维普资讯 http://www.cqvip.com

图形用户界面系统国际化的研究*)

姜 力 吴 健 孙玉芳

(中科院软件所 开放系统与中文信息处理中心 北京 100080)

摘 要 国际化的GUI系统不仅仅要能够正确地显现多语言文本,而且要能够产生与文字书写方向相应的用户界面。目前主要的GUI库仅支持镜像,还不存在一个GUI框架既支持垂直文字又支持双向文字。在本文中,分析了开发国际化GUI系统过程中面临的问题,给出了一个层次模型来实现动态用户界面风格。

关键词 图形用户界面,国际化,书写方向,用户界面风格,镜像,动态

Research on Internationalization of Graphical User Interface System

JIANG Li WU Jian SUN Yu-Fang

(Open System and Chinese Information Processing Center, Institute of Software, CAS, Beijing 100080)

Abstract International GUI system should not only display multilingual text correctly, but also generate user interface corresponding to writing direction. At present, major GUI libraries just support mirroring, while no framework is existed which support both of vertical script and bidirectional script. In this paper, the problems appeared in the process of constructing international GUI system are analyzed, a hierarchy model is given to implement the dynamic of user interface style.

Keywords GUI, Internationalization, Writing direction, User interface style, Mirroring, Dynamic

1 引言

随着社会的进步、经济的发展,全球化促进了对信息系统国际化的需求。来自不同国家使用不同语言的人参与到共同的活动中,从日常的经济活动到举世瞩目的国际盛会,都要求信息系统能够同时处理多种语言,或者方便地切换界面的语言。而传统的软件开发通常只针对某个区域或某种语言开发,当需要实现对其他语言的支持时再做本地化。但是这种模式不适合实现多语言共存的需求,现在的软件开发认识到在软件设计的初期就应该采用软件国际化的方法。

国际化软件的设计,要求操作系统、窗口系统和编程库都提供对国际化的支持,其中很重要的一个方面就是GUI(Graphic User Interface)库提供的界面显示的国际化。界面显示如果仅仅做到正确地显示文字,只是做到了多语言化,而国际化的要求是应用系统要做到与用户的使用习惯相一致。这就要求GUI系统不仅提供正确显示字符的能力,而且要提供生成符合用户使用习惯的应用程序的能力,也就是要能够支持不同的用户界面风格。

2 语言文字和用户界面风格的关系

用户界面风格的不同是因为用户的使用习惯的差异,使用习惯的差异主要来源于语言文字的书写方向。中国少数民族众多,语言文字形态各异,表现出不同于英语、汉语的多样性,它们之间的区别可以透彻地说明这个问题。

在语言文字的书写方向方面,藏语类似于汉语,从左向右书写(LTR,Left to Right);维文和阿拉伯文是一个语系,是从右向左书写(RTL,Right to Left),其中的数字又是从左向右书写,所以又称为双向文字;蒙古族的文字比较特殊,是采用每一行从上向下书写,在各个行之间是从左向右的排列方式(TTB-LTR,Top to Bottom－Left to Right)。不同种类的文字书写方向影响了用户界面中界面元素的布局和显示。这就要求支持国际化的GUI库提供与文字书写方向相适应的UI(User Interface)布局和UI元素。

用户界面风格于是可以根据不同的文字书写方向分为4种:

1. LTR,从左向右书写一行,从上向下排列不同行。
2. RTL,从右向左书写一行,从上向下排列不同行。
3. TTB-LTR,从上向下书写一行,从左向右排列不同行。
4. TTB-RTL,从上向下书写一行,从右向左排列不同行。

除了前面介绍的3种分别对应现代汉语(英文)、维文(阿拉伯文)、蒙古文;第4种对应传统汉语和日文的书写方法,适用于汉语和日文竖排的书写风格。

3 研究现状

近10年的软件开发对软件国际化的关注程度越来越高,主要的图形库都包含了对国际化用户界面的支持。

目前在国际化方面,Windows操作系统处于相对领先的位置。在本地化的Windows95版本,阿拉伯语和希伯莱语版中,使用了翻转的机制来实现UI的从右到左的风格。翻转机制重新定位了对话框中的UI元素的位置,改变了对话框的布局,但并没有解决标题栏、树视图等UI元素的正确显示。于是从Windows98的希伯莱语版本开始,引入了镜像技术[1]。镜像技术在坐标变换的基础上,设计了新的体系结构和编程接口。这种体系结构和编程接口提供了在不同层次上

*)国家863计划项目(2003AA1Z2110)资助。姜 力 硕士生,主要研究方向:软件国际化;吴 健 研究员,硕士生导师,主要研究方向:中文信息处理、软件国际化;孙玉芳 研究员,博士生导师,主要研究方向:操作系统、中文信息处理。

维普资讯 http://www.cqvip.com

控制镜像的能力，可以在进程、窗口、对话框资源、设备环境的基础上打开镜像功能，也可以在运行时关闭。可以说，Windows 提供了相当完善的体系，以实现对 RTL 界面风格的支持。

在 Java 的 AWT 图形库中，自 JDK1.4 起提供了相当完整的双向文字支持。ComponentOrientation 类封装了这种语言相关的布局方向性，它支持 LEFT_TO_RIGHT 和 RIGHT_TO_LEFT 两种风格[2]。应用程序可以使用 ComponentOrientation 类的方法实现 RTL 的界面风格。

在 UNIX 方面，IBM 的 AIX 操作系统通过支持双向文字的 Motif 图形库实现了 RTL 风格。在 X 的资源文件中，或在函数中使用参数，指定支持双向文字的相关设置。资源 XmNlayoutDirection 通常用来设置整个应用程序的布局方向，它影响 XmLabelGadget、XmPrimitive、XmManager 和与它们相关的控件[3]。受其影响的因素包括界面元素的位置、属性的缺省值、资源的含义和缺省值、人机交互的行为等等。

在 Linux 上，主流的图形库 GTK＋和 Qt 库目前都做到了对不同文字显示的支持，并且在部分 GUI 的控件中提供了不同的界面风格。以 Qt 为例，首先，它的图形库是自适应的。这种采用动态布局的自适应技术根据容器大小、控件最小尺寸和最佳尺寸等几何数据，通过设定的布局管理器算法计算得到最终的布局，可以很好地处理界面元素在多语言环境中尺寸变化后需要重新布局的问题[4]。其次，可以通过在启动应用程序时，传入参数 reverse 来启动进程级别的镜像功能。从目前运行的显示效果来看，仅有部分基础控件实现了镜像功能，例如按钮、菜单等已经实现了布局的镜像，但是在其他的控件中还存在许多问题。例如，在 ComboBox 中，镜像后的文字位置不对；从 QScrollView 类派生的 QTable、QIconView、QListView 等都没有实现显示内容的镜像。通过对 Qt 图形库内部代码的分析，发现在不同的控件中实现镜像的思路和方法不同，没有一致的解决方案，无法形成一个完整的显示处理体系。

综上所述，可以看到各种 GUI 系统都不同程度地提供了符合双向文字的镜像风格（RTL）。Windows 的国际化支持程度是相当好的，但是它存在一个缺陷，即它的 GUI 系统不是自适应的，在设计国际化程序时，要事先考虑为用户界面中的文字预留足够的空间。如果空间不够，则会造成文字被截断的后果。Linux 则整体发展较晚，国际化的进程相对滞后，图形库对 RTL 风格的支持还很不够，尤其是没有一个完整的框架。同时，在所有这些 GUI 系统中，都没有对垂直风格（TTB-LTR，TTB-RTL）的支持，在这方面完全是一片空白。在 Java 的 ComponentOrientation 类的参考手册中有提到 4 种不同的方向性，遗憾的是没有实现垂直风格。

针对 Linux 系统上的 GUI 库的不足，本文提出一个新的模型，使之可以支持动态界面风格，成为设计和实现国际化 GUI 库的基础。由于目前的 GUI 库的设计原理各不相同，在支持不同 UI 风格方面很不成熟，首先要分析可能面对的问题。

4 面临的问题

在国际化的应用程序中，支持不同的 UI 风格，是一个相当新颖的课题。在研究这个课题的过程中，将会面对许多问题。这些问题涉及以下方面：

1. 用户界面风格的差异；
2. 用户界面风格的应用层次；
3. 用户界面风格的动态切换；
4. 应用程序编程接口。

4.1 用户界面风格的差异

不同风格的界面元素在大小、布局、组成和交互行为几个方面存在很大的不同。

影响界面元素尺寸大小的主要因素是语言文字。不同语言版本的程序要翻译应用程序界面中的文字为本地文字，而不同语言文字表述相同含义的词汇往往在长度上存在差异。举例而言，一个法语的句子通常是一个等价的英语句子的 1.5 倍长度[5]。于是，界面元素要根据不同的语言文字改变占用的空间大小。

对于镜像风格而言，界面元素大小的改变源自界面语言的变化。对于垂直风格而言，即使界面文字不变，所有的界面元素都将作出大小的改变。宽度不再取决于显示的字符串的长度，而是字符串的行数，高度则相反。

不同的界面风格最明显的差异在于布局。布局不仅包括容器中控件之间的布局，也包括控件内部界面元素的布局。

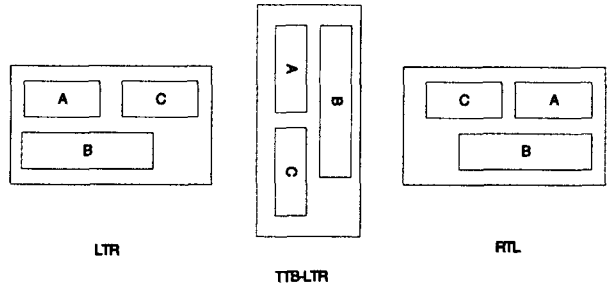


图 1 不同界面风格的差异

图 1 展示了 3 种不同风格的布局特点。通常，应用程序采用 LTR 风格，而其 TTB-LTR 版本和 RTL 版本的界面元素的大小和布局则发生了显著的改变。

对于镜像风格，布局的改变表现为由原来的左对齐转变为右对齐；对于 TTB-LTR 风格，由原来的左对齐转变为顶部对齐和左侧对齐；对于 TTB-RTL 风格，则是顶部对齐和右侧对齐。容器中的控件对齐于容器，控件中的文字则对齐于控件中的文本编辑框。

除了大小和布局的不同，控件的组成也可能发生改变，所有具有方向含义的界面元素都需要替换。例如，ComboBox 在 TTB-LTR 风格中要改变下拉箭头的方向，原来是向下的箭头要用向右的箭头替代。随之控件的行为也发生改变，点击 ComboBox 的箭头，响应是向右弹出列表框。

界面元素中的位图也很可能需要改变，可以分几种不同情况处理。对于那些高宽相同的图标，通常不改变；对于含有方向性含义的或高宽大小不同的图形，最好的解决方案是提供多个图像资源，以备在不同的风格中使用。

4.2 用户界面风格的应用层次

在 Windows 对镜像风格的支持中，有一个优点就是可以在不同的层次级别上开启镜像，而 Java 更是将方向性作为控件的属性。

显然，将方向性作为控件的属性是一个合理的选择。根据控件的方向属性，GUI 库在内部实现中使用不同的资源，生成不同风格的外观，并采用不同的代码以产生符合风格的交互行为。将显示风格局部于控件层面上，可以实现在一个

维普资讯 http://www.cqvip.com

进程、窗口、对话框中不同风格的控件共存，甚至运行时改变控件的方向属性，以应对复杂的多语言环境的应用。

4.3 用户界面风格的动态切换

有研究表明，若新的技术对于不同的人群具有不同的可访问性，则会造成人群的分化，懂得技术的人更强，而弱者对新的技术有排斥心理。开发国际化应用程序的目的就是要提供给所有用户相同的可访问性[6]。界面风格的动态切换功能，就可以让一个程序提供给各种人群相同的访问能力。

国际化的应用程序是单一二进制的，在不同的环境中，通过判断当前语言环境或指定的参数在运行时决定程序的界面风格。GUI 库根据参数和在设计时指定的控件方向决定运行时的程序界面显示效果，这样可以方便地将一个 LTR 风格的程序转变成为 TTB-LTR 或 RTL 风格。应用程序员在设计时可以仅针对 LTR 风格设计程序，GUI 库对其他风格的支持对程序员而言是透明的。当前英文和中文软件十分丰富，但是维文和蒙文的软件极其匮乏。通过参数变换界面风格，辅之以多语言资源，可以极大地改善少数民族语言软件匮乏的面貌。

4.4 应用程序编程接口

大多数控件的接口是与具体的坐标和大小信息无关的，但是类似 ScrollView 的控件需要管理其内部的控件位置，在控件的编程接口上必须提供位置参数。显然，应用程序编程接口要尽可能地设计为不使用具体的坐标。对于需要位置信息的控件，存在位置参数提供的信息是运行时位置还是设计时位置的问题。前面提到可以通过切换界面风格构造程序适应新的语言环境。基于这一原则，程序设计时并不明确将来运行时的环境，参数值是基于设计时的，所以 GUI 库在运行时要将参数映射到实际的坐标系中。

5 模型的设计与实现

5.1 模型的基本轮廓

基于以上的探讨，可以形成对模型的基本轮廓：GUI 库提供标准的接口给程序员，接口包括将代表风格的书写方向作为控件的属性，程序员可以通过控件的方向属性指定设计时控件的风格。通过环境、参数改变程序运行时控件的风格，GUI 库内部透明地实现动态界面风格。

模型的关键就在于将设计时控件的风格在运行时动态地转换为所需的界面风格。

5.2 模型的原理分析

GUI 库的基础是底层的绘制函数，这些绘制函数是与具体操作系统和窗口系统相关的。以跨平台的 Qt 为例，底层可以是 Windows 或 X Window，具体的绘制函数的设计都是基于 LTR 的坐标系的。作为高层图形库，不可能也不应该修改底层的窗口系统。同时，设计时的坐标系是由程序员指定的。要支持动态界面风格，就需要完成不同层次的坐标转换和计算。

图 2 展示一个设计时 LTR 风格的表格控件在运行时动态切换为 RTL 风格过程中的 3 个坐标系。在设计时，表格是位于一个坐标原点在左上角的坐标系中；在计算 RTL 风格的显示效果时，将原来的坐标原点映射到右上角；在调用底层的绘制函数前，要将控件的几何信息映射到实际的显示坐标中。这 3 个坐标系分别对应了程序员设计时的视觉平面、用户运行时的逻辑平面和实际底层函数的绘制平面。

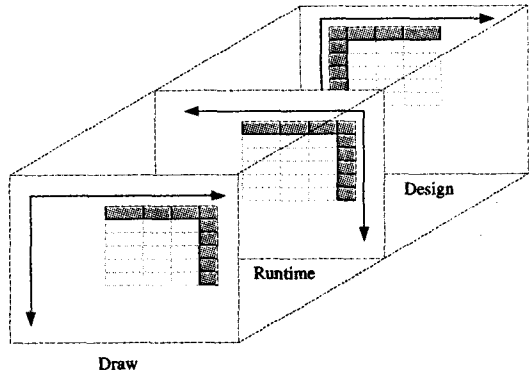


图 2 Table 控件的 3 层坐标系

5.3 层次模型

国际化 GUI 系统模型是根据坐标变换原理设计的层次模型，如图 3 所示。

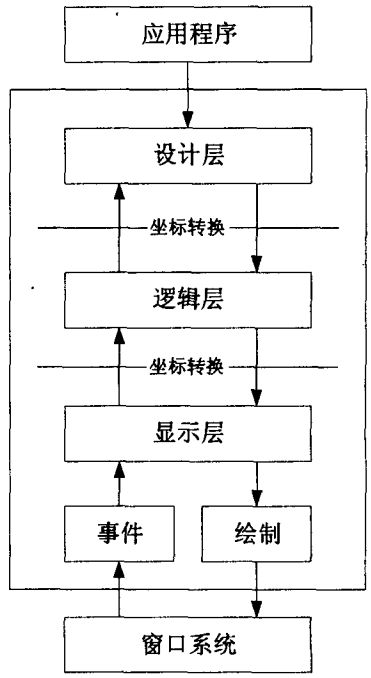


图 3 国际化 GUI 库模型

以下是层次模型的描述：

1. 应用程序指定设计时控件、容器的逻辑布局和接受切换界面风格的参数，传递给 GUI 系统。

2. 设计层对应设计时的坐标系，体现控件设计时的风格。

3. 逻辑层对应运行时的坐标系，体现用户实际需要的界面风格。针对容器，采用应用布局管理器计算控件的布局；针对控件本身，采用不同算法计算得到所需要的风格。逻辑层计算得到的结果就是最终的显示效果。

4. 显示层对应绘制时的坐标系，逻辑层的运算结果转变为显示层的坐标系后，由基础的图元绘制函数在绘图设备上实际绘制出来。

5. 来自窗口系统的事件信息经过不同的坐标系转换后，用于不同层次的运算。

6. 窗口系统提供基本的绘制功能，高层的 GUI 库将其封装为图元绘制函数，实现平台无关性。

（下转第 134 页）

维普资讯 http://www.cqvip.com

参 考 文 献

1 殷新春，严芬，汤克明，等．基于CMM的软件质量保障平台定义框架．小型微型计算机系统，2002，23(11)

2 Stratton R W. The EVM Maturity Model-EVM^{3TM}. 2002

3 Management Technologies. The Earned Value Management Maturity Model. 2002

4 赵昊翔，潘金贵．基于EVMS的软件过程改进．计算机科学，2004

5 Solomon P. Using CMMI to Improve Earned Value Management. CMU/SEI-2002-TN-016

6 Solomon P. Going From PBEV to CMMI. The Magazine of the Project Management Institute's College of Performance Management. March 2002

7 林锐，王慧文，董军．CMMI 3级软件过程改进方法与规范．北京：电子工业出版社，2003

8 McFeeley B. IDEAL: A User's Guide for Software Process Improvement. CMU/SEI-96-HB-001

9 万江平，郭荷清，夏志忠．IDEAL模型的研究．计算机应用研究，2001

10 Humphrey W S. The Team Software Process^{SM}(TSP^{SM}): [CMU-SEI-2000-TR-023]

(上接第130页)

模型根据控件的方向属性和切换界面风格的参数，采用不同的逻辑层坐标系，选择不同的算法，从设计层的设计时数据计算得到逻辑层的运行时数据，再转换到显示层，调用绘图函数绘制最终显示结果。控件对事件的响应，则先将事件信息转换成逻辑层或设计层信息，运算得出响应行为，最终由显示层表现出来。

5.4 布局管理和控件风格

模型的核心是逻辑层的布局管理和控件风格的运算，它们产生最终的用户界面。

现在的许多GUI图形库中都引入了布局管理的概念。在容器中，布局管理器负责动态地调整容器中控件的位置和大小。

控件内部的界面元素的位置则不是由布局管理器定位的，完全是依赖坐标系。布局管理器负责定位控件，控件负责绘制自身。实现动态的界面风格就是采用不同的绘制算法绘制控件。

虽然布局管理器也能完成类似坐标变换的功能，例如用右对齐的布局就可以模拟RTL风格。但是，由于动态界面风格的本质是坐标系的变换问题，而且布局管理器也无法管理控件内部的界面元素，因此层次模型采用坐标系变换的原理是从本质上解决动态风格的实现问题。

另外，基于坐标系变换带来的优点是层次模型也适用于没有布局管理机制的GUI图形系统。

5.5 实现

本文提出的层次模型已应用于扩充Qt3.3图形库，实现支持少数民族语言的GUI系统。在层次模型的实现过程中，我们构建新的设计结构，改变了原有设计中结构和数据定义的含义，在不同函数中体现层次模型的作用，修改和添加新的算法实现动态界面的计算和显示。

图4展示了Qt库的例子程序smalltable在扩充后的GUI库的支持下，动态切换为符合蒙文的TTB-LTR风格。实现动态界面的算法重新计算了Table中的TableHeader、TableItem等界面元素的位置和大小，修改了交互行为，以符合TTB-LTR界面风格的要求。

原始的Qt图形库没有完整地考虑国际化UI风格设计，内部许多算法仅仅对LTR风格是有效的，也缺乏层次概念，实现镜像功能的代码更多地体现为实现的技巧。相比而言，改进后的图形库保留了自适应的特点，同时实现了模型中的多个层次，形成一个完整的结构。

总结与展望 本文提出的层次模型结合坐标变换和自适应的布局管理，动态实现不同的界面风格。建立在控件层的方向属性使得程序可以呈现丰富的国际化特性。运行时改变界面风格的能力，使得可以快速地发布应用程序的国际化版本，这一点对于少数民族的信息化工作具有重大的意义。

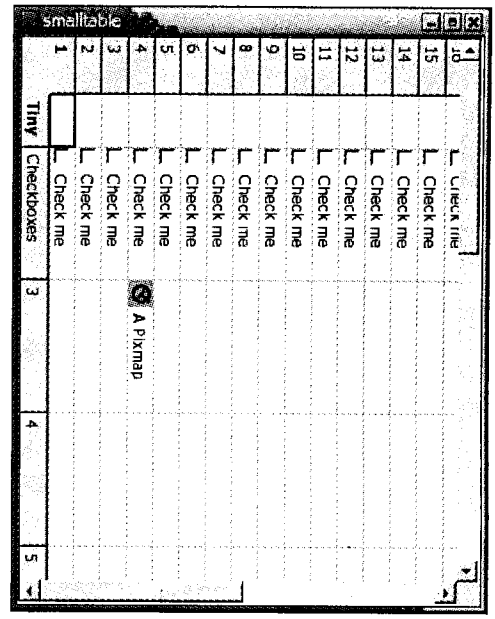


图4 TTB-LTR风格的Table

现有的实现是从实用性的角度出发，考虑兼容性，在原来的Qt图形库的基础上扩充代码实现层次模型。于是，不可避免地要部分受制于原有的框架，代码的复杂程度较高，这是目前模型实现的不足之处。进一步的研究和开发工作将是规范应用程序图形编程接口，清晰化内部层次结构，降低实现的复杂度，推动图形库向有利于模型实现的方向进化。

参 考 文 献

1 Dr. International. Developing International Software. Second Edition. Microsoft Press, 2003

2 Sun Microsystems, Inc. JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification

3 Charles P. Introduction to bidirectional languages: Bidi and AIX. IBM Corporate Globalization Team, Globalizing your e-business Newsletter - Issue X, June 2004. http://www-306.ibm.com/software/globalization/topics/bidiaix/index.jsp

4 Kurlow S. A new object-oriented technique for building a dynamic Graphical User Interface. In: Proc. of the 1995 Conf. of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada, Nov. 1995

5 Landau J. Testing for Globalization. In: PSQT & PSTT East Conference on Washington, 2004

6 Stephanidis C. Toward an Information Society for All: An International R&D Agenda. International Journal of Human-Computer Interaction. 1998, 10(2): 107～134