第 2 卷 第 8,9 期
1997 年 9 月

中国图象图形学报
Journal of Image and Graphics

Vol. 2, No. 8,9
Sep. 1997

一种新的通用可视化平台的模块设计工具

王晓梅　杨旭波　石教英

(浙江大学 CAD&CG 国家重点实验室,杭州　310027)

摘　要　介绍了一个模块自动封装工具 ME,它是通用的交互式可视化环境 GIVE 系统中的一个相对独立的部分。该系统提供了一个模块级编程环境,支持用户通过"搭积木"式的方法,从模块库中选取合适的模块,构建可视化流图。模块库的内容决定了系统的应用领域。ME 基于一套 API 库函数和模块的语法定义,通过可视的操作界面,支持用户直观交互地将自己开发的算法扩充到系统的模块库中,使系统具有良好的开放性,能适用于不同的应用领域。

关键词　科学计算可视化,可视化平台,模块级编程,模块封装

1　前　言

科学计算可视化[1](ViSC)已成为当今计算机图形学研究的主要热点方向之一,其中可视化软件平台的研究是沟通可视化理论与应用的一座桥梁,为开发可视化具体应用系统提供强有力的支持。

我们开发了以数据流为核心的可视化软件平台 GIVE[2,3](General Interactive Visualization Environment)。GIVE 提供了一个模块级编程环境,它由以下 4 个部分组成:数据流核心控制器、可视编程用户界面(VL)、模块自动封装工具(ME)和模块库。ME 是 GIVE 的一个相对独立的组成部分,完全能从 GIVE 中脱离出来而单独使用,甚至能配合其他系统一同工作。

与以往繁琐低效的手工封装模块方式相比,ME 的实现大大提高了模块开发效率。在 ME 的支持下,模块开发者在封装一个新的模块时,除了编写一个算法函数以外,勿需额外的编程工作,就能得到一个封装好的模块,因而,能迅速地封装大批模块,从而能满足应用开发的需要。

现有的几个成功的同类软件平台都采用数据流体系结构,并且都有功能较强的可视编程界面,如美国俄亥俄州立大学开发的 apE[4],Stellar Computer 公司开发的 AVS[5] 以及 SGI 公司的商品化软件 Explorer[6]。但它们的模块开发工具没有很强的模块自动封装能力,并且它们支持的数据类型有限。GIVE 系统中的 ME 不但有良好的可视交互界面,使得模块的自动封装工作快捷便利;而且具有封装数据类型的能力,因而能支持任何数据类型,使得系统具有更好的开放性。

2　模块封装原理

2.1　模块的结构与功能

在 GIVE 平台应用过程中,模块开发者常常会根据实际要求来编写一些特定的算法函数,而这些算法函数是不能直接被 GIVE 平台作为可运行的模块来使用的,因为 GIVE 平台能接受的模块都有一个标准的模式(图 1)。

从图 1 中我们可看到封装好的模块的界面,其最上部是模块名。紧邻模块名下有 4 个小图标,它们是为用户提供的对模块的直接操作界面,提供了模块的 4 个最常用的编辑和控制功能:destroy、reset、CP 板和 fire。其中 CP 板(Control Panel)具有参数控制板功能。该图标即用于弹出模块的 CP 板,供用户调整模块的参数,以控制模块的功能。图中带有箭头的小方框表示模块的数据端口,左边是入端口,右边是出端口。箭头的方向指示了数据通过端口时的

收稿日期:1996-06-11;收到修改稿日期:1997-01-20

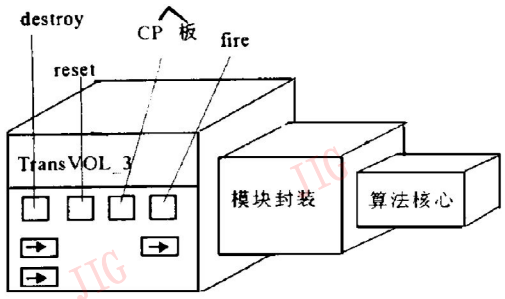

图 1　模块界面与算法核心的关系

Fig. 1　The relationship of module interface and algorithm kernel.

流动方向。且端口有其数类型。模块的入端口又有可选(optional)和必选(required)两种性质之分,当可选的入端口没有数据时,并不影响模块的可运行性,而必选性的入端口则必须获得数据,才能使模块运行。

界面上的模块与实际模块功能是用程序方式结合起来的,即用程序来实现模块的功能。当创建一个模块时,系统同时为其创建一个 UNIX 进程,该进程执行模块所对应的功能程序。

2.2　ME 的封装功能及步骤

ME 能将某种语言的函数代码封装成系统能够接受的模块,封装好的模块可执行程序形式,能直接在系统中使用。目前,ME 支持 C 语言函数代码的封装。这样,用户在 ME 的支持下,可将各学科领域的算法、自行开发的模块扩充到 GIVE 平台的模块库中,从而扩充 GIVE 平台的适用范围。

ME 的模块封装工作包括两方面内容。一方面是为模块做界面封装,即设计模块在 GIVE 中的图形表示形式,也即该模块对应的外观,这包括入端口、出端口、CP 板参量等主要部分。另一方面是为模块中用到的新的数据类型进行封装,目前系统直接支持 C 语言中的简单的数据类型,当用户在模块的入端口和出端口处要使用复杂的数据类型时,就要求用户说明数据类型的特征,来扩充新的数据类型,以便系统能够识别。

另外,ME 还为用户提供了 API 库(Application Programming Interface),用户可以直接调用 API 库函数进行模块的手工封装,而 ME 的自动封装功能也是基于 API 库来完成的。

ME 将一个算法函数封装成标准模块的过程可用如下步骤来描述:

(1)自动生成模块说明文件 xxx.spc,保存了待封装模块的模块界面细节。

(2)自动生成模块封装文件 xxx.me.c,它是在 xxx.spc 的基础上生成的,ME 采用 Unix 提供的 Lex,Yacc 编译工具将模块说明文件中的内容读入内存,据此自动生成模块封装文件。

(3)自动生成并运行 Makefile 文件,将模块封装文件与 API 库连接编译,得到可执行文件 xxx,该可执行文件就是 GIVE 平台中模块的运行实体。

值得一提的是 ME 具有一个基于 Motif 实现的良好的可视交互界面。ME 提供了一系列交互窗口,并采用鼠标作为主要的交互手段,使得用户对待封装模块的信息定义变得方便而又直观。

2.2.1　模块界面封装及 API 库函数

ME 在实现模块的自动封装过程中,最关键的工作是能够根据模块的算法函数以及待封装模块的说明信息自动产生一系列文件:模块说明文件 xxx.spc,模块封装文件 xxx.me.c 和模块可执行文件 xxx。

xxx.spc 根据模块的语法定义,保存了待封装模块的各类特性信息,它包括以下内容:

(1)xxx 函数所在的文件名;

(2)xxx 中要使用的头文件”

(3)xxx 要连接的函数库;

(4)xxx 的函数名;

(5)xxx 的函数参数表及各参数的模式和数据类型;

(6)xxx 封装后提供的入端口个数,各入端口的数据类型,其可选性(可选或必选);

(7)xxx 封装后提供的出端口个数,各出端口的数据类型;

(8)xxx 封装后提供的 CP 控制板参量个数,及各 CP 参量的图元类型,几何布局;

(9)xxx 的入端口,出端口和 CP 参量三者与函数参数表中参数之间的连接关系。

xxx.me.c 是在 xxx.spc 的基础上自动生成的。在 xxx.me.c 的设计与实现过程中,采用了大量的 API 库函数。API 库完全是由我们自己开发而成的,它为 ME 实现模块封装以及数据类型的扩充提供了许多实用的函数。

2.2.2　数据类型封装

系统中模块的入端口和出端口都有固定的数据

类型。系统目前支持的数据类型有:整数类型、字符串型、浮点型 3 种 C 语言中的基本数据类型,还支持体数据、几何数据、图象数据等几种结构数据类型。系统的设计原则是支持任意数据类型。这就需要 ME 的数据类型封装功能,它能将使用的任一种新的数据类型封装成能为系统所识别的类型。

ME 中采用 XDR 机制来实现数据类型的封装。这里 XDR(External Date Representation)是网络编程中使用的一个函数库,它的作用是将 C 语言中的结构化数据转换成能在网络上传送的连续字节流,即串行比,它提供一套基本函数库,用户只要为自己的 C 语言数据结构写出对应的 XDR 函数,XDR 机制就可以根据 XDR 函数对该种结构化数据进行串行化后在网络上传送,并在收到串行化的数据后再使用同一 XDR 函数对数据进行还原。

ME 中数据类型封装的实现可归结为以下步骤:

(1)将该数据类型定义成 C 语言的数据结构;

(2)根据 XDR 的语法规则,将该 C 语言数据结构写出其 XDR 函数,命名为 Xdr—(数据类型名);

(3)将该 XDR 函数作为 API 函数的调用参数。

3 应用实例

下面将给出采用 ME 封装的一组模块,构建成一定的流图,从而进行有限元可视化的一个应用实例。该应用实例处理的问题可描述为:将活塞受力情况的体数据读入内存,并且把体数据值映射成颜色,以显示物体表面或某一断面的数据分布。

为了构建处理该问题的流图,我们需要一组体数据处理模块,包括读入模块 ReadFEM,映射模块 MapFEM,断面映射模块 CutplaneFEM 以及显示模块 RenderFEM。在封装这组模块之前,我们首先应根据模块的算法函数来设计这些模块,例如体数据读入模块的界面可设计为:

ReadFEM 模块无入端口。有一个出端口,其数据类型为体数据类型(类型名为 VOLUME)。CP 板上有一个字符串槽型图元,对应的 CP 参量的名字为 Filename,用户可在此输入体数据所在的文件名。模块的功能:将 CP 板上指定的文件中的体数据读入到内存中,并将该体数据从出端口输出。

一旦设计好待封装模块后,我们就可借助 ME 来封装这些模块。我们再次以模块 ReadFEM 为例,封装 ReadFEM 时,根据所设计的模块界面信息,通过 ME 的图形交互界面,依次定义模块 ReadFEM 的入端口、出端口以及 CP 参量等说明信息。当定义完毕后,通过 ME 所提供的相应的菜单项就可自动生成一系列文件:模块说明文件 ReadFEM.spc,模块封装文件 ReadFEM.me.c 和可执行文件 ReadFEM,即模块在系统中的运行实体。这样模块 ReadFEM 就封装完毕了。类似地我们可设计和封装其它的模块。

当所有的模块都封装完毕后,就可构建如图 2 所示的流图。当运行此流图时,屏幕上显示了以色彩表示的活塞表面的受力分布图(图 2)。

通过此实例,我们就可看到模块级编程方法的便利性。借助 ME,大大缩短了模块封装的周期,该周期从手工封装方式下的 10 多分钟降至现在的 2~3 分钟。由此可见,较之以往的手工封装模块的方式,ME 具有更高的模块开发效率,从而使可视化软件平台的推广应用成为可能。

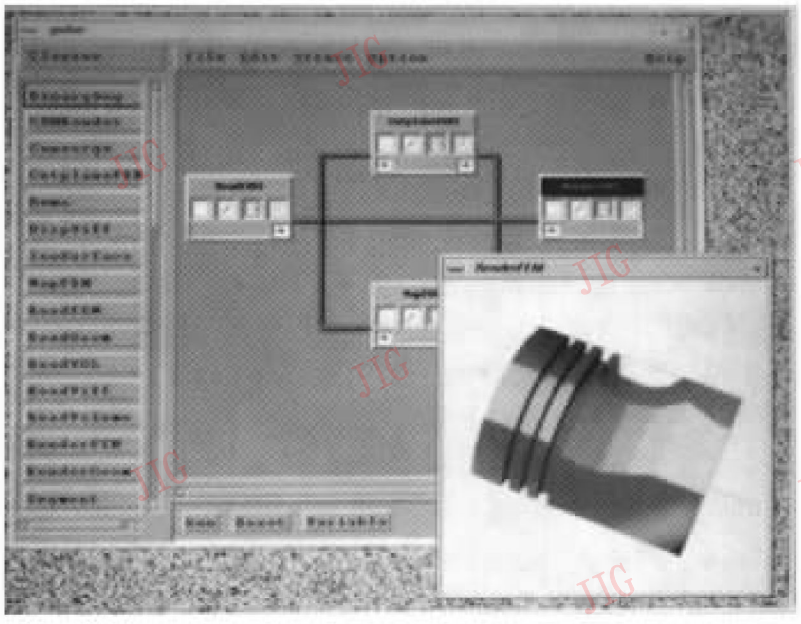

图 2 一个应用实例

Fig. 2 An example of application.

4　结束语

模块自动封装工具 ME 将模块开发者从以往繁琐低效的手工模块封装方式中解脱出来,它能高效便利地封装大量的模块,从而促进了可视化软件平台的推广应用。目前,我们借助 ME 已开发出一个丰富独特的模块库,该模块库中有映射等值面抽取,图像处理以及体绘制等功能多样的模块。今后我们将利用 ME 的自动封装功能继续扩充 GIVE 的模块库,使之能适应于更多的应用领域。

参 考 文 献

1　McCormick B H, Defanti T A, Brown M D, et al. Visualization in scientific computing. Computer Graphics, 1987, 21(5).

2　Shi Jiaoying, Cai Wenli. Object－oriented semantic data flow system for scientific visualization. IS&T/SPIE International Symposium on Electronic Imaging, 1993.

3　杨旭波,蔡文立,石教英. 通用交互式可视化环境 GIVE. 软件学报, 1996, 7(9).

4　Dyer D S. A dataflow toolkit for visualization, IEEE CG&A, 1990, 10(4).

5　Upson C, Faulhaber T Jr, Kamins D. The application visualization system: A computational environment for scientific visualization. IEEE CG & A, 1989, 9(4).

6　Silicon Graphics Inc. Explorer manuals.

王晓梅,1995 年获浙江大学计算机科学及工程学系学士学位,现为浙江大学计算机科学及工程学系硕士生,自 1993 年起进入浙江大学 CAD&CG 国家重点实验室,主要研究领域为体绘制,医学图象处理,科学计算可视化平台。

A New Module Design Tool of the General Visualization Platform

Wang Xiaomei, Yang Xubo, Shi Jiaoying

(*State Key Lab. of CAD & CG, Zhejiang University, Hangzhou* 310027)

Abstract　This paper presents an automatic module wrapping tool, Module Extender(ME). ME is a relatively independent part of the General Interactive Visualization Environment(GIVE). GIVE provides a convenient module－level programming environment. And it supports users to set up a visualization map by selecting appropriate modules from the Module Librarian. Thus, the contents of the Module Librarian decide the system's application fields. By the visual operative interface, ME supports users to directly build modules from existing algorithm codes into the Module Librarian on the basis of the API functions and the module's syntactical definition. This makes the system an opened environment and adaptable to different application fields.

Keywords　ViSC, Visualization platform, Module－level programming, Module wrapping