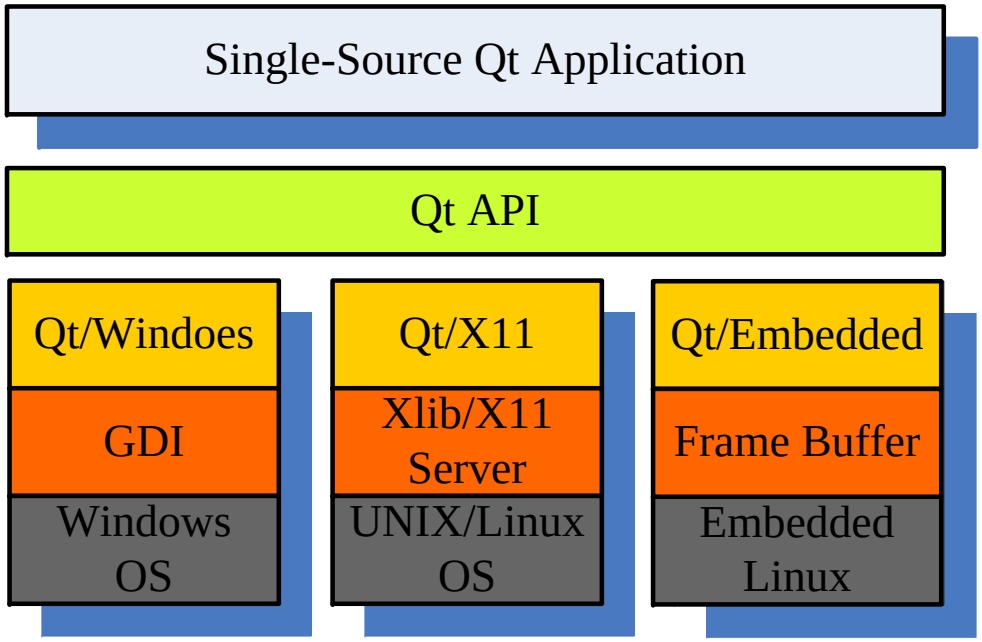
Qt Programming

課程：嵌入式系統與軟體工程
開發學校：台大電機系
王勝德 教授

What is Qt

- The Qt toolkit is a C++ class library and GUI toolkit for building multiplatform GUI programs using a "write once, compile anywhere" approach.



About Qt

- The Qt toolkit first became publicly available in May 1995. (Qt 0.9)
- Initially developed by Haavard Nord (Trolltech's CEO) and Eirik Chambe-Eng (Trolltech's president)
- Qt could be used for both Windows and Unix development, offering the same API on both platforms.
- Qt was available under two licenses from day one:
  - A commercial license was required for commercial development.
  - A free software edition was available for open source development.
- Matthias Ettrich's decision to use Qt to build KDE helped Qt become the de-facto standard for C++ GUI development on Linux. (Qt 1.2 was released in April 1997.)

About Qt (Cont.)

- Qt 2.0 was released in June 1999
  - Using the Q Public License (QPL).
  - Won the LinuxWorld award for best library/tool in Aug. 1999.
- Qt/Embedded was released in 2000
  - Designed to run on Embedded Linux devices.
  - Providing its own window system as a lightweight replacement for X11.
  - Won the LinuxWorld “Best Embedded Linux Solution” award in both 2001 and 2002.
- Qt 3.0 was released in 2001
  - available on Windows, Unix, Linux, Embedded Linux, and Mac OS X.
  - won the Software Development Times “Jolt Productivity Award” in 2002.
- Newest version is Qt 4.1
  - http://www.trolltech.com

INSTALLING Qt/Embedded

- Simple approach

```
tar zxvf qt-embedded-2.3.8.tar.gz
cd /usr/local/src/qt-2.3.8
export QTDIR=/usr/local/src/qt-2.3.8
export LD_LIBRARY_PATH=/usr/local/src/qt-2.3.8/lib
:$LD_LIBRARY_PATH
./configure
make
```

Building..

```
[root@Cedric ~]# cd /usr/local/src
[root@Cedric src]# ls
qt-2.3.8                      qt-x11-opensource-src-4.1.2
qt-embedded-2.3.8.tar.gz      qt-x11-opensource-src-4.1.2.tar.gz
[root@Cedric src]# tar zxvf qt-embedded-2.3.8.tar.gz
```

```
[root@Cedric qt-2.3.8]# export QTDIR=/usr/local/src/qt-2.3.8/
[root@Cedric qt-2.3.8]# export LD_LIBRARY_PATH=/usr/local/src/qt-2.3.8/lib/


[root@Cedric qt-2.3.8]# ./configure -gif -system-jpeg -thread

Usage: ./configure [-debug] [-release] [-shared] [-static] [-gif] [-no-gif] \
        [-sm] [-no-sm] [-thread] [-no-thread] [-qt-zlib] [-system-zlib] \
        [-qt-libpng] [-system-libpng] [-no-jpeg] [-system-jpeg] \
        [-no-<module>] [-kde] [-Istring] [-Lstring] [-Rstring] [-lstring]
```

Building.. (Cont.)

```
This is the Qt/Embedded Free Edition.

You are licensed to use this software under the terms of
the GNU General Public License (GPL).

Type 'G' to view the GNU General Public License.
Type 'yes' to accept this license offer.
Type 'no' to decline this license offer.

Do you accept the terms of the license?
yes

Choose a feature configuration:

  1. Minimal (890 kB)
  2. Small (1.7 MB)
  3. Medium (2.3 MB)
  4. Large (3.4 MB)
  5. Everything (5.6 MB)
  6. Your own local configuration (src/tools/qconfig-local.h)

Sizes are stripped dynamic 80386 build. Static builds are smaller.

Your choice (default 5):
5
```

Building.. (Cont.)

```
Choose pixel-depths to support:

   v. VGA-16 - also enables 8bpp
   4. 4bpp grayscale - also enables 8bpp
   8. 8bpp
  16. 16bpp
  24. 24bpp - also enables 32bpp
  32. 32bpp

Each depth adds around 100Kb on 80386.

Your choices (default 8,16):
16

Enable Qt Virtual Framebuffer support for development on X11 (default yes)
yes
```

Building.. (Cont.)

```
Building on:   linux-x86-g++-static
Building for:  linux-arm-g++-static

Thread support..................... yes
GIF support........................ yes
MNG support........................ no
JPEG support....................... no

Creating makefiles...
Qt is now configured for building. Just run make.
To reconfigure, run make clean and configure.
```

```
[root@localhost qt-2.3.8]# make

make[1]: Entering directory `/usr/local/src/qt-2.3.8'
cd src/moc; make
make[2]: Entering directory `/usr/local/src/qt-2.3.8/src/moc'
g++  -c -I/usr/local/src/qt-2.3.8/include -Wno-unused -Wno-parentheses
ptions -fno-rtti -O2 -fno-default-inline -Wall -W -DNO_DEBUG -DQT_NO_C
E -DQT_BUILDING_MOC -I../../include -I../tools -I. -o mocgen.o mocgen.
```

Congratulations!!

```
g++ -L/usr/local/src/qt-2.3.8/lib -Wl,-rpath,/usr/local/src/qt-2.3.8/lib   -o ./launcher lau
ncher.o   -lqte -lm
make[4]: Leaving directory `/usr/local/src/qt-2.3.8/examples/launcher'
make[3]: Leaving directory `/usr/local/src/qt-2.3.8/examples'
make[2]: Leaving directory `/usr/local/src/qt-2.3.8/examples'

The Qt library is now built in ./lib
The Qt examples are built in the directories in ./examples
The Qt tutorials are built in the directories in ./tutorial

Note: be sure to set $QTDIR to point to here or to wherever
      you move these directories.

Enjoy!    - the Trolltech team

make[1]: Leaving directory `/usr/local/src/qt-2.3.8'
[root@localhost qt-2.3.8]#
```

- But there are something else we still need…

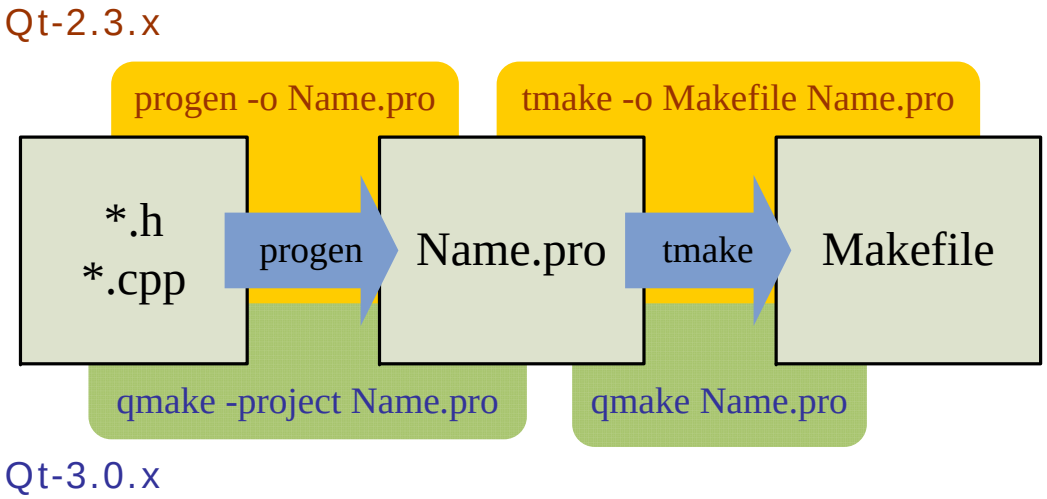
Tmake

- Download tmake package from Trolltech Co., Ltd.
  - ftp://ftp.trolltech.com/freebies/tmake/
- Unpack the package & set environment variables

```
tar zxvf tmake-1.8.tar.gz
mv tmake-1.8/* $QTDIR/tmake
export TMAKEDIR=$QTDIR/tmake
export TMAKEPATH=$TMAKEPATH/lib/qws/linux-x86-g++
export PATH=$TMAKEPATH/bin:$PATH
```

- We need the two utilities contained in $TMAKEDIR/bin under Qt-2.3.x
  - progen
  - tmake
- After Qt-3.0.x, qmake is used to substitute for progen & tmake

Develop Procedure



The Starting Hello..

It's me again!!

```
#include <qapplication.h>
#include <qlabel.h>

int main(int argc, char *argv[])
{
        QApplication app(argc,argv);
        QLabel hello("Hello World",0);

        app.setMainWidget(&hello);
        hello.show();

        return app.exec();
}
```

./hello -qws

The Signals & Slots

- Signals and slots are used for communication between objects.
- All classes that inherit from QObject or one of its subclasses can contain signals and slots.
- Signals are emitted by objects when they change their state, but It does not know if anything is receiving the signal at the other end.
- Slots can be used for receiving signals, but they are normal member functions and they have no idea if any signal(s) connected to them.
- Support for component programming using signals and slots.

The Signals & Slots (Cont.)

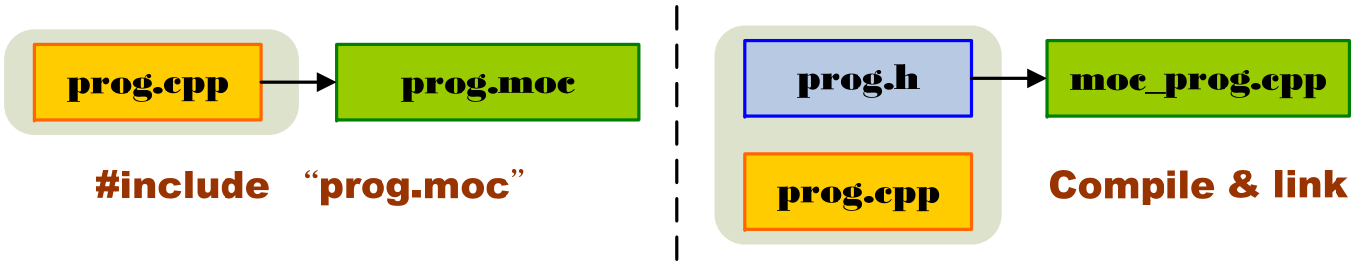
- All classes that contain signals and/or slots must mention Q_OBJECT in their declaration.
- You can connect as many signals as you want to a single slot, and a signal can be connected to as many slots as you desire. It is even possible to connect a signal directly to another signal. (This will emit the second signal immediately whenever the first is emitted.)

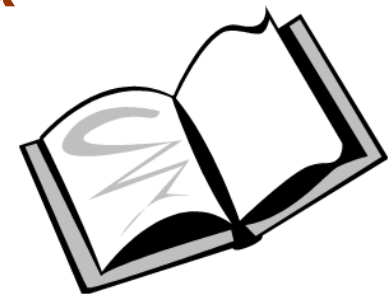
```
class Foo : public QObject
{
    Q_OBJECT
public:
    Foo();
    int  value() const { return val; }
public slots:
    void setValue( int );
signals:
    void valueChanged( int );
private:
    int  val;
};
```

- **The meta object compiler (moc) parses the class declaration in a C++ file and generates C++ code that initializes the meta object.**

The Meta Object Compiler

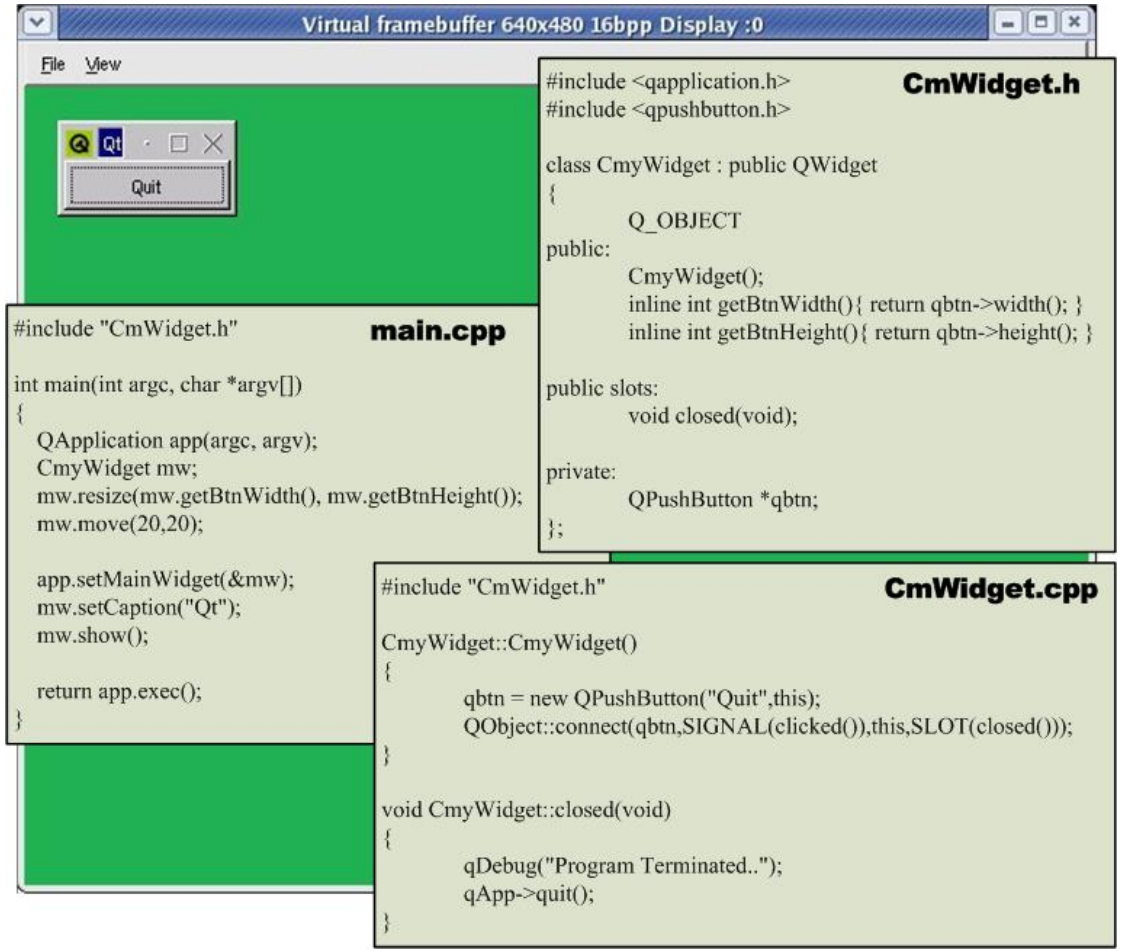
- Meta object code is required for the signal/slot mechanism, runtime type information and the dynamic property system.
- If it finds one or more class declarations that contain the "Q_OBJECT" macro, it produces another C++ source file which contains the meta object code for this class.
- Different Ways to use the MOC:





- http://doc.trolltech.com/2.3/moc.html

More Complicated..



CmWidget.h

```
#include <qapplication.h>
#include <qpushbutton.h>

class CmyWidget : public QWidget
{
        Q_OBJECT
public:
        CmyWidget();
        inline int getBtnWidth(){ return qbtn->width(); }
        inline int getBtnHeight(){ return qbtn->height(); }

public slots:
        void closed(void);

private:
        QPushButton *qbtn;
};
```

CmWidget.cpp

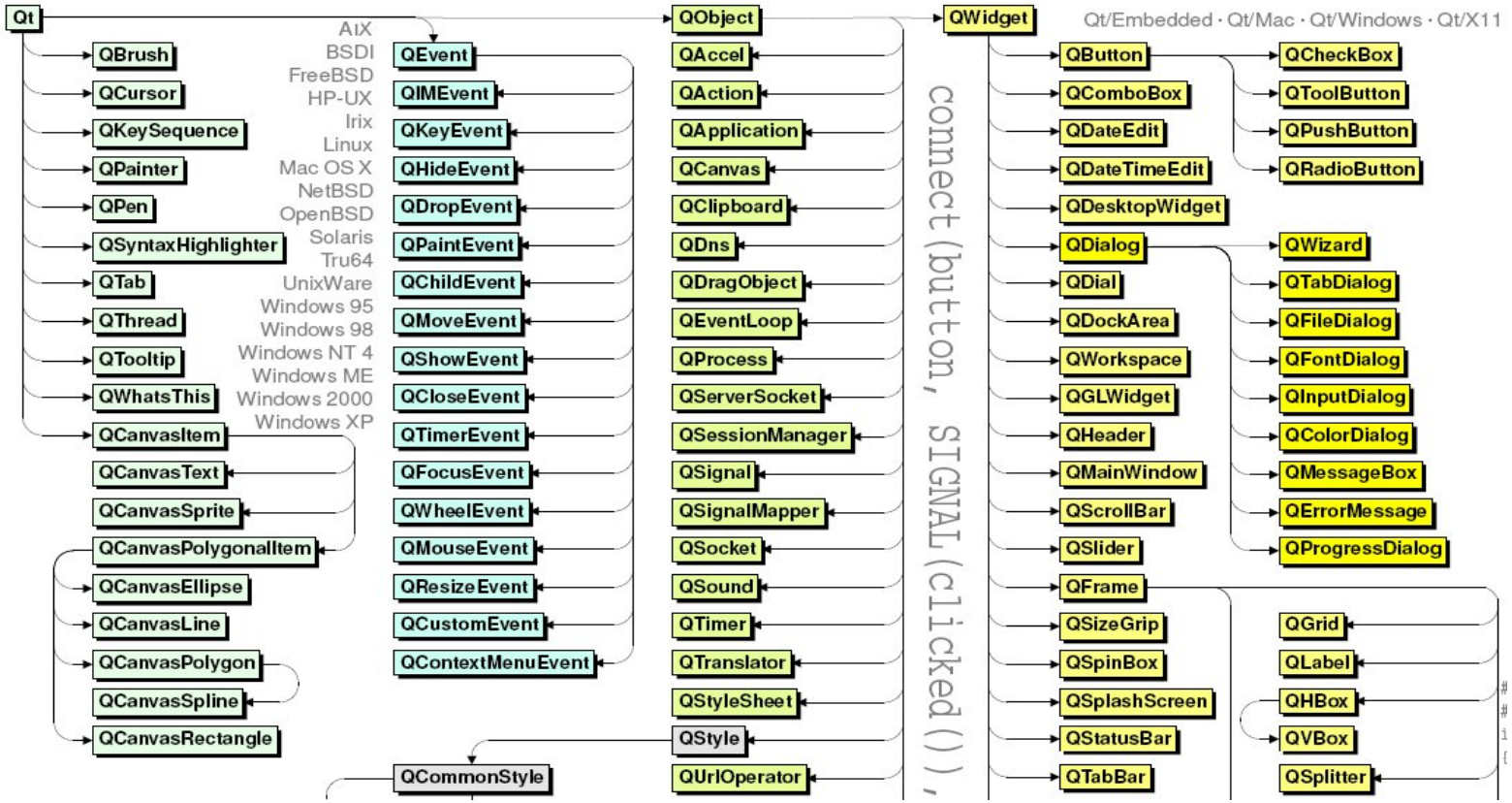
```
#include "CmWidget.h"

CmyWidget::CmyWidget()
{
        qbtn = new QPushButton("Quit",this);
        QObject::connect(qbtn,SIGNAL(clicked()),this,SLOT(closed()));
}

void CmyWidget::closed(void)
{
        qDebug("Program Terminated..");
        qApp->quit();
}
```

Main.cpp

```
#include "CmWidget.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    CmyWidget mw;
    mw.resize(mw.getBtnWidth(), mw.getBtnHeight());
    mw.move(20,20);

    app.setMainWidget(&mw);
    mw.setCaption("Qt");
    mw.show();

    return app.exec();
}
```

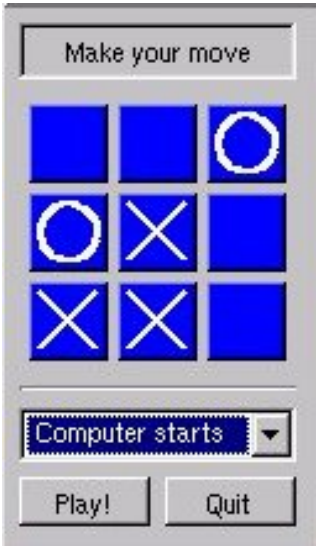
The Class Hierarchy

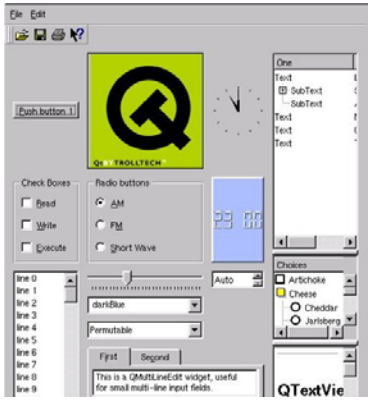
- http://doc.trolltech.com/3.3/classchart.html

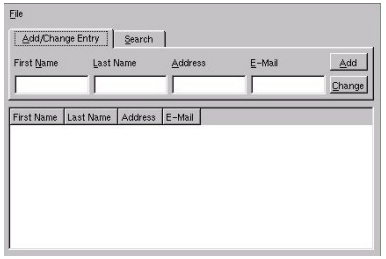


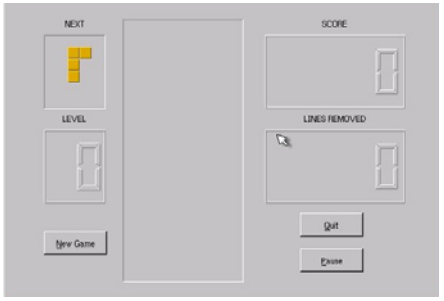
Qt Examples..

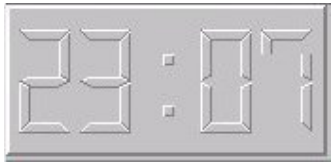
- It’s a good way to learn Qt programming by its examples… lots of examples…

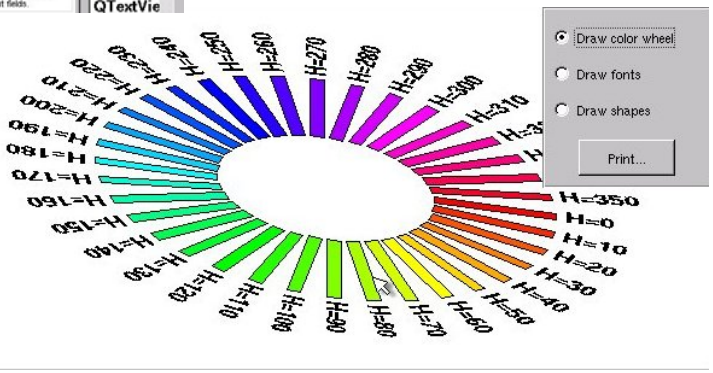








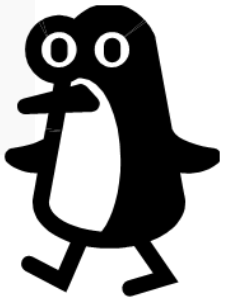




Frame Buffer

- Make sure that your kernel support framebuffer
- Add framebuffer options in your grub.conf
  - Kernel /vmlinux-2.6.5-1.358 ro root=LABEL=/ rhgb quiet

顏色	640x400	640x480	800x600	1024x768	1152x864	1280x1024	1600x1200
4 bits	?	?	0x302	?	?	?	?
8 bits	0x300	0x301	0x303	0x305	0x161	0x307	0x31C
15 bits	?	0x310	0x313	0x316	0x162	0x319	0x31D
16 bits	?	0x311	0x314	0x317	0x163	0x31A	0x31E
24 bits	?	0x312	0x315	0x318	?	0x31B	0x31F
32 bits	?	?	?	?	0x164	?	

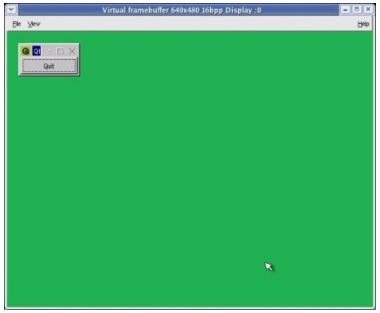


- How-To On GOT Wiki
  - http://wiki.gentoo.tw/mediawiki/index.php/HOWTO_FB

Qt Virtual FrameBuffer

- The virtual framebuffer allows Qt/Embedded programs to be developed on your desktop machine, without switching between consoles and X11.
- The virtual framebuffer is located in

  $QTDIR/tools/qvfb



- Compile qvfb as a *normal Qt/X11 application* and run it - do not compile it as a Qt/Embedded application.
- Online Reference
  - http://www.handhelds.org/~zecke/apidocs/qt/qvfb-qws.html
  - http://www.mizi.co.kr/developer/s3c2410x/Linuette_chap_7_en.htm

Tips

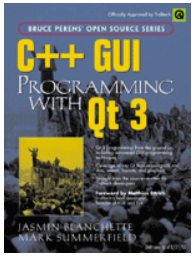
- We can put all the settings of environment variables into a shell script file, say qt2-env.sh, and source it before your work session (using command source qt2-env.sh).

```
export QTDIR=/usr/local/src/qt-2.3.8
export LD_LIBRARY_PATH=/usr/local/src/qt-2.3.8/lib export
export TMAKEDIR=$QTDIR/tmake
export TMAKEPATH=$QTDIR/tmake/lib/qws/linux-x86-g++
export PATH=$QTDIR/tmake/bin:$PATH
```

- If the cursor doesn't move
  - export QWS_MOUSE_PROTO=MouseMan:/dev/input/mouse0
  - export QWS_MOUSE_PROTO=TPanel:/dev/input/event0
  - http://doc.trolltech.com/qtopia4.1/html/qtopiacore-envvars.html
- **NOTE** You need to terminate Qt programs normally under framebuffer, or your screen will halt…

Reference

- Trolltech Online Reference Documentation
  - http://doc.trolltech.com/



- C++ GUI Programming with Qt 3
  - Written by Trolltech software engineer Jasmin Blanchette and Trolltech's documentation manager Mark Summerfield .
  - http://www.magiclinux.org/people/baif/C++ GUI Programming with Qt 3/fromURLs.html

- Installing Qt/Embedded Online Documentation
  - http://doc.trolltech.com/2.3/install-qws.html