基于 OpenGL 的三维物体的多视角显示

赵 丽 梅

(贵州大学 贵阳 550025)

摘 要 文章介绍了 OpenGL 中取景变换的基本概念、原理,并通过程序实例详细叙述了实现三维物体多视角显示的方法。采用该方法可实现更多角度的视图变化,进而实现对物体的平移和缩放。

关键词 OpenGL;视点坐标系;视点变换;投影变换;视景体;视口

Abstract: This paper introduces the basic concepts, principles of view transformation with OpenGL and takes an example to describe the process of how to get the multi-view of three-dimensional objects in detail.

Key words: OpenGL; eye coordinate; viewing transformation; projection transformation; viewing volume; viewport

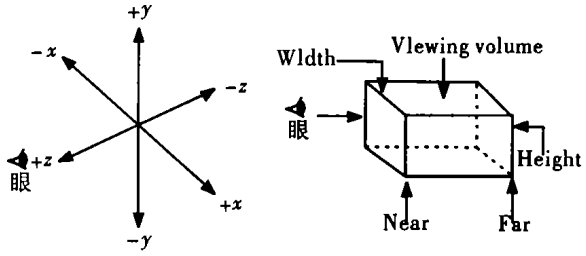
0 引 言

OpenGL(open graphics library)是一个优秀的三维图形硬件的软件接口,也是一个三维图形和模型库。OpenGL 具有建模、变换、光线处理、色彩处理、动画以及其他更先进的高级图形处理能力,如纹理映射、物体运动模糊效果等,可以制作真实感非常强的三维图形。目前 OpenGL 已被广泛接受,已有几十家大公司采用 OpenGL 作为标准图形的软件接口。OpenGL 事实上已成为高性能的图形和交互视景处理的工业标准,能够在 Windows 95/98、Windows NT/2000、Macos、Beos、OS/2 及 Unix 上应用。三维 CAD 软件广泛使用 OpenGL 进行实体造型与仿真。在 CAD 软件中,观察视图的选择,即定义从什么角度观察模型,是使用频率最高的操作之一,对于一个模型,用户需要从不同的位置和角度对它进行观察。在 OpenGL 中,对观察模型的位置和角度的定义是通过取景变换来实现的。

1 OpenGL 取景变换的基本概念

1.1 视点坐标系和视图变换

在 OpenGL 中,绘制图形顶点使用世界坐标系(用户坐标系)。它是一个右手三维直角坐标系,用户用来定义几何形体和图素。如图 1(a)所示,视点坐标系由视点位置和观察方向决定,是一个右手坐标系,视线方向垂直于 xy 平面,$+x$ 和 $+y$ 分别指向视点的右边和上边。$+z$ 方向和视线方向相反,$-z$ 方向表示从视点指向屏幕的方向。在 OpenGL 默认状态下,即没有任何视点变换的情况下,视点坐标系与世界坐标系一致。



(a) 视点坐标系 (b) 视景体

图 1 视点坐标系和视景体示意

视点变换就是移动视点,相当于反向移动物体。可以通过移动视点的位置和改变视点的观察方向,从不同位置、不同角度对世界坐标系中的场景进行观察。视图变换的本质是改变视点坐标系与世界坐标系之间的位置关系。实现视点变换的方法就是使用 OpenGL 实用库函数 *glLookAt*( )。该函数可直观、方便地定义一个观察位置和观察方向[1]。它的原型定义如下。

void *gluLookAt* (GLdouble *eyex* , GLdouble *eyey* , GLdouble *eyez* , GLdouble *centerx* , GLdouble *centery* , GLdouble *centerz* , GLdouble *upx* , GLdouble *upy* , Gldouble *upz*);

该函数的参数包括 3 个部分,即:

1)视点位置(*eyex*,*eyey*, *eyez*)。

收稿日期:2003-10-10

2)参考点(视线正前方的一个位置)的位置(*centerx*, *centery*, *centerz*)。

3)视线向上的方向(*upx*, *upy*, *upz*)。

1.2 投影变换与视景体

投影变换定义的是一个视景体,即所要观察的物体空间,显示在屏幕上的内容就是视景体中的内容。大多数 CAD 软件都采用正交投影,如图 1(b)所示。

正交投影定义的是一个长方形的视景体,物体在屏幕上的显示尺寸不受几何体距离的影响。OpenGL 中使用函数 *glOrtho*( ) 来定义正交投影:

glOrtho (GLdouble *right* , Gldouble *bottom* , Gldouble *top* ,Gldouble *near* ,Gldouble *far*);

其中:*left*、*right* 为垂直剪切面的坐标;*bottom*、*top* 为水平剪切面的坐标;*near*、*far* 分别为近视点剪切面和远视点剪切面。

1.3 视口变换

视口是指 Windows 窗口的客户区中绘制图像的区域。要把物体最终显示在窗口中,还需要进行视口变换。视口变换就是指把视景体的投影面映射到视口的过程。视口变换决定 OpenGL 绘制的物体将输出在窗口中的哪一部分。

OpenGL 中定义视口的函数如下。

Void *glViewport*(*GLint x*, *Glint y*, *Glint width*, *Glint height*)。

其中:*x* 和 *y* 分别是视中左下角窗口中的位置;*width* 和 *height* 分别是视口的宽度和高度。

2 观察视图的定义

定义观察视图实际上就是定义观察者的位置和视线的方向。在此实例中使用函数 *glLookAt*( ) 来定义视点的位置和方向,*glLookAt*( ) 使用视点位置 *m_eye*、参照点 *m_ref*、视线的上方向 *m_vecUp* 来定义视图变换,这样类似于用户用肉眼观察景物。定义一个典型观察视图,只需要定义合适的视点、参照点和视线的上方向 3 个参数。由于参照点 *m_ref* 通常取在被观察模型接近中心的位置,所以定义不同的观察视图时,尽量保持参照点的位置不变,再通过参照点位置以及视线方向推算,算出视点 *m_eye* 的位置。计算出新的视点位置以后,还需要重新计算视线的上方向 *m_vecUp* [2]。

3 程序实现

3.1 主要源代码

1)用于初始化的成员函数 *init*( ),对视点和观察方向、视景体定义、视口尺寸等进行初始化。

```
void GCamera::init()
{
.....}
```

2)用于视口设置的成员函数 *set_screen*( )。当 Windows 窗口尺寸变化时,需要对视口的尺寸进行重新设置。同时,为了使图形的显示比例不随窗口尺寸的变化面变化,还要根据新的窗口尺寸重新计算视景体的宽和高。

```
void GCamera::set_screen( int x, int y)
{
    glViewport(0,0,x,y);//设置视口
    if(y==0) y=1;
    double ratio = (double)x/(double)y;
    m_width * = (double)x/m_screen[0];//根据窗口尺寸变化更新视景体的宽与高
    m_height * = (double)y/m_screen[1];
    m_width =
    m_height * ratio;//保持视景体宽与高的比率与窗口一致
    m_screen[0] = x;
    m_screen[1] = y;
}
```

3)用于投影变换的成员函数 *projection*( ),是一个关键函数,它根据类中的数据来创建投影变换矩阵,并通过 OpenGL 实用库函数 *gluLookAt*( ) 来定义视点变换。

```
void GCamera::projection()
{
    //切换到投影变换矩阵设置
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();//初始化投影矩阵
    glRenderMode(GL_RENDER);//渲染模式
    //创建投影矩阵
    double left = - m_width/2.0;
    double right = m_width/2.0;
    double bottom = - m_height/2.0;
    double top = m_height/2.0;
    glOrtho(left,right,bottom,top,m_near,m_far);//正交投影方式
    glMatrixMode( GL_MODELVIEW );//切换到视图变换矩阵设置
    glLoadIdentity( );//初始化视图变换矩阵
    gluLookAt(m_eye.x,m_eye.y,m_eye.z,m_ref.x,m_ref.y,m_ref.z,m_vecUp.dx, m_vecUp.dy, m_vecUp.dz);
    //设置视点位置和观察方向
```

```
}
```

4)用于设置视点和视景体的成员函数。

```
void GCamera::set_eye(double eye_x,double eye_y,double eye_z)//设置视点的位置和方向
{
    ......}
void GCamera::set_ref(double ref_x,double ref_y,double ref_z)//设置参考点的位置和方向
{
    ......}
void GCamera::set_vecUp(double up_dx,double up_dy,double up_dz)//设置视线向上的位置和方向
{
    ......}
void GCamera::set_view_rect(double width,double height)//设置视景体
{
    ......}
void GCamera::get_view_rect(double& width,double& height)//设置视景体
{
    ......}
```

5)典型观察视图的选择函数 *set_view_type*( )。

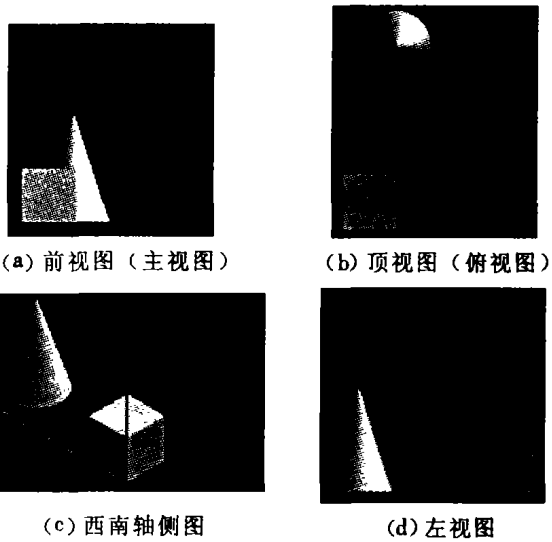
```
void GCamera::set_view_type( int type )//典型观察视图的选择
{
    double r;
    CVector3D vec;
    vec = m_ref - m_eye;//矢量 vec 表示视线方向
    r = vec.GetLength();//视点与参考点的距离
    if(IS_ZERO(r)) r = 50.0;
    if( r > 10000) r = 10000;
    switch(type){
    case VIEW_FRONT: //前视图(主视图)
      m_eye = m_ref + CVector3D(0,-r,0);//移动视点位置
      m_vecUp = CVector3D(0,0,1);
      break;
    case VIEW_TOP: //顶视图(俯视图)
      m_eye = m_ref + CVector3D(0,0,r);//移动视点位置
      m_vecUp = CVector3D(0,1,0);
      break;
    case VIEW_LEFT: //左视图
      m_eye = m_ref + CVector3D(-r,0,0);//移动视点位置
      m_vecUp = CVector3D(0,0,1);
    break;
    case VIEW_SW_ISOMETRIC://西南方向轴侧图
      m_eye = m_ref + CVector3D(-1,-1,1).GetNormal() * r;//移动视点位置
      update_upVec();
      break;
    }
}
```

6)函数 *update_upVec*( ),根据新的视点位置,更新视线上方向 *m_vecUp* 。

```
void GCamera::update_upVec()
{
    CVector3D vec = m_ref - m_eye;//视线方向矢量
    CVector3D zVec(0,0,1);
    CVector3D vec0;
    vec.Normalize();//矢量单位化
    vec0 = vec * zVec;
    m_vecUp = vec0 * vec;//矢量 m_vecUp 与视线方向垂直
}
```

3.2 实验结果

根据程序实现实验结果如图 2 所示。



(a)前视图(主视图) (b)顶视图(俯视图)

(c)西南轴侧图 (d)左视图

图 2 实验结果

4 结束语

本文详细介绍了用 OpenGL 提供的功能函数实现对三维物体的多视角显示,理解了上述基本原理和概念,不仅可实现更多角度的视图变换,还可进一步实现对物体的平移和缩放等操作。

参考文献

1 Wright Richard S, Sweet Jr. Michael . OpenGL 超级宝典 . 北京:人民邮电出版社,2001. 119 ~ 152

2 王清辉,王彪 . Visual C + + CAD 应用程序开发技术 . 北京:机械工业出版社,2003. 79 ~ 85