Qt in Education

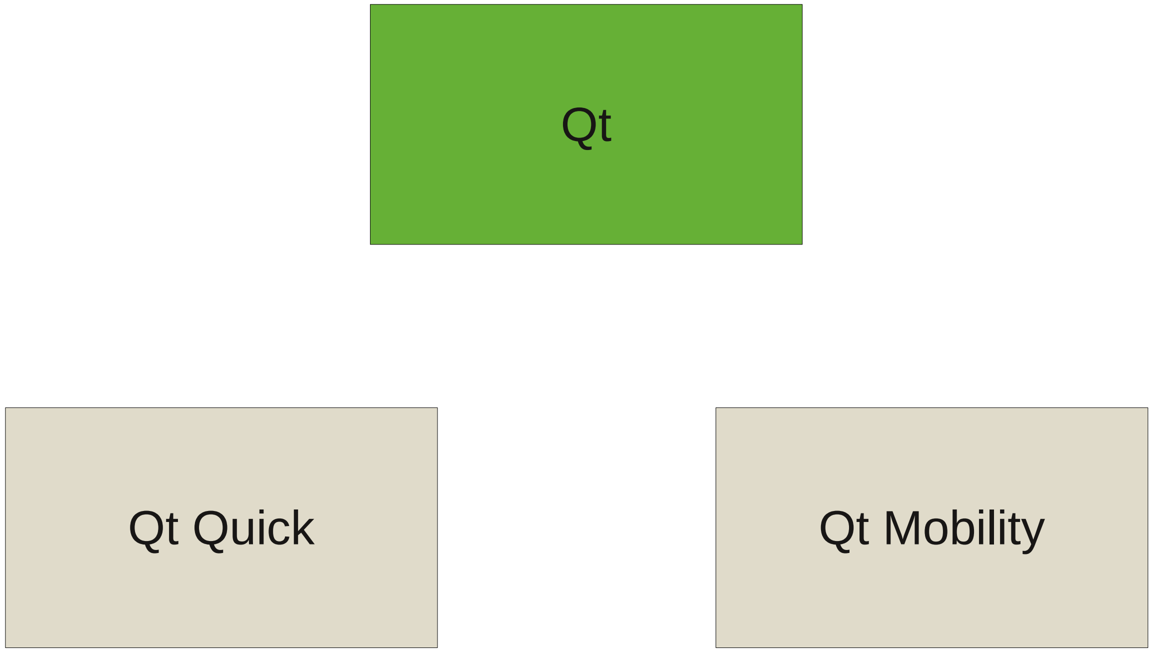
Developing Qt Apps for MeeGo

NOKIA

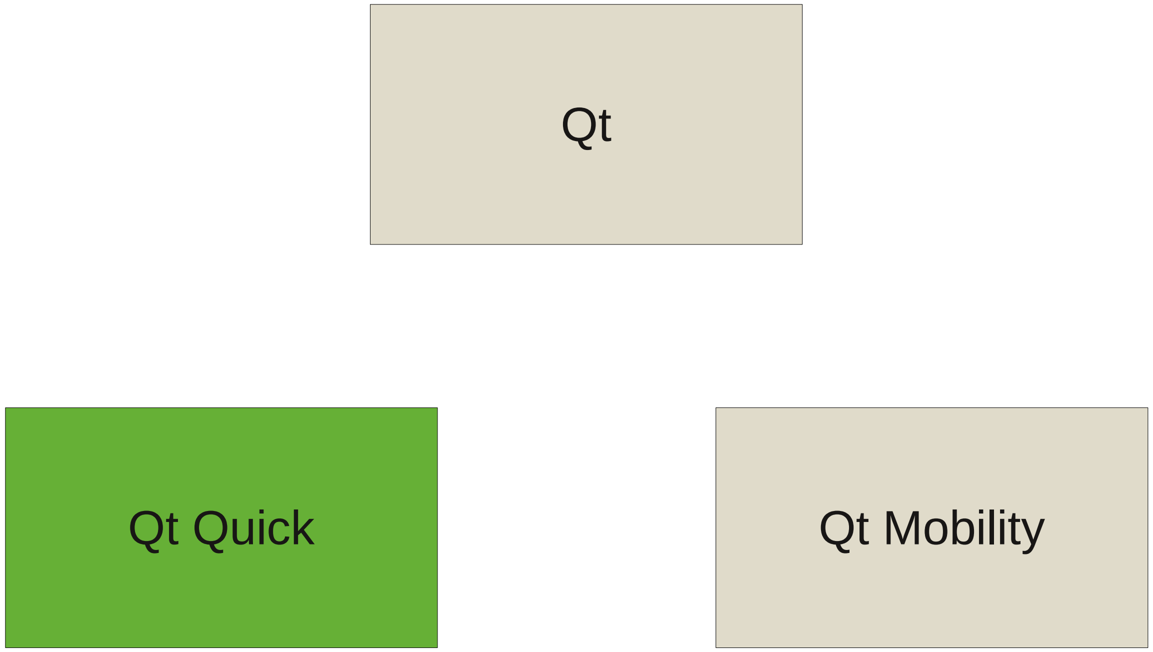
Development Tool-Chain

- The MeeGo SDK is built from the following components
  - QtCreator – integrated development environment
  - Qt Simulator – for running and debugging applications in a simulated environment
  - MADDE – for building and debuggin applications in real or emulated environments

APIs to Use

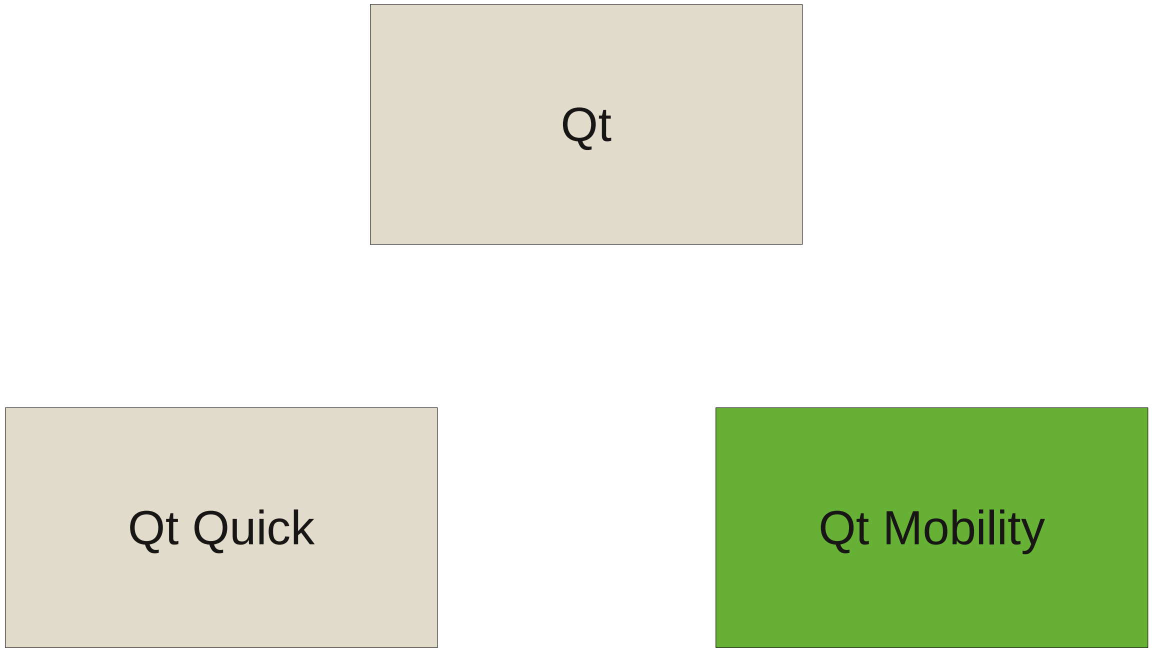


APIs to Use



QML – fluid user interfaces!
Direct access to parts of Mobility

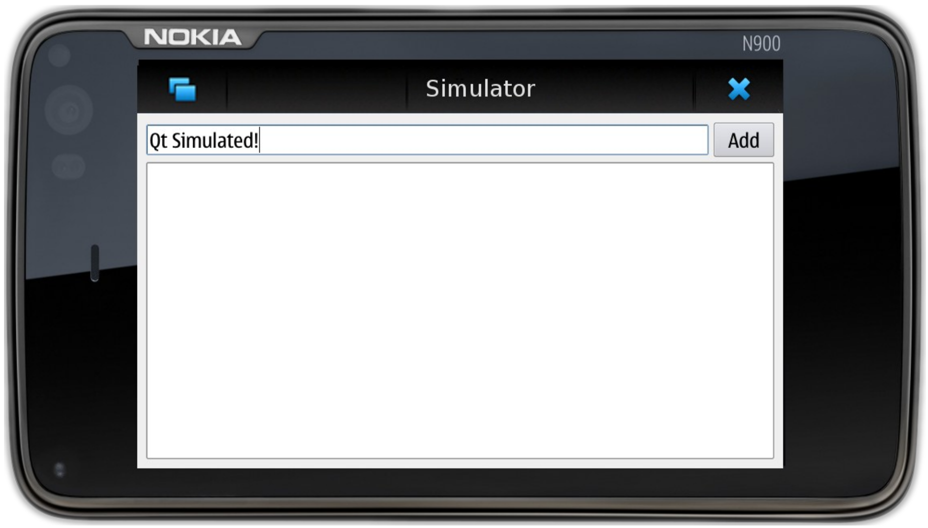
APIs to Use



Sensors, cameras, multimedia, documents, contacts, location, calendar, etc

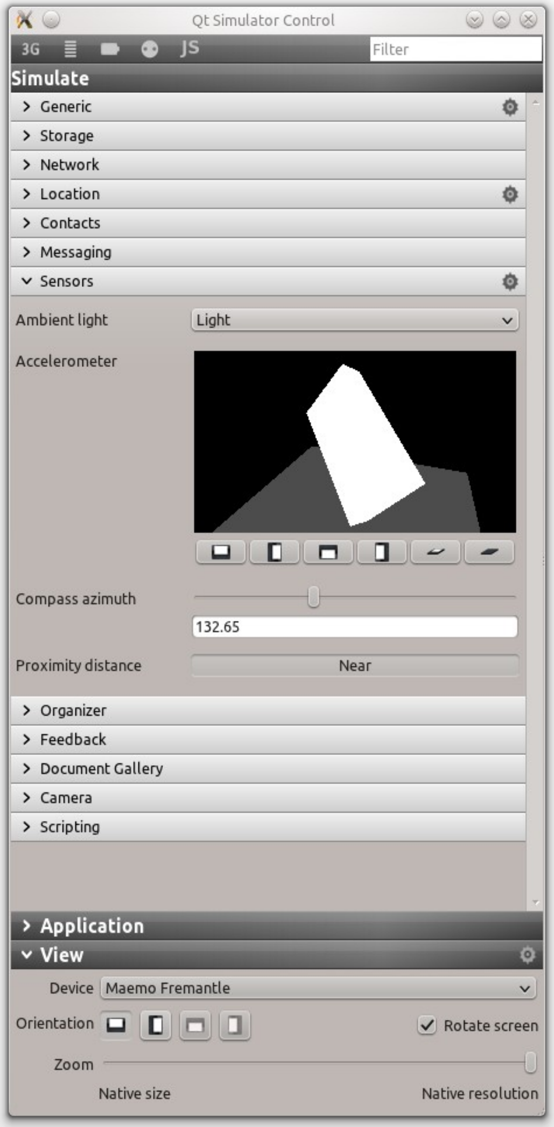
Qt Simulator

- Most applications are developed using the Qt Simulator
  - Code is executed natively – easy to debug
  - Simulated environment



Qt Simulator

- The simulator interface exposes much of Qt Mobility
  - Can trigger events
  - Can modify data
  - Can change sensor readings
- It is possible to script the simulator



Ambient light: Light

Accelerometer

Compass azimuth: 132.65

Proximity distance: Near

Device: Maemo Fremantle

Orientation

✓ Rotate screen

Zoom

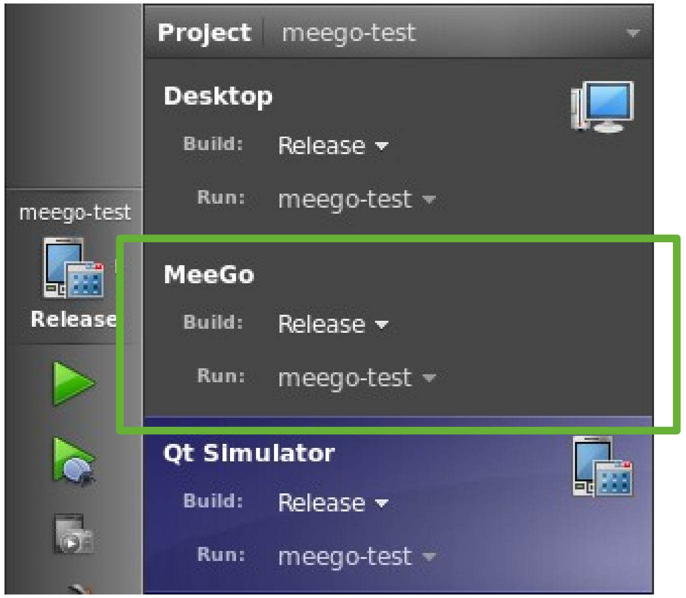
Deploying Apps

- Build for target
  - Results in an RPM-file (an installer package)
  - Test, test and test again
  - Publish to a store, or through your own channels

- Stores
  - Intel AppUp
  - Nokia Ovi Store

Building RPMs

- Simply select the MeeGo target
  - The resulting RPM resides in *project*-build-meego/rrpmbuild



Working with RPMs

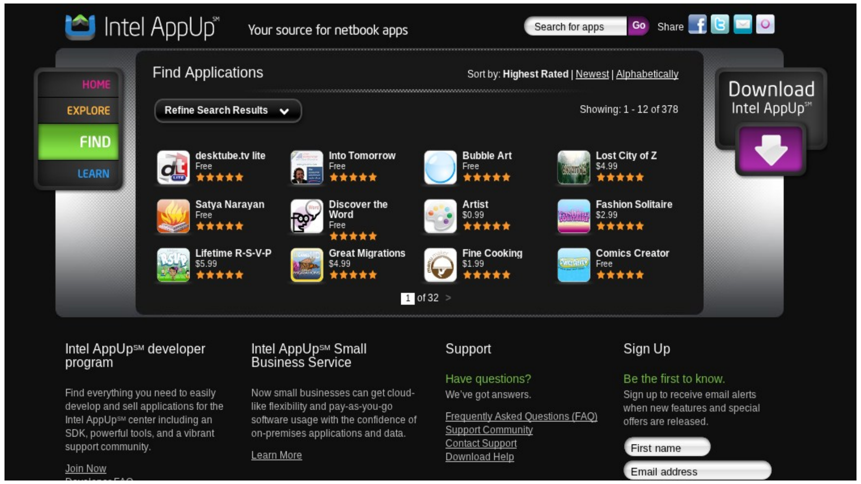
- When you have the RPM on your target, you can handle it using rpm
  - `rpm -i` *`myrpm-version.rpm`* – install
  - `rpm -e` *`myrpm`* – erase
  - `rpm -U` *`myrpm-version.rpm`* – upgrade
- or using zypper
  - `zypper install` *`myrpm.rpm`*
  - `zypper remove` *`myrpm`*

Repositories

- For easier installation and updates, you can place your RPMs in a repository
  - User the `createrepo` tool to create a repository
    - http://createrepo.baseurl.org/
  - Zypper handles repositories
    - `zypper ar` to add a repository to a target
    - `zypper refresh` to update meta-data from repositories
    - `zypper search` to search all repositories for a package
    - `zypper info` for information about a package
    - `zypper install` to install a package

Intel AppUp

- On-line app store for MeeGo netbooks and tablets
- Possible to sell applications, or to distribute them for free
- Apps are validated before they are made available through the store



Find Applications

Sort by: Highest Rated | Newest | Alphabetically

Showing: 1 - 12 of 378

desktube.tv lite Free	Into Tomorrow Free	Bubble Art Free	Lost City of Z $4.99
Satya Narayan Free	Discover the Word Free	Artist $0.99	Fashion Solitaire $2.99
Lifetime R-S-V-P $5.99	Great Migrations $4.99	Fine Cooking $1.99	Comics Creator Free

1 of 32 >

Intel AppUpSM developer program

Find everything you need to easily develop and sell applications for the Intel AppUpSM center including an SDK, powerful tools, and a vibrant support community.

Join Now

Intel AppUpSM Small Business Service

Now small businesses can get cloud-like flexibility and pay-as-you-go software usage with the confidence of on-premises applications and data.

Learn More

Support

Have questions?
We've got answers.

Frequently Asked Questions (FAQ)
Support Community
Contact Support
Download Help

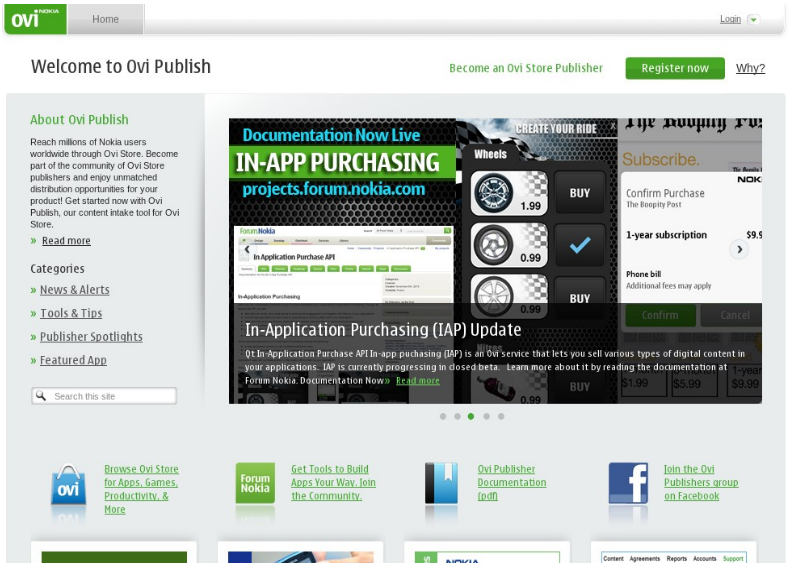
Sign Up

Be the first to know.
Sign up to receive email alerts when new features and special offers are released.

First name
Email address

Nokia Ovi Store

- On-line app store for MeeGo handsets
- Possible to sell applications, or to distribute them for free
- Apps are validated before they are made available through the store



Welcome to Ovi Publish

About Ovi Publish

Reach millions of Nokia users worldwide through Ovi Store. Become part of the community of Ovi Store publishers and enjoy unmatched distribution opportunities for your product! Get started now with Ovi Publish, our content intake tool for Ovi Store.

» Read more

Categories

» News & Alerts
» Tools & Tips
» Publisher Spotlights
» Featured App

In-Application Purchasing (IAP) Update

Qt In-Application Purchase API In-app puchasing (IAP) is an Ovi service that lets you sell various types of digital content in your applications. IAP is currently progressing in closed beta. Learn more about it by reading the documentation at Forum Nokia. Documentation Now» Read more

Browse Ovi Store for Apps, Games, Productivity, & More

Get Tools to Build Apps Your Way. Join the Community.

Ovi Publisher Documentation (pdf)

Join the Ovi Publishers group on Facebook

Stores in General

- Offers a profit sharing program
- Provides API for in-app purchases
  - Upgrades (try and buy)
  - Subscriptions
  - Virtual goods (for games, etc)
- Handles payments from multiple markets

Self-Publish

- It is possible to self-publish applications
  - Provide RPMs
  - Provide access to a repository

- What you do not get when self-publishing
  - Validated applications
  - Support for handling payments
  - Easy access through a global store

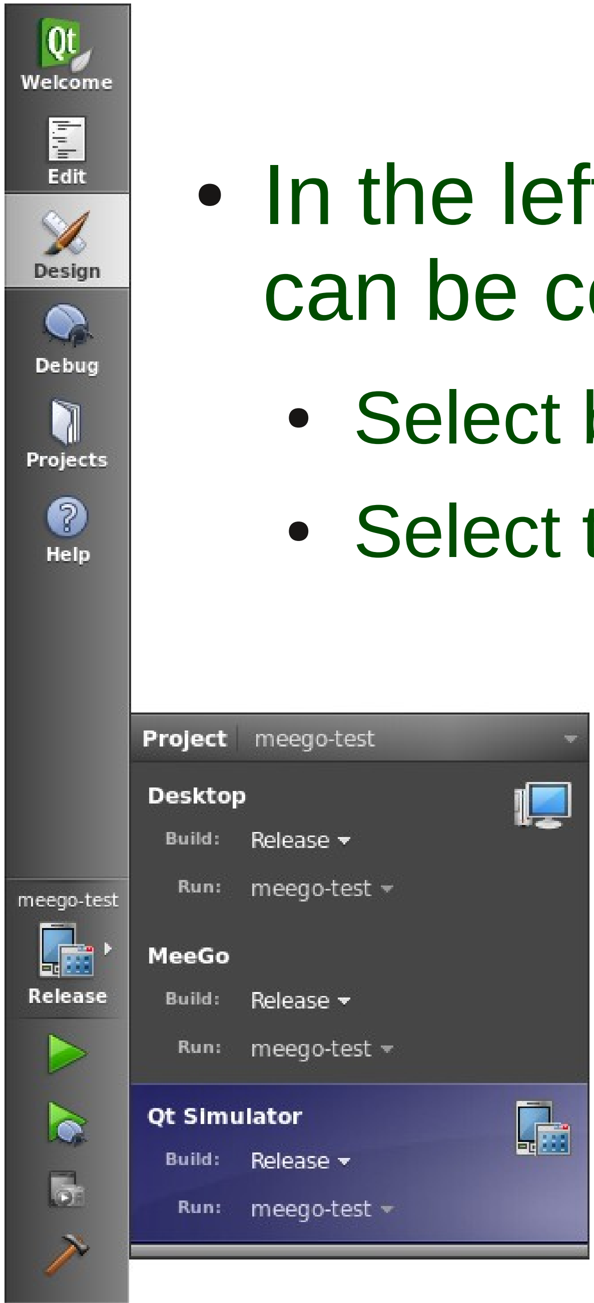
Break

An Example Project

- Setup a project
  - A C++ application with Qt Quick support
- Test on desktop
  - Basic testing and debugging
- Test using simulator
  - Testing and debugging with “sensor” input and proper screen resolution
- Test using remote (virtual) target
  - Final testing on an actual target

Selecting target

- In the left Qt Creator toolbar, the target can be configured
  - Select between Release or Debug
  - Select target
    - Simulator
    - Deployment
    - Any other target of the project, e.g. local



Project meego-test

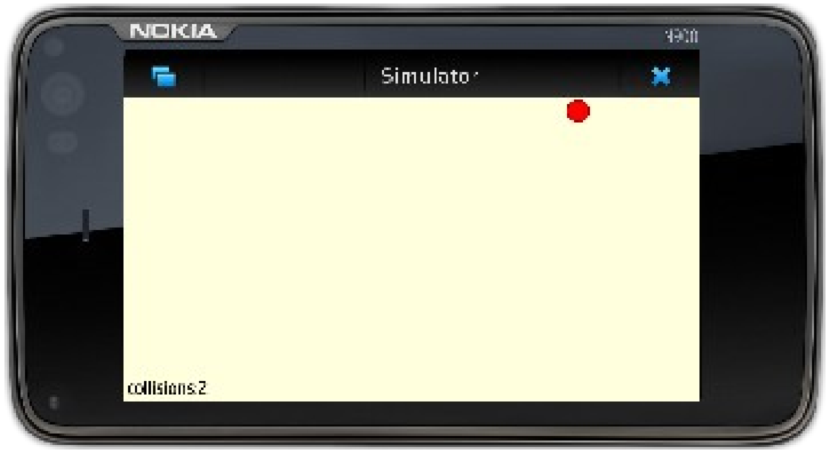
Desktop
Build: Release
Run: meego-test

MeeGo
Build: Release
Run: meego-test

Qt Simulator
Build: Release
Run: meego-test

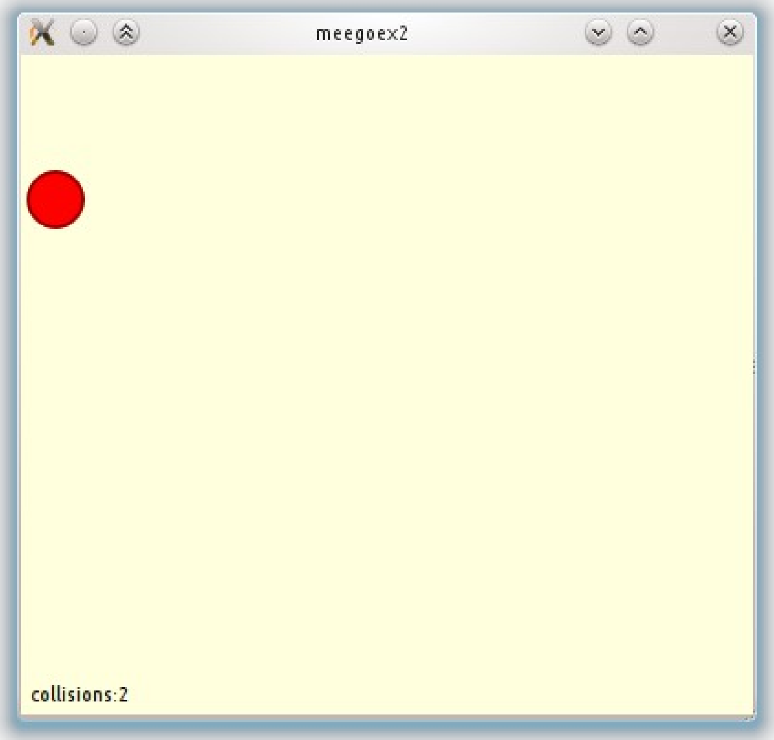
Project Outline

- Core application in C++
- Qt Quick user interface
- C++ slot called from Qt Quick event

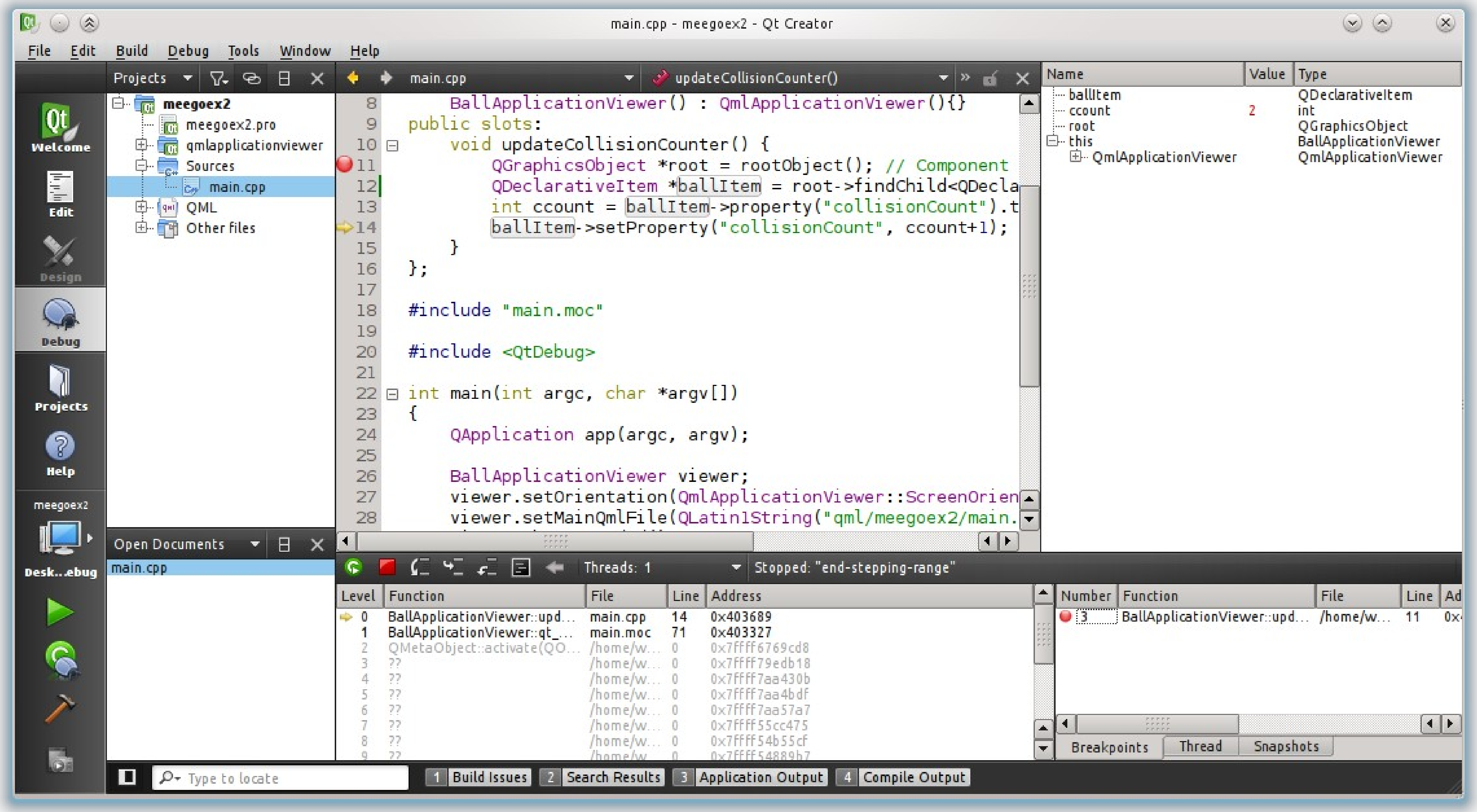


Desktop Experience

- Select the Desktop target
- Set a breakpoint in the slot
- Build and run



Desktop Debugging



```
    BallApplicationViewer() : QmlApplicationViewer(){}
public slots:
    void updateCollisionCounter() {
        QGraphicsObject *root = rootObject(); // Component
        QDeclarativeItem *ballItem = root->findChild<QDecla
        int ccount = ballItem->property("collisionCount").t
        ballItem->setProperty("collisionCount", ccount+1);
    }
};

#include "main.moc"

#include <QtDebug>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    BallApplicationViewer viewer;
    viewer.setOrientation(QmlApplicationViewer::ScreenOrien
    viewer.setMainQmlFile(QLatin1String("qml/meegoex2/main.
```

Name	Value	Type
ballItem		QDeclarativeItem
ccount	2	int
root		QGraphicsObject
this		BallApplicationViewer
QmlApplicationViewer		QmlApplicationViewer

Threads: 1 Stopped: "end-stepping-range"

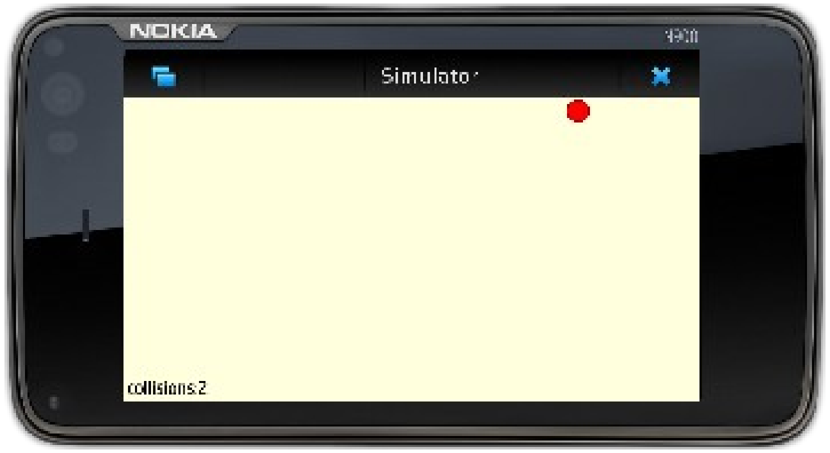
Level	Function	File	Line	Address
0	BallApplicationViewer::upd...	main.cpp	14	0x403689
1	BallApplicationViewer::qt_...	main.moc	71	0x403327
2	QMetaObject::activate(QO...	/home/w...	0	0x7ffff6769cd8
3	??	/home/w...	0	0x7ffff79edb18
4	??	/home/w...	0	0x7ffff7aa430b
5	??	/home/w...	0	0x7ffff7aa4bdf
6	??	/home/w...	0	0x7ffff7aa57a7
7	??	/home/w...	0	0x7ffff55cc475
8	??	/home/w...	0	0x7ffff54b55cf
9	??	/home/w...	0	0x7ffff54889b7

Number	Function	File	Line	Ad
3	BallApplicationViewer::upd...	/home/w...	11	0x

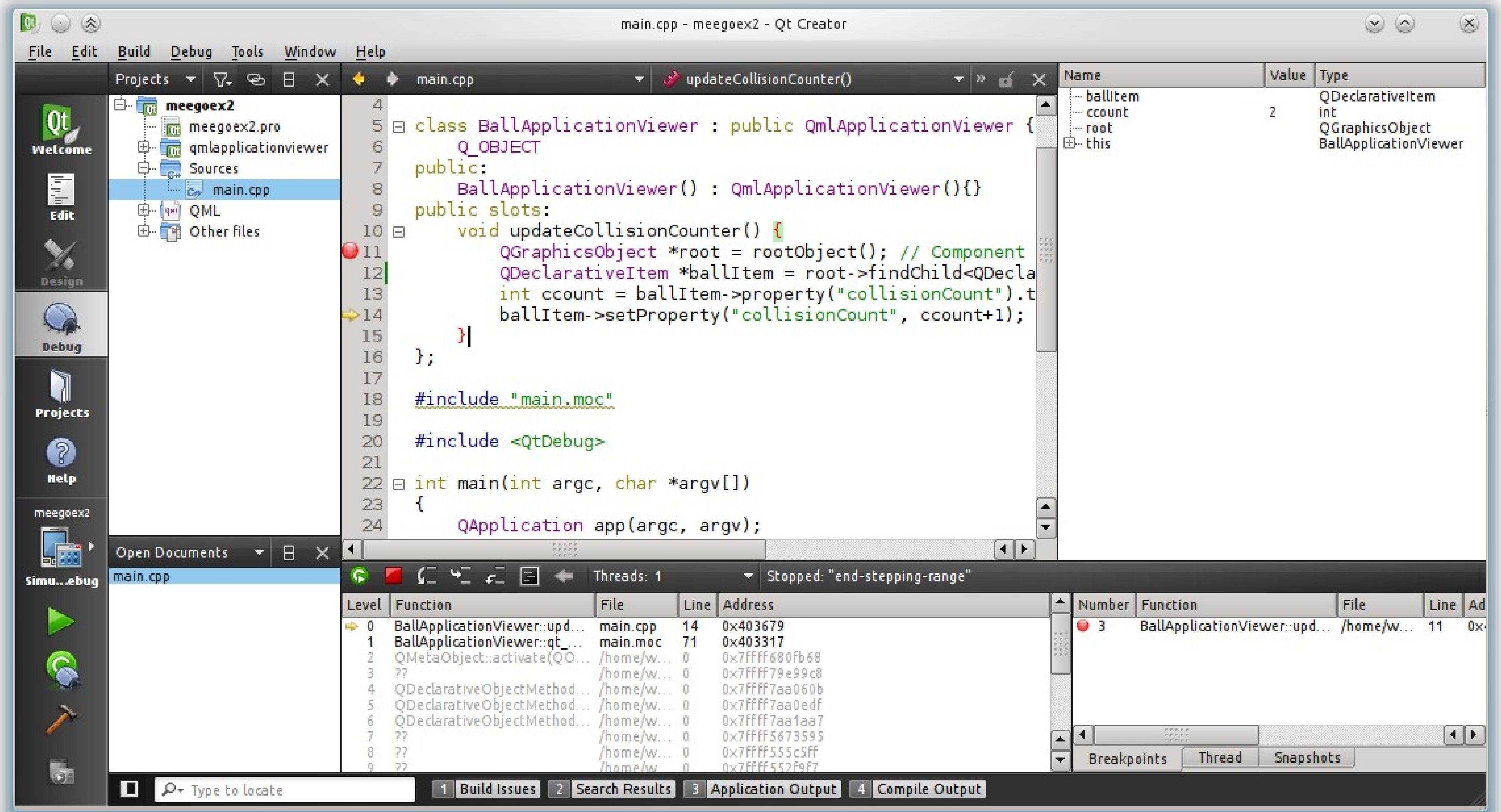
NOKIA

Simulator Experience

- Select the Simulator target
- Set a breakpoint in the slot
- Build and run



Simulator Debugging



```
class BallApplicationViewer : public QmlApplicationViewer {
    Q_OBJECT
public:
    BallApplicationViewer() : QmlApplicationViewer(){}
public slots:
    void updateCollisionCounter() {
        QGraphicsObject *root = rootObject(); // Component
        QDeclarativeItem *ballItem = root->findChild<QDecla
        int ccount = ballItem->property("collisionCount").t
        ballItem->setProperty("collisionCount", ccount+1);
    }
};

#include "main.moc"

#include <QtDebug>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
```

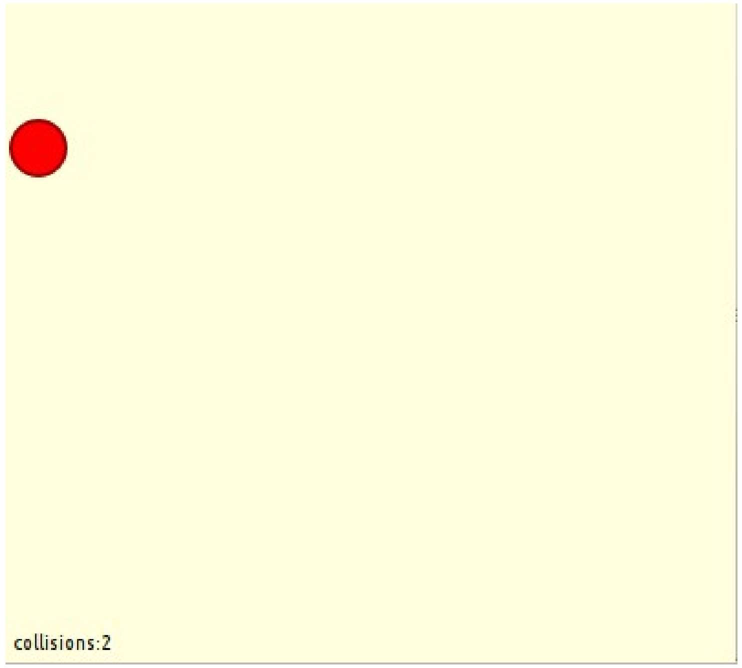
Name	Value	Type
ballItem		QDeclarativeItem
ccount	2	int
root		QGraphicsObject
this		BallApplicationViewer

Level	Function	File	Line	Address
0	BallApplicationViewer::upd...	main.cpp	14	0x403679
1	BallApplicationViewer::qt_...	main.moc	71	0x403317
2	QMetaObject::activate(QO...	/home/w...	0	0x7ffff680fb68
3	??	/home/w...	0	0x7ffff79e99c8
4	QDeclarativeObjectMethod...	/home/w...	0	0x7ffff7aa060b
5	QDeclarativeObjectMethod...	/home/w...	0	0x7ffff7aa0edf
6	QDeclarativeObjectMethod...	/home/w...	0	0x7ffff7aa1aa7
7	??	/home/w...	0	0x7ffff5673595
8	??	/home/w...	0	0x7ffff555c5ff
9	??	/home/w...	0	0x7ffff552f9f7

Number	Function	File	Line	Ad
3	BallApplicationViewer::upd...	/home/w...	11	0x

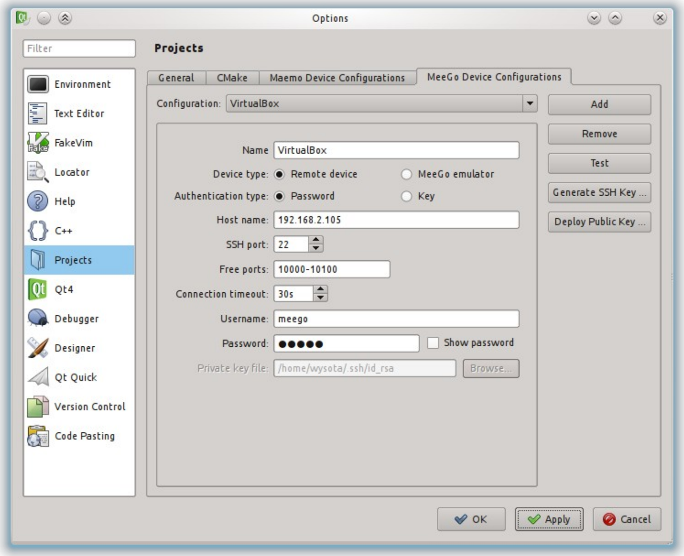
Target Experience

- Select the MeeGo target
- Set a breakpoint in the slot
- Build and deploy



Configure Target

- Select target device
  - Settings – Project – MeeGo Device Configuration
    - Host name – IP of remote target
    - User name / password
    - Target must run sshd



Options

Projects

General | CMake | Maemo Device Configurations | MeeGo Device Configurations

Configuration: VirtualBox

Name: VirtualBox

Device type: [✓ Remote device] MeeGo emulator

Authentication type: [✓ Password] Key

Host name: 192.168.2.105

SSH port: 22

Free ports: 10000-10100

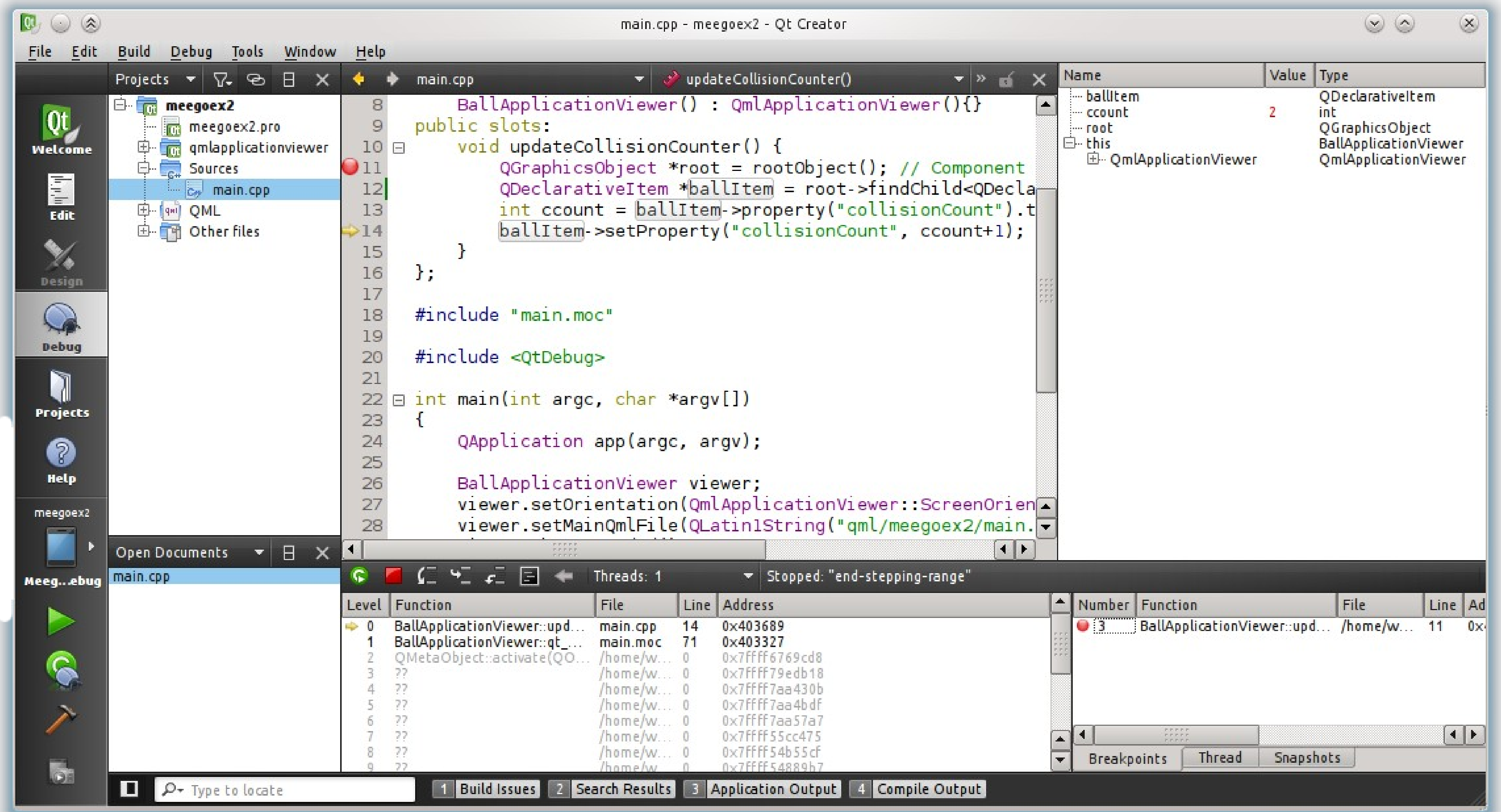
Connection timeout: 30s

Username: meego

Password: ●●●●● ☐ Show password

Private key file: /home/wysota/.ssh/id_rsa

MeeGo Debugging



main.cpp - meegoex2 - Qt Creator

File Edit Build Debug Tools Window Help

Projects: meegoex2 — meegoex2.pro, qmlapplicationviewer, Sources (main.cpp), QML, Other files

main.cpp — updateCollisionCounter()

```
 8         BallApplicationViewer() : QmlApplicationViewer(){}
 9     public slots:
10         void updateCollisionCounter() {
11             QGraphicsObject *root = rootObject(); // Component
12             QDeclarativeItem *ballItem = root->findChild<QDecla
13             int ccount = ballItem->property("collisionCount").t
14             ballItem->setProperty("collisionCount", ccount+1);
15         }
16     };
17
18     #include "main.moc"
19
20     #include <QtDebug>
21
22     int main(int argc, char *argv[])
23     {
24         QApplication app(argc, argv);
25
26         BallApplicationViewer viewer;
27         viewer.setOrientation(QmlApplicationViewer::ScreenOrien
28         viewer.setMainQmlFile(QLatin1String("qml/meegoex2/main.
```

Name	Value	Type
ballItem		QDeclarativeItem
ccount	2	int
root		QGraphicsObject
this		BallApplicationViewer
QmlApplicationViewer		QmlApplicationViewer

Threads: 1 Stopped: "end-stepping-range"

Level	Function	File	Line	Address
0	BallApplicationViewer::upd...	main.cpp	14	0x403689
1	BallApplicationViewer::qt_...	main.moc	71	0x403327
2	QMetaObject::activate(QO...	/home/w...	0	0x7ffff6769cd8
3	??	/home/w...	0	0x7ffff79edb18
4	??	/home/w...	0	0x7ffff7aa430b
5	??	/home/w...	0	0x7ffff7aa4bdf
6	??	/home/w...	0	0x7ffff7aa57a7
7	??	/home/w...	0	0x7ffff55cc475
8	??	/home/w...	0	0x7ffff54b55cf
9	??	/home/w...	0	0x7ffff54889b7

Number	Function	File	Line	Ad
3	BallApplicationViewer::upd...	/home/w...	11	0x

Differences

- Desktop
  - Native execution on local machine
  - No mobility services (in SDK build)
- Simulator
  - Native execution on local machine
  - Simulated mobility services
- Target
  - Native execution on remote machine
  - Actual mobility services

Platform Specific Code

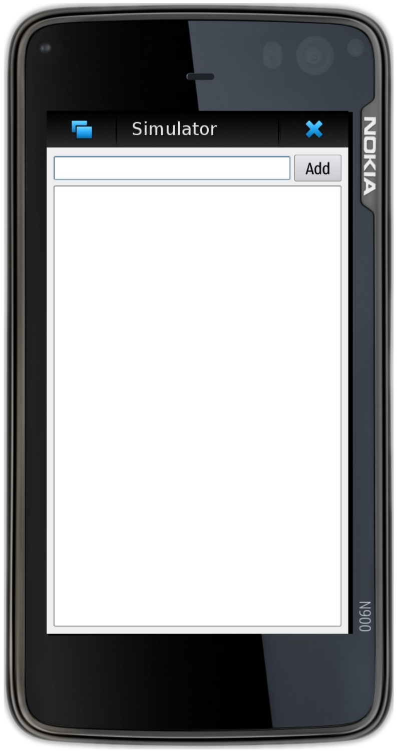
- Preprocessor defines identify the platform
  - Desktop
    - Q_WS_X11 / Q_WS_MAC / Q_WS_WIN
  - Simulator
    - Q_SIMULATOR
  - Target
    - Q_WS_X11

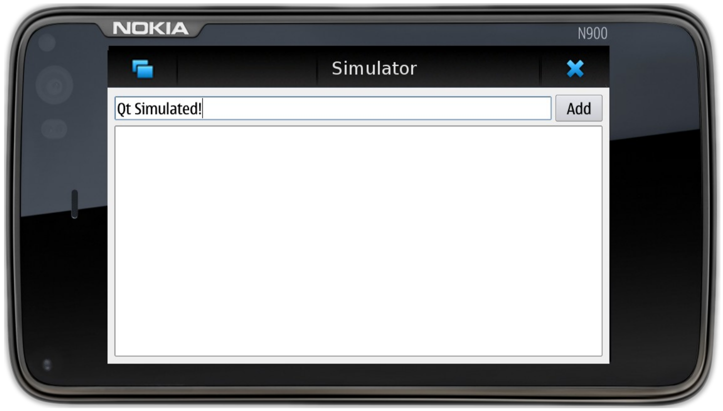
Development Considerations Overview

- Window sizes / orientation
- Limited input – touch and no keyboard
- Memory availability – RAM and persistent storage
- Service availability - sensors
- Cost of bandwidth / connection reliability
- Media codec availability

Window sizes

- Windows might be freely sized, but can also be locked to the geometry of the screen or a top-level window
- Your design must be adapted to this



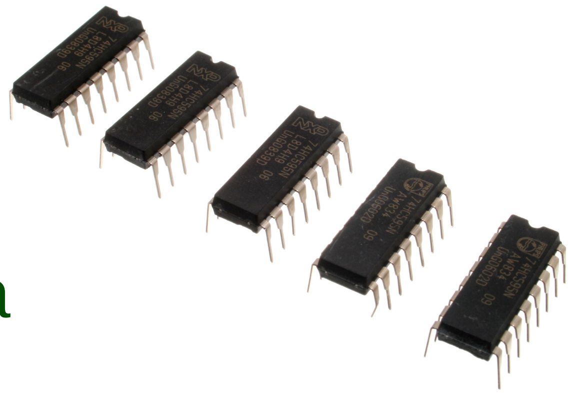


Limited input

- Keyboards can be limited to many different configurations
  - Keypad
    - Predictive text, or limited key set
  - Touch
    - On screen keyboard, uses screen estate
  - Finger-based touch
    - Limited accuracy, interactive areas must have a minimum (touchable) size

Memory availability

- Memory of embedded devices is more limited than on a desktop system
- Today, memories can be considered large
- It is still important to be smart
  - Avoid duplicated data
  - Avoid bloated data structures
  - Avoid shipping too much data



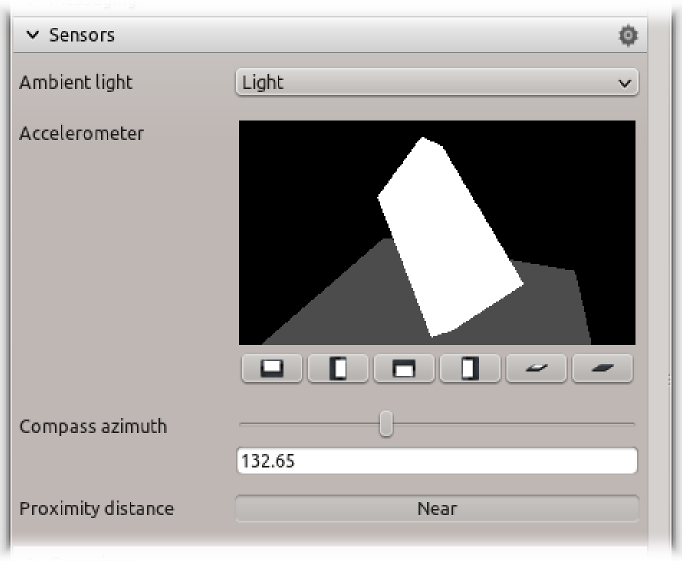
Cost of bandwidth

- Mobile devices can be assumed to use expensive and unreliable network connections
  - Data traffic roaming can be expensive
  - Connections can be interrupted at any time
  - Bandwidth can be limited at times
- Let the user control network access
- Support an off-line mode

CC-BY-SA http://www.flickr.com/photos/osde-info/4664431806/

Service availability

- Different devices have different features
  - Location resolution, and availability
  - Orientation sensor availability
  - Contacts and calendar availability
  - etc
- One of the key elements of the Qt Mobility API is that it can abstract services that are optional



Codec availability

- Media playback is an area affected by patents and licenses
  - All systems cannot decode all media
  - Ensure that codecs are available for your target platform
  - Qt Mobility / MultimediaKit relies on gstreamer to playback media on MeeGo devices