

Qt maemo hildon port, making Qt and GTK+ working together in mobile devices

Kate Alhola

NOKIA
Connecting People

Forum **NOKIA**

18.10.2008

Kate Alhola



- Maemo Chief Engineer at Forum Nokia
- Long term Open Source developer, first contributions 8-bit microprocessor in early 80's
- Linux kernel driver from early 1.x kernels
- Katix RTOS with IP stack for PC, 68K and PPC
- Multiple GUI applications with Qt and GTK (and X11/Athena, Motif ...)
- Before Nokia, long career embedded Linux and RTOS related development in small subcontractor companies
- Numerous embedded HW designs with 6809, 68xxx and PPC

What is maemo



- Maemo is optimized for mobile internet tablets with touch screen
- Maemo is not for desktops (although you can run Maemo on desktop if you like)
- Maemo is not for small screen mobile phones (without touch screen UI)
- No stuff that wont fit in your pocket
 - No hard disk
 - No full size keyboard
 - No mouse
 - No big heavy battery
 - No big screen
- No stuff that spoils your internet experience
 - No keyboard only navigation
 - No mini size screen
 - Not limited to one toolkit

Internet tablet

- Beats any mobile phone or PC in mobile internet usage
- 4.1inch 800x480 lcd touch screen
- Flash memory
- Hours of continuous internet usage
- Full multimedia capabilities, speakers, microphone, camera, codecs, DSP
- Connectivity Wlan, bluetooth, Wimax
- Alphanumeric mini keyboard and GPS in N810



Maemo Linux for internet tablets

- Small form factor 800x480 touch screen, 128MB RAM, 256M flash memory, 2GB internal flash “card”, ARM OMAP CPU
- Takes advantage of device's multimedia capabilities including DSP all device connectivity features
- Strongly optimized power management allows long battery life and small device battery size
- On screen virtual keyboard or mobile optimized small size qwerty keyboard
- maemo versions up to 4.1 Diablo was based GTK+
- Open architecture with X11 that allows to run any alternate GUI toolkit like Qt or Java – a rich development environment - this makes this environment very interesting and unique.

UI Framework progression



Fremantle
Hildon & GTK+ evolved



Harmattan
Qt integrated



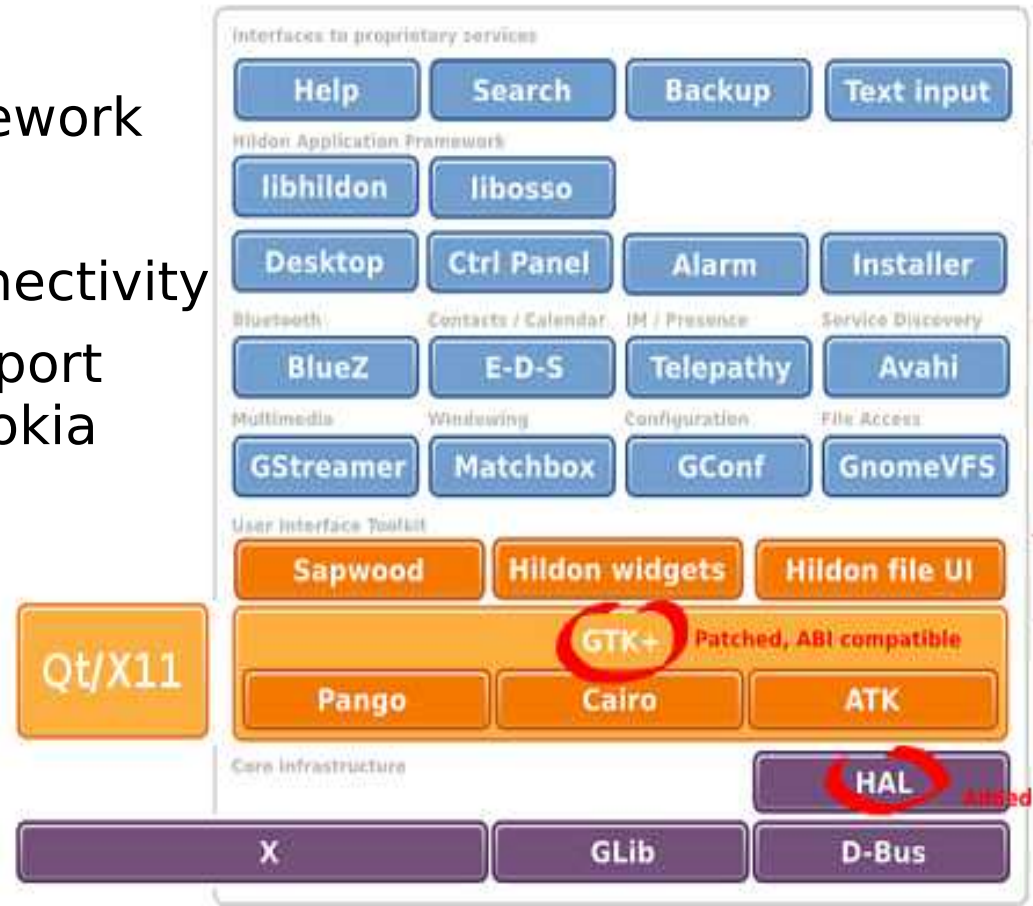
Consolidation of essential parts of the platform

Image: *Industrial-Glade*, by John Brian Silverio. CC Attribution-Non-Commercial-No-Derivative-Works License



Maemo 4 (Diablo)

- Current Maemo production release
- Nokia devices supported: N800 & N810 & N810 WiMAX edition
- OMAP 2 processor
- GTK+ toolkit with Hildon UI framework
- Maemo desktop UI
- WLAN, Bluetooth and WiMAX connectivity
- Qt 4.4 with Hildon framework support as community port from Forum Nokia



HSPA mobile broadband support :

- **Assume nothing less than always online** when designing applications that connect to the Internet

OMAP3 high-performance processor support:

- **Consider the boost of performance** enabling computing-intensive applications without user-perceived delays
- **Enabling up to 3X gain in performance** over ARM11-based processors (according to Texas Instruments)
- **Encode and decode videos at DVD resolutions**

High-definition camera sensor:

- **Build anything you can imagine assuming a camera in the device:** from camera algorithm optimizations to photo sharing on the web

Hardware-accelerated graphics with Open GL ES2.0 and Clutter:

- **Embed stunning UI transitions into your application**

.... And Qt 4.5 community port for from Forum Nokia
Dont wait to sart code cross platform applications, Qt is here
now !!



Clutter



- OpenGL based animated GUI toolkit
- Very simple, low level, but still easy. Building blocks which are straightforward to use.
- C / Glib
- API and conventions resemble that of Gtk+
- Bindings for Python, C++, Vala
- Mobile optimized as design principle
- Supports both OpenGL for desktop and OpenGL-ES in mobile
- Model: stage (=the main window & canvas) and actor (item you can press, animate etc. and use as building block widgets you write by yourself)
- Actors are objects that behave in stage
 - Move, rotate, scale, change opacity
 - Textures
 - Actor is a GObject
- Timelines and events and events control behavior
- Gstreamer media rendering as Actor
- Offers powerful and accurate timelines for animations

Clutter continued



- Clutter is a low level toolkit which does not contain any high level widgets people are used to (e.g. with Gtk+). You have got the entity called actor and have to live with that. There is for example no button, text box, list box etc. But you can make actors to behave like buttons for example.
- You have to manage the layout manually with coordinates, it is not automated
- Because of the advantage the timelines and fluid animations clutter provides, it is possible to make your UI feel more alive – e.g. if you make a button, you can animate the state change from unpressed to pressed and vice versa rather than just replacing the picture.



Clutter Continued



- There is no theming framework in Clutter. You can however do theming to clutter applications by creating graphics for each different used object size separately. It has some overhead and consumes some space, but it is not that big problem.
- You can also use clutter-cairo to draw textures that are then used as clutter actors. E.g. you can create button graphics with cairo and then make it a texture which is then used to represent a button.
- You can connect signals to actors, no matter if they are moving, rotated, etc. you can press them.
- Clutter is very optimized and efficient with detecting the user input (faster than e.g. Qt GraphicsView at the moment)

Understanding clutter



- So instead of using layout managers to manage your widgets, you place your actors where you want
- One actor can contain more than textures
- You can stack textures, and each texture can have alpha channel transparency
- There is no button, but you can do one by yourself.
- There is no list, but you can do one by yourself.
- And there is nothing else either but you can do everything by yourself, and quite quickly.
- For example: If you have two states on a button, pressed and unpressed state, you can have two textures on top of each other so that the pressed state texture is stacked on top of the button background texture, instead of replacing the button background texture.
 - This way you can create a different style button, it can be for example have highlight when pressed
 - And you can fade the highlight nicely away through alpha channel on button release if you like

Clutter examples continued



- Another example:
 - You can create a toggle button so that you have a toggle button background image and the toggle slider highlight that you stack on top of the background.
 - Then you stack on top of the textures your label.
 - When the user presses the toggle button, you can for example animate the transition from the other side to the another, it can for example slide, fade, etc. instead of just jumping to the another place like would happen on Gtk+.

Clutter is easy



- Making complex animations with clutter is not hard, even if it does not provide high level widgets, that is no problem, because using the couple basic elements it provides, you can create about anything.
- Making something move is just few lines of code
- Changing opacity is one line of code
- Rotating can be done with ease
- Fade out, fade in, show, hide, all with almost one function call
- There are predefined effect templates that can be used, and you can do basic transitions very efficiently without writing your own callbacks etc.
- Doing animations functions like a state machine. After animation has finished, a callback function is called, where your code continues. You can trigger another animation from it for example, and therefore you know exactly when the asynchronous animation finishes
- There is no limit how many actors can move at the same time other than processor speed and 3D chip speed, you can do radically different user interfaces with clutter.

Clutter is easy continued



- Simplest form of clutter actor
 - Is a texture that is loaded from a file (e.g. a png or jpg)
 - Does not need to be even reactive to user input – e.g. you can do a fake dialog background (which simulates a dialog even if it is not), and you can place e.g. your button actors (that you have to write by yourself, because clutter provides none by default as said before), on top of that.
 - Simplest reactive actor is a texture that behaves like a button, so you get clicks from the actor and your callback function gets executed like you would have on a gtk button a callback function connected to it.
 -
- And most importantly: **CLUTTER IS VERY MUCH FUN TO CODE!**

- How to make a good clutter based user interface?
 - Even though there are lots of capabilities of animations, don't overuse it.
 - Don't let the user wait animation to finish, if animation is long, it should be done so that user can interrupt it, e.g. if you have implemented a button where the transition from pressed state back to unpressed is a nice fade which does not finish immediately, user needs to be able to press another (and same button) again at any time and it needs to respond immediately
 - you can rewind the timeline in this kind of situation instead of creating a new one, if the user presses the same button actor
 - Use graphics that are clear despite you can have anything in a clutter actor (and you are not bound with a Gtk theme for example). Ideally your clutter app graphics should follow the style of the Gtk style, so these apps could live together in harmony.
 - To take full advantage of clutter, you can think out of the box. Think what the user needs in your case, and think if you can do it for example without a button, but instead so that user can directly touch the objects. With clutter, you can challenge the usual desktop user interface paradigm, if you can invent a better way, with clutter, you are not technologically limited by the toolkit that you would prevent it.

Why need to port ?



- maemo is like any Linux distro,
 - based on same standard open source components and debian packaging
- You can compile most of applications to Maemo without any modifications
 - in many cases this just don't make sense !!!!
- Maemo is mobile optimized and applications should be mobile optimized as well

Why Qt on maemo



- Qt is cross platform toolkit same applications can be compiled for Linux desktop, S60 Macintosh, Windows
- Native C++
- Python and Java bindings
- Nokia will release QT for S60
- Get KDE community applications ported on maemo

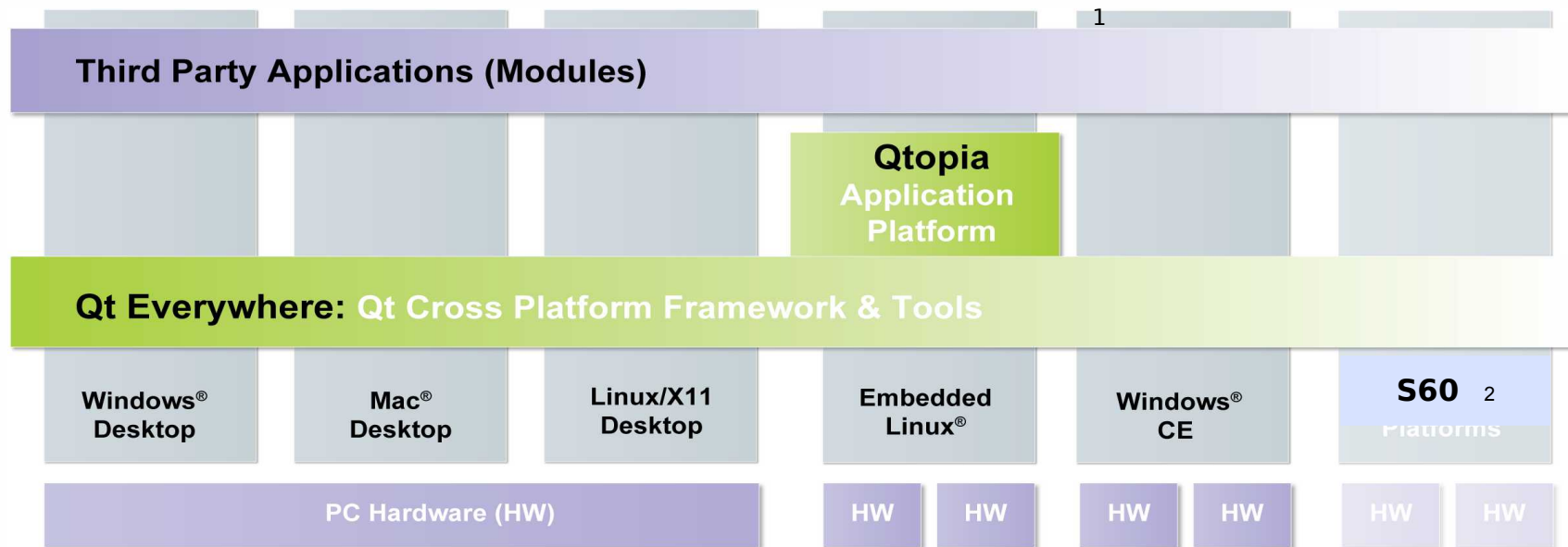
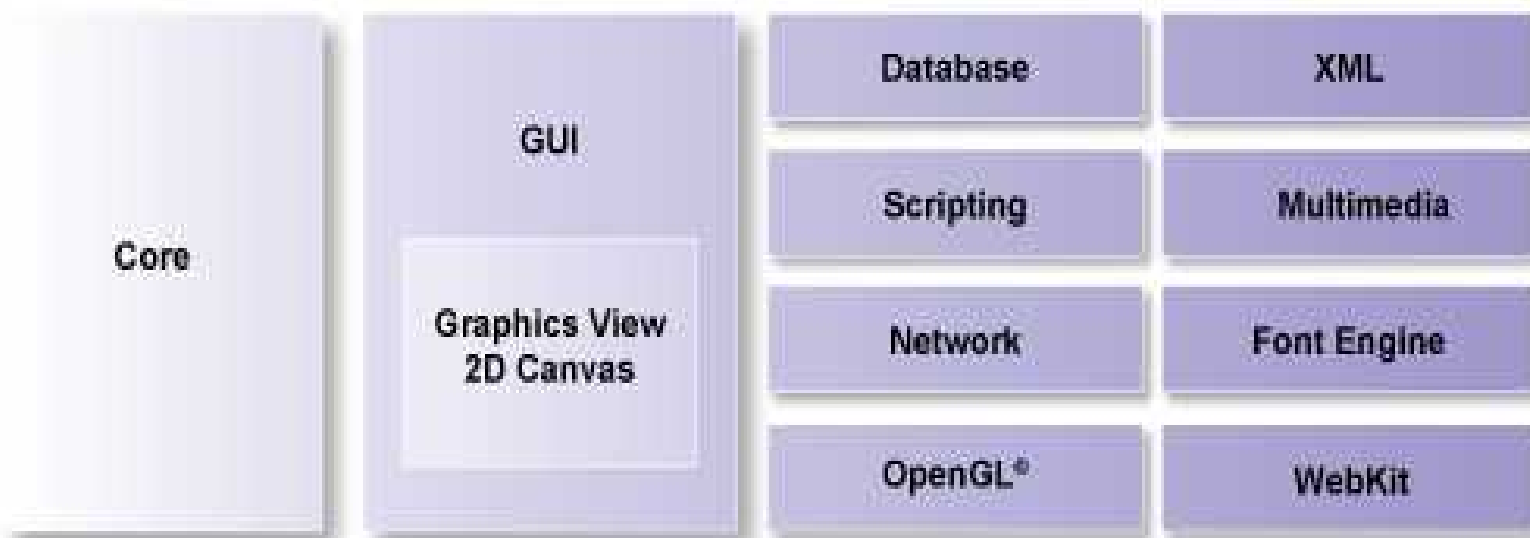


Diagram simplified

- Qt is just not a GUI library but complete collection of cross platform libraries
- Integrates features used to be many separate libraries with incompatible API's to common set
-



Qt GraphicsView



- Qt way to implement animated UI
- Similar functionality than Clutter
- GraphicsView is a low level toolkit which does not contain any high level widgets people are used to. You can make QGraphicsItems to behave like animated buttons for example or you can embed 2D QT Widgets but they lack lack animated behavior
- You have to manage the layout manually with coordinates, it is not automated
- QGraphicsScene, QGraphicsItem and QtimeLine
- Supports both OpenGL for desktop and OpenGL-ES in mobile
- GraphicsItems are objects that bahave in stage
 - Move, rotate, scale, change opacity
 - Textures, SVG, vectors
- Timelines and events and events control behavior
- Normal 2D Qt widgets can be embedded and transformed as GraphicsItem

GraphicsView Example



```
RPixmap::RPixmap(const QPixmap &pixmap): QGraphicsPixmapItem(pixmap)
{
    setAcceptsHoverEvents(true);

    timeLine.setDuration(180*4);
    timeLine.setFrameRange(0, 180*4);
    connect(&timeLine, SIGNAL(frameChanged(int)), this, SLOT(setFrame(int)));
}

void RPixmap::setFrame(int frame)
{
    QPointF center = boundingRect().center();
    resetMatrix();
    int rof=50;

    setTransform(QTransform().translate(rof,rof).rotate(frame / 4.0,Qt::XAxis).translate(-rof,-
rof).scale(0.1 , 0
.1 ));
}

void RPixmap::mousePressEvent ( QGraphicsSceneMouseEvent * event )
{
    if (timeLine.state() == QTimeLine::NotRunning)
        timeLine.start();
}
}
```

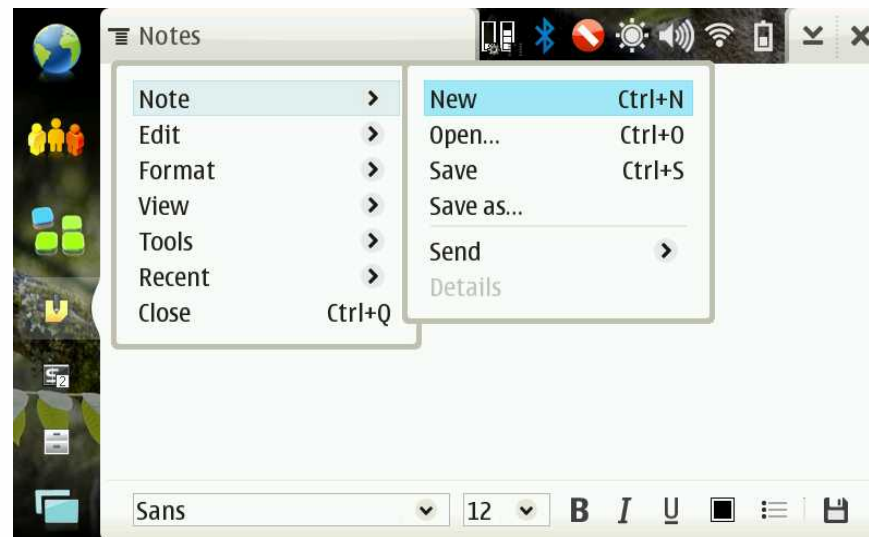
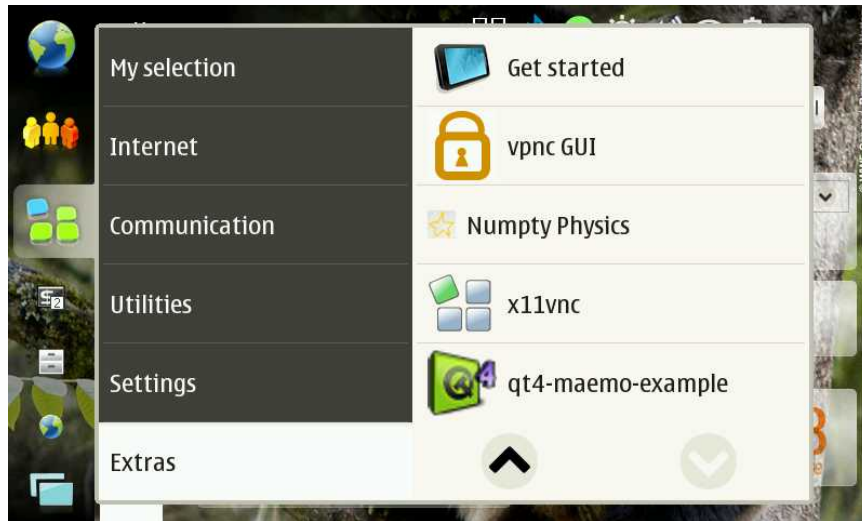
What is needed for maemo Qt

- Hildon menu
- Hildon input method
- maemo theming
- Debian packing for maemo
- Gnome VFS (or GVFS or KIO)
- Optimizing widgets for mobile screen
- OpenGL-ES support

Screen layout



Mobile optimized UI

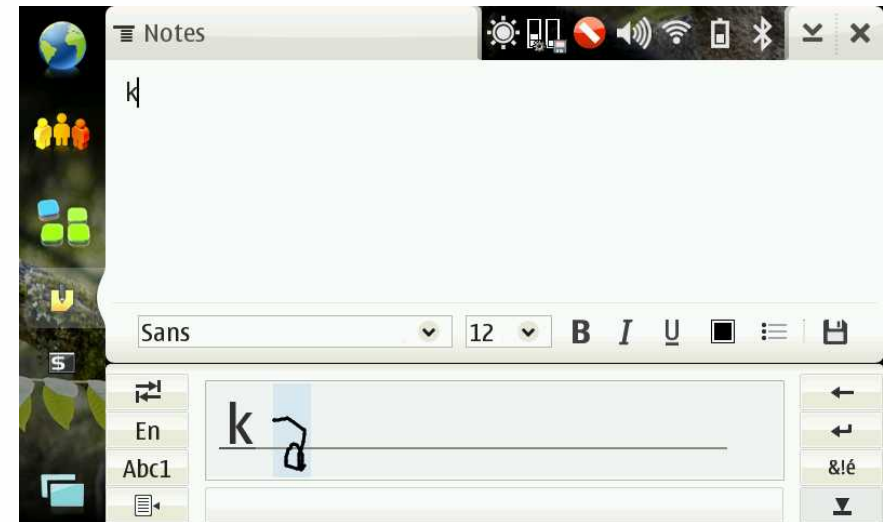


Multiple forms of IM

Virtual keyboard



Handwriting mode



Thumb mode

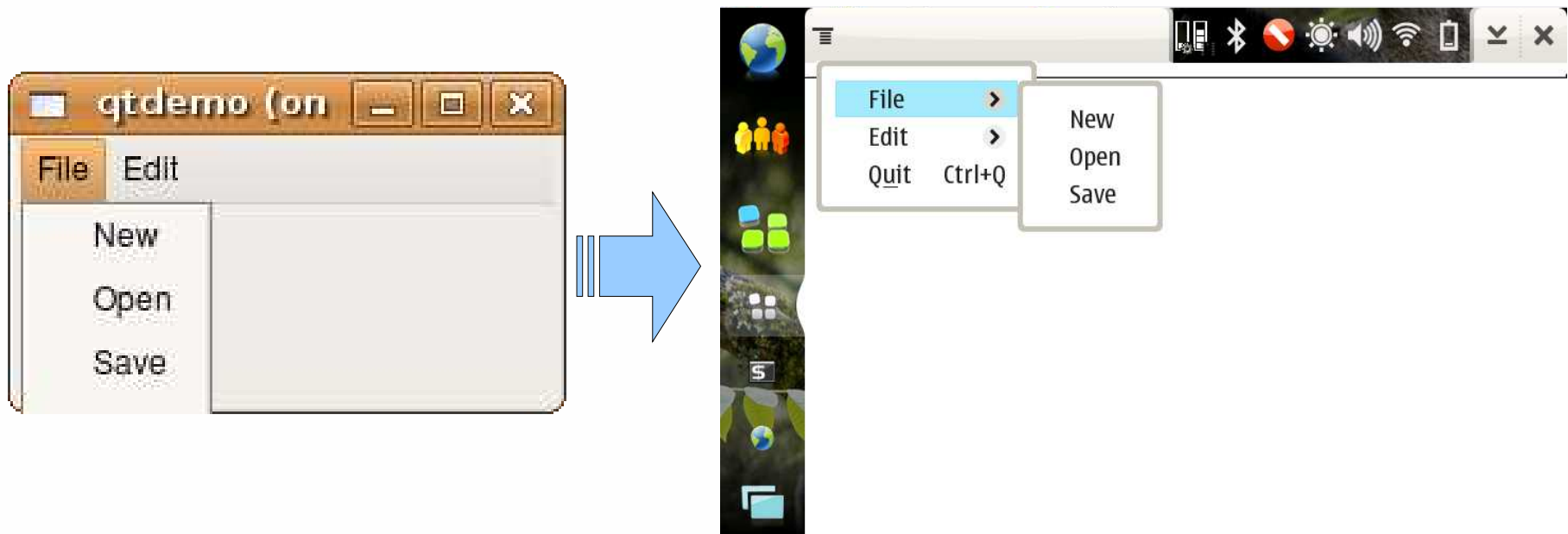


Qwerty



Maemo menu

- Application main menu is not in application owned window, it is in window manager/desktop owned area
- Application receives X11 Grab transfer event to show menu
- In full screen mode, the menu bar is not visible
- Tablet has hardware menu button, keycode as F4
- No changes are required to existing applications



Input method

- tablet keyboard uses normal X key events
 - there are many mandatory features that need to be supported
- small keyboard has a different modifier configuration
- Modifiers have three modes, “**traditional**” pressed same time with the key, “**sticky**”, pressed before key or “**lock**” pressed twice and modifier is active until pressed third time.
- Fn modifier to access numbers and special characters
- Char modifier to access some characters not in keyboard
- Auto completion mode
- Input mode can be changed by application



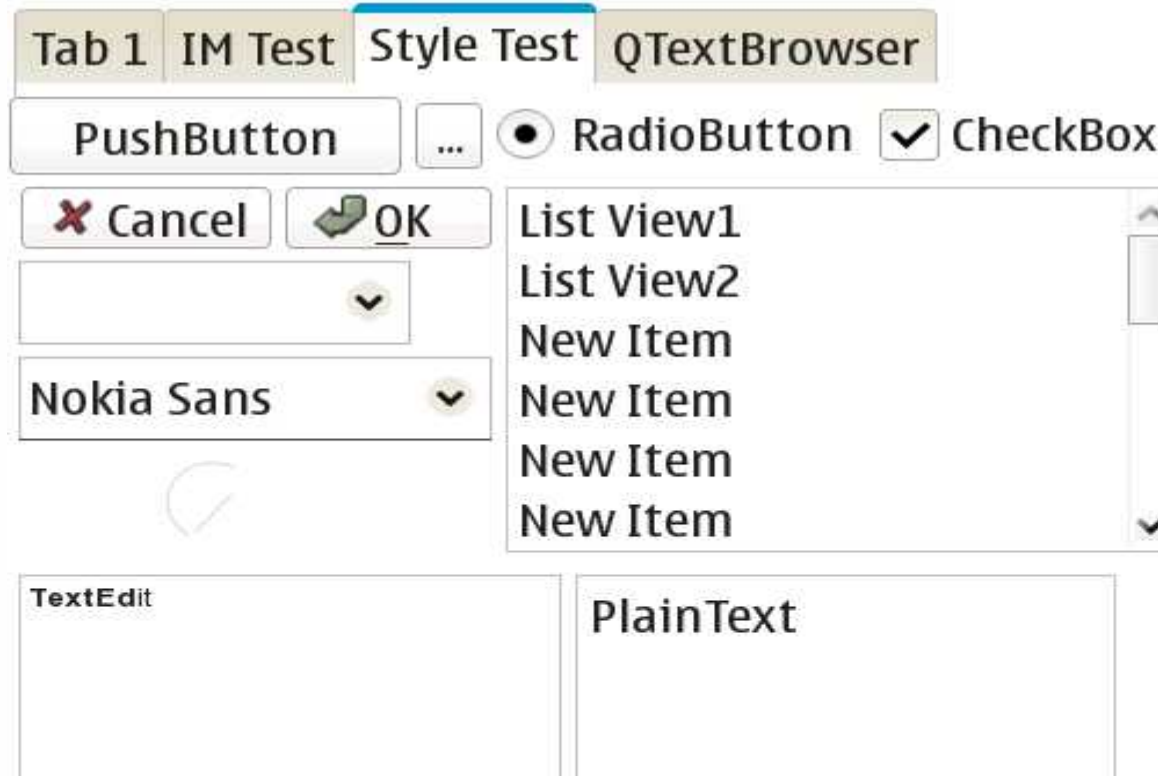
Input Method

- The user gives the focus to a key entry widget
- A new window appears at the bottom of the screen



Maemo styles

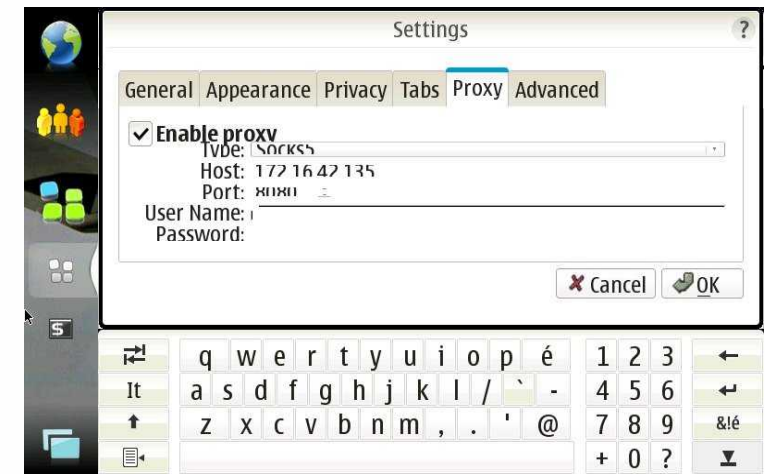
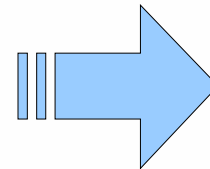
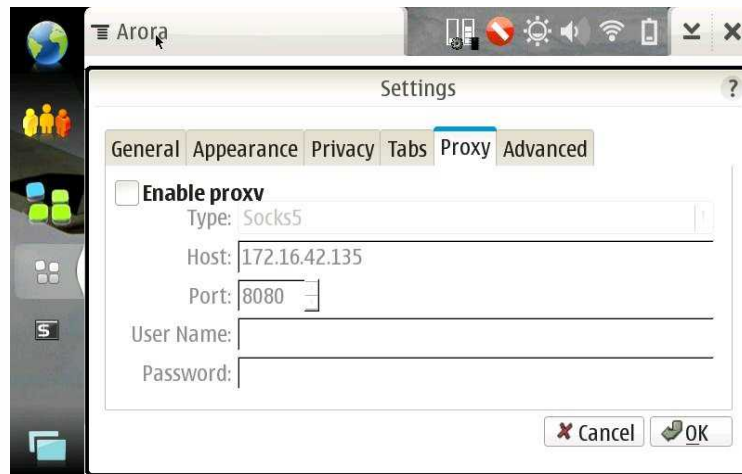
- Maemo uses optimized sapwood theming engine
- Maemo Qt uses sapwood themes
- Themes are optimized for small screen, larger sizes, larger fonts



- Qt application in Maemo uses QGTKStyle to change its Look & Feel
- QGTKStyle:
 - is a Qt style rendered using GTK
 - uses QCleanLookStyle as base style
- In Qt for Maemo, QGTKStyle:
 - has been integrated
 - is the default style
 - supports hildon specific theme element
(e.g. QSpinBox looks like a HildonNumberEditor).

When porting application to maemo

- Remember that the screen is small
- Remember, that user is using finger or stylus, try to allow using finger always when possible
- Allow scrolling by finger from content pane, not from small scrollbar with stylus
- Don't use absolute layout, in maemo style, fonts, buttons etc are much larger than in desktop styles
- When on screen virtual keyboard is displayed, arrange your dialogs and inputs so that they will fit in same screen.



- Scratchbox
- It is better to fix tools that fix every package for cross compilation
- masquerades host system and tools look a like target system
- can compile most standard packages without modifications
- Just say “apt-get source” for unmodified debian/ubuntu package and “dpkg-buildpackage” will do the work
- Or download source and say “configure” and “make”
- It works in most cases but of course it can't fix broken packages or insane dependencies. Typical example of problematic package is one that is depending some specific documentation tools that depends about everything in universe.
- Some limitations due limitations of qemu

Scratchbox 1



- chrooted environment
- Target filesystem image called “rootstrap”
- Toolset linked inside of the rootstrap target image
- Running mixed binaries, most tools as host CPU native, target code as target native with user mode QEMU
- Host system kernel is used
- dpkg-buildpackage, autotools, configure and compilation run inside of the chrooted rootstrap filesystem system that looks a like target device filesystem with the dev tools and files added

- Scratchbox
- It is better to fix tools that fix every package for cross compilation
- masquerades host system and tools look a like target system
- can compile most standard packages without modifications
- Just say “apt-get source” for unmodified debian/ubuntu package and “dpkg-buildpackage” will do the work
- Or download source and say “configure” and “make”
- It works in most cases but of course it can't fix broken packages or insane dependencies. Typical example of problematic package is one that is depending some specific documentation tools that depends about everything in universe.
- Some limitations due limitations of qemu

Installing maemo SDK

- Download installer scripts and INSTALL.tx from page <http://maemo.org/development/sdks/maemo-4-1-diablo-sdk/>
- http://tablets-dev.nokia.com/4.1/maemo-sdk-install_4.1.sh
- http://tablets-dev.nokia.com/4.1/maemo-scratchbox-install_4.1.sh
- <http://tablets-dev.nokia.com/4.1/INSTALL.txt>

```
$ sudo chmod a+x ./maemo-scratchbox-install_4.1.sh
$ sudo ./maemo-scratchbox-install_4.1.sh
$ /scratchbox/sbin/sbox_adduser <your linux username>
$ sh maemo-sdk-install_4.1.sh
$ apt-get install xserver-xephyr
```

- Xephyr server is needed to run graphical maemo environment
- You can also check maemo4mobile document installing chapter from http://maemo4mobile.garage.maemo.org/development_environment.html

Some tweaks for Hardy



- At least in Ubuntu Hardy you may need some tweaks (same may apply to Intrepid 8.10 which is soon to be released)
- More about them in INSTALL.txt
- You can find also some installation tips from Karoliina's Maemo blog: <http://karoliinamaemoblog.blogspot.com/>
-

```
/etc/sysctl.conf
```

```
vm.vdso_enabled = 0  
vm.mmap_min_addr = 4096  
net.ipv4.ip_local_port_range = 1024 65535  
and running 'sysctl -p' as root.
```

Using SB

- Login in to Scratchbox

```
kate@katti:~$ /scratchbox/login
```

```
Welcome to Scratchbox, the cross-compilation toolkit!  
Use 'sb-menu' to change your compilation target.  
See /scratchbox/doc/ for documentation.
```

Your scratchbox home ([sbox-DIABLO_ARMEL: ~] >
/scratchbox/users/<username>/home/<username>/

- I recommend just make symlink there from your home directory

```
kate@katti:~$ ln -s /scratchbox/users/kate/home/kate/ scratchbox  
kate@katti:~$ ls -l scratchbox  
lrwxrwxrwx 1 kate kate 33 2007-05-09 16:55 scratchbox ->  
/scratchbox/users/kate/home/kate/  
kate@katti:~$
```

Use it as any Linux ()
just remember that your favourite editors or GUI tools
may be unavailable inside of scratchbox environment
and you should use them from your system shell

Running maemo under scratchbox

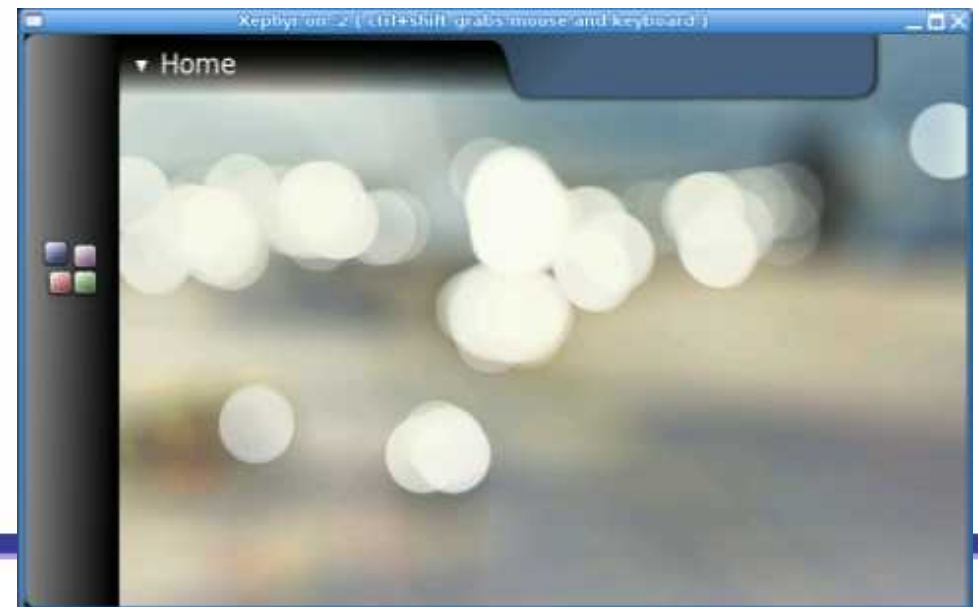
- To have graphical maemo window, you need to start xephyr in your host linux command prompt

```
$ Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac
```

Start maemo desktop inside scratchbox

```
[sbox-DIABLO_ARMEL: ~] > export DISPLAY=:2  
[sbox-DIABLO_ARMEL: ~] > af-sb-init.sh start
```

- And you get a cute maemo desktop in your xephyr window
- Just notice that there are by default much less applications or applets installed than in device
- You can install more using apt-get



Installing extra libraries

- C++ / GTKmm and QT needs some extra libraries to be installed in Scratchbox and in device
- Add extras-devel repository to /etc/apt/sources.list INSIDE of Scratchbox.

```
[sbox-DIABLO_ARMEL: ~] > vi /etc/apt/sources/list
deb-src http://repository.maemo.org/ diablo free
deb http://repository.maemo.org/extras/ diablo free
deb-src http://repository.maemo.org/extras/ diablo free
deb http://repository.maemo.org/extras-devel/ diablo free
deb-src http://repository.maemo.org/extras-devel/ diablo free
```

```
[sbox-DIABLO_ARMEL: ~] > fakeroot apt-get update
[sbox-DIABLO_ARMEL: ~] > fakeroot apt-get upgrade
[sbox-DIABLO_ARMEL: ~] > fakeroot apt-get install libqt4-devel
[sbox-DIABLO_ARMEL: ~] > fakeroot apt-get install libhildonmm-dev
libhildon-fmmm-dev
```

- In future releases, use extras in place of extras-devel

Helloworld Qt



- Write your favourite hello world application under scratchbox directory
- You can use any editor in host OS outside of scratchbox
- Under scratchbox run qmake and make
- You got a nice ARM binary

```
#include <QApplication>
#include <QtGui/QPushButton>

int main( int argc, char ** argv )
{
    QApplication a( argc, argv );
    QPushButton *w = new
QPushButton("Hello maemo");
    w->show();
    return a.exec();
}
```

```
[sbox-DIABLO_ARMEL: ~/qtdemo/hello] > qmake -project
[sbox-DIABLO_ARMEL: ~/qtdemo/hello] > qmake
[sbox-DIABLO_ARMEL: ~/qtdemo/hello] > make
g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB
-DQT_SHARED -I/targets/DIABLO_ARMEL/usr/share/qt4/mkspecs/linux-g++ -I.
-I/targets/DIABLO_ARMEL/usr/include/qt4/QtCore -I/targets/DIABLO_ARMEL/usr/include/qt4/
QtCore
-I/targets/DIABLO_ARMEL/usr/include/qt4/QtGui
-I/targets/DIABLO_ARMEL/usr/include/qt4/QtGui -I/targets/DIABLO_ARMEL/usr/include/qt4 -
I. -I. -I. -o hello.o hello.cpp
g++ -o hello hello.o -L/usr/lib -lQtGui -lQtCore -lpthread
[sbox-DIABLO_ARMEL: ~/qtdemo/hello] > file hello
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), for GNU/Linux 2.6.8,
dynamically linked (uses shared libs), not stripped
```

- Qt uses .pro -file for the same purpose as automake files are used in the gtk world.
- However, one major difference is that you can create the project pretty much automatically whereas on gtk-world you write files needed for compilation by yourself. qmake -project creates / recreates the project file whenever needed.
- Project file syntax is very simple and easy to learn and understand

Running applications under scratchbox

- When you have already started you xephyr and hildon application framework, you may just run your application as any linux application.
- Use run-standalone.sh that you get It with maemo theme

```
[sbox-DIABLO_ARMEL: ~/qtdemo/hello] > run-standalone.sh ./hello
```

Application is running under qemu, you may get some qemu error messages about some incompatibilities but in most cases you don't need care about these



Running application in device

- Install Qt libraries to device using apt-get same way as in scratchbox
- Copy your application from pc to device using

```
• scp hello root@192.168.1.1:/home/user
```

- Connect with USB cable and copy it to on device memory card
- You need to disconnect USB before running app from memory card because card has exclusive access to tablet or PC and it can't be shared

- Run your application from tablet terminal

```
~$ cd /home/user  
~$ ./hello
```

- To get root access in device use flasher with `-enable-rd-mode` then you can issue `sudo gainroot` from terminal

```
~$ sudo gainroot  
/home/user #
```

Useful Links:

- Maemo Qt4 project website:

<http://qt4.garage.maemo.org>

- Maemo Qt4 Developer ML:

<https://garage.maemo.org/mailman/listinfo/qt4-devel>

- Maemo for mobile programmers

<http://maemo4mobile.garage.maemo.org/>