 超级宝典[M].3 版.徐波,译.北京:人民邮电出版社,2005.
[7] 徐明亮,卢红星,王琬.openGL 游戏编程[M].北京:机械工业出版社,2008.

(上接第 1740 页)

参考文献:

[1] Steketee S N,Badler N I.Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control [C].Proceedings of SIGGRAPH'85,1985:255-262.
[2] Girard M.Interactive Design of 3D Computer-Animated Legged Animal Motion[J].IEEE Computer Graphics and Applications,1987,7(6):39-51.
[3] Bruderlin A,Calvert T W.Goal-Directed Dynamic Animation of Human Walking[J].Computer Graphics,1989(23):233-242.
[4] Hearn D,Baker M P.Computer Graphics with OpenGL, Third Edition[M].Publlishing House of Electronics Industry,2005:355-362.
[5] Saunder J B,Inman V T,Eberhart H D.The major determinants in normal and pathological gait[J].J Bone Joint Surgery,1953(35):543-558.
[6] 乔立,苏鸿根.基于分运动叠加的个性化虚拟人步行姿态研究[J].计算机工程与设计,2004(1):126-129.
[7] Chung S K,Hahn J K.Animation of Human Walking in Virtual Environments[J].Computer Animation',1999:4-15.