于连庆,胡争光. MICAPS 中天气图交互制作子系统. 应用气象学报,2011,22(3):375-384.

MICAPS 中天气图交互制作子系统

于连庆*　胡争光
(国家气象中心,北京 100081)

摘　要

该文详细讨论了 MICAPS 中天气图交互制作子系统主要功能的实现技术与方法,包括使用组合设计模式定义图形对象,得到的类结构体现了图形对象之间的联系,不仅使代码可复用性较高,而且便于新类型图形对象的加入;使用状态设计模式设计了图形对象交互绘制过程的程序结构,既为用户提供了自定义手势事件功能,又增加了程序代码的可读性和可维护性;提出了修改曲线型图形对象的算法;使用命令设计模式实现了编辑图形对象的撤销/重做功能和软件系统运行日志,提高了软件系统的易用性和可靠性;讨论了编辑文档的自动保存功能的实现方法。通过以上功能的实现,天气图交互制作子系统方便用户操作,具有较高的运行效率和可靠性。

关键词: 天气图交互制作; 设计模式; 文档自动保存

引　言

天气图的制作是气象台站天气预报业务中的重要工作之一,早期天气图制作是预报员在图纸上用铅笔手工绘制的,随着计算机技术的发展,天气图的制作已经改为由预报员在工作站上使用专业的天气图制作软件通过交互方式完成。国内广泛使用的 MICAPS[1-3],MESIS[4-7],美国的 AWIPS[8-9],欧洲的 METVIEW,SYNERGIE 等[10]都是有代表性的专业气象软件,其中均包括天气图制作这一项必不可少的功能。

本文从开发设计人员的角度,针对 MICAPS 1.2版天气图制作子系统中存在的问题和当前的实际需求,详细讨论了解决问题所用到的软件技术与实现方法。MICAPS 3 版提供的新功能包括:图形对象的抽象化描述和类结构定义,用户操作命令的管理,交互子系统的可靠性和用户编辑文档的完整性(包括文档的自动保存和程序异常退出后的程序环境恢复),图形对象的显示效率,程序代码的可复用性和可维护性。

面向对象技术是软件工程发展历史上最重要的技术突破之一,它推动了软件功能模块复用技术和设计模式[11]的发展。本文根据天气图交互制作子系统的特点,在设计中使用了多个设计模式,既解决了上文所提功能的实现问题,又提高了程序代码的质量,减轻了日后系统维护的难度。

1　图形对象的抽象化与类封装

一个专业气象行业软件(例如 MICAPS)的主要功能是分析、显示和编辑图形化的气象数据。在气象科研和业务中,每一种气象数据都有与之对应的图形表示方法[12-13]。通过对国内外气象文献和资料的统计,作者对描述气象数据的图形进行了分类和概括。常见的图形对象有天气符号(包括风向杆、雨、雪、沙尘等),线条(包括等值线、槽线)及其变化形式(包括各种类型的锋面、霜冻线、切变线等),闭合线条(如降水区等),二维位图(如传真图、地图、地形图、云图、雷达拼图、三维或者四维气象物理量数据的二维剖面图等),文字(如等值线标值、高低值中心等)以及这些图形对象的组合。今后随着气象科学的发展可能会有新的图形对象产生。

图层是图形对象表示中的一个重要概念,指逻辑上包含多个图形对象的集合,这个集合往往被当作一个整体看待。例如,当一个图层被置于显示(或

2010-08-16 收到, 2011-03-09 收到再改稿。

* E-mail: yulq@cma.gov.cn

隐藏)状态时,则该图层内的所有图形元素都被显示(或隐藏)。

如何使用面向对象技术有效地描述这些图形对象对于软件系统的设计者来说非常重要,图形对象的数据结构表示对于程序的响应速度、内存占用数量、图形的显示效率等方面有着至关重要的影响。一个合理的对图形对象的封装设计应该满足以下标准:正确反映图形对象之间的相互关系;便于实现对图形对象的各种操作,如新建、删除、搜索和显示;占用较小的存储器空间。

为了满足以上所提的标准,本文使用了组合(Composite)设计模式来实现对图形对象的抽象化与类封装。组合设计模式适用于系统中不同对象之间存在着个体与整体的组合关系的情况。在组合设计模式中,一个图形对象既可以是一个不可分解的原始(Primitive)图形对象,如文字图形和线条图形,也可以是由多个原始图形对象组合而成的组合图形对象,如等值线图形就由线条图形(等值线)和文字图形(等值线标值)组成。多个组合图形对象又可以组成一个更大的组合图形对象,如一张天气图由多个等值线图形、天气符号图形构成,前面提到的图层就是一个组合图形对象。

图 1 给出了使用组合设计模式对天气图交互系统中图形对象设计的类结构。图中带箭头的实线表示类的继承关系,箭头指向父类(即基类)。虚线表示引用(Reference)关系,箭头指向被引用的对象。组合设计模式的实现关键是设计一个抽象基类,该基类中定义了所有对象公有的属性和方法。抽象基类的子类定义了具体的图形对象。根据这一思路,本设计中定义了一个位于类结构最顶层的抽象基类:mGraphicsComponent。这个类定义了对于所有图形对象公有的属性和方法。属性包括图形对象标识(Identifier),绘制尺度比例,是否处于高亮状态或者被选中状态,表示图形对象大小的边界框等;方法有自身的绘制(Draw),添加/删除(Add/Remove)构成自身的原始图形对象个体,返回该对象的名称(Name)等。

类 mGraphcsCompositeComponent 是类 mGraphicComponent 的子类之一,该类表示一个组合图形对象。此外,类 mGraphicsComponent 的直接继承类还有 mGraphicsContour,mGraphicsText 等,其中类 mGraphicsContour 是等值线集合的图形表示。这个等值线集合通常是对观测场或格点场进行客观分析的结果,不能被用户修改。与之相比,类 mGraphicsIsoline 表示一条等值线,一般由用户绘制得到。

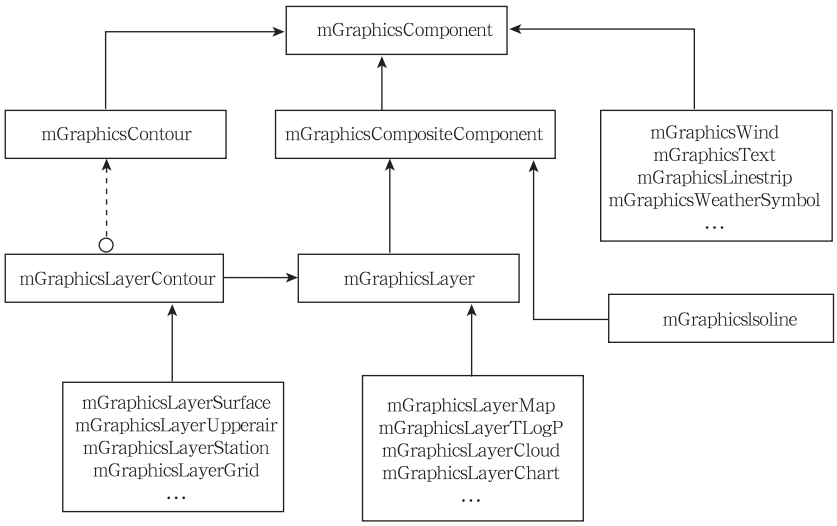

图 1 天气图交互制作子系统中图形对象的类定义结构

Fig. 1 The class hierarchy that encapsulates all types of graphical objects in a subsystem for interactively drawing synoptic chart

因为图层可以被看作一个组合图形对象,因此在类结构中,表示图层的类 mGraphicsLayer 被定义为 mGraphicsCompositeComponent 的子类。这里又将图层对象分为两类:包含等值线集合的图层

(mGraphicLayerContour)和不包含等值线集合的图层(mGraphicLayer)。这样涉及到等值线分析和显示的大量代码不必出现在所有图层类中,既实现了代码共享,又减小了程序对系统资源的要求。当然还可以根据图层对象的其他特点派生出类 mGraphicLayer 新的子类。例如,可以将所有包含二维位图的图层定义为类 mGraphicsLayer 的子类 mGraphicsLayer2DImage,然后将类 mGraphicsLayerMap, mGraphicsLayerCloud 作为其子类。

除了图 1 所示的表示图形对象的类结构外,还有其他两种可供选择的表示方法:一种表示方法是每一种图形对象对应一个唯一的类;另一种表示方法是使用一个类来表示所有的图形对象。MICAPS 1.2 版中使用了第 1 种表示方法。

第 1 种表示方法的问题是每个类之间缺乏逻辑上的联系,整个类结构没有体现出不同类型图形对象之间的关系,表现出来的缺点包括:操作图形对象的主体必须区分图形对象的类型,导致程序中出现过多的条件判断语句;当增加新的一种图形对象时,根据上一点,操作图形对象的代码必须随之修改,增加了程序维护的工作量,同时导致用户在二次开发中无法自定义新的图形对象;当不断加入新的图形对象后,图形对象类的数目随之激增。

第 2 种表示方法的问题是表示图形对象的类定义过于臃肿,造成程序耗费的内存资源过多。

新设计的类结构解决了以上问题,体现出来的优点包括:由于抽象基类 mGraphicComponent 提供了公有的属性和方法,因此操纵图形对象的主体不用区分目标对象的具体类型,这避免了程序中出现过多的条件判断语句;加入新的图形对象更为容易,便于用户在二次开发中自定义新的图形对象。当加入新的图形对象时,不需要更改操纵图形对象主体的程序代码,因为图形对象的类型对于主体是透明的;由于在逻辑上体现了不同类型图形对象之间的关系,提高了代码可复用性,降低了空间复杂度。

2　图形对象的交互编辑

在图形对象的交互绘制过程中,程序需要根据用户的输入信息,在视图窗口内实时绘制用户选择的图形对象,其中处理操作系统消息事件函数的程序结构一般相当复杂。主要原因包括:天气图交互制作子系统为用户提供了大量不同类型的图形对象(第 1 章);操作系统通常提供了大量的消息事件(如重画窗口、改变窗口大小、鼠标输入、键盘输入等)来驱动应用程序;同一消息事件又可以因为鼠标键和键盘键的不同状态而存在区别。将以上 3 点提到的情况组合起来会形成近百种不同的消息事件。MICAPS 1.2 版中采用了条件分支语句的形式处理如此众多的消息事件,导致程序代码的复杂度随着新消息事件的加入而急剧增加,因此增加了代码调试和维护的难度。

为了避免上述问题,本设计使用了状态(State)设计模式。该模式适用于一个对象的行为随着其自身状态变化而改变的情况,并且自身状态的变化发生在程序运行时刻,具有不确定性。这一特征与天气图交互绘制过程中程序的行为特征相一致,即程序因被绘制的图形对象不同而呈现出不同行为。例如,当用户绘制风向杆时,程序在鼠标移动事件处理函数中实时更新风向杆的方向;而当用户绘制曲线时,程序忽略鼠标移动事件。同时,在程序运行时用户选择绘制某一图形是不确定的。根据这一特点,在应用状态设计模式时,本设计将用户对某一类型图形对象的选择称为一个状态,将程序的行为用类抽象化,使其能够用类的对象描述。由此可见,与使用条件分支语句的实现方式相比,采用状态设计模式实际上是将条件分支语句中某一条件下的代码功能用一个类来实现。

图 2 给出了描述所有状态的类结构图。图 2 中类 mguiRubberbandView 代表绘图视图,该类的函数 OnMessage 处理操作系统向交互子系统发送的所有消息事件。OnMessage 函数调用状态类 mDrawState 的 Handle 函数实现每个绘图状态特有的行为。

为了便于检测某一消息事件是否为定义绘图过程开始、持续和结束的手势事件,在类 mguiRubberbandView 中定义了类 mguiEventCondition 的 3 个实例 getsture0, gesture1 和 gesture2。类 mguiEventCondition 用于判断一个接收到的消息事件是否与给定的手势事件相同,其作用相当于将判断一个消息事件的代码从条件分支语句转移到该类中。这种实现方式不仅提高了代码的可阅读性,而且还为用户提供了自定义手势事件的机会。当用户更改了手势事件

后，只需要改变变量 getsture0，gesture1 和 gesture2 即可。与之相比，在采用条件分支语句的实现方式中，当用户需要更改手势事件时，必须修改相应的程序代码。

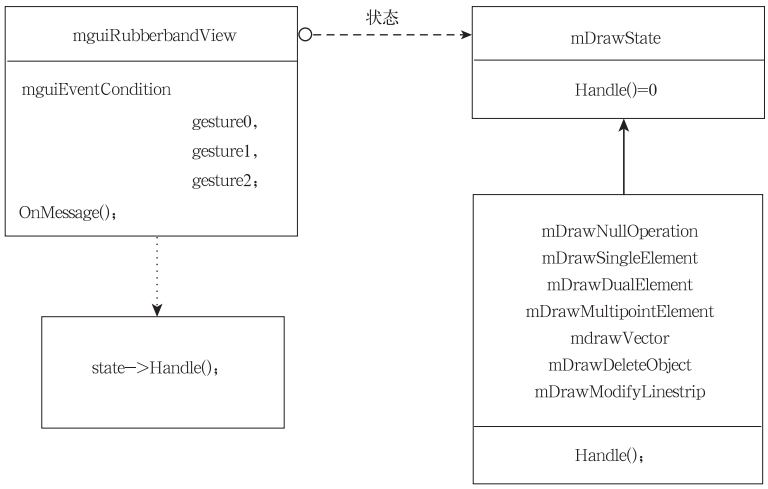

图 2 使用状态设计模式对用户绘图中所有状态进行描述的类结构图

Fig. 2 Using the state design pattern, a class hierarchy is proposed to represents all states corresponding to users' selection of graphical objects

图 2 中定义了抽象基类 mDrawState 以表示一个状态。点线表示函数定义，虚线和实线与图 1 相同。根据图形对象的特点，本设计定义了 7 种绘图状态，分别用类 mDrawState 的 7 个子类描述(表 1)。

表 1 描述 7 种绘图状态的 mDrawState 子类

Table 1 The subclasses of mDrawState represent seven drawing states

类名称	绘图状态	被操作的图形对象
mDrawNullOperation	空操作，即不绘制任何图形，对应于用户没有选择任何图形对象的情况	
mDrawSingleElement	绘制一个单点型图形对象，用户绘制此类对象一般需要在视图中单击 1 次鼠标左键	所有的天气符号(如雨，雪，台风等)
mDrawDualElement	绘制一个单点型图形对象，但该对象的具体标识与所使用的鼠标键有关(用户单击左键时，生成高压中心；点击右键时，生成低压中心)	高低压中心，冷暖中心
mDrawMultipointElement	绘制曲线型图形对象，用户绘制时需要多次单击鼠标键来确定曲线上点的位置	等值线，槽线，各种类型的锋面，霜冻线，降水区等
mDrawVector	绘制矢量型图形对象，用户首先通过单击鼠标确定图形对象位置，然后移动鼠标调整图形对象方向和长度	风向杆
mDrawModifyLinestrip	修改已绘制的线条型图形对象	曲线型图形对象
mDrawDeleteObject	删除选中的图形对象	所有的图形对象

状态设计模式的一个实现问题是由谁负责改变绘图的状态。根据天气图交互制作系统的特点，开发人员设计了一个绘图工具对话框(图 3)，对话框中的每个按钮对应一个图形对象。当用户需要画某个图形对象时，只需点击相应按钮即可。程序在点击按钮事件的回调函数(callback)中根据用户选择的图形对象来改变当前的绘图状态。

与 MICAPS 1.2 版中的交互子系统相比，新设计的优点包括：将一个状态的行为放到描述该状态的类中实现，避免了复杂冗长的条件分支语句，当增加新的状态时只要定义抽象基类 mDrawState 新的子类即可，而无需在已经十分复杂的条件分支语句中加入新的代码；为用户提供了自定义手势事件的功能。与之相比，在原系统的实现方式中，当用户要求更改手势事件时，必须修改相应的程序代码。

图 3　绘图工具对话框

Fig. 3　The toolbox dialog

2.1　修改曲线型图形对象

在天气图交互制作的实现中,修改曲线型图形对象、特别等值线具有较大难度。这是因为预报员在修改线条时已经习惯了 MICAPS 1.2 版引入的部分重画方式,而不是拖动节点的方式。因为部分重画方式的修改结果往往具有二义性,所以程序要猜测预报员修改线条时的意图。图 4 示例了两种等值线修改结果存在二义性的情况。其中每一行的左列显示了用户开始绘制的曲线(实线)和修改时画的曲线(虚线),中间一列和右列是两种可能的修改结果。它们之间的区别在于用户修改时绘制的曲线替代了原曲线中的哪一部分。这种二义性造成了 MICAPS 1.2 版在处理封闭等值线时,有时无法给出预报员期望的结果。

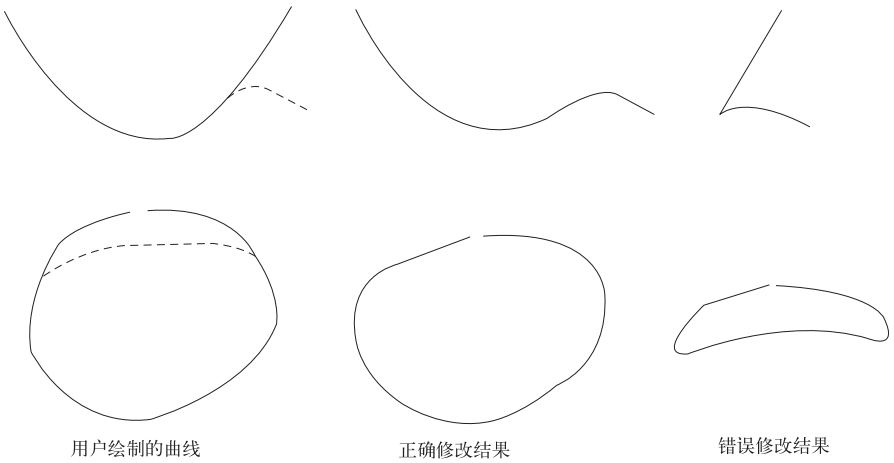

图 4　等值线修改结果存在二义性的两种情况

Fig. 4　Two cases demonstrating the ambiguity of modified isolines

虽然修改结果存在二义性,但在天气图制作背景下,由于曲线型图形对象都呈现出比较平滑的外观,没有拐角很大的走向,因此图 4 中间一列的结果是用户所希望的修改结果。

根据曲线型图形线条比较平滑的特点,本文设计出修改曲线型图形对象的算法,使其得到唯一、合理的修改结果。这里以修改等值线为例。首先对天气图中绘制的等值线进行分析和概括,根据等值线的形状,线条的走向等特征将修改前后的等值线归结为如图 5 所示的 12 种典型情况。其中箭头指向等值线的终点方向。修改等值线时出现的任何一种情况都可以通过对这 12 种典型情况里的某一种进行旋转和对称变换得到。针对这 12 种典型情况,修改曲线型图形对象的算法共 8 个步骤,具体如下。

①计算修改线条(图 5 中虚线)的方向。对于开放的等值线(图 5 中情况 1～4),算法中定义从左向右画的等值线方向为负;从右向左画的等值线方向为正。对于半封闭和近似全封闭的等值线(图 5 中

情况 5～12),定义等值线的方向等于连接等值线上所有点形成的简单闭合多边形的曲线方向(Curve Orientation)。即当沿着多边形行进时,如果多边形的内部始终在行进方向的右侧,则等值线方向为负;反之,则等值线方向为正。以上定义只是为了区别线条方向,并不影响算法最终结果。

②计算被修改的等值线(图 5 中实线)的方向。

③找到在被修改等值线上距离修改线条起点最近的点,将此点记为 P_1。若修改线条的起点不在被修改的等值线上,则算法退出;否则,进入到下一步。

④找到在被修改等值线上距离修改线条终点最近的点,将此点记为 P_2。注意在图 5 中情况 1,2,5,6 中,P_2 点不存在。

⑤如果被修改的等值线是半封闭和近似全封闭(图 5 中情况 5～12),则计算被修改等值线上从点 P_1 到 P_2 之间的曲线的方向,并用此方向取代②的结果,作为被修改等值线的方向。

⑥当被修改等值线的方向与修改线条的方向不一致时,对被修改等值线的所有点按照相反的顺序重新排列。否则,不改变被修改等值线中点的顺序。

⑦提取被修改等值线中从起点到 P_1 点之间的所有点,作为修改后等值线的第 1 部分。将修改线条作为修改后等值线的第 2 部分。提取被修改等值线中从 P_2 点到终点之间的所有点,作为修改后等值线的第 3 部分。对修改后等值线进行采样然后做样条插值[14],使其平滑。如果是图 5 中的情况 9 和 10,则修改后等值线由被修改等值线中从 P_1 到 P_2 点之间的所有点和修改线条中的所有点组成。

⑧检测被修改等值线的所有等值线标值,如果某一等值线标值的位置距离修改后等值线距离过远,则删除该等值线标值。

等值线是曲线型图形对象中形状变化最为复杂的图形。对于其他曲线型图形对象,如槽线、锋面等,该算法也能正确处理。总之,该算法保证了修改后的曲线是用户所希望的修改结果。

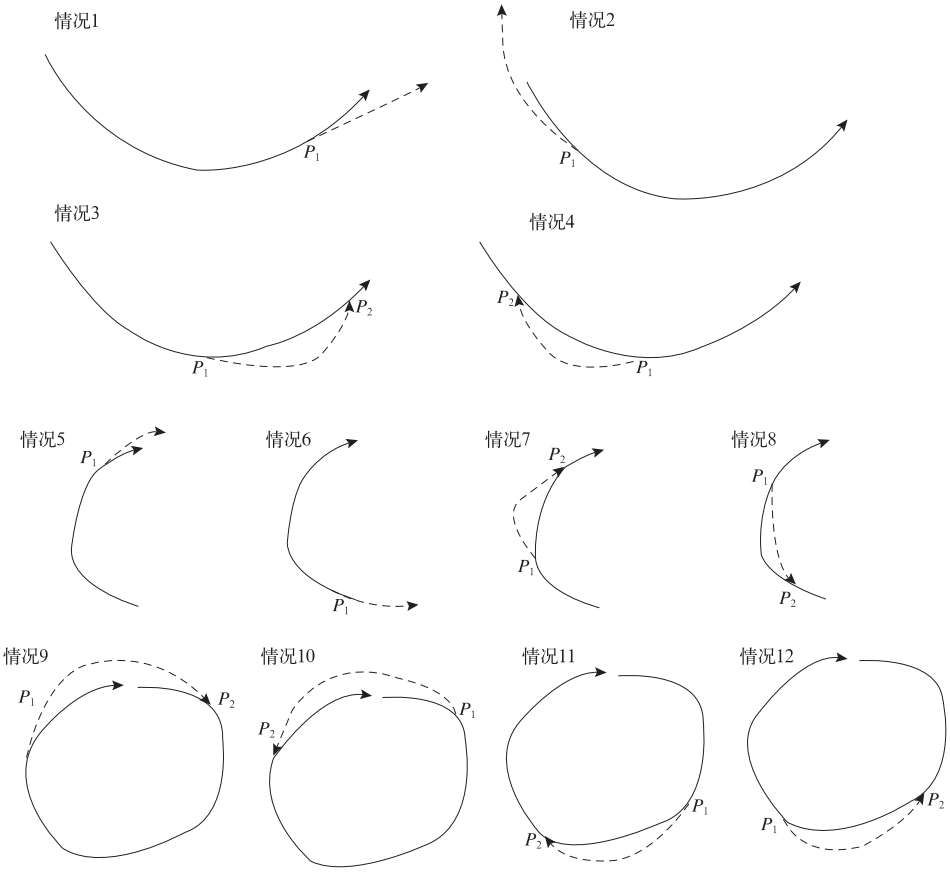

图 5 根据等值线的形状、走向等特征将修改前后的等值线归结为 12 种典型情况

Fig. 5 12 unique cases according to the shape and direction of isolines

3　易用性与可靠性

用户对软件系统的易用性和可靠性有较高要求,提出的具体需求有:编辑图形对象命令的撤销与重做;程序异常退出后的会话环境恢复;编辑文档的自动保存。MICAPS 1.2 版只提供了有限的编辑图形对象命令的撤销功能。为此,在新的设计中完善了该功能,同时增加了后面两项功能。

3.1　编辑图形对象时撤销/重做功能的实现

在交互式编辑软件中,用户操作的撤销(Undo)和重做(Redo)功能是必不可少的。这里使用命令设计模式(Command)实现这一功能。在命令设计模式中首先要定义一个描述用户操作的抽象基类 mCommand,该类中一般声明了表示执行用户操作命令的函数 Execute 和表示撤销用户操作命令的函数 UnExecute。类 mCommand 的子类用来表示一种具体的用户操作命令,子类中需要声明一个成员变量用来表示执行(或撤销)用户操作命令的主体。

图 6 显示了应用命令设计模式来描述天气图交互制作系统中部分用户操作命令的类结构。这里以保存天气图到磁盘文件和绘制天气图这两个操作为例。类 mSaveFileCommand 描述了用户保存天气图到磁盘文件的操作命令,类中声明了一个文档类指针 doc,即上文中提到的执行(或撤销)用户操作命令的主体。执行保存操作命令的函数 Execute 调用 doc 的 Save 函数。与此相似,类 mDrawCommand 封装了用户绘制天气图时的操作命令,类中声明了一个图层管理类的指针 glman,作为执行(或撤销)用户操作命令的主体。注意到类 mDrawCommand 仍然是一个抽象类,其子类才真正封装了用户的具体操作命令。例如,类 mDrawAddSymbolCommand 描述了用户创建一个天气符号图形对象的操作,该类的函数 Execute 通过调用 glman 的 Add 函数将新建的天气符号图形对象加入到已有的天气图数据中。图 6 中的类 mguiGraphicsView 表示显示图形对象的视图,该类中声明了保存所有用户操作命令对象实例的列表型变量 command_list。每当用户执行一个操作命令时,该命令对象的实例就被添加到 command_list 中。通过正(反)向遍历 command_list 并执行其中保存的对象实例的 Execute(UnExecute)函数,可以实现用户操作命令的无限层次的重做(撤销)。从而对 MICAPS 1.2 版中的相应功能进行了完善。

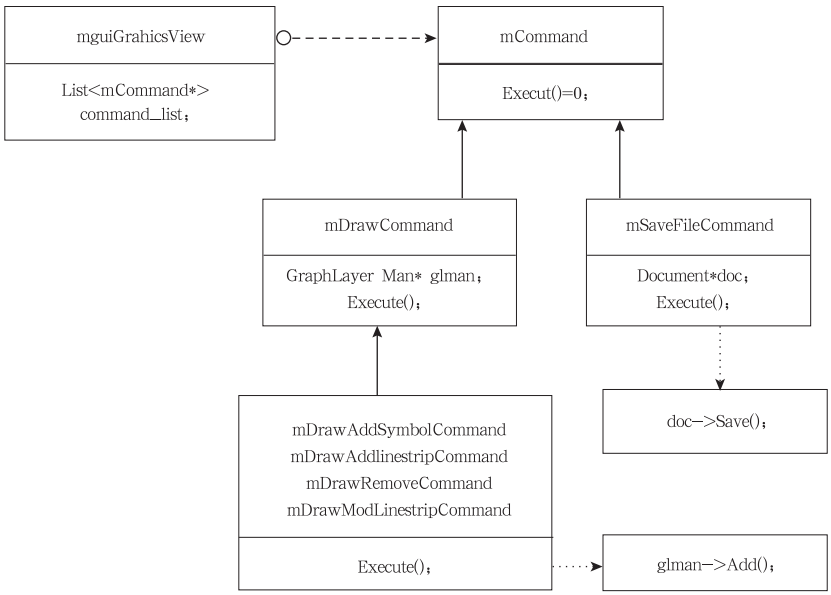

图 6　描述天气图交互制作子系统中部分用户操作命令的类结构

Fig. 6　The class hierarchy that represents some user operations in a subsystem for interactively drawing synoptic chart

实现命令列表的一个设计问题是类 mDrawCommand 中应该如何表示用户新建、删除和修改的图形对象。一个图形对象的表示可以有 3 种方法:图形对象的指针或者引用,图形对象的复制,创建图

形对象所需的数据。依据以下标准对这 3 种方法进行比较。

• 数据完整性。图形对象中的数据在整个用户操作期间不能丢失。

• 功能模块之间的耦合性。图形对象模块与操作命令管理模块之间应该是低耦合。

• 磁盘文件大小。用于保存命令列表的磁盘文件应该占用较少的磁盘空间。

• 内存使用大小。图形对象应该占用较少的内存空间。

对于图形对象的指针或者引用的表示方法,当用户删除 1 个图形对象时,该图形对象的指针或者引用将不再有效。另外,图形对象的构造(析构)必须在操作命令类的 Execute(UnExecute)函数中进行。进而对图形对象的一些操作必须放到操作命令类中,这增加了程序功能模块之间的耦合性。由于保存了图形对象类的所有信息,因此文件尺寸较大,但是根据 C/C++语言的实现机制,这种表示方法占用的内存空间最小。总之,该方法在数据完整性标准中存在问题。解决此问题的方法是将图形对象的所有操作放到操作命令类中,但这样增加了程序功能模块之间的耦合性。

对图形对象的复制的表示方法来说,由于是图形对象的复制,因此该方法非常耗费内存空间。该方法的优势是在数据完整性和功能模块的耦合性上不存在问题。

采用表示图形对象的数据的表示方法时,当用户删除 1 个图形对象时,该图形对象的内部状态信息(标识号、高亮状态)将会丢失,这导致该图形对象无法被忠实地复原。由于只保存了创建图形对象的数据,因此文件大小低于其他两种方法,并且占用内存大小介于其他两种方法之间。总之,该方法最大的问题是当删除图形对象时无法保持该图形对象信息的完整性。

分析表明,由于前两个标准更为重要,因此使用图形对象复制的表示方法是唯一选择。

3.2 程序异常退出后会话状态的恢复

使用命令设计模式带来的另外一个优势是该模式提供的日志功能。在对类 mCommand 增加磁盘文件读写方法后,将 command_list 保存到磁盘文件中。当程序异常退出时,可以通过读取文件来恢复 command_list,然后执行其中保存的用户操作命令,进而将程序恢复到异常退出之前的会话(Session)状态。

3.3 编辑文档的自动保存

编辑文档的自动保存也是交互式软件系统必备的一项功能。其实现方法是将用户编辑文档的最新内容保存到一个临时文件中(每一个被编辑的文档都对应于唯一的一个临时文件)。当程序正常(非正常)退出时,这个临时文件被(不会被)删除,下一次程序启动时检查是否存在某个临时文件,如果存在,则说明上次程序运行时没有正常退出,这时程序提示用户是否将文件内容恢复为最后一次自动保存的内容。

这里还有一点需要考虑:自上一次自动保存后,如果被编辑的文档未被真正修改,则不应该引发任何磁盘读写操作。例如当用户执行一个操作,然后又撤消了该操作后,文档实际上没有被修改。为了判断被编辑的文档是否被真正修改过,可以定义一个指向命令列表当前命令的指针,该指针实际上记录了用户最后一次执行的操作命令。当该指针不变时,则说明被编辑的文档内容没有发生变化。

通过使用命令设计模式,新的设计实现了交互图形绘制子系统所需的多项功能,做到了一举多得。

4 结论与讨论

天气图交互制作软件已经成为气象业务工作中不可缺少的工具之一。本文从软件开发设计者的角度对以下功能的技术实现进行了详细讨论:

1) 使用组合设计模式实现对图形对象的抽象化和类封装,得到的类结构体现了不同类型图形对象之间的联系。利用 C++语言提供的多态性,操纵图形对象的主体可以统一地看待不同类型的图形对象,避免了程序中出现复杂冗长的条件分支语句。同时当加入新的图形类时,无需更改操纵图形对象主体的程序代码,增加了代码的可扩展性和可维护性。

2) 使用状态设计模式设计了图形对象交互绘制过程的程序结构,将绘制某一类型图形对象的功能放到描述操作命令的类中。同时定义了表示交互绘制过程开始、持续和结束的手势事件类,为用户提供了自定义手势事件的功能。

3) 提出了修改曲线型图形对象的算法,修改后的线条是用户期望的修改结果。

4) 使用命令设计模式实现了编辑图形对象时

撤销与重做功能和软件系统运行日志,前者便于用户编辑修改图形对象,后者使程序异常退出后能够还原到上次会话的状态。这两项功能提高了软件系统的易用性和可靠性。

5) 讨论了编辑文档的自动保存功能的实现方法,保证了编辑文档的完整性。

本文设计的交互子系统已集成到 MICAPS 3 工作站版中,并在中央气象台天气图分析业务中投入运行。从反馈意见看,用户对交互系统的易用性、可靠性和运行效率予以肯定,许多预报员已经从使用多年的 MICAPS 1.2 版转而使用 MICAPS 3 工作站版。

由于天气图交互制作功能一般做成气象专业软件的一个功能模块,因此本文没有讨论诸如程序界面的设计、视图窗口的类定义、图形对象的硬件加速显示等功能,这些功能的设计与实现需要与软件的整体设计相一致。

本文仅从桌面客户端程序设计的角度对上述功能进行了讨论,随着软件技术的发展和气象业务需求的增加,天气图交互制作软件系统的运行平台、实现方法和功能要求都有可能发生变化。作为程序开发人员,紧跟计算机技术的发展,及时了解用户的需求,为用户提供高效、稳定的软件工具是本工作的最终目标。

参 考 文 献

[1] 李月安, 曹莉, 沃伟峰, 等. 强天气监测和潜势预报系统. 应用气象学报, 2006,17(4): 141-146.

[2] 李月安, 曹莉, 高嵩, 等. MICAPS 预报业务平台现状与发展. 气象, 2010, 26(7):50-55.

[3] Yu L, Cao L, Li Y, et al. Introduce on Typhoon Forecast Operational System// Proceedings of 27th IIPS for Meteorology, Oceanography, and Hydrology. 2011.

[4] 郑卫江, 吴焕萍, 罗兵,等. GIS 技术在台风预报服务产品制作系统中的应用. 应用气象学报, 2010, 21(2): 250-255.

[5] 孙利华, 吴焕萍,郑金伟,等. 基于 Flex 的气象信息网络发布平台设计与实现. 应用气象学报, 2010, 21(6): 754-761.

[6] 吴焕萍, 罗兵, 王维国, 等. GIS 技术在决策气象服务系统建设中的应用. 应用气象学报, 2008, 19(3): 380-384.

[7] 吴焕萍, 罗兵, 曹莉. 地理信息服务及基于服务的气象业务系统框架探讨. 应用气象学报, 2006, 17(增刊): 135-140.

[8] Hopkins T, Henry R, Mandel E, et al. AWIPS II Migration Status// Proceedings of 27th IIPS for Meteorology, Oceanography, and Hydrology. 2011.

[9] Griffith F. AWIPS-II Into the Future// Proceedings of 27th IIPS for Meteorology, Oceanography, and Hydrology. 2011.

[10] Daabeck J. Overview of Meteorological Workstation Development in Europe // Proceedings of 21st International Conference on Interactive Information Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology. 2005.

[11] Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Professional, 1994.

[12] Moran J M, Morgan M D, Pauley P M. Meteorology: The Atmosphere and Science of Weather. New Jersey: Prentice Hall, 1996.

[13] 寿绍文. 天气学分析. 北京:气象出版社, 2002.

[14] Boor C de. A Practical Guide to Splines, Revised Edition. New York: Springer-Verlag, 2001.

[15] Stroustrup B. The C++ Programming Language. Boston: Addison-Wesley Professional, 2000.

Subsystem for Interactively Drawing Synoptic Chart in MICAPS

Yu Lianqing Hu Zhengguang

(*National Meteorological Center*, *Beijing* 100081)

Abstract

A subsystem is developed for interactively drawing synoptic chart to solve issues in previous versions of MICAPS and meet new user requirements. First of all, the composite design pattern is used to define the class hierarchy that describes graphical objects. This class hierarchy fits well with the relation among different graphical objects and therefore reduces the spatial complexity of the program, making manipulations to graphical objects much easier to implement. Thanks to the polymorphism feature provided by modern object-oriented programming languages, clients manipulating the graphical objects can treat all objects in this composite structure uniformly. The consequential benefit of this design advantage in practice is that the client code does not need to be modified when new types of graphical objects are introduced. This is especially useful for developers to define customized graphical object in their own applications. Above all, the proposed class hierarchy not only promotes code reusability, but also makes it easy to add new types of graphical objects. The code that draws rubber band graphical objects is constructed using the state design pattern, in which the behavior of drawing a particular type of graphical object is implemented in the corresponding class. The advantage of this approach is that introducing new types of graphical objects only require deriving new subclass and does not affect existing code. In addition, the gesture event class is proposed to represent the start, continue and end gestures in the rubber band drawing. In this way, the gestures can be customized at runtime by users. On the contrary, if the conventional code structure using large conditional statements is employed for rubber band drawing as in the previous implementation, the code complexity will increase dramatically when new types of graphical objects are introduced and the code defining the gestures has to be modified when user changes these gestures. The disadvantage makes it difficult to extend, debug and maintain the code. An algorithm to modify linestrip-type graphical objects, in particular isolines, is proposed. By taking advantage of the smoothness constraint, the proposed algorithm is able to solve the ambiguity in isolines modification and make the modified result meet the user expectation. The command design pattern is employed to implement undo/redo and runtime logging. The former feature facilitates users manipulation and the latter feature can restore the program session to the last successful state after the program quits abnormally. These two features enhance usability and reliability of the program. The implementation of automatically saving edited document which ensures the integrity of the documents is discussed. With all these features implemented, the proposed system not only facilitates users manipulation but provides good performance and reliability as well.

Key words: interactively drawing of synoptic chart; design pattern; automatic save