Lecture 5. Event Processing in Qt

Cross-Platform Application Development

October 12, 2017

Event Classes

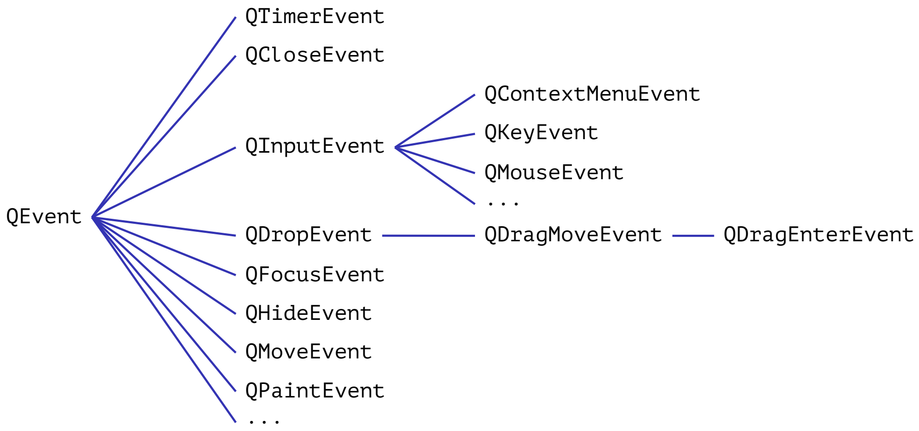

Figure 1: a part of event class hierarchy

Events

QEvent Non-Static Methods

- `void accept()`
- `void ignore()`
- `bool isAccepted() const`
- `bool spontaneous() const`
- `QEvent::Type type() const`

QEvent Static Methods

- `int registerEventType(int hint = -1)`

None	FocusIn	MouseButtonPress
ApplicationStateChange	FocusOut	MouseButtonRelease
Clipboard	KeyPress	MouseMove
Close	KeyRelease	Paint

Table 1: examples of Qt event types (**enum** `QEvent::Type`)

Event Handlers

QObject Event Handlers

- `bool event(QEvent *pEvent)`
- `void customEvent(QEvent *pEvent)`
- `void timerEvent(QTimerEvent *pEvent)`

QWidget Event Handlers

- `void closeEvent(QCloseEvent *pEvent)`
- `void dragEnterEvent(QDragEnterEvent *pEvent)`
- `void keyPressEvent(QKeyEvent *pEvent)`
- `void mouseMoveEvent(QMouseEvent *pEvent)`
- `void paintEvent(QPaintEvent *pEvent)`
- `void resizeEvent(QResizeEvent *pEvent)`

Painting Device Classes

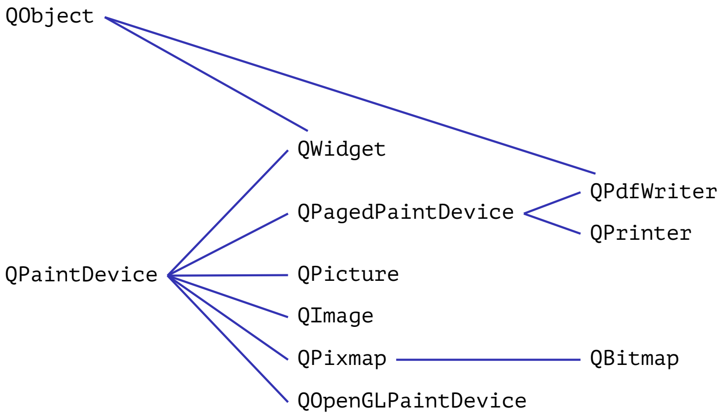

Figure 2: a part of 2D graphics painting device class hierarchy

Example

Example (my-window.cpp)

```
void MyWindow::paintEvent(QPaintEvent *pEvent)
{
  QPainter painter(this);
  // ...
  // painter.draw/* ... */(/* ... */);
  // ...
}
```

Properties of Graphics Drawing

Class	Properties
`QPen`	• color; • thickness; • style of ending (cap); • style of line joining.
`QBrush`	• color; • style; • texture (opt.); • gradient (opt.)

Class	Properties
`QFont`	• family; • italic style; • boldness; • underlining; • ...

Table 2: main properties of `QPainter` class

Methods for Graphics Output

drawPoint()	drawLine()	drawPolyline()
drawPoints()	drawLines()	drawPolygon()
drawRect()	drawRoundRect()	drawEllipse()
drawArc()	drawChord()	drawPie()
drawText()	drawPixmap()	drawPath()

Table 3: main methods of `QPainter` class

Example

Example (my-window.cpp)

```
void MyWindow::paintEvent(QPaintEvent *pEvent)
{
  QPainter painter(this);
  //
  painter.setRenderHint(QPainter::Antialiasing, true);
  painter.setPen(QPen(Qt::black, 12, Qt::DashDotLine, Qt::RoundCap));
  painter.setBrush(QBrush(Qt::green, Qt::SolidPattern));
  painter.drawEllipse(80, 80, 400, 240);
}
```

Example

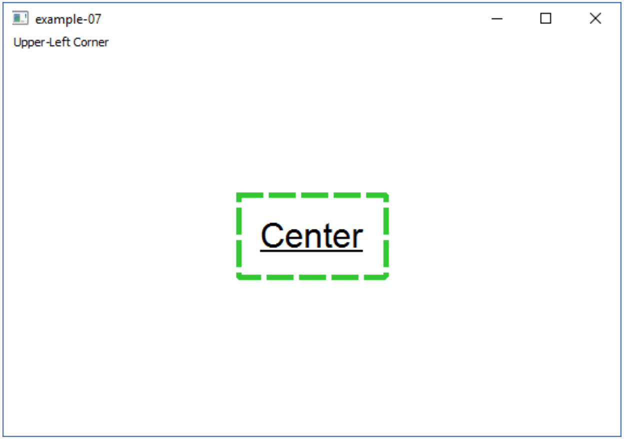

Figure 3: a window with event processing

Example

Example (main-window.h)

```
#ifndef MAIN_WINDOW_H__
#define MAIN_WINDOW_H__

#include <QWidget>

class MainWindow : public QWidget
{
public:
  //
  MainWindow(QWidget *pParent = 0);
  //
```

Example (cont.)

Example (main-window.h, end)

```
protected:
  //
  void resizeEvent(QResizeEvent *pEvent);
  void paintEvent(QPaintEvent *pEvent);
  //
private:
  //
  QPixmap m_Pixmap;
};    // class MainWindow

#endif    // MAIN_WINDOW_H__
```

Example (cont.)

Example (main-window.cpp)

```
#include "main-window.h"

#include <QtWidgets>

MainWindow::MainWindow(QWidget *pParent)
  : QWidget(pParent)
{
  setAttribute(Qt::WA_NoSystemBackground, true);
}
```

Example (cont.)

Example (main-window.cpp, cont.)

```
void MainWindow::resizeEvent(QResizeEvent *pEvent)
{
  m_Pixmap = QPixmap(pEvent->size());
  m_Pixmap.fill(Qt::white);
  //
  QPainter painter(&m_Pixmap);
  //
  const QString cstrL = QString::fromLocal8Bit(
    "Upper-Left Corner");
  painter.drawText(10, 10, cstrL);
  //
```

Example (cont.)

Example (main-window.cpp, cont.)

```
QFont font("Arial", 24);
font.setUnderline(true);
//
const QString cstrC = QString::fromLocal8Bit("Center");
const int cnFlags = Qt::AlignCenter | Qt::TextSingleLine;
QRect rectText;
painter.setFont(font);
painter.drawText(m_Pixmap.rect(), cnFlags, cstrC, &rectText);
rectText.adjust(-20, -20, 20, 20);
QPen pen(QColor(50, 200, 50));
```

Example (cont.)

Example (main-window.cpp, end)

```
  pen.setStyle(Qt::DashLine);
  pen.setWidth(5);
  painter.setPen(pen);
  painter.drawRect(rectText);
}    // MainWindow::resizeEvent()

void MainWindow::paintEvent(QPaintEvent *pEvent)
{
  QPainter painter(this);
  painter.drawPixmap(0, 0, m_Pixmap);
}
```

Example (end)

Example (`example-07.cpp`)

```
#include "main-window.h"

#include <QApplication>

int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  MainWindow *pWindow = new MainWindow;
  pWindow->show();
  //
  return app.exec();
}
```

Example

Example (key-window.h)

```
#ifndef KEY_WINDOW_H__
#define KEY_WINDOW_H__

#include <QWidget>

class KeyWindow : public QWidget
{
public:
  //
  KeyWindow(QWidget *pParent = 0);
  //
```

Example (cont.)

Example (key-window.h, end)

```
protected:
  //
  void keyPressEvent(QKeyEvent *pEvent);
  void keyReleaseEvent(QKeyEvent *pEvent);
  void paintEvent(QPaintEvent *pEvent);
  //
private:
  //
  bool m_bKeyPressed;
};    // class KeyWindow

#endif    // KEY_WINDOW_H__
```

Example (cont.)

Example (key-window.cpp)

```
#include "key-window.h"

#include <QtWidgets>

KeyWindow::KeyWindow(QWidget *pParent)
  : QWidget(pParent),
    //
    m_bKeyPressed(false)
{
  // setAutoFillBackground(false);    // by default
}
```

Example (cont.)

Example (key-window.cpp, cont.)

```
void KeyWindow::keyPressEvent(QKeyEvent *pEvent)
{
  m_bKeyPressed = true;
  repaint();
}

void KeyWindow::keyReleaseEvent(QKeyEvent *pEvent)
{
  m_bKeyPressed = false;
  repaint();
}
```

Example

Example (key-window.cpp, end)

```
void KeyWindow::paintEvent(QPaintEvent *pEvent)
{
  QPainter painter(this);
  painter.fillRect(
    rect(),
    m_bKeyPressed ? Qt::darkGray : Qt::lightGray);
}

// End of File
```

Example

Example (timer-window.h)

```
#ifndef TIMER_WINDOW_H__
#define TIMER_WINDOW_H__

#include <QWidget>

class TimerWindow : public QWidget
{
public:
  //
  TimerWindow(QWidget *pParent = 0);
  //
protected:
  //
```

Example (cont.)

Example (timer-window.h, end)

```
  void showEvent(QShowEvent *pEvent);
  void hideEvent(QHideEvent *pEvent);
  void timerEvent(QTimerEvent *pEvent);
  void paintEvent(QPaintEvent *pEvent);
  //
private:
  //
  int m_nTimerId;
  int m_nRadius;
};    // class TimerWindow

#endif    // TIMER_WINDOW_H__
```

Example (cont.)

Example (timer-window.cpp)

```
#include "timer-window.h"

#include <QtWidgets>

TimerWindow::TimerWindow(QWidget *pParent)
  : QWidget(pParent),
    //
    m_nTimerId(0),
    m_nRadius(0)
{
  //
}
```

Example (cont.)

Example (timer-window.cpp, cont.)

```
void TimerWindow::showEvent(QShowEvent *pEvent)
{
  m_nTimerId = startTimer(100);
}

void TimerWindow::hideEvent(QHideEvent *pEvent)
{
  killTimer(m_nTimerId);
}
```

Example (cont.)

Example (timer-window.cpp, cont.)

```
void TimerWindow::timerEvent(QTimerEvent *pEvent)
{
  if (pEvent->timerId() == m_nTimerId)
  {
    m_nRadius = (m_nRadius + 1) % 10;
    repaint();
  }
  else
    QWidget::timerEvent(pEvent);
}
```

Example

Example (timer-window.cpp, end)

```
void TimerWindow::paintEvent(QPaintEvent *pEvent)
{
  const int cnRadius = 10 + 10 * m_nRadius;
  const QColor cColor(
    30 + 22 * m_nRadius, 255 - 20 * m_nRadius, 128 + 10 * m_nRadius);
  //
  QPainter painter(this);
  painter.setPen(cColor);
  painter.fillRect(rect(), Qt::white);
  painter.drawEllipse(rect().center(), cnRadius, cnRadius);
}

// End of File
```

Example

Example (Lengthy Event Handling)

```
void MyWindow::someEvent(QSomeEvent *pEvent)
{
  for (int i = 0; i < m_nMax; ++ i)
  {
    doLengthyTask(i);
    QCoreApplication::instance()->processEvents();
  }
}
```