Secondary camera in QML

This article explains how to use the secondary (front-facing) camera in QML

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

Introduction

The reference library topic Using the device camera describes how to use the primary camera in various ways. However, access to the secondary camera is only available to C++ apps.

This article describes some options for using the secondary camera in QML.

Basic solution

The secondary camera (i.e. front-facing camera) is available though the C++ API. The basic solution is to create a declarative component and expose this to QML.

First include multimedia in the project file.

```
CONFIG += mobility
MOBILITY += multimedia
```

The C++ part of the FrontCameraApp component is shown below:

```
#include <QGraphicsVideoItem>

FrontCameraApp::FrontCameraApp(QDeclarativeItem *parent) : QDeclarativeItem(parent)
{
   myViewfinder = new QGraphicsVideoItem(this);
   QByteArray frontCamera = QCamera::availableDevices()[1];
   myCamera = new QCamera(frontCamera);

   myCamera->setViewfinder(myViewfinder);
   myCamera->start();
}
```

Lastly the FrontCameraApp object has to be registered for QML in the main.cpp file.

```
    qmlRegisterType<FrontCameraApp>("cz.vutbr.fit.pcmlich", 1, 0, "CameraFront");
```

Then use the new `CameraFront` element in QML is as follows:

```
import com.nokia.meego 1.0
import cz.vutbr.fit.pcmlich 1.0
Page {
  CameraFront {
    anchors.fill: parent;
  }
}
```

Advanced solution/workaround

The basic solution above does not provide the methods which are in the QML Camera Element. If you need more advanced functionality through the front-facing camera one approach is to simply modify the existing QML Element for the main camera to work with the front-facing camera.

First download the source code from gitorious. The implementation of the Camera element is in /plugins/declarative/multimedia/qdeclarativecamera.cpp. Change the code as follows:

```
QDeclarativeCamera::QDeclarativeCamera(QDeclarativeItem *parent) :
    QDeclarativeItem(parent),
    m_camera(0),
    m_viewfinderItem(0),
    m_imageSettingsChanged(false),
    m_pendingState(ActiveState),
    m_isStateSet(false)
{
//    m_camera = new QCamera(this); // remove this line
   QByteArray frontCamera = QCamera::availableDevices()[1]; // add this line
   m_Camera = new QCamera(frontCamera); // add this line
    ...
}
```

Next, it is necessary to add included private header files to your project.

(C) Copyright Nokia 2013. All rights reserved.

```
#include "qdeclarativecamera_p.h"
#include "qdeclarativecamerapreviewprovider_p.h"
```

There could be a conflict in the class names `QDeclarativeCamera` and other problems, so it is recommended to perform refactoring of `QDeclarativeCamera` class. Finally, the modified class is registered into QML, as is described in the previous section.

Flip the image

The image from the secondary camera could be confusing to the user. Therefore it is appropriate to flip it.

```
        transform: Rotation { origin.x: frontcam.width/2; origin.y: frontcam.height/2; axis { x: 0; y: 1; z: 0 } angle: 180  }
```

Limitations

The primary and secondary cameras cannot be opened at same time.

Summary

After these modifications, we have a QML component for the secondary camera which allows us modify white balance mode, exposure compensation, and other configuration of the secondary camera. It is also possible to capture images and monitor camera states. This solution is only a workaround, since the Qt Mobility is not providing a suitable interface for the secondary camera. All mentioned components are available in the attached file: Media:SecondaryCameraInQml.tar.gz

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.