基于 uCLinux 和 MiniGUI 的广告机用户界面设计与实现①

Design and Implementation of User Interface of Ad-Displayer Based on uCLinux and MiniGUI

舒　璇　郁晓飞　陈晓光　(复旦大学 通信科学与工程系 上海 200043)

摘　要： 基于嵌入式 uCLinux 内核和 MiniGUI 软件平台，设计并实现了公交车站广告机的用户界面。对相应的开发环境进行了搭建，并给出了运行环境的执行步骤。在用户界面设计中，我们采用了双缓冲技术，对动画和滚动字幕的速率进行了精确控制并提出了相应算法，最后实现了画面的剪裁。本文提出的解决方案已在上海市部分公交车站成功运行。

关键词： 嵌入式系统 uCLinux MiniGUI 双缓冲 动画 滚动字幕 剪裁

1　引言

根据国际电气和电子工程师协会(IEEE)的定义[1]，嵌入式系统是“用来控制、监视和辅助设备、机器和对象的装置”。业内一个普遍被认同的定义是：以应用为中心、以计算机技术为基础、软件硬件可剪裁、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

从 20 世纪 80 年代末开始，陆续出现了一些嵌入式操作系统，其中比较著名的有 VxWorks，pSOS，Neculeus 和 Windows CE[2]。Linux 作为嵌入式操作系统，与其它商业化的嵌入式系统相比有着与生俱来的优势：(1)开放的源码，丰富的软件资源；(2)功能强大的内核，性能高校、稳定，多任务，易于剪裁；(3)支持多种体系结构，如 X86、ARM、MIPS、ALPHA、SPARC 等；(4)完善的网络通信、图形、文件管理机制；(5)支持大量的周边硬件设备；(6)良好的开发环境，不断发展的开发工具箱；(7)价格低廉，能有效降低产品成本，适用于对成本敏感的嵌入式系统。

uCLinux[3]是最流行的嵌入式 Linux 之一，它表示 Micro-Control-Linux，字面上的理解就是“微控制领域中的 Linux 系统”。它专门针对没有 MMU 的 CPU(现在也可以根据需要用于有 MMU 的 CPU)，并且专为嵌入式系统作了许多小型化的工作。

在嵌入式 Linux 上面已经存在多种图形用户界面，如 Qt/Embedded、Microwindow、OpenGUI 以及 MiniGUI[4]。其中 MiniGUI 是国内生产的比较优秀的基于嵌入式 Linux 的图 5 形用户界面软件。它提供了完备的多窗口机制，实现了类 Win32 的消息传递机制，能够支持多字符集和多字体，以及全拼、五笔等汉字输入法，可以支持常见的图像文件，如 BMP、GIF、JPEG、PCX、TGA、PNG 等；它还支持 Windows 的资源文件，如位图、图标、光标等。此外，MiniGUI 还具有小巧、可配置、移植性好等优点，包含全部功能的库文件大小为 300kB 左右。可以根据开发的需要自行配置和编译，特别适合于作为嵌入式 Linux 系统的图形用户界面。与国外同类型的图形用户界面相比，MiniGUI 有一个很大的优势，就是完全支持中文，这样有利于在中文的平台上开发出全中文的应用程序。

最初，我们也曾考虑过采用界面视觉效果比较丰富的 Qt/Embedded，但是由于 Qt/Embedded 对 uCLinux 不支持，再加上 Qt/Embedded 对系统 CPU、内存等硬件要求比较高，最终我们还是选择了性价比较高的 MiniGUI。

① 收稿时间:2008-12-09

2　用户图形界面设计要求

图形用户界面的整体安排如图 1 所示,在图 1 中标明了各个矩形区域的大小及用途：

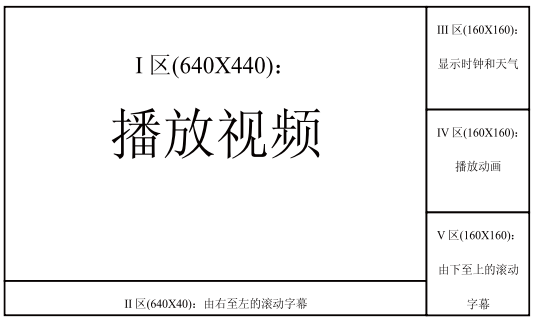

图 1　广告机用户界面模块设计

整个窗体被分为五个矩形区域。其中 I 区为视频播放区,视频播放要占用 uCLinux 系统的大部分 CPU 和内存，所以在接下来的设计中必须使得其他区域的应用模块尽量占用更少的 CPU 和内存。II 区为横向滚动字幕区，要达到字幕滚动无顿挫感、无闪烁，并且可以控制字幕滚动的速率的要求。III 区为时钟和天气显示区，其中天气的显示可能会配合图片加以修饰，所以本区域也要支持图片的显示。IV 区是动画播放区，主要要做到画面连贯、无跳帧现象。V 区为纵向滚动字幕区，要求与 II 区基本相同。

3　开发环境与运行环境

3.1 前期准备

1)Mipsel-linux-gcc 交叉编译工具链 ;2)开发板(我们采用的开发板主芯片为：Sigma Designs SMP8635 RevC)；3)对联网线；4)显示屏(我们采用的显示屏型号为：Samsung SyncMaster 940MG)；5)Linux 操作系统，如 RedHat9。

3.2 搭建过程

1) 在 Linux 操作系统上开启 nfs 服务，我们在 RedHat9 上将 /etc/exports 设置成 /works/nfs 192.168.110.0/255.255.255.0(rw,no_root_sqash)，运行 service nfs status 如果服务已经打开则直接进入第 2 步,如果未打开则运行 service nfs restart

2) 在 nfs 下建立用于安装 MiniGUI 的文件夹，运行 mkdir －p /works/nfs/sigma

3) 把 minigui.tar.bz2，minigui_smp 863x.tar.bz2 解压到/works/nfs/sigma 目录下

4) 开启板上电源，用对联网线连接开发板和 PC

5) 板上默认 ip 是 192.168.110.101,用相应用户名和密码登录

6) 在板上挂载 nfs，我们使用的 PC 的 ip 是 192.168.110.135，运行 mount －o nolock 192.168.110.135:/works/nfs/sigma /mnt/ide

7) 运行cd /mnt/ide/minigui_ smmp863x/mrua

8) MiniGUI 厂商提供的编译好的程序在/works/nfs/sigma/minigui_smp863x 目录下，对开发板来说，该程序即在/mnt/ide/minigui_smp863x 目录下。

9) 分别是运行 MiniGUI 需要的配置文件和资源文件,也就是第 3 步里解压的 minigui.tar.bz2,MiniGUI.cfg 可根据需要做相应配置。

10) 在 /works/nfs/sigma/minigui_smp863x 目录下编译源文件(假设源文件名为 demo)，运行 mipsel-linux-gcc－o demo demo.c －lminigui －lmgext－ljpeg －lpng －lz –lpthread

11) 在/mnt/ide/minigui_smp863x 目录下运行程序，此时，显示屏上会显示程序效果。

4　实现中的技术难点

4.1 双缓冲切换技术[5]

窗体在响应绘图消息的时候要进行复杂的图形处理，尤其是在刷新时，对图像的擦除和重写造成了图像颜色的反差。当绘图消息的响应很频繁的时候，这种反差也就越发明显，于是我们就看到了闪烁现象。

解决窗体闪烁最直接的办法是避免背景色的重绘，但单纯的禁止重绘是不够的，为了抑制屏幕的闪烁，提高帧在屏幕上的刷新速度，我们采用了双缓冲技术构造前台和后台两个缓冲区。我们知道，显示器显示的内容可以用一个二维点阵来表示，在计算机程序里就是一个二维数组。因此可以用这样一个二维数组来虚拟一个“显示器”，先将一帧图像绘制在此虚拟的“显示器”上，待整帧图像绘制完毕，在将虚拟“显示器”上的图像复制到屏幕上。在 MiniGUI 中我们使用函数 CreateCompatibleDC(hdc)或者函数 CreateCompatibleDCEx(HDC hdc, int width, int height)来在内存中创建与给定 DC(设备上下文)兼容的虚拟 DC。创建的虚拟 DC 与给定 DC 具有完全兼容的像素格式。另外我们使用函数 BitBlt(HDC hsdc, int sx, int sy, int sw, int sh, HDC hddc, int dx, int dy,

DWORD dwRop)实现位块的传递，也就是后台缓冲区到前台缓冲区的复制。hsdc 表示源设备上下文，sx、sy、sw、sh 分别用来指定源设备上下文的 x、y 坐标和矩形区域的宽度和高度，hddc 表示目标设备上下文，dx、dy 分别用于指定目标区域的 x、y 坐标。双缓冲切换技术的示例程序段如下：

```
hdc = BeginPaint(hwSubtitle);  /*获取窗体设备上下文句柄*/
hdcstring = CreateCompatibleDCEx(hdc, 2200, 80);
/*创建在内存中显示字幕的后台缓冲区*/
SelectFont (hdcstring, lfSong60);  /*选择逻辑字体*/
SetTextColor (hdcstring, COLOR_blue);  /*选择字体颜色*/
TextOut(hdcstring, 480, 0, subtitle);  /*输出字符至后台缓冲区*/
hdcmemsubtitle=CreateCompatibleDC(hdc);
/*再次创建后台缓冲区以减少画面闪烁*/
……  /*对显示的位置、速率、刷新频率等进行控制*/
FillBox(hdcmemsubtitle, 0, 0, 480, 80);  /*将图像输出至窗体*/
BitBlt(hdcstring, visibleleft, 0, 480, 80, hdcmemsubtitle, 0, 0, 0);
BitBlt(hdcmemsubtitle, 0, 0, 480, 80, hdc, 0, 0, 0);
EndPaint (hwSubtitle, hdc);  /*结束绘图*/
```

4.2 动画和滚动字幕速率的精确控制及其算法

用 MiniGUI 实现动画，实际上就是把一系列静止的图像以一定的频率在窗体上依次显示出来。用 MiniGUI 实现滚动字幕，也就是让文本的显示坐标以一定的速率发生变化。均匀的刷新频率对于动画和字幕的显示效果非常重要。在这里，我们要使用到定时器。

在 MiniGUI 中，应用程序可以调用函数 SetTimer(HWND hWnd, int id, unsigned int speed)创建定时器，这里 hWnd 指定了接收 MSG_TIMER 消息的窗口，id 是定时器的标识号，speed 是定时器的到期时间或频率，在默认情况下到期时间以 10ms 作为单位。根据 MiniGUI 库的设置，如果定义了_TIMER_UNIT_10MS，speed 表示的是定时器的到期时间，如果没有定义过，则 speed 表示的是定时器频率。我们实现动画需要 speed 表示定时器频率，从而可以将一系列图像以指定的频率在窗体上显示。当创建的定时器到期时，创建定时器制定的窗口就会收到 MSG_TIMER 消息，并传递到期的定时器标识号。在不需要定时器时，应用程序可以调用 KillTimer 函数删除定时器。

如果将定时器频率设置为 SetTimer(HWND hWnd,int id, unsigned int speed)，动画速率的控制条件为 if(animcount>animation_ctrl)，则动画帧速为

$$\frac{1}{animation\_ctrl \times speed \times 10 \times 10^{-3}}\ frame/\sec ond \quad (1)$$

如果滚动字幕速率的控制条件为 if(subcout>sub_ctrl_1){subposition+=sub_ctrl_2}，则滚动速率为

$$\frac{sub\_ctrl\_2}{sub\_ctrl\_1 \times speed \times 10 \times 10^{-3}}\ pixel/\sec ond \quad (2)$$

动画帧速控制算法的伪代码如下：

1) 设置定时器频率 speed；

2) 设置内存 DC,在内存 DC 中显示动画帧；

3) 初始化：animCount = 0; /*帧速控制的计数变量*/

index = 0; /*帧号*/

4) 定时器产生 MSG_PAINT 消息：animCount++;

if(animCount>animation_ctrl) /*通过此条件语句控制动画播放速率*/

{

显示动画;

index++;

if(index>numFrame) /*动画一共有 numFrame 帧*/

{

index = 0;

}

animcount = 0;

EndPaint(hwAnimation, hdc);

}

滚动字幕速率控制算法的伪代码如下：

1) 设置定时器频率；

2) 设置内存 DC，在内存 DC 中显示字幕；

3) 初始化：visibleleft = 0; /*字幕左端位置*/

```
subcout = 0;  /*滚动字幕速率控制的计数变量*/
4) 定时器产生绘图消息：subcout++;
if(visibleleft>leftBoundary) /*字幕最左端边界*/
{
    visibleleft = 0;
}
if(subcout>sub_ctrl_1)  /*通过此条件语句控制字幕滚动速率*/
{
    visibleleft += sub_ctrl_2;  /*每 sub_ctrl_1 个单位时间向左移动 sub_ctrl_2 个像素*/
    显示字幕;
    EndPaint (hwSubtitle, hdc);
    subcout = 0;
}
```

4.3 画面剪裁

画面剪裁技术主要有两方面的应用，一是可以完成对窗体的某部分区域进行局部刷新从而减弱或消除闪烁的效果，二是可以实现一个大图的某个小部分的显示。

要实现画面的剪裁，必须先进行剪切域的定义，如下：

```
Typedef struct CLIPRECT
{
RECT rc;
Struct CLIPRECT* next;
#ifdef USE NEWGAL
Struct CLIPRECT* prev;
#endif
} CLIPRECT;
Typedef CLIPRECT* PCLIPRECT;
```

当我们要对某个图形进行拖放时，或者在动画的两帧之间，画面只在一个局部区域内发生变化，则可以使用 ClipRgnCopy 函数，剪裁出需要进行重绘的区域，使得只有被剪裁的区域被重绘而其余部分保持不变。由于需要重绘的面积减少，从而加快了画面的刷新速度，也可以达到抑制屏幕闪烁的效果。

剪裁技术的另一个重要的应用就是显示图像的一个局部。如果对一幅图片使用 SetClipRgn 函数定义一个剪切域，然后再绘制图形，则此时的剪切域就相当于一个窗口，只有绘制在窗口内的图形才能显示在屏幕上，窗口以外的图形不可见。

5 系统性能分析

广告机正常运行时系统的 CPU 和内存占用情况如下表所示(以下数据由 linux 系统命令 top 得到)：

表 1 广告机各模块 CPU 和内存占用率

应用模块	左右滚动字幕	时钟和天气	动画	上下滚动字幕
CPU 占用率	5%	<1%	<10%	3%
内存占用率	3%	1%	25%	2%

由以上数据可以看出，动画模块是各个模块中占用系统资源最多的一个模块，但是总的系统资源的使用效率处于一个比较合理的范围内，有足够的 CPU 和内存留给视频播放模块，满足了系统设计的要求。

6 结束语

本文详细说明了公交车站广告机用户界面在嵌入式 uCLinux 内核和 MiniGUI 软件平台上的设计与实现，讨论了采用 MiniGUI 进行用户界面设计的一些重要技术，如双缓冲技术、动画与滚动字幕速率控制、画面剪裁等。本文提出的解决方案已经在上海市部分公交车站的广告机上成功运行。由于其它原因，本文设计并实现的图形用户界面并没有涉及到用户的输入，而 MiniGUI 对用户输入也有着比较好的支持。因此，经过一定的改进，完全可以将此方案应用于有更多需求的用户界面项目如电视分播系统、移动传媒等。同时，本文也为其他的嵌入式软件开发，特别是用户界面的设计，提供了参考。

参考文献

1 王学龙.嵌入式 Linux 系统设计与应用.北京:清华大学出版社, 2001.

2 Stevens W R. 龙晋元译.Unix 环境高级编程.北京:机械工业出版社, 2002.

3 Sarwar SM, Koretsky R, Sarwar SA.李善平,施韦,林欣译.Linux 教程.北京:清华大学出版社, 2005.

4 赵振亮,徐立鸿,邓梧鹏,徐惠惠.基于 Linux 系统中嵌入式 GUI 的研究与分析.微型电脑应用, 2006,22(11):42－44.

5 左黎明,盛梅波,艾剑锋.JAVA 技术在远程实时数据的图像处理中的应用.计算机系统应用, 2003,13(8):41－44.