Archived:Painting only the dirty area of the widget in Qt

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (using C++ for the Qt app UI) is deprecated.

Overview

This example shows how to paint only the area of the widget whose content has been marked as invalid and needs to be updated. This will speed up the updating of the screen.

When the user draws a line using the stylus on the screen, the line is first painted into a `QPixmap` picture and then the picture is updated to the screen. The application only updates the dirty area of the picture.

Dirty area painting can be handled by setting the `Qt::WA_OpaquePaintEvent` attribute of the widget as `true` and defining the dirty rect to `QPainter` with `QPainter::setClipRegion()`.

Preconditions

- Qt is installed on your platform.
  - Download Qt release from here: [1]
  - Qt on Maemo can be found here: Qt4 Maemo port

Header

```
#include <QtGui>

class Widget: public QWidget
{
Q_OBJECT

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

    void mouseMoveEvent(QMouseEvent *);
    void mousePressEvent(QMouseEvent *);
    void mouseReleaseEvent(QMouseEvent *);

    void paintEvent(QPaintEvent *);
    void resizeEvent(QResizeEvent *);

private:
    void paintToPixmap();
    void paintLine(QPainter&);
    QRect dirtyRect();

private:
    QPoint p1;
    QPoint p2;

    QPen pen;
    QColor color;

    QPixmap pixmap;
};
```

Source

```
Widget::Widget(QWidget *parent) :
    QWidget(parent)
{
    p1 = QPoint(-1, -1);
    p2 = QPoint(-1, -1);

    color = QColor(Qt::black);
    pen.setColor(color);
    pen.setStyle(Qt::SolidLine);
    pen.setWidth(2);
}

Widget::~Widget()
{
```


(C) Copyright Nokia 2013. All rights reserved.

```
}

void Widget::resizeEvent(QResizeEvent *event)
{
    QWidget::resizeEvent(event);
    // Create pixmap that size is same as screen has
    pixmap = QPixmap(event->size());
    pixmap.fill(Qt::white);
}

void Widget::mousePressEvent(QMouseEvent *event)
{
    p1 = event->pos();
    p2 = QPoint(-1, -1);

    // Opaquea paint, we paint only changed area of the screen
    setAttribute(Qt::WA_OpaquePaintEvent, true);
}

void Widget::mouseMoveEvent(QMouseEvent *event)
{
    if (p2.x() == -1) {
        p2 = event->pos();
    }
    else {
        p1 = p2;
        p2 = event->pos();
    }
    // Paint to pixmap and then pixmap on screen
    paintToPixmap();
}

void Widget::mouseReleaseEvent(QMouseEvent *)
{
    // Stop using opaquea paint
    setAttribute(Qt::WA_OpaquePaintEvent, false);

    // Update screen
    update();
}

void Widget::paintToPixmap()
{
    // Paint line to pixmap
    QPainter painter;
    painter.begin(&pixmap);
    painter.setRenderHint(QPainter::Antialiasing);
    painter.setPen(pen);
    paintLine(painter);
    painter.end();
    // Draw pixmap on the screen
    update();
}

void Widget::paintLine(QPainter& painter)
{
    if (p1.x() != -1 && p2.x() != -1) {
        painter.drawLine(p1, p2);
    }
}

QRect Widget::dirtyRect()
{
    int left, top, width, height;
    if (p1.x() == -1 || p2.x() == -1) {
        return rect();
    }

    if (p1.x() < p2.x()) {
        left = p1.x();
        width = p2.x() - p1.x();
    }
    else {
        left = p2.x();
        width = p1.x() - p2.x();
    }

    if (p1.y() < p2.y()) {
        top = p1.y();
        height = p2.y() - p1.y();
    }
    else {
        top = p2.y();
        height = p1.y() - p2.y();
    }
    return QRect(left - 15, top - 15, width + 30, height + 30);
}

void Widget::paintEvent(QPaintEvent *)
{
    QPainter painter(this);

    // Paint only changed area of the screen
    if (testAttribute(Qt::WA_OpaquePaintEvent)) {
        painter.setClipRegion(dirtyRect());
    }

    // Update pixmap into screen. If dirty
    // area is set painter updates only that
    painter.drawPixmap(QPoint(0, 0), pixmap);
}
```

Postconditions

The screen updates quickly.

See also

(C) Copyright Nokia 2013. All rights reserved.

- QPainter

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.