ON THE QT

JONO BACON

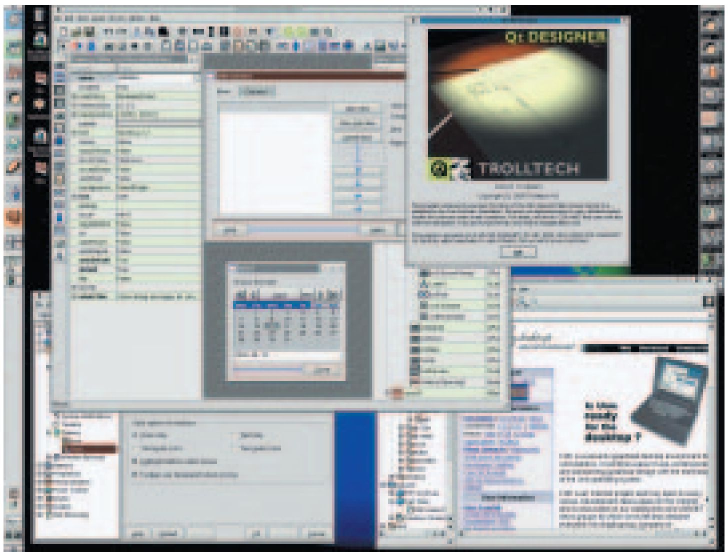
When the KDE project was started by Mathias Ettrich, he needed to decide on which graphical toolkit to use to base the environment on. There were some options available, but one of the most capable toolkits was Qt by Trolltech in Oslo, Norway.

One of the great things about Linux is the sheer number of available options and choices you can make. There are often multiple solutions for a single problem. An example is in windowing environments; we have KDE, Gnome, fvwm, Afterstep, Enlightenment etc.

One of the most popular windowing environments on Linux and UNIX-based systems is the K Desktop Environment (KDE). KDE has been around for approximately five years and has developed into a mature project with hundreds of developers and some very competent, stable releases.

A graphical toolkit is a number of buttons, checkboxes, scrollbars etc and other facilities to write graphical software with. There are a few toolkits available but Qt is one of the most capable, if not *the* most capable.

From those early days five years ago, Qt has developed into an incredible piece of software with thousands of users using the toolkit around the world on a number of platforms and operating systems. This series of articles is going to look at Qt, what it can do and what kind of software you can write with it.


Qt Designer main window

Getting to know Qt

To start with, it is a good idea that we get a grasp of what Qt can do and what facilities are available. In this issue we will look at these features and get a good grounding of Qt's facilities.

Firstly, Qt is a very portable toolkit. Qt has been made available and ported to the following platforms:

- Linux (all major distributions)
- UNIX (all major UNIXes)
- Microsoft Windows
- Mac
- Embedded Devices (Casiopeia, iPAQ etc)

There are also some other ports, so Qt always remains a portable toolkit.

Not only is Qt portable, it is licensed to your needs in a very flexible way. Qt is available in three versions; Free, Professional and Enterprise. The free version involves Qt for X11, Embedded (and recently Windows). The free versions for X11 are licensed under the GPL and include full source code. The free version of Qt/Windows is binary only and does not contain the source code of the toolkit. For those wishing to write closed source or commercial software there is the Professional and Enterprise editions, which have different licensing to enable this. This licensing scheme makes Qt a very flexible toolkit, and as it is maintained by a company full time it always remains competent, competitive and, in our case, free.

What can it do?

In these articles I am going to be focusing my efforts on the newly available Qt 3.0 version. There are previous editions such as the 2.x.x series and the 1.4.x series that are floating about, but I suggest you use the most current version available.
Although at first glance Qt looks like a number of

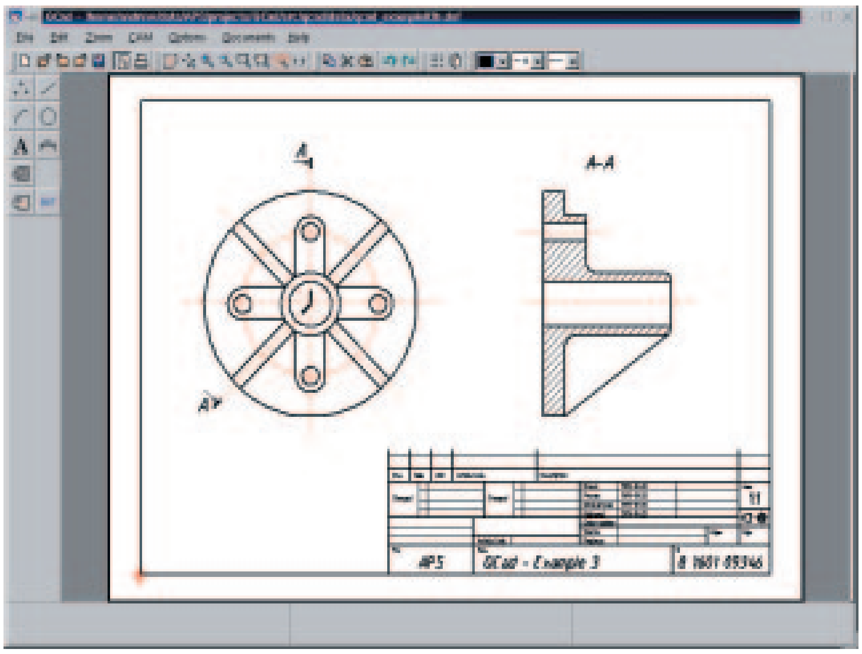
widgets (controls) that you can use to develop software, it is much more than that. Qt not only offers you a number of pre-designed widgets, but also the capability to roll your own widgets. Qt also includes a number of classes for managing data such as strings, numbers, vectors, linked lists, stacks, XML, DOM trees etc. On top of this Qt offers a number of networking features, support for multiple image formats and international text support.

Qt does not stop there, there's also the following:

- **Multiple monitor support**
  Qt allows applications to utilise multiple screens. On UNIX, this will support both Xinerama and the traditional multi-screen technology.
- **New Component model**
  Qt provides a platform-independent API for runtime loading of shared libraries and accessing of their functionality using a COM-like interface concept.
- Support for the latest evolutions in GUIs
  Qt supports the docking/floating window concept of modern, complex GUIs. It also adds a GUI control for interactive editing of rich text.
- **Regular Expressions**
  Qt-3.0 features a new and powerful regular expression engine greatly simplifying complex text manipulation operations. The syntax is compatible to, and as powerful as, Perl regular expressions while at the same time including full support for Unicode.
- **Accessibility Support**
  Qt controls provide information for accessibility architectures, so that visually or mobile-impaired people can use applications written in Qt with the standard tools provided (eg the Windows Magnifier and Narrator).
- **64-bit Safety**
  The emerging, next generation of 64-bit hardware is supported by Qt 3.0.

Database Programming

Qt 3.0 includes a platform and database-independent API for accessing SQL databases. The API has both ODBC support and database-specific drivers for Oracle, PostgreSQL and MySQL databases, and custom drivers may be added. Database-aware controls that provide automatic synchronisation between GUI and database are included in Qt 3.0. The Qt Designer has full support for these new controls, resulting in a RAD solution for database applications.


QCAD application written with Qt

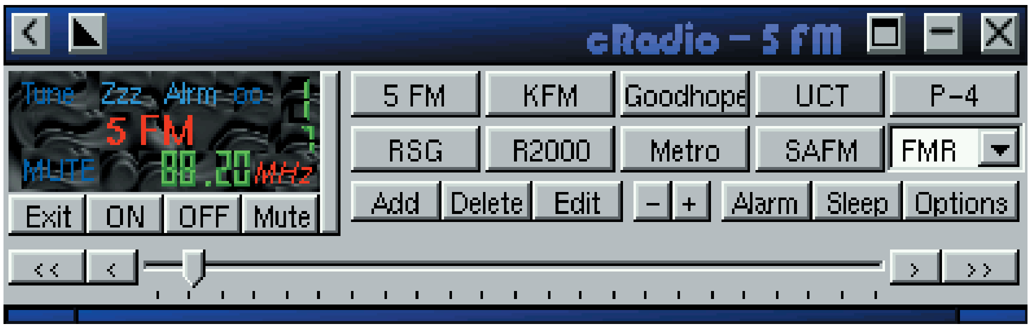
C++ support

Qt itself is written in and natively supports C++ as a language. You don't need to hunt far on the Net for a raging debate on whether C or C++ is better for GUI development, but it is acknowledged in a number of places that C++ is inherently better for GUI development. Qt is a good example of a well-crafted toolkit C++ and object orientated paradigms. In the coming months I will be giving tutorials on using Qt, and some C++ knowledge is assumed. If you are unfamiliar with C++, there are a number of tutorials and good books available on the Net.

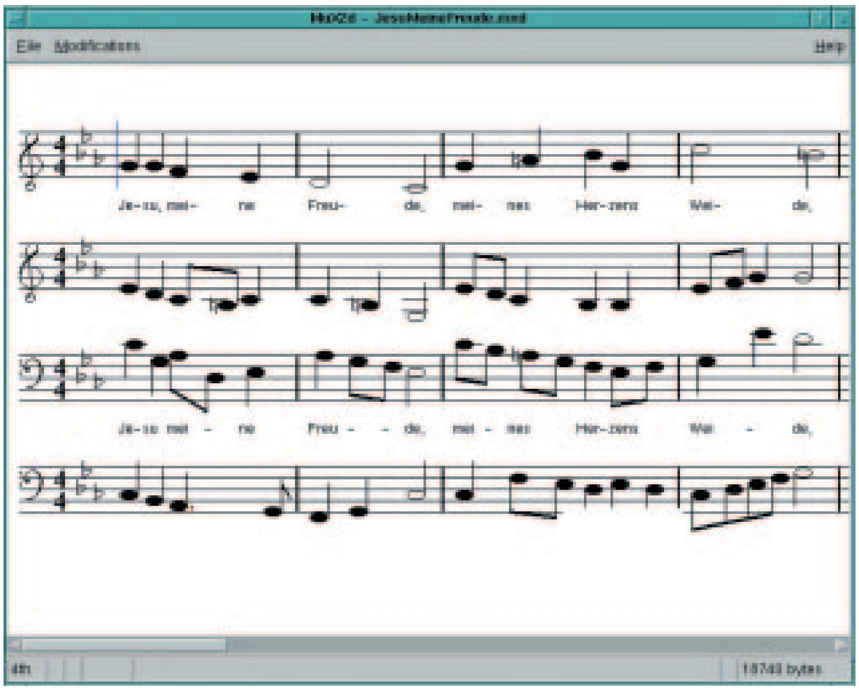
Qt Designer

Qt includes a graphical interface building tool called Qt Designer which can be used to build dialog boxes, interfaces and more. Qt Designer is an important component in your software development with Qt and can save a lot of time in creating your software.

Qt designer also has support for KDE widgets if



cRadio for controlling PCI Radio Cards


MuX2d - JesuMeineFreude.mxd
File Modifications
Help
Je-su, mei- ne Freu- de, mei- nes Her-zens Wei- de,
Je-su mei - ne Freu - - de, mei - nes Her-zens Wei - de,
18740 bytes

MuX2d music typesetting package for TeX

desired and can be compiled with this support. I will be covering installation of Qt in the next issue.

Documentation

One of the great benefits of Qt as a development tool is its incredible documentation. Qt includes documentation on all of the classes included in it as well as a tutorial, information on Qt modules and other information. This documentation is essential when coding to look up method names and details. Trolltech has built an application to support this documentation called Qt Assistant. Qt Assistant provides an interface to the documentation provided by Qt, and also enables more documentation to be added.

Internationalisation

If you want to write software in multiple languages (natural languages, not programming languages), then you are going to need to translate strings of text across your programs. Trolltech has developed Qt Linguist to assist in this process. Qt Linguist provides a number of features for making your programs more internationally aware.

Uses for Qt

For those of you new to Qt, you may be reading this article and wondering just what kind of software you can write with Qt? Well basically you can write any kind of software that you need to. Qt provides most of the visual GUI elements that you will need and Qt also provides the backend and internal facilities. Remember that Qt is a commercial product with a free edition that is not restricted or cut away at. You are getting an industry-strength toolkit for free.

KDE Integration

Many people are starting use KDE as their standard interface for Linux. This has been due to a number of solid KDE releases, an easy to use interface, and good quality application software.

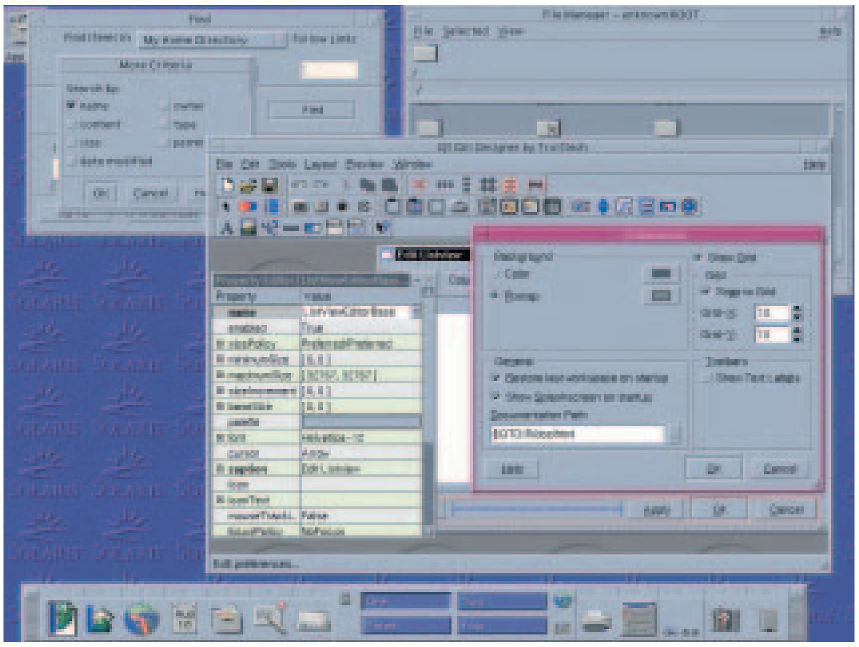
KDE itself is written in Qt. This is a major benefit as it means that you have a massive wealth of code already written for KDE that is available for a reference. KDE has also developed extensive features on top of the Qt features and making a KDE-aware program out of a Qt program is a nominal job.

This means that by learning Qt, you are also making yourself skilled enough to write software that is compatible with one of the most popular desktops for Linux.

Where to now?

Now we have had a brief look at Qt, we will be taking a look at writing some software with it. This will begin in the next issue, but before then, there are some preparations I suggest you take.
Qt is written in C++ and uses C++ as the language to write Qt software in (although there are other unofficial bindings). If you are unfamiliar with C++, I suggest you take a look at it and get to grips with it. It is a powerful language and needs some practice to get to use properly. There are plenty of good books and tutorials available to get you started.

Qt is a powerful, flexible toolkit for professional grade software development. Qt has been written from the ground up as a capable toolkit for development of free and commercial software, and using it is very satisfying. Next issue we will begin using Qt to write some software. I will hopefully see you then! ■


Find
File Manager – unknown ROOT
Qt GUI Designer by Trolltech
File Edit Tools Layout Preview Window
Help

Qt GUI Designer running in a Solaris environment