Lecture Qt009
Events

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Event Loop
- Handling Events
- Event Handling Options
- Hands-On Example: Stopwatch
- Timers
- Event Filters
- Key Points

Event Loop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Performs two tasks
  - Manages window events
  - Generates its own events
    - Derived from QEvent (base class of all events)
    - Each event has a type
    - Events processed by event loop

Event Loop

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Events passed to objects that inherit from **QObject**
- The **event()** method of target object processes or ignores the event
- Events may be queued for later delivery
- If processing of queued events is delayed, application may appear to have crashed

Event Handling Options

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

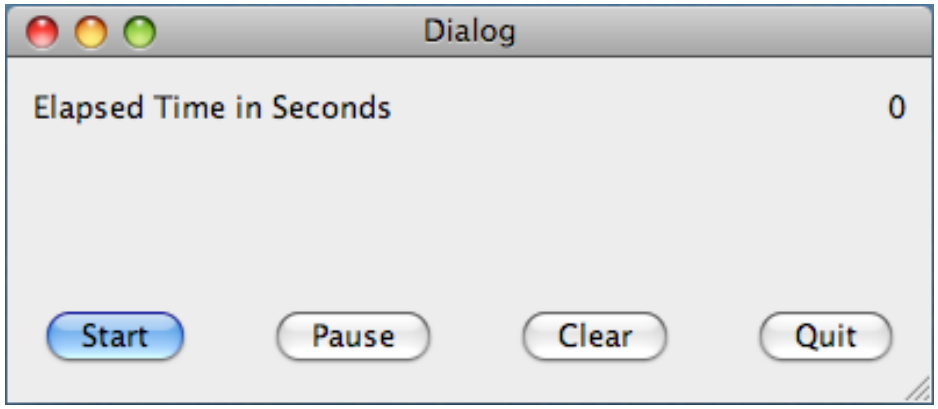
1. Reimplement specific event handler
2. Reimplement generic event handler
   QObject::event()
3. Use event filters on **QObject**
4. Use event filter on **QApplication** object
5. Subclass **QApplication** and reimplement **notify()**

C++ GUI Programming with Qt 4, 2nd Ed. by Jasmin Blanchette and Mark Summerfield

Example: Stop Watch

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Create a Stopwatch dialog with four controls
  - Start
  - Pause
  - Clear
  - Quit



- Display updates every second when not paused or cleared

Example: Method #1
Reimplement Specific Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized main.cpp
#include <QApplication>
#include "dialog.h"

int main(int argc, char *argv[])      // Standard main function
{
    QApplication a(argc, argv);
    Dialog w;
    w.show();
    return a.exec();
}
```

Example: Method #1
Reimplement Specific Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.h
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>

namespace Ui
{
  class Dialog;
}
```

```
class Dialog : public QDialog
{
  Q_OBJECT

public:
  Dialog(QWidget *parent = 0);
  ~Dialog();

protected:
  void timerEvent(QTimerEvent* someEvent);

private:
  Ui::Dialog *ui;
  int updateTimer;
  bool paused;

private slots:
  void initiateTimer();
  void pauseTimer();
  void clearTimer();
};

#endif // DIALOG_H
```

Timer-Specifc Event Handler

Example: Method #1
Reimplement Specific Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp
#include "dialog.h"
#include "ui_dialog.h"
#include <QTimer>
#include <QtDebug>

Dialog::Dialog(QWidget *parent) : QDialog(parent), ui(new Ui::Dialog)
{
    ui->setupUi(this);

    ui->timeLabel->setText("0");
    updateTimer = QObject::startTimer(1000); // Timer event every 1000msec
    paused = true;

    connect(ui->startButton, SIGNAL(clicked()), this, SLOT(initiateTimer()));
    connect(ui->pauseButton, SIGNAL(clicked()), this, SLOT(pauseTimer()));
    connect(ui->clearButton, SIGNAL(clicked()), this, SLOT(clearTimer()));

    QTimerEvent* event = new QTimerEvent(updateTimer);

    QCoreApplication::postEvent(this, event);  // Manually add to event queue
} // End Dialog::timerEvent()
```

Example: Method #1
Reimplement Specific Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp – continued
void Dialog::timerEvent(QTimerEvent* someTimerEvent)
{
    if (!someTimerEvent)   return;
    if (someTimerEvent->timerId() == updateTimer) // updateTimer event?
    {
        if (!paused)
        {
            bool ok;
            int n = ui->timeLabel->text().toInt(&ok) + 1;
            if (ok)
            {
                ui->timeLabel->setText(QString::number(n));
                qDebug() << "Updating display";
            }
            else
            {
                ui->timeLabel->setText("");
                qDebug() << "Dialog::eventTimer(): Unknown error";
            }
        }
        else
            qDebug() << "Timer paused";
    }
    else
        QObject::timerEvent(someTimerEvent);
} // End Dialog::timerEvent()
```

Example: Method #1
Reimplement Specific Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp – continued

void Dialog::initiateTimer()
{
    qDebug() << "Initiating Timer”;
    paused = false;
} // End Dialog::initiateTimer()

void Dialog::pauseTimer()
{
    qDebug() << "Pausing Timer";
    paused = true;
} // End Dialog::pauseTimer()

void Dialog::clearTimer()
{
    qDebug() << "Clearing Timer";
    ui->timeLabel->setText("0");
    paused = true;
} // End Dialog::clearTimer()

Dialog::~Dialog()
{
    delete ui;
}
```

Event Handlers: Guidelines

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- If a specialized event handler already exists, override it instead of the general event handler
- Input parameter will already be of the correct data type
- No need to forward unhandled events

Timers

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- In most cases, the signal-slot mechanism is the simplest and most direct way to make use of **QTimers**
- At the end of each time interval, a **timeout()** event is emitted
- This event can be used to trigger an appropriate slot function

Hands-On Example: Timers

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.h
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>
#include <QTimer>

namespace Ui
{
  class Dialog;
}

class Dialog : public QDialog
{
  Q_OBJECT

public:
  Dialog(QWidget *parent = 0);
  ~Dialog();

private:
  Ui::Dialog *ui;
  int updateTimer;
  bool paused;
  QTimer* timer;

private slots:
  void initiateTimer();
  void pauseTimer();
  void clearTimer();
  void updateDisplay();
};

#endif // DIALOG_H
```

Hands-On Example: Timers

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp
#include "dialog.h"
#include "ui_dialog.h"
#include <QTimer>
#include <QtDebug>

Dialog::Dialog(QWidget *parent) : QDialog(parent), ui(new Ui::Dialog)
{
    ui->setupUi(this);
    ui->timeLabel->setText("0");
    timer = new QTimer(this);
       // Allocate timer and set interval but don't start timer

    timer->setInterval(1000);
    paused = true;
    connect(ui->startButton, SIGNAL(clicked()), this, SLOT(initiateTimer()));
    connect(ui->pauseButton, SIGNAL(clicked()), this, SLOT(pauseTimer()));
    connect(ui->clearButton, SIGNAL(clicked()), this, SLOT(clearTimer()));

    connect(timer, SIGNAL(timeout()), this, SLOT(updateDisplay()));

} // End Dialog::timerEvent()
```

Hands-On Example: Timers

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp -- continued

void Dialog::updateDisplay()
{
    if (!paused)
    {
        bool ok;
        int n = ui->timeLabel->text().toInt(&ok) + 1;
        if (ok)
        {
            ui->timeLabel->setText(QString::number(n));
            qDebug() << "Updating display";
        }
        else
        {
            ui->timeLabel->setText("");
            qDebug() << "Dialog::eventTimer(): Unknown error";
        }
    }
    else
    {
        qDebug() << "Timer paused";
    }
} // End Dialog::updateDisplay()
```

Hands-On Example: Timers

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.cpp -- continued

void Dialog::initiateTimer()
{
    qDebug() << "Initiating Timer";
    paused = false;
    timer->start();
} // End Dialog::initiateTimer()

void Dialog::pauseTimer()
{
    qDebug() << "Pausing Timer";
    paused = true;
    timer->stop();
} // End Dialog::pauseTimer()

void Dialog::clearTimer()
{
    qDebug() << "Clearing Timer";
    ui->timeLabel->setText("0");
    paused = true;
    timer->stop();
} // End Dialog::clearTimer()

Dialog::~Dialog()
{
    delete ui;
}
```

Example: Method #2
Reimplement Generic Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Customized dialog.h
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>

namespace Ui
{
  class Dialog;
}
```

```
class Dialog : public QDialog
{
  Q_OBJECT

public:
  Dialog(QWidget *parent = 0);
  ~Dialog();

protected:
  bool event(QEvent* someEvent);

private:
  Ui::Dialog *ui;
  int updateTimer;
  bool paused;

private slots:
  void initiateTimer();
  void pauseTimer();
  void clearTimer();
};

#endif // DIALOG_H
```

Example: Method #2
Reimplement Generic Event Handler

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
bool Dialog::event(QEvent* someEvent)
{
    if (!someEvent)  return QObject::event(someEvent);
    if (someEvent->type() == QEvent::Timer)
    {
        QTimerEvent* timerEvent = static_cast<QTimerEvent*>(someEvent);
        if (timerEvent->timerId() == updateTimer)   // updateTimer event?
        {
            if (!paused)
            {
                bool ok;
                int n = ui->timeLabel->text().toInt(&ok) + 1;
                if (ok) {
                    ui->timeLabel->setText(QString::number(n));
                    qDebug() << "Updating display";
                }
                else  {
                    ui->timeLabel->setText("");
                    qDebug() << "Dialog::eventTimer(): Unknown error";
                }
            }
            else
                qDebug() << "Timer paused”;
        }
    }
    return QObject::event(someEvent);
} // End Dialog::timerEvent()
```

Event Filters

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- QObject-derived classes have both event handlers and event filters
- Object A can receive Object B events
- Each B event received by A may be
  - Forwarded on to B
  - Removed from B's event stream

Event Filters

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Event filters must be installed
  - Within A's constructor, call installEventFilter() to monitor B events

    b->installEventFilter(this);

    where b is a pointer to B
  - B events now pass to A first where A has the option of filtering them

Event Filters

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

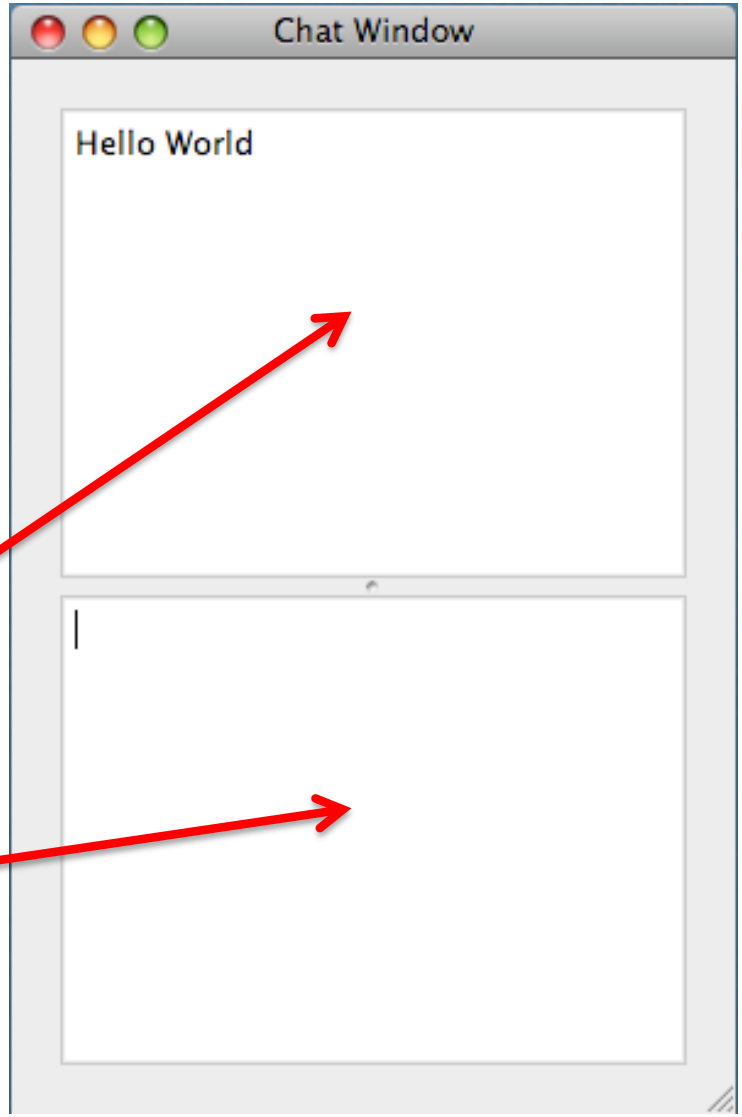
- Event filters must be installed (continued)
  - An **eventFilter()** must then be reimplemented in A (virtual method of QObject)

```
bool QObject::eventFilter(QObject *watched, QEvent *e);
```

  - A **false** return value causes Qt to forward event to B
  - A **true** return value results in the event does not get forwarded to B

Example: Chat Window

- Classic event filter example
  - Enter/Return key press forwards typed text rather than starts new lines of text



Example: Chat Window

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// chatwindow/chatwindow.h
// Molkentin, Book of Qt4

#ifndef CHATWINDOW_H
#define CHATWINDOW_H

#include <QWidget>
#include <QTextBrowser>
#include <QTextEdit>

class QTextBrowser;
class QTextEdit;
class QEvent;

class ChatWindow : public QWidget
{
  Q_OBJECT

  public:
    ChatWindow(QWidget *parent = 0);
    void submitChatText();
    bool eventFilter(QObject *watched, QEvent *e);

  private:
    QTextBrowser *conversationView;
    QTextEdit *chatEdit;
};
#endif // CHATWINDOW_H
```

Example: Chat Window

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// chatwindow/chatwindow.cpp
// Molkentin, Book of Qt4

#include <QVBoxLayout>
#include <QSplitter>
#include "chatwindow.h"

ChatWindow::ChatWindow(QWidget *parent) : QWidget(parent)
{
  QVBoxLayout *lay = new QVBoxLayout(this);
  QSplitter *splitter = new QSplitter(Qt::Vertical, this);
  lay->addWidget(splitter);

  conversationView = new QTextBrowser;
  chatEdit = new QTextEdit;

  splitter->addWidget(conversationView);
  splitter->addWidget(chatEdit);

  ChatWindow->chatEdit->installEventFilter(this);

  setWindowTitle("Chat Window");
  setTabOrder(chatEdit, conversationView);
}
```

Example: Chat Window

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// chatwindow/chatwindow.cpp
// Molkentin, Book of Qt4

bool ChatWindow::eventFilter(QObject *watched, QEvent* e)
{
  if (watched == chatEdit && e->type() == QEvent::KeyPress)
  {
    QKeyEvent *ke = static_cast<QKeyEvent*>(e);

    if (ke->key() == Qt::Key_Enter || ke->key() == Qt::Key_Return)
    {
      submitChatText();
      return true;
    }
  }

  return QWidget::eventFilter(watched, e);
}
```

Example: Chat Window

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// chatwindow/chatwindow.cpp
// Molkentin, Book of Qt4

void ChatWindow::submitChatText()
{
   // append text as new paragraph
   conversationView->append(chatEdit->toPlainText());

   // clear chat window
   chatEdit->setPlainText("");
}
```

Key Points

- Event handling may be fine-tuned by
  - Modifying an event-specific handler
  - Modifying the generic event hander
- Approach
  - Inherit from the appropriate base class
  - Modify the event-specific handler or generic event handler as needed
  - In some cases after executing the custom code within the modified event handler, you may still want to call the unmodified event handler code from within your modified event handler