Forum Nokia

Quickstart
Qt for Windows, Symbian and Maemo / MeeGo

Andreas Jakl

Senior Technical Consultant
Forum Nokia

10 January, 2011
v2.0.8

Task

- **Run a *Hello World* application on**
  - Windows
  - Maemo 5 device
  - Symbian device

Windows 7

Maemo 5

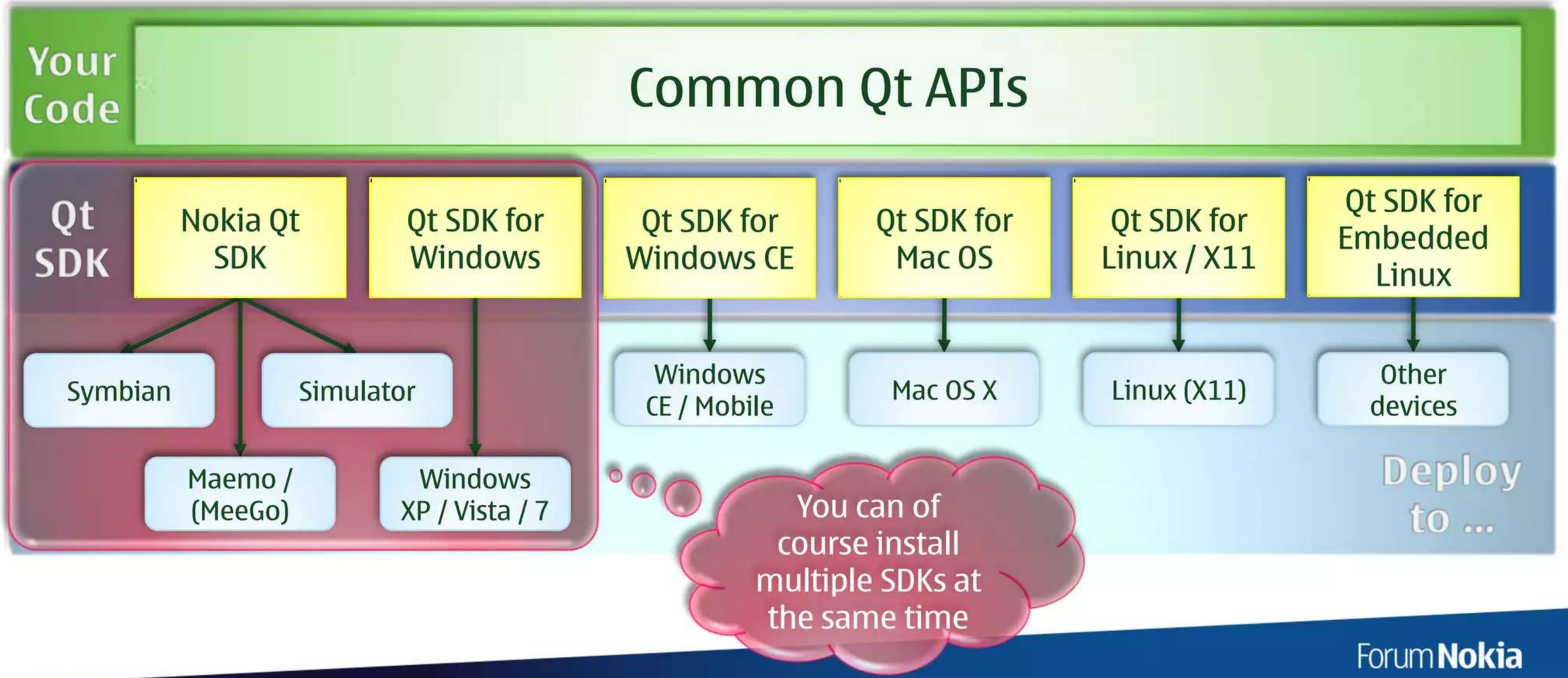
Symbian^1 (S60 5th Edition)

Prerequisites

- **Requirements for this tutorial:**
  - Windows XP / 7
  - Development on Linux / Mac OS similar (partly in beta right now)!
- **General hints:**
  - Install all tools to the same drive (e.g., C:\)
  - Do not use network drives
  - Use default installation paths. Be wary of paths that contain spaces / special characters

General Qt
(plus: for Windows)

Qt SDKs from qt.nokia.com



Nokia Qt SDK

- **One-Click installation:**
  - Development tools
  - Build for and debug in real devices
    - Symbian
    - MeeGo / Maemo
  - Test on host PC
    - Simulator
  - Qt Mobility
- **No extra device SDKs required anymore**

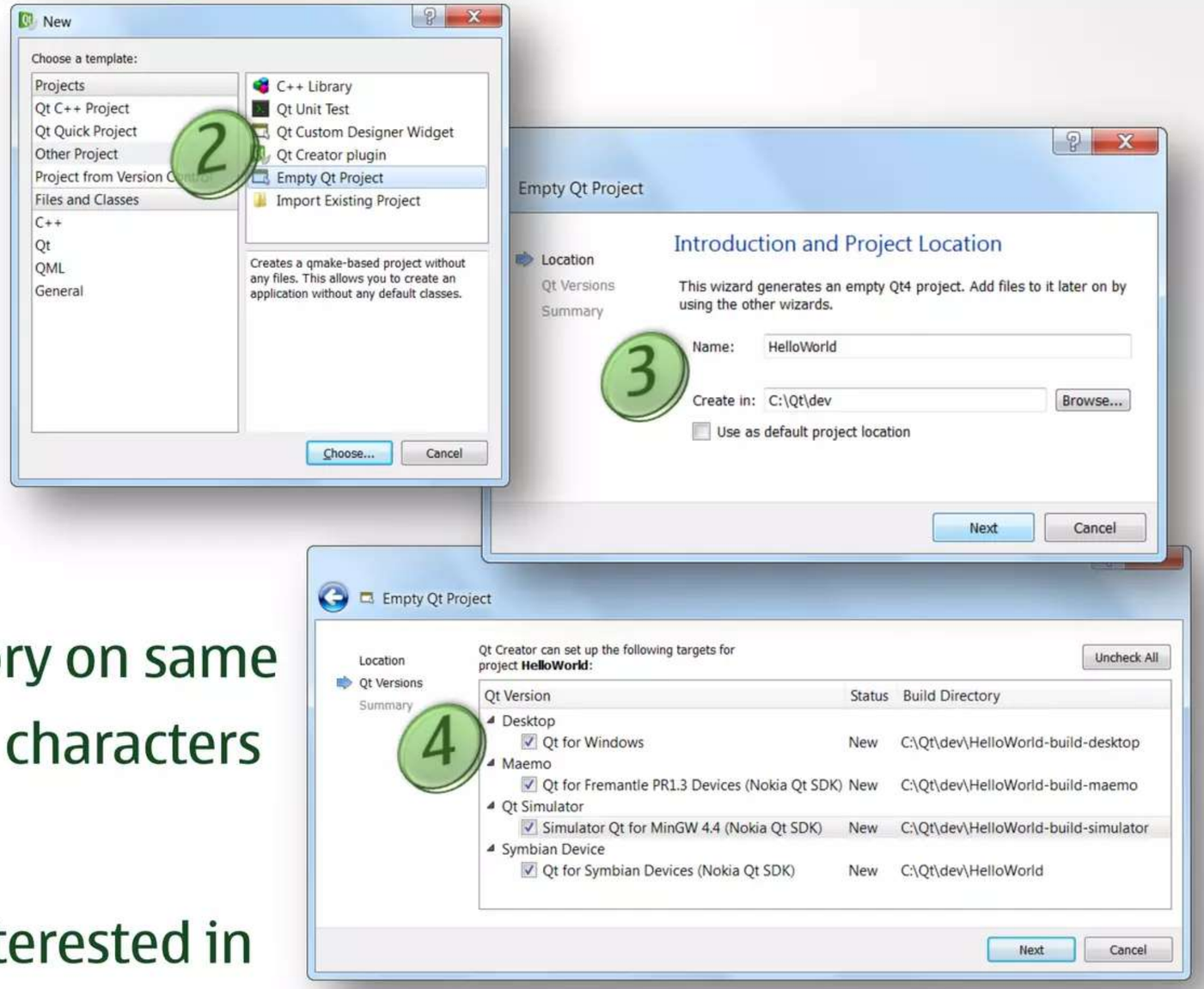
Installation

- **Install Nokia Qt SDK**
  - For mobile development
  - Works on Windows, Mac & Linux
- **Install Qt SDK for Windows**
  - **Optional**
  - For desktop development

http://www.forum.nokia.com/Qt
http://qt.nokia.com/downloads

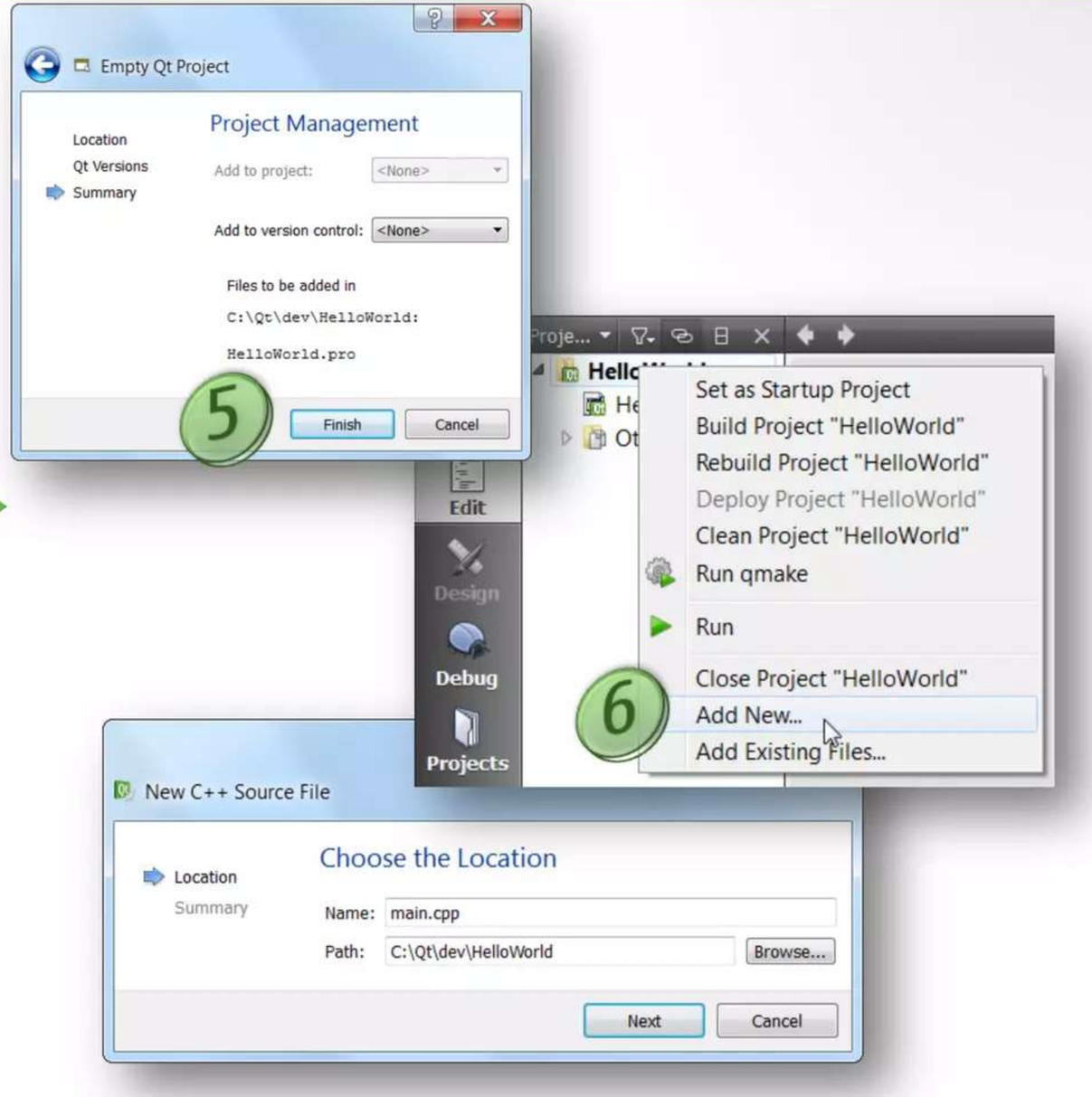
New Qt Project

1. File → New File or Project...
2. Empty Qt Project
3. Project properties
   - **Name:** Hello World
   - **Create in:** workspace directory on same drive as tools, without space characters
4. Qt versions
   - **Select all targets** you are interested in



Main Source File

5. **Project management**
   - Accept defaults
6. **Right-click on project → Add New... → C++ source file**
   - **Name:** main.cpp
   - **Path:** project path (default)
   - Accept **defaults** on next page



Hello World – Source Code

7 **Write following code into empty main.cpp:**

main.cpp

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QPushButton helloButton("Hello World");
    helloButton.resize(150, 50);
    helloButton.show();

    return app.exec();
}
```

Run the Application

8. Make sure “Desktop” is current build target
9. Click on play arrow

Hello World – Components

- **QApplication**
  - One per GUI app
  - Manages app-wide resources (default font, cursor, ...)
- **QPushButton**
  - Default widget, based on QWidget
  - Can process user input and draw graphics

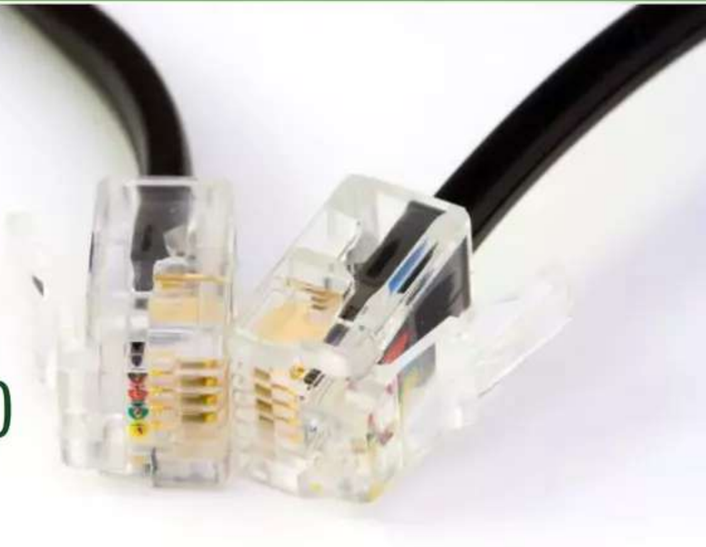
Extend Hello World: Quit

- **Add functionality to exit the Hello World example:**

```
QObject::connect(&helloButton, SIGNAL(clicked()),
                 &app, SLOT(quit()));
```

  - Button emits clicked() signal
  - Connected to QApplication::quit()

Signals & Slots



- **Signal**
  - **Emitted when a particular event occurs** (e.g., clicked())
  - Qt widgets: predefined signals
  - Also create your own signals
- **Slot**
  - **Function called in response to a signal**
  - Qt widgets: predefined slots (e.g., quit())
  - Also create your own slots
- **Connection signals ↔ slots established by developer, handled by Qt framework**

Nokia Qt SDK: Simulator

- **Efficient Testing**
  - Quick launch
  - Scripting possibilities
    - Using JavaScript
  - Simulate Qt Mobility Project features
    - Location, contacts, etc.
  - Simulate phone events
    - Battery, messages, etc.
  - Skins for different platforms / form-factors
    - Resolutions, orientation, etc.

Launching the Qt Simulator

- **Execute Hello World on Simulator**
  - Select **Qt Simulator** target
  - Click on Play

Qt for Maemo / MeeGo

Qt for Maemo

- **Development options**
  - **Full Linux Environment (with Scratchbox)**
    - Install Linux on your PC (e.g., Ubuntu)
    - Guide: http://wiki.maemo.org/Documentation/Maemo_5_Final_SDK_Installation
    - Or use VMware on Windows / Mac (image is a bit outdated)
      - http://maemovmware.garage.maemo.org/
  - **Nokia Qt SDK (based on MADDE)**
    - Works on Windows, Linux, Mac OS X
    - **Full Qt support out of the box!**

MADDE

- **MADDE?**
  - **Comes with Nokia Qt SDK**
  - **M**aemo **A**pplication **D**evelopment and **D**ebugging **E**nvironment
  - Toolchain that supports compiling and deploying MeeGo applications without setting up an own Linux environment
  - **Cross-Compilation**
  - http://wiki.maemo.org/MADDE

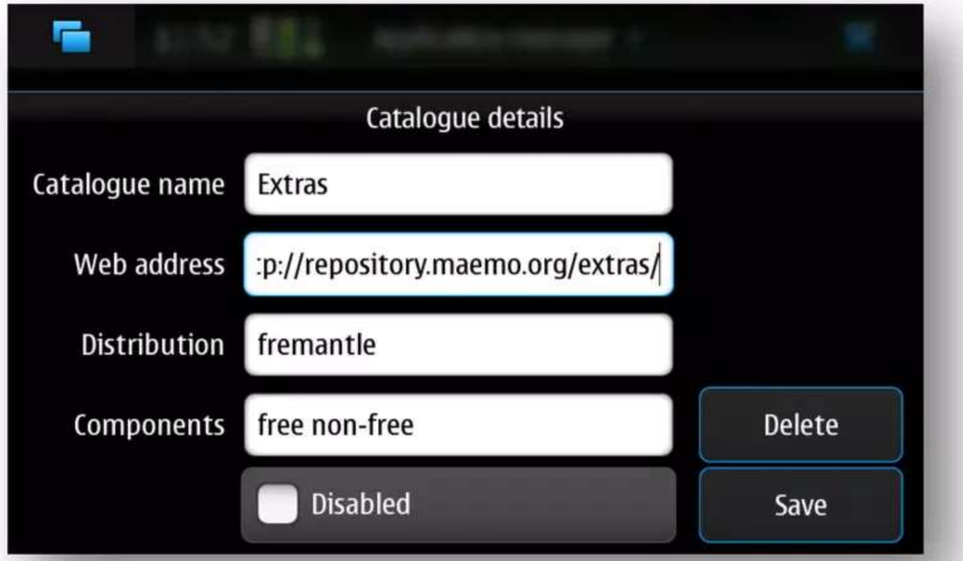
Maemo → MeeGo

- **MeeGo fully compatible to Qt & Qt Creator**
  - As a 3rd party developer, most changes to the OS are irrelevant to you if you use Qt
  - Start developing now on Maemo 5!

MeeGo

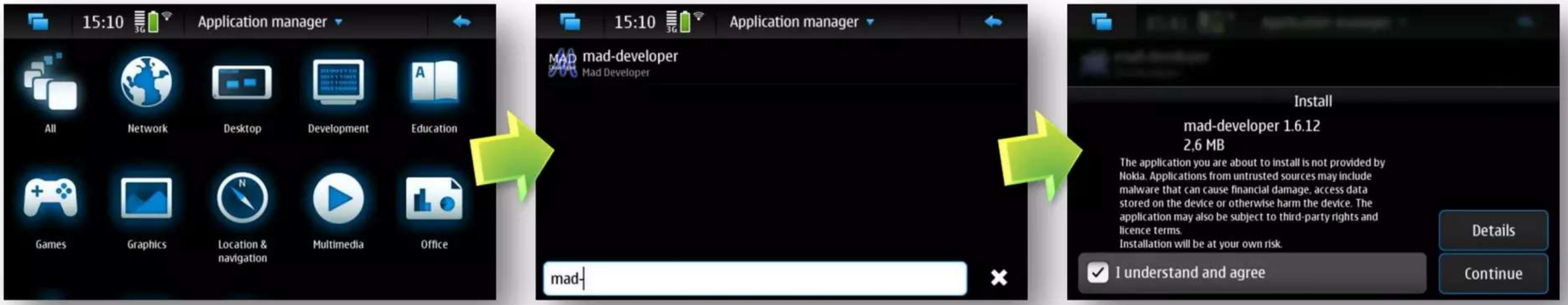
Setting up the Device

- **Configure your N900**
  - To enable direct deployment and debugging from Qt Creator
  - Requires installing **MAD Developer tools**
- **Add developer repositories to download tools**
  - Open the **App manager** on the N900
  - Menu → Application catalogues → New
    - Catalogue name: Extras
    - Web address: http://repository.maemo.org/extras/
    - Distribution: fremantle
    - Components: free non-free



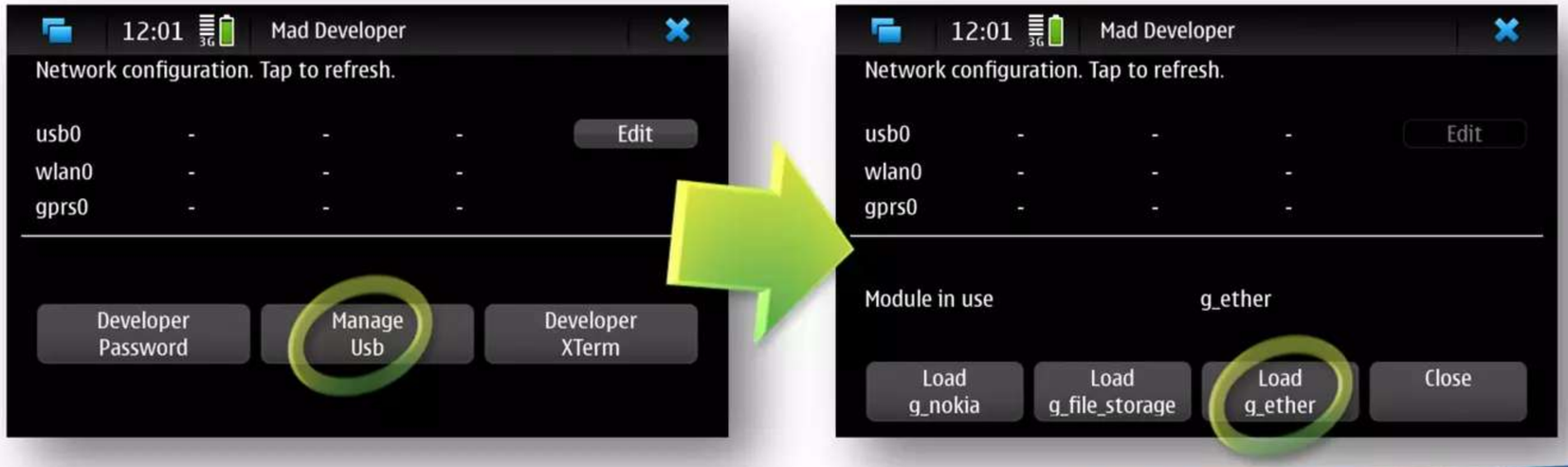
Mad Developer

- **Download and install**
  - **mad-developer** package
  - In Application manager, click on Download → All → start typing “mad-” on hardware keyboard to filter list and search for apps



Connect the Device

- **Select connection mode**
  - Start the “Mad Developer” application on the N900
  - Click on “Manage USB” → **“Load g_ether”**
    - In any connection mode prompts that pop up, just close the dialog by clicking outside of it
  - Make sure “Module in use” is “g_ether” and click close



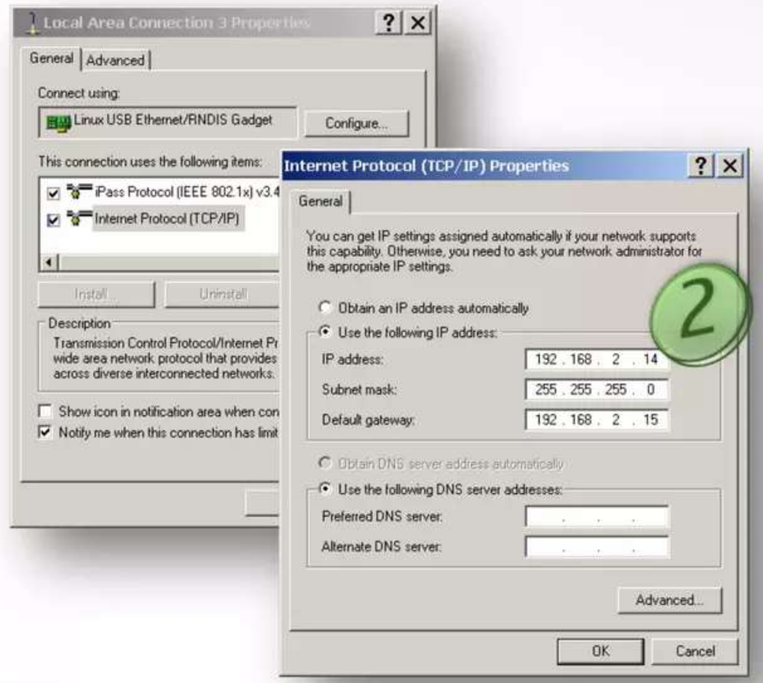
Connect the Device

- **Configure USB / Ethernet connection**
  - Click on **"Edit"** and ensure following setup
    - IP Address: 192.168.2.15
    - Netmask: 255.255.255.0
  - Click on **"Configure"** to apply

Configure the PC

Windows 7: Network and Sharing Center → Change adapter settings

- **Set a static IP for the USB Ethernet connection to the N900**
  - Open the **Network Connections**
  - 1 – Right-click on the **Local Area Connection** of the "Linux USB Ethernet/RNDIS Gadget" device → **Properties**
  - Select "**Internet Protocol** (TCP/IP)" → Properties
  - 2 – Enter a **static IP** setting
    - IP address: 192.168.2.14
    - 255.255.255.0
    - Default gateway: 192.168.2.15 (or leave empty)

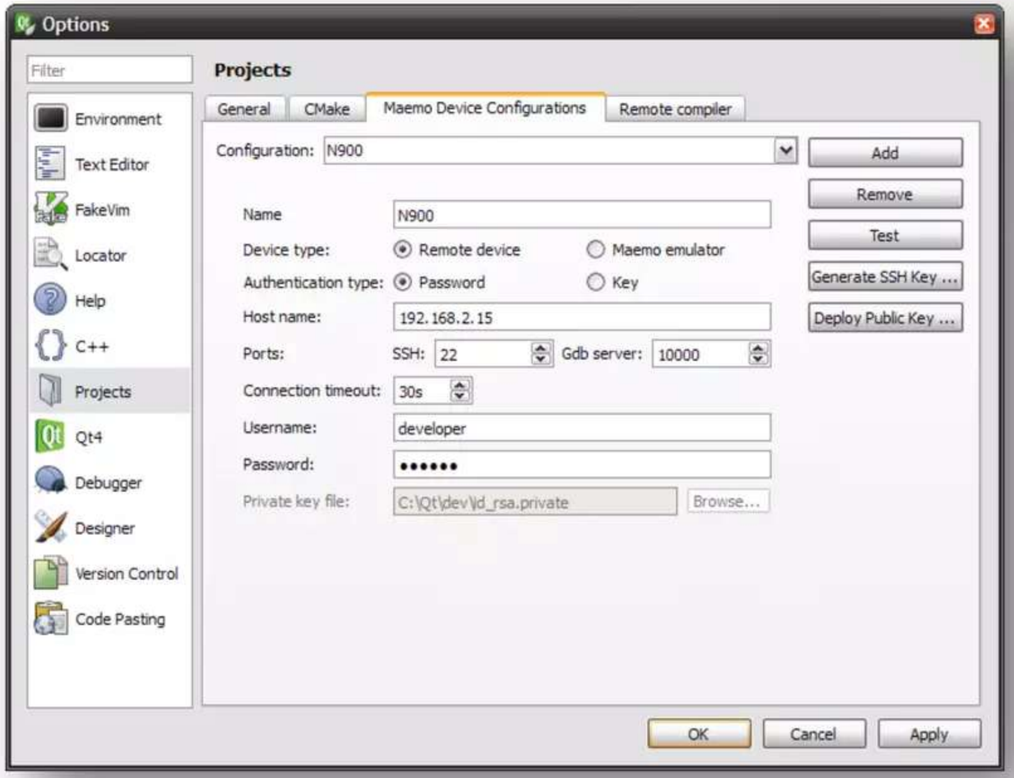


1

Name	Type	Status	Device Name
LAN or High-Speed Internet			
Local Area Connection 3	LAN or High-Speed Internet	Connected	Linux USB Ethernet/RNDIS Gadget
Local Area Connection	LAN or High-Speed Internet	Connected	Intel(R) 82567LM Gigabit Network Connection
1394 Connection	LAN or High-Speed Internet	Connected	1394 Net Adapter
Wireless Network Con...	LAN or High-Speed Internet	Not connected	Intel(R) WiFi Link 5300 AGN

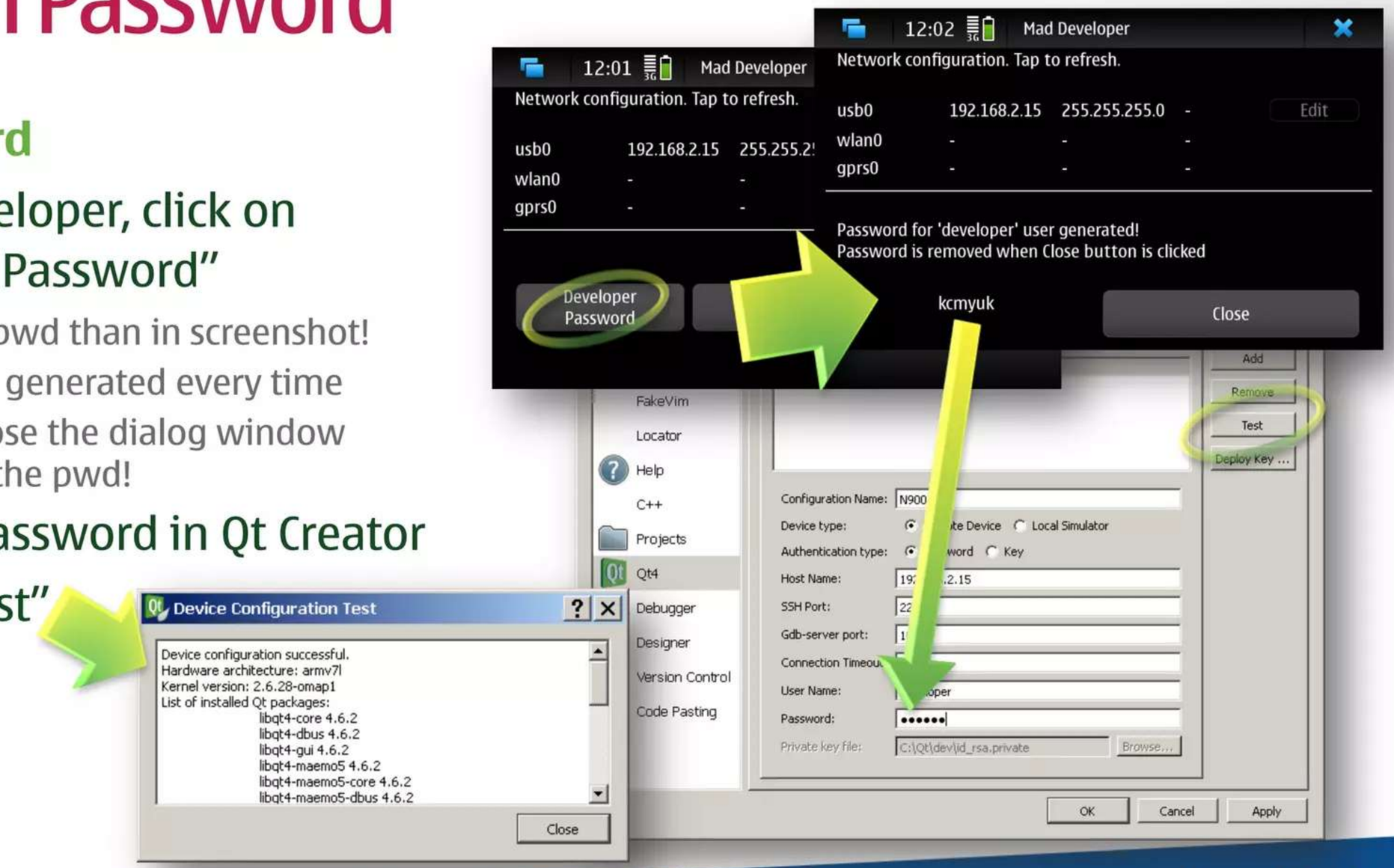
Connect Qt Creator to Maemo Device

- **Test the connection**
  - Start **Qt Creator**
  - Options → Qt4 → **Maemo Devices**
  - Add a device, choose a configuration name (e.g., “N900”)
  - Ensure **setup** is as shown in screenshot
    - Authentication type: Password
    - Host Name: 192.168.2.15



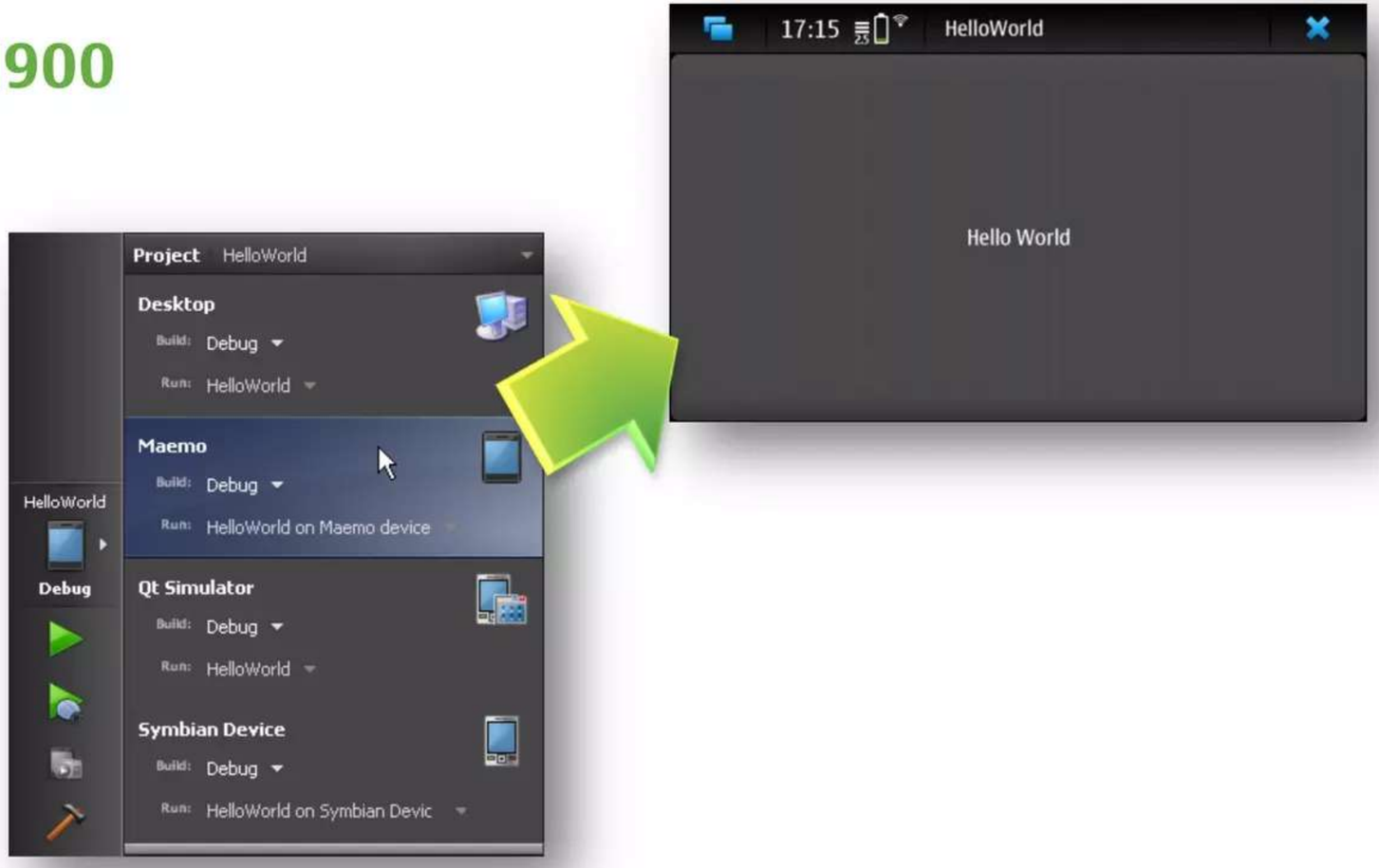
Connection Password

- **Create password**
  - In Mad Developer, click on "Developer Password"
    - Different pwd than in screenshot!
    - New pwd generated every time
    - Do not close the dialog window showing the pwd!
  - Enter the password in Qt Creator
  - Click on "Test"



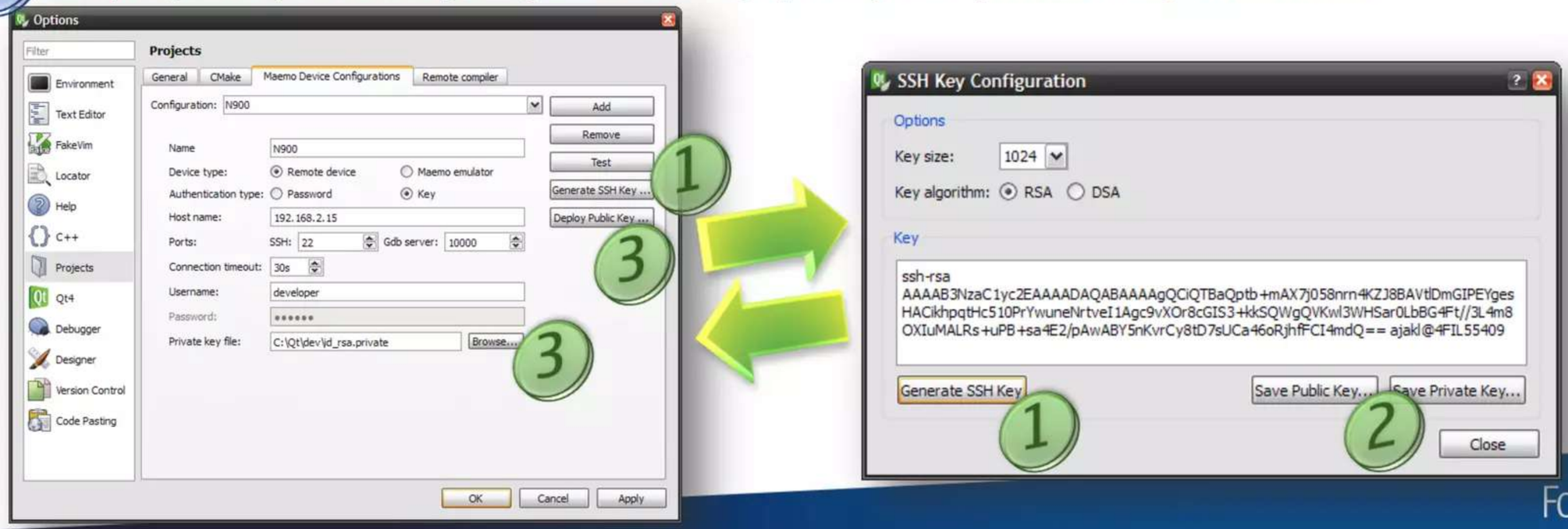
Deploy to N900

- **Execute Hello World on N900**
  - Select **Maemo** target
  - Click on Play



Faster Execution: SSH Key

- **Create SSH keys once instead of defining a new password for every connection**
  - Make sure your active connection with the password is working
  1. Generate SSH Key ... → Generate SSH Key
  2. Save both the public and the private key to your computer
  3. Deploy the public key to your device, specify the private key on the PC



Optification

The "Storage Usage" application gives a graphical overview of memory usage and also lists the memory by package

- **Storage memory on the N900**
  - **256 MB NAND memory:** Fast, but high power requirements → not so big by design. For bootloader, kernel and root.
  - **32 GB eMMC:** large, but slower.
    - /home: ~ 2 GB
    - /home/usr/MyDocs: ~ 29 GB
- **Memory usage rules for applications**
  - **Put large files on eMMC memory, not into small rootfs → "Optification"**
  - Required for **Ovi Store** submission: http://wiki.maemo.org/Ovi_Store_publishing
  - **Documentation:** http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Packaging,_Deploying_and_Distributing/Installing_under_opt_and_MyDocs

Qt for Symbian

Symbian & Qt

- **Qt for Symbian:**
  - Compatible to **S60 3.1+** (Nokia N95, E71)
  - Nokia devices platform versions:
    http://www.forum.nokia.com/devices/

Nokia N95 (2007)

** Qt can be installed on all compatible devices. Not all devices are enabled in the Smart Installer, which is required for deploying Qt apps through the Ovi Store. More devices are constantly added as compatibility tests are being performed. Current Smart Installer device deployment support: http://www.forum.nokia.com/Distribute/Packaging_and_signing.xhtml*

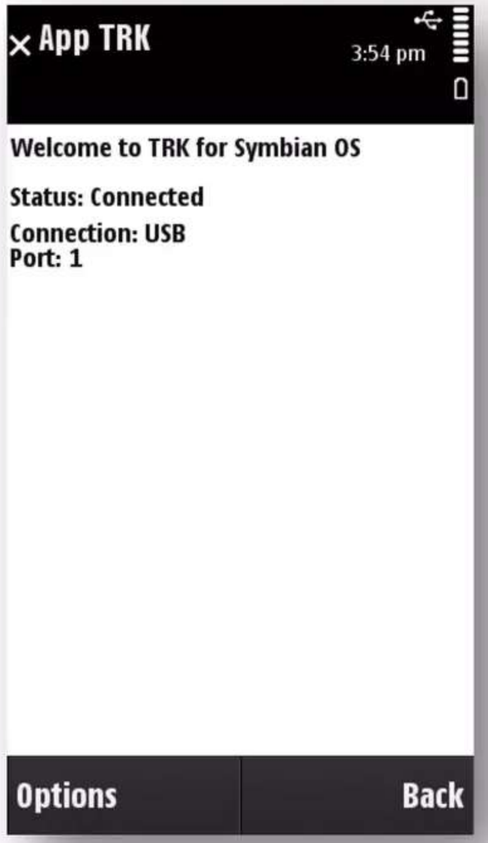
Install Qt to the Device < Symbian^3

- **.sis-file: Symbian installation file**
- **Qt installation files in your Qt/Symbian SDK dir**
  - C:\NokiaQtSDK\Symbian\sis**qt_installer.sis**: Installs Qt libraries
  - C:\NokiaQtSDK\Symbian\sis**qtmobility.sis**: Contains additional Qt Mobility APIs *
  - C:\NokiaQtSDK\Symbian\sis**fluidlauncher.sis**: Installs demos
- **Installation to mass memory. Choose one method:**
  1. Send through Bluetooth to device
  2. Use Ovi Suite, connect your device and double-click .sis-file
  3. Copy .sis-file to device / SD card and start installation through file manager on device
- **Restart the device**

* Note: Symbian^3 requires a different Qt Mobility installation file, which is part of the Nokia Qt SDK 1.0.1+.

Automatic Deployment & On-Device Debug

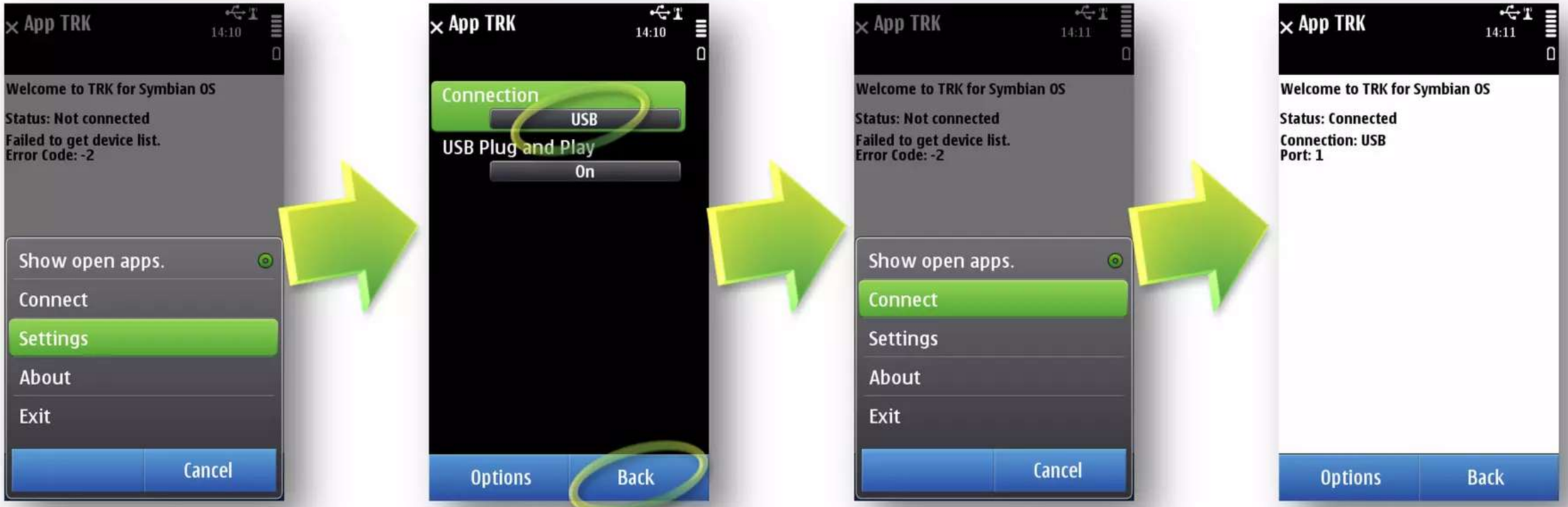
- **Requires debug agent running on device: App TRK**
- **Install:**
  - C:\NokiaQtSDK\Symbian\sis**s60_5_0_app_trk_3_*x*_*x*.sisx** for S60 5.0. Choose appropriate TRK for your phone.
  - **Install to phone memory** (C:\)
- **Run the TRK on the device**
  - Setup connection using Bluetooth or USB cable
  - **Recommended:** USB



TRK contained in Nokia Qt SDK. Online download: http://tools.ext.nokia.com/trk/

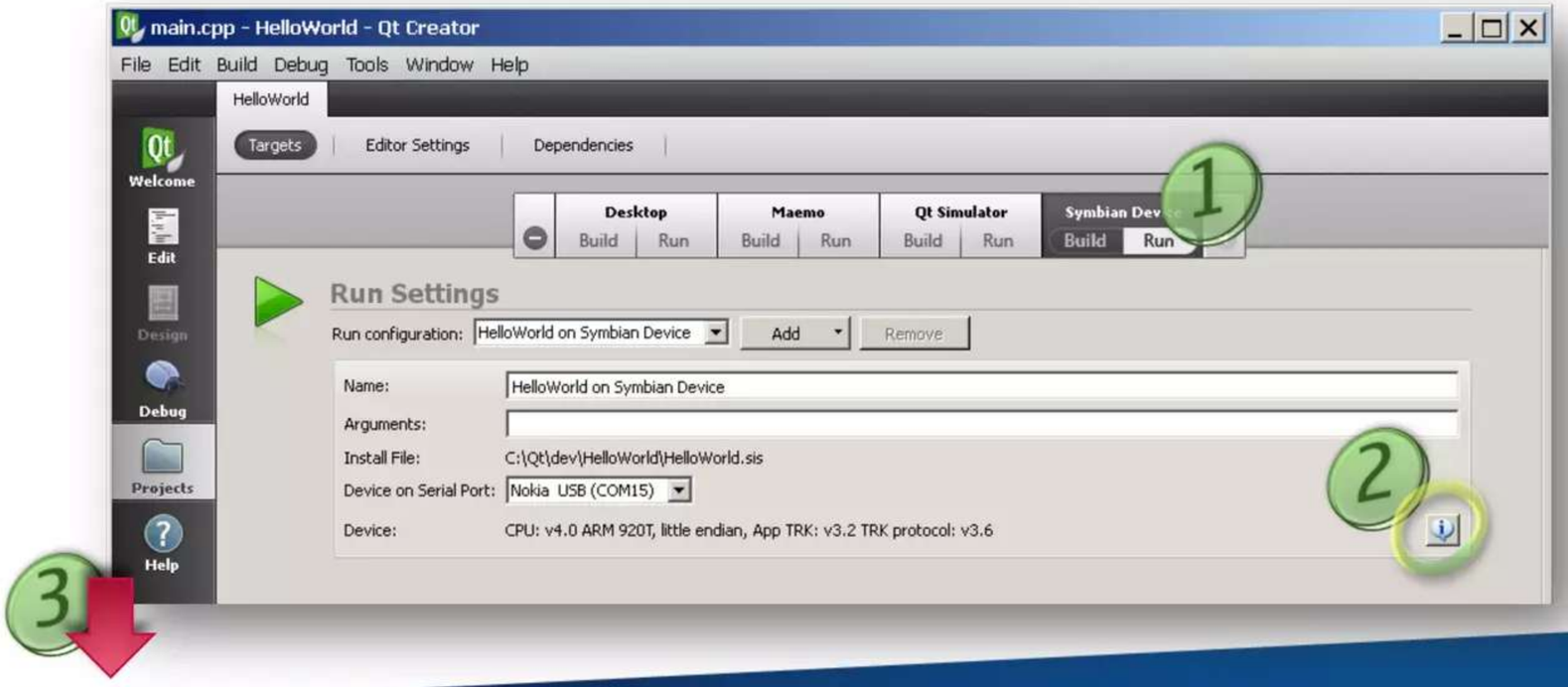
TRK: USB Connection

- **Debugging through USB is faster and easier**
  - Very first start of TRK on device: cancel Bluetooth connection attempt

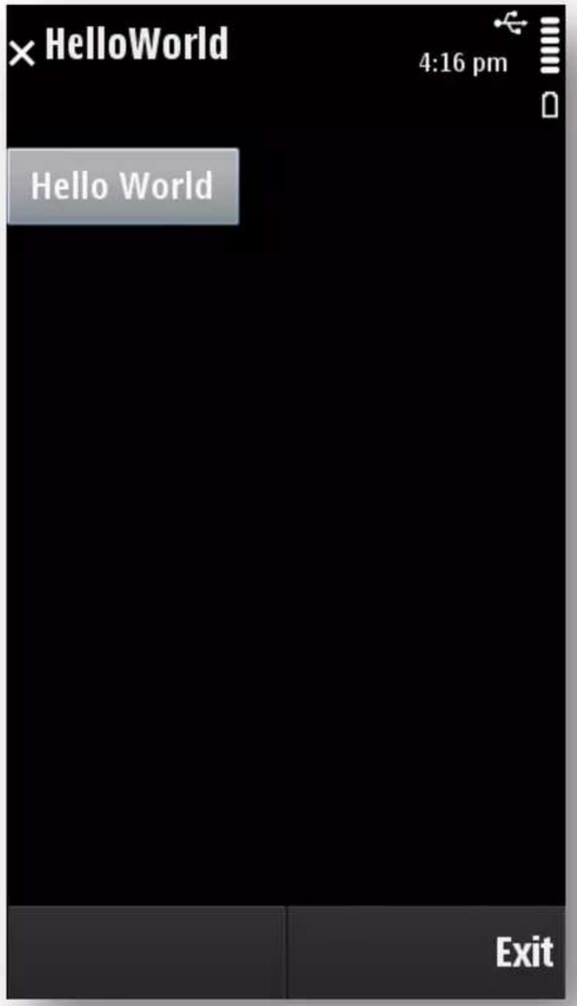


Testing the Device Connection

1. Go to the Run Settings tab of the Symbian Device
2. Click on the **i** symbol to check the active TRK connection
3. Press play to deploy and run



Hello World on Symbian



Smart Installer

http://wiki.forum.nokia.com/index.php/Nokia_Smart_Installer_for_Symbian

Signing & Certificates

- **Self-Signed application (default):**
  - Security warning during installation
  - No access to restricted features (e.g., powering off the device)
- **Reasons for signing:**
  - **Prevent sabotage** of installation files (.sis)
  - **Identification** of the software developer
  - Access to APIs (**Capabilities**) for sensitive features (calendar, location, etc.)

Open Signed (Developer Certificate)

Self-Signed app installation

Symbian Distribution

- **Now free through Ovi Publish program**
  - **Sign up:** http://www.forum.nokia.com/Distribute/Packaging_and_signing.xhtml
  - Get a **test certificate for development** (if you require restricted capabilities in your app)
  - Thoroughly **test your app:** http://tiny.symbian.org/testcriteria
  - **Submit finished app to the Ovi Store for free**

Traditional, non-free signing (still available)

- **Publisher ID from TrustCenter**
  - $200 / year, only for companies
  - Allows creating developer certificate
  - http://www.trustcenter.de/en/products/tc_publisher_id_for_symbian.htm
- **Sign your app through:**
  - **Express Signed: instant, €10**
  - (or Certified Signed: external test house, €185+)
  - https://www.symbiansigned.com/

Good to Know

Taking Screenshots

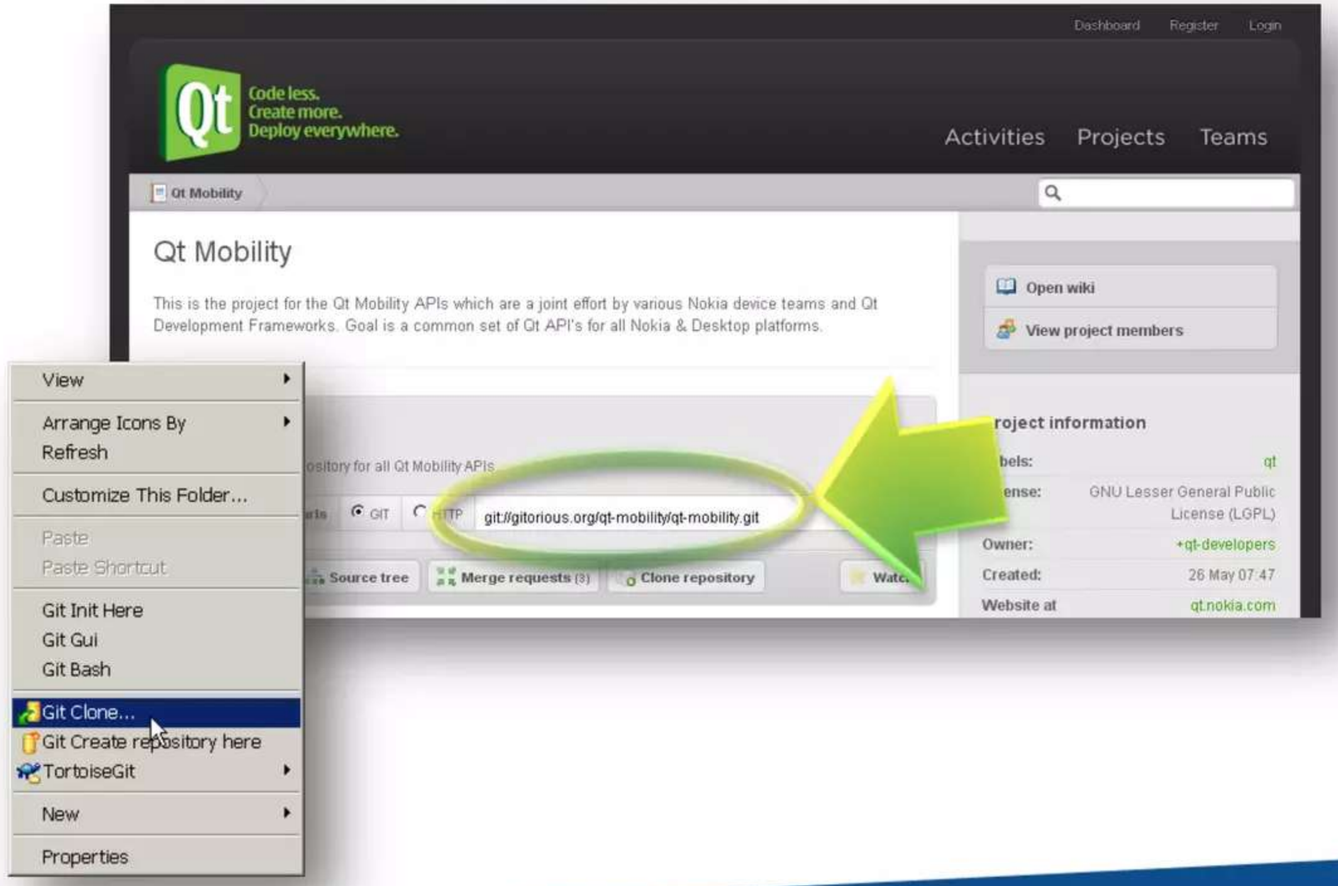
- **Maemo**
  - **Ctrl + Shift + P:** Saves to /home/user/MyDocs/.images/Screenshots
- **Symbian**
  - Download **Best Screen Snap** for S60 (Freeware)
    http://www.smartphoneware.com/screen_snap-for-s60-5th-edition-download.php
  - Set image type to “BMP – true color” for maximum quality

Working with GIT from Windows

- **Use bleeding edge code?**
  - Qt developed as open source
  - You can download latest development source from GIT repositories
  - **Warning:** not for beginners! Needs manual compilation.
    Commercial ARM compiler required for compiling Qt for Symbian.
- **Install Windows client software**
  - **msysgit:** http://code.google.com/p/msysgit/
  - **TortoiseGit:** http://code.google.com/p/tortoisegit/

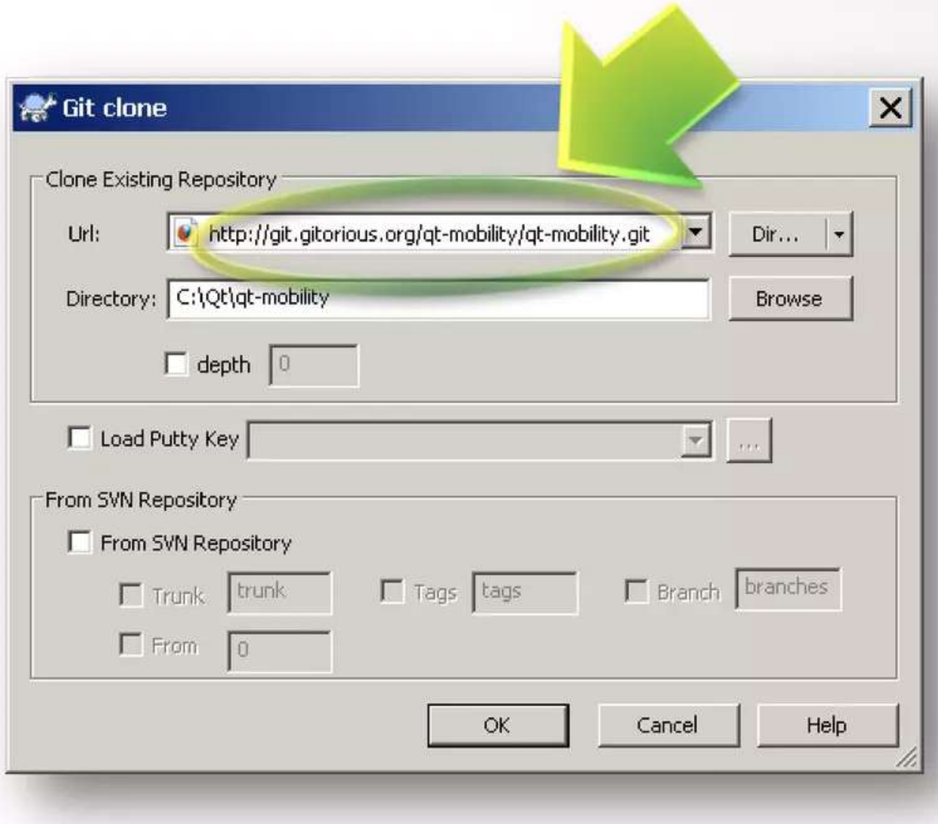
Downloading from GIT

- **Search repository you want to download locally (clone)**
  - http://qt.gitorious.org/
  - Get repository URL
- **Clone the repository**
  - In Windows Explorer, go to parent of target directory (e.g., `C:\Qt\`)
  - Right-click → Git Clone...



Downloading from GIT

- **Configure download**
  - Enter repository URL
  - Make sure your firewall / proxy doesn't block download



Troubleshooting

Troubleshooting: Qt Not Found

Platform: Windows

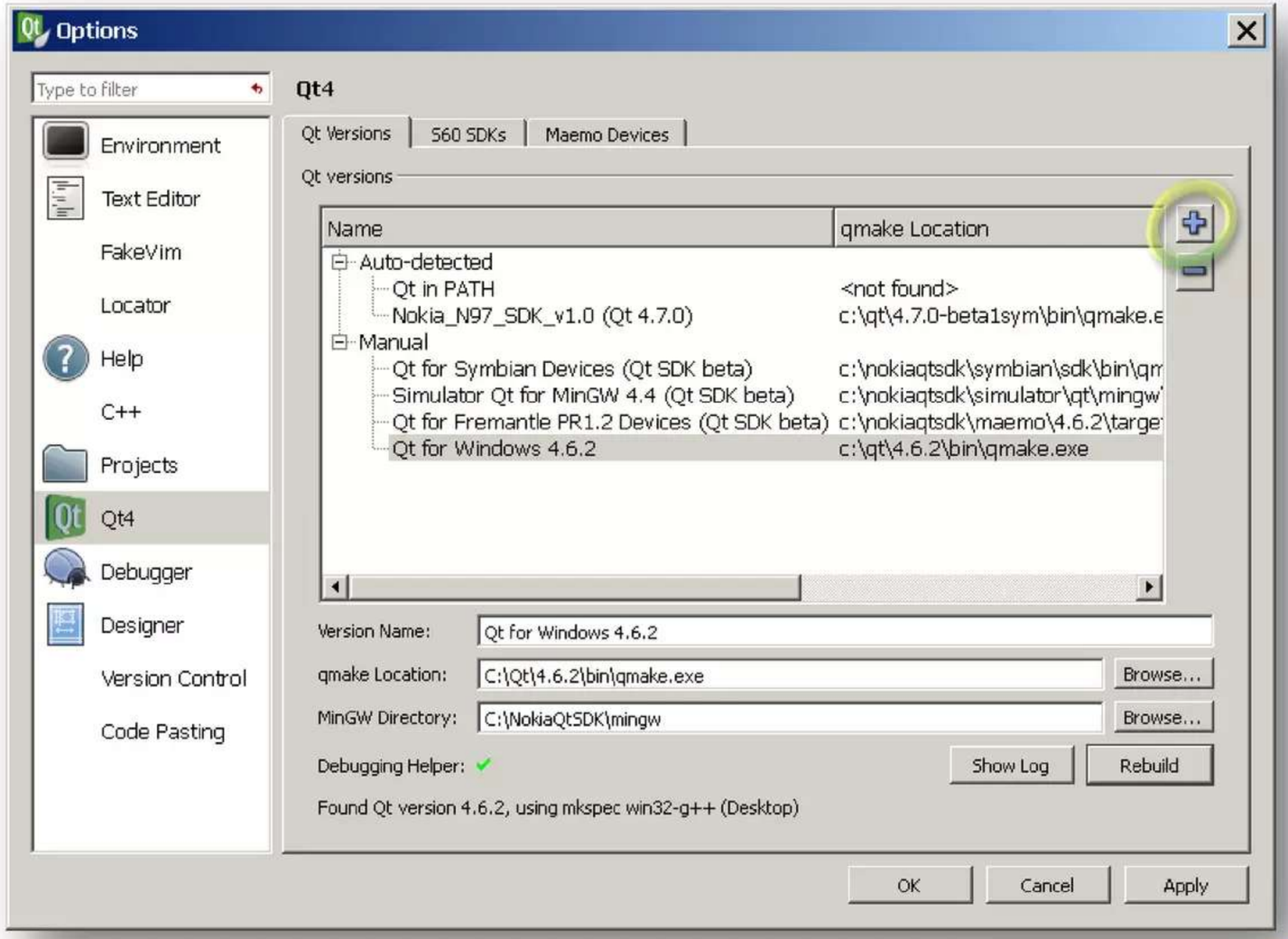
- **Situation**
  - Full Qt for Windows SDK is installed, created a Qt Creator project
- **Problem**
  - Error when compiling, similar to:

```
No valid Qt version set. Set one in Tools/Options
Error while building project GuiTest
When executing build step 'QMake'
Canceled build.
```

- **Solution**
  - Go to Tools → Options → Qt4. Click on the "+" button to define a manual Qt version. Set the QMake and MinGW location to your SDK dirs (see screenshot on next slide)

Troubleshooting: Qt Not Found

Platform: Windows



Solution:
Add Qt manually

Troubleshooting: Network Drives

Platform: Windows

- **Situation**
  - Created / opened a project on a network drive on Windows
- **Problem**
  - Error when compiling, similar to:

```
Error processing project file: //fshome/.../TestProject.pro
Exited with code 3.
Error while building project TestProject
When executing build step 'QMake'
```

- **Solution**
  - Create your project on a local drive, not a network drive.
    If a network drive is required, make sure it is accessed through a drive letter instead of the //.../-path

Troubleshooting: vtable references

Platform: Generic

- **Situation**
  - Added metaobject functionality to existing plain C++ class (signals / slots, added derivation from QObject, added Q_OBJECT macro)
- **Problem**
  - Error when compiling, similar to:

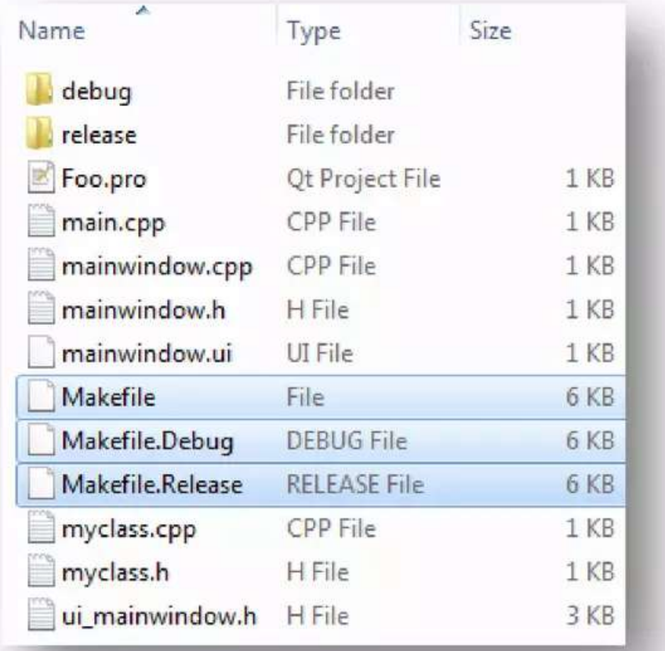
```
debug/myclass.o: In function `MyClass':
C:\Qt\workspace\Foo/myclass.cpp:3: undefined reference to `vtable for MyClass'
C:\Qt\workspace\Foo/myclass.cpp:3: undefined reference to `vtable for MyClass'
collect2: ld returned 1 exit status
mingw32-make[1]: *** [debug\Foo.exe] Error 1
mingw32-make: *** [debug] Error 2
Exited with code 2.
```

  - Build → Clean All doesn't help

Troubleshooting: vtable references II

Platform: Generic

- **Solution**
  - Qt Creator doesn't necessarily recognize changes, as your class now needs to be processed using the meta object compiler (moc) during the compilation process.
  - **Elegant solution:** in Qt Creator, choose: Build → Run qmake
  - **Less elegant solution:** Directly delete makefiles from explorer / terminal and compile again. This ensures that all makefiles are re-generated.



Troubleshooting: Qt DLLs

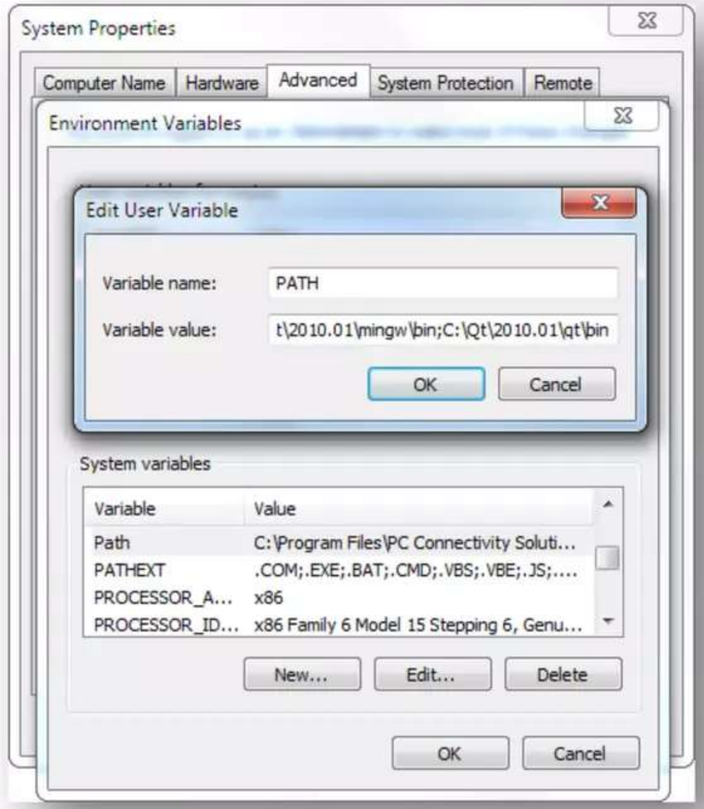
Platform: Windows

- **Situation**
  - Successfully compiled application, execution through Qt Creator works.
- **Problem**
  - When executing .exe-file directly through Windows Explorer: error message – **DLL files not found**.
  - **Common:** mingwm10.dll, qtcore4.dll, qtgui4.dll, qtcored4.dll, qtguid4.dll

Troubleshooting: Qt DLLs II

Platform: Windows

- **Solution**
  - Search for location of DLL files on PC, add directories to system environment variables.
  - ⊞-Key + Pause → Advanced system settings → Advanced → Environment Variables...
    Add at the end of Path variable in user or system variables.
  - Commonly needed:
    `C:\NokiaQtSDK\mingw\bin;`
    `C:\Qt\2010.02\qt\bin`
  - Not allowed to modify PATH? Copy required DLLs to executable directory.
  - More information: http://doc.qt.nokia.com/deployment-windows.html



Troubleshooting: DLL Entry Point

Platform: Windows

- **Situation**
  - Successfully compiled application, executing .exe through Windows Explorer
- **Problem**
  - Error message like:

    The procedure entry point ?end@QListData@@QBEPAPAXXZ could not be located in the dynamic link library QtCore4.dll
- **Solution**
  - Wrong version of dynamically linked Qt Dll was found and is used by Windows.
  - Modify PATH environment variable, move Qt directories to the front of System Variables.

    Or: copy DLLs to executable directory.

.sis Installation Fails

Platform: Symbian

- **Situation**
  - Successfully compiled application for a Symbian device
- **Problem**
  - Installation fails with an error message like:

```
Deploying application to 'Nokia N97 mini USB (COM9)'...
Copying installation file...
Installing application...
Could not install from package C:\xxx.sis on device: General OS-related error
Finished.
```

- **Solution**
  - **Install the .sis-file manually** through the Nokia Ovi Suite or by simply sending the .sis-file to the phone using Bluetooth or an USB connection.
    You will then **see the full error message** on the device. For example, you might have an old Qt version installed on your phone.

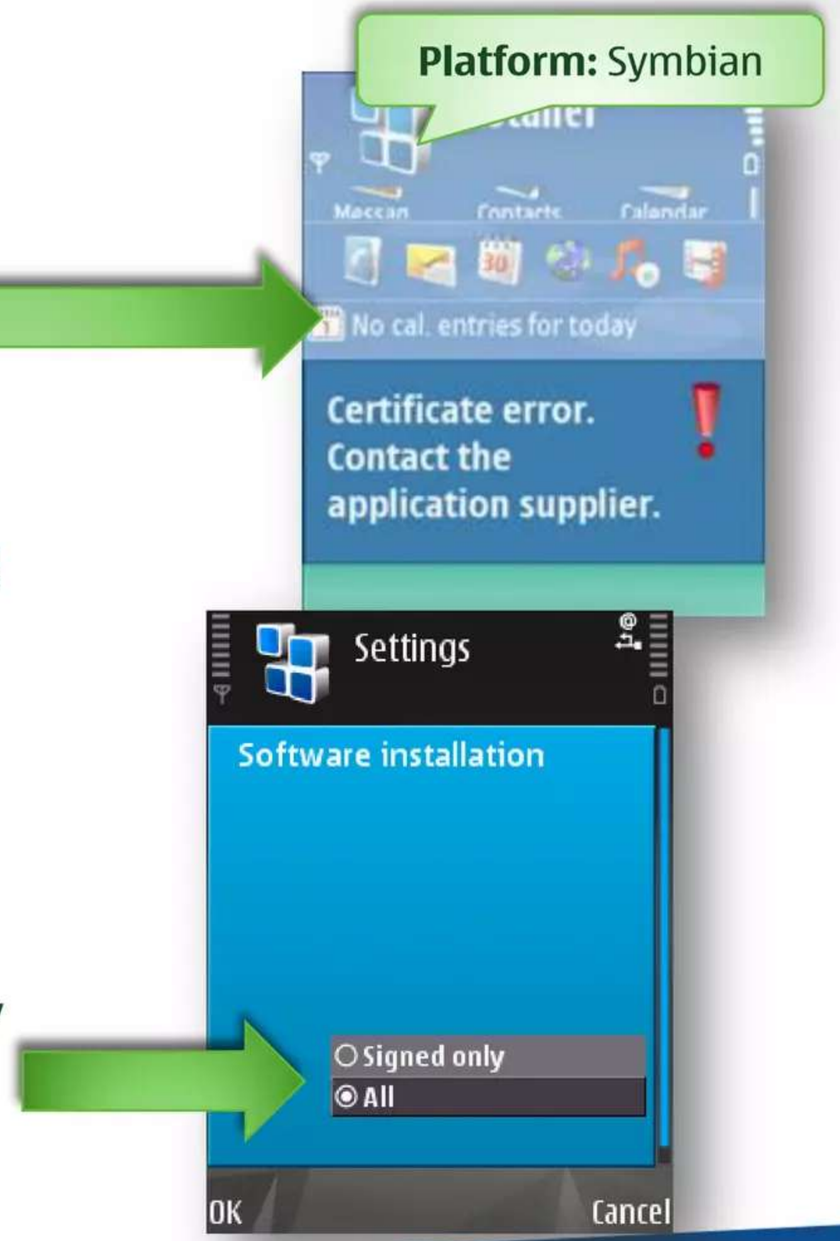
.sis Installation Fails II

Platform: Symbian

- **Common issues**
  - **Certificate validity:** are both PC and device date & time correct?
  - Changed **certificate type or UID**: uninstall old app version from device before installing new version
  - **Correct certificate:** only "**User Capabilities**" can be used with **self-signed certificate** created by default from Qt Creator. For other capability groups, you need a developer certificate – get this for free through Publish to Ovi.
    http://developer.symbian.org/wiki/index.php/Capabilities_%28Symbian_Signed%29
  - **Checklist:** http://developer.symbian.org/wiki/index.php/Troubleshoot_install_errors
- **Getting more details**
  - Install **ErrRd utility** to the phone: http://www.symbianresources.com/cgi-bin/schlabo/dl.pl?ErrRd
  - You will get **error code** during installation. Overview of codes: http://blogs.forum.nokia.com/blog/lucian-tomuas-forum-nokia-blog/2009/09/29/the-ultimate-software-installer-debug-guide

Troubleshooting: .sis vs .sisx?

- **What's the difference?**
  - No difference to Symbian device, it's just the filename.
  - Common: **.sisx** = signed version of unsigned .sis. But:
  - **Nokia Qt SDK: .sis is already signed, no .sisx is created!**
    - When not specifying own certificate: .sis is self-signed with automatically generated certificate
- **Installation of self-signed apps not allowed by default on some operator branded devices**
  - **Change:** Application manager → Installation settings → Software installation → change from "Signed only" to "All"



Forum Nokia

Thank You.

Want to learn more?
www.forum.nokia.com/Qt