



When Execution Matters

# Merits of QT for developing Imaging Applications UI

Amitkumar Sharma  
January 08, 2008



## Table of Contents

1.0 Executive Summary. -----	3
2.0 GUI Applications with Qt. -----	3
2.1 Key Features. -----	3
2.2 Layouts. -----	4
2.3 Inter - Object communication. -----	4
3.0 Graphics. -----	5
3.1 Image Manipulation. -----	5
3.2 Graphics View Framework. -----	7

## 1.0 Executive Summary

The Graphical User Interfaces (GUI) for applications is getting more intuitive in terms of ease of use, attention to details, and the experience offered to the user. Although RAD tools like Visual Basic are capable of creating user interfaces in quick time, the flexibility they provide is very poor in terms of customizing the visual effects of the user interface. In the process the user experience is diminished.

The Microsoft solution Microsoft Foundation Classes (MFC), which is built over Microsoft Visual C++, offers considerable support in building applications in Windows. Although MFC is ideal for Windows application development, building complex and visually appealing GUI is very time consuming.

Qt is the C++ framework for developing cross-platform software. The framework of Qt is built in a manner that it can seamlessly get integrated with several different development platforms like Microsoft Visual C++. The Qt C++ framework has been at the heart of commercial applications since 1995. Qt is widely used by fairly large organizations like Adobe®, Boeing®, Google®, IBM®, Motorola®, NASA, Skype® etc., as well as by numerous smaller organizations.

## 2.0 GUI Applications with Qt

Building modern GUI applications with Qt is fast and simple, and can be achieved by hand coding, or by using Qt Designer - Qt's visual design tool.

### 2.1 Key Features

- Qt provides a large set of framework classes and functionalities necessary to create modern Imaging GUI applications. Qt can be used to create window based or dialog based applications. It also supports development of SDI and MDI applications.
- Qt supports all GUI features built in – such as drag n drop, clipboard, common dialogs etc.
- Qt supports storing application settings in a platform-independent way by making use of the native methods in each platform. This enables the developer to store information such as user preferences, most recently used files, window and toolbar positions and sizes to be recorded for later use without knowledge of the target compilation platform.
- Qt framework has built in multithreading support.
- Applications built using Qt can make use Qt's desktop integration features to interact with services provided by the user's desktop environment. These ranges from System tray icon, which is often used by long-running background applications to provide a persistent indicator in the system tray, to Desktop services, which allows resources such as hyperlinks to get it processed by the most appropriate applications on each platform.

- Qt comes with various UI style implements. These provide the “look and feel” of the user interface on a particular platform. Qt automatically uses the native desktop style for an application’s look and feel. Qt applications respect user preferences for colors, fonts, sounds, and other desktop settings. Qt programmers are free to use any of the supplied styles and can override any preferences. Programmers can modify existing styles or implement their own styles using Qt’s powerful style engine. A style is complemented by the user’s desktop settings, which include the user’s preferences for colors, fonts, sounds, etc. Qt automatically adapts to the computer’s active theme. For example, Qt supports scroll and fade transition effects for menus and tooltips.

## 2.2 Layouts

Layouts provide a powerful and flexible alternative to fixed sized windows and control positions. Layouts free the programmers from having to perform size and position calculations, and provide automatic scaling to suit the user’s screen, language, and fonts. Qt provides layout managers for organizing child widgets (basic user interface object) within their parent widget’s area. They feature automatic positioning and resizing of child widgets, sensible minimum and default sizes for top-level widgets, and automatic repositioning when the content or text font changes.

Layouts are also useful for internationalization. With fixed sizes and positions, the translation text is often truncated; with layouts, the child widgets are automatically resized. Additionally, widget placement can be reversed to provide a more natural appearance for users who work with right-to-left writing systems.

## 2.3 Inter - Object communication

Qt provides a flexible but powerful loosely coupled mechanism – called “signals and slots” -- for inter-object communication. This mechanism can be used to completely replace the callbacks and message maps used by legacy toolkits. Signals and slots are flexible, fully object-oriented, and implemented in C++.

Generic callback mechanisms always suffer in the case of type safety. Qt’s Signal-Slot mechanism has built in type checking to ensure the type safety of the callbacks.

Qt’s signals and slots mechanism is different. Qt widgets emit signals when events occur. For example, a button will emit a “clicked” signal when it is clicked. The programmer can choose to connect to a signal by creating a function (a “slot”) and calling the “connect”) function to relate the signal to the slot. Qt’s signals and slots mechanism does not require classes to have knowledge of each other, which makes it much easier to develop highly reusable classes. Since signals and slots are type-safe, type errors are reported as warnings and do not cause crashes to occur. Connections can be added or removed at any time during the execution of a Qt application. They can be set up so that they are executed when a signal is emitted or queued for later execution, and they can be made between objects in different threads.

## 3.0 Graphics

Qt supports raster and vector graphics. It can load and save large number of image formats including BMP, GIF, JPEG, MNG, PNG, PNM, TIFF, XBM and XPM. Qt can export text and graphics to Portable Document Format (PDF) files. Qt can draw transformed Unicode rich text, Scalable Vector Graphics (SVG) drawings, and provides a fully-featured canvas for demanding interactive applications.

### 3.1 Image Manipulation

Graphics are drawn using device-independent QPainter objects that allow the developer to reuse the same code to render graphics on different types of device, represented in Qt by paint devices. This approach ensures that a wide range of powerful painting operations are available for each of the devices supported by Qt, and also allows developers to choose the devices that are most suitable for their needs.

Graphical applications that require an interactive canvas can take advantage of the Graphics View framework to manage and render scenes with large numbers of interactive items using multiple views, if necessary.

Qt's support for OpenGL enables developers to take advantage of the modern graphics hardware on desktop platforms. It also enables integration of the 3-D graphics in the imaging applications.

Qt supports device-independent colors. Colors are specified by ARGB, AHSV, or ACMYK values, or by common names (e.g., "skyblue"). Qt's color channels are 16 bits wide, and colors can be specified with an optional level of opacity; Qt automatically allocates the requested color in the system's palette, or uses a similar color on color-limited displays.

Qt provides advanced features such as transformations and clipping. It also provides standard functions to draw points, lines, ellipses, arcs, Bezier curves, and other primitives. More complex painting operations include support for polygons and vector paths, allowing detailed drawings to be prepared in advance and drawn using a single function call. Text can also be painted directly with a painter or incorporated in a path for later use.

Qt's painting system also provides a number of advanced features to improve overall rendering quality:

- Alpha blending and Porter-Duff composition modes enable the developer to use sophisticated graphical effects and provide a high level of control over the output on screen.
- Anti-aliasing of graphics primitives and text can be used to mimic the appearance of a higher resolution display than the one in use.
- Linear, radial, and conical gradient fills allow more detailed graphics, such as 3D bevel buttons, to be created without much effort by the developer.

Qt supports clipping using regions composed of rectangles, polygons, ellipses, and vector paths. Complex regions may be created by combining simple regions using standard set operations.

QImage and QPixmap are the classes provided by Qt for input, output, and manipulation of images. QPixmap is usually used when applications need to render images quickly. The images with QPixmap are stored in the VGA memory of the display driver for swift display. QImage is a device-independent image and is more useful for pixel manipulation, and handles images in a variety of color depths and pixel formats. Programmers can manipulate the pixel and palette data, apply transformations such as rotations and shears, and reduce the color depth with dithering if desired. Support for “alpha channel” data along with the color data enables applications to use transparency and alpha-blending for image composition and other purposes.

The range of graphics file formats that can be used with these classes can be extended through the use of an extensible plugin mechanism.

Qt provides QPainter class which can operate on any paint device. The code required to paint on any supported device is the same, regardless of the device.

Qt supports the following paint devices:

- All QWidget subclasses are paint devices. Qt uses a backing store to reduce flickering during the painting process.
- All QGLWidget subclasses are paint devices that convert standard QPainter calls to OpenGL calls, enabling two-dimensional graphics to be accelerated on devices with appropriately supported hardware.
- A QImage is a device-independent image with a specified color depth and pixel format. Images can be created with support for varying levels of transparency and painted onto custom widgets to achieve certain effects.
- A QPixmap is essentially a QImage with the same properties as a widget on the screen, and is often a system device that can be accessed quickly and efficiently.
- A QSvgGenerator represents a Scalable Vector Graphics (SVG) drawing. Paint commands are translated to the corresponding structures in the SVG file format.
- A QPicture is a vector image that can be scaled, rotated, and sheared gracefully. Pictures are stored as a list of paint commands rather than as pixel data.
- A QPrinter represents a physical printer. On Windows, the paint commands are sent to the Windows print engine that uses the installed printer drivers. On Unix, PostScript® is written out and sent to the print daemon. Portable Document Format (PDF) and PostScript files can be generated on all platforms, enabling applications to create high quality documents that can be viewed using suitable reader applications.

## 3.2 Graphics View Framework

Qt 4 introduces a powerful new framework for interactive graphical applications that is used to manage and display large numbers of items in a two-dimensional scene. Graphics View provides both an object-based API for adding new items to a scene and a traditional canvas-style API containing convenience functions for creating predefined items. Once created, items can be placed with the required position, orientation, and scale in a scene.

Graphics View has been designed with animations in mind. Qt can be used to create animated objects that are transformed according to a series of transformations defined at certain points on a timeline. Each animated item is run according to a time line that controls the rate and duration of its animation. Qt provides support for printing the scene, either directly to a printer or to a PDF file.

## About the Author

Amit kumar Sharma, a Senior Software Engineer at Trianz, has more than 2.5 years of experience in implementing Windows applications. His expertise is in C++, VC++, MFC, QT, image processing and network protocols. Since joining Trianz, Amit has designed and implemented the core modules for a couple of client products. With his excellent problem solving skills, he has resolved very critical and tricky issues in the products.

## About Trianz

Trianz is a management consulting, technology, engineering, and outsourcing services firm that helps senior leaders execute business and technology initiatives. A unique mix of business consulting and technology capabilities has helped Trianz rapidly grow since its inception in 2000. Trianz provides services to a diversified global client base. Clients are result-focused leaders employed in businesses ranging from Fortune 1000 corporations to emerging rapid-growth companies. Service offerings focus on the following areas:

- Operations Consulting Services

- Enterprise Applications Services

- Software Product Engineering Services

- Transformational Outsourcing Services

Disclaimer for white papers

©2006 Trianz. All rights reserved

*Copyright in whole and in part of this document belongs to Trianz, Inc. This work has been provided for informational purposes only, and may be copied for personal use only. This work may not be used, sold, transferred, adapted, abridged, copied, or reproduced in whole or in part, in any media, by enterprises, without the prior written permission of Trianz, AND an acknowledgement of "Trianz" as the source of the content. All trademarks and copyrights mentioned in this white paper are the property of their respective owners. Neither the author nor Trianz bears any responsibility for damage resulting from the use of the information contained herein.*