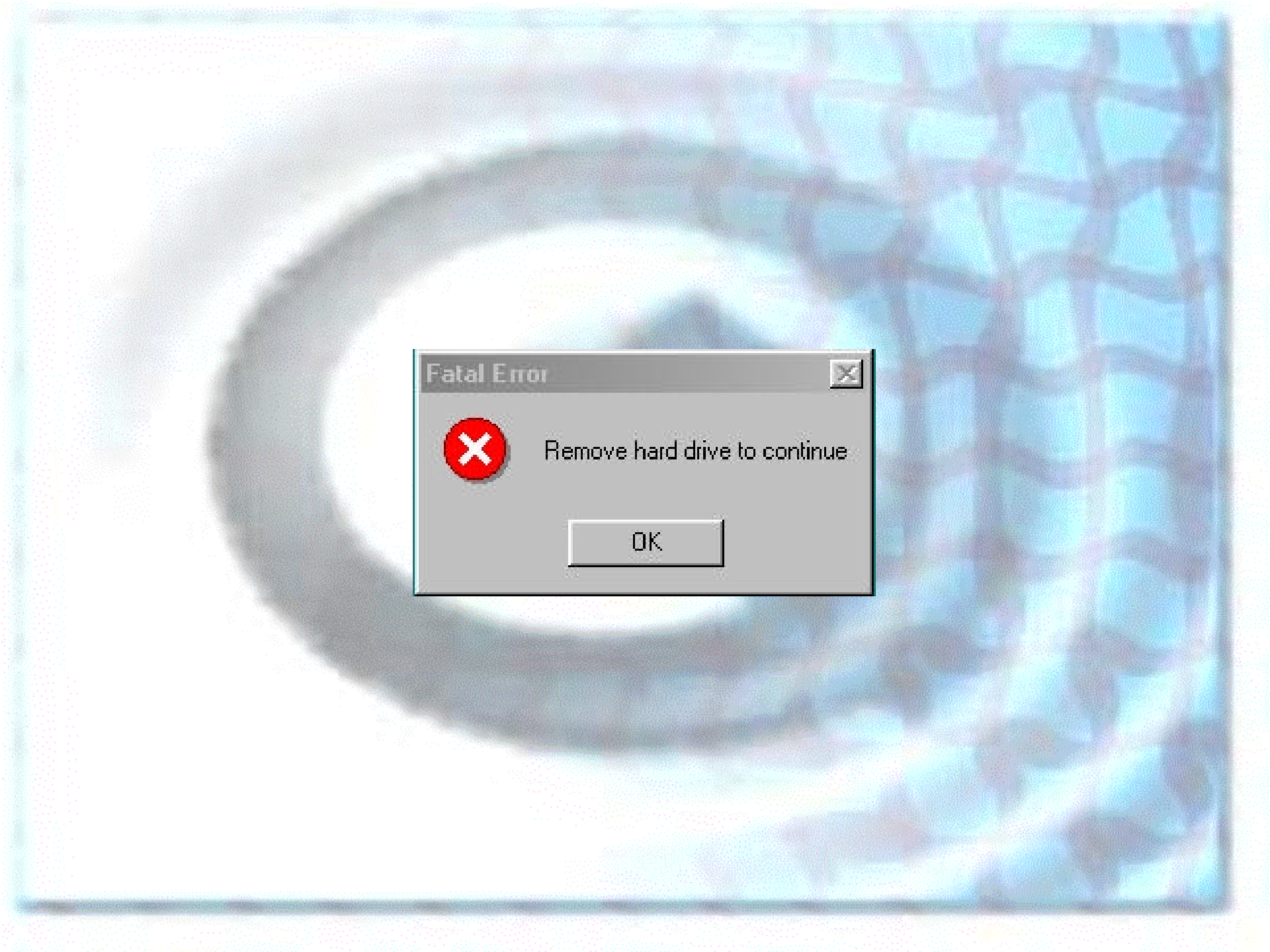
Qt Application Development

Federico Lucifredi


Fatal Error
Remove hard drive to continue
OK

KDE: Origins and controversy

- 1996 – KDE project

  Stallman: “completely free OS”

  RedHat: “our customer base wishes”

- 1997 – Gnome, Harmony Projects
- 2000 – KDE 2.0

  April – GNOME 1.0

  September – Stallman: “beg for forgiveness”

- 2001 – KDE 2.1

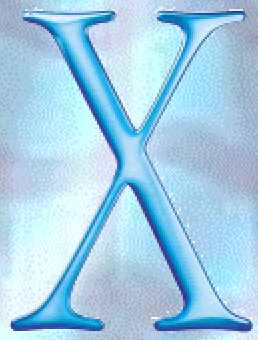
  Ximian/Helixcode incident

Trolltech's Qt

- http://www.trolltech.com
- The toolkit is free (beer, speech) for those who use it to develop Linux software (GPL, QPL)
- License required for Windows. Not free (beer) nor cheap. *But* NCE is available.
- Available for embedded systems

Trolltech's Qt

- NCE "not commercial edition" for Windows available at TrollTech's Website



- first release for MacOs X just out
- NCE not yet available but it will be

- Current Release 3.0 released three weeks ago

Qt features

- Gui Emulation architecture
- Astonishing cross-platform capabilities (Screenshots)
- Sane object model

GUI Toolkit comparison

- GUI Toolkits – “Widget set” compared
  - Motif/Xt
  - Java Awt (old/new event model)
  - Java Swing (new event model)
  - Gtk+
  - wxWindows
  - Qt

RAD

- IDE
  - Qt GUI designer
  - K Developer
  - Borland (Inprise on odd-numbered days) Kylix
- Qt Linguist
  - Internationalization tool

Favorite quotes

"It is *ridiculously* easy to use"

-Daniel Solin

[emphasis added]

Recent improvements

- 2.x series
  - Threading (finally thread safe; cross platform)
  - Network support
- 2.3
  - Xfree86 4.0.2 support for TrueType fonts
  - QPSPrinter for improved PS printing support
- 3.0

Coding in Qt

- Hello World

```
#include <qapplication.h>
#include <qlabel.h>
int main(int argc, char* argv[])
{
   QApplication base(argc,argv);
   QLabel* scritta = new QLabel (“hello world”);
   scritta->resize(320, 120);

   base.setMainWidget (scritta);
   scritta->show();

   return base.exec();
```

Explanation - I

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

- Every Qt Application has to include the qapplication header
- Widgets employed must also be *#included* (Qlabel this time)

Explanation – II

```
int main(int argc, char* argv[])
{
   QApplication base(argc,argv);
   QLabel* scritta = new QLabel (“hello world”);
   scritta->resize(320, 120);
```

- main returns int and takes the command line, just like an Xt application would (geometry, resources, etc)
- QApplication parses the command line for X and KDE parameters.
- Widget QLabel allocated with new (not good to have too many stack variables – always go with dynamic allocation)
- Resizing

Explanation III

```
base.setMainWidget (scritta);
   scritta->show();

   return base.exec();
}
```

- We must tell Qt which is the parent of the widget hierarchy (inheritance vs widget hierarchy)
- Tell the label widget to become visible
- Entering application loop
- At exit, returning exitcode

Widget Hierarchy

- Lets add a “Quit” button

```
#include <qpushbutton.h>

QWidget* genitore = new QWidget();
genitore->setGeometry(400,300, 120, 90);

QLabel* scritta = new Qlabel(“Hello world”, genitore);
Scritta->setGeometry (10,50,100,30);
```

- Declaring a new widget and assigning a child widget

Event Model

- Signals and Slots model

```
…
QPushButton* quit=new QPushButton(“Quit”, genitore);
Scritta->setGeometry(10,10,80,30);

Qobject::connect(quit, SIGNAL(clicked()), &base,
   SLOT(quit()));

base.setMainWidget(genitore);
Genitore->show();
…
```

How is this accomplished ?

- Moc (meta object compiler)
- Additional Headers (.moc)
- Defined in C++ - like syntax with proprietary extensions over the language
- emit() is used to fire signals
- QObject::connect is used to connect signals and slots

Defining signals and slots

```
Class classname : public QObject
{
Q_OBJECT
…
signals:
//I segnali vanno dichiarati qui
void qualcosa_accadde_ora();
Public slots:
//gli slot pubblici vanno dichiarati qui
void fai_qualcosa();
Private slots:
//gli slot privati invece vanno qui
void fai_qualcosa_in_privato();
};
```

Reboot!

Time for a real OS…

…and we’re back!

Lets try it here….

Designing new widgets

- Ever tried on Windows or Motif ?
  - Aaaargh! (*Aaaargh!)*
- Java-like:
  - Choose Base class
  - Access modifiers
  - Accessor methods
  - Event handlers
    - paintEvent()
    - mousePressEvent() perhaps ?
  - Which signals/slots ?

More features

- Themes
- Internationalization
- Unicode
- Cross-platform utility classes (*use them*)

- And DB support, of course

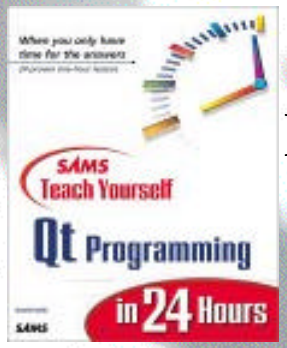
last remarks

- OOP and UNIX
  - Issues do not affect Qt noticeably, but DO affect large KDE programs
  - Optimization schemes were developed that can be used for Qt on Linux as well
    - *but they should not be necessary if you do your job well*

Useful references

Official Docs – the BEST

Matthias Kalle Dalheimer's *Programming with Qt*



Daniel Solin's *Qt Programming in 24 hours*

Patrick Ward's *Qt Programming for Linux and Windows 2000*

Any Questions ?

<flucifredi@acm.org>

<http://www.lucifredi.com/talks>

Thanks for Coming!


© 2001 by Federico Lucifredi -