基于 X 窗口的图形用户接口编辑器的设计与实现*

吴庆炜　　蔡士杰

(中国科学院计算技术研究所 CAD 开放实验室,北京 100080)
(南京大学计算机科学与技术系,南京 210093)

摘要　本文介绍了一个用于描述、生成用户接口的具有彩排功能的图形用户接口编辑器的设计思想和实现方法.叙述了对话控制树和输入数据队列在用户接口和应用程序的分离中所起的作用.最后还讨论了一种基于事件驱动机制的对话控制字解释程序.

关键词　对话控制字,交互件,输入数据队列,彩排,多线.

在计算机应用系统中,用户接口是决定该系统能否被用户接受,能否保证用户工作效率的关键部分之一.设计一个好的用户接口,除了要有程序设计技能外,设计者还必须具有对计算机各种硬设备细节以及有关软件环境有较好的了解,一般的应用程序员难以开发出高质量的用户接口[1].

X 窗口系统是一个工作站上的窗口系统工业标准,由于它具有很强的窗口功能以及事件驱动和面向目标机制,因而受到普遍欢迎.但如果直接使用 X 的库(Xlib)函数,则有相当难度(因为它约有 300 个函数),即使使用 X 工具包(Xtoolkit)也会因必须了解其复杂的调用格式而困难重重,例如设计一个菜单,设计者必须指出它的位置、风格、菜单键的布局和颜色等诸多属性[2].

本文介绍一个由作者开发的图形用户接口编辑器(GUIEditor),它旨在帮助用户接口设计者利用直接的图形交互描述用户接口中的对象并组织成对话控制字来生成具有交互、图形、直接操纵等风格的用户接口,并最终生成 C 语言代码.它引入了一组输入数据队列 IDA(Input Data Array),作为用户接口和应用程序的联系桥梁,以保证实现两者较好的分离.该编辑器具有彩排(Rehearse)功能,支持多线序列(Multi－threaded),缺省及对话的回退(Undo)等功能,提供集成的嵌入式帮助系统.

1　GUIEditor 的设计思想

1.1　基于 X 窗口系统,充分利用 X 的优点

* 本文 1993－11－13 收到,1994－02－21 定稿

本文得到中国科学院计算技术研究所 CAD 开放实验室的资助.作者吴庆炜,1968 年生,硕士研究生,主要研究领域为计算机图形学,人机交互.蔡士杰,1944 年生,教授,主要研究领域为计算机图形学,CAD,人机交互.

本文通讯联系人:蔡士杰,南京 210093,南京大学计算机科学与技术系

X窗口是一个操作系统独立及网络透明的系统,利用把窗口管理程序及窗口服务器分开而将不同的硬件连接起来.用户界面只作X窗口调用,不存在对任何操作系统的依赖,故以往用户被开发图形及窗口应用的不可移植所困扰,也因X的设备独立性而得以解决.X的另一重要特点是X的开发者为X确立的"只提供机制不提供策略(风格)"的原则.不同于MS Windows, Macintosh等窗口系统仅支持某种显示风格的界面,X的基本窗口系统提供了一个灵活的、充分的窗口基本操作的集合,而把与界面显示风格有关的实现留给它的高层机制和应用本身去完成,X的这一特点是它被广泛采纳的一个重要原因.

1.2 让用户接口设计者免除程序设计工作,对用户接口描述直观化

用户接口包括外部接口和内部接口,其外部接口应按用户要求来开发,GUIEditor设计的指导思想就是尽量把程序设计问题隐藏在内部,给出一个环境,设计者直接以用户要求来和系统交互,用以设计具有如下特征的目标用户接口:直接操纵,多线序列,支持缺省以及对话的取消,并提供集成的嵌入式帮助系统.这样设计外部接口,可以大大提高设计效率和质量.其内部接口的产生缘于对话部件和应用程序的分离,它是联系UI和应用程序的桥梁,因而在运行时对用户是透明的,仅在设计时由设计者描述.

1.3 提供直接操纵式、可视的界面,能快速检查设计效果,支持快速原型

作为一个图形描述的UIMS,GUIEditor的接口必须是直接操纵式的,描述过程必须是可见的,设计者面对的是一个逐步完善的最终用户接口的各个片断,而且可以直接修改它.GUIEditor不仅易于学习和使用,不需要设计者具备程序设计的技巧.另外,它还支持快速原型化,在应用程序尚未编好时也能运行生成的用户接口,以便尽快地从最终用户得到反馈,用以修改和完善用户接口的设计.

1.4 GUIEditor的结构

GUIEditor由描述修改工具、彩排器(Rehearser)以及生成器(Generator)等三部分组成.描述和修改工具用于描述和修改对话序列,以及相关的菜单、对话框、警告信息、帮助信息和应用程序接口;彩排器可以在产生接口代码之前根据描述来"运行"接口;生成器生成接口的C代码,该代码可以与应用程序经编译、链接得到最终的交互式应用程序.

2 用户接口交互件的描述

GUIEditor把交互式应用系统中用于用户输入的部件称为交互件(Interactor),它可以是一些交互技术部件,例如定位技术用于输入一个点;也可以是一些菜单、对话框等用来输入代表各种不同含义操作的数据.

2.1 菜单的描述及内部表示

菜单是用户接口所不可缺少的交互部件,菜单描述用综合性对话框以及一个反映设计结果的动态变化的菜单窗口来进行(图1).设计者可在其上描述不同风格的目标用户接口菜单类型的对象,如水平的,垂直的,以及二维形式的菜单,单层或多层次的菜单等.

在应用系统中,菜单起着控制对话分支的作用,一个菜单项相当于一条命令,对应着一连串的动作,按照菜单控制对话分支的不同,我们把其属性"Attribute"项分成以下三类:

(1)等待型菜单:当对话进行到要选择该类菜单时,应用系统必须等待用户对菜单的选择操作.

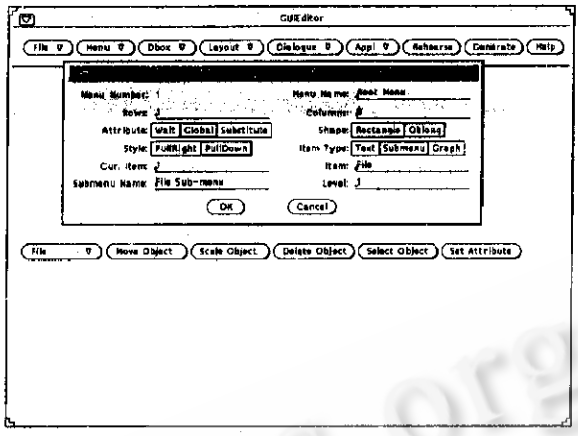

图 1 用 GUIEditor 描述菜单

(2)全局型菜单:全局型菜单在任何时刻均可响应用户的选择.即当对话进行到需选择此类菜单时,应用系统并不仅仅等待用户对该菜单的选择,而是在等待的同时可响应另外的操作.为了实现这一目标,在定义对话控制字时,全局型菜单所对应的分支要比菜单叶子数多 1,如果用户选择某一菜单项,对话便跳转到相应的分支.全局型菜单在设计应用系统时很有用,不使用时它被旁路,使用时便可轻易引出另一线.一般它可作为整棵对话树的根分支源,这样它在任何时候都是活动的.

(3)替换型菜单:替换型菜单总有一个当前分支.当对话进行到需选择此类菜单时,应用系统也不等待用户选择,而是直接跳转到当前分支,例如 GUIEditor 一开始定义为第一分支.执行完当前分支的所有对话后,回到当前分支的源结点处循环执行.如果接收到用户选择该菜单第 n 菜单项的事件,在确认不存在多线的情况下,第 n 分支便成为新的当前分支,对话转移到该分支继续执行.替换型菜单大大方便了用户操作,使系统可循环执行相同的命令操作而不用用户输入相同的命令.

2.2　对话框的描述及表示

对话框一般用于各种输入及设置,因而也是用户接口的一重要组成部分.一个对话框一般由几个子对话框组成,它们可以属于不同的类型(图 2),在运行时,对话框的对话结束后将返回所定义的各种类型的数据.设计者可以从类型列表中选择所要描述的子对话类型,它的输入数据类型,缺省值和范围,并在提示信息的指导下定位子对话框.

如果设计者不欲把某个子对话框的用户输入仅仅作为一个过程的参数使用,他可以定义一个全局变量来存储该子对话的用户输入,该全局变量就可被应用系统的各处所使用.这可以减少应用程序的参数个数.

2.3　交互技术库及其使用

交互技术是帮助用户输入数据的又一交互部件,GUIEditor 事先定义好一些交互技术,并把它们组织成一个交互技术库给用户接口设计者选择.设计者也可以自行定义交互技术.在使用交互技术时,设计者必须定义相应的语义反馈窗口.当对话进行到该交互技术时,便会在相应的反馈窗口出现提示光标,用户便可在窗口操作,结束后提示光标消失.当应用程

序有反馈时,交互部件负责在该反馈窗口中出现反馈信息.

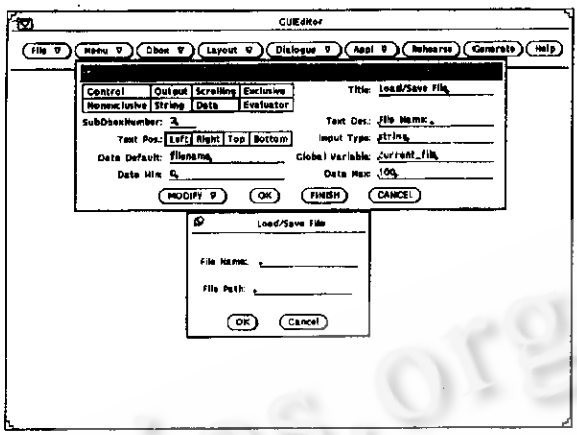

图 2 用 GUIEditor 描述对话框

3 事件驱动和对话控制树

X 窗口系统提供了事件驱动机制,在一个事件驱动型程序里,程序将控制交给用户,用户利用一系列事件驱动程序的行为.事件驱动型编程的特点是事件可以在任何时候按任何方式进入,这就使大多数事件驱动程序内核处于一个中心循环中,它每接收一个事件,便以某种方式对其做出反应.

3.1 用户接口人机交互序列控制——对话控制

GUIEditor 为了描述用户接口的人机交互,引入了对话控制字的概念[3]并将它与事件驱动机制相结合.用户和用户接口之间的对话一般是一树型结构,GUIEditor 使用对话树来表示对话序列,对话树中的每一节点即对话控制字包含许多域用以表示相关的对话的内容,这些域实现对话的外部控制,内部控制和序列控制.外部控制域处理菜单、对话框、交互技术、帮助信息和出错信息等交互件(图 3).内部控制域指示当一条命令的所有参量被输入以后调用哪一应用过程,并且指出语义反馈的窗口.序列控制域指示当前对话与前后对话的关系,包括后续对话的指针,兄弟对话的指针和父对话的指针.

有 5 种可能的序列类型:根、顺序、结果、循环、分支.

3.2 基于事件驱动的对话树控制方式

建立了对话控制树,GUIEditor 把事件驱动型程序的控制权交给了对话控制字.如果当前控制字有菜单对话,应用系统便依三种类型的菜单做不同的处理,并准备接收菜单选择事件.若当前控制字有对话框操作,系统就弹出对话框等待用户在对话框上的操作;若当前控制字要交互技术操作,系统便启动交互技术等待用户输入数据;最后若控制字要调用应用过程,这意味着用户经过一系列的操作,系统调用相应的应用过程,完成了事件驱动型程序到控制字解释程序的转换.

解释完当前对话控制字后,若当前控制字是顺序类,便开始后续对话;若当前控制字是循环类,便重新开始当前对话直到接收到一个循环结束事件,才开始后续对话;若当前控制字是分支类,系统根据从该分支的源所输入和输出的数据或控制信息跳转到相应的分支上

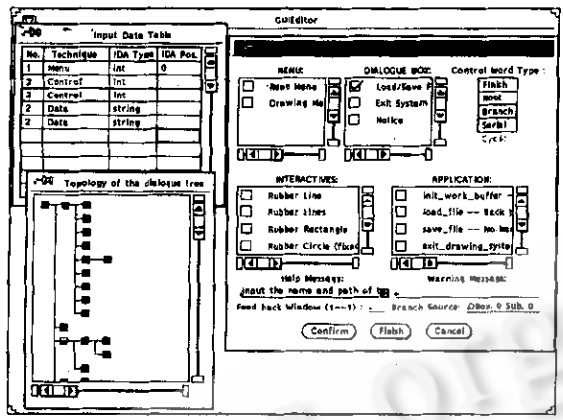

图 3 对话控制字的描述

进行下一对话;若当前控制字是结束类,系统查看是否有被中断的命令,若有按照"后进先出"的原则恢复被中断命令的断点环境继续对话,否则控制回退到最近的替换型菜单对话处,循环执行相同的命令,若无这样的替换型菜单对话,则回退到根执行新的命令.

3.3 多线控制及其实现

一般说来,用户和用户接口之间的对话不会如树型结构那么简单,经常需要两条或多条相关的命令互相穿插以获得满意的结果,这就是 GUIEditor 所提供的多线序列. 多线序列允许用户暂停一命令的描述过程,而转去执行另一条命令. 例如在画线段时输入第一点之后,用户选择了属性设置命令,系统则将当前输入数据及索引,对话控制字地址等所有相关状态保存起来,随后开始属性设置命令,当一个结束型对话控制字解释完之后,控制字解释器按"后进先出"的原则弹出堆栈中信息,并恢复中断现场,继续被中断的对话. 多线控制的引入使得控制字解释器在"感觉"上与事件驱动型程序无二样.

4 IDA(输入数据队列)——内部接口的桥梁

GUIEditor 负责生成应用系统的界面部分,由于界面部分与应用部分分别进行设计和编程,这两者之间的接口——内部接口的描述和管理也就显得异常关键. 内部接口在程序运行时是对用户透明的,但在设计过程中是连接外部接口和应用过程的桥梁.

4.1 用户接口变量及其管理——输入数据队列 IDA

在一般过程设计中,程序员大都喜欢定义许多变量来存放用户从各交互件输入的数据,供以后使用,这些分散的变量给内部接口的描述及生成带来很大困难. 为了使内部接口描述规范化,GUIEditor 定义了许多类型的数组来存放用户利用交互件输入的数据,每一类型的数组有一索引指针来指出下一数据存放的位置,一些交互件的输出和应用过程的返回值也存于这些输入数据队列 IDA 中. 归纳起来,IDA 有如下功用:(1)存储用户从各交互件输入的数据;(2)存储某些交互件的输出和应用过程的返回;(3)存放提供给应用过程的参数;(4)支持彩排及其多线控制的功能.

4.2 内部接口的描述

在描述内部接口之前,先得完成应用过程的描述. 应用过程是一个应用系统的核心,用

户接口只是为其准备参数,没有应用过程也就不需要用户接口.为了描述一个应用过程,GUIEditor 提供一个综合性对话框和一个反映设计结果的应用过程定义窗口来和设计者交互.设计者可以修改初始过程名和返回类型,并逐个键入各参数名及类型(图 4),由于参数名只是内部使用的一个标识符,所以如果不愿专门定义其名称可用它的缺省名.如果该过程有返回值,在它被调用时 IDA 会提供一位置存放该返回值供后使用.

对于每一参数,用图 4 中"Data Transmit Direction"来指明该参数和 IDA 间的关系.

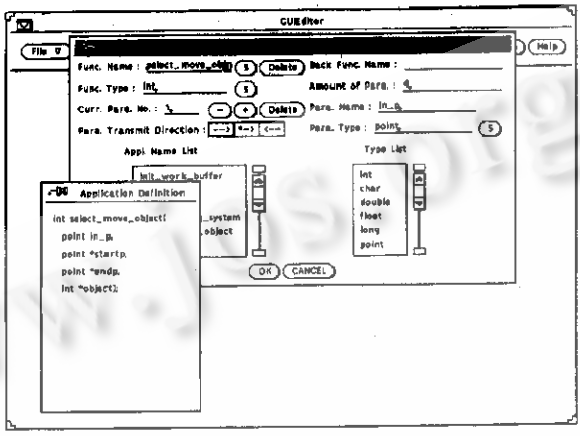

图 4 应用过程的描述

(1)"→"说明该参数仅从 IDA 接收数据;

(2)"*→"说明该参数是一指针,它从 IDA 中接收数据,过程执行完后,把该参数的返回值存储到 IDA 中调用前的位置;

(3)"←"说明该参数也是一指针,但不同于第(2)类参数,当过程执行结束后,IDA 提供一个位置来存放该参数的返回值.

若定义一个对话控制字调用应用过程,设计者可以从所有登记的过程名列表之中选择此过程,而后定义参数传递.对于自动生成用户接口代码,参数传递的定义是非常重要的一个环节,有了 IDA,参数的表示有了规范化.一个应用过程的参数来源于两方面,一是用户从交互件的输入;二是其他过程的输出,如过程的返回值以及参数返回值.在描述参数传递时,GUIEditor 显示一张输入数据表 IDT,该表记录了从根对话到当前对话的所有参数来源,即所有交互件输入和应用过程输出的数据类型及其在 IDA 中的位置.在定义应用过程的参数传递时,设计者只要定义该过程的返回值和各参数与输入数据表 IDT 中的各数据的配对关系(图 5).随后 GUIEditor 就可以产生一个完整的应用程序调用语句,供接口程序使用.例如,图 5 中的描述将产生以下调用语句:int_array[1]=select_move_object(point_array[0],&point_array[1],&point_array[2],&int_array[2]);该函数是选择一待移动对象,point_array[0]由用户利用交互技术部件输入作为对象内任一点,返回对象标识符到 int_array[2]中和选择成功与否到 int_array[1] 中,返回待移动区域的对角两点到 point_array[1]和 point_array[2]之中,供以后的程序使用.这里,IDA 成功地进行了内部控制,为应用程序准备了数据,也为设计者提供了引用数据和存储数据的场所.

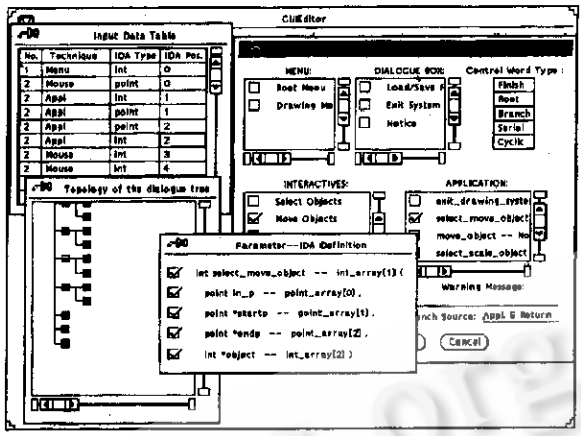

图 5 IDA 和应用过程返回值及各参数配对关系的描述

4.3 回调函数的简化

写过窗口管理程序的程序员都知道,创建一个交互构件 Widget 都要配上相应的回调函数 Call－back,用以指出当该 Widget 被用户操作时所要进行的动作,这是事件驱动型程序的最显著特征.一个中等复杂的程序用户接口经常包含成百上千的 Widget,程序员势必提供大量的 Call－back 函数.而 GUIEditor 引入了 IDA,使得为每个 Widget 写 Call－back 的繁琐之事消失了.有了 IDA,程序员所要使用的变量和数据得到了规范化,许多互相之间有联系的 Call－back 可以结合成一个应用过程,而 IDA 数据的得到可由 GUIEditor 生成器实现,从而使 Call－back 的数量大为减少,例如互斥型子对话框"Exclusive",每一选项都应有一个 Call－back,但在 GUIEditor 中,设计者可以把用户选择存入 IDA,并把它作为一个参数传递给应用过程,由应用过程对其进行分别处理,从而化简了设计,并且使用户接口和应用程序得以分离.

5 彩排——用户接口的快速演示

5.1 GUIEditor 的彩排功能及实现

一个完美的用户接口生成系统都必须提供演示已生成接口的功能,GUIEditor 的彩排器(Rehearser)便是为此而存在的.在彩排用户接口之前,为了使运行时有一美好的视感,GUIEditor 提供了屏幕布置的机制.任何屏幕的布置都有一侧重点,GUIEditor 是用来生成图形用户接口,应用程序的语义反馈窗口和交互技术的作图环境便不可缺少,为了支持多线操作,GUIEditor 要求全局型和替换型菜单必须永驻屏幕,设计者可以定义屏幕菜单的位置,背景醒目色等.

GUIEditor 中彩排的实现是将应用程序调用模拟器扩充成对话控制字解释器,按照对话树的结构从根开始解释执行,根据设计者和用户接口的交互控制彩排走向.由于应用过程尚未存在,当其被定义调用时,彩排器列出过程名和传递给应用过程的参数值,对于需返回的参数,可让用户指定它的值以便使交互完整进行下去.例如要选择一个待移动的对象,GUIEditor 的交互技术被启动,输入一点后调用应用过程,而此时应用过程尚未存在,用户便可在一对话框里输入欲返回的区域始末点及成功与否、对象标识符等参数.若返回选择成

功,屏幕便会出现一个选中区域,而后继续执行成功所对应的分枝;若选择不成功,便无反馈区域,解释器转向不成功的分枝.

5.2 GUIEditor的彩排修改功能

为了方便设计者,GUIEditor彩排器提供了在不影响对话树内容的前提下的修改功能.在彩排过程中,设计者可以重新布置屏幕、菜单、对话框和一些文字描述,包括反馈窗口大小、位置、背景色,菜单位置和风格等,以获得更愉悦的视觉效果,任何时刻均可对GUIEditor所提供的嵌入式帮助系统进行修改,使得用户操作更加易学方便.当用户的输入数据有误时,GUIEditor的彩排器除了自动置成设计者所定义的缺省值外,还弹出警告信息,设计者也可对之进行修改.所有的修改均即时存入相应的文件中.

5.3 彩排机制及控制字解释程序与事件驱动型程序的比较

为了提高设计质量,使生成的用户接口不至因用户的使用不当而出现异常,GUIEditor对彩排提供了检测机制.彩排器对描述生成的各文件及对话控制树进行一致性检查,凡有不当之处提醒设计者修改.彩排过程对一些不应有的操作不予处理,对已经执行的和即将执行的操作提供醒目色或相应的提示光标,令用户易学.

GUIEditor生成的基于事件驱动的对话控制字解释程序和传统的事件驱动型程序比较起来,有许多优点.由于它引入了对话控制树,使得操作条理性增强,任一时刻不该接受的事件即在对话控制字中查找不到的事件不予处理,同时GUIEditor支持对话的回退,它定义了一些特殊功能键来取消当前对话,恢复上一次对话的环境,使得用户可以方便地修改在此之前输入的数据,这些均依赖于对话树的存在得以方便地实现,而在一般的事件驱动型程序中,得专门引进回调函数才能实现.另外,GUIEditor还提供了集成的嵌入式帮助系统,在任何时刻均可查看当前对话以及其前后对话的帮助信息.

6 使用实例

利用GUIEditor我们已为一个规模不算大,但很有代表性的绘图系统Drawing设计了用户接口.Drawing有两棵菜单树,根菜单如图1中所描述,它是一个全局型菜单,并且有一子菜单用于文件操作.画图菜单如图6所示,是一替换型菜单,用来不断地生成图元,它也有一个子菜单.Drawing定义了3个对话框,第一个如图2所描述,用来存取文件,第二个用于确认是否退出系统,第三个用于当欲装入的文件不存在时,给出一个反馈信息.Drawing共调用19个应用过程,其中"装入文件"过程有相应的回退过程,用来当要装入的文件不存在时,重新设置当前文件名.图6中的标尺可以在屏幕布局时创建,目的是方便用户的交互技术操作.Drawing的对话控制树的描述结果如图3和图5所示,它使用了橡皮条线、橡皮条折线、橡皮条矩形、橡皮条圆、橡皮条Bezier线、文字输入和对象选择、对象移动、对象变比等许多交互技术.利用GUIEditor完成Drawing的用户接口描述约需一小时.

7 结束语

GUIEditor目前是在AT&T 6386SX的AT&T OPENLOOK平台上开发的,并成功地移植到SUN4的OpenWindows 2.0平台上.GUIEditor对用户接口设计的交互方面尤为关

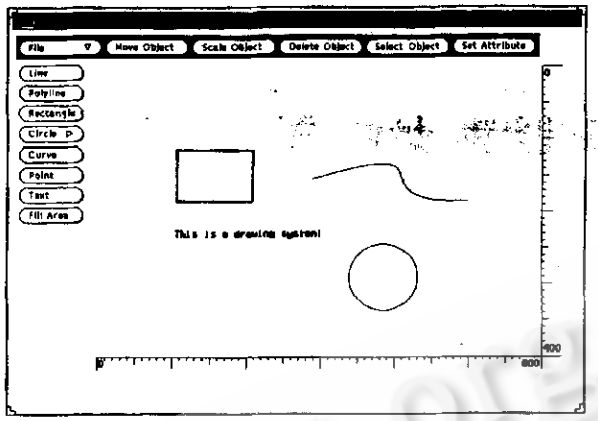

图 6 GUIEditor 的一个生成例子

注,设计者描述的每一步骤都可以立即看到.使用 GUIEditor,设计者不必具备程序设计的技巧,也不必学习任何特殊的语言,它提供了强有力的缺省,以及给设计者很大帮助的对象显示功能.彩排器的一致性检测保证了接口的正确生成,其修改功能的存在方便了设计者.目标用户接口的多线控制提高了用户的工作效率,嵌入式帮助的系统使得用户接口更加友善,尤其有利于帮助新手或没有经验的用户尽快熟悉接口.

有一些 UIMS 系统与 GUIEditor 在某些功能上类似.如 DialogEditor[4],它提供一对象集合利用直接操纵技术来创建用户接口.但是,GUIEditor 的处理特性是把对象看成是层次式的并构造成更高层的对象.又如 Druid 是一描述性的快速用户接口开发系统[5],它描述了对象的表示和动作,并由彩排来检验设计结果,这相似于 GUIEditor.但是 GUIEditor 的彩排器提供了在彩排时的修改功能,而 Druid 系统没有这种修改功能.

GUIEditor 也有待改进之处:各子对话框之间的联系描述不够;如何方便地描述更多的数据类型,并实现其间的转换.由于用户的输入大都简单或跟交互件的属性相关,GUIEditor 可以适用于较多类型的应用系统.

参考文献

1 Myers B A. User—interfaces tools: introduction and survey. IEEE Software, January 1989.

2 Asente P J, Swick R R. X window system toolkit. The Complete Programmer's Guide and Specification.

3 蔡士杰,张福炎.CAD 系统的一种通用用户接口.计算机学报,1988,11(7).

4 Cardelli L. Building user interfaces by direct manipulation. Proc. ACM SIGGRAPH Symposium on User Interface Software, ACM, New York, 1989. 152—166.

5 Singh G, Kok C H, Ngan T Y. Druid: a system for demonstrational rapid user interface development. Proc. ACM, New York, 1990. 167—177.

THE DESIGN AND IMPLEMENTATION OF X—BASED GRAPHICAL USER INTERFACE EDITOR

Wu Qingwei Cai Shijie

(*CAD Laboratory, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing* 100080)

(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210008)

Abstract In this paper, the design idea and implementation method of a graphical user interface editor(GUIEditor) used to specify and generate the user interface which has the rehearsal function is introduced. The usage of dialogue control tree and input data arrays in the separation of the user interface from the application is described. Finally, the dialogue control word interpretation program based on the event—driven mechanism is discussed.

Key words Dialogue control word, interactor, input data array(IDA), rehearse, multi—threaded.