HttpRemote

Overview

This example show how to implement a custom QGraphicsWidget that uses not so common graphics operations. This widget is then used to trigger custom http-get-requests. Such GET-requests are commonly used by cheap embedded hardware to trigger a certain action. The use case for this example is to remote control a media player by sending those requests when pressing on of the buttons.

Graphics

The idea behind the graphics is to show a remote that reminds the user of a situation related to media playback. Digital pictures are usually composed of three color channels: Red, Blue and Green. Other colors are mixed from these channels. If these channels are shifted in position, the user will see the three separate channels. This UI is composed of white controls on a dark background. But whenever the user presses a control, the color channels separate, orbit around the center of the control and the vibra goes on to simulate the effect of a motor or other device.

To be able to draw the three channels above each other and mix the colors, a special composition mode has to be set in QPainter: QPainter::CompositionMode_Plus.

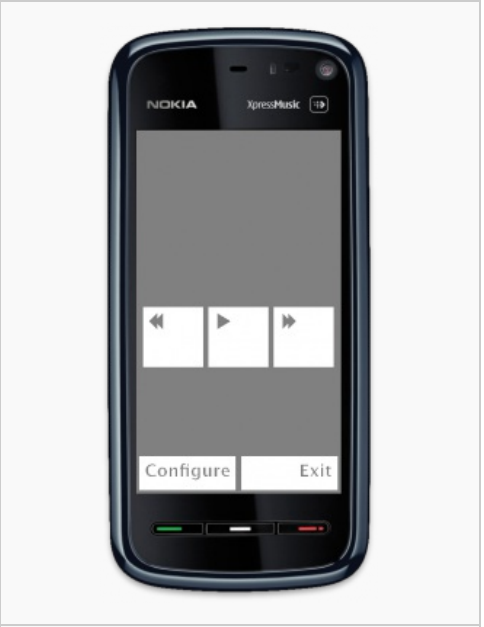
```
void ChannelButton::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    // Store current composition mode, brush and pen to be able to restore them. QPainter's native
    // save/restore seem not to work for composition mode.
    QPainter::CompositionMode old_mode(painter->compositionMode());
    QPen old_pen(painter->pen());
    QBrush old_brush(painter->brush());

    // Set special composition mode
    painter->setCompositionMode(QPainter::CompositionMode_Plus);

    double offset(m_phase * 5);

    painter->setPen(QPen(Qt::NoPen));

    // Draw red channel
    painter->save();
    painter->setBrush(Qt::red);
    painter->translate(QPointF(cos(M_PI * (m_angle + 15.0) / 180.0),
                               sin(M_PI * (m_angle + 15.0) / 180.0)) * offset);
    painter->drawPath(m_path);
    painter->restore();

    // Draw green channel
    painter->save();
    painter->setBrush(Qt::green);
    painter->translate(QPointF(cos(M_PI * (m_angle + 135.0) / 180.0),
                               sin(M_PI * (m_angle + 135.0) / 180.0)) * offset);
    painter->drawPath(m_path);
    painter->restore();

    // Draw blue channel
    painter->save();
    painter->setBrush(Qt::blue);
    painter->translate(QPointF(cos(M_PI * (m_angle + 285.0) / 180.0),
                               sin(M_PI * (m_angle + 285.0) / 180.0)) * offset);
    painter->drawPath(m_path);
    painter->restore();

    // Restore previos painter configuration
    painter->setPen(old_pen);
    painter->setBrush(old_brush);
    painter->setCompositionMode(old_mode);
}
```

Network handling

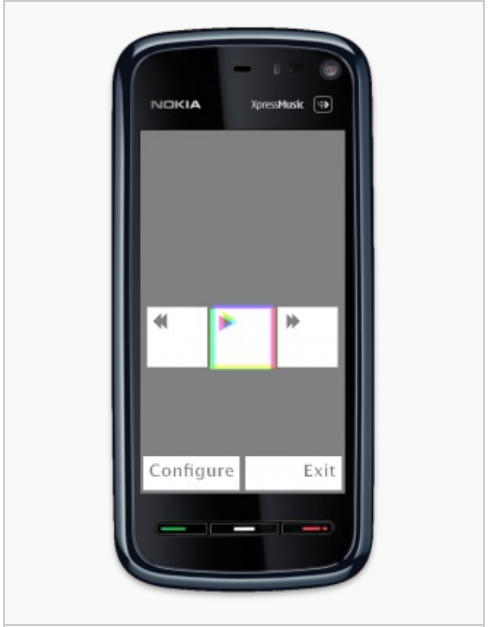
Originally developed to showcase the Qt Bearer Management Library, the example is meant to connect to a local network. To my surprise, I did not had to do anything but my phone automatically connected to the local network. Now the only thing that was needed was to issue QNetworkManager for the http-requests.

Screenshots

(C) Copyright Nokia 2013. All rights reserved.



Idle



Button pressed

Keep in mind that the screenshot shows only one frame of the animation of the actual running application.

Testing the application

The example is either be meant to be compiled directly for the device or as a desktop application.

Download Code Example

File:Httpremote.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.