通用的交互式可视化环境*

杨旭波 蔡文立 石教英

(浙江大学 CAD&CG 国家重点实验室 杭州 310027)

摘要 本文介绍了通用的交互式可视化环境 GIVE(general interactive visualization environment)的设计和实现技术. GIVE 以数据流机制为核心,采用可视编程界面,为用户开发可视化应用提供了一个方便交互的模块级编程环境. 与同类软件相比,GIVE 具有如下特点:提供分支和循环控制结点,支持复杂应用程序的构建;支持应用模块和数据类型的扩充,系统具有良好的开放性;提供丰富的模块库.

关键词 科学计算可视化,数据流,平台,可视编程.

科学计算可视化(ViSC)是当今计算机图形学研究的热点之一,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了强有力支持.

可视化应用开发有其自身的特点,首先,每一用户对每一数据集的可视化要求不同,很难提供一个封闭的包罗万象的可视化平台,因此,可视化平台要求是开放型的,这不仅要求平台中模块的功能是开放的,而且也要求平台所用数据类型是开放的. 其次,可视化的过程是一种探测数据集内涵的过程,用户在该过程中会不断调整其需求,从多个侧面来分析. 为此我们在开发可视化平台 GIVE(general interactive visualization environment)[1,2]时提出了"通用的交互"的概念,它包括3个方面:与应用的交互、与模块的交互、与图形的交互. GIVE 的用户通过可视编程界面支持,采用"搭积木"式的模块级编程方法,从模块库中选择模块,连接通道,直观交互地构造出所需要的可视化应用,而无需编写任何代码;在构造和运行应用时,用户能对模块设定参数、修改模块或自己开发模块,体现了与模块的交互;模块的强大图形交互能力则让用户能直接与数据场进行各种常规的图形交互.

现有的几个成功的同类软件平台都采用数据流体系结构,并都有功能较强的可视编程界面,如 Ohio 州立大学开发的 apE[3], Stellar Computer 公司开发的 AVS[4]以及 SGI 公司的商品化软件 IRIS Explorer[5]等. 但是它们大多只支持线性流图和简单的数据回送,不能适应较为复杂的可视化应用开发,而且它们支持的数据类型有限. GIVE 也采用了数据流结

---

* 本文研究得到国家自然科学基金资助. 作者杨旭波,1971年生,博士生,主要研究领域为计算机图形学,科学可视化平台. 蔡文立,1966年生,副研究员,主要研究领域为体绘制,医学图象处理,矢量场可视化,科学可视化平台. 石教英,1937年生,教授,博士导师,浙江大学 CAD&CG 国家重点实验室主任,主要研究领域为可视化,分布式图形处理以及虚拟环境.

本文通讯联系人:杨旭波,杭州 310027,浙江大学 CAD&CG 国家重点实验室

本文 1996-03-04 收到修改稿

构,并引入了分支和循环结构,能构造复杂的应用.GIVE中的数据类型可以任意扩充,因而开放性更好.

1 概念与术语

1.1 数据流模型

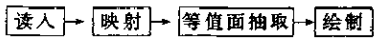

图1 数据流例子

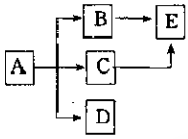

图2 流图例示

在数据流模型中,一个应用程序是由多个功能独立的模块通过数据流通道互联构成,每个模块带有若干输入口和输出口,模块的功能是将其输入口的数据进行变换后形成数据流送给与之相连的其它模块.与传统的控制流思想相比,数据流的思想非常适于并行和模块级编程,也非常适于用图示的方式来描述和分析应用程序,因此现有的可视化平台都选用了数据流模型作为系统的体系结构,平台的可视编程界面也是建立于这一模型之上的,图1就是一个典型的数据流应用例子,数据在经过一个个模块的变换后,最后成为几何数据而被绘制显示.

1.2 结点、管道、流图

结点是相对独立的具有一定数据变换功能的模块,有入端口和出端口,管道是结点间传递数据的通道.GIVE中还提供具有分支和循环功能的控制结点,控制结点只控制数据的流向,不进行数据变换.流图是由结点和管道组成的有向图,一个流图对应一个具体的可视化应用实例,它表示了各模块间的数据传递和依赖关系.GIVE中的流图可以表示为有向图$G=\langle V,E,F,C,S\rangle$,其中$V$是结点集,包括控制结点在内的所有结点;$E$是边集,即流图中的管道;$F$是映射关系,$F:E\rightarrow V\times V$;$C$是约束条件集,它规定了$G$中各边的合法性,即用户在创建管道时应遵循的规则;$S$是状态集,表示每个结点的当前状态.若在图$G$中存在从结点$V_1$到结点$V_2$的路径,则称$G$中$V_1$可达$V_2$,记为$access(V_1,V_2)$.在$G$中可达$V$的所有结点的集合称为$V$的上游,记为$upstream(V,G)=\{V'|access(V',V)\}$,其中$V'$称为$V$的上游结点.在$G$中$V$可达的所有结点集合称为$V$的下游,记为$downstream(V,G)=\{V'|access(V,V')\}$,其中$V'$称为$V$的下游结点.

1.3 驱动方式(点火方式)

GIVE中流图结点的驱动方式有2种:(1)在数据驱动(Data Driven)方式下,当某结点所有入端口数据到达后,该结点就被自动点火执行,执行完后,其出端口数据由管道传递到所有依赖该结点的入端口.如图2所示,当源结点A点火后,$downstream(A,G)$中所有结点,即B,C,D,E均要执行,在此方式下,常会导致不必要的结点点火;(2)在命令驱动(Demand Driven)方式下,流图中除了有数据流外,还存在一股与数据流逆向而行的命令流,它源于用户对数据结果的请求.只有当一结点的所有入端口数据均到达,且有命令流到达该结点时,该结点才被点火执行.如图2所示,若用户需要结点E的执行结果,则发一请求,命令流由E出发,经B,C到A,则A,B,C,E4结点被点火执行,而D结点不会被点火,即只有$upstream(E,G)$中的结点被点火.因此其执行效率较好.

2　总体结构和设计

2.1　总体结构

数据流可视化软件平台的关键技术包括数据流控制核心、流图编辑界面、模块封装工具、数据交换工具、模块库及其管理工具. GIVE 的结构框图如图 3 所示,主要包括 4 个部分:(1)数据流核心控制器实现数据流核心控制功能,包括流图构造、流图运行、结点的并行控制、结点间数据传递等功能的实现;(2)可视编程界面 GIVE/VL 以图形界面表示流图、结点和管道,支持用户直观交互地编辑和运行流图,并将用户的操作反映到数据流核心控制器;(3)模块封装工具 GIVE/ME 支持用户将新模块或新数据类型进行封装后扩充到 GIVE 环境中,包括模块界面封装和模块数据封装的实现;(4)模块库为用户开发可视化应用提供各类可视化技术的数据处理模块,GIVE 中目前主要实现了 2 类模块:有限元处理模块和图象处理可视化模块.

2.2　流图运行规则

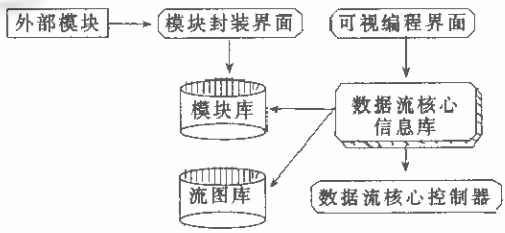

图3 GIVE结构框图

流图中的结点有 3 种状态: UPDATED, RUNNING 和 FIRED, 其中 UPDATED 表示此结点的运行结果已失效,需重新计算,对于有出端口的结点,还意味着其出端口当前数据是无效的,不能在管道上传送;RUNNING 表示此结点正在运行;FIRED 表示此结点已运行完毕,对于有出端口的结点,意味着其出端口的当前数据是有效的,可在管道上传送. 一个结点是可运行结点的充要条件是:(1)其状态为 UPDATED;(2)其所有必选的入端口均有数据;(3)其所有连有管道的可选的入端口均有数据. 可运行结点是否真正运行,还要取决于流图的驱动方式.

2.3　控制结点的语义

GIVE 引入了 4 种控制结点,以下简要介绍其在流图中的语义和运行方式.

1) IF－THEN－ELSE 结点提供了数据流的分支功能,它有 1 个入端口(I 端口)和 2 个出端口(T 端口和 F 端口). 当该结点运行时,首先计算条件表达式的值,①如为真,则 I 端口的数据从 T 端口传出,由此形成的数据流所流经的路径称为该结点的真分支. ②如为假,则 I 端口的数据从 F 端口传出,由此形成的数据流所流经的路径称为该结点的假分支.

2) MERGE 结点用于合并流图中 2 个数据流路径,常用于合并 IF－THEN－ELSE 结点的 2 个分支,它有 2 个入端口(I_1 端口和 I_2 端口)和 1 个出端口(O 端口). 通常 MERGE 结点的 2 个入端口只有 1 个会有数据达到,此时,MERGE 结点便将此入端口数据从其出端口流出. 如果 2 个入端口同时有数据达到,就强制选择 I_1 端口的数据.

3) COUNT－LOOP 结点用于提供简单的增量计数循环,其结构如图 4. COUNT－LOOP 结点有 1 个循环变量 *Loopvar* 和 3 个控制参量:初值(*Initial*)、增值(*Increment*)和终值(*Final*). 结点运行时,将 *Initial* 赋给 *Loopvar*,然后测试条件表达式 $Loopvar \leqslant Final$, ① 如为真,I 端口数据传至 C 端口,执行循环;② 如为假,I 端口数据传至 E 端口,结束循环. 一次循环结束后,循环变量递增,$Loopvar = Loopvar + Increment$,再测试条件表达式

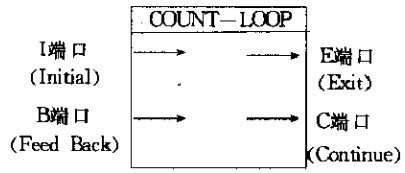

图4 COUNT—LOOP结构

Loopvar≤*Final*. 首次循环从I端口取数据,以后的循环则从B端口取数据.每次循环,只有当循环体中所有结点均运行完后,才能开始下一次循环,数据流可从循环体中任一结点出端口回送到COUNT—LOOP结点的B端口.循环体可有2种连接方式,如图5所示.在(a)中,每次循环,B端口的数据与I端口是同一数据,在(b)中每次循环B端口都产生新数据,并将新数据用于下一次循环.

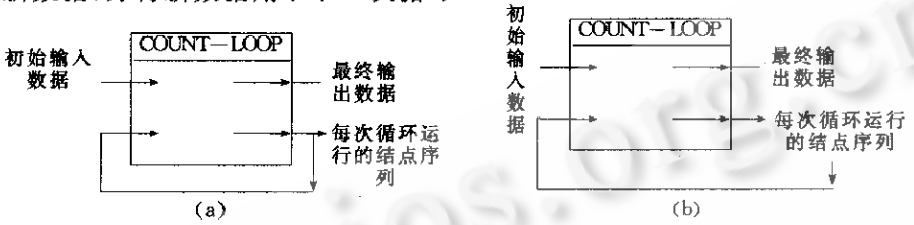

图5 COUNT—LOOP的2种连接方式

4) WHILE—LOOP结点用于提供简单的条件循环,其结构与COUNT—LOOP结点类似.不同之处是它有3个表达式:初始表达式、修整表达式和测试表达式,表达式中的变量值可被循环体中结点修改,以达到循环控制的目的.

3 实现技术

3.1 主要数据结构

GIVE的数据流核心主要利用模块表、结点表和管道表这3个表来记录流图编辑和运行所用信息.(1)模块表用于保存系统模块库中每一模块的接口信息(如模块名、执行文件名、端口个数与数据类型、控制参量的说明等),每一表项对应一个模块;(2)结点表用于保存流图编辑区中当前所有结点的信息(如结点名、结点的几何位置、结点对应的进程、结点状态、结点所关联的管道等),每一表项对应一个结点;(3)管道表用于保存流图编辑区中当前所有管道的信息(如管道所关联的始结点和终结点),每一表项对应一条管道.

实际上,模块表保存了每个模块的所有实例结点的公共信息和静态信息,这些信息最初保存在.spc文件中,系统启动后将其装入内存的模块表中,在系统运行期间模块表的内容始终不变.结点表中保存了每个结点的动态信息和私有信息,当系统创建某模块的结点实例时,就在结点表中新增一项,并根据模块表的内容填充之.管道表保存了所有结点之间的数据联系,用户删除或增加一管道,系统就在该表中删除或增加一项.结点表和管道表动态保存着流图编辑区中当前流图的拓扑结构及其运行控制信息.当前流图按流图的语法定义可保存在.grf文件中,同样.grf文件也能被随时装入到当前流图编辑区.

有了这3个表就能完成流图编辑,流图运行则还依赖于另一重要数据结构—运行表:

```
struct runTerm {
    NODEp    NodeP;      //结点指针
    int      DepNum;     //结点还需要几个结点的运行结果
    NODEp    DepNodeP[DEPNODES];   //与该结点入端口相连的所有结点
    Bool     DepFlagA[DEPNODES];   //标记DepNodeP表中的结点是否已运行好
};
```

3.2 数据流控制算法

数据流控制核心主要是 2 个以运行表为核心的算法:调度算法(Scheduler)和分派算法(Dispatcher).调度算法根据流图的不同驱动方式,将用户请求运行的结点以及这些结点对其它结点的依赖情况组建在运行表中.分派算法则反复搜索运行表中各结点,找出当前可运行结点,将这些结点的对应进程激活,并将这些可运行结点从当前运行表中删去,当某结点运行完后,根据结点的运行情况,相应地修改运行表的内容,以便找出新生成的可运行结点,直至运行表为空,或运行出现异常.运行表中所有可运行结点均可并行运行.

GIVE 中的每个结点用一个独立的 UNIX 进程来实现,并采用信号机制和消息机制来控制其运行.其中使用了 2 个消息队列,1 个用于 GIVE 向结点子进程发送点火消息 FIRE－MSG,另 1 个用于结点子进程向 GIVE 报告其运行结果,结点子进程返回 3 种消息:FIRED－MSG(正常返回),ERROR－MSG(报错),EXCEPTION－MSG(异常).结点间的数据传递通过共享内存来实现.

3.3 模块界面封装的实现

GIVE 包括一个模块自动封装工具 ME(module extender),它提供可视的操作界面,支持用户直观交互地将自己开发的 C 语言模块扩充到 GIVE 的模块库之中.用户通过 ME 说明模块的 C 语言函数接口、模块的入端口、出端口和控制参数等信息,ME 将为该新模块生成.spc,Makefile 和可执行文件等一系列文件,GIVE 根据该模块的.spc 文件就可载入该模块到模块表中,并创建子进程运行其可执行文件.除了 ME 外,GIVE 还提供了一套库函数,支持用户自己编程对模块进行手工封装.

3.4 数据类型及封装

GIVE 系统中结点的入端口和出端口都有固定的数据类型.GIVE 目前支持整数、字符串、浮点数 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.GIVE 的设计原则是支持任意数据类型,即任意的 C 语言数据类型只要经过一定的封装就可在 GIVE 中使用,这是通过 XDR(external data representation)来实现的.XDR 是网络编程中使用的一个函数库,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.

4 结束语

经过近 3 年的努力,GIVE 系统已完成并通过鉴定,是国内第 1 个拥有自主版权的可视化软件平台.我们在 GIVE 上与浙江大学土木系合作开发了“大坝优化结果显示”的可视化应用实例,与杭州汽车发动机厂合作开发了“X6130 发动机机身模态分析”的可视化应用实例,均取得满意效果.GIVE 目前运行于 SGI 公司的 IRIS 系列工作站上,软件环境为 UNIX 操作系统、Motif 窗口系统和 GL 图形库.与国外同类软件相比,GIVE 的主要特点在于:

(1)在采用数据流结构的同时,引入了 IF－THEN－ELSE,COUNT－LOOP,WHILE－LOOP 等控制结点,使流图的内容更丰富,不仅能构造简单的应用,而且能构造一些较为复杂的应用实例,目前只有最新版本的 AVS 5.0 中才包括有控制结点.

(2)开放式的工作方式,支持用户的进一步开发,用户可以将自己开发的模块经过封装

直接放入到模块库中,数据流中流动的数据类型不作任何约束,可以是各种简单的和复杂的数据类型.

(3)丰富独特的模块库,GIVE 的模块库中不仅有常用的映射、等值面抽取、图象处理等模块,而且拥有我们自己独特的细节加强的体绘制算法[6],该算法所得到之细节远远超过了现有的算法.

参考文献

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

2 杨旭波.科学计算可视化平台的可视编程界面的设计与实现[硕士论文].浙江大学,1995.

3 Dyer D S. A data flow toolkit for visualization. IEEE Computer Graphics and Applications, 1990,**10**(4).

4 Upson C *et al*. The application visualization system: a computational environment for scientific visualization. IEEE Computer Graphics and Applications, 1989,**9**(4).

5 Silicon Graphics Inc. Explorer manuals.

6 蔡文立,石教英.基于输运方程的混合式体绘制模型.计算机学报,1995,**18**(8).

GENERAL INTERACTIVE VISUALIZATION ENVIRONMENT

Yang Xubo Cai Wenli Shi Jiaoying

(*State Key Laboratory of CAD & CG Zhejiang University Hangzhou* 310027)

Abstract This paper presents a GIVE(general interactive visualization environment), which supports the development of ViSC applications. GIVE owns a data flow kernel, a visual programming interface and extensible module libraries. These provide users a convenient interactive module—level programming environment. GIVE has these features: it introduces control nodes, e.g. IF—THEN—ELSE, COUNT—LOOP, WHILE—LOOP, to support conditional alternative structure and loop structure for building complex ViSC application; it is an opened environment, the module libraries and data types are both extensible; the module libraries contain a lot of algorithms.

Key words ViSC, data flow, platform, visual programming.