qmob
SOLUTIONS

TRAINING, CONSULTING
& DEVELOPMENT SERVICES

2022

// C++ AND Qt **SPECIAL SERVICES**

https://qmob.solutions

EDITORIAL

Software Development with C++ and Qt

Why C++?

The C++ language (http://isocpp.org) is currently used by millions of developers around the world and the decision to use this language in many projects is due to several factors: availability of high-level abstractions with excellent performance, access to low-level when needed, cross-domain applicability, high portability, zero overhead (no additional penalties for not using advanced language features), excellent resource management, and industry dominance. With the modernization of the programming language in the C++11/14/17/20 standards, its adoption is considerably expanded, mainly in the areas of mobile and cloud computing.

Why Qt?

Qt (http://www.qt.io) is a cross-platform software development toolkit adopted by more than 70 industries worldwide, in desktop, mobile, embedded systems and IoT (Internet of Things) areas. Qt allows the development of systems for different platforms such as Windows, GNU/Linux, OS X, Android and iOS with a single source code, has extensive documentation, high productivity and excellent performance even on the most modest platforms.

ABOUT QMOB SOLUTIONS

Qmob Solutions (http://qmob.solutions) is a company specialized in training, consulting and development services based on C++ and Qt technologies. Created by C++ and Qt specialists with more than 20 years of experience, Qmob Solutions values the offer of training sessions adapted to the customer's needs and the development of correct and efficient solutions from a Software Engineering point of view, reducing maintenance costs and meeting the most important non-functional requirements of the project. Our focus is on delivering elegant and functional solutions for multiple platforms: Windows, Linux, OS X, iOS, Android and embedded platforms, building on top of the many benefits that C++ and Qt have in maintaining a single codebase for multiple platforms. Qmob Solutions engineers are also highly experienced in the definition and implementation of mature development processes, QA (Quality Assurance) and CI/CD (Continuous Integration/Continuous Delivery), combining agility and process/product quality.

As presented at https://www.qt.io/contact-us/partners, Qmob Solutions is the first and only official The Qt Company's Service Partner (the company that currently develops Qt) in Brazil and Latin-America. As a result, Qmob Solutions further strengthens its relationship with the global Qt community, ensuring always up-to-date training, effective and productive consulting, as well as access to the latest Qt news.

Qt The Qt Company

Service Partner

TABLE OF CONTENTS

Training Services

Consulting Services

Development Services

FUNDAMENTALS OF C++

Overview and Goals

The C++ language is a powerful tool for building flexible, integrable systems with excellent performance. Its abstraction mechanisms and high degree of portability make C++ the language of choice for systems development in areas such as manufacturing, high performance, cloud computing, and mobile applications.

In this training, we present the basics of C++ and how the main features from Object Orientation are used in it. At the end of the training, the student should be able to correctly and systematically apply the main language resources; building systems that are easy to maintain, with high performance and capable of running on multiple platforms.

Prerequisites

- Programming logic
- Basics of object-oriented programming (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- C++ language overview
- Objects and classes
- Member functions and data members
- Visibility and encapsulation operators
- Constructors, destructors and copy constructors
- Aggregation and composition
- Sub-classing
- Overriding member function
- Sub-typing
- Virtual functions, polymorphism and dynamic binding
- Abstract interfaces and abstract classes

DEVELOPING GUI APPLICATIONS WITH Qt WIDGETS

Overview and Goals

Qt is a cross-platform development toolkit, created over 25 years ago and widely used in industries around the world. QtWidgets is the Qt's module for creating graphical user interfaces (GUI) using the C++ language and offers a wide set of features for creating complex workflows and sophisticated interfaces.

In this training, the basic concepts of Qt (used not only in visual applications) and its main resources for cross-platform development of graphical user interfaces are presented. At the end of the training, the student should be able to productively implement visual systems that can be executed on different platforms.

Prerequisites

- Foundations of C++
- Basics of graphical user interfaces development (desirable)

Duration

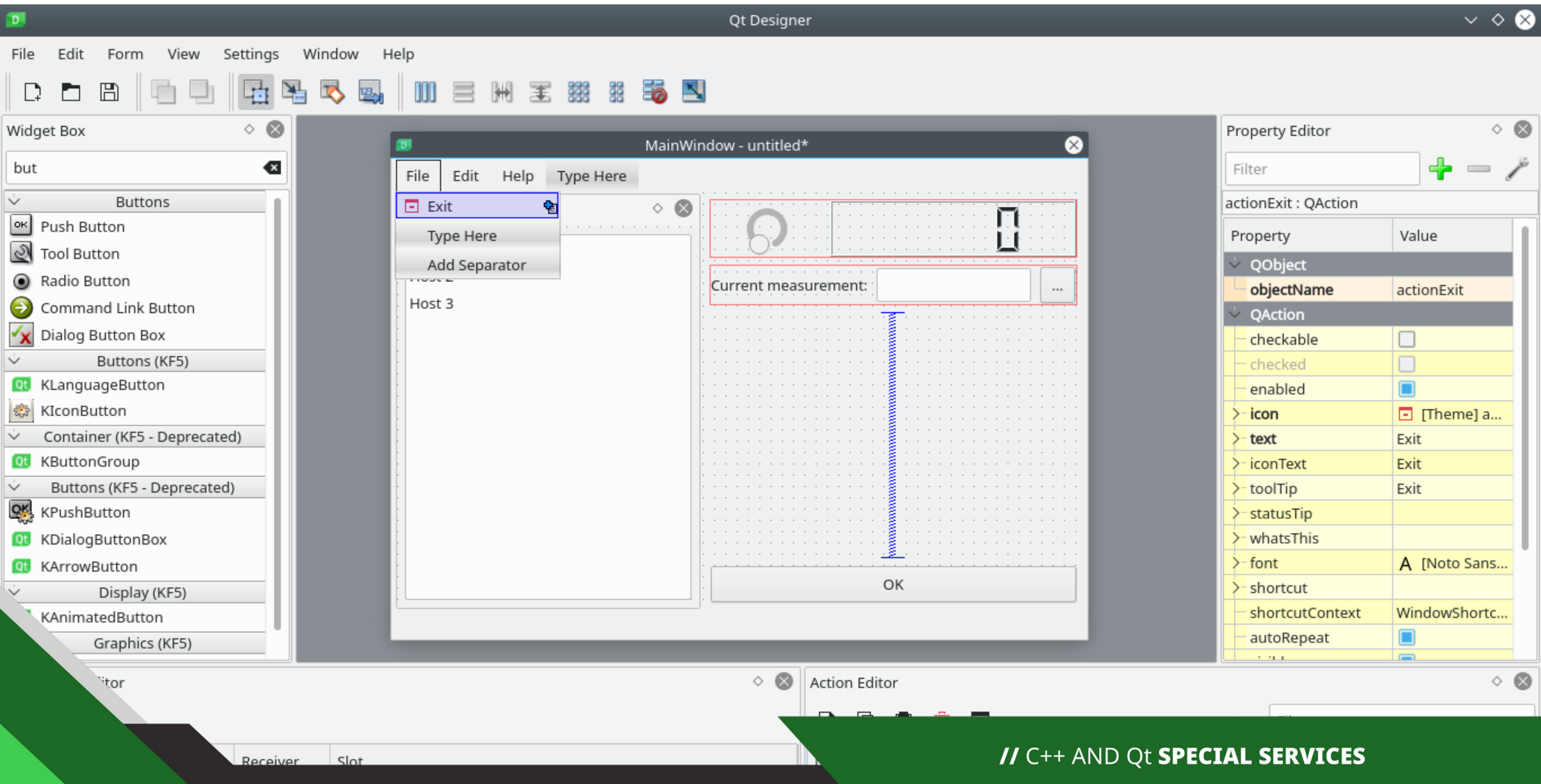
40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Qt overview
- MOC (Meta-Object Compiler) and the Qt Object Model
- Signals and slots
- Object (dynamic) properties
- Meta-objects, parentship relations, and memory ownership
- UIC (User Interface Compiler) and the Qt Designer
- Main windows, layouts and dialogs
- Programming with model/view
- Qt containers
- Database access
- Input/Output with XML and JSON



actionExit : QAction

Property	Value
QObject	
objectName	actionExit
QAction	
checkable	□
checked	□
enabled	☑
icon	[Theme] a...
text	Exit
iconText	Exit
toolTip	Exit
statusTip	
whatsThis	
font	A [Noto Sans...
shortcut	
shortcutContext	WindowShortc...
autoRepeat	☑

FUNDAMENTALS OF QML

Overview and Goals

QML is a declarative language for creating graphical interfaces and is an integral part of the Qt toolkit. Among its main advantages, the following stand out: low learning curve, high productivity, expressiveness in the construction of graphical interfaces for tablets and smartphones, high performance leveraged by the support to run on GPUs and ease of integration with languages such as JavaScript and C++.

In this training, the fundamentals of the QML language and its main resources for creating modern interfaces are presented. At the end of the training, the student should be able to productively design and implement graphical interfaces suitable for execution on multiple devices.

Prerequisites

- Basics of graphical user interfaces development (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Introduction to Qt, QML, and QtQuick
- Basic QML syntax (import sentences, object declaration, comments)
- QML object attributes
- Property binding
- Integrating QML and JavaScript
- QML type system
- Documents and modules
- Handling data entry
- Qt QuickControls
- Positioning elements (bindings, manual, anchors, positioners and layouts)
- Model/View with QML
- QML components and dynamic instantiation
- Animations and state machines
- Internationalization and localization

CLIENT-SERVER APPLICATIONS WITH Qt AND RESTful

Overview and Goals

RESTful is a technology widely used in the implementation of web services, it is based on the HTTP protocol and is influenced by the REST architectural style. Designing client-server applications with back-end functionality implemented as RESTful services has been a common architectural pattern in domains such as mobile applications, web systems and cloud computing applications.

In this training, we cover Qt features for invoking RESTful services and handling XML and JSON data, as well as techniques for implementing RESTful services. At the end of the training, the student should be able to design and implement client-server systems using Qt.

Prerequisites

- Programming logic
- Fundamentals of QML
- Fundamentals of JavaScript (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Introduction to RESTful architectures
- RESTful servers with Ruby Sinatra
- RESTful clients with QML and JavaScript
- RESTful clients with QML and C++
- Working with local caches
- Disconnected operations and synchronization
- Basic features for web scraping
- RESTful client architectures with QML

DEVELOPING ANDROID APPLICATIONS WITH Qt

Overview and Goals

Applications for mobile platforms, such as smartphones and tablets, have drastically changed the way we interact with technology. In this scenario, Qt has the particular advantage of supporting platforms such as Android and iOS with a single codebase, reducing costs and facilitating maintenance, while providing excellent application performance.

In this training, the main resources provided by Qt for implementing applications for the Android platform are presented. At the end of the training, the student should be able to create applications that use resources such as cameras, sensors, geolocation and multimedia.

Prerequisites

- Programming logic
- Fundamentals of QML
- Fundamentals of JavaScript (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Qt for Android overview
- QML on mobile devices
- Touch and multitouch support
- Camera access
- Geolocation and map services
- Using sensors
- Bluetooth and NFC
- Graphic effects and particle systems
- Audio and video
- Integrating Java code
- Implementing Android Services with Qt
- Dealing with permissions
- Generation of apks and AAB packages
- Handling SSH
- Publishing to app stores

DESIGN PATTERNS WITH Qt

Overview and Goals

The software architecture defines basic aspects in relation to its structure and behavior. Well-designed architectures are critical to improving maintainability, postponing software aging, and enabling other non-functional requirements. Design patterns represent quality solutions to recurring problems during the detailed design phase of architectures.

In this training, we present the main architectural styles/patterns used in Qt applications, the design patterns already available in the toolkit, and how new patterns can be implemented. At the end of the training, the student should be able to design and implement effective software architectures for Qt applications.

Prerequisites

- Fundamentals of C++
- Developing GUI applications with Qt Widgets

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Software architecture fundamentals
- Microkernel architectures
- Working with plug-ins
- Building SDKs on top of Qt Creator
- Abstract Factory and Factory Method
- Composite, Bridge, Decorator, and Adapter
- Template Method, Observer, and Command
- Strategy, Flyweight, and Visitor
- MapReduce and Future
- Programming idioms for C++ and Qt

COMPUTER GRAPHICS WITH Qt QUICK 3D

Overview and Goals

Computer Graphics is currently present in a number of areas, from games and digital displays to use in cinema, augmented reality, and scientific visualization. The use of productive frameworks, with good performance and satisfactory abstractions is fundamental in this application domain.

In this training, the main features of Qt Quick 3D are presented: a QML-based module provided by Qt to create modern 2D/3D software applications. At the end of the training, the student should be able to create Qt applications that perform the visualization of polygonal meshes, as well as the use of resources related to texture and animations.

Prerequisites

- Fundamentals of C++
- Developing GUI applications with Qt Widgets
- Fundamentals of QML

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Introduction to Qt Quick 3D
- Polygonal meshes and geometries
- Materials
- Shaders
- Shadow mapping
- Ambient occlusion
- Physics simulation
- Collision detection
- Rigid bodies
- Particles
- Animation techniques

EMBEDDED SYSTEMS WITH Qt

Overview and Goals

Embedded systems are currently present in several areas such as medicine, agribusiness, automotive industry, consumer electronics and entertainment. Developing software solutions that work satisfactorily on platforms with limited memory, processing and communication is a challenging task.

In this training, the basic concepts for building Linux images with Qt support are presented, using buildroot and Yocto technologies. Technologies such as Boot2Qt and Qt Safe Renderer are discussed and the main features of Qt for interaction with sensors and actuators are presented. At the end of the training, the student should be able to design and implement embedded solutions based on Qt.

Prerequisites

- Fundamentals of Linux
- Programming logic

Duration

32h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Embedded Linux fundamentals
- Toolchains, bootloaders and cross-compilation
- Qt rendering plugins for embedded systems
- Building Embedded Linux images with Buildroot
- Building Embedded Linux images with Yocto
- Interacting with general purpose peripherals
- Qt development for Raspberry Pi
- Qt development for Toradex Colibri i.MX 8
- Optimizations for low-power consumption
- Bluetooth LE
- Architectural patterns for embedded systems

OPTIMIZING Qt APPLICATIONS

Overview and Goals

Although Qt is a toolkit designed to enable the development of high-performance systems, some care is needed when working in domains such as high-performance computing, data-intensive systems, or embedded systems. In these cases, correctly using the resources offered by Qt is essential to meet the non-functional requirements involved.

In this training, we discuss critical points where Qt applications generally lose performance, and present the main debugging and profiling tools adopted and basic guidelines for optimizing Qt applications. At the end of the training, the student should be able to analyze existing Qt applications and refactor those solutions aiming at improved performance.

Prerequisites

- Developing GUI applications with Qt Widgets
- Fundamentals of QML

Duration

32h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

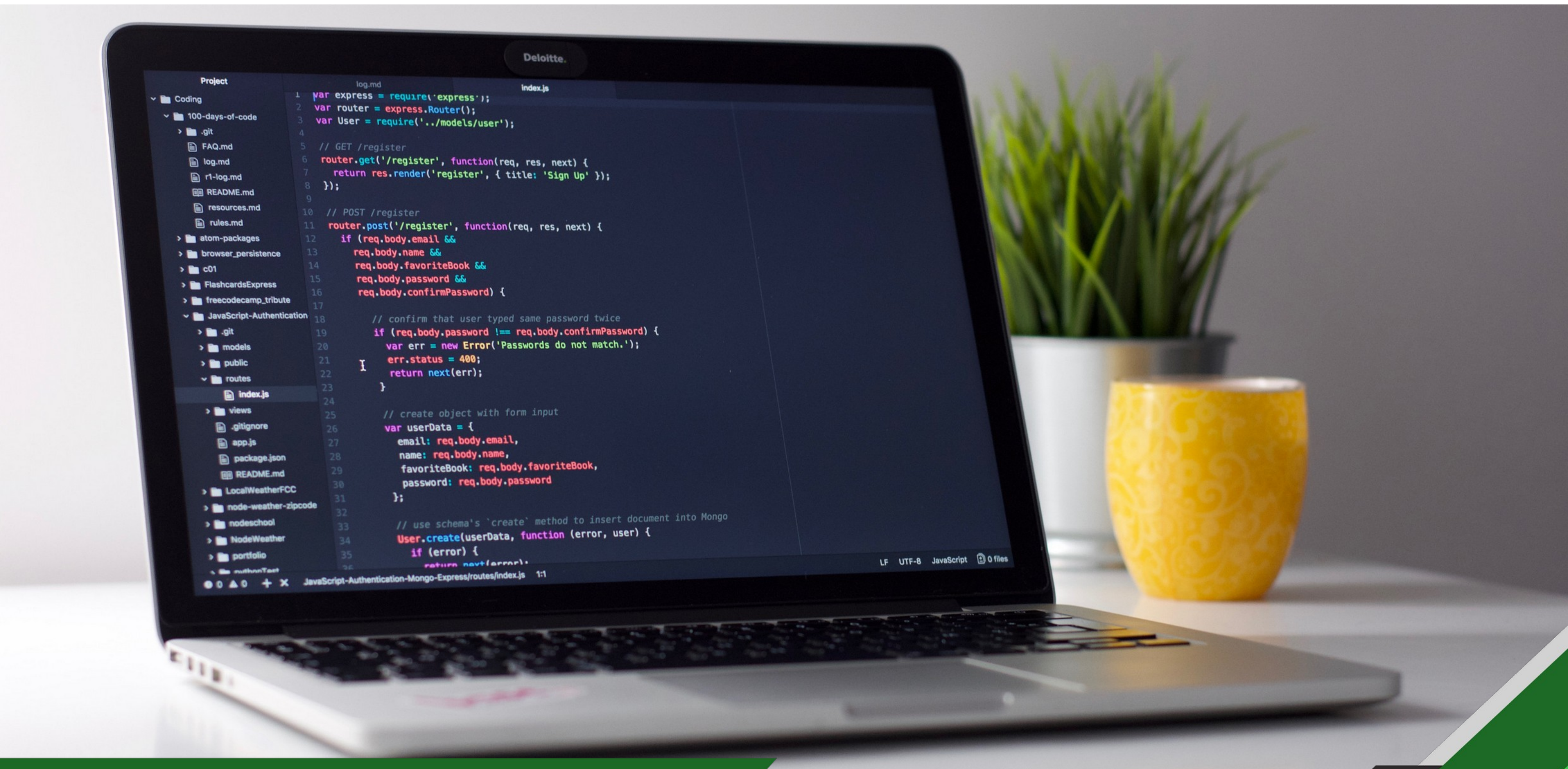
- Debugging and profiling Qt applications
- Performance pitfalls in Qt
- GDB, AddressSanitizer, ThreadSanitizer, and Gamma Ray
- Static code analyzers
- Linux perf, hotspot, and heaptrack
- Optimizing QML applications with the QML Profiler
- Valgrind and Massif Visualizer
- KCachegrind
- Optimizing Qt applications startup
- Optimizing memory footprint

ADVANCED C++

Overview and Goals

C++ is a multi-paradigm language, which makes it suitable for use in a variety of scenarios and application domains. Advanced features such as templates, metaprogramming, Standard Template Library, RTTI and the improvements introduced in C++11, 14, 17 and 20 are quite important for building better quality systems and for solving problems that require not-so-common object-oriented strategies.

In this training, C++ features that contribute to the development of more flexible and robust systems are presented. At the end of the training, the student should be able to master techniques such as p-impl, templates, metaprogramming, smart pointers, constexpr, perfect forwarding and lambda expressions.


Deloitte.
log.md
index.js
JavaScript-Authentication-Mongo-Express/routes/index.js 1:1
LF UTF-8 JavaScript 0 files

```
var express = require('express');
var router = express.Router();
var User = require('../models/user');

// GET /register
router.get('/register', function(req, res, next) {
  return res.render('register', { title: 'Sign Up' });
});

// POST /register
router.post('/register', function(req, res, next) {
  if (req.body.email &&
    req.body.name &&
    req.body.favoriteBook &&
    req.body.password &&
    req.body.confirmPassword) {

      // confirm that user typed same password twice
      if (req.body.password !== req.body.confirmPassword) {
        var err = new Error('Passwords do not match.');
        err.status = 400;
        return next(err);
      }

      // create object with form input
      var userData = {
        email: req.body.email,
        name: req.body.name,
        favoriteBook: req.body.favoriteBook,
        password: req.body.password
      };

      // use schema's `create` method to insert document into Mongo
      User.create(userData, function (error, user) {
        if (error) {
```

Prerequisites

- Fundamentals of C++

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Binary compatibility and and the p-impl idiom
- Templates and generic programming
- Metaprogramming fundamentals
- Standard Template Library (STL)
- Smart pointers
- constexpr
- Type inference (auto and decltype)
- Move semantics and perfect forwarding
- Lambda expressions

ADVANCED QML

Overview and Goals

QML has been, for some time now, the language of choice for developing graphical user interfaces based on Qt. Advanced features such as dynamic element management, integration with C++, customization through styles and WebSockets make QML the technology of choice for areas such as IoT, automotive and aerospace systems, medical devices, among others.

In this training, advanced resources are presented that enable the synergistic integration between QML and C++, as well as functionalities for dynamic management of QML elements. At the end of the training, the student should be able to design and implement QML/C++ hybrid applications in a productive way.

Prerequisites

- Fundamentals of C++
- Fundamentals of QML

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Dynamic management of QML objects (repeaters, loaders and components)
- QML/C++ integration (advanced aspects):
  - Motivation for QML/C++ integration
  - QML/C++ integration types
  - Defining and registering QML types in C++
  - Interacting with QML objects via C++
- QML/OpenGL integration
- User Input (advanced aspects): input focus, virtual keyboard and multi-touch
- QML plugins
- WebChannel and WebSocket with QML
- Working with graphics in QML
- Advanced animations
- Unit tests with QML
- Configuring the look'n'feel of QML applications with styles

QUALITY ASSURANCE, DEVOPS E CONTINUOUS DELIVERY FOR Qt PROJECTS

Overview and Goals

Today's software systems are complex enough to make the use of basic tools such as IDE's and debuggers no longer sufficient to deliver quality systems. Current Software Engineering practices such as DevOps and Continuous Delivery (CD) have become crucial for the development of successful solutions.

In this training, the main technologies and processes responsible for quality management in Qt projects are presented. This includes, for example, the use of appropriate branching models, Continuous Integration (CI), Continuous Delivery (CD), policies for tests and reviews , as well as project management tools. At the end of the training, the student should be able to design and implement a quality management policy suitable for the reality of Qt projects in a given company.

Prerequisites

- Developing GUI applications with Qt Widgets
- Fundamentals of QML
- Basics of project management (desirable)

Duration

32h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Introduction to Quality Management
- Git and branching models
- The quality pipeline
- Qt and automated tests
- Sanity tests
- Static code analyzers
- Qt and GitLab (merge requests, reviews and pipelines)
- Continuous Integration and Continuous Delivery with Qt and GitLab
- Continuous Deployment with Qt and Docker
- Release management
- Quality management and agile methods

Qt FOR PYTHON

Overview and Goals

The Python language is currently one of the most used technologies in the development of systems in several domains and for multiple platforms. Features such as excellent productivity, high-level abstractions, dynamic typing, multi-paradigm support, and the availability of a variety of libraries for various purposes are some of the main factors contributing to the widespread adoption of Python in the industry today.

This training introduces the fundamentals of Qt for Python – the official bindings for using Qt in Python applications. Topics such as generation of bindings via Shiboken, available Qt modules, integration with Qt Designer and Qt Creator, as well as implementation aspects are covered.

Prerequisites

- Programming logic
- Basics of Python (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Motivation for Qt for Python
- History of Qt's Python bindings
- Creating a Qt Widgets Python application
- Signals and slots
- Layouts
- Creating a QML/Qt Quick application
- Integration with Qt Designer
- Data visualization in Qt for Python
- Working with multimedia
- Web content integration with WebEngine
- Integration with C++
- Generating new bindings with Shiboken
- Deployment aspects (with fbs, PyInstaller and cx_Freeze)
- Integration with third-party libraries

GAME DEVELOPMENT WITH Qt

Overview and Goals

The gaming industry is one of the most dynamic, promising, and challenging sectors in the industry today. Developing solutions with high performance, high flexibility, modern UIs and acceptable time-to-market is a challenge faced by all companies in the area. Additionally, designing and developing solutions that work on the multiple hardware and communication platforms currently available makes the situation even more difficult.

This training presents the fundamentals for building 2D and 3D games using Qt. The main aspects related to the rendering of 2D and 3D elements, collision detection, multiple animation techniques, ECS (Entity-Component-System) architectures, functionalities for multimedia and network communication aspects. At the end of the training, the student should be able to design and implement 2D and 3D games in Qt.

Prerequisites

- Programming logic
- Basics of Computer Graphics (desirable)
- Computer Graphics with Qt Quick 3D (desirable)

Duration

40h

Method

- Live classes with fundamentals and practices
- Laboratory practices

Contents

- Developing 2D games with Qt Widgets and Qt Graphics View Framework
  - Coordinate systems and creation/manipulation of graphical items
  - Animating properties with easing curves and detecting collisions
- Developing 2D games with QML/Qt Quick
  - Positioning QML objects
  - Animating properties with easing curves and detecting collisions
- Developing 3D games with Qt Quick 3D
  - Key Features of C++ and QML APIs
  - Polygonal meshes, textures, mappings and collision detection
- Adding multimedia resources
- Fundamentals of building network games with Qt
- Games for Android and iOS

ARCHITECTURE DESIGN & ANALYSIS

Overview and Goals

Delivering quality software systems is an activity that involves the adoption of appropriate techniques and tools, from requirements analysis to system deployment and evolution. An important activity in this process is the design and analysis of the software architecture. It is the software architecture that makes it possible to meet the main non-functional requirements and enables the software to remain useful and modifiable over several years.

In these consulting activities, we carry out the design of architectures for new systems or the analysis of already existing architectures, with the objective of identifying points of malformation or areas that could be improved.

Activities

Architectural design; modeling architectural views; definition of architectural constraints; detailed design and definition of technologies; architectural analysis; architecture integration ↔ quality management; generative techniques; architectures in agile processes.

SOFTWARE DEVELOPMENT AND QUALITY ASSURANCE PROCESSES

Overview and Goals

Developing quality software requires the implementation of mature development processes, where the fulfillment of steps and the products generated are institutionalized. The objective is to systematize activities and adopt techniques and tools that reduce the occurrence of bugs and architectural degradations.

In these consulting activities, we map the characteristics of your business and build a plan to implement a quality development and management process, particularly adapted to the reality of your company.

Activities

Business characterization; determination of the required degree of agility; definition of the development process; definition of the quality management process; selection of practices and tools; deployment plan; team training.

Qt MIGRATION PLANNING

Overview and Goals

The use of legacy systems, with restrictions and/or obsolescence related to architectures, programming languages, graphical user interfaces and integration capabilities, is a very common situation in many corporations. Migrating such solutions to more powerful and flexible technologies should always be considered, with the aim of delaying software aging. In these consulting activities, we analyze the solution currently adopted in your company and specify a migration plan focused on the adoption of Qt and other related technologies. The objective is to carry out a preliminary survey of costs, deadlines and the team needed to implement the new solution.

Activities

Analysis of the architecture of the current solution; analysis of functional and non-functional requirements; assessment of the need for architectural redesign; definition of necessary technologies; migration planning; estimation of costs, deadlines and staff.

Qt LICENSING ANALYSIS

Overview and Goals

Qt is currently licensed in two modalities: Qt Open Source (free licensing) and Qt Commercial (paid licensing). The use of Qt Open Source is regulated through the (L)GPL v3 license, which brings a series of benefits and obligations to be met. In some scenarios, application domains and commercialization models, however, certain aspects of the (L)GPL v3 cannot be met and therefore a commercial license must be acquired.

This group of activities aims to analyze the customer's technical, commercial and strategic scenario, in order to indicate whether the (L)GPL v3 can be fully met or if there are demands for the use of a commercial license.

Activities

Survey of the type of product being developed (application, embedded solution, SDK, etc); analysis of the requirements of LGPL v3; static vs dynamic linking considerations; license types.

OUTSOURCE SOFTWARE DEVELOPMENT

Overview and Goals

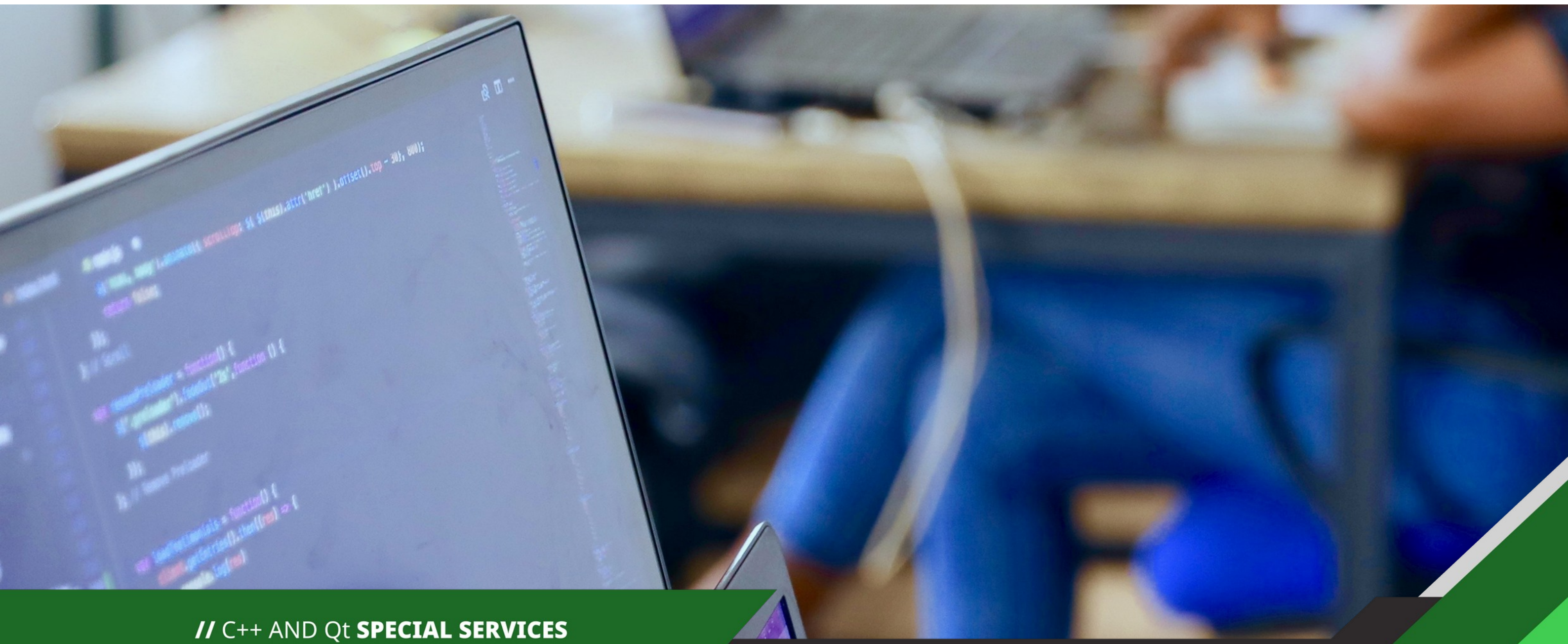
Qmob Solutions has a complete team of Qt engineers and architects to build your desktop, mobile, or embedded system. In addition to developing front-end solutions, we have experienced professionals in the development of RESTful back-ends using technologies such as Sinatra (Ruby), Flask (Python), and Amazon Web Services. As a result, Qmob Solutions delivers complete solutions, ensuring the adoption of best practices in Software Engineering and the delivery of well-architected and easy-to-evolve systems.

Prerequisites

- Requisites elucidation and documentation
- Development project planning

Duration

Variable



Method

- Agile methods
- Lean Architecture

Activities

- Requirements elucidation
- Architectural design
- Detailed project
- Sprint planning
- System development
- Definition of test strategies
- Definition of Continuous Integration (CI) strategies
- Definition of Continuous Delivery (CD) strategies
- Release process
- Deployment planning
- Evolution planning

qmob SOLUTIONS

TRAINING, CONSULTING
& DEVELOPMENT SERVICES

2022

// C++ AND Qt **SPECIAL SERVICES**

https://qmob.solutions