QML Performance Meter

This article presents a custom QML element for measuring and displaying its display rate as a frames-per-second (FPS) curve. The element provides a very lightweight graphical means for assessing the impact of changes on app performance.

29 Apr 2012

Overview

When developing an application, it can be useful to know how changing the code affects performance. This QML Element measures the time between its own redraws and displays the frame rate (FPS) as a curve against time. This approach is more useful for performance analysis than simply displaying the current FPS rate.

The element itself has a minimal performance impact because its content is only updated in specified intervals and "as fast as possible" pixmap drawing is used otherwise.

Header file

```
#ifndef __PERFORMANCEMETER__
#define __PERFORMANCEMETER__

#include <QDeclarativeItem>
#include <QImage>
#include <QTime>

// Max samples to keep in history
#define PERFORMANCE_SAMPLES 256
// The interval between measurements
#define MEASURE_INTERVAL_MSECS 500

class PerformanceMeter : public QDeclarativeItem
{
    Q_OBJECT

public:
    PerformanceMeter(QDeclarativeItem *parent = 0);
    virtual ~PerformanceMeter();

    /**
     * Note that performance is measured between paints.
     * This means that if the framework is not constantly
     * calling paint, FPS reading is trivial.
     *
     */
    void paint(QPainter *painter,
               const QStyleOptionGraphicsItem *option,
               QWidget *widget);


protected:
    void line(unsigned int *vline, const int pitch, int y,int y2,
              const unsigned int col );
    void hline(unsigned int *d, int length, int col );
    bool measure();

protected:
    QTime lastMeasurementTime;
    int frameCounter;
    QPixmap displayPixmap;
    unsigned int maxSample;
    unsigned int sampleTable[PERFORMANCE_SAMPLES];
};

QML_DECLARE_TYPE(PerformanceMeter)

#endif
```

Source file

```
#include "performancemeter.h"
#include <math.h>
#include <QDebug>
#include <QPainter>


/*!
  \class PerformanceMeter
  \brief Measure performance between paints
*/


/*!
  Construct and reset
*/
PerformanceMeter::PerformanceMeter(QDeclarativeItem *parent)
    : QDeclarativeItem( parent )
{
    setFlag(QGraphicsItem::ItemHasNoContents, false);
    for (int f=0; f<PERFORMANCE_SAMPLES; f++) sampleTable[f] = 0;
    frameCounter = 0;
    lastMeasurementTime = QTime::currentTime();
    maxSample = 256;
}
```


(C) Copyright Nokia 2013. All rights reserved.

```
PerformanceMeter::~PerformanceMeter()
{
}


bool PerformanceMeter::measure()
{
    frameCounter++;
    QTime ctime = QTime::currentTime();
    int msecsPassed = lastMeasurementTime.msecsTo( ctime );

    if (msecsPassed >= MEASURE_INTERVAL_MSECS) {
        float secsPassed = (float)msecsPassed / 1000.0f;
        float fps = (float)frameCounter / secsPassed;
        qDebug("%d frames in %f secs: fps: %f", frameCounter, secsPassed, fps);

        // Scroll samples to right
        for (int f=PERFORMANCE_SAMPLES-1; f>0; f--) sampleTable[f] = sampleTable[f-1];
        // Add the new sample
        sampleTable[0] = (int)( fps * 256.0 );
        if (sampleTable[0]+1024>maxSample) maxSample = sampleTable[0]+1024;

        lastMeasurementTime = ctime;
        frameCounter = 0;
        return true;
    }
    return false;
}


/**
 * Draw a vertical line into memory
 *
 */
void PerformanceMeter::line( unsigned int *vline, const int pitch, int y, int y2, const unsigned int col )
{
    if (y>y2) {
        int temp = y2;
        y2 = y;
        y = temp;
    }
    do {
        vline[(y>>8)*pitch] |= col;
        y+=256;
    } while (y<y2);
}

/**
 * Draw a horizontal line into memory
 *
 */
void PerformanceMeter::hline( unsigned int *d, int length, int col ) {
    while (length>0) { *d = col; d++; length--; }
}


/*!
 QDeclarativeItem's paint. This method render's the internal infoImage when
 required.
*/
void PerformanceMeter::paint(QPainter *painter,
                             const QStyleOptionGraphicsItem *option,
                             QWidget *widget)
{
    int myWidth = boundingRect().width();
    int myHeight = boundingRect().height();
    if (myWidth<16 || myHeight<16) return;      // area too small to be rendered

    // Update samples, if a new sample is received
    // repaint the curve.
    if (measure())
    {
        if (displayPixmap.isNull() ||
            displayPixmap.width() != myWidth ||
            displayPixmap.height() != myHeight)
            displayPixmap = QPixmap( myWidth, myHeight );

        QImage timage = QImage( myWidth, myHeight, QImage::Format_ARGB32 );
        int f,g;
        unsigned int *vline = ((unsigned int*)timage.bits());

        vline = (unsigned int*)timage.bits() + myWidth-1;
        int drawPitch = timage.bytesPerLine()/4;

        g = 0;
        int sinc = (maxSample) / myHeight;
        // scale
        for (f=0; f<myHeight; f++) {
            hline( (unsigned int*)timage.bits() + (myHeight-1-f)*drawPitch, myWidth, 0xaa000000 | (((((g>>8)/10)&1)*100+64)<<16) );
            g+=sinc;
        }

        int sample;
        int prevSample = ((myHeight-1)<<8)-(sampleTable[0])*(myHeight<<8) / maxSample;
        for (f=0; f<myWidth; f++) {
            if (f<PERFORMANCE_SAMPLES)
                sample = ((myHeight-1)<<8)-sampleTable[f]*(myHeight<<8) / maxSample;
            else
                sample = 0;     // out of history

            line( vline, drawPitch,
                  prevSample,
                  sample,
                  0x00FFFFFF );
            prevSample = sample;
            vline--;
        }
        displayPixmap.convertFromImage( timage );
    }

    if (!displayPixmap.isNull())
        painter->drawPixmap(0,0, displayPixmap );
}
```


(C) Copyright Nokia 2013. All rights reserved.

Deploying the component

The application must define this component before it can be used. Before QDeclarativeView is launched, the registration is done with:

```
qmlRegisterType<PerformanceMeter>("CustomElements", 1, 0, "PerformanceMeter");
```

After it's registration, the component can be defined in a QML file:

```
PerformanceMeter {
 id: performanceMeter
    width: 128; height: 64
}
```

Result

The number of paints is displayed as a white curve on top of dark- and light-red bars. Each of these bars indicate 10fps.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.