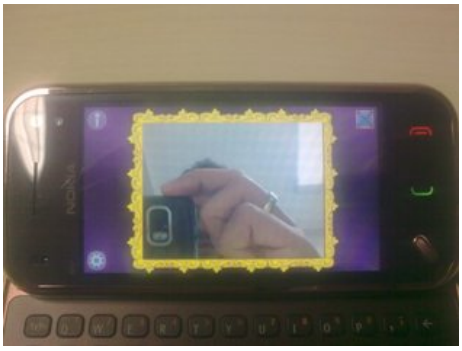
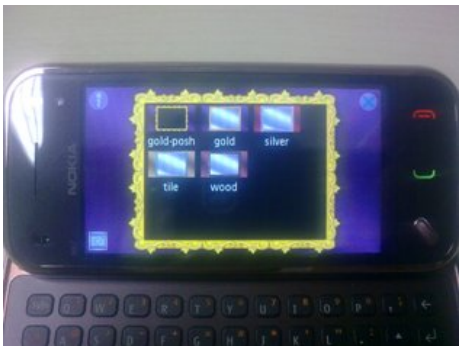
Qt mobility application: QtMirror

The QtMirror based application turns your phone into a mirror using live video captured from the secondary camera of Symbian devices like: N97,N97 mini, Nokia 5800, etc. The application uses the QtMobility classes to access the camera and show the images and QWidget classes for the UI.

22 Apr 2010





The graphics are courtesy of Signove funlabs Funlabs

The Project

The QtMirror is a project under development. It is available under Nokia Developer Hosting Service (Nokia Developer Projects). Please download the project from here.

To build the project you will need the QtMobility version 1.0.0 installed on you SDK for S60 5th edition.

Capabilities

As the application is quite simple it depends only on one capability:

```
TARGET.CAPABILITY = UserEnvironment
```

Code snippets

Locking screen orientation to landscape

The code bellow shows the 'main.cpp' file that contains code to lock the screen orientation to landscape. Note that the **Q_OS_SYMBIAN** macro is used to define code that is only valid to Symbian platform.

```
#include "Mirror.h"

#include <QtGui>
#include <QApplication>


// Needed Symbian specific headers
#ifdef Q_OS_SYMBIAN
#include <eikenv.h>
#include <eikappui.h>
#include <aknenv.h>
#include <aknappui.h>
#endif
#include <unistd.h>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
```


(C) Copyright Nokia 2013. All rights reserved.

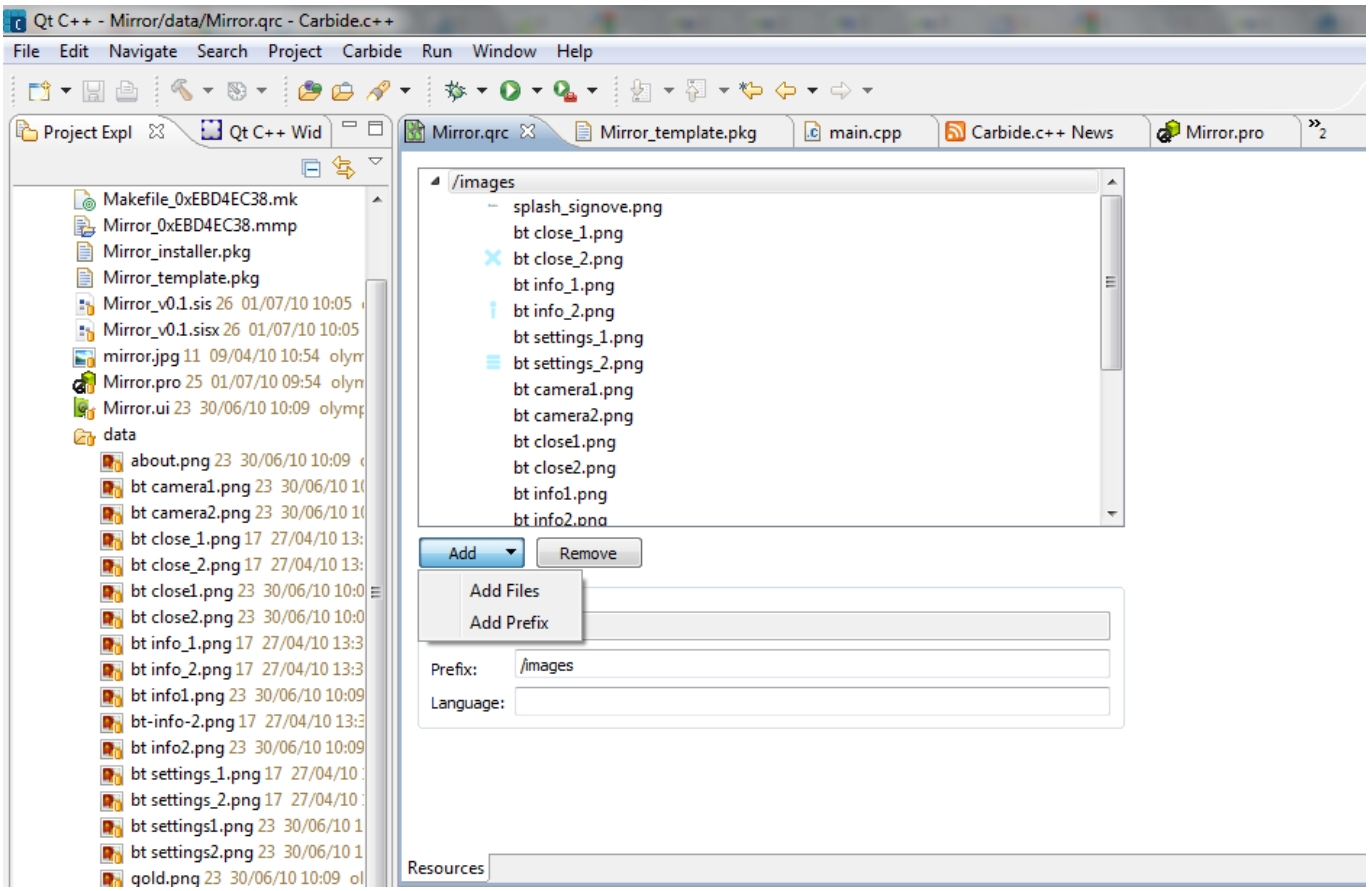
```
    // Symbian specific code
    #ifdef Q_OS_SYMBIAN
    CAknAppUi* appUi = dynamic_cast<CAknAppUi*> (CEikonEnv::Static()->AppUi());
    TRAP_IGNORE(
        if (appUi) {
            // Lock application orientation into landscape
            appUi->SetOrientationL(CAknAppUi::EAppUiOrientationLandscape);
        }
    );
    #endif

    . . .
}
```

Display two timed splash screens

The code bellow follows the previous one and shows how to display two splash screens before the application start. Note that the last screen is synchronized with the beginning of the widget so that it only disappear when the widget is ready to be shown.

```
. . .

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    . . .

    QPixmap pixmapSignove(":/images/splash_1.png");
    QSplashScreen splash(pixmapSignove);
    splash.show();
    sleep( 2 );

    QPixmap pixmapMirror(":/images/splash_2.png");
    splash.setPixmap(pixmapMirror);
    splash.repaint();
    sleep( 2 );

    Mirror w;
    w.showFullScreen();
    splash.finish(&w); //line used to synchronize with the widget ready status
    return a.exec();
}
```

It should be noticed that the image files are contained by the Qt Resource System. If you are using Carbide.c++ the resource file has an .qrc extension. Double click on this file opens an editor view where you can add files, changes prefix (in our case :/images/), and more. The picture bellow shows how to add a file to an existing prefix. For more information regarding the Qt resource System access this link.



Setup the video widget and the camera

To display the camera video onto the screen we will need to use two classes of the QtMobility project: QVideoWidget and QCamera.

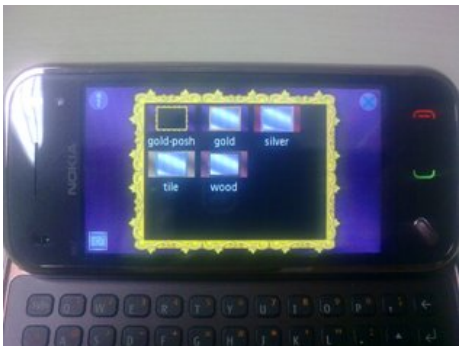
The code bellow shows the constructor of the main widget of the QtMirror application as well as the the two functions need to setup the video and the camera. Note that we define the mirror area to be at the center of the screen in the rect MIRROR_SIZE.

(C) Copyright Nokia 2013. All rights reserved.

```
const QRect MIRROR_SIZE = QRect(129, 37, 389, 292);

Mirror::Mirror(QWidget *parent) :
    QWidget(parent), camera(0), videoWidget(0)
{
    ui.setupUi(this);
    setupVideo();
    setupAbout();
    setupCamera();
    setupSettings();

    . . .
}

void Mirror::setupVideo()
{
    videoWidget = new QVideoWidget(ui.frame);
    videoWidget->setObjectName(QString::fromUtf8("videoWidget"));
    videoWidget->setGeometry(MIRROR_SIZE);
}

void Mirror::setupCamera()
{
    QByteArray cameraDevice = QCamera::availableDevices()[1];//selecting the secondary camera
    camera = new QCamera(cameraDevice);

    connect(camera, SIGNAL(error(QCamera::Error)), this, SLOT(error(QCamera::Error)));

    camera->setFocusMode(QCamera::AutoFocus);

    videoWidget->setMediaObject(camera); //setting the media object

    if (camera->state() == QCamera::ActiveState) {
        camera->stop();
    }
    camera->start();//starting the camera
}
```

Configuring the settings view

The QtMirror application has an settings view that allows the user to select different backgrounds. These backgrounds are used as frames to the mirror. The settings view is placed over the video widget to save space.



This view is implemented with a QListWidget with a grid view. When the user selects any item of this list the background changes automatically allowing a better view of the design.

The QListWidget was added using the Qt Design. The code bellow shows how to fill this list with background images contained by the Qt Resource System.

```
void Mirror::setupSettings()
{
    QDir dir(":/images/background", FILTER_PNG);
    QFileInfoList list = dir.entryInfoList( QDir::Files );
    for (int i = 0; i < list.size(); ++i)
    {
        QIcon icon(list[i].absoluteFilePath());
        QListWidgetItem* item = new QListWidgetItem(icon, list[i].baseName(), ui.listWidget );
    }

    //connect signals here to change the background under user selection.
    connect(ui.listWidget,SIGNAL(itemClicked(QListWidgetItem*)),
        this, SLOT(changeBackground(QListWidgetItem*)));

    . . .
}
```

Changing the background through Qt Style Sheets

Changing the background image when the user selects the item on the Settings view showed above is quite simple. You will need to implement a function that handles user-press events in each `QListWidgetItem`. This function is shown in the code below:

```
void Mirror::changeBackground(QListWidgetItem* item)
{
    //: List background picture
    QString style = tr("background-image: url(:/images/background/%1.png);").arg(item->text());
    ui.frame->setStyleSheet(style);
    update();
}
```

We use a QString to define a style sheet to set a new background image. [Qt Style Sheets http://doc.trolltech.com/4.6/stylesheet.html], which are much similar to Cascading Style Sheets (CSS), allow for an easy redefinition of styling aspects. The style are defined with its own syntax (for more information

(C) Copyright Nokia 2013. All rights reserved.

about style sheet syntax take a look at this link). With this syntax we define that the new background image is the same the user has click[illegible] `QWidget::tr()` allows translators to influence the styling, the comment above starting with `//:` is passed on to the translator through Qt's translation system. If this is not needed, you can safely remove this call. After setting the new style sheet of the main frame we call the base class function **update()** to update the widget.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.