在 VxWorks 系统中使用 TrueType 字库

孙　枫,陈业夫,郭勇鹏
(哈尔滨工程大学 自动化学院,黑龙江 哈尔滨　150001)

摘　　要:为了得到高质量而且灵活(包括对字形的各种变形操作)的字符显示,同时又能实现所见即所得(WYSIWYG)的打印,传统的使用点阵字库的方法已经不能满足要求.有一个很好的方法——使用 TrueType 字库,可以同时解决以上两个问题.介绍了 TrueType 技术的基本原理及在 VxWorks 系统中如何通过 FreeType 应用 TrueType 字库.在实际应用中,该技术很好地解决了上述问题.但这种技术也可以很容易地移植到很多其他系统中,如 Linux.

关　键　词:VxWorks;TrueType;UGL 字体驱动;打印
中图分类号:TP315　**文献标识码**:A

Using TrueType font in VxWorks system

SUN Feng, CHEN Ye-fu, GUO Yong-peng
(School of Automation, Harbin Engineering University, Harbin 150001, China)

Abstract: In order to obtain high quality and flexible (all kinds of transformations) characters display and What You See Is What You Get (WYSIWYG) printing, the traditional way of Using Dot - Matrix Font can no longer satisfy our needs. Here is a good solution that can solve both the problems: Using TrueType font. The basic knowledge of TrueType was introduced as well as how to use it in VxWorks system with the help of FreeType. It well solved the problems above. This technology can also be used in many other systems like Linux.

Key words: VxWorks; TrueType; UGL font driver; printing

与传统的使用点阵字库相比,TrueType 字库至少会带来这样的好处:可以高质量地实现字符的无级放大或缩小,高质量地实现字符的旋转、倾斜等操作(如图 1),方便地实现"所见即所得".

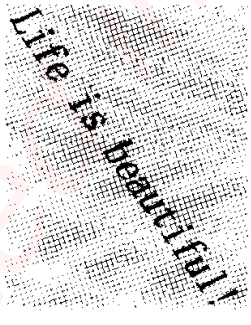

图 1　应用效果图

由于有以上优点,在很多对字形有特殊操作要求的地方最好使用 TrueType 字库. Windows 中现在使用的就是 TrueType 字库.

1　TrueType 字库的基础

1.1　TrueType 简介

TrueType 字库是一种轮廓字库,在 TrueType 字库中,字形的信息是通过使用一系列的点来描述的.这些点之间或通过直线段,或通过二次贝塞尔曲线来连接,从而形成字形轮廓.图 2 是汉字"乾"的字形示例.

图 2 左边图形是直接连接字库中描述"乾"的点形成的,其中的圈表示描述字形的点:小圈表示直线段或贝塞尔曲线的端点,大圈表示贝塞尔曲线的控制点.

形成字形轮廓时对点的处理是这样的:假设有了起点 $P_0(x_0, y_0)$(它肯定是端点,即小圈),再找下一个点 $P_1(x_1, y_1)$.如果 P_1 也是端点,则用直线段连接 P_0, P_1,再将 P_1 作为新的起点继

收稿日期:2002 - 12 - 24.
作者简介:孙枫(1944 -),男,教授,主要研究方向:惯性导航技术.

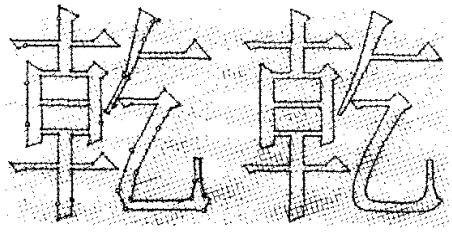

图2 字形轮廓线示例

续连接剩下的点；否则 P_1 是贝塞尔曲线控制点，再找下一个点 $P_2(x_2, y_2)$. 如果 P_2 是端点，则将 P_0, P_1, P_2 用贝塞尔曲线连接，P_2 作为新的起点继续连接剩下的点；否则再找下一个点 $P_3(x_3, y_3)$，计算 P_2 与 P_3 的中点 P_x，将 P_0, P_1, P_x 用贝塞尔曲线连接，P_x 作为新的起点继续连接剩下的点. 图2右边的图形是加入了二次贝塞尔曲线形成的实际见到的字形轮廓.

得到字型轮廓后，再对它进行填充就得到了需要的字符位图. 剩下的工作就很简单了，通过打点或贴图都可以画出字符来.

1.2 FreeType 的介绍

使用 TrueType 字库也有一定的困难：主要就是需要对 TrueType 字库的格式要有所了解，这样才能正确地提取出字形信息，而这对大多数用户来说很困难，也是很麻烦的. 但是现在有了 FreeType 这个开放源码的共享软件(可以从网上下载)，就不必亲自做这些工作了. FreeType 可以在很多平台下编译并使用(如 Windows、Linux、VxWorks 等). 利用 FreeType 提供的 API，得到字形信息，形成位图等工作都可以很方便地完成.

1.3 FreeType 使用

FreeType 提供了丰富的 API，具体用法可以参见其文档，这里仅作简要介绍.

用 FreeType 的 API 画出一个字符的流程大致如下(仅列出函数名)：

```
……
/* 初始化 FreeType 库 */
FT_Init_FreeType;
/* 建立字体，如宋体、楷体等 */
FT_New_Face;
/* 指定查找字符字形的编码，如 Unicode、GB2312 等 */
FT_Select_Charmap;
……
/* 根据字符编码得到字符字形在字库中的位置 */
FT_Get_Char_Index;
/* 根据字符字形在字库中的位置得到其字形描述，即字形轮廓 */
FT_Load_Glyph;
/* 将轮廓填充成位图 */
FT_Render_Glyph;
/* 用户自定义函数画出位图 */
Drawimage;
……
/* 撤消字体 */
FT_Done_Face;
/* 撤消 FreeType 库 */
FT_Done_FreeType;
……
```

当然实际系统中的应用不会这么简单，但是其基本过程是这样的.

2 UGL 字体驱动

UGL 是 Zinc 的基础，首先在 VxWorks 的 UGL 字体驱动中加入 TrueType 字库支持：

1)根据 VxWorks 的 UGL 字体驱动的接口标准写出驱动代码.

2)用行命令方式编译 UGL. 在 UGL 文档中介绍了行命令方式编译 UGL，以及为加入新字库支持而修改相关文件(如 uglInit.h 等)的方法.

现有 UGL 字体驱动只提供了用于水平方向字符显示的接口，为了能够更加灵活地显示字符，可以在驱动接口中增加函数指针 UGL-STATUS (* textDrawFree)；为了支持打印，可以加入函数指针 UGL-STATUS (* textPrint). 当然如果用户要调用这2个函数，就得在 ugl.h 中加入相应的函数声明，并在 uglfont1.c 中加入相应的函数定义.

```
typedef struct ugl-font-driver
{
  /* 原来的驱动接口 */
  ……
  /* 新加入的驱动接口 */
  UGL_STATUS (* textDrawFree)
  (struct ug_lgc * pGc,
  const char * text,
  unsigned long length,
  const UGL_TT_PARAM * param);
  UGL_STATUS(* textPrint)(struct ugl_font_
  driver * pFontDriver,
  const char * pFaceName,
```

```
  FILE * outfile,
  const char * text,
  int length,
  const UGL_TT_PARAM * param);
}UGL-FONT-DRIVER;
```

textDrawFree 中的结构 UGL_TT_PARAM 中,定义了一些要用到的参数,如大小、前景及背景颜色、旋转角度、倾斜程度等.

另外,为了提高字符显示速度,可以用下面的方法:

1)将 TrueType 字库调入内存.即用 FT_New_Memory_Face 代替 FT_New_Face;

2)利用 FreeType 提供的缓存功能.具体用法可以参见 FreeType 提供的文档.

3)对固定尺寸的字体(一般用于界面,这种情况下对速度有较高的要求)专门建立一个 Cache.现举例说明如下:

假设 Cache 的结构如下:

```
struct tt_cache_
{
  /* code_array 存储字符编码 */
    unsigned long    code_array[TT_CACHE_
                     NUM];
    /* cache_glyph 存储相应的位图 */
    struct tt_glyph_   cache_glyph[TT_CACH
                       E_NUM];
};
```

TT_CACHE_NUM 是需要建立 Cache 的字符数量.一般来说,Cache 中应包括 ASCII 可显示字符、汉字中的一级字及一些常用符号.

1)假设 tt_cache->code_array 中已经存储了需要建立 Cache 的字符编码,首先对它排序(用 C 语言的 qsort 即可):

```
qsort( tt_cache->code_array, TT_CACH
E_NUM, sizeof(unsigned long), compare);
```

2)为所有字符生成相应的位图.

3)系统运行过程中需要查找字符的字形时,执行以下步骤:

①用 bsearch 在 tt_cache->code_array 中查找字符编码.

②若查找到,则相应的位图也找到了.

③若没查找到,则到字库中去找.

4)销毁字体时要同时销毁 Cache.

应该注意的是上面的步骤 1)、2)是在系统启动时做的,如果第 2)步时间太长的话,可以不执行步骤 2),只需改变步骤 3):

①用 bsearch 在 tt_cache->code_array 中查找字符编码.

②若查找到,再看相应的位图是不是存在.如果已经存在,则相应的位图也找到了.如果还没有,则根据字库生成位图,并添加到 Cache 中.

③若没查找到,则到字库中去找.

实践证明,这种方法稍好一点.

3 打 印

下面介绍在窗口系统 Zinc 中利用 UGL 字体驱动如何实现打印中的"所见即所得".

Zinc 中的打印流程是:先生成 PS(PostScript)文件,再把 PS 文件发到打印机端口进行打印,所以下面只讨论如何生成 PS 文件.

由于现有系统 Zinc 不支持汉字的打印,所以必须回到 UGL 中用打点的方式画出汉字.这就是为什么实现 UGL 字体驱动时要加入函数指针 UGL_STATUS (*textPrint)的原因.

textPrint 中的参数意义为:

pFaceName 字体名,如宋体、黑体等;

outfile 指向打印的输出目标:PS 文件.

修改 Zinc 中的 ZafPrinter,当要打印字符时就调用 UGL 中的打印函数,UGL 中实现打印的部分基本与屏幕显示相同,不同的只是把向屏幕(或内存位图)打点(或贴图)换成了向 PS 文件写入打点(或贴图)命令.

由于打印和屏幕显示两者只是输出目标的不同:前者是显示器(如果是内存位图也一样);后者是 PS 文件.这样就很容易实现 WYSIWYG——只要将坐标单位设置为英寸(或英寸的千分之一)就行了.这里介绍一个技巧,由于汉字的打印是通过打点实现的,所以这样会使得 PS 文件比较大.要减小 PS 文件,可以将连续的打点用画线来代替.由于打印机的分辨率很高,这样做的效果会很明显:一般可以将 PS 文件减小到原来的 1/5,这将大大减少写打印机端口的时间.

4 结束语

在 VxWorks 中使用 TrueType 字库的方法已成功地应用于 VxWorks 嵌入式地理信息系统中,并可以很容易地移植到很多其他的系统中.

参 考 文 献:

[1] 于明俭,陈向阳,方 汉. Linux 程序设计权威指南[M]. 北京:机械工业出版社,2001.