Fast, efficient user interface development for embedded systems

Abstract

The processing and display capabilities of consumer electronic and industrial devices have advanced rapidly over the past decade or so, as the costs of processor, memory and display hardware have all fallen. At the same time, end-users expect electronic devices to do much more, more intuitively and faster than ever before. This paper describes how Qt Embedded – a development of the Qt C++ desktop development framework – can help developers meet the needs of this evolving embedded systems marketplace, by offering shorter time-to-market, enhanced product performance, and the ability to target multiple software and hardware platforms.

NOKIA

Introduction

As use of the internet and computers become more widespread in people's everyday life, services and software is expected to work on an increasingly wider range of platforms. Processing power which was only available on high-end computing facilities in the early days of computing can now be found in cell phones and PDA's everywhere. Because of this, the computer is becoming fragmented into a variety of smaller form-factors which is expected to deliver the same services, with the same functionality and performance as their dedicated desktop counterparts.

The processing and display capabilities of consumer electronic devices and control panels have advanced rapidly over the past decade or so, as the costs of processor, memory and display hardware have all fallen. At the same time, end-users expect electronic devices to do much more, more intuitively and faster than ever before.
Traditionally, embedded systems were tasked with performing a limited set of functions, with only basic user interfaces. Today they often called upon to provide high-resolution color graphics, touch-screen control and dynamic content with web access.

Devices like mobile phones, MP3 players, gaming consoles and satellite navigation units have led the way in embedded systems development. Now, everything from office equipment and domestic appliances, through medical and test equipment, to industrial controls are keeping pace and offering more user-friendly, dynamic capabilities.
To succeed in what is a highly competitive market, embedded systems increasingly need to offer levels of performance, services, and user-friendliness approaching those available from desktop computers.

At the same time, manufacturers need to bring new products to market quicker than ever before, at lower production cost. Development cycles are shorter, while specialist developer skills are in shorter supply. Portable devices and control panels often need to mimic the look and feel of applications running on desktop PCs, and embedded applications may well need to run across multiple operating systems.

This paper describes how Qt Embedded – the embedded Linux and Windows CE version of the cross-platform Qt application framework – can help developers meet the needs of this emerging embedded systems marketplace, by offering:

- Shorter time-to-market and cycle-to-market
- Enhanced product performance
- Flexibility for a range of needs
- Ongoing global development and support.

Speeding time to market

Developers of embedded systems are increasingly expected to deliver complete products that combine advanced capabilities with a small software footprint, within tight budgets and timescales.

Today's more complex environment has driven the high-level C-programming skill sets to the limit. To exploit the capabilities of larger, full-color displays and powerful hardware running advanced operating systems, a new approach to rich UI embedded systems development is needed. This means developers need a complete set of high-level programming tools that fully utilize hardware potential to create advanced user interfaces, quickly and efficiently.

The Qt development framework – well established in desktop application development – has been adapted to make embedded system development just as easy and intuitive. With its high-level C++ programming environment and intuitive approach to object orientation, Qt increases programmer productivity by providing extensible prefabricated components for user interfaces along with classes and modules which abstract common functionality in operating systems. Qt Embedded eases developers' transition to embedded GUI design with full graphics user interface.

Desktop developers can quickly become developers for embedded systems, and the resulting code is highly readable and maintainable. This means there is less time spent on development, and shorter time-to-market for new products.

Flexible framework

Qt offers a modular library of over 700 proven C++ classes. These classes offer more than just GUI design. By offering classes for XML, networking, IPC, SVG, threading, SQL, internationalization and multimedia, Qt Embedded provides a complete application framework that enables developers to focus on their key competencies and product differentiators rather than the basics.

The Qt framework is very comprehensive but not all projects need the full capabilities of Qt. Qt Embedded can be configured to only include those modules that are required in order to keep memory requirements down.
Qt's long history of helping developers create innovative user interfaces is especially useful in the embedded world as many devices require unique interfaces tailored for their target audience. This ranges from HMI, simple push button controls for factory workers and other industrial uses, to some of the flashiest interfaces to help device manufacturers compete with the innovative iPhone from Apple.

Qt Embedded is designed from the ground up to be extensible. Every prefabricated user interface component can be customized in a number of ways, from the class level to customizing event handling, to specifying the color palette or a theme. Qt achieves this by making key functionality 'virtual', meaning a developer can replace this part of a class' functionality with his own implementation.

Design tool

Qt Designer enables embedded user interfaces to be developed in a graphical environment, offering developers the same advanced layout and design tools commonly available in the desktop world. This makes embedded development just as easy as desktop development and shortens design time. It also means developers are 'shielded' from the low-level development traditionally required for embedded systems – broadening the potential talent pool.
Qt's advanced layout engine also helps simplify the process of adapting interfaces for different screen sizes and orientations.

Desktop development

Quite often projects start before the hardware is available so there needs to be a way for developers to get started and to prototype their project. Even when hardware is available, the compile and download cycles even with today's modern embedded operating systems is a barrier to productivity. To address this, Qt for Embedded Linux has a virtual framebuffer implementation that runs on the host Linux development platform. The virtual frame buffer emulates the native resolution and color depth – accelerating development massively. In addition, developers can use a skin around the virtual frame buffer and define the location of buttons, enabling final applications to be demonstrated without any embedded system hardware.

Internationalization

Modern electronic devices must meet the needs of a global market. Qt Embedded includes a tool called Qt Linguist that enables developers to translate and adjust applications to different world languages, including support for Asian characters and right-to-left script.

Rapid portability

Qt Embedded has minimal hardware dependencies, offering simple platform migration across embedded Linux and Windows CE, as well as desktop Linux, Windows, Mac and Unix operating systems. This cross platform portability has great value where a product is developed for one operating system but for future generations a different one will be used. The Qt application code can simply be recompiled for the OS.

Another scenario demonstrating the value of the cross platform capability is when there is a large product portfolio using a variety of development groups what each may have chosen a different operating system. Maybe there's been a decision to standardize on Linux for the future but product lifecycles continue and switching operating systems may not be an immediate option. Introducing Qt allows engineers to be redeployed on other projects with minimal retraining and encourages code reuse across projects.

Desktop code can be ported to an embedded platform, and vice versa, just through recompiling. Application code can be written once and deployed across multiple device types as long as components and user interfaces adapt to differing screen sizes. Qt goes a long way to help with this, by providing a powerful layout engine which enables user interfaces to dynamically scale. However, some care must be taken for the smaller screen sizes.

This enables GUI reuse across a range of hardware platforms. The user interface would be able to retain a consistent unique look & feel and user experience, while working on different hardware and software platforms, such as Embedded Linux and Windows CE.

Qt Embedded has also been optimized to run on a vast array of hardware platforms. These include ARM, x86, PowerPC and MIPS – as well as native display depths ranging from 32-bit to monochrome.

Enhancing performance

The trend in embedded devices is towards high-resolution screens with much bigger area. There is also a strong trend towards touch-screen, icon-based interfaces with more structured menu options, using more advanced graphical features like edge smoothing, semi-transparency and gradients – and of using devices which has onboard hardware support for display acceleration.

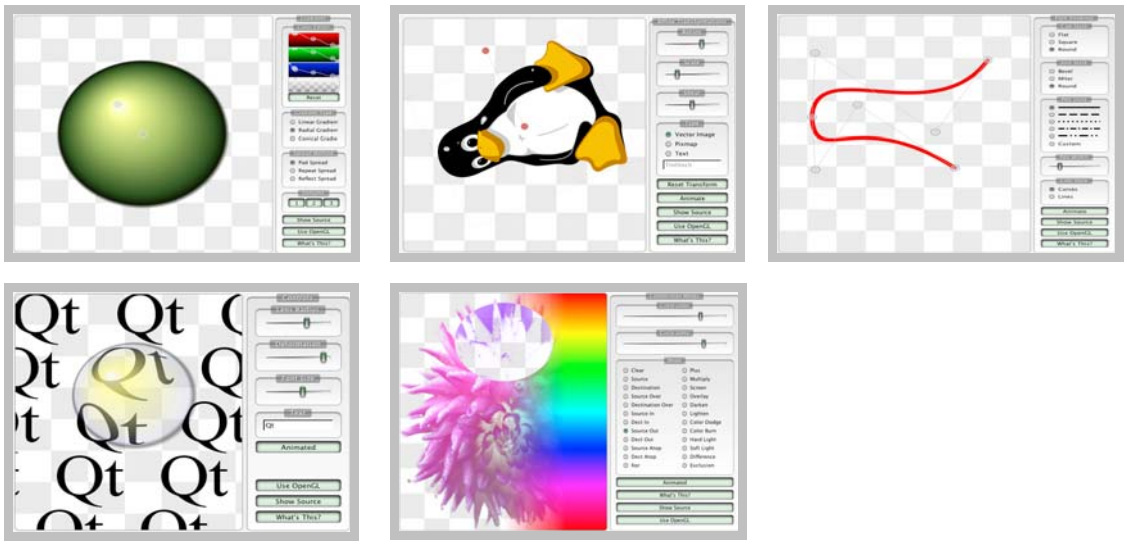
Embedded systems software needs to be able meet these requirements as efficiently and cost-effectively as possible. Qt Embedded is specifically designed to utilize the available hardware to its fullest potential, and enable advanced user interfaces to be created quickly and easily.

The multi-threading class library enables separate processor threads to be used for the user interface and for other functionality. This helps ensure that the interface stays responsive – enhancing end-user experience and satisfaction with the product.

Qt Embedded has its own windowing system, eliminating the need for X11 on embedded devices, and is easily extendable to allow for further customization.

High-speed, attractive graphics

The Qt framework allows the creation of highly attractive user interfaces for embedded devices through its advanced paint engine, which includes floating point coordinates – enabling smooth pixels in all rendering resolutions, support for lines, curves and gradients, blending and smoothing, composition mode support for alpha-channels (for semi-transparency), and support for Scalable Vector Graphics (SVG) and Portable Document Format (PDF) files. It also provides high-performance text rendering, including the ability to pre-render fonts to reduce processing overhead.



Qt Embedded supports OpenGL ES to accelerate the device, for example for 3D graphics and fast painting. It also supports OpenVG for accelerated 2D graphics. Accelerated hardware can be easily integrated, and multiple screens can be supported in a single device. For example, this enables the main and secondary displays of multi-screen devices to be driven from the same software and hardware.

Additional display drivers can be supported on embedded Linux by leveraging Qt's DirectFB support.

OpenGL® support

Qt includes support for OpenGL, the cross-platform standard for producing 2D and 3D computer graphics. With Qt you have a component which allows you to write OpenGL code and show a 3D scene. This component behaves much like a traditional UI component in Qt, except that it sets up an OpenGL display buffer for you where you can render using the OpenGL API. This makes it convenient to integrate existing OpenGL code into an application with support for Pixel Buffers and Frame Buffers.

You can also use this component with the Qt raster painting engine. When a QGLWidget is used as a paint device, paint primitives will use OpenGL commands, and as a consequence be accelerated by the graphics card. This happens transparently to the developer, as long as the paint device being painted on is a QGLWidget.

Greater flexibility

Portable devices and electronic control panels are naturally used across a wide range of sectors and applications, all of which will have their own particular user requirements.

With its 700-plus classes, including supporting modules for web access, networking and databases, and its ability to mix hard-coded and dynamic elements, Qt Embedded offers the flexibility to serve a wide variety of needs.

Connectivity options

Web access is becoming increasingly important across a range of embedded applications. For example, in industrial plant applications, a web access from a control panel can provide the operator with the latest operating and repair manual. This simple web access can also be used to download or upload production statistics, supply requests or support and maintenance information.

In consumer or business portable devices, web access can be used to retrieve dynamically updated information – such as an RSS feed – and deliver it to a small section of the screen. This could be news updates or company announcements, or feedback on performance or production status, for example. The web access could even be used to gather feedback from users through a simple touch-screen interface.

Qt Embedded integrates WebKit, enabling developers to enrich embedded systems with dynamic web content, through interaction with web servers and web services. This content could include online maps, music services, social networking, instant messaging or up-to-date web content such as 'how-to' demonstrations or visually appealing documentation.

The integration of Qt and WebKit enables developers to build advanced user interfaces that incorporate real-time web content and services more smoothly and easily than previously possible. This will help create dynamic, interactive software that combines online and offline content, in a way that enables users to define their own browser environment – guaranteeing a consistent user experience across operating systems and devices.

Qt Embedded provides a complete database interface that enables databases to be used in embedded devices, together with advanced methods to display or input this data. It also has a cross-platform interface for writing TCP/IP clients and servers, which supports IPv4 and IPv6.

User interface freedom

Touch-screen, icon-based interfaces are increasingly popular for small-screen portable devices and control panels. Qt Embedded supports these, plus a wide range of other input and output options.

Cascading style sheets (CSS) are supported for most widgets available in Qt Embedded. This enables the look and feel of applications to be customized freely, for example to match a customer's design – or simply support the host platform's native look and feel – without the need to rewrite any code. Advanced painting capabilities enable attractive visual features like alpha blending, semi-transparency and gradients.

A complete multimedia framework is now available in Qt Embedded to enable audio and video playback. This builds on and abstracts existing formats and frameworks on the host platform (QuickTime on Mac, DirectShow on Windows, GStreamer on Linux). The framework focuses on playback of multimedia content, making it easier to use existing codecs on small-screen devices. This could be used, for example, for video/audio calls in VoIP phones, or to play back multimedia content on a portable device.

Qt enables native look and feel on each supported platform, by providing and using a style API. Styles are used to draw window controls and decorations with the platform's own drawing functionality – ensuring a truly native look and feel. This can be further extended by developers to define application-wide styles which can ensure a unified look an feel to differentiate a product from the competition.

Continuous development

Qt Embedded draws from over a decade of development and refinement of the Qt framework, which is used by more than 5,000 customers, including significant players like Adobe, Google, Nasa and Skype.

Qt Embedded itself is a proven and professional toolkit, with millions of installations around the world.

Qt's professional programmers continue to develop and improve Qt Embedded, and the company is committed to the future cross-device and cross-platform development of features and bug-fixing, as well as professional support for users.

About Qt:

Qt is a cross-platform application framework. Using Qt, you can develop applications and user interfaces once, and deploy them across many desktop and embedded operating systems without rewriting the source code. Qt Development Frameworks, formerly Trolltech, was acquired by Nokia in June 2008. For more details about Qt please visit http://qt.nokia.com.

About Nokia

Nokia is the world leader in mobility, driving the transformation and growth of the converging Internet and communications industries. We make a wide range of mobile devices with services and software that enable people to experience music, navigation, video, television, imaging, games, business mobility and more. Developing and growing our offering of consumer Internet services, as well as our enterprise solutions and software, is a key area of focus. We also provide equipment, solutions and services for communications networks through Nokia Siemens Networks.

Code less.
Create more.
Deploy everywhere.

NOKIA

Oslo, Norway +47 21 60 48 00 ■ Redwood City, CA, USA +1 650-551-1676 ■ Beijing, China +86 10 6588 8836 ■ qt-info@nokia.com qt.nokia.com