基于 Uniscribe 和 OpenType 的蒙古文字处理软件 MWord 的设计与实现*

斯·劳格劳[1] 华沙宝[2] 萨如拉[3]
内蒙古大学 蒙古学学院 呼和浩特 010021
E_mail: sloglo@sina.com

摘 要：MWord是一种集文本、表格、图形、图象处理于一身的字处理软件，它采用OpenType字库技术和Uniscribe布局引擎解决了Windows环境下蒙古文编码国际标准的实现问题。本文重点介绍了MWord的内部数据模型、文档格式（外部数据模型）以及采用OpenType字库和Uniscribe布局引擎后蒙古文文本的编辑、解释、排版和输出技术。
关键词：蒙古文，复杂文本，Unicode，OpenType，Uniscribe

Realization and Design of Mongolian Word Processor Based on Uniscribe and OpenType

S. Loglo[1] HUA Sha-bao[2] Sarula[3]
College of Mongology, Inner Mongolia University, Hohhot 010021,China
E_mail: sloglo@sina.com

Abstract: MWord is a word-processing software, which can edit and layout text, tables, graphics and images. After uses OpenType font technolog y and Uniscribe lay out eng ine, it has solved the Mong olian cod ing und er the Windows environment. In this paper mainly introduce the MWord's internal data model, document format (external data model) and the editing, interpreting, typesetting and output technologies of Mongolian text, after using OpenType fonts and Uniscribe layout engine.
Key Words: Mongolian, Complex Text, Unicode, OpenType, Uniscribe

1 前言

蒙古文编码国际标准只定义了名义字符，对变形显现字符没有提供固定码位，在 Windows 环境下，编码的系统实现必须采用 OpenType 字库技术。而现有的蒙古文字处理软件和编辑排版系统并不能有效处理 OpenType 字库[1]。鉴于上述情况，我们设计开发了基于 OpenType 字库技术和 Uniscribe 布局引擎的蒙古文字处理软件 MWord(Mongolian Word)和智能输入法 MSIM(Mongolian Smart Input Method)。MWord 和 MSIM 是系列软件，其中 MSIM 解决了蒙古文编码国际标准的录入问题，而 MWord 解决了蒙古文复杂文本的解释、编辑、排版与输出。对于 MSIM，2007 年 5 月份已经发布在内蒙古大学蒙古学中心网站(http://mgxzx.imu.edu.cn)上，可以免费

作者简介：斯·劳格劳(1972~)，男，蒙古族，内蒙古人乌审旗人，讲师，博士研究生，研究方向为蒙古文信息处理. 华沙宝(1950~)，男，蒙古族，教授，博士生导师，研究方向为蒙古文信息处理. 萨如拉(1975~)，女，蒙古族，博士研究生，副教授，研究方向为中国少数民族语言文学.

下载使用。并在“基于Uniscribe和OpenType的蒙古文智能输入法”（内蒙古大学学报哲学人文社会科学蒙文版2007年第5期）一文中对其实现方法做了详细介绍，在此不再赘述。本文主要介绍MWord的数据模型和实现方法。

2 数据模型的设计

在许多类型的程序设计中，数据结构的选择是一个基本的设计考虑因素。很多大型系统的构造经验表明，系统实现的困难程度和系统构造的质量都很大程度上依赖于是否选择了最优的数据结构。很多时候，确定了数据结构后，算法就容易得到了。字处理软件的数据可以分为运行时的内存数据和存取时的外存数据。其中，内存数据与处理过程和所采用的算法有关，而外存数据与文档格式有关。

2.1 内部数据模型

文本是字处理软件的处理重点，根据其所携带的格式信息可以分为纯文本（Plain Text）和格式化文本（Rich Text Format）两种。纯文本没有任何格式信息，在处理过程中选用一个合适的集合类来存储字符编码就可以了，而格式化文本具有字体、字号、段落和颜色等多种附加的格式信息，必须选用不同的数据结构来存储和处理。

另外，由于采用的编码标准和字库技术不同，文本还有简单和复杂之分。在传统的TrueType字库条件下，文本的编辑、排版和输出主要以字符*为处理对象，因此采用的数据结构和算法都比较简单。而在Unicode标准和OpenType技术条件下，文本处理需要编辑、排版和输出以外还要通过解释（关于文本解释详见本文第3.2节）生成字型**数据，并且几个处理过程所需的数据也不相同，编辑和解释主要以字符为处理对象，排版和输出主要以字型为处理对象。总而言之，文本所采用的编码标准、字库技术以及所携带的格式信息决定了字处理软件内部数据模型的复杂程度。

我们在MWord的设计中参考了OpenOffice.org等优秀开源软件的数据模型，并对计算机的处理能力综合考虑后采用了以双向链表为主要形式的数据描述方法。MWord 文本数据大致可以分为编辑类数据、解释类数据和排版、输出类数据。各类数据间的关系如图1所示。

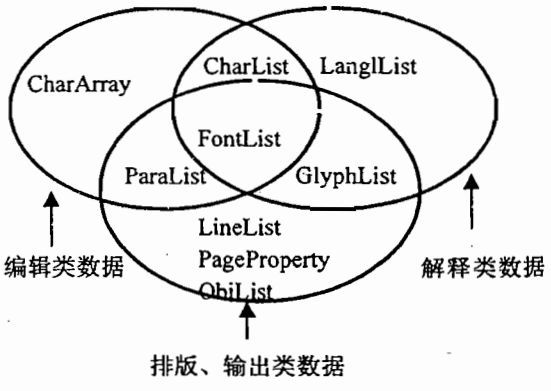


图1 MWord内部数据模型

Fig.1 The internal data model of MWord

其中，编辑类数据中包括字符数组CharArray(用于字符串查找)、字符链表CharList(用于字符的插入/删除)、段链表ParaList和字体链表FontList。解释类数据中包括字符链表CharList、字型链表GlyphList、字体链表FontList和语言链表LangList。解释算法依据FontList、CharList和LangList生成GlyphList。排版、输出类数据中包括段链表 ParaList、字型链表

* 在此“字符”指字符编码，对于蒙古文OpenType字库而言，指名义字符。

** 在此“字型”指通过USP布局引擎的选型函数得到的字型ID号，对于蒙古文OpenType字库而言，指变形显现字符。

GlyphList、字体链表 FontList、行链表 LineList、版心数据 PageProperty 和插入式对象链表 ObjList。

通过对 MWord 的系统测试发现上述数据描述方法是可行的，并且是高效的，它降低了编辑、解释、排版和输出算法的实现难度，同时也提高了程序的运行速度。

2.2 外部数据模型（文档格式）

在蒙古文字处理领域中，除编码系统以外文档格式也是防碍信息交换和数据共享的主要因素[2]，封闭的文档格式威胁着数据安全。我们在研发 MWord 时参考了已有的北大方正书版系统的注解格式和 XML 标记语言，并根据蒙古文自身特点和 OASIS（结构化信息标准促进组织）的相关要求[3]后制定了如下的文档格式。这是一种开放的、可扩充的文档格式。

1）版面格式{ 纸张高，纸张宽，起始页码，页码位置，页码修饰，页眉说明，页脚说明，分栏说明，默认字体，默认字号，默认行距，文字方向 }

2）段落格式{ 对齐方式，首行缩进，悬挂缩进，上缩进，下缩进，行距，段前距，段后距 }

3）字符簇格式{ 字体，字号，斜体，粗体，下画线，字符间距，前景颜色，背景颜色 }

4）表格格式{ 位置，表线格式，表元格式，背景颜色 }

① 表线格式{ 起点位置，终点位置，线宽（线型号），线颜色 }

② 表元格式{ 位置，高度，宽度，文本方向，背景颜色（背景颜色，背景图片文件名），表元内容格式 }

5）图片格式{ 文件名，位置，高度，宽度，亮度，对比度，原图高度，原图宽度，原图亮度，原图对比度 }

6）图形格式{ 图形种类号，位置，控制点位置 1，控制点位置 2，…，控制点位置 n，高度，宽度，填充颜色，边框颜色，边框宽度 }

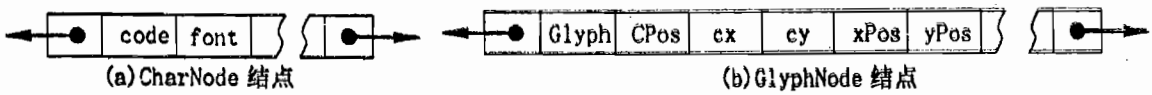
7）文本框和图文框格式{ 位置，高度，宽度，背景颜色（背景颜色，背景图片文件名），文本方向，边框类型（线型号），边框宽度，边框颜色，内容格式 }

在 MWord 中，内外数据模型是这样联系在一起的：打开文档时把外部数据模型转换为内部数据模型，初始化成与编辑、解释、排版和输出相适应的数据结构。保存文档时将内部数据模型转换为外部数据模型，以便于使用其他编辑软件查看和修改。用户可以使用任何一个编辑软件来建立和修改 MWord 文档，只要其格式符合 MWord 的文档格式就可以。

3 文本处理

3.1 文本编辑

字处理软件中，文本编辑是指单个字符的插入/删除以及文本块的粘贴/删除等操作。完成上述操作时首先要通过键盘和鼠标进行光标定位和文本选择，找到精确的字符位置或范围后才能实施编辑操作。下面给出了在文本编辑中用到的字符链表（CharList）和字型链表（GlyphList）的结点结构（如图 2 所示）以及光标定位、插入/删除操作的算法描述（如算法 1、2 所示）。



(a) CharNode 结点　　(b) GlyphNode 结点

图 2 CharList 和 GlyphList 的结点结构

Fig. 2 The node structure of the CharList and GlyphList

说明：图 2 中只画了 CharNode 结点和 GlyphNode 结点的部分字段，箭头指向链表中的前后结点。图 2(a)中："code"为 TCHAR 类型，表示字符编码；"font"为指针类型，指向 FontList 中与该字符对应的字体结点。图 2(b)中："Glyph"为 WORD 类型，表示字型 ID 号；"CPos"指针类型，指向 CharList 中与该字型对应的第一个 CharNode 结点；"cx"和"cy"为整数类型，分别表示该字型的宽度和高度；"xPos"和"yPos"为整数类型，分别表示该字型的输出位置的横坐标和纵坐标。

```
Void SetCaretPosition(光标定位键或鼠标位置){
if(鼠标操作){
DPToLP(鼠标位置);                  // 把鼠标位置从客户区映射到文档的逻辑坐标上。
LPToGlyphpos (鼠标逻辑坐标);  // 把鼠标逻辑坐标映射到字型上。文本经过排版后字型链表
                                   //(GlyphList)每个结点
                                   // 中保存着当前字型的横坐标和纵坐标，LPToGlyphpos()函数通过比较标逻
                                   // 辑坐标与字型的横坐标和纵坐标，找到 GlyphList 中与鼠标位置最近的那
                                   // 个字型结点。 }
if(键盘操作){ 对 GlyphList 调用 (n 次) GetPrev 或 GetNext 函数，找到需要定位的那个字型结点。}
SetCaretPosition(GlyphNode);  // 通过字型结点的 xPos 和 yPos 值定位光标。
Return; }
```

算法 1 光标定位算法

Algorithm 1 The cursor positioning algorithm

```
Void EditCharacter(BackSpase||delete||字符键){
Point=GetCaretPosition(void);  // 获得当前的光标位置。
CaretPosToGlyphPos(光标位置); // 通过比较光标位置的横、纵坐标与 GlyphNode 结点的 xPos、yPos
                                   // 值得到当前 GlyphNode。
GlyphPosToCharPos(GlyphNode); // 文本经过解释后 GlyphList 的每个结点中存放着与其对应的第一字
                                   // 型符结点 (CharNode)的链表位置(CPos),通过 CPos 得到与光标位置相对应
                                   // 的字符。
if (nChar=BackSpase||delete){删除字符;}
else{插入字符;}}
```

算法 2 字符编辑算法

Algorithm 2 The text edit algorithm

在粘贴/删除等操作中，文本选择的基本操作是光标定位。确定起始位置后，在选择过程中进行多次 SetCaretPosition (GlyphNode) 函数来确定终止位置。当起止位置已定的情况下循环调用相应次数的 EditCharacter (字符键) 函数就能实现粘贴/删除等块操作。

3.2 文本解释

文本解释是指在 Unicode 标准和 OpenType 技术条件下从字符(Character)生成字型(Glyph)的处理过程[4,5]。我们在 MWord 中采用了 Microsoft 公司研制的布局引擎 Uniscribe (简称 USP) 对蒙古文复杂文本 (采用蒙古文编码国际标注和 OpenType[6]字库的文本) 进行了解释。MWord 的文本解释主要由下面的三个过程来实现：

1) ScriptItemize 函数依据语言特点和文本方向将字符串分解成多个条目 (Item)，这样产生的 Item 中每个字符具有相同的语言和方向属性。

2) 再依据语言特点、文本方向和字体属性将每个 Item 进一步分解为多个片段 (Run)，这样产生的 Run 具有特定的格式信息。

3) 每个 Run 中存在许多字符簇，USP 的 ScriptShape 和 ScriptPlace[1]函数以此为单位进行选

型和置位[7,8]。字符簇是指有紧密联系的一组字符，必须同时参与选型和置位，比如蒙古文控制符和被控制字符。

在具体处理时我们已单词为单位对蒙古文Unicode字符进行了解释，算法流程如图 3 所示。为了处理上的方便，对非蒙文字符也建立了一个字型结点，但这种字符的字型不是通过解释得到的，而是从字符直接产生的。

文本解释生成或刷新了 GlyphList 链表，并填充了每个结点的 Glyph、CPos、cx、cy 等属性字段，对于 xPos 和 yPos 将在排版过程中填写。

文本经过解释后生成字型数据，由字型代表字符参与所有显示、定位、插入、删除、选择和格式化等操作。因此，字型变化必须实时地转换成字符变化才能保证字符数据的安全性。字符与字型之间存在着复杂的对应关系（如图 4 所示）。下面，用一个简单的例子来说明之。文本片段："…Hi 你好 ᠮᠣᠩᠭᠣᠯ …"

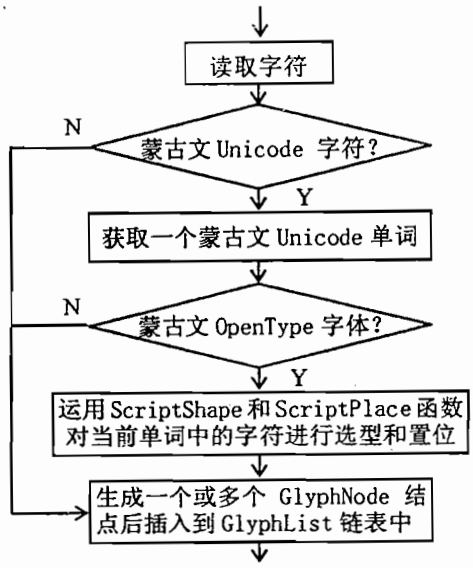


图 3 文本解释流程

Fig. 3 The process of text interpreting

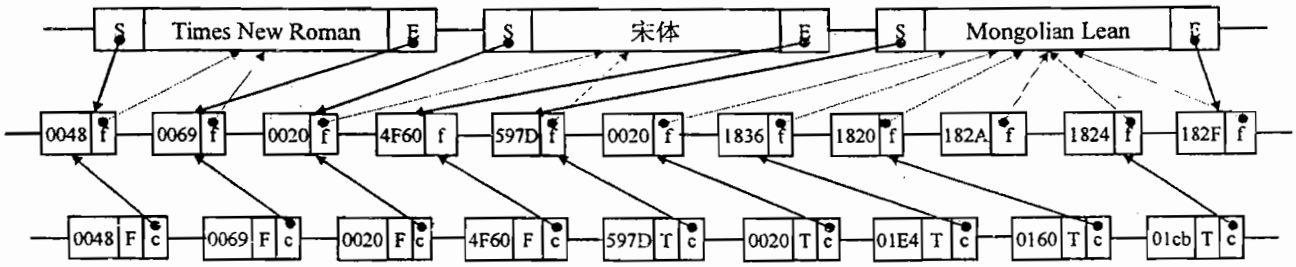


图 4 字体、字符与字型之间的关系

Fig. 4 The relations between fonts, characters and glyphs

说明：第一行表示字体链，其中 S 表示首字符位置，E 表示尾字符位置，字体名表示逻辑字体；第二行表示字符链，其中数字表示字符编码，f 表示字体位置；第三行表示字型链，其中字型结点包括字型高度、宽度、横坐标、纵坐标、字型值、是否复杂文本字符、字符位置等 7 个域，在此仅给出其中与字符有关的三个域。数字表示字型值，F/T 表示是否复杂文本字符，c 表示字符索引。

3.3 文本排版

在"所见即所得"的"傻瓜"文字处理软件中排版是指页面布局发生变化时重新计算并输出页面的过程[9,10]。MWord 排版算法的主要思想是：先通过文本解释获得每个字型的高度和宽度，然后依据面设置、纸张大小、插入式对象的位置和大小、段落格式进行计算得到每个文本行的起止字型、行宽、行高和输出位置，并创建或刷新行链表。排版算法与数据结构之间的关系如图 5 所示。

当系统接到页面刷新指令时按照行链表中的描述信息显示字型数据。排版主要由行链表的刷新模块 Update 来完成。Update 采用了一种局部刷技术，每当页面布局发生变化时通过下面的

三条规则计算获得一个最小的刷新区域。

1） 选取因当前操作受影响的第一行的前一行作为刷新的起始行。如果第一行或全部文档受影响，选择第一作为刷新起始行。

2） 选取因当前操作受影响的最后一行，如果这一行在窗口右边界以内，则用它作为刷新结束位置，否则用窗口右边界处的行作为刷新的结束位置。

3） 文档初始化时，选取第一行作为刷新的开始位置，窗口右边界处的行作为结束位置。

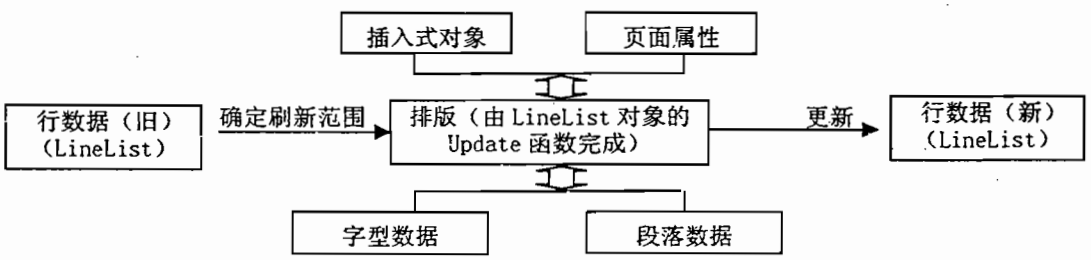


图 5 排版算法与数据结构之间的关系

Fig.5 Relations between data structures and layout algorithm

3.4 文本输出

每当文档内容或版面布局发生变化时必须重新绘制失效区域（Invalidated Region）。而绘制区域的选择是影响输出质量和响应速度的关键因素。可以按字型、单词、文本行、段落或全文为单位选择刷新区域，按字型选取时能够做到必须刷新的区域和实际刷新区域的完全一致，但算法的时间开销很大。最简单的方法是每次输出全部内容，而对于“所见即所得”字处理软件，这是不可思议的，尤其是文档内容达到一定长度后操作延时显得特别明显。为了提高 MWord 的页面刷新速度，我们在设计中采取了如下的区域选择算法：

1）因当前操作受影响的第一个字符所在行的前一行的行首字型作为起始位置。

2）因当前操作受影响的最后一个字型所在自然段的段尾字型作为终止位置。

3）当前区域与窗口区域取交叉区域，两者共有的部分为新的刷新区域。

具体绘制工作由 LineList 对象的 Draw 模块来完成。因字符对象的性质不同（是否复杂文本字符），而选用的输出函数也不同，普通字符用设备描述表的 TextOut 函数，而复杂文本字型则用 USP 的 ScriptTextOut 函数输出。

由于输出函数不断地刷新背景并在新位置重新绘制图形，因此屏幕给人的感觉是在不断的闪烁，这种闪烁给人眼造成很大的不适。我们在视图窗口的 OnDraw（CDC *pDC）函数中采用了一种叫“内存画图”的输出技术后消除了屏幕的闪烁现象。具体方法如下：

先定义两个局部变量：CDC dc; CBitmap bitmap;

如果当前输出不是针对打印机（通过 CDC:: IsPrinting 的返回值来判断），那么为局部变量 dc 生成与 pDC 相容的内存设备表（CreateCompatibleDC）。同时为另一个局部 CBitmap 类对象 bitmap 生成与 pDC 相容的内存位图。生成的内存位图的大小与当前裁剪区域相同，裁剪区域由 CDC:: GetClipBox 函数获得。内存设备描述表 dc 与 pDC 应具有相同的映射模式。当绘制输出是针对屏幕时，把图形绘制在内存位图上。可调用 CDC:: BitBlt 函数将已绘好的位图复制到屏幕上。

修改了视图窗口的 OnDraw 函数还不能消除闪烁，还必须阻止窗口客户区背景的自动刷新。

修改视图窗口的 WM_ERASEBKGND 消息处理函数，让它总是返回 TRUE 就可以。Windows 根据 OnEraseBkgnd 的返回值来决定是否重绘窗口客户区背景。如果 OnEraseBkgnd 的返回值为 TRUE，Windows 将不重绘窗口背景。

4 结语

MWord 是用 UML（Unified Modeling Language）设计，用 VC.Net 开发完成的，集文本、表格、图形、图象处理于一身的字处理软件（如图 6 所示），程序代码约有 50，000 行。目前，系统的主要功能模块已全部实现，正在进行经测试和完善。由于篇幅我们在本文中没有介绍表格和图形图像处理技术。

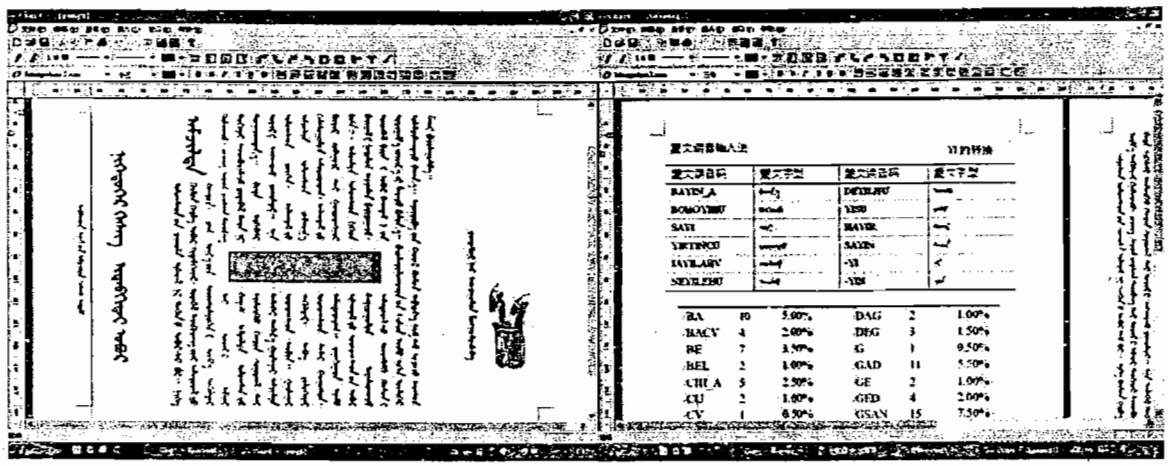


图 6 MWord 的编辑、排版界面

Fig. 6 The edit and layout interface of MWord

参 考 文 献

[1] 姚延栋, 吴健, 孙玉芳等. 传统蒙古文变形显示机制研究与实现[J]. 中文信息学报, 2005, 18(5):84-89.

[2] 嘎日迪, 赵小兵, 刘彦文等. 多民族文字信息处理的计算机平台研制[J]. 内蒙古师范大学学报(自然科学版), 2000, 29(3): 185-190.

[3] 路广. 办公软件呼唤开放的文档格式标准[EB/OL]. http://www.OpenOffice.org, 2004 年 12 月.

[4] 董治江, 吴健, 钟义信. 在 ICU 中实现少数民族文字的处理[J]. 中文信息学报, 2004, 18(2):66-72.

[5] 芮建武, 吴健, 孙玉芳. 基于ISO/IEC10646 标准的藏文操作系统若干问题研究[J]. 中文信息学报, 2005, 19(5):59-66.

[6] Microsoft Corporation. OpenType Specification[EB/OL]. http://www.microsoft.com /typography/ otspec/.2009.

[7] Microsoft Corporation. Uniscribe Introduction[EB/OL]. http://www.microsoft.com/ typography/ developers/ uniscribe. 2000.

[8] Microsoft Corporation. Uniscribe API's[EB/OL]. http://www.microsoft.com/ typography/ developers/ uniscribe. 2000.

[9] 侯晓阳. "傻瓜" 文字处理系统排版算法的设计与实现[J]. 微机发展, 1996(1):54-56.

[10] 高光来, 侯宏旭, 李彦等. 蒙汉混排文字处理系统的研究与设计[J]. 内蒙古大学学报(自然科学版), 1994, 25(6):685-688.