QHdrCamera component for High Dynamic Range Imaging

This article explains how to use QHdrCamera, a custom QML component for creating High Dynamic Range Images using the device camera.

24 Jun 2012

Note: This is an entry in the PureView Imaging Competition 2012Q2

Introduction

High **D**ynamic **R**ange **I**maging (HDRI or HDR) is a set of methods used in image processing, computer graphics, and photography, to allow a greater dynamic range between the lightest and darkest areas of an image.

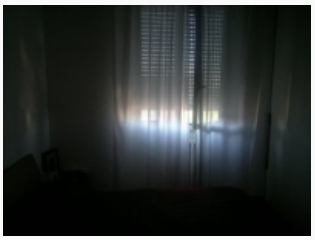
Non-HDR cameras take pictures at one exposure level with a limited contrast range. This results in the loss of detail in bright or dark areas of a picture, depending on whether the camera had a low or high exposure setting. HDR compensates for this loss of detail by taking multiple pictures at different exposure levels and intelligently stitching them together so that we eventually arrive at a picture that is representative in both dark and bright areas.

This article presents custom QML components that can be used to capture and preview an HDR image. It includes demonstration images taken using the components, an explanation of how to use the component in your projects, and an overview of the algorithm used.

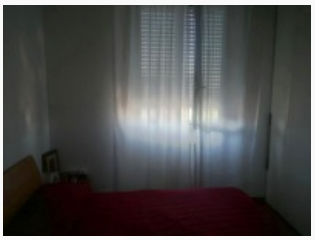
Note: HDR was added as an option to iPhone 4 in iOS version 4.1 on September 8th 2010. At that time you would have had to implemented it in Symbian C++ - it's much easier now using Qt!

Demonstration images

Images taken with a **Nokia N8**



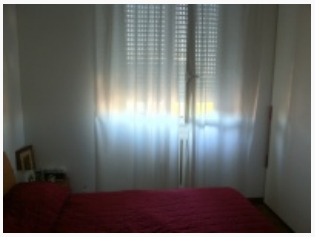
Original shot



Final result using QHdrCamera component

The same images taken with an **iPhone 4Gs** using his built in HDR technology

Original shot



Final result using iPhone HDR algorithm

Images taken with a Nokia N8 with **very bad light** conditions:

(C) Copyright Nokia 2014. All rights reserved.

Original shot

Final result using QHdrCamera component

Performance & Limitations

Taking the three "seed" images takes significant time, as does the subsequent calculation of the final HDR image. The table below shows the approximate time (in seconds) to take the initial photos and then to create an HDR image on the Nokia N8.

Resolution	Shooting time	Processing time
2000x1500	6sec	16sec
1024x768	4sec	3sec
800x600	2.5sec	1.5sec
640x480	2.5sec	1sec

All operations are time consuming and require a huge amount of resources:

- **CPU**. The 600Mhz processor on the Nokia N8 is not particularly fast, so we usually have to work with low resolution images in order to keep the calculation time down to an acceptable level. The1.3GHz processor on the Nokia 808 should allow higher resolution images to be calculated in a reasonable time.
- **Memory**. Starting from 256MB of available RAM it's impossible to allow very high resolutions.
- Mobile devices can't provide the same exposure range compared to professional cameras.

Despite the above limitations, the HDR element demonstrated by this article can provide much better images in difficult light conditions than that provided by the default camera app. After all, much better to capture those special moments at some resolution, than to get a blank image at higher resolution.

Recommendations for using HDR

HDR is recommended for the following use-cases:

- **Landscape**: A classic use of HDR in regular photography is on landscape shots with a bright sky above the horizon line and darker foreground below.
- **Outdoor portraits**: The midday sun’s harsh light usually makes for unflattering portraits. It can cast strong shadows on a person’s face and create dark circles around the eyes. It also bounces off of skin and accentuates shiny spots. HDR can minimize the effect of these two extremes and create a more evenly lit portrait.

HDR is less effective for:

- **Capturing motion**: To create an HDR image, `QHdrCamera` takes three photos in quick' succession. If you're taking a photo of a fast moving subject or if you move the phone while shooting, the final HDR image will show "ghosting". That's when the multiple images aren't aligned and objects appear in multiple places. If you experience this problem frequently with your HDR images, consider mounting the phone on a tripod.
- **Capturing vivid colors** : The HDR mode can bring colors back into blown-out or dark areas. But when photographing brightly colored subjects that are properly exposed, HDR results in a disappointing desaturation. If the allure of your image is that it shows vivid colors, HDR is not the best choice.

Using the component

1. Download source code from QHdrCamera

(C) Copyright Nokia 2014. All rights reserved.

2. Include `QHdrCamera` directory into your project

.pro

```
#include(./QHdrCamera/hdr.pri)

symbian: TARGET.EPOCHEAPSIZE = 0x20000 0xC000000
```

We highly recommend increasing the heap size available to the app, as shown above.

main.cpp

This code shows how to add the HdrCamera Element so that it is available for use within QML.

```
#include <QtGui/QApplication>
#include "qmlapplicationviewer.h"

#include <QtDeclarative>
#include "qhdrcamera.h"

Q_DECL_EXPORT int main(int argc, char *argv[])
{
    QScopedPointer<QApplication> app(createApplication(argc, argv));

    QmlApplicationViewer viewer;

    qmlRegisterType<QHdrCamera>("com.imaging.hdr", 1, 0, "QHdrCamera");

    // Resize the root QML element to view size
    viewer.setResizeMode(QDeclarativeView::SizeRootObjectToView);

    // Performance optimization flags
    viewer.setAttribute(Qt::WA_OpaquePaintEvent);
    viewer.setAttribute(Qt::WA_NoSystemBackground);
    viewer.viewport()->setAttribute(Qt::WA_OpaquePaintEvent);
    viewer.viewport()->setAttribute(Qt::WA_NoSystemBackground);
    viewer.setViewportUpdateMode(QGraphicsView::FullViewportUpdate);


    viewer.engine()-
>addImportPath(QString(QApplication::applicationDirPath()+"/qml/module") );
    viewer.engine()-
>addImportPath(QString(QApplication::applicationDirPath()+"/../qml/module") );

    viewer.setOrientation(QmlApplicationViewer::ScreenOrientationLockLandscape);
    viewer.setMainQmlFile(QLatin1String("qml/QHdrSample/main.qml"));
    viewer.showExpanded();

    return app->exec();
}
```

QML

The following code shows how to use the `QHdrCamera` QML element:

- Declare the component along with a resolution and exposure range if the default values (shown commented out below) are not acceptable.

(C) Copyright Nokia 2014. All rights reserved.

- Call `captureImage()` to start taking the images.
- Component provide a built in preview management through boolean `preview` property, anyway you can implement `onFinished()` to display or handle the image once all calculation has been finished. In this example we've used the custom `ImagePreview` component to display the image, based on the `onFinished()` signal's parameters `pathResultImage` and `pathOriginalImage`. There is nothing to stop you rolling out your own viewer component.
- `availableDevices` is an array property with the list of all available camera devices on your phone. Typically on first entry you will find the main camera.

```
import QtQuick 1.1
import com.nokia.symbian 1.1
//import com.nokia.meego 1.0
import com.imaging.hdr 1.0

Rectangle {
    id: mainPage
    width: 640
    height: 480
    color:"black"

    QHdrCamera {
        id:camera
        anchors.fill: parent
        //resolution: Qt.size(2000,1500)
        resolution: Qt.size(1024,768)
        exposureRange:2

        onStarted: {
            busy.running = true; busy.visible = true;
        }

        onFinished: {
            //console.log(pathResultImage); console.log(pathOriginalImage)
            busy.running = false; busy.visible = false;
            camera.preview = true;
            previewb.text = "Close Preview"
        }

    }

    Column {
        spacing: 40

        Button{
            id:shot
            text: "Shot"
            onClicked: {
                camera.captureImage()
            }
        }

        Button{
            id:previewb
            text: "Show Preview"
            onClicked: {
                camera.preview = !camera.preview
                if( camera.preview )
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    previewb.text = "Close Preview"
                else
                    previewb.text = "Show Preview"
            }
        }


        Button {
            id:close
            text: "Close"
            onClicked: {
                Qt.quit();
            }
        }
    }

    BusyIndicator {
        id: busy
        anchors.centerIn:  parent
        running: false
        visible: false
        width: 80
        height: 80
        z:2000
    }

    Component.onCompleted:{
        camera.start(camera.availableDevices[0])
    }
}
```

Component behavior

QHdrCamera

`QHdrCamera` takes three photos in a sequence when `captureImage()` is called, each with a different exposure level based on the range specified in the `exposureRange` property.

The resulting HDR image is saved on default's phone image directory with the prefix **hdr**, followed by a timestamp, in a **jpeg** format.

Signal `finished(string pathResultImage, string pathOriginalImage)` provides information on where image is saved so that it can be displayed in the preview.

ImagePreview component

`ImagePreview` QML element provides a simple HDR image preview: users swipe left or right cycle between the original (exposure=0) and resulting HDR image. This is implemented as a `ListView`.

API Reference

Properties

- `exposureRange` : real - sets the exposure working range. The default value of 2 means that three photos will be taken: the first at exposure 0, the second with exposure +2 and the third with exposure -2. Setting 1 the range will be -1,0,1.
- `captureResolution` : size - The resolution to capture the image at. If empty, the system will pick a resolution of 800x600

Slots

- `captureImage()` - starts the process of capturing three images with different exposure ranges.

(C) Copyright Nokia 2014. All rights reserved.

Signals (handlers)

- `onFinished(string pathResultImage, string pathOriginalImage)` - handler for the `finished(string pathResultImage, string pathOriginalImage)` signal. Invoked when the rendering process is finished.

Variable `pathResultImage` contains the location of the resulted saved image, `pathOriginalImage` contains the location of the original saved image, useful for the preview process.

HDR Algorithm Overview

Warning: This section isn't for the faint-hearted! You also don't need to read this in order to use or broadly understand how the component works. It will however be of interest to power users and anyone who wants to collaborate to improve the component.

The algorithm I've used comes from the paper "Multi-exposure Imaging on Mobile Devices"[1], a collaboration between Nokia Research Center and Stanford University. The reference provides full details - but here is a brief overview.

To provide an HDR Image we need to record a number of images at different exposures, and then merge them using an *exposure fusion* algorithm. Exposure fusion computes a scalar-valued weight map for each image, and performs a weighted blend of inputs to obtain the final result.

The blending is computed using a multi-resolution algorithm that build a Laplacian pyramid of the inputs and a Gaussian pyramid of the maps, blends each layer separately and then collapes the resulting Laplacian pyramid to obtain the result.

Let (I_1 .. I_n) the stack of n input images captured by the camera (in that context just three). The weight for each pixel (*i*,*j*) of image *k* is computed as:

$$W_k(i,j) = \exp(-\frac{(I_k(i,j) - \mu \cdot 255)^2}{2(\sigma \cdot 255)^2}).$$

(1) Gaussian filtering

Current parameters used are μ=0.5 and σ=0.2.

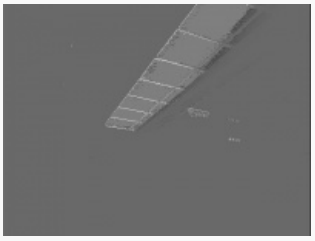
Tone Mapping. The result of Gauss filter (1) for each image. Interesting to note the **different light details** on each image.

Exposure: -2 (Darker)

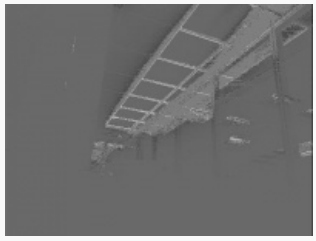
Exposure: 0 (Original Image)

Exposure: +2 (Brighter)

Laplace application on Gauss filter result:



Exposure: -2 (Darker)



Exposure: 0 (Original Image)

Exposure: +2 (Brighter)

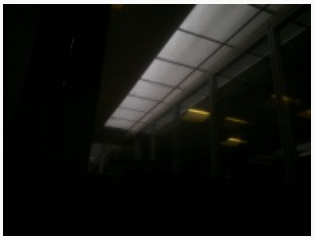
The result consists of multiplyng the images by their weight maps and blending:

(C) Copyright Nokia 2014. All rights reserved.

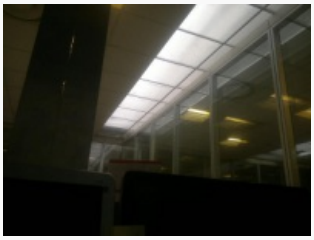
$$R(i,j) = \sum_{k=1}^{n} W_k(i,j) \cdot I_k(i,j).$$

(2) Blending formula

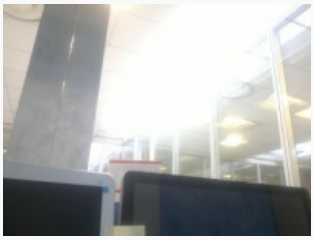
Images taken with a **Nokia N8** using `QHdrCamera` QML Element



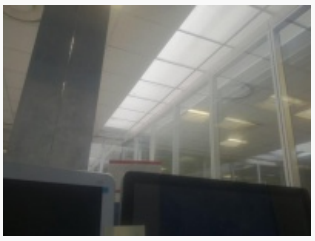
Exposure: -2 (Darker)



Exposure: 0 (Original Image)



Exposure: +2 (Brighter)



Result after HDR algorithm used by QHdrCamera component

For performances reasons HDR algorithm steps above all had to be implemented in C++ rather than within QML javascript. For the curious, you can see the source code QHdrCamera. The most important methods of `QHdrCamera` class are:

- `process()` - implements all steps described above
- `gauss()` - implements all steps described by formula (1)
- `laplace()` - implements the Laplacian operation

Future evolution

This article describes the first "alpha" version of the project. While a huge amount of work has been done to make it an effective "proof of concept" there is still a lot more that could be done. For example:

- Dynamically exposure range calculation
- Better camera motion compensation algorithm
- Inclusion of metadata information inside the header image like gps location
- Generic performance improvements

I plan to continue this development based on user feedback and interest. Please add your comments in the Comments section below.

References

1. ↑ Multi-exposure Imaging on Mobile Devices Natasha Gelfand, Andrew Adams, Sung Hee Park, Kari Pulli. 2010

Downloads

- Sample application

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.