Lappeenranta University of Technology
The School of Industrial Engineering and Management
Degree Program of Computer Science

Bachelor's thesis

Antti Herala

APPLYING QT TO PROGRAMMING COURSES

Examiner: Master of Science Erno Vanhala

ABSTRACT

Lappeenranta University of Technology
The School of Industrial Engineering and Management
Degree Program of Computer Science

Antti Herala

Applying Qt to programming courses

Bachelor's Thesis

2013

33 pages, 9 figures, 1 table, 2 appendixes

Examiner: Master of Science Erno Vanhala

Keywords: Qt, programming, programming courses, teaching

This thesis is fitting Qt into a teaching curriculum. It contains a brief history and a review of the current state of Qt. The current state review contains three aspects: how and where Qt can be used, Qts uses in industry and in educational institutes. As a product, this thesis will provide a small program as a demonstration for lectures, using C++ and Qt Designer with the basic user interface library objects. Another product is a draft about the programming courses of Lappeenranta University of Technology, where Qt could be used to aid students to see how graphical program is created and prepare them to understand the advantages of frameworks and graphical libraries.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tuotantotalouden tiedekunta
Tietotekniikan koulutusohjelma

Antti Herala

QT:n käyttö ohjelmointikurssien tukena

Kandidaatintyö

2013

33 sivua, 9 kuvaa, 1 taulukko, 2 liitettä

Työn tarkastaja: Diplomi-insinööri Erno Vanhala

Hakusanat: Qt, ohjelmointi, ohjelmointikurssi, opetus
Keywords: Qt, programming, programming courses, teaching

Työn tavoitteena on sovittaa Qt opetussuunnitelmaan. Työ sisältää Qt:n lyhyen historian sekä katsauksen sen nykytilaan. Nykytilakatsaus sisältää kolme näkökulmaa: miten ja missä Qt:ta voidaan käyttää, sekä sen käyttötarkoitukset teollisuudessa ja opetuksessa. Työn tuloksena syntyy luentodemonstraatiota varten pieni ohjelma, joka on luotu C++:n ja Qt Designerin avulla ja käyttää olennaisia käyttöliittymäkirjaston olioita. Toisena tuotteena työssä syntyy luonnos Lappeenrannan Teknillisen Yliopiston ohjelmointikursseista, joissa Qt:ta voitaisiin käyttää avustamaan opiskelijoita näkemään, miten graafinen ohjelma luodaan sekä valmentaa heitä ymmärtämään viitekehyksien ja graafisten kirjastojen tuomat edut

PREFACE

This thesis is written as a part of computer science degree of Lappeenranta University of Technology. I would like to thank my examiner Erno Vanhala for his outstanding patience and endless support.

TABLE OF CONTENTS

ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Styling Sheets
GIS	Geographic Information System
GNOME	GNU Network Object Model Environment
GPL	General Public License
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
LGPL	Lesser General Public License
PDA	Personal Digital Assistant
QML	Qt Meta Language
SMS	Short Message Service

1 INTRODUCTION

1.1 Background

Qt is a framework for creating working interfaces and applications for desktop, embedded and mobile platforms. It provides developers a modular C++ class library and different easy to use tools for developers. Qt is a cross-platform framework and so, it targets for example Windows, Mac, Linux/X11, Solaris, Embedded Linux and Embedded Windows, Green Hills Software INTEGRITY, QNX, VxWorks, Android and iOS. This way, Qt provides developers a way to create applications for multiple platforms without the need to customize the source code, so developers may create applications for one platform and then build and run it on another platform. [1]

The Qt framework was created by Haavard Nord and Eirik Chambe-Eng, in a company called Trolltech, and it was first published publicly in May 1995 as Qt 0.90 for Windows and Unix with the same API (Application Programming Interface). In 1996, European Space Agency became a customer of Trolltech and so Qt 0.97 was created, and later that year Qt 1.0 and 1.1 were released. Year 1999 marked the end of Qt 1, when Qt 2.0 was released in June. Qt 2.0 brought a new open source licence and it won the LinuxWorld award for best library/tool. [2]

In 2000, Trolltech released Qt for embedded Linux and X11 publicly and the first version of Qtopia, an application platform for mobile phones and PDAs (Personal Digital Assistant). These new instalments won more awards from LinuxWorld from 2001 to 2004, while Qt 3.0 was released in 2001, expanding the availability for Windows, Mac OS X, Unix and Linux (embedded and desktop). [2]

Trolltechs final major instalment, before selling Qt to Nokia, Qt 4.0 was released in 2005 and it expanded the usability considerably compared to its predecessor. Qt became accessible to non-C++ programmers through different, unofficial language bindings, e.g. PyQt for Python programmers and Qt Jambi for Java programmers. [2]

In 2008 Nokia acquired Qt from Trolltech and continued to develop it, mainly for their own platforms.[1] However, Nokia sold Qt to Digia completely in 2012.[2] Digia has developed Qt further and released Qt 5 in late 2012.[3]

1.2 Objectives and limitations

This thesis contains three parts, two research questions that'll be answered in the text part and the third part's a demo program in Qt, that is created in C++ for a lecture demonstration about Qt, its tools and its libraries. The research questions addressed are presented below:

- What is the current state of Qt in industry and teaching?
- How does Qt fit into teaching generally and in Lappeenranta University of Technology?

1.3 Structure

The second section of the thesis will concentrate to Qt's current state in industry and teaching, trying to provide some insight, how companies and universities deploy Qt in their daily business. The third section aims to fit Qt into a teaching curriculum and find its positive sides and strengths for students, so they might apply it in their own projects. It also contains a small presentation about Qt Creator. This section is followed by a demo version for a programming course and afterwards Qt is fitted into different courses of Lappeenranta University of Technology.

[1] http://arstechnica.com/information-technology/2008/01/nokia-buys-trolltech-will-become-a-patron-of-kde/
[2] http://www.guardian.co.uk/technology/2012/aug/09/nokia-sells-qt-software-business
[3] http://blog.qt.digia.com/blog/2013/01/31/qt-5-0-1-released/

2 QT'S CURRENT STATE

From the software engineering's point of view, creating GUI (Graphical User Interface) is a major hurdle in developing applications. Especially challenging is to create a portable GUI, one that can be reused on multiple platforms and multiple devices. For every program, it is necessary to design a useful and working GUI and its importance is on the same level as the programs logic and data structures. This planning process includes four main topics, that has to be covered. [3]

- Programming language
- Graphical toolkit/library
- Virtual machine/runtime environment
- Operating system

With these main principles, a GUI can be separated from the program, increasing its portability and reusability. Compared to the program, the GUI can be programmed with a different language, target multiple platforms and use different virtual machines. [3]

First, Qt provides multiple third party language bindings, so it can be developed with any major programming language in current use [4]. Second, one of Qts major advantages is, that it has a common API for multiple platforms, so it is possible to build one GUI for different platforms easily as presented in Figure 1 [5]. Third, Digia provides Qt its own IDE (Integrated Development Environment), Qt Creator, but Qt is also supported by multiple different IDEs, so developers can choose their programming language, platforms and IDEs from the ones they are already using [6].

Qt is also very useful to developers; Digia provides free training material for beginners as well as for experienced users. This way every developer can learn to apply Qts many functions and features whenever they feel like they need them. Digia also provides teaching resources directly for educational institutions, which provides safe, reliable and up to date source about Qt. [7]

2.1 Different frameworks

Qt is not the only GUI framework out there, but it is one of the most popular alongside with GTK+, especially when developing to X11. [8]

GTK+ is a GUI toolkit for cross platform development, however it provides support only to GNU/Linux and Unix, Windows and Mac OS X, which makes it more limited than Qt [9]. And just like Qt, GTK+ provides multiple language bindings for different programming languages, but there is no consistency what version supports each language and mainly just official GNOME (GNU Network Object Model Environment) bindings support all versions [10]. The main difference between these two toolkits is, that while Digia provides licensing terms for Qt by GPL (General Public License) and Qt Project by LGPL (Lesser General Public License), GTK+ is provided only by LGPL [11] [12].

2.2 Different platforms

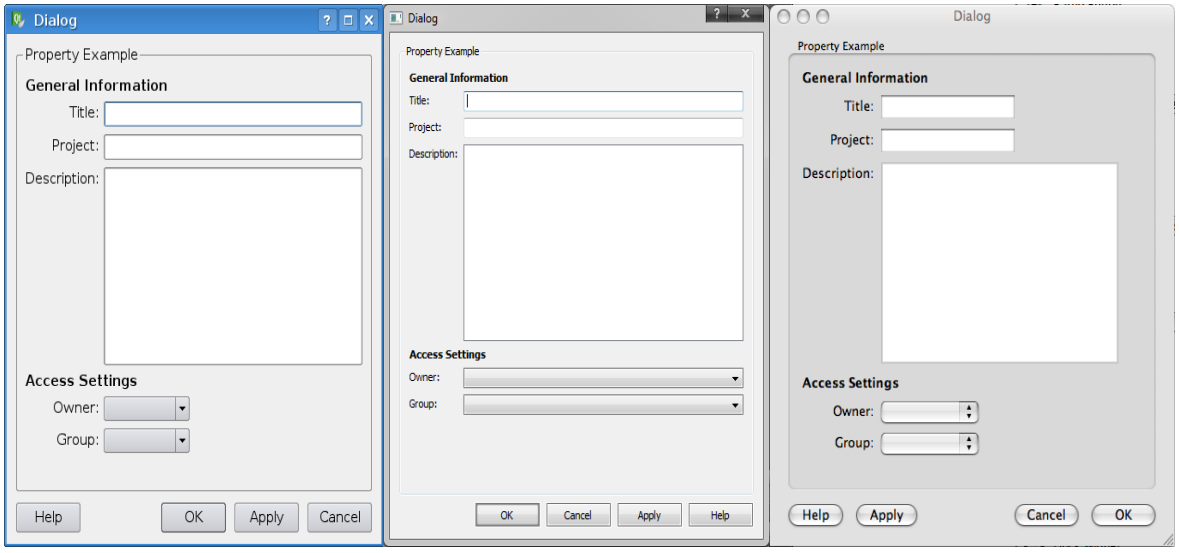

Figure 1. One code, three platforms (KDE, Windows, OS X) [6]

Digia provides a table presenting the support for variety of platforms and their supported compilers for Qt 4.7, 4.8 and 5.0. They have divided these supports to two groups, primary supported and secondary supported platforms and present tables for both, desktop and

embedded. For primary supported platforms, that are presented in Appendix 1, these supports are constantly monitored and regularly tested so bugs and errors can be fixed as fast as possible between releases. Secondary supported platform in Appendix 2, on the other hand may have errors that are not necessary fixed between releases, since they do not undergo through continuous testing and are not in the main focus. This divination comes from the fact, that some platforms are more popular than others, so Digia wants to prioritize their support and stable releases for the developers. [5]

As presented in Figure 2, Qt fits in between the operating systems kernel and the created application. The application calls the GUI component through front-end user interface and it gives control to assigned back-end component, that works with kernel and actually executes the command. [13]

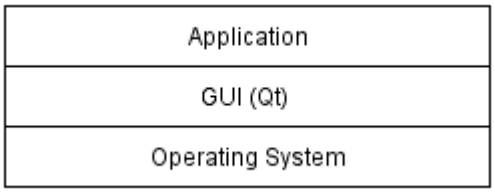

Figure 2. System platform structure in Qt/Embedded (modified from source 13)

2.3 Different languages

Qt Project provides an open source developing community for Qt, QML (Qt Meta Language) and its developing tools. From there has also risen the third party language bindings, that adds the versatility of Qt, especially in programming languages. While Qt Project provides language support directly to C++ and QML, third party developers are continuously updating support for other languages, that are partially presented in Table 1. Some of the language bindings are incomplete or they just use Qts GUI toolkit partially, but major languages are covered. JavaScript is not listed as a binding, since it is built into QML with CSS (Cascading Styling Sheets) and HTML (HyperText Markup Language). [4]

Table 1. Some third party language bindings, whole list can be found from source [4]

Name	Language	Qt 4	Qt 5	Platforms	Other information
PySide	Python	4.6, 4.7 and 4.8	Not supported	Linux/X11, Maemo 5, Windows, Mac OS X	Open source project, LGPL
PyQt	Python	Supported	Supported	Linux and various Unix, Windows, Mac OS X	Created and developed by Riverbank (GPL/commercial)
QtAda	Ada2005	4.6 and newer	Not supported	Windows, Linux, Mac OS X	
Qyoto	C# and other .NET languages	Supported	Not supported	Debian, Ubuntu, Arch Linux, Windows, Mac OS X	
qtHaskell	Haskell	4.3 and newer	Not supported	Linux, Windows	Haskell added with Qts logic, e.g. signals and slots

2.4 How has Qt spread in industry

Because of Qts versatility, it has been used in multiple projects of different scales, varying from operating systems to specialized, pro-used softwares. Even multiple regularly used and widespread programs have their GUI created with Qt, as the following examples indicate.

2.4.1 Skype and Google Earth

One widespread software that uses Qt is Skype; a free, multi-platform software, that is used for messaging (instant or SMS (Short Message Service)), calling over internet or through landlines, sharing files or screen and video calls. Skype is an application, that can be found on desktops as well as on mobile platforms and it can connect between normal mobile phone and a computer, using mainly its software connection over internet, but it also can connect to landlines. [14]

A more specialized Qt developed software is Google Earth; a virtual globe, map and geographical tool created by Google and released in 2005. The program maps the world using images obtained from satellites, aerial photographers and GIS (Geographic Information System). Google Earth is supported on multiple different platforms, operating systems and devices; it has versions for desktop, mobile and web. [15]

2.4.2 EmoCoW

EmoCoW, that stands for Emotional Color Wheel, is a innovative tool for real-time facial animation. The system itself was created using C++ and its GUI uses Qt framework. The interface contains a circle, where two sets of emotions are mapped into two separate wheels and those wheels are nested into each other. The user can use a 6-axis mouse to control both of these wheels simultaneously as well as a vertical blend slider, which blends any two emotions together as a result. This software is created for animators and movie creators, so they could use it on the scene in real-time and cut down the time used afterwards, with no need of memorizing the necessary facial emotion patterns. [16]

2.4.3 Sailfish and Ubuntu Unity

Sailfish is a new mobile platform, developed by Jolla. The user interface of Sailfish leans heavily on Qt framework and its interfaces are created by using QML. This ensures a rich set of components for any user interface in Sailfish. [17]

The core of Sailfish is built together with Qt and Mer, called Mer core [18]. Mer is a open core distribution that optimizes itself to mobiles and it is powered by Qt/QML and HTML5 [19]. The core as well as Sailfish itself is built up from MeeGo technology, which provides it a solid base for future improvement [19].

Ubuntu Unity is a desktop platform, that is built on an OpenGL toolkit called Nux, while the other Unity shells (e.g. mobile) are developed with Qt/QML. This was justified with the fact, that Qt 4 did not provide a sufficient support for OpenGL, but that changed in Qt5/QML2. In creation of Ubuntu Unity Next, it has been decided that the desktop version of Unity is also completely developed with QML. [20]

2.5 How is Qt used in teaching

Qt in its current form is being used in 26 different counties and in 9 educational institutes in Finland, presented here [7]:

- Aalto University School of Science and Technology
- Arcadia
- Central Ostrobothnia University of Applied Sciences
- Helsinki Metropolia University of Applied Sciences
- Kajaani University of Applied Sciences
- Oulu University of Applied Sciences
- Savonia AMK
- TAMK University of Applied Sciences
- Tampere University of Technology

At least Oulu University of Applied Sciences, Kajaani University of Applied Sciences and Helsinki Metropolia University of Applied Sciences provide a basic course about Qt using material by Digia. The courses demand that students have knowledge about object-oriented programming and the course scope is from 3 to 5 Ects. The courses also include a larger assignment, where students have to create their own GUI programs. [21][22][23]

In Aalto University School of Science and Technology, Qt is applied in different ways. In User Interface Construction Qt is one of the tools being used by students, to create a working GUI for a bigger assignment in a course that teaches, how to create a working user interface that is fit for testing [24]. Before this course the students will use Qt as a voluntary library in C++ programming, where they must create a GUI for a project of varying topic [25].

3 HOW TO FIT QT INTO TEACHING

Usually, basic programming course does not prepare student to use complex frameworks and class libraries. Even if a course presents one or many frameworks, it does not give students readiness to use it or them efficiently. Especially GUI frameworks are very underrated in this aspect, even though beginner programmers have only seen the interface of the programs they have used. By creating only text based programs, students may lose motivation and enthusiasm when comparing their own programs to those they use. The second problem arises with students, who use a framework they found and create their assignments using it, forfeiting the required functionality of the assignment. By teaching the students the art of using frameworks and especially GUI frameworks, they gain a lot of experience for their future careers. [26][27]

3.1 How to use Qt in teaching environment?

Teaching environment can be approached in two different ways, teaching how to use Qt and its IDE Qt Creator or using Qt to create a software, that students can use in their studies.

Teaching Qt can be considered its own course, where the main focus is in applying Qt efficiently, but it can be taught as a part of frameworks course, as a part of a whole. For a few lectures, Ken Harness provides a great outline, how to efficiently teach Qt in a small package [27]:

1. Overview of Qt and its uses
2. Availability of Qt (how to get it)
3. Simple interfaces
    a. Simple example (temperature conversion)
    b. Exercise (4-function calculator)
4. Simple graphics
    a. Description of basic functions
    b. Example in simple drawing

5. Polymorphism example
    a. Description of a program that animates "Drawable" objects
    b. Quick review in inheritance and virtual methods and their use in this program
    c. Exercise in creating a new "Drawable" object

For a whole course, Digia has provided 16 lectures worth of material, already cut into 2x45 minutes sets. Those slides go through Qt bit by bit, including multiple laboratory exercises and bigger assignments. [28]

A way to use Qt as a simulation tool requires a abstract and difficult field of science, e.g. Exploration Seismology. By using a program created with Qt, students gain the advantage to actually see theories in practice and they can also modify the parameters themselves, to see how that affects the simulations. Qt is a user friendly tool for creating this program, for its wide use, great support and open source license. [29]

3.1.1 Pros

The pros and cons of using Qt as the first GUI framework are a bit semantic in a sense, since students have to face some framework, but since Qt is provided with a documentation and learning materials, it is a great choice for beginners.

Since Qt has been created with C++, it is rather easy for students to adapt into it, especially if used in C++ course, when a basic knowledge about C/C++ and objects has been acquired. C++ also is useful language when considering the professional careers of students, providing them a basics for using frameworks in general. [26]

3.1.2 Cons

While C++ is a useful language, there are still other languages and frameworks, that employers require even more, e.g. .NET frameworks. .NET also gives students readiness for web based technologies, more than Qt/QML can provide. [26]

HTML5, the newest version of HTML, provides even more possibilities for web based technologies, especially since it is a web based language and runs on browser, so it automatically becomes a language for multiple platforms and devices. Also, to develop a program in HTML5 does not require a complex IDE, so it can be set up much more lightly than Qt. [30] Another so called HTML-using competitor for Qt is PhoneGap, that uses HTML, CSS and JavaScript to develop a program for any mobile device by building the code for selected platform. [31]

Seeing how HTML5, CSS and JavaScript are seen as competitors for Qt, Qt Project has noticed this and made a possibility in Qt Creator for developers to actually create HTML5 applications. QML, as a language also has very close resemblance to CSS and JavaScript, thus rendering the competition to minimum and leaving the decision for the developers themselves. [32]

3.2 Qt Creator

Qt Creator is a cross-platform IDE, a part of Qt Project. It is created especially for programmers, who use Qt in their daily life. Qt Creator is a tool mainly for experienced programmers with a knowledge of Qt, but it also includes an informative help function, where the Qt classes and their methods, signals and slots can be easily found. Visually Qt Creator works like any other high-level IDE; it has functions for code folding, style hints, syntax highlighting and many other user friendly features. [33]

Qt Creator includes also a visual GUI designer called Qt Designer, although it is used in other IDEs as well. In Qt Designer, the Qt widgets can be easily placed on a form and their signals and slots can be connected visually, without the need to actually code them. This however stands only to default signals and slots, the customized properties must be still connected in the C++ file. By using Qt Designer, the user can build and design the form without any knowledge about the actual code and the form can be customized using the modifiable stylesheets, that resembles CSS very closely. [33]

4 LECTURE DEMO USING QT

Part of this thesis was to create a program, that can be used as a demo in a programming course. The programs main focus is in the user interface, that tries to use the basic widgets, signals and slots provided by the toolkit, with some customized parts. The demo is created using Qt Creator 2.7.2 and NetBeans 7.3.1 and its user interface is built with Qt Designer and C++.

4.1 What the example program contains

The demo program is a go-kart simulator, presented in Figure 3, where user can develop a new car by giving it the required specifications. After the car has been created, it can be tweaked using the sliders on main window and it can be put to a racetrack, to test its speed and lap time. However, there can be only one car and the used one must be trashed until a new one can be created, but the cars data and lap times are saved in a top 10-list. If user accidently causes an error, the program opens a message box with information about what went wrong.

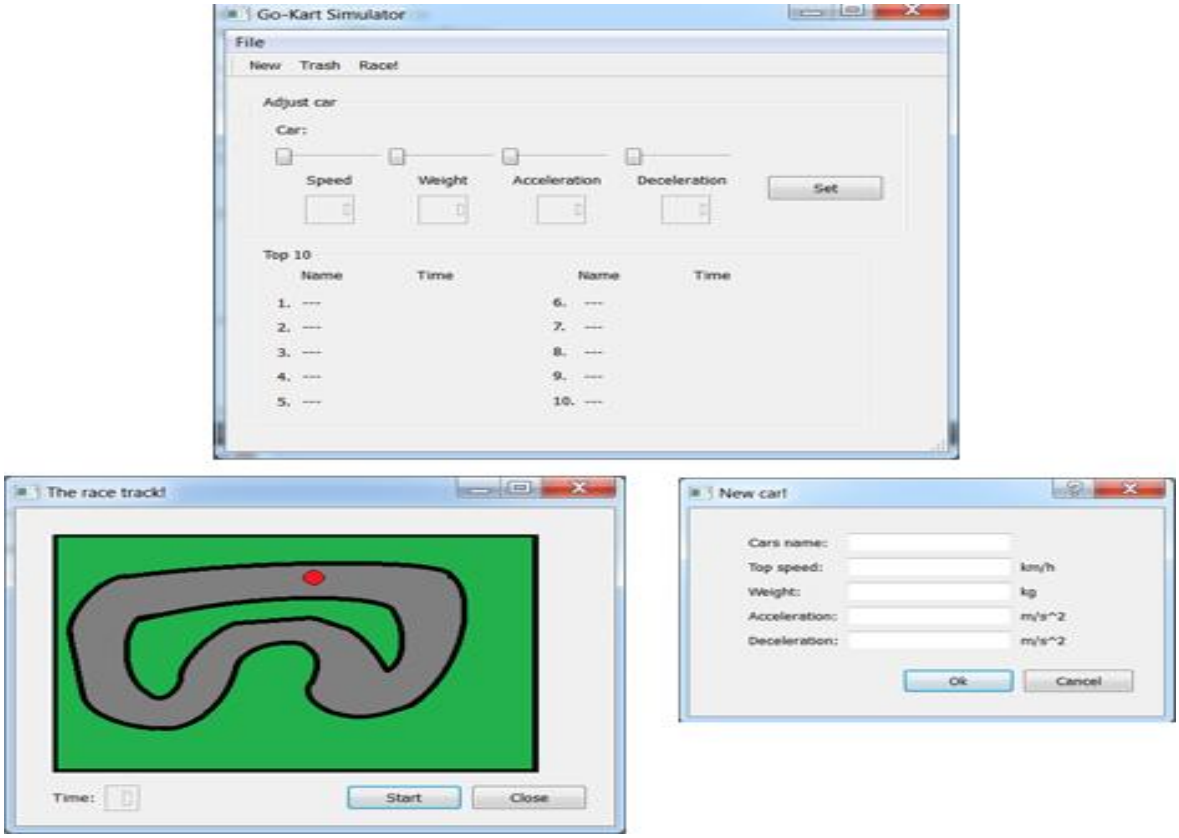

Figure 3. All windows of the program

4.2 How the GUI is constructed

The user interfaces of different windows can be easily presented as a class diagram or as a list from Qt Designer, presented in Figure 4. All the windows are constructed using the basic UI classes and their signals/slots, that can be easily placed into a form and bound to each other using Qt Designer. Every time a widget is placed on another widget, the bottom widget becomes the parent and the one on top becomes the child. Qt Designer creates this connection automatically, when a widget is placed, but it can be created manually in code as well, in the constructor of the parent.

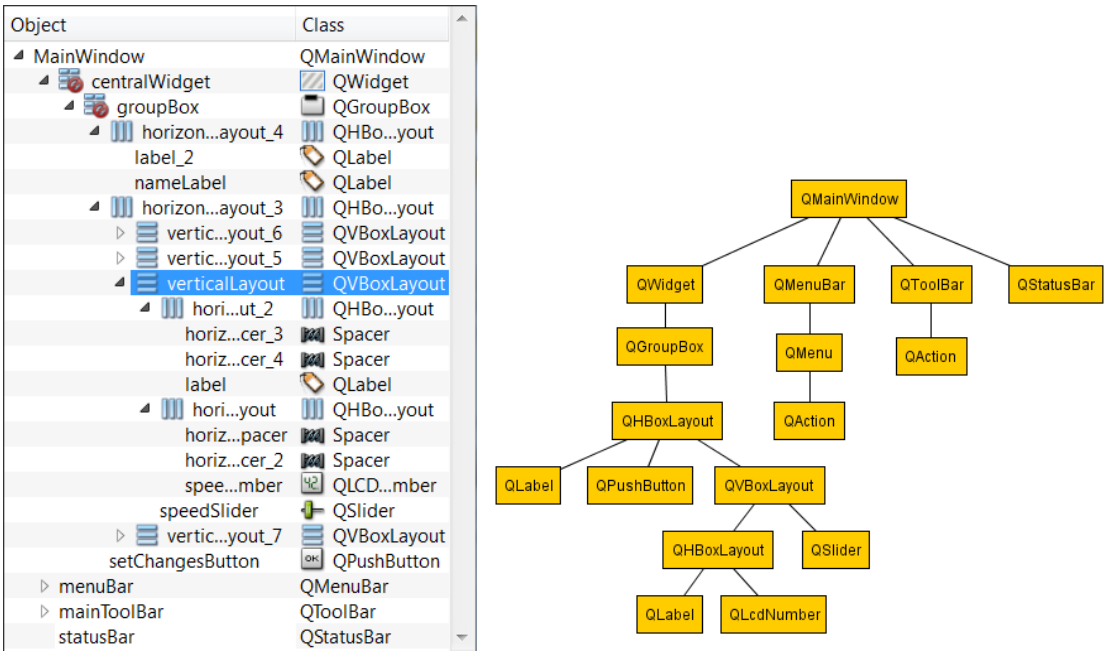

Figure 4. Main window user interface as a list and a diagram without top 10-list

Signals and slots are Qts logic to send execution commands inside a window or from one window to other. Seeing how Qt Designer makes connecting widgets easier, it also provides a way to create the signal and slot connections without the need of actually writing any line of code as can be seen in Figure 5. This however works only on standard signals and slots, the customization may be necessary, when sending a lot of data or multiple variables, since Qts logic demands, that signals and slots have the same amount of parameters with matching data types.

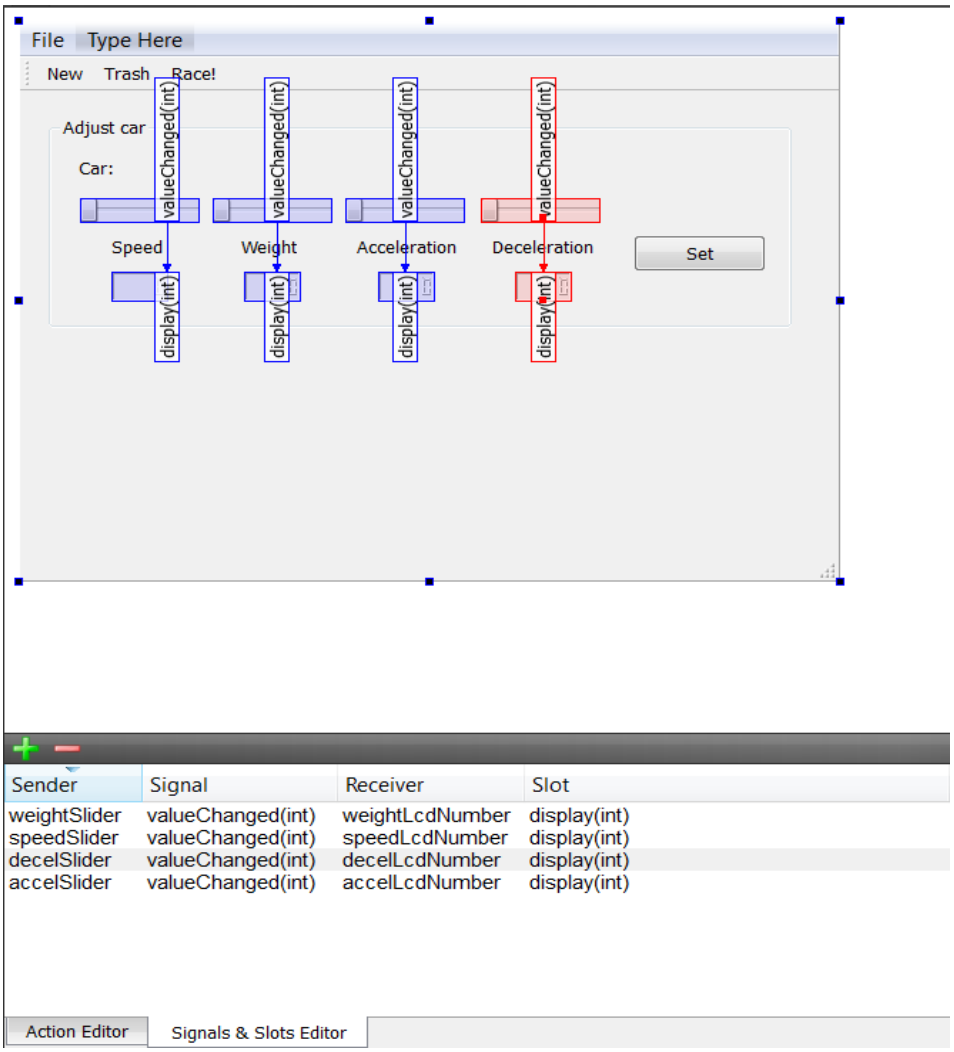

Figure 5. Qt Designer Edit signals/slots-view of main window

In Figure 5, the QSlider class has been connected to QLcdDisplay-class, so when the sliders value is changed, the value in the display changes as well. This is created with QSliders signal valueChanged(int), that sends the modivied value of the slider to a slot called display(int), that shows the value in display.

Sometimes it is necessary to send data from one window to another, e.g. when creating a new car. In that case, the data gathered in one window must be sent to another, using the signals and slots. Connecting two windows is not possible in Qt Designer, so a signal and a slot must be created separately. It is possible to connect a custom signal to a standard slot, but only if the parameter types are matching. Creating and using a custom signal is presented in Figure 6 and a custom slot in Figure 7.

```
25  signals:
26      void sendData(QString, int, int, int, int);
```

```
53      else {
54          emit sendData(name, speed, weight, accel ,decel);
```

Figure 6. Codes from *.h and *.cpp of the sending UI class, declaring and emitting signal

A signals return type must always be void and the data that is sent must be declared. When the signal is emitted the data is sent to the slot, where it has been connected.

```
26  private slots:
27      void on_actionNewcar_triggered();
28      void on_actionExit_triggered();
29      void emergeError();
30      void receive data(QString, int, int, int, int);
```

Figure 7. From the *.h file of the receiving UI class, declaring slot

After the signal has been created and it has data to send and the slot has its execution created, these signal and slot still have to be connected, using the connect()-function from QObject in Figure 8.

```
16      connect(car, SIGNAL(sendData(QString, int, int, int, int)),
17              this, SLOT(receive_data(QString, int, int, int, int)));
```

Figure 8. Connecting objects from Figure 6 and Figure 7

The objects "car" and "this" in Figure 8 are the widgets that are connected. Widget "car" (the new car window) emits the signal and widget "this" (the main window) is the one that has the slot prepared to use the data. Connecting signals to slots normally works the same way, but in that case the signals and slots does not have to be declared in the headers.

Qt Designer also provides the developer an easy way to modify any object by having a property window as a part of the tool. Using that window, the developer can change all the properties of the object from every class it inherits. In Figure 9 the object modified is an instance of QPushButton, but it still has basic properties from QAbstractButton and QWidget, that determine how and where the button is shown in the form, how it reacts to changes, how the cursor will be show on it, etc. Without this window, all the changes from the button default should be modified in its constructor in code.

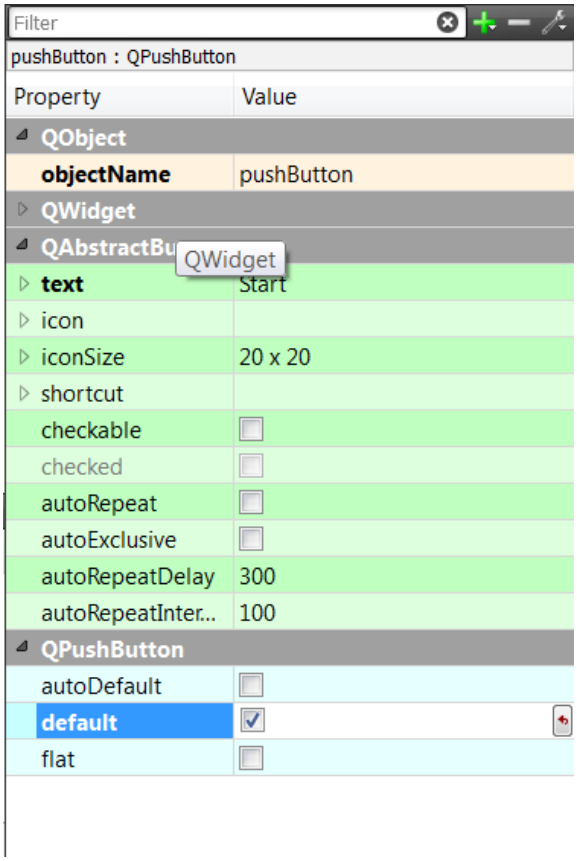

Figure 9. Properties of start-button in racetrack.ui

5 DISCUSSION

There are few courses available, where Qt could be applied in some way. However, these opinions and ideas are from a students point of view, using first-hand experience and insight concerning the matter. These courses could be:

- Ohjelmoinnin perusteet (Fundamentals of Programming)
- Käytännön ohjelmointi (Practical Programming)
- Tietorakenteet ja algoritmit (Data Structures and Algorithms)
- Olio-ohjelmointi (Object Oriented Programming)
- Käyttöliittymät ja käyttäjäkeskeinen suunnittelu (User Interfaces and User Centric Design)

5.1 Case LUT

In the basic programming courses, Fundamentals of Programming and Practical Programming, Qt does not stand in a big focus, so it could be used as a demonstration to create a user interface, mainly because of Qt Designer, where beginner programmers can see, how easily a GUI can be created. The cons of this are that Qt Designer is a C++ based software, it doesn't work with Python without problems and in C a GUI is not that necessary in most cases, so it should not be used by students. However, both courses could benefit if students were given a working front-end GUI and they would have to construct its back-end, the working logic of the program. This would provide them a working program, that is more than text on a screen and number choices from a list. It would also prepare them to the fact, that usually the programmers must work with the code done by someone else.

One possibility in Fundamentals of Programming is that either Qt Creator or any commonly used IDE is presented to the students. Working with a blank, text-based IDE is much work for beginners, since it doesn't provide any spell checks or dynamic error handling, which could be useful in their cases.

As a teaching tool creator, Qt could be useful in Data Structures and Algorithms, where a lot of sorting algorithms and decision trees are presented, by providing a GUI that does the sorting and decisions graphically. This would provide students a consistent and definitely a correct way to see, how the algorithms work. It could also serve as a bigger assignment, where students must create the back-end logic for different algorithms with a working front-end.

In Object Oriented Programming, Qt has been used, but it could be used more as a way of teaching. Qt in itself is very hierarchical and object oriented, so it is natural to use its frame to show the students how objects and generalization works. The real life examples that are normally used are fine as they are, but Qt could be a way to visualize what that actually means.

User Interfaces and User Centric Design is a course where students have to create a GUI, but it doesn't have to actually work. With Qt and other frameworks/web languages, the course could be driven to actually design and implement a GUI, when it currently concentrates only to design. The design is an important part of GUI, but when students aren't demanded that the program has to work, its benefits aren't the best for students and not that close to practice. Students could then actually do something instead of writing down, what they might do if they had to.

5.2 Emerged problems

When the thesis was planned, it was decided to use Python, C++, Java and JavaScript for the demo program. After more research, Java was dropped since its support was rather old. Next problem emerged with JavaScript, since Qt doesn't provide a language binding for it, but it has been embedded into QML so JavaScript was replaced with QML. However, after completing the program with C++, there were two notions:

- PyQt (python language binding) does not built in support in IDE for Qt Designer, the tool that was used to create the program

- QML is a language only for user interfaces, it holds no powerful means to create back-end logic for the program, so the program would still need a logic created with C++

Because of these problems, it was easier to drop Python and QML from the plan and stick to C++ entirely.

6 CONCLUSION

This thesis provides an up to date review to the current state of Qt, covering different frameworks as well as platforms and languages that can be used to work with Qt. Also the uses of Qt in industry are being reviewed, using such examples as Skype, Google Earth, EmoCoW, Ubuntu Unity and Sailfish. The uses of Qt in educational institutes are presented by targeting the institutes of Finland and presenting few of their courses that actually have Qt in use or some relation to it.

When fitting Qt to teaching curriculum, two possible uses are approached: teaching the use of Qt as a part of a course or as its own course or it could be used to create a GUI for a course, where the different theories and models could be shown. For students there are few positive sides to use Qt, but there are some negative aspect as well, mainly about targeting the software. Regarding this, a simple draft considering the courses of LUT is presented, without any consent from the department only to provide a students insight and experience about some courses, where Qt could and maybe should be used.

The lecture demo is a go-kart simulator, that uses the basic, fundamental classes of Qt and it is presented with Qt Designer. The example shows, how to actually create a working Qt application using C++, concentrating on the GUI and basic functions, with some instructions concerning the uses.

After experiencing the use of Qt it has become clear, how hard it is actually to design a working user interface that is easy and pleasant to use. This knowledge should be experienced by information technology students while they are still studying for real life.

To continue this thesis, more and more lecture demos and helpful assignments could be used to support the education. Using graphical interfaces help students to create better, working programs by themselves and maybe even motivates them to study harder, when the materials are visually pleasant.

REFERENCES

1. Digia, *The Power of a Complete Development Framework*, URL: http://qt.digia.com/Product/, accessed 19.5.2013
2. Blanchette J; Summerfield M, *C++ GUI Programming with Qt 4*, 2nd edition, Prentice Hall, USA, 2008
3. Bishop, Judith, *Multi-platform User Interface Construction - A Challenge for Software Engineering-in-the-Small*, ICSE'06, May 20–28, 2006, Shanghai, China
4. Qt Project, *Programming Language Support & Language Bindings*, URL: http://qt-project.org/wiki/Category:LanguageBindings, accessed 9.6.2013
5. Digia, *Supported Platforms*, URL: http://qt.digia.com/Product/Supported-Platforms/#.UbNqofn0G6M, accessed 8.6.2013
6. Johan Thelin, *How to Be Cute on All Desktops with Qt*, URL: http://www.linuxjournal.com/magazine/how-be-cute-all-desktops-qt, accessed 15.6.2013
7. Digia, *Qt in Education*, URL: http://qt.digia.com/Product/Learning/Education/#.UbNntfn0G6N, accessed 8.6.2013
8. X.org Foundation, *Documentation*, URL: http://www.x.org/wiki/Documentation, accessed 24.6.2013
9. GTK+ Project, *Cross Platform*, URL: http://www.gtk.org/features.php (24.6.2013)
10. GTK+ Project, *Language Bindings*, URL: http://www.gtk.org/language-bindings.php, accessed 25.6.2013
11. GTK+ Project, *Are there any licensing restrictions?*, URL: http://www.gtk.org/, accessed 25.6.2013
12. Qt Project, *Qt Project*, URL: http://qt-project.org/, accessed 25.6.2013
13. Ran, F; Wang, T; Ran, S, Implementation of the Qt4 Test Bench Based on Arm9, 2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM)
14. Skype, *Features*, URL: http://www.skype.com/en/features/, accessed 02.08.2013
15. Google, *Google Earth Showcase*, URL: http://www.google.com/intl/en/earth /explore/showcase/, accessed 05.08.2013

16. Sielaff, C, *EmoCoW - An Interface for Real-Time Facial Animation*, SIGGRAPH Asia 2010, Seoul, South Korea, December 15 – 18, 2010.
17. Sailfish, *Technology*, URL: https://sailfishos.org/about-technology.html, accessed 26.6.2013
18. Sailfish, *Architecture*, URL: https://sailfishos.org/about-architecture.html, accessed 26.6.2013
19. Mer, *Mer*, URL: http://www.merproject.org/, accessed 26.6.2013
20. Gunn, K, *Unity Next*, URL: https://wiki.ubuntu.com/UnityNextSpec (26.6.2013)
21. OAMK, *Johdatus Qt-ohjelmointiin*, URL: http://www.oamk.fi/koulutus_ja_hakeminen/avoin/opetustarjonta/opintojaksot/?id=3753, accessed 26.6.2013
22. KAMK, *Qt-ohjelmointi*, URL: http://www.kamk.fi/loader.aspx?id=75656f14-1ae2-4873-8065-89325f1f0293, accessed 26.6.2013
23. Metropolia AMK, *Qt kehitysympäristönä*, URL: https://wiki.metropolia.fi/pages/viewpage.action?pageId=11633843, accessed 26.6.2013
24. Marko Nieminen, *Assignments*, URL: https://noppa.aalto.fi/noppa/kurssi/t-121.5300/harjoitustyot, accessed 05.08.2013
25. Raimo Nikkilä, *Assignments*, URL: https://noppa.aalto.fi/noppa/kurssi/as-0.3301/project_work, accessed 05.08.2013
26. Pais, R; Barros, J, *Two Possible Approaches for an Intermediate GUI Course*, ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal
27. Harness, K, *Graphics and User Interfaces in C++ with Qt*, 2005 by the Consortium for Computing Sciences in Colleges
28. Digia, *Qt in Education Course Materials*, URL: http://qt.digia.com/Product/Learning/Education/course-materials/, accessed 26.6.2013
29. Jin, Q; Yan, Z; Liu, X; Ma, S; Guo, T; Tong, S, *Qt Programming to realize teaching simulation demonstration of Exploration Seismology*, 2011 International Conference on Electronic & Mechanical Engineering and Information Technology
30. W3Schools, *HTML5 Introduction*, URL: http://www.w3schools.com/html/html5_intro.asp, accessed 05.08.2013
31. PhoneGap, *About the Project*, URL: http://phonegap.com/about/, accessed 05.08.2013

32. van Donderen, C, *QCompare(HTML5, QML) A comparison of UI development technologies*, HU University of Applied Sciences Utrecht
33. Qt Project, *Qt Creator*, URL: http://qt-project.org/wiki/Category:Tools::QtCreator, accessed 05.08.2013

APPENDIXES

Appendix 1. Primary supported platforms and compilers

Platform	Qt 4.7	Qt 4.8	Qt 5.0
Windows XP 32bit	VS 2005 VS 2008 VS 2010 MinGW 4.4	VS 2008 VS 2010 MinGW 4.4	-
Windows 7 32bit	VS 2005 VS 2010 MinGW 4.4	VS 2010 MinGW 4.4	VS 2010 MinGW 4.7
Windows 7 64bit	VS2008 VS2010 MinGW 4.4	VS2008 VS2010	-
Windows 8 64bit	-	-	VS 2012
Linux (Ubuntu 11.04) 32bit	gcc 4.2	gcc 4.4	gcc 4.43
Linux (Ubuntu 11.04) 64bit	gcc 4.2	gcc 4.4	
Linux (Ubuntu 11.10) 32bit / 64bit	-	-	gcc 4.6.1
Linux (Ubuntu 12.04) 64bit	-	-	gcc 4.6.3
Linux (RHEL 6) 32bit/64bit	-	gcc 4.4	-
Mac OS X 10.5 Carbon/Cocoa	Apple compiler (gcc)	-	-
Mac OS X 10.6 Cocoa	Apple compiler (gcc)	Apple compiler (gcc)	-
Mac OS X 10.7 Cocoa	-	Apple compiler (gcc)	-

(continue)

Appendix 1. Extention

Mac OS X 10.6 Cocoa (deploy only) 32bit/64bit	-	-	Apple compiler (gcc)
Mac OS X 10.7 Cocoa 32bit/64bit	-	-	Apple compiler (clang)
Mac OS X 10.8 Cocoa 64bit	-	-	Apple compiler (clang)
Solaris 10	Sun Studio 12 (CC 5.9)	Sun Studio 12.2 (CC 5.11)	-
Linux (QWS) on Arm	gcc 4.4	gcc 4.4	-
Windows Embeded Compact 7 (ARM and x86)	-	VS 2008	-
Windows Embeded Standard 7 (x86)	-	VS 2008	-
Integrity 10.0 (ARM and x86)	-	Multi IDE 6	-
QNX Neutrino 6.5 (ARM and x86)	-	gcc 4.4	-
VxWorks 6.9.2 (DKM & RTP mode - x86 simulator, IA and ARM/ARMv7)	-	Vendor supplied gcc	-

Appendix 2. Secondary Supported Platforms and Compilers

Platform	Qt 4.7	Qt 4.8	Qt 5.0
IBM AIX 6	xlc 11	xlc 11	-
Mac OS X 10.7 Carbon	Apple compiler	Apple compiler	-
Mac OS X 10.8 Cocoa	-	Apple compiler	-
Windows XP 64bit	VS 2008 VS 2010	VS 2008 VS 2010	-
Windows XP 32bit/64bit (deploy only)	-	-	VS 2010 VS 2012 Min GW 4-7
Windows Vista 32bit/64bit	-	-	VS 2010 VS 2012 Min GW 4.7
Windows 7 32bit	-	VS 2012	VS 2012
Windows 7 64bit	VS 2005	VS 2005 VS 2012	VS 2012
Windows 8 32bit	-	VS 2010 VS 2012	VS 2010 VS 2012 MinGW 4.7
Windows 8 64bit	-	VS 2010 VS 2012	VS 2010 MinGW 4.7
Linux (RHEL 5.8)	-	-	gcc 4.4
Linux (SuSE 12.2)	-	-	gcc 4.7.1
Linux (RHEL 6.3)	-	-	gcc 4.4
Embedded Linux (x86, MIPS, PowerPC)	gcc 4.4	gcc 4.4	Arm-none-linux-gnueabi-gcc (Sourcery G++ Lite 2010q1-202) 4.4.1

(continue)

Appendix 2. Extention

Windows CE 5.0 (ARMv4i, x86, MIPS)	VS 2005 VS 2008	VS 2005 VS 2008	-
Windows CE 6.0 (ARMv4i, x86, MIPS)	VS 2005 VS 2008	VS 2005 VS 2008	-
Windows Mobile 5.0 / 6.0	VS 2005 VS 2008	VS 2005 VS 2008	-