LinuxCon 2011

Improving Graphic Performance of MeeGo on ARM

Akira Tsukamoto
August 19th, 2011
Nomovok Ltd.

Who am I

How to improve graphic performance?

Why?

Because embedded systems are not resource rich

Graphic performance is critical for good UX feelings

Honest speaking Qt 4.7, on MeeGo is not fast :)

What can be done for current MeeGo 1.1/1.2

- X11(Window Manager) is not lightweight
- Qt 4.7 is not fully optimized for facilitating hardware graphic acceleration

What to do for Window Manager?

The feature of **Qt lighthouse**

- Provides abstraction layer of Window Manager

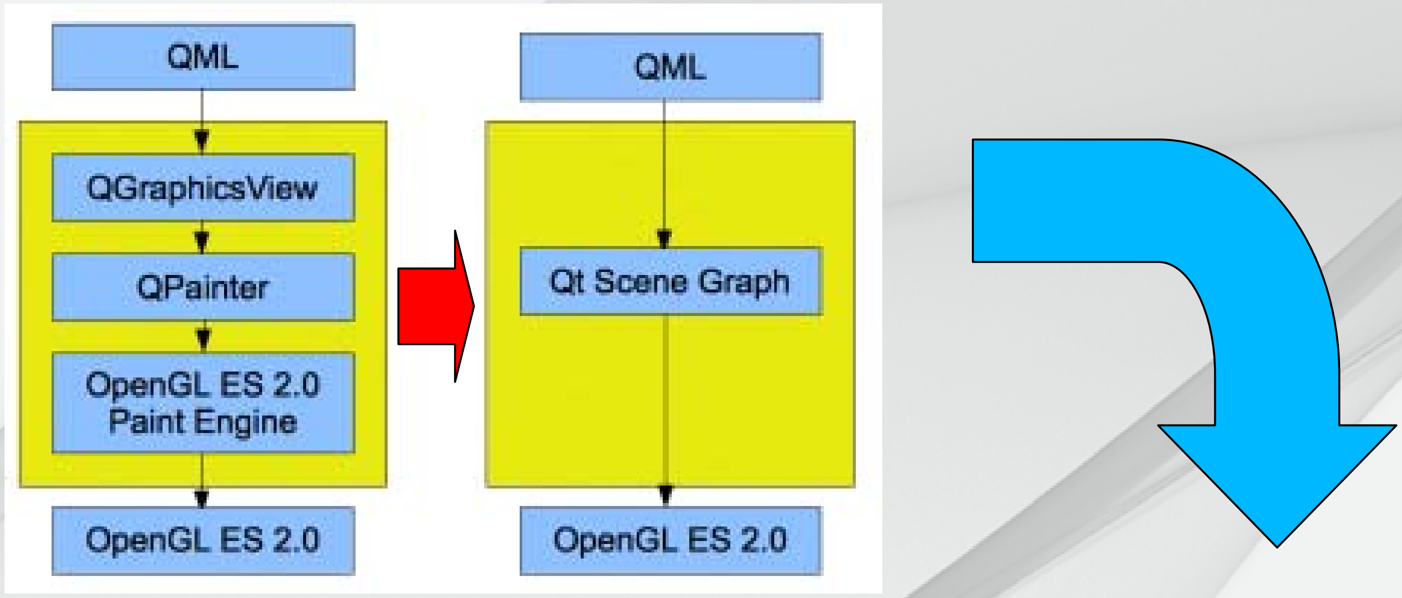
Able to

- Replace X11 with other Window Manager, such as, Wayland
- Remove X11 (This presentation uses this method)

How to utilize HW acceleration?

The feature of **Qt scene graph**

- Reduce the graphic API layers



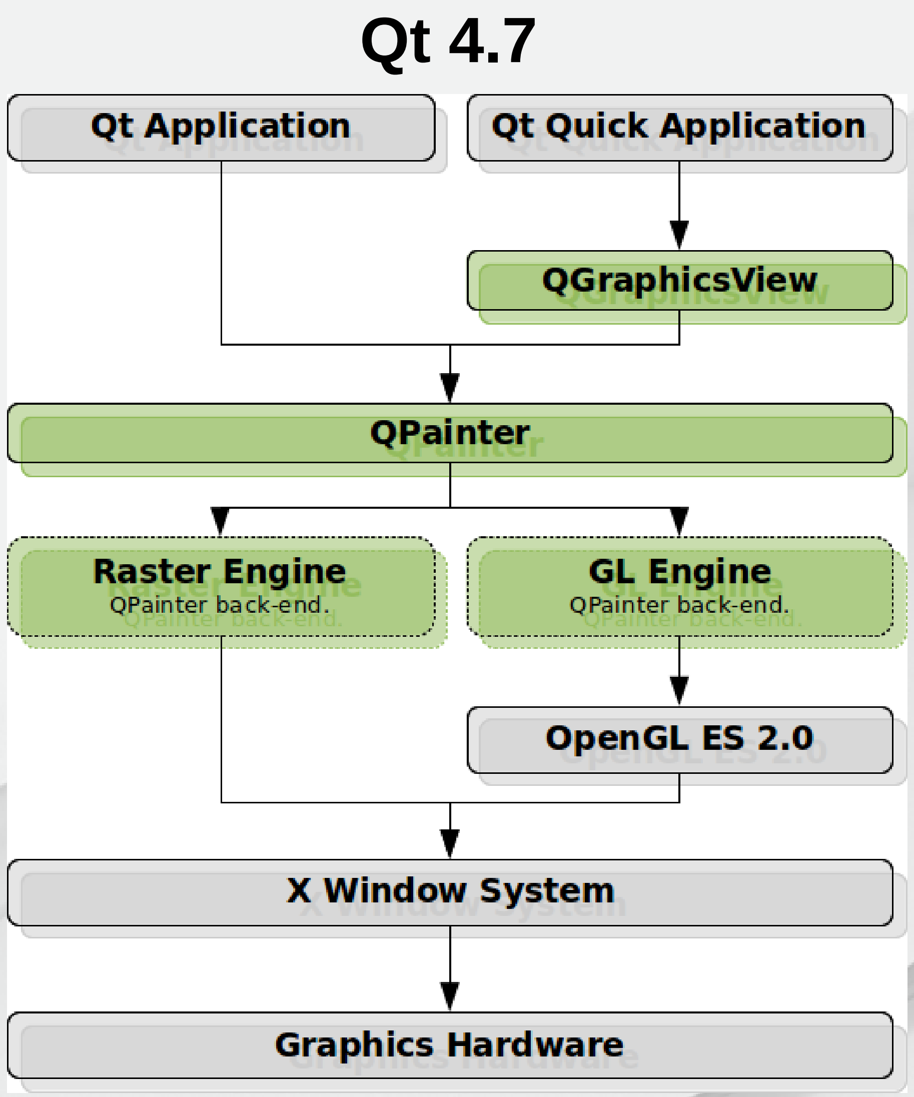
- Provides Qt, QML directly access hw acceleration of OpenGL ES

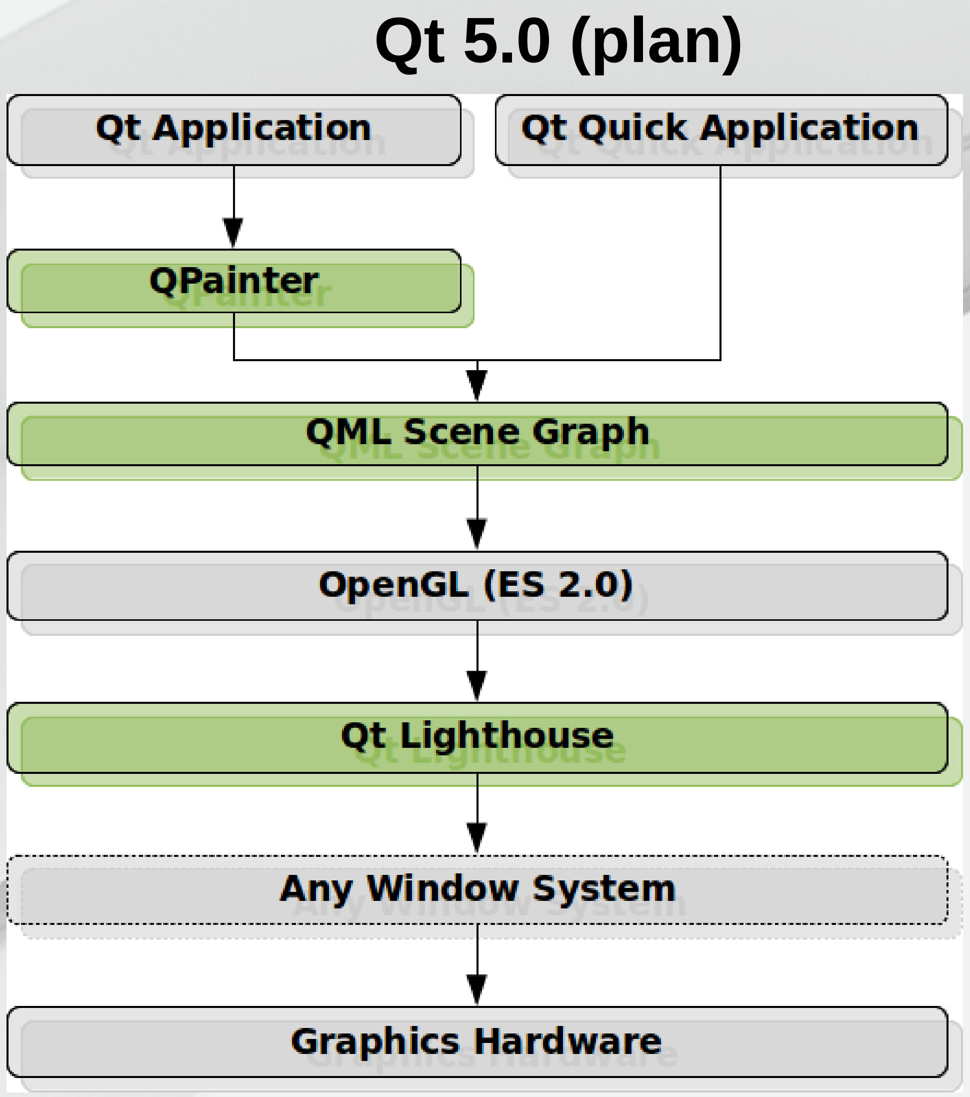
How to enable lighthouse and scene graph

Requires rebuilding specialized Qt framework by yourself on MeeGo

Not easy, but this slide will show you the overview with step by step instruction

Qt 4.7 and future 5.0





Instruction to rebuild Qt framework with Qt lighthouse and scene graph

Prerequisite (1/2)

Linux BSP for your ARM board

- Source to cross build Linux Kernel and etc
- Building BSP is out of the scope of today’s topic

Since we are removing X11 to improve the performance

- OpenGL ES and EGL libraries which are not linked to X11

Header for OpenGL ES and EGL to cross build optimized Qt

Expecting Ubuntu as a host machine for this slide

Prerequisite (2/2)

hard float and soft float binaries

- The ABI is not compatible between hard float and soft float binaries
- MeeGo 1.1 is soft float
- MeeGo 1.2 is hard float

If the vender provided OpenGLES and EGL binaries are only soft float, then need to use MeeGo 1.1 packages or request strongly to the vendor to provide hard float binaries to use MeeGo 1.2

Prepare cross build env for ARM

MeeGo provides convenient tools for cross building ARM binaries

- mic-tools with kickstart file
- prebuilt MeeGo package binaries for ARMv7 to create target sysroot image

Install mic tools

Install dependent packages to ubuntu first

- sudo apt-get install yum rpm kpartx parted syslinux isomd5sum kvm zlib1g-dev squashfs-tools python2.6-dev qemu-arm-static python urlgrabber

Specify the repository of mic tools

- deb http://repo.meego.com/MeeGo/tools/repos/ubuntu/10.10/ /

Import public keys for mic tools

- gpg --keyserver subkeys.pgp.net --recv 0BC7BEC479FC1F8A
- gpg --export --armor 0BC7BEC479FC1F8A | sudo apt-key add -

Install mic tools

- sudo apt-get update
- sudo apt-get install mic2

Obtain official kickstart file for ARM

Download any sample .ks file

- http://mirrors3.kernel.org/meego/builds/1.2.80/1.2.80.14.0.20110809.2/builddata/image-configs/handset-armv7hl-madde-sysroot.ks

Customize the kickstart file to remove X11 for your ARM board

How does customized .ks file look

```
...
%packages --excludedocs
pam
rootfiles
bash
sysvinit
coreutils
cpio
fuse
fuse-libs
…
```

Create target image containing MeeGo

use mic-image-creator as following

- sudo mic-image-creator --run-mode=0 --cache=./mic-cache-12/ --format=fs --arch=armv7hl --save-kernel --config=meego-qt-lightweight.ks

After you have used "mic-image-creator" you will have disk image similar in the directory bellow.

- meego-qt-lightweight-1.2.20110601.1558/

The directory above will be the disk image for cross building Qt.

Rename to more intuitive directory name

- sysroot-meego-qt-lightweight/

For more details:

http://wiki.meego.com/Image_Creation#Official_Meego_.ks_files

Obtain cross building toolchains

You may build your own toolchains for ARMv7 on your host machine from scratch but there are prebuilt toolchains (much easier)

Use cross toolchain in the BSP or

Download toolchain from:

- http://www.codesourcery.com/sgpp/lite/arm/portal/subscription3057

Untar to a directory to contain the cross compilers

- tar xjvf arm-2010.09-50-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

Source of Qt lighthouse with scene graph

Currently the sources for scene graph are not merged to master branch of Qt

Get the source from Qt staging

- git clone git://qt.gitorious.org/qt/staging.git qt-staging
- cd qt-staging
- git checkout origin/qml-team/qtquick2 -b qtquick2

Cross building Qt (env variables)

Setup environment variables for cross building

```
CPUS=<number of cores>
CROSS_COMPILER_BIN=</directory/to/toolchain>/Sourcery_G++/arm-2009q1/bin
SYSROOT_DIR=</directory/to/target-image>/sysroot-meego-qt-lightweight

export PATH=$CROSS_COMPILER_BIN:$PATH
export PKG_CONFIG_PATH=$SYSROOT_DIR/usr/lib/pkgconfig
export PKG_CONFIG_SYSROOT_DIR=$SYSROOT_DIR
export SYSROOT=$SYSROOT_DIR
```

Cross building Qt (configure and make)

Please add –qpa arm for lighthouse and –opengl es2 for scene graph to the options

- ./configure -platform linux-g++ -xplatform qws/linux-arm-gnueabi-g++ -qpa arm -opengl es2 -release -opensource -fast -arch arm -force-pkg-config -webkit -nomake examples -nomake demos -no-fontconfig -no-openvg -confirm-license -no-sql-ibase -no-sql-mysql -no-sql-odbc -no-sql-psql -plugin-sql-sqlite -no-sql-sqlite2 -no-sql-tds -system-sqlite -no-qt3support -xmlpatterns -no-multimedia -audio-backend -no-phonon -no-phonon-backend -svg -script -scripttools -system-libtiff -system-libpng -system-libjpeg -no-rpath -optimized-qmake -no-separate-debug-info -verbose -gtkstyle -no-nas-sound -no-openvg -dbus-linked -plugin-kbd-linuxinput -plugin-gfx-powervr -plugin-mouse-pc -qt-mouse-linuxinput -plugin-mouse-tslib 2>&1 | tee ../qt-lighthouse--configure.log

- make -j$CPUS 2>&1 | tee ../qt-lighthouse--make.log

- sudo make INSTALL_ROOT=$SYSROOT_DIR install

Qt lighthouse and scene graph

Then you will have Qt framework with the latest scene graph in the following directory:

- sysroot-meego-qt-lightweight /usr/local/Trolltech/QtLighthouse-4.8.0-arm

Cross building qmlscene

The following procedure only required on Qt staging

- cd tools/qmlscene
- make -j$CPUS

Then manually install qmlscene to target disk image

- cp bin/qmlscene sysroot-meego-qt-lightweight /usr/local/Trolltech/QtLighthouse-4.8.0-arm/bin/.

Finally

Write UI or applications with QML using scene graph API

Copy the target disk image to your ARM board (or use nfs boot)

Launching qmlscene

- export PATH=/usr/local/Trolltech/QtLighthouse-4.8.0-arm/bin:$PATH
- export LD_LIBRARY_PATH=/usr/local/Trolltech/QtLighthouse-4.8.0-arm/lib:$LD_LIBRARY_PATH
- /sbin/ldconfig
- cd /opt/<directory/to/qml>
- qmlscene -platform EglFS -frameless -dragthreshold 30 your.qml &

Result

Renesas AP4 mackerel

- Cortex A8 single 800MHz
- Imagination Technologies SGX 540

Improvement

12FPS -> 39FPS average, 50FPS maximum

Movie uploaded on YouTube

http://www.youtube.com/watch?v=zWOt0z--2M8

Honestly speaking

Cross building Qt framework is not this easy.

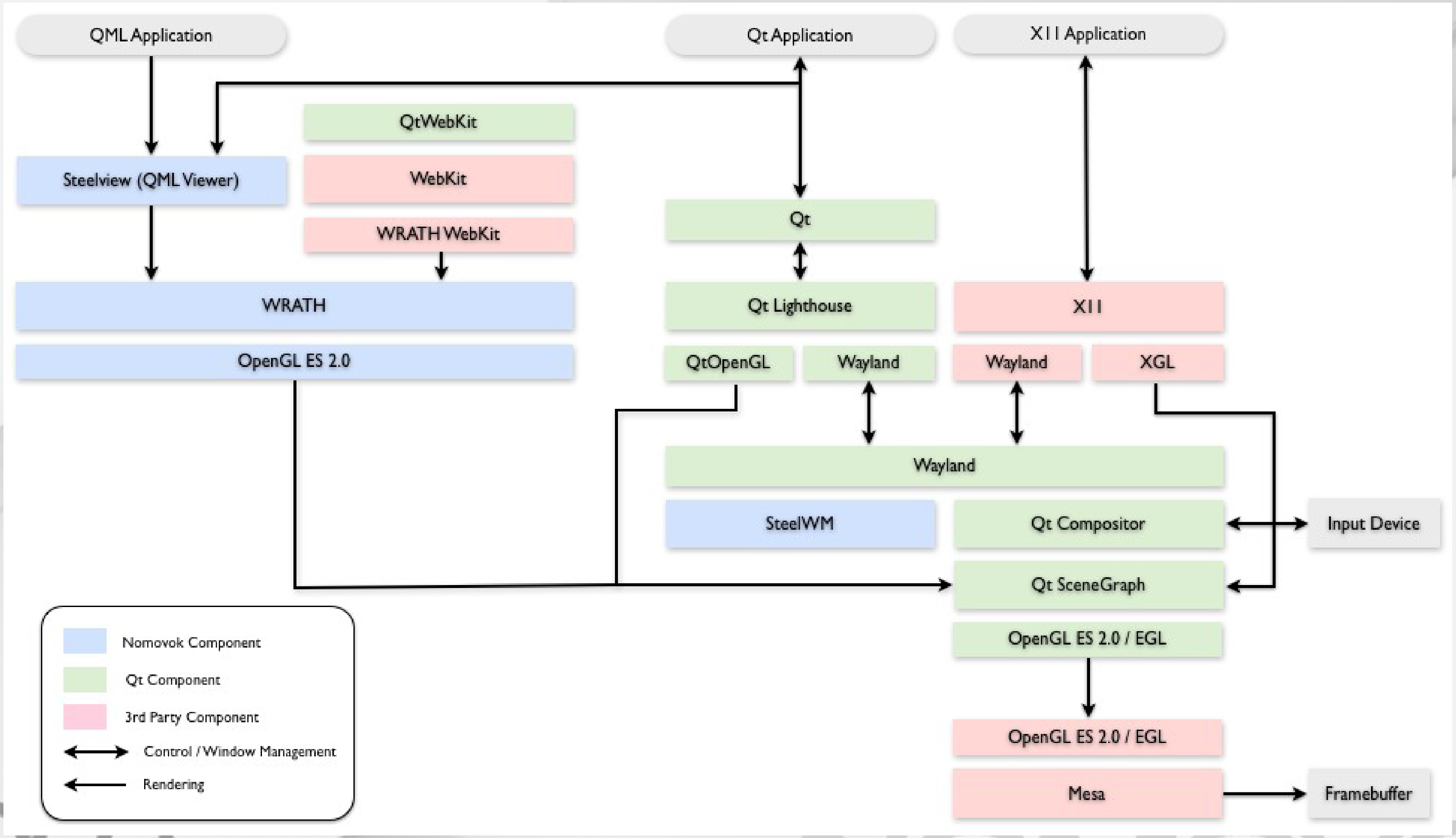
- Long building time (about 4 hours) -> be patient
- Build break -> patch and fix the source
- and etc

Please have fun hacking!! :)

For more performance by Nomovok

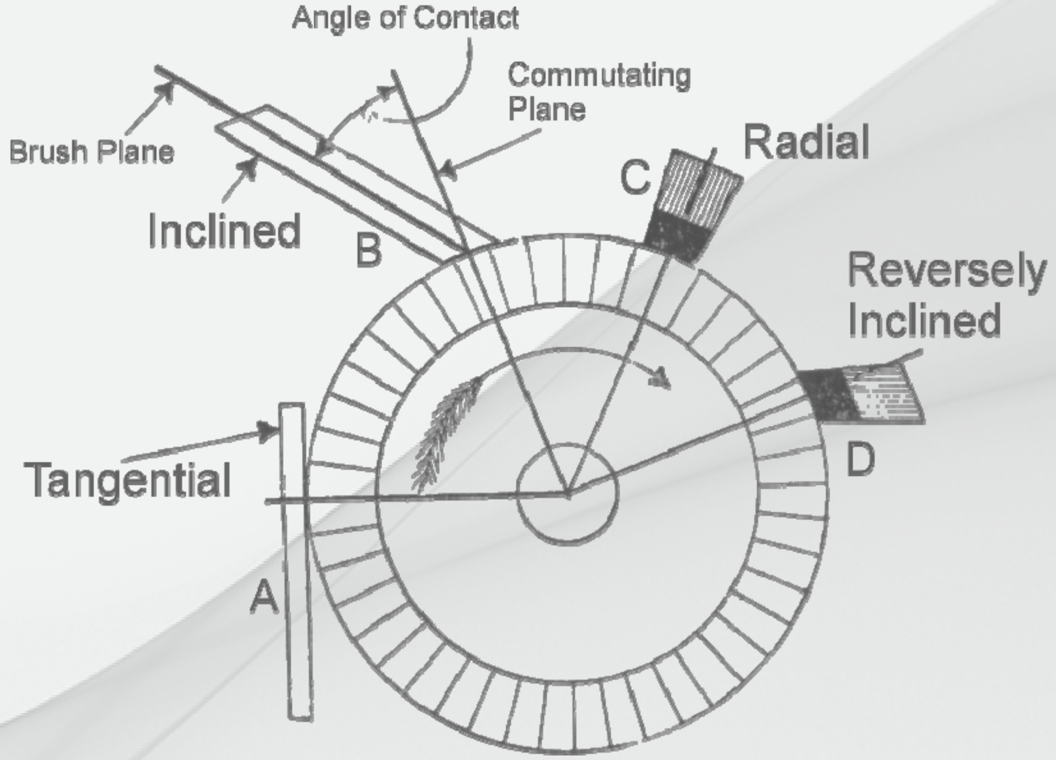
ST-Ericsson U8500 development board

- 20.9FPS (original Qt 4.7) -> 81.8FPS



zCoco.net

NOMOVOK



Thank you!
どうもありがとうございます。

For more information, please contact:

Akira Tsukamoto
Principle Architect

Akira.tsukamoto@nomovok.com
080-4426-6667