HEROES 集成软件控制子系统的研究

顾耀林　（无锡轻工大学信息与控制工程学院　214036）
贺晓燕　（中 国 船 舶 科 学 研 究 中 心　214082）

摘要：HEROES 软件包是我国独立开发研制的大型水弹性力学工程应用集成软件系统。本文简要介绍 HEROES 软件的控制子系统，包括需求分析、系统结构、数据结构和子系统实现的若干关键技术。

关键词：软件　集成系统　控制

一、系统的需求分析

HEROES 集成软件在需求分析阶段对系统的功能和开发平台作了具体的规定。

1. 对系统功能的规定

集成系统必须具备以下功能：

(1) 采用先进的国际标准，与国际上通用的一流软件兼容。

操作系统采用 UNIX，软件集成环境选择 X—WINDOW 和 Motif，计算机图形标准使用 OpenGL，这样可保证本集成系统的起点较高，与国际软件标准接轨。

(2) 开放式系统，可扩性好。使用开放式软件基金会（Open Software Foundation）的软件标准 Motif 来开发本集成软件的窗口管理系统，从而确保系统的开放性和可扩性。从此意义上说，本集成系统是无限可扩的，可随时加入新的水弹性分析程序及其相应的处理模块。

(3) 窗口管理，鼠标驱动，界面友善。集成软件系统必须实现窗口管理，用户直接控制，进退自如。还要求界面友善，提示信息丰富，包括联机 HELP 功能。

(4) 图形丰富，具备完善的前后处理功能。本系统要有较为完备的图形处理功能，对二维曲线，三维曲面、三维光照造型及三维动画的处理，应采用国际上通用的图形标准 OpenGL。

(5) 易于移植，更新和维护。集成系统要求采用美国国家标准 C（ANSI C）编写源代码，它与 UNIX 有很好的兼容性，易于在不同的图形工作站上实现程序移植，也便于系统更新和维护。

2. 开发平台运行和环境的规定

HEROES 开发和运行的硬件平台是 SGI 图形工作站（型号不限）。其软件平台是：IRIX 操作系统（6.2 版以上），ANSI C 和 FORTRAN 77 编译系统，Motif 1.2 和 X11 R5 窗口管理系统，以及 OpenGL 和 Xlib 图形系统。

二、控制子系统的设计

HEROES 的控制子系统是四个子系统中举足轻重的一个子系统。（见图 1）控制子系统的设计包括总控逻辑的设计，控制流的组织，系统接口的控制以及控制子系统数据结构的设计。

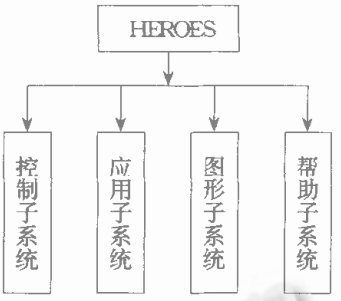

图 1　HEROES 系统结构

1. 总控逻辑的设计

用早期的 X—WINDOW 集成环境研制的软件包的控制逻辑，是基于文件窗菜单项所派生的下拉式一级菜单作总控的主线，辅之以若干少量的弹出式选项，组成总控逻辑的基本框架，目前应用广泛的结构分析软件“NASTRAN”就是其中的典型代表之一。

这样控制逻辑的缺点是，第一，其主控窗（根窗口）必须占据整屏，因为所有的控制逻辑都依赖于主控窗的一级菜单项，这样二级窗口或更低级的窗口只能覆盖于其上。第二，用户无法控制各级窗口在屏幕上的位置，即窗口不能任意拖动，不灵活。第三是窗口切换比较单调，立体感不强。

HEROES 集成软件的总控逻辑，参照了 1994 年美国新版的二个图形软件包 VolVis 2.0 和 Geomview 1.4.3 的成功经验设计而成，其特点是：

(1) 总控分级:由主控窗和各级子窗分级实现,改变过分依赖主控窗的缺点。

(2) 按钮驱动:避免使用下拉式菜单和弹出式菜单,改为三维按钮(3D Push Button)驱动的弹出式窗口控制,立体感强,使用方便。

(3) 操作方便:窗口均可由用户任意拖动,用户可将感兴趣的窗口放到感兴趣的位置上。

2. 控制流的组织

程序流机制研究的理论认为,现代计算机的驱动机制分为三类,即控制驱动,数据驱动(数据流计算机)和需求驱动。目前大多数计算机仍然是采用控制流驱动的方式。HEROES集成系统同样属于控制流驱动,只不过是其软件的控制流和水弹性应用子系统各程序的数据流是基本一致的。(见图 2)

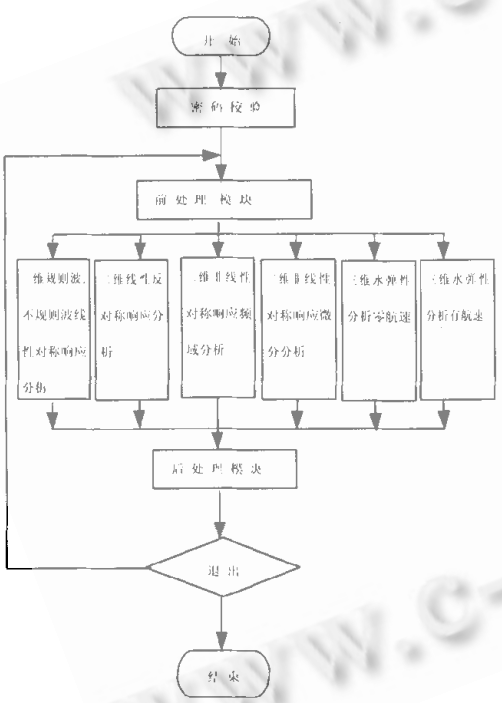

图 2　HEROES 系统的控制流

3. 系统接口的控制

控制子系统要实现系统外部和内部接口上的控制,其外部接口如图 3 所示。

其内部接口有二块,一是 FORTRAN 和 Motif 的接口,用 ANSI C 处理。

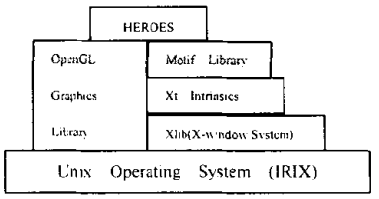

图 3　HEROES 集成系统外部接口

二是 FORTRAN 和 OpenGL 的前后处理接口,见图 4。

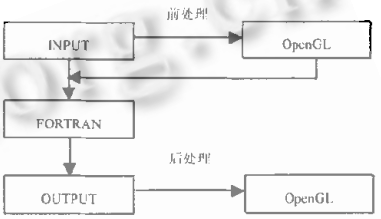

图 4　FORTRAN 和 OpenGL 的接口

4. 控制子系统的数据结构

控制功能的实现实际上是依赖于它的数据结构。这些数据结构由自行设计的数据和调用系统资源的数据结构两部分组成。

(1) HEROES 独立设计的数据结构。用于控制的数据结构主要有二大类,一类是基于对象的基本数据结构 Widget 变量,分全局变量和局部变量,例如:

```
extern Widget: toplevel, pb;
(定义全程 Widget 变量,用于窗口移动的控制)
static Widget: ldialog, form, seform;(定义局部 Widget 变量)
```

另一类主要是各种指针型变量,有文件指针、字符指针和浮点型动态数组指针。例如:FILE fp,fp1;(定义文件指针)

```
char * s1, * s2;(定义字符指针,用于数据共享)
static float * p0, * p1;(定义可调数组指针,用于数据共享)
```

(2)系统资源中用于控制的数据结构。调用系统资源中起控制作用的数据结构涉及四个部分,Xlib(X-WINDOW)、X11(XIntrinsics)、Motif 和 GL/OpenGL。

例如:

```
Display * dpy;(Xlib 数据结构,定义指向显示器的指针变量 dpy)
XtAppContext app;(X11 数据结构,定义应用程序内容的变量 app)
XmListCallbackStruct * cbs;(Motif 数据结构,定义
```

列表回调结构指针 cbs)

GLXDrawCallbackStruct * glx;(GL 数据结构,定义 GL 回调结构指针 glx)

GLwNvisual Info * vi;(OpenGL 数据结构,定义 OpenGL 可视化信息变量指针 vi)

三、控制子系统实现的关键技术

为了保证系统设计时的功能,特别是实现对每个子系统的全程闭环控制,实际上其难度还是不小的。由于篇幅关系,我们不能细述如何利用全局变量实现动态数组的共享等内容,此处仅介绍在图形子系统中如何确保控制流的畅通无阻的关键技术 - OpenGL 的混合编程模式。

1. 混合模式的功能

UNix 的 X - WINDOW 系统中,提供了三种界面平台,分三个层次。其中最底层是 X 库平台,即 Xlib(X LIBRARY)。中间一层是 Xt 内部函数平台(Xt Intrinsics)。最高层是 Motif 图形用户接口,它是基于 Xlib 和 Xt 的。这三个平台的共同点是它们都属于面向对象的程序设计范畴,其基本的数据结构都是代表对象的 Widget。

控制子系统是在上述三个平台上开发的,从本质上讲就是要实现对对象的控制。OpenGL 虽有自己的头文件和库函数,也能自己打开一个图形窗。但当启动这个图形窗后,即进入 OpenGL 环境,控制流对原有对象的控制被中断,使整个图形子系统进入失控状态中,这对集成系统而言是万万不许可的。

所以,必须使用 OpenGL 的混合模式,将控制子系统的 Widget 数据结构嵌入 OpenGL 环境中(请注意 OpenGL 本身并不支持这种数据结构),从而实现对图形窗口的设置、显示、更新、关闭等控制功能,即将 OpenGL 的运行控制加入到 HEROES 集成系统的控制中。混合模式让程序设计人员拥有 X 服务器管理的所有区域的控制权,包括窗口、菜单、滚动条等。

2. OpenGL 混合模式的实现

OpenGL 提供三种功能强大的混合模式,一是 OpenGL 和 Motif/Xt 的模式,二是 OpenGL 和 Xlib 的模式。本文只介绍前一种混合模式。

(1)头文件与图形上下文。混合模式需要特殊的头文件〈GL/GLwDrawA.h〉,同时还需要定义图形上下文 GLContext,以下给出定义的示例。

```
/* OpenGl & Motif mixed model */
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/GLwDrawA.h>
GLXcontext glx-context;
```

(2)回调函数的初始化。混合模式需要回调函数,它是实现控制流闭环必不可少的重要手段。OpenGL 混合模式的回调函数比较复杂,其回调函数的初始化更为困难,它是实现混合模式的关键,下面提供我们实际使用的初始化函数,以供参考。

```
/* The initial function of callback function for OpenGL & Motif mixed model */void ogl-init(w,client-data,call-data)
Widget w;
XtPointer client-data;
GLwDrawAreaCallbackStruct call-data;
{
    XVisualInfo * vi;
    Arg args[2];
    XtSetArg(args[0],GLwNvisualInfo,&vi);
    glx-context = glXCreateContext(XtDisplay(w),vi,0,
GL-FALSE);

    if(! glx-context)
    {
    fprintf(stderr,"Cannot create context. \n");
    }
    GLxDrawingAreaMakeCurrent(w,glx-context);
    glFlush();
}
```

四、结语

本文的工作是九五预研重点项目"水弹性力学工程应用集成软件研究"的一部分,目前已即将完成,其控制子系统已经完成。在 Unix 平台做软件的系统集成要比 PC 平台困难得多,作为三年多工作的一个小结,以此文与国内同行交流。同时,作者深切地感谢本项目负责人、中国工程院院士吴有生的组织、指导和帮助。

参考文献

[1] "OSF/Motif Programmer's Guide", Open Software Foundation, 1996

[2] "OpenGL Programming Guide", Addison- Wesley Publishing Company, 1996

[3] Rom Fosner, "OpenGL Programmintg for Windows 95 and Windows NT", Addison- Wesley Developers Press, 1997

(来稿时间:1999 年 5 月)