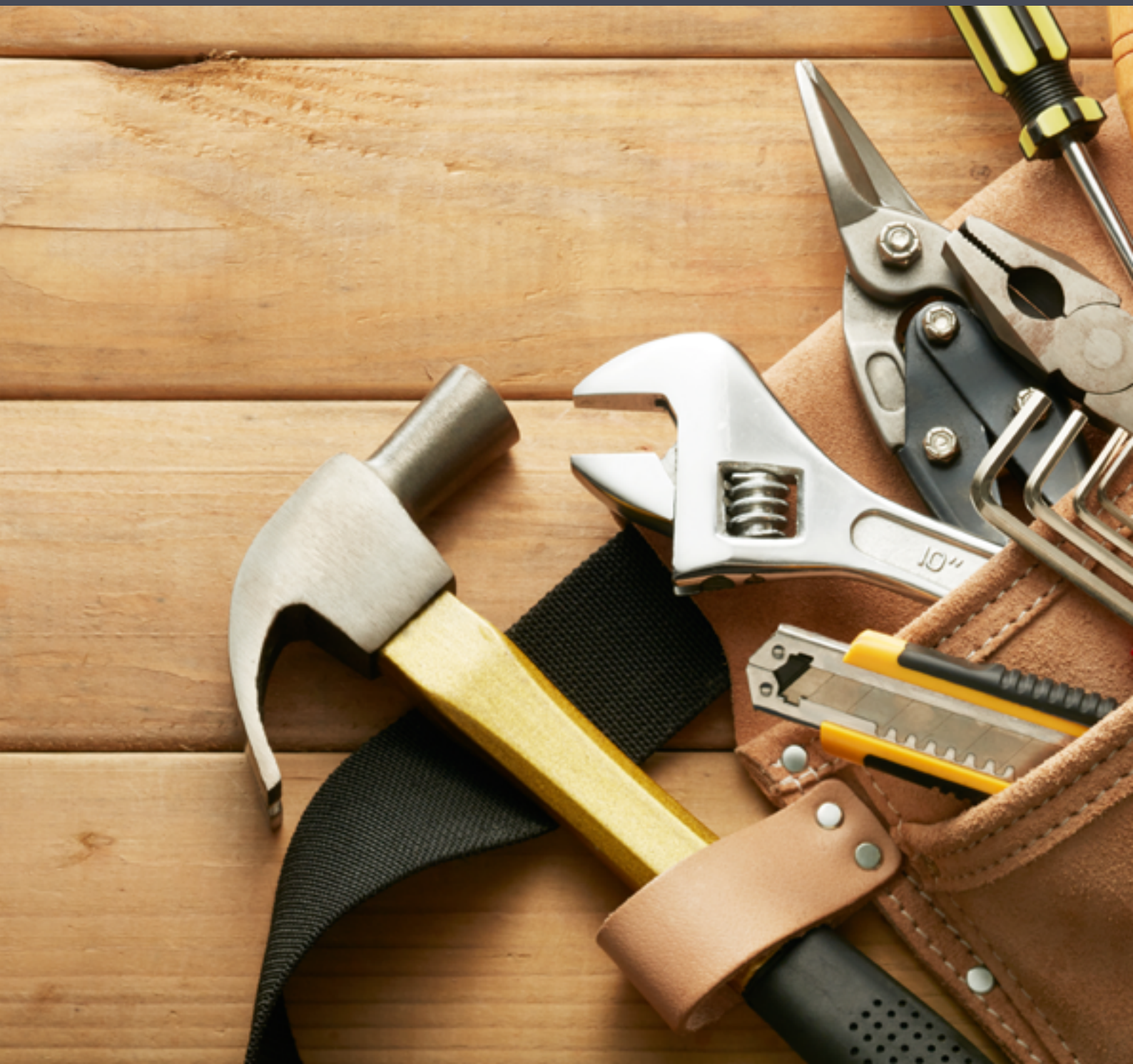
# Modern Qt Development

## The top 10 tools you should be using

KDAB

Matthias Kalle Dalheimer | President and CEO



**It takes time to learn a new tool – so how do you know where to invest your time? Here are our favourites.**

GammaRay provides Qt-aware debugging and live value edits for significantly better and faster debugging of Qt apps.

Why is using the right tool for the job so important? Efficiency and results are two reasons that immediately spring to mind. You don't see construction workers using shoes to drive in nails – so why as software developers do we so often make do with manual solutions to find bugs or optimize code? It's certainly less efficient and much more frustrating, and the final results can be less than ideal.

It always takes time to learn a new tool – so how do you know where you should invest your time? We at KDAB would like to share our favourite Qt development tools that we think are worth the effort. We regularly rely on these tools to help us locate and fix troublesome bugs and solve difficult optimization challenges. If you live at the cutting edge of Qt development, you may be familiar with many of these, but we know there'll be something new for you regardless of your level of expertise.

## GammaRay

***Introspection tool that adds Qt-awareness to the debugger***

If you've been frustrated by debugging through endless Qt structure internals, you'll definitely want to give this a try. GammaRay understands most of the core Qt components from a Qt perspective – QtQuick scene graphs, model/view structures, QTextDocuments, signal/slot activations, focus handling, GPU textures, QWidgets, state machines, and more – letting you observe and edit values at run-time in a natural way. You can debug applications from launch or attach to running apps (both local and remote).

KDAB page: GammaRay

KDAB blog: GammaRay release

**Clazy gives you compile-time warnings for Qt best practices: it's a great way to improve your Qt code.**

**You don't need to be a C++ standards expert to make small changes that significantly help.**

### Clazy

***Compiler plug-in that understands Qt semantics***

This is something that needs to be part of every Qt developer's bag of tricks. By adding clazy to clang, you'll get compile-time warnings for Qt best practices – unneeded memory allocations, misused APIs, or inefficient constructs. Clazy is a great way to improve your Qt code and best of all, it can provide automatic refactoring fixes for some of the errors it finds – no coding required!

KDAB video: Clazy

KDAB blog: Clazy visualizer

KDAB blog: Clazy release

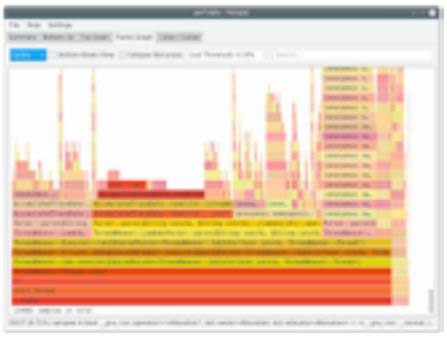
### Modern C++

***Source code that uses C++11/14/17 improvements***

Although C++11 and C++14 have been around for a while now, there are many old coding habits that die hard. Many developers don't take advantage of newer C++ constructs that are more efficient, understandable, and maintainable. The thing is you don't need to be a C++ standards expert to make small changes that can significantly help your code. We've got a few of the more important highlights covered in a paper below – or you can attend a training class or two for the real low-down.

KDAB brochure: modernizing C++

**Clang Tidy flags older C++ idioms that could use updating, including some it can refactor automatically.**



HotSpot interprets and visualizes Linux perf logs so you don't have to

## Clang Tidy

***Compiler tool to help modernize your C++***

This is the lazy person's way to modernize C++. Another clang-based tool, Clang Tidy points out older C++ idioms that could use updating. It flags where these should be replaced with new C++11 or C++14 improvements, and in many cases can do the refactoring automatically. That's productivity for you!

KDAB blog: clang tidy modernizing source

KDAB blog: clang tidy integrating build systems

## HotSpot

***Tool to visualize your app's CPU performance***

When it comes to optimizing, nothing beats a profiler – but reading raw perf logs is a punishment that should only be reserved for people who believe zip files are a proper form of source control. HotSpot reads Linux perf logs and lets you see multiple different views (callers, timeline, top-down, bottom-up) to help you easily understand where you're burning up your time.

KDAB video: HotSpot

KDAB blog: HotSpot

## apitrace

***Set of tools to debug your graphics APIs and improve their performance***

If you're writing an app with a GUI, profiling doesn't stop at your C++ code. You need a way to see the calls you're making to OpenGL, Direct3D, or DirectDraw, view the contents of those calls in a graphical interpretation, and profile their performance. That's exactly what apitrace does. It can also replay a trace file, allowing you to compare and benchmark performance once you've seen where to make improvements.

https://apitrace.github.io/

**A system profiler is an incredibly invaluable tool that can find problems that nothing else will.**

## Kernel/System Profiler

*Tools for visualizing your operating system's performance*

Sometimes performance problems aren't found in your app – they're in multiple-process interactions, hidden in driver stacks, or a result of how you're calling the operating system. For these kinds of really low-level debugging, you've got to have a system-profiling tool. It can feel like swatting a fly with a bazooka, but a system profiler is an incredibly invaluable tool that can find problems that nothing else will.

**OS-level debuggers are specific to each platform, but multiple options are usually available.**

*Linux:*

- KDAB blog: slaying latency with linux kernel tracepoints
- LTTng
- TraceCompass
- Intel VTune
- CodeXL

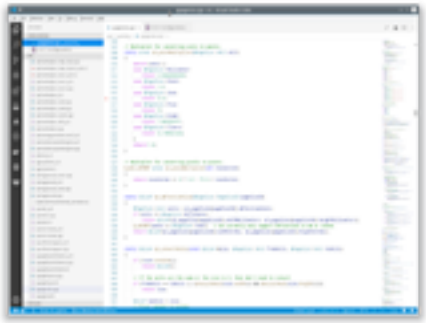
*Windows:*

- Intel VTune
- CodeXL

*QNX Neutrino RTOS:*

- QNX System Profiler

**Heaptrack has several tools for finding your app's biggest memory hogs and putting them on a diet.**



A programmer's IDE choice can be contentious – other popular choices besides Qt Creator are Sublime Text and Visual Studio Code.

## Heaptrack

*Tooling to watch your app's memory usage*

Sometimes optimization isn't about speed – it's about memory. If you're trying to profile your application's memory usage, you'll want to check this out. By showing your application's peak memory usage, memory leaking functions, biggest allocators, and most temporary allocations, you'll be able to really narrow down where your app's memory is going and how to keep it on a diet.

KDAB Blog: Heaptrack release

## Qt Creator

*The Qt IDE*

Perhaps you think it's cheating to include Qt Creator in this list since it's already installed on every Qt developer's desktop. Yes, but did you know you can find slow spots in your Qt Quick code through the built-in QML profiler? How about hitting Alt+Enter to get a list of all the refactoring options at the cursor location? What about other handy key sequences to find symbol references, do a git diff, or record a macro, along with many other super helpful navigation and editing aides? Shortcuts that you might use ten times a day if you only knew they were there. Don't be a slave to your mouse – print out our handy reference card and pin it up on your cube wall.

Qt documentation: QML Performance Monitor

Qt documentation: refactoring

KDAB resource: Qt Creator reference card

**A new entry on our tool list is C++ Best Practices, an invaluable resource for C++ programmers.**

**The other tool that can improve your coding is you! Don't forget to regularly train and update your skills.**

### Extras

When we originally published this article, we included just our top ten Qt developer tools. Others in the community have since pointed out some great resources we were missing, and we'd be neglectful if we didn't mention just one more: C++ Best Practices. This is a github page maintained by Jason Turner (lefticus) of the C++ Weekly YouTube series and CppCast C++ podcast. It's got a ton of great tool suggestions and links, and while many of them are included here, there are so many other good suggestions on this page that we had to include it in our list.

C++ Best Practices

### Summary

Those are our top ten (plus one) tools for improving your Qt development tool chest. Don't forget there are also things that can't be automated but for which there are courses and customized training – such as effective code reviews or best coding practices.

**Nearly every single one of these tools is open source and free – there's no reason you can't start employing them today.**

KDAB offers many training courses on development best practice, tool deep dives, profiling and debugging, graphics, and more.

## About the KDAB Group

The KDAB Group is the world's leading software consultancy for architecture, development and design of Qt, C++ and OpenGL applications across desktop, embedded and mobile platforms. KDAB is the biggest independent contributor to Qt and is the world's first ISO 9001 certified Qt consulting and development company. Our experts build runtimes, mix native and web technologies, solve hardware stack performance issues and porting problems for hundreds of customers, many among the Fortune 500. KDAB's tools and extensive experience in creating, debugging, profiling and porting complex applications help developers worldwide to deliver successful projects. KDAB's trainers, all full-time developers, provide market leading, hands-on, training for Qt, OpenGL and modern C++ in multiple languages.

www.kdab.com