基于Qt/Embedded的控件扩展研究与实现

芦东昕[1,2],周建彬[1,2],谭振华[1]

(1. 中兴软件技术(南昌)有限公司 成都研究所,四川 成都 610041;
2. 华北电力大学 计算机科学与技术学院,北京 102206)

摘 要:嵌入式系统的广泛应用促进了嵌入式Linux图形用户界面(GUI)的发展,Qt/Embedded是一个易扩展且被广泛使用的嵌入式图形系统。为满足特定的界面需求,需对现有图形系统的控件进行扩展研究。分析了嵌入式图形系统Qt/Embedded的实现技术,然后对控件扩展方法进行了研究,在此基础上实现了一个具有特殊外观且透明度可调的容器控件,最后阐述了将扩展控件并入Qt/Embedded控件集的方法。提出的控件扩展方法,可以快速地定制出个性化控件,丰富图形界面;扩展控件合并到控件集不仅完善了原有控件功能,还提高了开发效率。

关键词:嵌入式Linux;Qt/Embedded;控件;扩展

中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2006)10-0097-04

Research & Implementation of Extended Widgets Based on Qt/Embedded

LU Dong-xin[1,2], ZHOU Jian-bin[1,2], TAN Zhen-hua[1]

(1. Chengdu Institute, Zhongxing Software Technology (Nanchang) Co. Ltd, Chengdu 610041, China;
2. School of Computer Science & Technology, North China Electric Power University, Beijing 102206, China)

Abstract: With the wide use of embedded systems, it accelerates the development of GUI in embedded Linux. Qt/Embedded is an embedded graphic system which is easily extended and widely used. In order to satisfy the requirements of special interface, it is needful to research how to extend widgets on existing graphic systems. This paper analyzes the technologies of implementation of embedded graphics system Qt/Embedded in detail, and researches into the methods of extended widgets, and then implements a container widget which has a special appearance and its transparence is adjustable. Lastly, the paper expounds the method of merging extended widgets into the sets of widget in Qt/Embedded. The extended widgets can make graphic interface colorful and ameliorate the function of original widgets. The method of extending widgets which is putted forward in the paper can help us to customize special widgets quickly and improve the efficiency of programming.

Key words: embedded Linux; Qt/Embedded; widget; extend

0 引 言

随着嵌入式系统的不断发展,特别是嵌入式处理器运算能力的不断增强,嵌入式系统被广泛应用于信息家电、移动通信、手持信息设备以及工业控制等众多领域,与此同时用户对于嵌入式系统图形用户界面(GUI)的需求也不断提高。为了满足特定应用环境的需求,经常需要定制一些个性化的独具风格的GUI。

嵌入式Linux是一种流行的嵌入式系统平台,它具备稳定、高效、易定制、易裁减、易移植、硬件支持广泛等优点,结合其免费、源码开发的特征,使得Linux在嵌入式操作系统中的地位日益重要。

收稿日期:2006-02-07
基金项目:国家"八六三"计划项目(2002AA1Z2306)
作者简介:芦东昕(1971-),男,黑龙江哈尔滨人,博士后,教授,研究方向为嵌入式系统、软件过程、软件开发管理。

目前进行嵌入式Linux开发,常用的GUI系统一般有以下几种可供选择:MicroWindows, MiniGUI, OpenGUI和Qt/Embedded等。这些嵌入式Linux的GUI提供了较为丰富的图形窗口部件(控件),但对于一些特定应用环境的界面需求,现有的控件并不能满足要求,因此需要对现有控件集进行扩展研究,定制出独具风格的控件以满足用户的个性化需求。文中就将如何扩展Qt/Embedded控件进行阐述,最后给出一个扩展控件实例。

1 Qt/Embedded实现技术

1.1 Qt/Embedded简介

Qt/Embedded是著名的Qt库开发商TrollTech公司(http://www.trolltech.com/)发布的面向嵌入式系统的Qt版本。Qt/Embedded是一个专门为小型设备提供图形用户界面的应用框架和窗口系统,提供了丰富的窗口小部件(widgets),并且还支持窗口部件的定制,因此它可以为

用户提供漂亮的图形界面。Qt 是 KDE 等项目使用的 GUI 支持库,所以由许多基于 Qt 的 X Window 程序可以非常方便地移植到 Qt/Embedded 版本上[1,2]。

Qt/Embedded 提供自身的轻量级窗口系统,比使用 Xlib 和 X Window 更紧凑;Qt/Embedded 的设计原则是不依赖于 X server 或者 Xlib,而是直接访问 framebuffer(帧缓存)作为底层图形接口[3],其体系结构如图 1 所示。同其他解决方案如 Qt/X11 相比其最显著的效果是减少了内存消耗,而只需要一个 Qt/Embedded 动态链接库就足以替代 X server,Xlib 库和其他嵌入式解决方案的图形工具包。因而非常适合嵌入式环境使用。

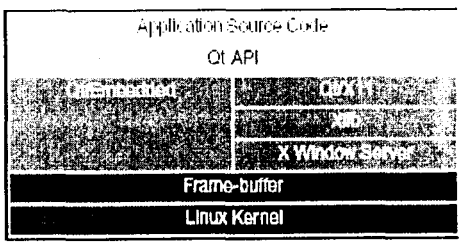

图 1 Qt/Embedded 与 Qt/X11 结构比较

1.2 Qt/Embedded 图形引擎

如图 1 所示,Qt/Embedded 的底层图形引擎基于 framebuffer。framebuffer 是在 Linux 内核版本 2.2 以后推出的标准的显示设备驱动接口[4,5],它将图形硬件的具体特性屏蔽起来,给程序员提供了一个抽象的编程界面。这种接口采用 mmap 系统调用,把显存映射到一维线性内存缓冲区,从而将显示设备抽象为帧缓存区,用户对这个缓冲区的读写就相当于对显存的操作,而写操作可以立即反映在屏幕上[6]。

在 Qt/Embedded 中,图形引擎的底层抽象基类主要是 QScreen 类和 QGfx 类。QScreen 类是从 framebuffer 抽象出的底层显示设备基类,它声明了显示设备的基本描述和操作方式;QGfx 类抽象出显示设备的图形环境具体操作接口[4],比如画点、画线、画矩形等。Qt/Embedded 对于具体的显示设备的操作,都需要继承并重载此两基类中的虚函数来实现。

1.3 Qt/Embedded 控件的绘制流程

Qt/Embedded 中绘图处理机制是建立在事件驱动基础之上的。每一控件都有其自身的特定绘制方法,此方法一般是通过继承并重载父类中的虚函数来实现的。当需要绘制控件时,处理过程是首先通过调用 update()或 repaint()方法来产生绘制事件(QPaintEvent),并把此事件添加到事件队列,然后应用程序的 notify()函数把它发送到事件接收者,最后事件接收者通过 paintEvent()函数调用特定的绘制方法来实现自身的绘制。图 2 为控件绘制的处理流程。

1.4 Qt/Embedded 控件透明机制

1.4.1 控件间的位置关系

控件(widget)在父窗口(parent widget)上都有一个相对固定的区域,此区域可以由控件在 parent widget 上的顶点坐标(x,y)、控件宽度(width)和控件高度(height)来确定。图 3 是控件与 parent widget 的关系图。实现控件透明的基本方法是,首先获取自己在 parent widget 上的区域,并截取此区域在 parent widget 上的背景图,将此背景图作为控件自身的背景即可实现透明效果。当然,这种直接贴图的方法是全透明的处理方式,如果要实现半透明的效果,需要把截取的背景图与基色(可自定)进行某种混色运算产生新的背景图,再对控件贴图。

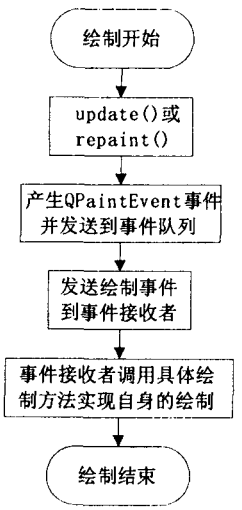

图 2 控件绘制处理流程

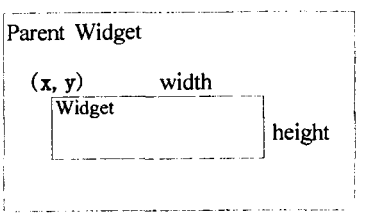

图 3 控件与 parent widget 的关系图

1.4.2 背景图转换过程

parent widget 的背景图主要可以分为图像模式和纯色模式,通过 backgroundMode()方法可以获得模式类别。对于纯色模式,无需进行背景图的转换,只需分离出背景色的 r,g,b 的值进行混色运算得出新的 r,g,b,然后用 setPalette(QPalette(QColor(r,g,b)))设置背景色就实现了透明效果。对于图像模式,从 parent widget 截取的背景图是 QPixmap 类的对象,需要使用 convertToImage()方法把它转换成 QImage 类对象,并对此 QImage 对象中的每个像素点(pixel)的 r,g,b 值与一个基色进行混色运算并用运算结果设置新值;然后通过 QPixmap::convertFromImag()方法把运算后的 QImage 对象转换成 QPixmap 对象,于是具有透明效果的背景图就生成好了;最后使用 setBackgroundPixmap()方法对控件贴图就实现了控件的透明处理。

1.4.3 混色运算方法

假定基色为 BASECOLOR(QColor 对象,可以定义为常量),其 rgb 值分别为:br,bg,bb,可以通过 QColor 类的 rgb()方法获得;QImage 对象中 pixel 的 rgb 值分别为:r,

g,b,这 3 个分量可用函数 qRed()、qGreen()和 qBlue()分别获得。在取得像素点颜色信息后就可以进行混色运算了。为了实现可调的透明效果，需要设置一个透明度(transparence,是一个百分数，可用整型表示)。混色的运算的公式是：r = ((br * nontrans + r * transparence)/100)，g,b 分量的运算与此雷同，对应的分量更换为 bg,g 和 bb,b 即可，公式中的 transparence 表示透明度，nontrans 为基色含量，其值为 100 - transparence。

以上介绍的是图像模式的混色运算过程，纯色模式相对简单。在分离出背景色的 r,g,b 的值后，直接使用上述混色运算公式就行了。

2　控件扩展方法

由于 Qt/Embedded 已经提供了比较丰富的控件，所有的控件都是通过继承 QWidget 实现的，因此在定制控件时需要选定功能比较相近的控件作为基类。在 Qt/Embedded 中，常用于扩展的抽象基类有按钮控件抽象类 QButton、用户界面对象抽象类 QWidget、具有框架的窗口控件抽象类 QFrame。

控件的扩展方法一般采用如下两种方式。一种是对已有的控件直接继承，这种方法适用于定制功能、界面相对单一的控件；另一种是把已有或已实现的控件放到容器控件中进行组合(必要时需进行控件的布局管理)而生成新的控件，这种方法一般用于定制功能相对复杂的组合控件。

在选定基类和扩展方式后，剩下的工作就是控件的具体定制。定制的控件一般都有相对特殊的功能或外观，虽然它们都依据具体需求来确定，但是外观需要通过实现基类提供的绘制接口来达到定制的目的。不同的基类提供的绘制接口不尽相同，比如 QFrame 抽象类提供的绘制接口是 drawContents(QPainter * p)；而 QButton 提供的接口是 drawButton (QPainter * p)；和 drawButtonLabel (QPainter * p)，因此只需要实现基类提供的相关接口，就可以定制扩展控件的个性化外观了。

3　控件扩展实例

在这里详细讲述一个具有特殊外观(气泡状)且透明度可设置的容器控件的实现过程。由于需要定制的控件是容器控件，因此选择具有框架的窗口部件抽象类 QFrame 作为基类。此容器类的声明结构简要如下：

```
//QFrame 作为基类
class Q_EXPORT TBubble: public QFrame
{
public:
  {枚举定义气泡的方位;}
  {构造函数声明;}
protected:
  {绘制函数声明;}
private:
  {气泡的偏移量;}
  {气泡的透明度;}
  {气泡的位置;}
};
```

3.1　定制控件外观

从图 2 中可以看出，定制控件外观的工作主要集中在具体绘制方法的实现。对本例来说，基类 QFrame 提供了 drawContents(QPainter * p)接口，只需对此接口加以实现就可以定制外观了。在此接口的实现中，可分为三步完成。首先画出边框，然后依据气泡的位置和偏移绘制气泡，最后再擦除多余的线段。为了便于处理，用常量定义了气泡的高度，const int triaHigh = 8。下面的代码段实现了气泡朝下时控件的绘制过程，其他情况与此雷同，故而省略不再列出。

```
{获取外框顶点(x,y)、高 h 和宽 w 等参数;}
//绘制圆角外框
p->drawRoundRect(x, y, w, h, 10, 10);
//绘制开口朝下的气泡
p->drawLine(x + tria_Off, y + h - 1, x + tria_Off +
triaHigh, y + h - 1 + triaHigh);
p->drawLine(x + tria_Off + triaHigh, y + h - 1 +
triaHigh, x + tria_Off + triaHigh * 2, y + h - 1);
//设置底色后擦除多余的线段
p->drawLine(x + tria_Off + 1, y + h - 1, x +
tria_Off + triaHigh * 2 - 1, y + h - 1);
```

3.2　透明的实现

如前所述，parent widget 的背景图有两种模式，鉴于纯色模式比较简单，在本例中采用图像模式。由于背景图转换和混色运算相对独立，不完全依赖于控件本身，因而采用外部函数来实现。需要指出的是，由于 QWidget 是所有用户界面对象的基类，所以可以采用 QWidget 指针作为参数传递，来实现控件成员函数的调用。下面的伪代码完成了透明处理：

```
{取父窗口背景图;}
{获取控件在父窗口背景图上的图片 QPixmap 对象;}
{混色处理;}
{使用 setBackgroundPixmap()设置控件的背景图;}
{update()产生绘制事件，对控件进行重绘从而实现透明效果;}
```

代码中的混色处理函数，其实现过程如下：

```
{QPixmap 转换为 QImage;}
{获取基色的颜色分量;}
//对 QImage 中的每一像素点进行混色运算
  r = ((br * nontrans + r * transparence)/100);
  g = ((bg * nontrans + g * transparence)/100);
  b = ((bb * nontrans + b * transparence)/100);
  {重设像素点颜色;}
{QImage 转换为 QPixmap;}
```

至此，一个完整的具有个性化外观的透明控件已经实现了。图 4 是此控件的运行实例截图，图中有两个气泡状

的容器控件,它们的透明度分别是 80% 和 100%(全透明),采用的基色是 Qt::gray。另外,在两个气泡容器控件中共画了 3 个 Label 控件,它们也被透明函数 transparentize()处理了,其透明度分别是 80%、50%和 100%。

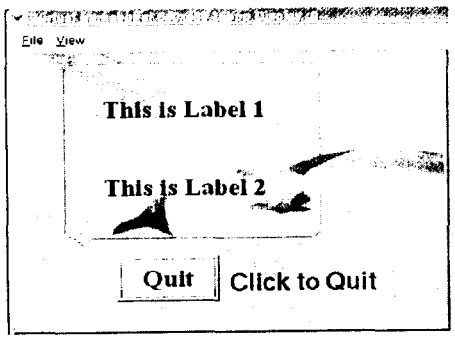

图 4 扩展控件的运行实例截图

3.3 控件合入 Qt/Embedded 类库

定制的控件完成后,可以把它合入 Qt/Embedded 类库,这样在以后需要使用时就可以当作 Qt/Embedded 基本控件直接使用了,从而提高工作效率。合入控件的方法比较简单,只需要在 $QTEDIR/src/Makefile.in 文件中增加控件的相应编译项就可以了。以本控件为例来进行描述,需要新增的内容如下:

(1)控件的 moc 原文件,通过控件的头文件(tbubble.h)生成:

SRCMOC=moc_tbubble.cpp

(2)对 moc 原文件编译生成的目标文件:

OBJMOC=moc_tbubble.o

(3)控件的目标文件:

OBJECTS_widgets=tbubble.o

4 结束语

随着嵌入式技术的蓬勃发展,人们对嵌入式系统的图形用户界面也不断地提出新的需求,而现有的图形系统往往不能满足这一要求。笔者在对嵌入式图形系统 Qt/Embedded 的实现技术进行分析的基础上,成功地实现了个性化控件的定制,不仅满足了特定界面的需求,同时丰富了 Qt/Embedded 控件功能。

为了进一步定制结构复杂的控件以及实现动态的透明效果,今后还需做以下工作:继续研究 Qt/Embedded 控件绘制方法和事件驱动机制,在背景变化的情况下能实现动态的透明效果,从而定制出功能强大的扩展控件。

参考文献:

[1] Trolltech Inc. Qt Reference Documentation 3.0.6. 2002 [EB/OL]. http://doc.trolltech.com/3.0/index.html, 2002.

[2] Trolltech Inc. Qt Reference Documentation4.1.0.2005[EB/OL]. http://doc.trolltech.com/4.1/index.html,2005.

[3] Trolltech Inc. Qt/Embedded Whitepaper[EB/OL]. http://www.trolltech.com/pdf/whitepapers/qt-embedded-whitepaper-a4.pdf,2002-03.

[4] 徐广毅,张晓林,崔迎炜,蒋文军. Qt/Embedded 在嵌入式 Linux 系统中的应用[J]. 单片机与嵌入式系统应用,2004(12):14-18.

[5] 吴伟清,王 磊,吴朝晖. 基于 QTE 的嵌入式 Linux 中文环境解决方案[J]. 计算机工程,2005,31(2):87-88.

[6] 邹思轶. 嵌入式 Linux 设计与应用[M]. 北京:清华大学出版社,2002.

(上接第 96 页)

中得到了应用,为生成问句向量提供信息来源,进而间接地为答案提取服务,效果良好,达到了预期目的。在后续的答案提取实验中答案匹配的准确率可达到 82.05%。然而在实验过程中也发现一些问题:在神经网络模型中,输入信息的提取十分重要,输入参数应该能够很好地反应问题的特征。本方法的一个不足是输入层的节点数目偏多,使得效率下降。对输入层的设计进行改进,使得输入层的节点减少,而又不会损失统计特征的特性,而且能够提高网络的效率和正确率。这是下一步要做的工作。

参考文献:

[1] Abney S. Parsing by chunks[A]. In:Berwick R, Abney S, Tenny C. Principle-Based Parsing[C]. Dordrecht: Kluwer Academic Publishers,1991.

[2] Abney S. Prosodic structure, performance structure and phrase structure[A]. In: Proc Speech and Natural Language Workshop[C]. SanMateo, CA: Morgan Kaufmann Publishers, 1992.

[3] 李宏乔. 汉语组块分析技术及应用研究[D]. 北京:北京理工大学计算机系,2004.

[4] 李素建,刘 群,杨志峰. 基于最大熵模型的组块分析[J]. 计算机学报,2003,26(12):1722-1727.

[5] 李 珩,朱靖波,姚天顺. 基于 SVM 的中文组块分析[J]. 中文信息学报,2004,18(2):1-7.

[6] Schmid H. Part-of-speech Tagging with Neural Networks [A]. In Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)[C]. Kyoto, Japan:[s.n.],1994.172-176.

[7] 奚晨海,孙茂松. 基于神经元网络的汉语短语边界识别[J]. 中文信息学,2002,16(2):20-26.

[8] 樊孝忠,李宏乔,李良富,等. 银行领域汉语自动问答系统 BAQS 的研究与实现[J]. 北京理工大学学报,2004.24(6):528-532.

[9] Zhang Huaping, Yu Hongkui, Xiong Deyi, et al. HHMM-Based Chinese lexical analyzer ICTCLAS[A]. 2nd SIGHAN workshop affiliated with 41st ACL[C]. Sapporo, Japan:[s.n.],2003.184-187.