Comprehensive Qt and QML Development: From Basics to Intermediate

Prerequisites

1. Basic programming knowledge, preferably in C++ or JavaScript.
2. Familiarity with object-oriented programming concepts and basic software development practices.

Day 1: Installation, Setup, and Introduction to Qt

Module 1: How to Install Qt

- Setting up the Qt development environment

Module 2: Getting Started With Qt Creator 14

- Launch Qt Creator IDE and explore its basic views
- Create a new project to try out basic functionalities

Module 3: Introduction to Qt Widgets

- Basics of Qt Widgets
- Creating and managing basic widgets

Day 2: Deep Dive into Qt Widgets and Basic QML

Module 4: Advanced Qt Widgets Concepts:

- Advanced widget concepts
- Customizing widgets and layouts
- Detailed use of Qt Widgets
- Implementing more complex user interfaces

Module 5: Introduction to QML

- What is QML?
- Why would you use QML?
- The QML syntax.
- The key QML concepts.
- How do you structure QML?
- How do you compose QML UIs?

Module 6: QML Integration Basics

What are the roles of QML and C++ in Qt applications?

- What are the benefits of combining QML and C++ in Qt applications?
- How do you register and use C+ + classes so that they are usable within QML?

Day 3: Advanced QML and Qt Concepts

Module 7: Introduction to Qt Quick Controls

- What are Qt Quick Controls?
- What properties are shared between Controls?
- How can you define the layout of elements within your applications?
- How do you use the basic Qt Quick Controls style?
- How can you simply style your components?
- How do you define your QML types with a QML file?

Module 8: Providing Models from C++ to QML

- What is the Model/View Programming in Qt/QML?
- What Models and Views Qt offers?
- How to create custom C++ models?
- How to access C++ Models from the QML?
- What are Proxy models?

Module 9: Qt Quick 3D: Views, Scenes & Nodes

- What is the Model/View Programming in Qt/QML?
- What Models and Views Qt offers?
- How to create custom C++ models?
- How to access C++ Models from the QML?
- What are Proxy models?

Module 10: Introduction to Qt Quick 3D: Custom Materials, Render Settings & Post-Processing

- What are custom materials, and how do you write them?

- What are some of the different settings that can be used to control the global rendering environment?
- What are post-processing effects?
- How can you write your own post-processing effects?

Day 4: UI Design,Testing and Advanced Integration

Module 11: Getting Started with Qt Design Studio

- Learn what Design Studio is and why it is a powerful tool supporting the collaboration of designers and developers
- Launch Qt Design Studio for the first time
- Go through its basic views
- Create a new project that you can use to try out some of the basic functionalities

Module 12: UI Design with Qt Design Studio

- Advanced UI design techniques
- Implementing complex designs using Qt Design Studio

Module 13: Creating a Simple Qt Quick Application

- Building and deploying a simple Qt Quick application
- Testing and debugging

Module 14: How to Expose C++ to QML

- Recognise why mixing C++ and QML is beneficial.
- Understand what registering is and how it is done.
- Learn how to use a custom QML type as a property type.

Module 15: Automated Testing with Squish

- How to manage Applications Under Test in the Squish IDE.
- How to create new tests in the Squish IDE.
- How to use verification points to validate GUI objects.

Day 5: Final Project and Review

Module 16: Building with CMake: Getting Started with CMake and Qt

- Learn what CMake is and how it is used in application development with Qt.

Module 17: Let's Get Thready! Multithreading with Qt

- Understand the concept of multithreading and how it can be used to improve the performance of applications.
- Learn how to create threads in a Qt application.
- Learn how to manage threads in a Qt application.
- Understand how to use a thread pool to manage threads for background computing.

Module 18: Final Project

- Creating Your First App with Qt Design Studio
- Combining all learned concepts into a comprehensive project
- Designing and developing a functional Qt and QML application
- Review and presentation of the final project