KDAB | training

TrainingGuide

www.kdab.com

KDAB 20 years ahead

Qt, C++ and OpenGL Experts

Dear Reader,

A word about training with KDAB

KDAB is now over 20 years old and was founded in 1999 by Matthias Kalle Dalheimer. One of the key elements of Kalle's original vision for KDAB was to create a vibrant knowledge community for talented top-notch professionals working together across borders and continents to develop sophisticated software solutions – ahead of the industry.

KDAB is the recognized leader in Qt training. We have honed our offering over more than 15 years, specifically focusing on core engineering competencies like C++, OpenGL, platform creation, and UX design in the embedded space as well as automotive.

These are some of the principles that drive KDAB training success:

- Always engineers. We use active engineers who love teaching, not just dedicated trainers. That's a big difference – our trainers put their material into practice every day with insights into real-life problems and solutions. They can show you exactly what you need to fix your specific issue. Many of our trainers are recognized and active contributors in their specialties to open source projects and/or standards work.

- Always hands-on. Our three-day training courses are split between presentations and hands-on labs, not just statically delivered. The labs let students code important techniques themselves, reinforcing the lessons and making the material stick. Our small class sizes ensure one-on-one assistance so every student feels comfortable before moving on to new concepts.

- Always enthusiastic. Our trainers are enthusiastic about both giving training classes and the topics being taught. It's a safe bet you will find the trainer eagerly discussing corner cases with students after a class, and if you check out KDAB blogs, you'll often find them posting new material there too.

- Always focused. The courses we teach are focused on delivering. Firstly, we adapt to suit each class, modifying class curriculum on the fly to educate and challenge everyone in the room. Secondly, we narrow in on frequent problems and common misunderstandings to address real-world trouble spots. And finally, we do our best to include content tailored to our students' particular goals.

- Always small classes. In our regular scheduled training we never sell more than 12 seats, and in onsite trainings we urge the customer not to put more people into the class than that. The reason is simple: the more people there are in class, the less the training will be for your exact needs. Also, fewer people means our trainer has more time to help and guide you during labs. In a regular training we aim for parity between presented material and labs. This gives participants maximum opportunity to play, practice and learn at their own pace, so they can immediately put to use what they have learnt in a KDAB training when they get back to work.

  We hold scheduled training classes throughout the year in Europe, the USA and Asia that are guaranteed to run. We look forward to seeing you at one of them!

Qt

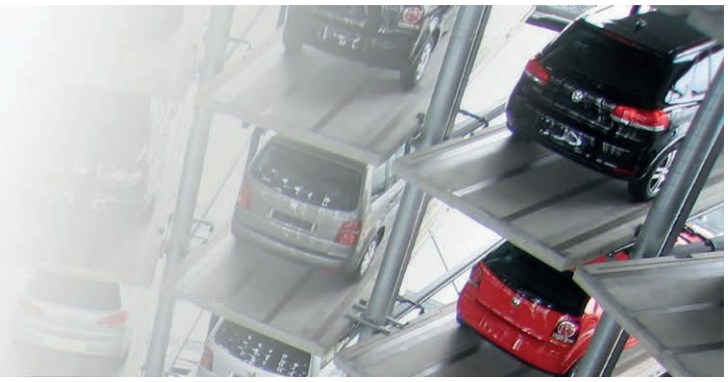
C++

```
QPushButt
AbstractButton::click
PushButton;
ed(false);
(style()->standardIcon
&QAbstractButton::cli
= new QSlider(Qt::Horizont
->setRange(0, 0);
ositionSlider, &QAbstractSlider:
&VideoPlayer::setPosition);
*controlLayout = new QHBoxLayout
->setMargin(0);
->addWidget(openButton);
->addWidget(m_playButton);
```

OpenGL

Qt 3D

Qt Automotive



contact us: training@kdab.com

About KDAB Trainings

Improve your productivity with Qt, OpenGL and C++

KDAB is the world's largest independent source of Qt knowledge and the market leader of training for Qt, OpenGL, Qt 3D and C++.

KDAB has a wealth of well-structured, constantly updated training material and we offer hands-on, practical programming training for beginners as well as experienced developers, in Europe, Asia and the USA.

Our experienced instructors are all expert developers working on real-life customer projects. You will get the best-in-class training from Qt, OpenGL, Qt 3D and C++ experts.

We run scheduled 2 day and 3 day courses for a maximum of 12 developers at a time, occasionally offering shorter courses at special events such as Qt World Summit. Customized longer courses can be arranged in-house.

Shell, IBM, Boeing, Schlumberger, Accenture, EADS, Motorola, Ericsson and Siemens are examples of the many companies whose engineers have become more productive as a result of training with KDAB.

Choose a scheduled training if you:

- can wait for and travel to a training course in one of our training facilities
- need training for yourself or only a few engineers in your company
- like to discuss your experience and get inspiration from other engineers that participate.

Choose an on-site training if you:

- want your training at a specific date and location you choose
- are a group of developers with similar needs; e.g. working with the same project or the same company
- want a more tailored training – with a focus on a set of particular topics.

The majority of our scheduled trainings are held in Germany, the United Kingdom, France and the United States in either English, French or German.

Many other language options are available on request.

See our Training Schedule:

Course selector

Check what you need to do most

Choose the course that fits your specific challenges

	Challenges ▶ / Courses ▼	Upgrade platforms	Migrate to Qt	Improve UX	Incorporate realistic rendering or 3D	Optimize or improve performance	Support multi-platform or mobiles	Build internal frameworks/tools	Improve processes	Improve testing	Build to standards	Create toolkits/SDKs
Qt	Qt Widgets for the Desktop	✓	✓				✓	✓				
	Advanced Programming with Qt Widgets	✓	✓	✓		✓		✓				✓
	Qt/QML	✓	✓	✓								
	Qt/QML for Embedded	✓	✓	✓			✓					
	Advanced QML	✓	✓	✓		✓		✓				✓
	Qt's Model/View framework for widget dev.	✓	✓	✓		✓						
C++	Modern C++: Introduction	✓	✓						✓	✓		✓
	Modern C++: C++ 11 / C++ 14 / C++ 17	✓				✓		✓	✓	✓	✓	✓
OpenGL	Qt 3D			✓	✓		✓					
	Qt 3D Studio			✓	✓							
	Modern OpenGL			✓	✓		✓				✓	
	Modern OpenGL: Rendering and Effects			✓	✓	✓						
	Modern OpenGL: Pipeline and Performance				✓	✓	✓					
Tools	Debugging & Profiling for Qt Development (Windows/Linux)	✓				✓	✓		✓	✓		
	Debugging & Profiling for C++ Development (Windows/Linux)	✓				✓	✓		✓	✓		
	CMake	✓							✓		✓	✓
	Testing Qt with Squish		✓					✓	✓	✓	✓	✓
	User-Centered Development and Usability	✓		✓	✓		✓					
	In-depth Multithreading	✓	✓	✓		✓	✓	✓				
Car	Qt Automotive Suite	*Auto courses may be applicable to all challenges within automotive industry*										
	Qt for Automotive Development											

INTRO

Qt Courses

Qt Widgets for the Desktop

Improve your teams programming skills

This introductory Qt course:

- is designed to take development teams in your organization that are new to Qt, from the basics to a deep functional understanding of the best Qt practices
- equips developers with the Qt skills and know-how to boost their productivity at work
- offers hands-on Qt training with different kinds of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Designing forms and dialogs with Qt Designer
- Layout management
- Event handling
- Model/View programming

Target audience:	Developers targeting desktop
Prerequisite:	Students should have a functional knowledge of C++ but no previous experience with Qt is required. Developers with Qt/Widgets experience will easily benefit from this course which is regularly updated.
Duration:	3 days

Why learn Qt?

Qt 5 is a modern, mature and highly featured frame work for developing application software that can run across desktop and/or embedded and/or mobile platforms. Qt is often used in combination with other technologies. KDAB's Qt experts are the maintainers (steer the development) of Qt for Android, Qt for QNX and Qt for Windows CE as well as the areas of Qt 3D (OpenGL), Qt Widgets, Qt WebChannel and parts of Qt Core.

Qt is used by tens of thousands of companies across 70+ industries and in all regions of the world. It is available for development with an open source license or with a commercial license.

KDAB can help you determine whether you need licensing as well as help you evaluate whether Qt is the right fit for your project.

Since KDAB was the original Qt training provider and still is the market leader in Qt training, you will always enjoy a tailored training with up to-date materials, hands-on programming exercises and trainers who are active project engineers.

Find out more:

contact us: training@kdab.com

ADVANCED

Qt Courses

Advanced Programming with Qt Widgets

Take your understanding to the cutting edge

This Advanced Qt Widgets course:

- is designed to take programmers who are already acquainted with Qt Widgets to a deeper level of functional understanding of Qt internals
- equips you with the cutting edge Qt Widgets skills and know-how to boost your productivity at work
- provides you with detailed knowledge about the versatile frameworks for large application development and system integration
- offers hands-on training with different kinds of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Qt Fundamentals Recap
- Application Infrastructure
- Model/View with Qt
- Graphics and Styling
- Text Documents
- Supplemental Topics including:
  - In-depth Application Infrastructure
  - In-depth Graphics
  - In-depth Qt Core
  - Development and Testing
  - Multithreading,
  - XML, JSON and SQL
  - Interprocess Communications

Target audience: Experienced Qt developers.

Prerequisite: Programmers enrolled in this Advanced Qt Widgets course should already have a functional knowledge of Qt Widgets. Developers with more than one year of Qt experience will benefit from this course which is regularly updated.

Duration: 3 days*

* A typical Advanced Qt Widgets training lasts 3 days and includes time for core topics and specific areas of focus, which vary depending on a poll we make among participants to ensure topic relevance.

Find out more:

INTRO

Qt Courses

Qt/QML

Learn Qt/QML with courses tailored to your team's needs

This introductory Qt/QML course:

- is designed to take development teams in your organization that are new to Qt or QML, from the basics to a deep functional understanding of best practices
- equips developers with the skills and know-how to boost their productivity at work
- offers hands-on Qt training with different kinds of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Connecting a QML UX with C++ business logic
- Complex list views including data provided from C++ models
- Custom objects implemented using Qt Quick scene graph
- Includes time for core topics to establish a working knowledge of QML
- Profiling and best practices

Target audience:	Developers targeting desktop, embedded or mobile platforms
Prerequisite:	Students should have a functional knowledge of C++ but no previous experience with Qt is required. Developers with Qt/QML experience will easily benefit from this course which is regularly updated.
Duration:	3 days

Why learn Qt?

Qt 5 is a modern, mature and highly featured frame work for developing application software that can run across desktop and/or embedded and/or mobile platforms. Qt is often used in combination with other technologies. KDAB's Qt experts are the maintainers (steer the development) of Qt for Android, Qt for QNX and Qt for Windows CE as well as the areas of Qt 3D (OpenGL), Qt Widgets, Qt WebChannel and parts of Qt Core.

Qt is used by tens of thousands of companies across 70+ industries and in all regions of the world. It is available for development with an open source license or with a commercial license.

KDAB can help you determine whether you need licensing as well as help you evaluate whether Qt is the right fit for your project.

Since KDAB was the original Qt training provider and still is the market leader in Qt training, you will always enjoy a tailored training with up to-date materials, hands-on programming exercises and trainers who are active project engineers.

Find out more:

INTRO

Qt Courses

Qt/QML for Embedded

Learn Qt/QML with courses tailored to your team's needs

The introductory Qt course "Programming with QML for Embedded":

- is designed to take development teams in your organization that are new to Qt or QML for embedded, from the basics to a deep functional understanding of best practices
- equips developers with the skills and know-how to boost their productivity at work
- offers hands-on Qt/QML training with different areas of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Connecting a QML UX with C++ business logic
- Complex list views including data provided from C++ models
- Custom objects implemented using Qt Quick scene graph
- Introduction to the OpenGL pipeline
- Developing for Embedded with Qt Creator
- Performance tuning

Target audience:	Developers targeting embedded platforms
Prerequisite:	Students should have a functional knowledge of C++ but no previous experience with Qt is required. Developers with Qt/Widgets or Qt/QML experience will easily benefit from this course which is regularly updated.
Duration:	3 days

Why learn Qt?

Qt 5 is a modern, mature and highly featured frame work for developing application software that can run across desktop and/or embedded and/or mobile platforms. Qt is often used in combination with other technologies. KDAB's Qt experts are the maintainers (steer the development) of Qt for Android, Qt for QNX and Qt for Windows CE as well as the areas of Qt 3D (OpenGL), Qt Widgets, Qt WebChannel and parts of Qt Core.

Qt is used by tens of thousands of companies across 70+ industries and in all regions of the world. It is available for development with an open source license or with a commercial license.

KDAB can help you determine whether you need licensing as well as help you evaluate whether Qt is the right fit for your project.

Since KDAB was the original Qt training provider and still is the market leader in Qt training, you will always enjoy a tailored training with up to-date materials, hands-on programming exercises and trainers who are active project engineers.

Find out more:

ADVANCED

Qt Courses

QML

Take your team's understanding of QML to the cutting edge

The advanced QML course:

- is designed to take programmers who already know QML to a deeper level functional understanding
- equips you with the cutting edge QML skills and know-how to boost your productivity at work
- offers hands-on training with different areas of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Advanced integration of QML, JavaScript and C++
- Using OpenGL in custom QML elements
- Analysing and profiling the Qt Quick scene graph
- Understanding and removing bottlenecks that affect a QML UI
- Best practices for achieving optimal performance, even on constrained hardware resources.

Target audience:	Experienced QML developers targeting embedded and/or mobile platforms
Prerequisite:	Developers enrolled in this Advanced QML course should already have a functional knowledge of QML. Developers with more than one year of Qt/QML experience will benefit from this course which is regularly updated.
Duration:	3 days

Find out more:

contact us: training@kdab.com

INTRO

Qt Courses

Qt's Model/View framework for widget development

Qt's model/view framework is without doubt one of the hardest parts in Qt to get right. Implementing a list model is relatively easy, but once you turn to implementing tree models you need to handle your own indexes, which we've seen people grow lots of grey hair over, for more than a decade now.

Now add to the equation that you want to filter, sort, or transform these models in many different ways (using proxy models), and that on top of that you need to communicate, say in response to mouse events, from the top of the proxy stack to the model at the bottom etc.

You might also want to paint and handle events differently per row, per column, or even per cell in your view. In that case you will need to implement your own delegates.

In this training we will walk you through all of the above steps. Starting with a simple list model, then turning to tree models where we will discuss strategies for how to go from the world of model indexes to the world of your data and back again.

Following that we will discuss proxy models, and especially, how to pass data through them. Finally we will discuss delegates.

The training will be a mix of presentations and hands-on exercises (of which we aim to do 3-4). You need to have a development environment set up.

As the title suggest, this training is aimed at people developing for Qt Widgets. Should you be developing for QML, then you are of course welcome to join, but between the two environments there is quite a bit of difference. This training would, however, ensure a solid understanding of the framework.

Course Contents

- Model/View Concepts
- List Models
- Table Models
- Tree Models
- Proxy Models
- Delegates
- Supplemental topics:
  - Customizing the views
  - value based models

Target audience: Qt Widgets developers
Prerequisite: Experience with Qt Widgets for Desktop
Duration: 1 day

Contact us for more information:

INTRO

C++ Courses

Modern C++: Introduction

Including best practices for Qt

This training introduces developers to the C++ language, with a focus on how it is used with Qt.

The course is targeted at software engineers with prior knowledge in system or application development, for example experience with software written in C.

Rather than going into all the corners of C++, this training focuses on the parts that are needed when developing with Qt. As an example we will not delve a whole lot on exceptions in C++, as they are seldom used with Qt.

Course Contents

- Introduction to the base language features
- Custom data types such as classes
- Life time and memory management of objects
- Code organisation features such as namespaces
- Runtime polymorphism
- Introduction to generic programming with templates
- Usage of templates in the C++ standard library
- Aspects of functional programming such as lambdas
- Best practices for C++ with Qt

Target audience: Developers who want to get started with C++

Prerequisite: The course is suitable for developers with no prior C++ knowledge.

Duration: 3 days

Why learn C++?

C++ is widely used and has become a de facto standard with extensive libraries. This course provides a base on which to build on for our Advanced C++: What's new in C++11 / C++14 or any of our Introductory Programming with Qt courses.

Find out more:

contact us: training@kdab.com

ADVANCED

C++ Courses

Modern C++: C++11 / C++14 / C++17

Learn the relevant library and language changes

In this hands-on C++11, C++14 and C++17 training for professional C++ developers, you will learn the language changes and the standard library changes introduced in C++11, C++14 as well as changes from C++17.

In class, the new standards will be demonstrated with the aid of many examples, and you will get the opportunity to use them right away in our lab projects. Since not every developer has a C++14 or C++17-capable compiler yet, features only available in C++14 or C++17 are clearly marked as such in the material.

The training also teaches some advanced C++ features like variadic templates and perfect forwarding, but they are introduced gently, and some of them are optional.

Course Contents

Important language changes, including:

- C++11/14: lambdas, range based for loops, strongly typed enums
- C++11/14: constexpr, uniform initialization, move semantics, ...
- C++17: improved lambdas, structured bindings, constexpr if, ...
- C++11/14: Functional programming, including lambda, bind and function objects
- C++11/14: Template meta programming, including variadic templates and perfect forwarding
- C++11/14: Multithreading (including the C++11 memory model, std::thread, std::async, ...)
- C++17: Templates: Fold Expressions, Class template deduction, ...

Target audience:	Professional C++ developers
Prerequisite:	Students are required to have basic C++ knowledge, for example knowing about inheritance and virtual functions.
Duration:	4 days

Why join our Modern C++: C++11 / C++14 / C++17 training?

C++11, released in 2011, is a new major version of the C++ standard and brings many new features to C++ that make the language safer and faster as well as easier and more fun to use. Every professional C++ developer will sooner or later come into contact with C++11/C++14/C++17, and introducing their advantages early can only be beneficial.

Because of all the numerous changes, the C++ creator Bjarne Stroustrup said that "C++11 feels like a new language". C++14 and C++17, released in 2014 and 2017 respectively, bring further incremental improvements.

Find out more:

INTRO

3D / OpenGL Courses

Qt 3D

Integrate 3D content into your applications

In this course you will learn Qt 3D and its use of modern programmable shader-based pipelines which can be applied to both OpenGL and OpenGL ES.

In the Qt 3D training you will learn how to:

- write Qt 3D applications from the ground up
- develop your own materials using Qt 3D and GLSL
- implement your own geometries
- take control of Qt 3D's rendering algorithm using the Frame Graph
- integrate Qt 3D inside a QtQuick application.

In class, our expert trainers will walk you through all necessary topics using a mixture of presentations, examples, and hands-on exercises. By the end of the course you will be armed with all the knowledge you need to become immediately productive with Qt 3D and GLSL, and, in addition, take away over 60 working examples for future reference.

Course Contents

- Overview: Features, Entity Component System
- Drawing: Geometries, Materials and Lights
- User Input: Picking, Keyboard Handling, Logical Devices
- Integration and Helpers: Dynamic Scenes, QtQuick Integration
- Scene Graph: Graphics Pipeline, GLSL, Coordinate Systems, Texturing
- Frame Graph: Viewports, Layers, Selecting Shaders at Runtime, Post-processing Effects
- Advanced Topics: Procedural Texturing, Instanced Rendering
- Animation: Key Frame Animations, Combining Animations
- Advanced Topics: Procedural Texturing, Instanced Rendering, Physically Based Rendering Materials, 2D and 3D Text

Target audience:	Developers who want to conveniently and efficiently integrate Qt 3D
Prerequisite:	This course requires use of Qt 5.7.1 or higher and assumes prior knowledge of Qt Quick. No prior OpenGL knowledge is required.
Duration:	3 days

Why Learn Qt 3D?

Integrating 3D content in applications is becoming a clear trend in many fields and is likely to become even more important in the future with the growth of Virtual Reality and Augmented Reality. Qt has always allowed you to integrate with OpenGL fairly easily, but managing the rendering code itself was still a very challenging task, often limited to a few specialized team members.
Qt 3D provides Qt with advanced 3D features with very flexible APIs available both in C++ and QML.

It allows you to take full advantage of the enormous parallel processing power of today's GPUs and maximizes performance by scheduling work across all CPU cores.

Find out more:

contact us: training@kdab.com

INTRO

3D / OpenGL Courses

Qt 3D Studio

Learn how to integrate 3D content into Qt Quick and QML

This training introduces designers and developers to 3D content workflows and concepts, giving them a solid grounding in creating content using Qt 3D Studio, achieving the visual results they require, and integrating their work with other Qt content such as Qt Quick.

Existing 2D and 3D designers, user-experience designers, as well as developers working on visuals, will all benefit from learning the features and workflow offered by Qt 3D Studio. The training includes a large amount of hands-on time with the software, giving students confidence to begin creating their own content immediately, taking full advantage of the core features available.

Course Contents

- Concepts and workflows for 3D content with Qt 3D Studio
- Presentation structure including scenes, slides and layers
- Understanding animations and materials
- Adding user interactivity and exposing data to QML
- Deploying and evaluating presentations on remote devices and targets

Target audience: Principally designers, or developers who need to get an overview of QML integration

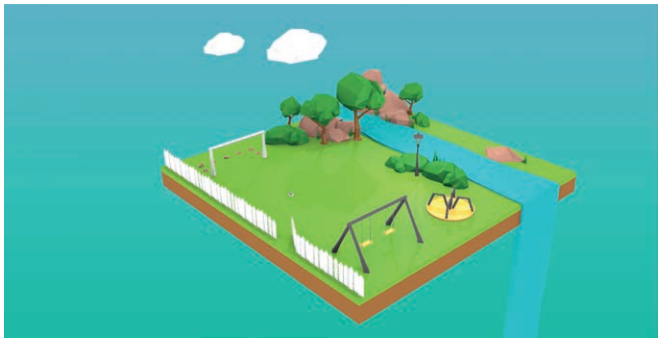
Prerequisite: No coding experience required, some familiarity with 3D concepts would be useful.

Duration: 1 day

Why Learn Qt 3D Studio?

Qt 3D studio gives a tool-led workflow for integrating 3D content into Qt and QtQuick. It provides a simple environment for structuring 3D content, defining animations and behaviors, specifying materials and visual effects, and exporting complete presentations to an optimized runtime component.

While a wide range of workflows are possible when integrating 3D content, this training allows you and your team to focus on the best-practice approaches when working with Qt 3D Studio, to give a robust and well-defined path from 3D content into your final product.



Find out more:

INTRO

3D / OpenGL Courses

Modern OpenGL: Introduction

Skill up on the latest hardware accelerated graphics

This training provides a comprehensive introduction to modern OpenGL development. Beginning with the basic concepts, all the fundamental topics to develop flexible, high performance OpenGL code that run on the desktop and embedded / mobile devices will be covered. Key techniques including lighting, texturing, framebuffer objects and transformations are introduced, in a format suitable for any developer working in C or C++.

Course Contents

- Basic OpenGL window and context management
- The modern OpenGL pipeline
- Rendering geometry with GLSL shaders and vertex buffer objects
- Transformations and coordinate systems, projection and camera handling
- Fundamentals of lighting including the standard Phong model
- Working with frame-buffer-objects (FBOs)

Target audience:	Developers who want to start working with hardware-accelerated graphics, or update their OpenGL knowledge based on current best practices
Prerequisite:	This course is suitable for developers with no prior graphics experience, but familiarity with C and/or C++ is necessary.
Duration:	3 days

Why learn modern OpenGL?

OpenGL is the industry standard for high performance graphics and computation on desktop, mobile and embedded platforms. Whether the requirement is visualizing large data sets, creating engaging user interfaces or breathtaking real-time visuals, getting the topics covered in this course is essential.

Find out more:

contact us: training@kdab.com

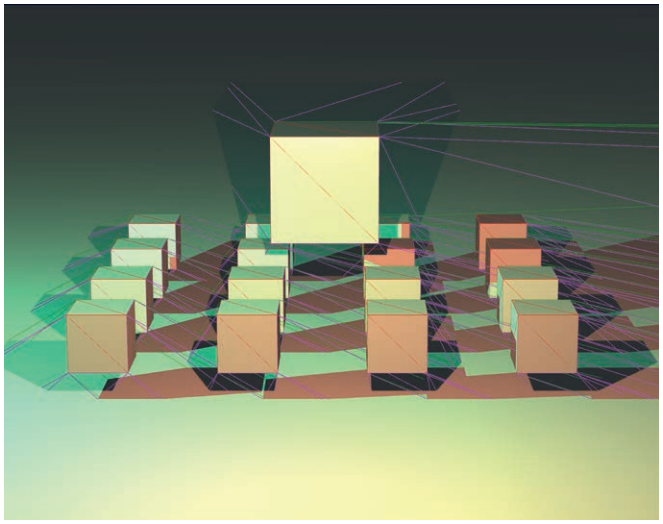
ADVANCED

3D / OpenGL Courses

Modern OpenGL: Rendering and Effects

Learn to bring advanced visuals into your project

This training explores the implementation of many different rendering techniques to achieve cutting-edge visuals in OpenGL applications. Techniques are explored in depth with many examples, analysis of shader code and implementation details.



Course Contents

- Normal mapping and parallax mapping
- Environment mapping
- Procedural texturing
- High dynamic range (HDR) rendering
- Physically based rendering (PBR)
- Deferred rendering
- Shadowing via shadow maps and shadow volumes
- Screen-space ambient occlusion
- Image-processing filters and effects: blur, noise
- Anti-aliasing techniques
- Dealing with transparency
- Picking

Target audience:	Developers wanting to understand and incorporate realistic advanced visuals into their projects, for realism, visualization or artistic effect
Prerequisite:	Developers already working with OpenGL, comfortable with the basics of specifying geometry, writing basic shaders and working with image data
Duration:	3 days

Why learn advanced rendering?

Hundreds of papers, textbooks and examples of different rendering techniques exist. In this course we explore some of the most widely used and generally applicable, to give students the confidence to create their own versions and gain a deep understanding of GLSL.

Find out more:

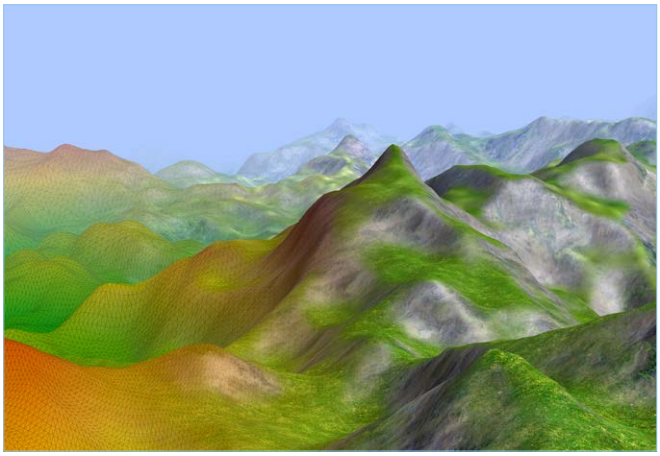
ADVANCED

3D / OpenGL Courses

Modern OpenGL: Pipeline and Performance

Get the maximum out of your hardware

This training explores strategies to increase the performance of new and existing OpenGL code, with multi-pass rendering and use of uniform buffers, shader storage buffers and indirect drawing to reduce driver overhead.



Course Contents

- Cost of state changes, batching and sorting the modern OpenGL pipeline
- Culling, occlusion queries, spatial data-structures
- Debugging and profiling OpenGL via extensions, timers and tools
- Synchronization and timer queries
- Buffering and streaming strategies for large data sets
- Instanced rendering and multi-draw indirect
- Uniform and shader storage buffer objects
- Shader subroutines
- The OpenGL memory model and image load/store
- Geometry and tessellation shaders
- Transform Feedback and Compute shaders

Target audience:	Developers wanting to create or improve existing rendering code, using every technique at their disposal to understand and maximise performance, and extract the full potential from their hardware
Prerequisite:	Developers already working with OpenGL, comfortable with the basics of specifying geometry, writing basic shaders and working with image data
Duration:	3 days

Why learn about Advanced Pipeline and Performance?

Getting the best from available hardware resources, especially on constrained systems, means deeply understanding the costs of different graphics operations, and how to optimise the rendering architecture to meet visual requirements. This course teaches how to increase performance effectively.

Find out more:

contact us: training@kdab.com

INTRO

Qt Auto Courses

Qt Automotive Suite

Learn the benefits of the comprehensive package

This training introduces you to the Qt Automotive Suite, Qt's specialized variant for automotive in-vehicle infotainment (IVI) systems.

The course is targeted to Qt developers who want to learn about the additional runtime components and development tools provided by the Qt Automotive Suite, as well as how to best employ these components in their IVI project.

Course Contents

- Device setup and target development with Qt Creator
- Device image creation and customization
- Multi-process application management with Qt ApplicationManager and Wayland
- Qt Virtual Keyboard
- Creating and integrating SCXML-based state machines
- Interfacing with CAN buses using Qt SerialBus
- Developing vehicle data and middleware access APIs using Qt IVI
- Integrating 2D and 3D UI content
- GammaRay runtime introspection

Target audience:	Developers needing to get up to speed with the Qt Automotive Suite
Prerequisite:	Basic Qt or QML knowledge
Duration:	3 days

Why learn Qt Automotive Suite?

Qt Automotive Suite extends the regular Qt frameworks and tools into a comprehensive package tailored to the needs of automotive HMI projects. Additional components support you with interfacing with middleware services and with application management. Extensions to QtCreator ease on-target development and powerful runtime inspection tooling gives you new insights at runtime. In this course you will learn about the additional APIs and components provided by the Qt Automotive Suite, as well as how to effectively employ the development and diagnostic tools included.

Find out more:

INTRO

Qt Auto Courses

Qt for Automotive Development

Design your IVI architecture with Qt

This training covers techniques for designing and implementing large-scale Qt Quick based user interfaces, as they are commonly found in automotive in-vehicle infotainment systems.

This course is targeted at Qt developers faced with having to architect a complex HMI system that will hold up to changing feature requirements and challenging performance goals.

A particular focus is put on:

- scalability
- start-up performance
- maintainability

Course Contents

- Multi-page application architectures
- Single- or multi-process system architectures
- Input, event and focus handling
- Seamless integration of 2D and 3D UI content
- Application infrastructure such as styling/theming, popup management, localization and LTR/RTL vs LHD/RHD layouting
- Startup and runtime performance considerations, profiling and optimization techniques

Target audience:	Developers working on IVI systems using Qt
Prerequisite:	Basic Qt or QML knowledge
Duration:	3 days

Why learn about Qt for Automotive Development?

In this course you will learn how to apply Qt effectively when designing the architecture of an IVI system, and what pitfalls to avoid. In-vehicle infotainment projects tend to be particularly complex. Not only do they have a huge feature scope, but at the same time high requirements for startup and runtime performance.

They also need to support a wide range of international markets and a highly variable product line, all while keeping the codebase maintainable for years to come. Qt provides you with the comprehensive toolbox needed to achieve all of this.

Find out more:

contact us: training@kdab.com

INTRO

Tools

Debugging & Profiling Qt applications on Windows

Get up-to-speed with the latest Windows Qt debugging and performance tools

This training gives an introduction to various tools which help developers and testers find bugs and performance issues. The tools presented cover a wide range of problems, from general purpose debugging and CPU profiling to Windows specific high-level analyzers. Often, it is relatively simple to run a tool, but interpreting the results, or even just using some of the more advanced tools, requires deep technical knowledge.

Course Contents

This training focuses on tooling available to developers working with Microsoft Windows on the desktop.

The following tools will be covered:

Debugging

- general purpose debuggers: Visual Studio, CDB
- memory error detectors: Application Verifier, AddressSanitizers
- thread error detectors: ThreadSanitizer
- tracing: Dependency Walker, Process Monitor, API Monitor
- various Qt-builtin features
- GammaRay to investigate internals of Qt applications
- OpenGL: apitrace

Profiling

- CPU: Intel VTune Amplifier XE, Visual Studio, Windows Performance Analyzer
- heap memory: Visual Studio, Windows Performance Analyzer
- QML: QML profiler
- OpenGL: apitrace, NVidia nSight

Testing

- Qt TestLib: unit tests and benchmarks
- static code analysis: clang analyzer, Coverity, Clazy
- code coverage: gcov

Target audience: Developers who want to find and fix problems
Prerequisite: Knowing the basics of C++ and Qt
Duration: 3 days

Why learn about Debugging and Profiling Qt?

The time spent writing code is often dwarfed by the time required to find bugs and improve performance. This training makes your development workflow more efficient. You will learn what tool to use in which situation, how to set it up and run it on an application and, finally, you will learn how to analyze and interpret the results obtained from the various tools.

Find out more:

INTRO

Tools

Debugging & Profiling Qt applications on Linux

Get up-to-speed with the latest Linux Qt debugging and performance tools

This training gives an introduction to various tools which help developers and testers find bugs and performance issues. The tools presented cover a wide range of problems, from general purpose debugging and CPU profiling to Linux specific high-level analyzers. It is often relatively simple to run a tool, but interpreting the results, or even just using some of the more advanced tools, requires deep technical knowledge.

Course Contents

This training focuses on what a developer should know to be efficient on both desktop as well as embedded Linux.

The following tools will be covered:

Debugging on Linux

- general purpose debuggers: GDB, RR
- memory error detectors: valgrind's memcheck, AddressSanitizers
- thread error detectors: ThreadSanitizer
- tracing: ldd, strace
- various Qt-builtin features
- GammaRay to investigate internals of Qt applications
- OpenGL: apitrace

Profiling on Linux

- CPU: valgrind's callgrind, Linux perf, Intel VTune Amplifier XE
- heap memory: valgrind's massif, heaptrack
- QML: QML profiler
- OpenGL: apitrace
- System-wide: LTTng

Testing on Windows and Linux

- Qt TestLib: unit tests and benchmarks
- static code analysis: clang analyzer, Coverity, Clazy
- code coverage: gcov

Target audience:	Developers who want to find and fix problems
Prerequisite:	Knowing the basics of C++ and Qt
Duration:	3 days

Why learn about Debugging and Profiling Qt?

The time spent writing code is often dwarfed by the time required to find bugs and improve performance. This training makes your development workflow more efficient. You will learn what tool to use in which situation, how to set it up and run it on an application and, finally, you will learn how to analyze and interpret the results obtained from the various tools.

Find out more:

contact us: training@kdab.com

INTRO

Tools

Debugging & Profiling C++ applications on Windows

Get up-to-speed with the latest Windows C++ debugging and performance tools

This training gives an introduction to various tools which help developers and testers find bugs and performance issues. The tools presented cover a wide range of problems, from general purpose debugging and CPU profiling to Windows specific high-level analyzers. It is often relatively simple to run a tool, but interpreting the results, or even just using some of the more advanced tools, requires deep technical knowledge.

Course Contents

This training focuses on tooling available to developers working with Microsoft Windows on the desktop.

The following tools will be covered:

Debugging

- general purpose debuggers: Visual Studio, CDB
- memory error detectors: Application Verifier, AddressSanitizers
- thread error detectors: ThreadSanitizer
- tracing: Dependency Walker, Process Monitor, API Monitor
- OpenGL: apitrace

Profiling

- CPU: Intel VTune Amplifier XE, Visual Studio, Windows Performance Analyzer
- heap memory: Visual Studio, Windows Performance Analyzer
- OpenGL: apitrace, NVidia nSight

Testing

- static code analysis: clang analyzer, Coverity
- code coverage: gcov

Target audience:	Developers who want to find and fix problems
Prerequisite:	Knowing the basics of C++
Duration:	3 days

Why learn about Debugging and Profiling C++?

The time spent writing code is often dwarfed by the time required to find bugs and improve performance. This training makes your development workflow more efficient: You will learn which tool to use in which situation, how to set it up and run it on an application and, finally, you will learn how to analyze and interpret the results obtained from the various tools.

Find out more:

INTRO

Tools

Debugging & Profiling C++ applications on Linux

Get up-to-speed with the latest Linux C++ debugging and performance tools

This training gives an introduction to various tools which help developers and testers find bugs and performance issues. The tools presented cover a wide range of problems, from general purpose debugging and CPU profiling to Linux specific high-level analyzers. It is often relatively simple to run a tool, but interpreting the results, or even just using some of the more advanced tools, requires deep technical knowledge.

Course Contents

This training focuses on what a developer should know to be efficient on both desktop as well as embedded Linux.

The following tools will be covered:

Debugging

- general purpose debuggers: GDB, RR
- memory error detectors: valgrind's memcheck, AddressSanitizers
- thread error detectors: ThreadSanitizer
- tracing: ldd, strace
- OpenGL: apitrace

Profiling

- CPU: valgrind's callgrind, Linux perf, Intel VTune Amplifier XE
- heap memory: valgrind's massif, heaptrack
- OpenGL: apitrace
- System-wide: LTTng

Testing

- static code analysis: clang analyzer, Coverity, Clazy
- code coverage: gcov

Target audience:	Developers who want to find and fix problems
Prerequisite:	Knowing the basics of C++
Duration:	3 days

Why learn about Debugging and Profiling C++?

The time spent writing code is often dwarfed by the time required to find bugs and improve performance. This training makes your development workflow more efficient: You will learn which tool to use in which situation, how to set it up and run it on an application and, finally, you will learn how to analyze and interpret the results obtained from the various tools.

Find out more:

contact us: training@kdab.com

INTRO

Tools

CMake

The best thing a build system can do is not get in the way

CMake is the de facto standard build system for C and C++ outside of frameworks that require their own. It has earned this place by supporting the situations and special cases that arise in real projects.

Support for CMake within Qt is being significantly improved and there are longer term plans to switch to CMake for building Qt itself. It's currently a good alternative if you hit limitations in qmake.

This course will teach the basics of creating and building projects with CMake. In recent years, CMake has introduced some cleaner and more precise constructs. The course will focus on the new constructs where possible.

Course Contents

- Build system overview; targets and dependencies
- Building executables and libraries
- CMake language and debugging
- Platform-independence
- Using and writing package finders
- Code generators
- Cross compilation

Target audience: C and C++ Developers
Prerequisite: Experience with build systems
Duration: 2 days

Why learn CMake?

CMake has broad functionality that covers many real world problems. Learning CMake enables you to solve advanced build requirements. This includes cross-platform builds, feature detection based on platform or available libraries, built-time configurable feature switches and custom build steps.

Find out more:

INTRO

Tools

Testing Qt with Squish

Learn how to test your Qt application with Squish

What is the most important thing about a tool for automatic testing? Often the answer is not that it will reliably test your application, but that it will reliably test your application tomorrow even when you have adapted the application slightly!

The aim of this training is that you will learn to use Squish for testing your application, over and over again – much more effortlessly than before. Techniques you will learn include recording a script with Squish and then adapting this into a piece of reusable code that is much less likely to break with the next version of the application it tests.

Testing Qt with Squish – Course Contents

- Recording your first script, verifying the result
- Getting stable scripts by programming
- Optional, Python introduction and/or Java script introduction
- Refactoring your scripts
- Object identification, data driven testing, Qt event handling, file access
- Customized object identification, automatic test runs, special purpose Squish

Target audience:	Testers or programmers to support testers
Prerequisite:	Limited programming experience. (You know the basics of programming, but you don't necessarily have much experience)
Duration:	3 days

Why learn about Squish GUI Tester?

Manual testing of user interfaces in applications is often a very complex and error-prone activity. Squish is a proven GUI test automation tool for functional GUI regression tests. Companies in all types of industries, including KDAB, use Squish to reduce the time spent on GUI testing software releases while increasing the quality of their applications.

Find out more:

contact us: training@kdab.com

INTRO

Tools

User Centered Development and Usability

It needs more than excellent code

In this course you will learn how to determine user requirements so you can provide solutions for your product's interaction challenges that are both mentally satisfying and visually appealing.

You will also learn how to uncover and estimate the effects of usability deficits and establish benchmarks so as to steer the GUI development for an optimum result.

With this introductory training you will not only understand the theory, but also take away tools and strategies you can directly introduce to your product development team.

In order to tighten your newly gained knowledge, extensive exercises accompany the theory: We will develop an application from the first idea to user testing the final design.

Some of the topics covered

- ISO 9241-210 or "what is a User Centered Development Process?"
- Assess users' needs and make them actionable (e.g. Personas)
- How to communicate user requirements in your team
- Designing for the mind: Translating user needs into interfaces
- Tools and strategies for rapid UI prototyping
- Testing the Usability: Guerilla or The Lab?
- Understand the why: Psychological background
- How to talk with users - getting the feedback right
- Interweaving UX processes with general development

Target audience:	Developers, Development Leads, Product Managers, Managers, Decision Makers wanting to learn how to involve users to boost a product's success
Prerequisite:	Being part of a product development team
Duration:	3 days

Why learn about User Centered Development & Usability

Users ultimately decide about the success of your product – at the latest when they are supposed to use it. With this training we provide you with strategies for turning your users into allies early in the development process. These strategies will not only lead to better products and higher customer satisfaction, but will also help you to improve the development process itself.

Find out more:

ADVANCED

Tools
In-Depth Multithreading

This three-day training teaches multithreading application development techniques, using the Qt/C++ object technology. Participants will gain insight into multithreading problems in general, and how they pertain to Qt programs in particular.

Examples include how to offload work from the GUI thread, how to increase throughput to the maximum in your application, and optimal strategies for communication between threads.

The training covers topics such as cross-thread signal/slot connections, QThreadPool, QObjects and multithreading, QtConcurrent, QFuture, and Qt atomic operations.

Participants are expected to have a working knowledge of C++ and Qt. Prior experience with multithreaded programming is recommended, but not necessary.

The Multithreading with Qt course:

- is designed to take programmers who are new to Qt multithreading from the basics to a deep functional understanding of the best practices
- offers hands-on multithreading training with different kinds of focus, depending on your needs
- is delivered by authorized KDAB trainers with real-life practical experience

Course Contents

- Multithreading Concepts
- Synchronization Primitives
- Multithreading Foundation
- QtConcurrent
- The C++11 Memory Model
- Atomic Operations
- Relation to Model/View
- Qt and the Standard Library threading facilities

Target audience:	Qt and C++ developers.
Prerequisite:	Professional experience of object-oriented programming is recommended, as well as experience in C++. A working understanding of Qt programming is also recommended.
Duration:	3 days

Find out more:

contact us: training@kdab.com

Our Trainers at KDAB

When not conducting training sessions, all KDAB trainers are actively working on client projects, whether developing, consulting or mentoring.

Albert Astals Cid

Software Engineer at KDAB, Albert has been using Qt since 2002. Since then, he has applied Qt in a broad range of industries ranging from transit simulation, medical devices to games and many more. His main expertise is in C++ and Qt/QML. In 2005, Albert won KDE's Akademy Award for his work on improving PDF rendering on Free Software platforms. Albert holds a MSc in Computer Engineering. *Languages: Spanish, English*

Jim Albamont

Senior Software Engineer at KDAB, Jim has actively developed with Qt since 2001. He has a background in computer graphics and data visualization, including 6 years as the lead developer of a 3D visualization tool for financial data built with Qt and OpenGL. He has held Qt training classes throughout the US where he is based. Jim holds an MSc in Computer Science. *Language: English*

Nicolas Arnaud-Cormos

Senior Software Engineer and team lead at KDAB, Nicolas has actively developed with Qt since 2001 and is a founding member of Qtfr, the French Qt community site. He has worked on multiple Qt widgets or QML projects, with a particular emphasis on API design and software architecture. He has held Qt trainings for companies such as Michelin, Ford and ST-Ericsson. Nicolas holds an MSc in Computer Science. *Languages: French, English*

Franck Arrecot

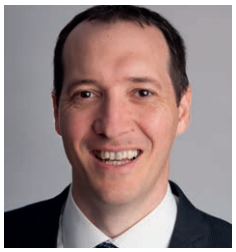
Software Engineer at KDAB, Franck has actively developed with Qt since 2011 when he started contributing to open source projects. He has been an active KDE contributor to Zanshin: KDE task management software and more recently to the Qt3D module. He holds a Masters degree in Computer Science. *Languages: French, English*

Björn Balazs

Senior Experience Engineer at KDAB, Björn has specialized in Usability and User Experience since 1999 and holds a Diploma in Psychology. A longtime member of the KDE visual design group, Björn has worked for many free software projects. For the past ten years, he has regularly given trainings on user-centric development. *Languages: German, English*

Giuseppe D'Angelo

Senior Software Engineer at KDAB, Giuseppe is a long time contributor to Qt, having used Qt and C++ since 2000, and is an Approver in the Qt Project. His contributions in Qt range from containers and regular expressions to GUI, Widgets and OpenGL. He is a UNIX specialist and passionate about free software. Before joining KDAB, he organized conferences on open-source around Italy. He holds a BSc in Computer Science. *Languages: Italian, English*



David Faure

Senior Software Engineer who is also the head of KDAB's French office, David is a Qt user since its beginning. He has made numerous contributions to Qt, including new classes for QtCore in Qt 5. David is well known in the KDE project for his work on the web browser and especially on KDE Frameworks. He has taught Qt development at numerous conferences and to companies such as Michelin, Schlumberger and Orange. He has become a specialist in multithreading with Qt, as well as in performance optimizations. David holds an MSc in Computer Science. *Languages: French, English*

Kevin Funk

Kevin has actively developed with Qt/C++ since 2006 and has a special interest in tooling and profiling. He's an active contributor to KDAB's GammaRay analyzer (a high-level Qt application debugger) and has a strong emphasis on state machine tooling. He is co-maintainer of the KDevelop IDE, a powerful C/C++ developmen-t environment backed by Clang, and is pushing for cross-platform success inside KDE. Kevin holds a Masters Degree in Computer Science. *Languages: German, English*

Steffen Hansen

Senior Software Engineer and team lead at KDAB, Steffen has actively developed with Qt since 1997 and was responsible for several of the components of the KDE Desktop such as the display manager kdm and the service manager kded. He has taught numerous Qt classes for companies such as ChevronTexaco, BBC and J.D. Edwards. Steffen holds an MSc in Computer Science. *Languages: Danish, English*

Sean Harmer

Dr. Sean Harmer is a Senior Software Engineer at KDAB where he is the head of our UK office and also leads the 3D R&D team. He has been developing with C++ and Qt since 1998 and is Qt 3D Maintainer and lead developer in the Qt Project. Sean has broad experience and a keen interest in scientific visualization and animation in OpenGL and Qt. He holds a PhD in Astrophysics along with a Masters in Mathematics and Astrophysics. *Language: English*

Tobias Koenig

Senior Software Engineer at KDAB, Tobias has actively developed with Qt since 2001 and has been an active KDE contributor during this time. His contributions have been mainly to the KDE PIM project and the KDE libraries, but also to other open source projects. He holds an MSc in Computer Science. *Languages: German, English*

Kevin Krammer

Senior Software Engineer and team lead at KDAB, Kevin has actively developed with Qt and contributed consistently to KDE since 2000. He is a founding member of the QtCentre website and has mentored at Google's Summer of Code program for 10 years. One of KDAB's most experienced trainers, Kevin keeps our training material up-to-date and has trained engineers from Blackberry, Lockheed Martin, Siemens and Safegate and many others. Kevin holds a BSc in Software and Communications Engineering. *Languages: German, English*

Volker Krause

Senior Software Engineer at KDAB, Volker coordinates KDAB's activities in the automotive space. He is a long-term contributor to KDE and Qt and has been developing with Qt since 2002. Volker's experience spans desktop and embedded platforms, with a special focus on development tooling. In 2010 he started the GammaRay project to address the need for QML debugging tools, and has led the project since then. Volker holds an MSc in Computer Science. *Languages: German, English*

Anton Kreuzkamp

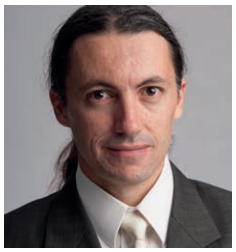
Software engineer at KDAB, Anton began using C++ and Qt in 2009 by contributing to the KDE project. He is one of the active contributors to KDAB's GammaRay tool (a high-level Qt application debugger) mainly focusing on QML and QtQuick introspection. Besides active development in large-scale C++ applications at KDAB, he closely follows the ISO standardization process for the C++ language, providing up-to-date, in-depth insight into the reasoning that leads the language development. *Languages: English, German*

Mike Krus

Dr. Mike Krus is a Senior Software Engineer at KDAB. He has been developing with C++ since 1996 and Qt since 2004. He has a broad range of experience in scientific applications, mainly in civil engineering and oil & gas industries. His range of expertise includes C++, QML and interactive 3D visualization software design on desktop and mobile as well as MacOS development. Mike is the Qt maintainer for the tvOS platform and is very interested in building mobile applications with Qt, mainly on iOS. He has a PhD in Computer Science. *Languages: English, French*

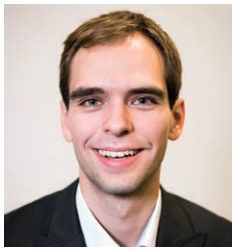
Paul Lemire

Senior Software Engineer at KDAB, Paul is a Qt approver and active contributor to the Qt 3D module where he is one of the main developers. He has been developing Qt based C++ and QML applications since 2010 and has particular interest and expertise in OpenGL and GPU assisted computing. Paul holds a Master's degree in Computer Science.
Languages: French, English



András Mantia

Senior Software Engineer at KDAB, András has actively developed with Qt since 2002. He is a core developer of KDE's web development environment Quanta Plus and contributor to other parts of KDE. András speaks regularly at free software events about Qt-based products and has held trainings for companies such as Accenture, TietoEnator and Nokia.
Languages: Hungarian, Romanian, English



Jan Marker

Software Engineer at KDAB, Jan has been using Qt since 2009 when he started contributing to the KDE project. Since joining KDAB he has worked on multiple large Qt and QML projects, while more recently also developing Wayland compositors. Besides in-depth knowledge of Qt and C++, Jan also has a deep interest in other technologies like NodeJS. He holds a BSc in Computer Science. *Languages: German, English*

Marc Mutz

Marc is a Senior Software Engineer with KDAB and author of the "Effective Qt" series of articles. He originated KDAB's "In-depth Multithreading With Qt" and C++11 courses, and runs "-Wmarc", a blog about Qt, C++ and Boost. A regular contributor to Qt and maintainer of the QtWidgets module, he has actively used the framework for more than a decade, first as a KDE contributor, and then on the job. Marc is a sought-after speaker at conferences on Qt and C++ topics and has an MSc in Theoretical Physics. *Languages: German, English, Norwegian*

Tobias Nätterlund

Senior Software Engineer and team lead at KDAB, Tobias has actively developed with Qt since 2006. He is the main author of the Squish training material and has been involved in numerous projects and workshops helping customers get started with Squish, as well as integrating Squish with existing test frameworks or user applications. Tobias has held Squish training courses in the United States, Finland, Germany and Sweden. He holds an MSc in Computer Science. *Languages: Swedish, English*

Jesper K. Pedersen

Senior Software Engineer at KDAB, Jesper has actively developed with Qt since 1998 and initiated numerous KDE programs and components. He is the main author of the course material we use for the Qt courses and has taught more than 70 Qt classes for companies such as Boeing, IBM and Veritas. He holds an MSc in Computer Science.
Languages: Danish, English

Nuno Pinheiro

Senior UX/UI Designer at KDAB, Nuno did the first QML training for designers and actively uses the QML language for fast UX/UI prototyping and UI solutions deployment. His works include general illustrations, UI design, corporate design, interactive mock-ups, animation examples and much more. Known for his contribution to the award winning Oxygen Project where he is the current coordinator, his computer art is used on KDE computer platforms worldwide. Nuno has an MSc in Civil Engineering. *Languages: Portuguese, English.*

Nate Rogers

Senior Software Engineer at KDAB, Nate has been developing software with C/C++ since 2005 and has a special interest in QML and embedded systems. He has experience with all aspects of developing Qt/QML applications from concept to first customer shipment. Nate is an active contributor to KDAB's Hotspot, the Linux perf GUI for performance analysis. Nate holds a BS in Software and Information Systems. *Language: English*

André Somers

Software Engineer at KDAB, André has been using Qt since 2002 when he wrote an open source Qt 2-based flight computer for use in glider planes. Since then, he has applied Qt in a broad range of industries ranging from oil & gas exploration, scientific data manipulation and visualization to medical devices and many more. His main expertise is in C++ and QML. André holds a MSc in Philosophy of Science & Technology.
Languages: Dutch, English

James Turner

Senior Software Engineer and team lead at KDAB, James has been developing with Qt since 2002. He contributes to the current maintenance of Mac platform support as well as the development of OpenGL and 3D support in Qt. James has a background in user-interface, graphics and simulation development as well as a long history of development on OS-X and prior versions of Mac OS. He is a lead developer on FlightGear, the open-source flight simulator, and holds a BSc in Computer Science. *Language: English*

Bogdan Vatra

Senior Software Engineer at KDAB, BogDan has been using Qt since 2002 and is the original author of the Qt for Android port. He is a regular contributor to Qt and currently the maintainer of the Qt Android platform and QtCreator Android plugin. Bogdan also actively updates Android users with his series of blog posts about Qt on Android.
Languages: Romanian, English

Milian Wolff

Senior Software Engineer at KDAB, Milian leads the R&D in tooling and profiling in which he has a special interest. Milian created Massif-Visualizer and heaptrack, both of which are now used regularly to improve the performance of C++ and Qt applications. When not applying his knowledge to improving code base performance for KDAB's customers, Milian maintains QtWebChannel for the Qt Project and is co-maintainer of the KDevelop IDE. In 2015, Milian won KDE's Akademy Award for his work on Clang integration. He has a Masters Degree in Physics.
Languages: German, English

Testimonials for KDAB Training

Read what our customers say

Well organized, well-paced, the exercises are very interesting. It was both enjoyable and felt thorough on the topics at hand. Very much recommended!

Cédric Dourneau, SISW, France

One of the best run and presented programming training courses I have participated in. The coverage of modern OpenGL techniques and API usage was comprehensive, and the examples and labs demonstrated how theory could be applied in practice. I would highly recommend this course to anyone working in the scientific computing and data visualisation fields.

Ben Fletcher, Senior Software Engineer, Defence Science & Technology Organisation (DSTO), Department of Defence, Adelaide, Australia

The exercises were well designed for all levels of Qt programming experience providing additional challenges for more advanced users. The instructor had a good balance of real programming experience and training expertise. His energy and enthusiasm kept the course interesting and enjoyable.

Jennifer Patton, Software Engineer, Pelco, Fresno, Ca, USA

I found the course to be excellent. It provided a good grounding in the basics of writing programs with Qt, with plenty of useful and realistic examples. The trainer provided many anecdotes and examples of real-world experience of using Qt. I would sincerely recommend the training course to everyone who has an interest in the Qt tool-kit and/or mixed system programming.

Conor O'Neill, Radio Planning, Logica, Bristol, UK

Read more testimonials:

About KDAB:

The original Qt services company and a leading Qt contributor.

Successfully delivering hundreds of C++, OpenGL and Qt projects. Thousands of developers trained since 1999.

German office
+49 (0)30 5213 254 70
info-de@kdab.com

French office
+33 (0)4 90 84 08 53
info-fr@kdab.com

UK office
+44 (0)1625 809908
info-uk@kdab.com

Swedish office
+46 (0)563 540090
info@kdab.com

US office
+1.866.777.5322
info-us@kdab.com

We can help!
Contact us about your project.
www.kdab.com

V:21