

数据结构可视化类库的设计与实现

苏莹, 吴伟民

(广东工业大学 计算机学院, 广东 广州 510090)

摘要:本工作室开发的数据结构可视化类库(JVDSCL, Visual Data Structures Class Library in Java)将可视化技术引入数据结构类库,实现了数据结构可视化。介绍了对数据结构类的可视化扩充方法,给出了实现各种数据结构可视化布局算法的基本框架。JVDSCL可以应用在程序调试和软件开发,提高软件的可视性、重用性与开发效率。

关键词:数据结构;可视化;类库

中图分类号:TP311.5

文献标识码:A

文章编号:1673-629X(2006)05-0061-04

Design and Implementation of Visual Data Structures Class Library in Java

SU Ying, WU Wei-min

(Computer Faculty, Guangdong University of Technology, Guangzhou 510090, China)

Abstract: JVDSCL (Visual Data Structures Class Library in Java) implements the data structures visualization through the visualization technology. This paper presents an extendible method to visualize data structures, and the basic algorithms frame of the key layout for various data structures visualization. JVDSCL can be used in visual debugging and learning programming, improving software visualization, reusability and software development efficiency.

Key words: data structures; visualization; class library

0 引言

数据结构可视化是软件可视化的一个分支领域。软件可视化系统^[1]通过图形的方式,监控和探测指定格式的程序,实现程序可视化。程序可视化的技术通常可以分为4种类型^[2]:事件驱动(Event Driven),状态驱动(State Driven),可视程序设计(Visual Programming),自动动画(Automatic Animation)。

采用事件驱动方法的系统有ANIMAL系统^[3]等。采用事件驱动方法,需要在程序需要可视的地方用interesting events注释,相应的演示动画根据这些interesting events编写。采用状态驱动方法的系统有LEONARDO^[4]等。状态驱动方法是指在程序与可视化状态之间确定一个映射关系,这个映射关系通常在程序运行之前由可视化工具指定。采用可视程序设计的目的在于通过可视符号表示程序命令和句子来使程序更容易确定。涉及自动动画这个领域的系统有Jeliot^[5]。Jeliot通过可视解释器实现基于数据类型的自我可视(self-animation);开发人员在源代码中选择要可视的数据类型,Jeliot就自动产生演

示动画。

笔者开发的数据结构可视化类库(JVDSCL, Visual Data Structures Class Library in Java)将可视化技术引入到数据结构类库中,提出了一种对数据结构类的可视化扩充方法:构造可视数据结构。可视数据结构是在原有的数据结构的基本属性与操作的基础上,增加一组可视属性,提供可视化接口。这种方法的优点在于无需利用可视解释器对数据结构进行可视化解释,通过调用其可视化接口就可以实现数据结构的自我可视。JVDSCL提高了软件的重用性和开发效率,它可以应用于软件开发和程序调试。

1 JVDSCL的设计及其实现

JVDSCL是在Java集合库的基础上,对其原有的数据结构类中进行扩展,并增加一些复杂的数据结构,如树、图等。在JVDSCL中,通过构造可视数据结构来实现数据结构的可视化。可视数据结构就是在Java集合库中原有的数据结构类的基本属性和操作的基础上,增加可视属性(如大小,颜色,坐标值等),并提供可视化接口供程序调用。通过调用这些接口,开发人员可以选择在现有的画图区域中显示数据结构,也可以选择在一个独立的新窗口显示数据结构。每一种数据结构都有多种显示模式,开发人员可以选择任意一种进行显示。在JVDSCL中,针对每一种数据结构,提供多种布局方法对其进行布局。

收稿日期:2005-08-09

作者简介:苏莹(1980-),女,广东清远人,硕士研究生,研究方向为数据结构与算法、可视计算;吴伟民,副教授,硕士研究生导师,研究方向为数据结构与算法、可视计算、程序语言系统和嵌入式系统等。

1.1 基本的可视化接口

JVDSCL 最基本的接口是 VCollection 接口。VCollection 接口除了提供 Collection 接口的基本方法外,还提供了基本的可视化接口。

基本的可视化接口有:

* void draw (Graphics g, int panelWidth, int panelHeight, int displayMode, int layoutMethod, Color c)

基本操作:重画指定的数据结构。参数 panelWidth 表示显示区域的宽度;参数 panelHeight 表示显示区域的高度;参数 displayMode 表示数据结构的显示模式,每个数据结构都有若干种显示模式,由 displayMode 参数值来决定选用的显示模式;参数 layoutMethod 表示对可视化的数据结构采用的布局方法,对于每一种可视化的数据结构,类库都有相应的多种布局方法,供开发人员选择,默认值为 0;参数 c 表示数据结构显示的颜色。

* LinkedList setLayout (int panelWidth, int panelHeight, int layoutMethod)

基本操作:对指定的数据结构进行布局,确定数据结构中每个元素的坐标位置。参数 panelWidth 表示显示区域的宽度;参数 panelHeight 表示显示区域的高度;参数 layoutMethod 表示对可视化的数据结构采用的布局方法。返回类型为链表类型,用一个链表来保存可视数据元素的信息。

* void show()

基本操作:将数据结构独立显示在另一个窗口上。

1.2 显示模式

在 JVDSCL 中,每一种数据结构都有不同的显示模式。图 1 表示单链表数据结构的两种显示模式:矩形表示和圆形表示。开发人员可以根据需要选择不同的显示模式。

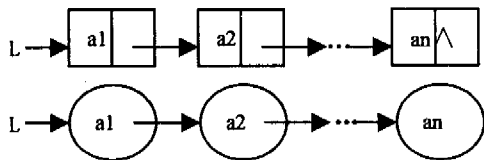


图 1 单链表的显示模式

1.3 布局方法

数据结构可视化的关键问题是图形的布局问题,图形的布局好坏直接影响开发人员理解数据结构及其算法的效果。在 JVDSCL 中,基本的布局方法有线性布局方法(见图 2)、树布局方法(见图 3)和图布局方法(见图 4)。每一种布局方法都有一种或多种布局算法实现。对于每一种数据结构都有默认的布局算法,开发人员可以根据自己的需要选择不同的布局算法。

* 线性布局方法。

线性布局方法适用于线性表、栈、队列等数据结构。

线性布局方法的基本算法框架如下:

- (1) 获取数据元素的个数;
- (2) 根据显示区域的大小和数据元素个数,计算数据

元素直线均匀布局的大小值;

- (3) 依次设置数据元素的大小及其相应的坐标值。

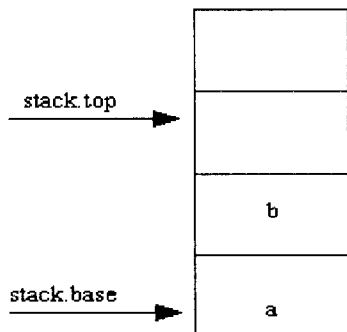


图 2 线性布局方法

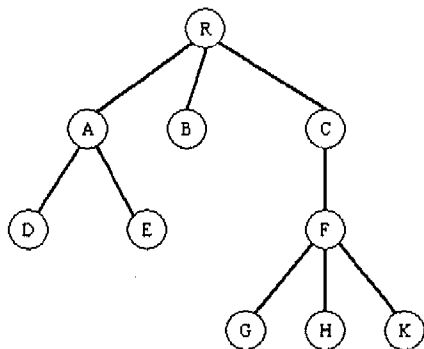


图 3 树布局方法

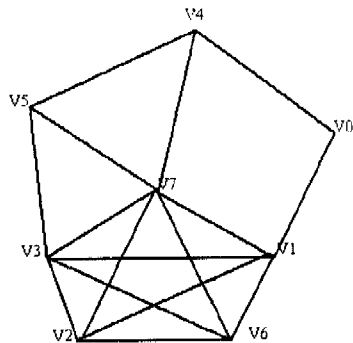


图 4 图布局方法

* 树布局方法。

树布局算法的基本思想:若树非空,后根遍历树。预先设置整棵树的最左下的结点的位置,对于树的每个结点进行以下操作:从左到右依次遍历结点的每棵子树,每一次遍历完毕,返回每棵子树的根结点的位置和最右结点的位置,根据每棵子树的最右结点位置设定下一棵遍历的子树的最左结点位置(此最左结点位置应在前一棵子树的最右结点的位置的右边且相隔一定的常数距离)。若结点是叶子结点,则其子树的最左结点和最右结点位置就是此结点的位置;若结点是非叶子结点,则结点的位置根据其孩子的位置进行定位,位于其最左孩子和最右孩子的位置的中线上。最后返回结点的位置和结点的子树的最右结点位置。这样,遍历完整棵树,就完成树的结点的定位。

根据上述思想,算法中设定 5 个输入参数:参数 t 表示遍历的子树的根结点的指针;参数 lx 表示遍历的子树

的最左结点的 x 坐标值;参数 tx 表示遍历的子树的根结点的 x 坐标值;参数 ty 表示遍历的子树的根结点的 y 坐标值;参数 rx 表示遍历的子树的最右结点的 x 坐标值。其中参数 tx 和参数 rx 也是输出参数。

用算法语言描述如下:

```
treePosition(Tree t,int lx,int &tx,int ty,int &rx) {
    if (!t) return; //如果是空结点,则返回
    s=firstChild(t); //s 指向 t 的第一个孩子结点
    if (!s) { //若 s 为空,则 t 为叶子结点
        tx=lx; //定位 t 的 x 坐标
        if (rx < lx) rx=lx; //令 rx 始终大于或等于 lx
        storeNode(t,tx,ty); //保存 t 的位置信息
    } else { //若 t 为非叶子结点
        i=0;
        //遍历 t 的第一个孩子结点的子树,sx[0] 记录第一个孩子结点的
        //x 坐标值,孩子结点的 y 坐标值由其父结点的 y 坐标值加上一个
        //常数 YINTERVAL 设定
        treePosition(s,lx,sx[0],ty+YINTERVAL,rx);
        while(s=nextChild(t)) { //若 t 还有孩子结点 i++,遍历 t 的第 i
        //个孩子结点的子树,第 i 个孩子结点的子树的最左位置是第 i-1
        //个孩子的子树的最右结点位置加上一个常数 XINTERVAL,sx
        // [i] 记录第 i 个孩子结点的 x 坐标值
            treePosition(s,rx+XINTERVAL,sx[i],ty+INTERVAL,rx);
        }
        //t 的 x 坐标值是 t 的最左孩子结点和最右孩子结点坐标值的二
        //分之一
        tx=(sx[0]+sx[i])/2;
        storeNode(t,tx,ty); //保存 t 的位置信息
    }
}
```

整个树遍历完毕后,根据显示区域大小,按比例调整结点的大小及其坐标值。

参照树布局方法,可实现二叉树和广义表等层次结构的可视化布局。

* 图布局方法。

图布局方法适用于图等数据结构。JVDSCL 提供多种算法实现图的可视化,如二维弹性模型算法^[6]、PrED 算法^[7]、动态图的实时三维可视化的稳定性算法^[8]、基于遗传模拟退火算法的图的三维可视化^[9]。

图布局方法的默认布局算法是二维弹性模型算法。二维弹性模型算法的基本思想是在二维平面上,将一个无向连通图看作是顶点为铁环、边为弹簧的动态系统,而布局算法则是一个逐步减少整个系统总能量的迭代过程。

二维弹性模型算法的基本框架如下:

- (1) 计算图中顶点间的最短路径长度;
- (2) 计算顶点在二维空间中的映象之间的初始弹簧长度;
- (3) 计算顶点间的弹力;
- (4) 初始化顶点在二维空间中的映象;
- (5) 迭代计算系统的总能量,若总能量不小于某一阈

值,调整各顶点的坐标值;

(6) 迭代结束后,根据显示区域的大小,重新调整各顶点的坐标值。

2 JVDSCL 的应用

2.1 在 Java 程序中的应用

在 Java 程序的调试过程中,当开发人员需要知道当前建立的数据结构状态是否正确时,可以通过调试器设置断点,查看内存中数据结构元素的值是否正确。这样开发人员难以得到数据的直观、形象的整体概念,容易造成开发低效。应用 JVDSCL,就可以在需要的时候在独立的窗口中将数据结构用图形显示出来,使开发人员对数据结构的当前状态有一个直观形象的了解,提高开发效率。

在 Java 程序中,首先应用 JVDSCL 构造了一个数据结构对象,然后调用这个数据结构对象的 show() 方法,这样就可以在独立的窗口用图形的方式显示出数据结构的当前状态。

图 5 显示单链表数据结构,有 6 个结点,结点值分别为 a,b,c,d,e,f。

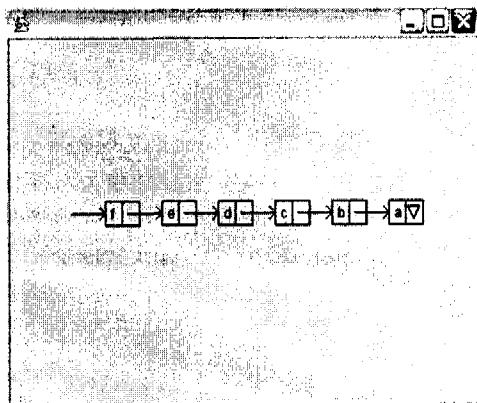


图 5 显示单链表数据结构

在 Java 程序中嵌入以下代码:

```
//创建一个 VLinkedList 对象 VLinkedList list = new VLinkedList
//构造 list
.....
//弹出窗口显示 list
list.show();
```

2.2 在数据结构动态演示系统中的应用

在笔者参与开发的基于算法演示引擎(限于篇幅,详细设计与实现将另文介绍)的数据结构动态演示系统中,JVDSCL 得到了有效的应用。数据结构动态演示系统演示各种算法中不同数据结构逐步变化的过程,实现动态演示。开发人员通常需要实现大量复杂的画图操作,如绘制链表结点,要从点、线开始编码实现,这样软件复用性不高,软件开发效率低。应用 JVDSCL,无需自己编码实现复杂的画图操作,就可以实现各种数据结构的演示动画。JVDSCL 提高了开发动态演示系统的层次,它使开发人员只需考虑操作本身(如链表结点的绘制操作),而不用自己

实现这个操作(不需要从点、线开始编码实现结点的绘制);开发人员也不需要考虑各种数据结构可视化的布局,JVDSCL 本身就提供了基本的布局方法,也可以根据自己的需要,重写这些方法达到个性化的实现。

开发人员只需调用数据结构类中的 draw() 方法,就可以在现有的画图区域实现数据结构的可视化。图 6 演示了后序遍历二叉树。

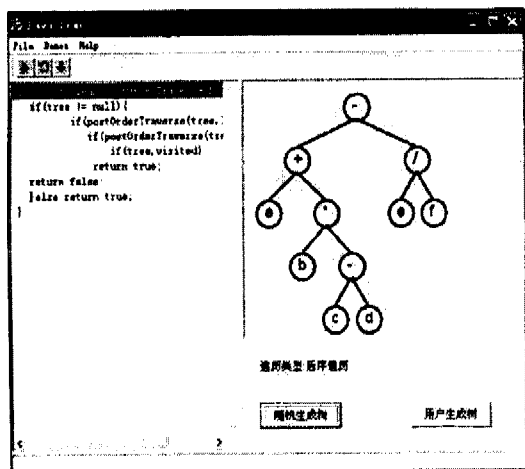


图 6 后序遍历二叉树

3 结束语

实践表明,构造可视数据结构是实现数据结构可视化的一种有效的方法。JVDSCL 为设计相关可视化程序提供了一个可复用的类库,提高了软件开发的效率。现阶段,JVDSCL 只是初步的实现,功能还不够完善。进一步

的工作是完善 JVDSCL 的功能,提高软件的重用性和扩展性。

参考文献:

- [1] Price B A, Baecker R M, Small I S. A principled taxonomy of software visualization[J]. Visual Languages and Computing, 1993, 3(3): 211 - 264.
- [2] Kerren A, Stasko J T. Algorithm animation[EB/OL]. www.ads.tuwien.ac.at/people/kerren/pubs/22690001.pdf, 2002.
- [3] Ossling G R, Freisleben B. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation[J]. Journal of Visual Languages and Computing, 2002, 13(3): 341 - 354.
- [4] Demetrescu C, Finocchi I. Smooth Animation of Algorithms in a Declarative Framework[J]. Journal of Visual Languages and Computing, 2001, 12(3): 253 - 281.
- [5] Ben - Ari M, Myller N, Sutinen E, et al. Perspectives on Program Animation with Jeliot[A]. International Seminar[C]. Berlin: Springer, 2002. 31 - 45.
- [6] Kamada T. Visualizing Abstract Objects And Relations[M]. Hong Kong: World Scientific Publishing Co Pte Ltd, 1989. 69 - 104.
- [7] Bertault F. A force - directed algorithm that preserves edge - crossing properties[J]. Information Processing Letters, 2000, 74(1 - 2): 7 - 13.
- [8] 吴伟民, 樊敏, 曹勇锋. 动态图的实时三维可视化稳定性算法[J]. 计算机工程与设计, 2005, 26(3): 801 - 802.
- [9] 孙炜, 吴伟民, 陈志峰. 基于遗传模拟退火算法的图的三维可视化[J]. 广东工业大学学报, 2002, 19(1): 37 - 41.

(上接第 60 页)

4 基于本架构开发的优点

与传统的开发方法相比,采用基于架构的方法可以使开发人员有共同的参考基础——架构。使得各成员之间的交流变得非常容易^[4]。采用基于架构的方法也具有其他显著的优点:

(1) 编码时间明显缩短。因为大量使用重用机制,可以使编码数量大为减少。

(2) 产品的测试和维护更加容易。由于框架定义了代码规范和大量的程序模板,所以开发很有规律。而且测试的时候可以先对某一模块进行测试,其他模块则根据测试结果进行类似的修改、调整,大大节约了测试时间。

(3) 产品实施的灵活性提高。基于服务的软件部署和配置方案,使管理软件的实施更加灵活。比如:很容易就可以同时支持局域网应用和基于互联网的远程下订单、综合查询等应用。同时,面向服务的架构对界面的多样性支持得很好。

5 结束语

软件架构对中小企业管理软件的功能和质量有着重

要影响,良好的产品架构设计可以从整体上保证管理软件质量,提高软件的灵活性、可扩展性和可重用性^[5]。面向服务的软件架构具有协议开放、易于集成、支持业务流程持续改进等特点,可以作为基于 Internet 的下一代中小企业管理软件架构设计的参考模型。

参考文献:

- [1] Perry D E, Wolf A L. Foundations for the Study of Software Architecture Software Engineering Notes[J]. ACM SIGN-SOFT, 1992, 17(4): 40 - 52.
- [2] Kruchten P B. The 4 + 1 View Model of Architecture[J]. IEEE Software, 1995, 12(6): 42 - 50.
- [3] 柴晓路. Web Services 技术、架构和应用[M]. 北京: 电子工业出版社, 2003.
- [4] Binns P, Vestal S. Formal real - time architecture specification and analysis[A]. In Tenth Workshop on Real - Time Operating Systems and Software[C]. New York: [s. n.], 1993. 125 - 133.
- [5] 柴晓路. Web Services 架构[EB/OL]. http://www.900.cn.ibm.com/developerWorks/cn/webservices/-ws-wsar/index.shtml, 2001 - 12 - 03.