一种 OpenGL 局部缩放算法及应用

张立成[1]，张 鸽[2]
（1.长安大学 信息工程学院，陕西 西安 710064；
2.西安中交土木科技有限公司，陕西 西安 710075）

摘 要：缩放是 OpenGL 三维模型展示的基本操作之一，一般缩放时由于整个场景围绕视景体中心缩放，感兴趣区域在视图窗口中的位置会不断变化甚至离开视景体，需要不断地执行平移操作，不断地修正感兴趣区域在视图窗口中的位置，无法集中精力观察工程计算中模型的变化细节。为了解决该问题，从图形学角度提出了一种改进的局部缩放算法，实现了用鼠标滚轮缩放模型时，鼠标选择处的模型不离开视景体而相对视图窗口的位置保持不变，省去了传统算法中用户进行缩放操作时需要不断进行平移的操作，改进了用户体验。该算法在多个可视化项目中得到实际应用，取得了很好的操作体验。

关键词：计算机应用；坐标变换；局部缩放；可视化；视景体
中图分类号：TP319 **文献标识码**：A **文章编号**：1674-7720(2013)19-0044-04

An OpenGL-based local scaling algorithm and its application

Zhang Licheng[1], Zhang Ge[2]
(1.School of Information Engineering, Chang'an University, Xi'an 710064, China;
2.CCCC Civil Engineering Science & Technology Co., Ltd., Xi'an 710075, China)

Abstract: Local scaling is one of the most popular operations in OpenGL graphic applications. Normally, the entire scene zooms around the center of the view volume with mouse wheel, making the region of interest unstable in position and even be out of sight. Thus, many translation operations are required to move the target in sight. So, attentions can not be focused on detail deformation of engineering calculation. A new algorithm from the viewpoint of computer graphics is provided, which makes the selected point of the model fixed relative to the viewport while the mouse wheels. Thus, users needn't perform translation operations to move the region of interests in sight. The algorithm has been applied in several visualization projects and achieved good operating experience.

Key words: computer application; coordinate transformation; local scaling; visualization; view volume

OpenGL 是一个与硬件平台无关、与系统平台无关的三维图形库，其在三维真实感图形制作中具有优秀的性能，已经成为高性能的图形和交互视景处理的标准。OpenGL API 由 200 多个函数组成，主要提供图形绘制、变换操作、颜色模式、光照、图像效果增强、位图和图像、纹理映射、交互与动画 8 个方面的功能。一些跨平台的三维库(如 OSG)也是建立在 OpenGL 之上的，对OpenGL 的 API 进行了封装，利用这些三维库，开发人员可以开发丰富的交互式应用程序[1-3]。GPU 高性能计算的支持，使得 OpenGL 开发的三维程序运行更加流畅、场景更加逼真[4]。

OpenGL 具有强大的图形处理功能，包括图形的平移、旋转、缩放等。灵活运用 OpenGL 的这些功能，可以实现很多更复杂的操作。但在一般情况下，当利用 OpenGL 进行图形的缩放时，往往是以视景体的中心为缩放中心进行整体缩放[5]，该算法虽然在整体上实现了一定的缩放功能，但缩放的过程中，用户期望的缩放区域会在 Windows 视图窗口中不断变化甚至离开视景体。以放大为例，为了观察感兴趣区域的细节，如模型在外力作用下的变化过程，往往需要借助平移将目标区域移到合适位置后才能继续放大，在放大达到一定系数时，放大-平移的操作非常频繁，严重影响了操作体验。参考文献[6]提出了一种拉框放大算法，计算鼠标框选区域在视景体中的位置，然后将这个位置重新投影到视口

上，该算法虽然能够实现放大，但是不能够局部缩小，且放大与缩小操作的切换不连贯。本文提出的算法不仅能够实现局部缩放的效果，而且在操作方法上也作了改进，用户滚动鼠标的滚轮，默认以光标处的模型为感兴趣区域中心，缩放将围绕该中心进行，而且用户期望的缩放区域在视景体中原地缩放，缩小和放大操作切换流畅。

1 OpenGL 变换的基本原理

三维模型显示到二维屏幕的过程分为造型变换、取景变换、投影变换和视口变换 4 个阶段[7]，如图 1 所示。其中，方框中为坐标变换名称，箭头上为坐标系名称。

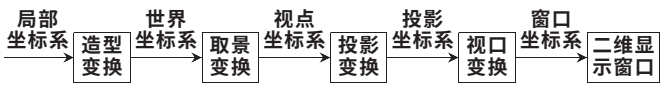


图 1 OpenGL 三维图形显示流程示意图

造型变换将各个处于自身局部坐标系中的模型变换到世界坐标系中组成整个场景；取景变换将定义在世界坐标系下的场景变换到视点坐标系中；投影变换将视点坐标系中的场景投影到二维视窗区域；视口变换将视窗中的投影结果转换到屏幕坐标系中[8]。

本文讨论的相关算法涉及变换序列中的投影变换和视口变换，为了使物体在屏幕上的显示尺寸不受所处距离远近的影响，本文采用正投影。正投影将裁剪后的视景体投影到二维投影平面上，映射到以像素为单位的屏幕坐标系的过程这里简化为式(1)，因为 Windows 视图窗口所在坐标系的默认形式为：视图窗口左上角为原点，向右为 X 正方向，向下为 Y 正方向(不考虑普通二维变换和光栅化等步骤)。

$$x_v=x_w;$$
$$y_v=Win_H-y_w \tag{1}$$

其中，(x_w,y_w) 为窗口中某个像素点的坐标，(x_v,y_v) 为视景体投影到二维投影平面上的坐标，Win_H 为窗口的高度。

2 算法的详细实现过程

以放大为例，放大的实现原理如图 2 所示。视景体中的点和 Windows 视图窗口中的点是一一对应的(此处不考虑深度)。图 2 中左侧为投影面，右侧为视图窗口。其中，投影面上的实线表示放大前视景体的尺寸，虚线表示放大后视景体的尺寸。

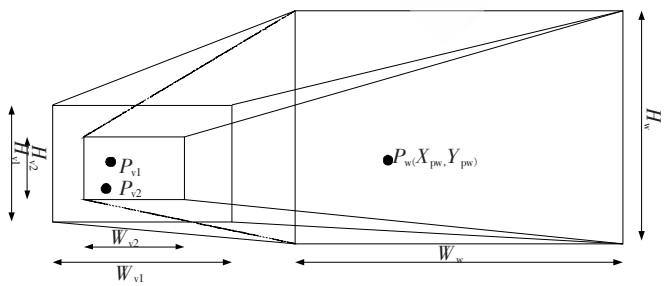


图 2 一般缩放算法的原理图

由于放大后的视景体的长度和宽度都小于放大前的视景体的长度和宽度，在视图显示窗口尺寸不变的前提下，模型只显示虚线中的部分，从而达到放大的效果。

假设点 $P_w(X_{pw},Y_{pw})$ 为视图窗口中任一点，它对应于视景体中点为 P_{v1}，P_{v1} 到视景体左边缘和下边缘的距离分别为 X_{pw1} 和 Y_{pw1}，假设视图窗口的尺寸在放大的过程中保持不变，长度和宽度分别为 W_w 和 H_w。放大前视景体的长度和宽度为 W_{v1} 和 H_{v1}。则有：

$$X_{pw1}=\frac{X_{pw}}{W_w}\times W_{v1}$$
$$Y_{pw1}=\frac{H_w-Y_{pw}}{H_w}\times H_{v1} \tag{2}$$

假设放大后视景体的长度和宽度分别为 W_{v2} 和 H_{v2}，那么在 W_w 和 H_w 不变的情况下，点 P_w 在新的视景体中对应点 P_{v2} 为：

$$X_{pw2}=\frac{X_{pw}}{W_w}\times W_{v2}$$
$$Y_{pw2}=\frac{H_w-Y_{pw}}{H_w}\times H_{v2} \tag{3}$$

其中，X_{pw2} 和 Y_{pw2} 分别是点 P_{v2} 到视景体左边缘和下边缘的距离。此时点 P_{v1} 的位置有以下两种情形：

(1)位于新的视景体外部，此时视图窗口已经看不见 P_{v1} 对应的点。

(2)仍位于新的视景体内部，此时视图窗口仍能看见 P_{v1} 对应的点。设放大操作的系数为 α ($\alpha<1$ 即为放大)，则有：

$$W_{v2}=\alpha\times W_{v1}$$
$$H_{v2}=\alpha\times H_{v1} \tag{4}$$

由式(2)、式(3)和式(4)得：

$$X_{pw2}=\alpha\times X_{pw1}$$
$$Y_{pw2}=\alpha\times Y_{pw1} \tag{5}$$

设 P_{v1} 到新的视景体左边界和下边界的距离分别为 X'_{pw1} 和 Y'_{pw1}，则有：

$$X'_{pw1}=X_{pw1}-0.5\times(1-\alpha)W_{v1}$$
$$Y'_{pw1}=Y_{pw1}-0.5\times(1-\alpha)H_{v1} \tag{6}$$

令 $X_{pw2}=X'_{pw1}$ 且 $Y_{pw2}=Y'_{pw1}$，可得 $X_{pw1}=0.5\ W_{v1}$，$Y_{pw1}=0.5H_{v1}$，即只有视景体中间的点在正常缩放时一直在 Windows 视图窗口的中间。由于 X_{pw2} 和 X'_{pw1}、Y_{pw2} 和 Y'_{pw1} 在一般情况下不等，此时除视景体中间的点会在 Windows 视图窗口中不断改变位置，变得很难定位和控制。

若要使放大后 P_w 对应的三维模型的点不变，需要在视景体变化之前对模型进行平移，即：

$$t_x=X_{pw2}-X'_{pw1}$$
$$t_y=Y_{pw2}-Y'_{pw1} \tag{7}$$

其中，t_x 和 t_y 分别为模型要平移的偏移量。

由式(5)、式(6)和式(7)可得：

$$t_x=(1-\alpha)\times(0.5W_{v1}-X_{pw1})$$
$$t_y=(1-\alpha)\times(0.5H_{v1}-Y_{pw1}) \tag{8}$$

3 程序实现与应用

以 VC 8.0 为开发工具，OpenGL 最小系统的搭建过程省略，本文算法的主要实现过程及代码如下。

3.1 定义变量

在应用程序的头文件里定义变量如下，这里变量的名称和算法中描述的一致。

```
//鼠标当前点坐标
CPoint  m_MousePos;
//鼠标当前点距离视景体左边缘和下边缘的距离
double  Xpv1, Ypv1;
```

3.2 分别为应用程序添加 WM_MOUSEMOVE 和 WM_MOUSEWHEEL 消息

在 WM_MOUSEMOVE 消息的响应函数中为定义的变量赋值，部分代码如下。

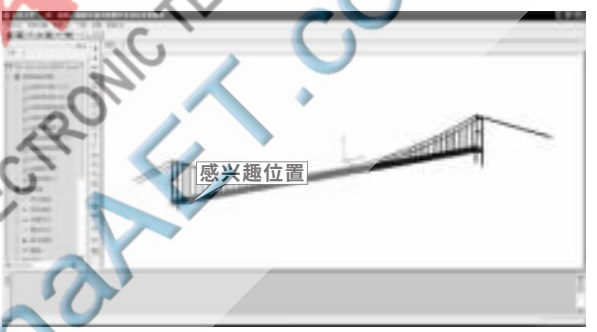
```
//获取视图窗口的尺寸
CRect  rect;
GetClientRect(&rect);
//计算 Xpv1 和 Ypv1
m_MousePos=CPoint(point.x, rect.Height()-point.y);
double  w, h;
m_Camera.get_view_rect(w, h);
Xpv1=w*m_MousePos.x*1.0/rect.Width();
Ypv1=h*m_MousePos.y*1.0/rect.Height();
```

WM_MOUSEWHEEL 消息响应函数核心代码为：

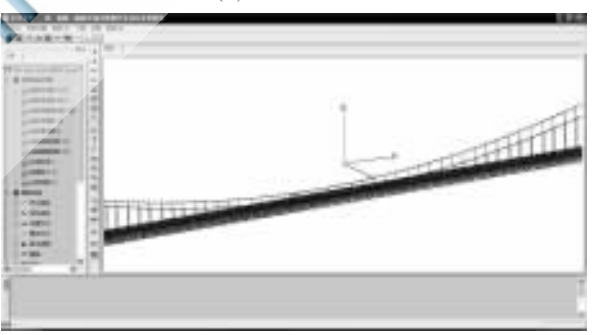
```
//缩放系数
double  a;
//视景体宽度和高度
double  w, h;
//平移量
double  t_x=0.0, t_y=0.0;
if(zDelta<0)
{
//缩小系数
a=1.1;
}
if(zDelta>0)
{
//放大系数
a=0.9;
}
//获取视景体宽度和高度
m_Camera.get_view_rect(w, h);
//根据算法计算平移量
t_x=(1-a)*(0.5*w-Xpv1);
t_y=(1-a)*(0.5*h-Ypv1);
//累计平移量
m_Camera.set_move_view(t_x, t_y);
//缩放
m_Camera.zoom(a);
//计算新 Xpv1, Ypv1
m_Camera.get_view_rect(w, h);
CRect  rect;
GetClientRect(&rect);
Xpv1=w*m_MousePos.x*1.0/rect.Width();
Ypv1=h*m_MousePos.y*1.0/rect.Height();
InvalidateRect(NULL, FALSE);                    //刷新视图
```

3.3 算法在工程计算可视化项目中的应用

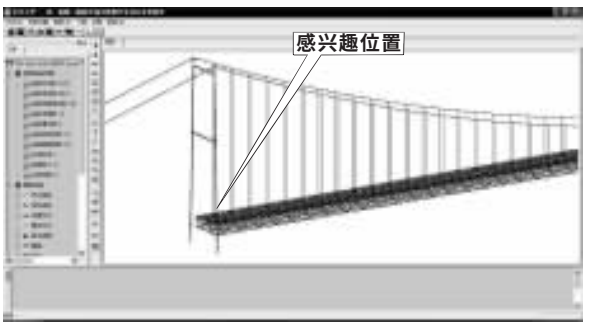
算法在多个可视化项目中得到应用，图 3 为“风、地震、随机车流与桥梁交互动力分析软件”截图。一方面它能更真实地揭示出桥梁结构在汽车车辆荷载作用下的动态受力与变形本质，另一方面又能描述出桥梁结构在地震作用下的系统影响。因为数据计算量大，单纯用 VC++ 8.0 程序计算耗时过多，容易造成计算中断，所以本软件采用 Fortran 语言作为数值计算程序主体，采用 VC++ 8.0 作为人机界面设计平台，通过混合编程实现 3 种程序设计语言的组合、相互调用、参数传递、数据结构与信息共享，从而形成统一的桥梁动力学分析可视化软件。



(a) 缩放前的模型



(b) 一般算法放大后的模型



(c) 本文算法放大后的模型

图 3 软件截图

该项目充分发挥 Fortran 语言在科学计算方面的优势及 OpenGL 在三维可视化渲染方面的优势，将工程人员多年积累的代码资源及计算结果以图形化展示，实现了桥梁结构在风、地震及随机车流作用下的动态受力与变形的可视化。

缩放是该软件的重要功能之一，工程可视化软件与一般的三维模型渲染软件的区别是：一般三维模型软件的点元、线元是基本不变的，而工程可视化软件中渲染的模型是变化的，重点是将模型在外力（车辆负荷、风力、地震等）作用下的变化展示出来，即点元、线元的相对位置是变化的。为了将精力集中在观察桥梁的结构在风、地震及随机车流作用下的动态受力与变形，在对模型进行浏览时就不能不停地执行平移放大操作。假设图 3(a)中标记的点为感兴趣点，缩放应围绕该目标点进行。图 3(b)为一般算法放大后的模型效果图，此时的目标点已经离开 Windows 视图窗口，需要不断平移再放大才能找到。图 3(c)为本文算法放大后的模型效果图，目标点依然在视图窗口内，在对模型进行浏览时不需要不停地平移即可看到模型的细节，很好地改善了用户的操作体验，让工程人员能够专注于桥梁结构在风、地震及随机车流作用下的影响。

计算机图形学给人们提供了一种直观的信息交流工具，计算机图形学已被广泛地应用于各个不同的领域，尤其是在计算机辅助设计、计算机辅助制造和科学计算可视化等应用领域。利用本文介绍的算法，用户可以方便地以某一点为中心进行自由缩放，及时捕捉工程计算的效果，减少冗余操作，改善用户操作体验，对开发图形图像应用软件有一定的帮助。

参考文献

[1] 梅章明，张秀山.基于 MFC 和 OpenGL 的喷泉模拟实现[J].微型机与应用，2012，31(14)：41-43.

[2] 何煦佳，杨荣骞，黄毅洲，等.基于 OpenGL 的医学图像实时交互处理技术[J].计算机应用与软件，2013，30(4)：48-50，64.

[3] 胡平平，刘建明，王晶杰.OpenGL 显示 3DS 模型若干问题的研究[J].工程图学学报，2010(4)：189-193.

[4] 张舒，褚艳利.GPU 高性能运算之 CUDA[M].北京：中国水利水电出版社，2009.

[5] SHREINER D，WOO M，NEIDER J，等.OpenGL 编程指南(第 4 版)[M].邓郑祥，译.北京：人民邮电出版社，2005.

[6] 崔洪斌，秦国海，常玮.利用 OpenGL 实现图形的局部放大[J].工程图学学报，2005(6)：58-61.

[7] 姜卫东.基于 OpenGL 的三维函数图象绘制[D].长春：吉林大学，2007.

[8] 彭群生，金小刚，万华根，等.计算机图形学应用基础(第 1 版)[M].北京：科学出版社，2009.

(收稿日期：2013-07-24)

作者简介：

张立成，男，1987 年生，硕士，助教，主要研究方向：虚拟现实。

张鸽，女，1987 年生，硕士，主要研究方向：图形图像处理，三维仿真。