The Practice of Porting Qt Applications to Android Platform

Kirill Kulakov
Petrozavodsk State University
Petrozavodsk, Russia
kulakov@cs.karelia.ru

Abstract

Nowadays each large company prefer to use own software platform in mobile devices. Mostly platforms based on Linux, Windows and Java and use own native software development kit to create applications. Thereby the problem of porting application from one platform to another is urgent. One of the solutions is to use cross-platform frameworks.

In this paper we describe the practice of porting various applications based on cross-platform Qt/QML framework to Android platform. It used Necessitas project, provided by Bogdan Varta and Qt community. Each application can be download from application store for Android, Harmattan and Symbian platforms.

Index Terms: Qt, Android, Necessitas.

I. INTRODUCTION

The mobile platforms are more and more popular today. There are many peoples have mobile devices. It's uses devices every day for work and play. Each device connected to some applications store and user can buy and download any available application. Some platforms support in-app pushcare services and user can buy any additional packages or application resources. Therefore mobile applications are one of important area of software developing.

Nowadays we have several popular mobile platforms, like Android [1], iOS [2], Symbian [3] and Windows [4]. The core of most platforms based on Linux or Windows kernels but they have own middle environment. Each platform use own native programming language, compiler and provide own software development kit. In some cases two versions of one platform has different configuration options or constants and software developer must know it.

Let's view the following case. The software company produces the idea for new application and they want to sell implementation it to many users. If they use native platforms, then they need to implement, for example, three different applications – for Android, Symbian and iOS. So, we need three time more resources for developing and supporting.

The one of solutions for optimize resource consumption is to use cross-platform technologies. The one of this is Qt [5]. It's based on C++ language and have official ports for several mobile platforms, like mobile Linux and Symbian. Also we have community ports, for example for iOS and Android. The main advantage is that Qt integrates to platform in middle or low level. User don't need to run additional software or virtual machine to run Qt application. Also Qt supports themes and styles and Qt applications looks like native applications without modification of source code.

In this paper we describe our experience of porting various Qt applications between Symbian, Harmattan [6] and Android platforms. Qt applications are used QML and Qt components. We use Necessitas framework [7] which are provided by Qt Community. The idea of Necessitas is to compile application as Android library and provide Qt libraries in separate package called Ministro.

The rest of the paper is organized as follows: in Section II we describe the common algorithm for porting Qt application to Android platform. Section III contains results of porting our applications and our solutions for founded problems. Section IV concludes the paper.

This research is a part of grant KA179 "Complex development of regional cooperation in the field of open ICT innovations" of Karelia ENPI programme, which is co-funded by the European Union, the Russian Federation and the Republic of Finland. The article was published with financial support from the Strategic Development Program of Petrozavodsk State University.

II. PORTING APPLICATION

Necessitas SDK based on original Qt SDK and has Desktop and Android as target platforms. SDK includes Android NDK and Android tools. Configuration of communication between Android NDK and Qt SDK organized as separate form in preferences menu. The project configuration for Android part presents in Run part of common project configuration. The common scheme of application running is show in Fig. 1.

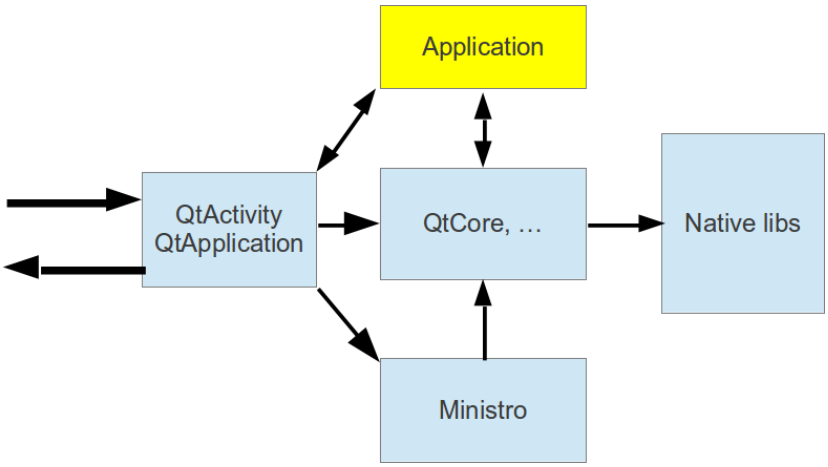

Fig. 1. Qt application running scenario

Communication between user and application contains in Java classes QtActivity and QtApplication. At start they calls Ministro service. Ministro checks requested Qt libraries and download missing or updated items. Then QtApplication loads needed Qt libraries and run application. All communications between native platform components and application are occurs in Qt libraries.

Lets assume that we have a source code of Qt application for Symbian platform and it uses Qt Quick and Qt components. We need to process the following sequence:

- create extensible user interface and functionality;
- prepare resources to package;
- configure project environment;

– add necessary components to package.

Usually developer creates user interface for selected device. For example, he/she strongly fix screen size. When we port this application to android platform, we have several devices with various screen resolution. So, the main change is to remove screen size restriction and add response to the size of screen. The another problem that each device has own configuration of hardware options and user must decide: check hardware availability or restrict of available devices.

The package contains specified Android data, like manifest, logo and icon, and used libraries. The system uses these libraries in package to start application and doesn't have any API to access to other package files. Therefore if project use any data files, e.g. images, these files must be include to project as resources or use Android "assets".

Each project must specify own icon and package name. Also the project must specify used Qt libraries and components. It's needed because when application starts Ministro checks requested and available libraries and download missing libraries.

When application uses Qt components the developer must include information about it to the project. Ministro doesn't have these libraries, so user must compile these libraries and add it to list of installed files.

III. RESULTS OF PORTING

We are porting applications “Liquid”, “Loader”, “Shariks” and “Mushrooms” to Android platform [9]. “Liquid” and “Shariks” are action games with using accelerometers, “Loader” uses box2D technology and “Mushrooms” is a reference application. With this set of applications we try to cover most popular categories of applications.

All applications are tested in Android simulator and various devices with Android 2.3-4.0. Unfortunately, some applications didn't run in some devices but we found no dependence. During porting we got the several problems.

During porting “Shariks” game we found that Symbian and Android platforms has different directions of axes. To solve this problem with minimal code changes we adds context property with value according to the platform and reverse X-axis for Android platform. Also we cannot run application with OpenGL support. We solve it by disabling OpenGL.

The another problem which we a found in most application is that the code for disabling screen rotation don't work in Android platform. In the Symbian we write the following line in main function:

```
viewer.setOrientation(QmlApplicationViewer::ScreenOrientationLockLandscape);
```

For Android we solve it by adding the following attribute to “activity” section in AndroidManifest.xml

```
android:screenOrientation="landscape"
```

In applications with splash screen we get small size of screen (without top panel space) because Qt application didn't run as normal application and don't have full screen at startup. The one of solution is to write special class ScreenSizeLauncher which are connects to signal workAreaResized() in QApplication.desktop() during application startup. When screen size are changed the instance of ScreenSizeLauncher sets context property with current screen size.

In “Mushrooms” application the one of problems was cell size. In Symbian platform we use two sizes for portrait and landscape:

cellWidth: (parent.width / parent.height > 1.5) ? 213 : 120;

This formula didn't work in Android because devices has various screen size. During testing we got the following formula:

cellWidth: (parent.width / parent.height > 1.5) ? 213 : (parent.width > 400 ? 120 : Math.ceil(parent.width / 3));

We test this solution on Android smartphones and tablets.

IV. Conclusion

In this paper we describe or experience of porting Qt application to Android platform. All applications are published to Nokia Store [8] and Android market [9]. They are shows the future of cross-platform technologies. Today main problem of Qt in Android is reduced integration to platform and huge size of Qt libraries.

Acknowledgment

The author would like to thanks Iurii A. Bogoiavlenskii for his feedback and expertise.

References

[1] "Android". [Online]. Available: http://www.android.com/
[2] "Apple -iOS". [Online]. Available: http://www.apple.com/en/ios/
[3] "Symbian Foundation". [Online]. Available: http://licensing.symbian.org/
[4] "Windows Phone". [Online]. Available: http://www.windowsphone.com/
[5] "Qt Developer Network". [Online]. Available: http://qt-project.org/
[6] "Develop for the Nokia N9". [Online]. Availabe: http://www.developer.nokia.com/Devices/MeeGo/
[7] "Necessitas". [Online]. Available: http://community.kde.org/Necessitas
[8] "Nokia store". [Online]. Available: http://store.ovi.com/publisher/FRUCT/
[9] "Google Play". [Online]. Available: https://play.google.com/store/apps/developer?id=FRUCT