Qt for N8
Hands-On

Instructions

digia

Trademarks and Acknowledgements

The exercises for this training have been developed by Digia Plc Training team.

Adaptive Flashlight

Objectives

Play around with the Sensors API

References

Exercise starting point:

\templates\Flashlight

Exercise example solution:

\qt_hands_on_solutions\Flashlight

Overview

In this exercise we will create a very simplistic, adaptive "flashlight" application: When it's dark, the screen will be white and when it's less dark it will be gray and in full sunlight it will be black:

		
Dark	Bright	Sunny

Practical Outline

1. Open the **project template** in QtCreator from the set of codes distributed. The template creates a white screen.
2. Try in QtSimulator to see it works.
3. Add a QAmbientLightSensor for your mainwindow
    a. Remember to modify the .pro file and to use the mobility name space
4. Create your own slot function where you will change the color of the screen (graphicsview's background brush) depending on the reading of the sensor
5. Start listening to changes in the reading
6. Voilá! Try in QtSimulator and in device!
7. Optional: Add eye-Candy, publish in Ovi store, make a million dollars

Qt Mobility Hands-On: Locationer

Objectives

Try using the new QtMobility APIs with an application called "Locationer"

References

Exercise starting point:

From the scratch!

Exercise example solution:

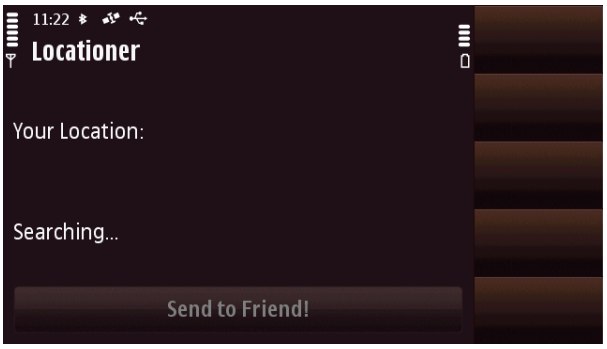
\qt_hands_on\Solutions\Locationer

Overview

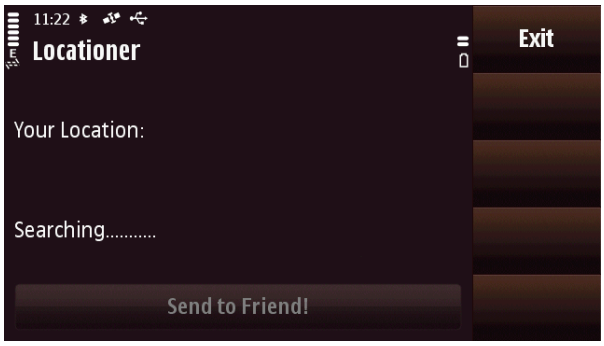
In this exercise we will do a very simple application that shows the location of the user and allows the user to send his GPS coordinates to a friend with a SMS.

> ***N.B.*** *In order to do this exercise you will need to have the Mobility APIs installed both for your development environment and device. The device also needs to have a GPS receiver.*
>
> *If you do not have a device, you can try to implement only the part of the application that shows the location but does not allow sending an SMS. The Simulator in Nokia Qt SDK supports GPS simulation well enough.*

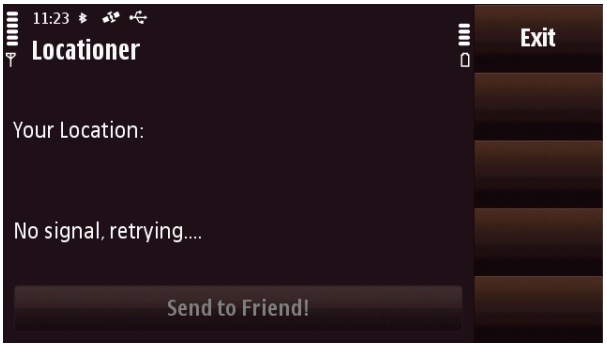
You have rather free hands in implementing this how you wish, but here are few screenshots of one solution that simplify the idea of the application:



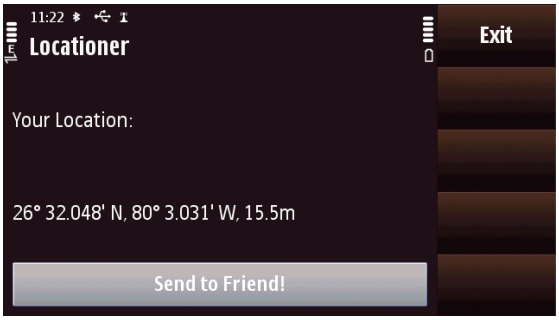
Searching for GPS Signal



Still searching… (more dots, one added each second).



First try did timeout, re-try



Finally, coordinates! These can now be send to a friend by clicking the button…

Which opens the default editor for SMS (S60 Messaging app here) with premade contents.

Practical Outline

1. Start a new Qt application from the scratch

2. Implement suitable UI contents with Qt Designer (you can do it how ever you wish)

3. Implement some logic to find the user's location and show the result in your UI. The following classes might interest you:
   - `QGeoPositionInfoSource`
   - `QGeoPositionInfo`
   - `QGeoCoordinate`

   Things you might want to take into account:
   a. No GPS signal found immediately
   b. No GPS signal found after timeout
   c. You can choose the way coordinates are presented (or let user choose!)
   d. What if user is moving, do you want to update the coordinates even after they are found?
   e. While GPS is trying to locate (can take several seconds), some animation could be helpful for the user, like just increasing the amount of dots in the text (see screen shots).
   f. Sending an SMS should not be possible before the location is found
   g. Using GPS in S60 requires the capability "Location". Also in the .pro file, you need to modify the field with "`MOBILITY =`" by adding "`location`" and "`messaging`"

4. Then, when coordinates are found, generate a `QMessage` with the suitable contents and launch the default editor for sending the message. You will also need class `QMessageService` to do this.

Tips for installing/executing the application in the device:

From Nokia Qt SDK, you can do this directly by following the instructions in document "Getting started with Symbian device development" which is found under the Nokia Qt SDK Start Menu folder. For Symbian devices, this should not be a problematic step.

Animations!

Objectives

To develop mobile applications using the Nokia Qt SDK and to play around with the Qt Animation Framework and graphics effects.

References

Exercise starting point:

From the scratch!

Exercise example solution:

\qt_hands_on_solutions\Animations

Overview

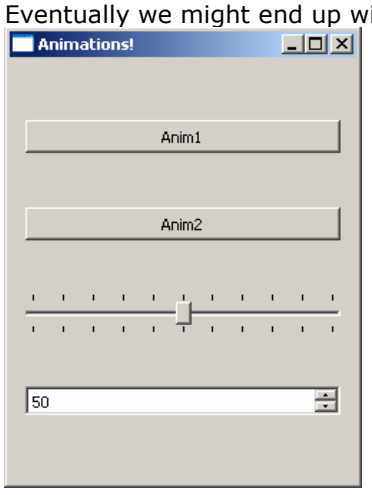
In this exercise you will explore the Animation Framework and graphical effects. We will first start by just adding an extremely simple animation and then extend our application a bit by adding more animations.

Practical Outline

1. Open Qt Creator from Nokia Qt SDK and create a new project. Select "Mobile Qt Application". Give whatever name you wish for your project, select a desired folder. Select the platforms you want to develop for (at least Desktop and the Qt Simulator, if you have a Symbian/maemo device, then select those as well). For the other selections you can rely on the default ones.

2. Qt Creator creates a new project with already some files included. From the "Forms" folder you can find the .ui file, which is the Qt Designer (or *Form Designer*) file. Open the file and you will have the UI Designer view where you can design your applications UI layoutb.

   Eventually we might end up with something like this:



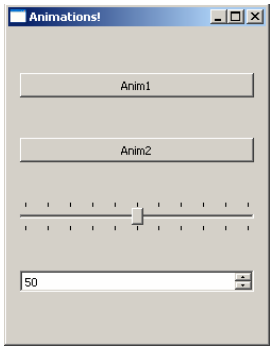
3. Play around with the UI Designer checking what can be done with it *and start with two push buttons in a vertical layout*.

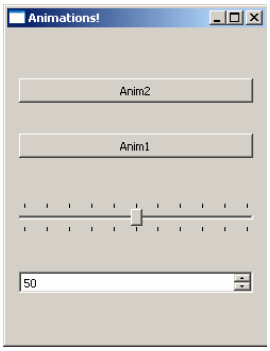
4. When the initial UI layout is ready you can start modifying your QMainWindow class, the one you just designed:

   a. Open mainwindow.h and add a new slot function called "`void runAnim1()`"
   b. Implement the slot function to mainwindow.cpp. Inside, create an animation of your own that for instance moves the two push buttons to each others' places and back. Start with a simple `QPropertyAnimation`

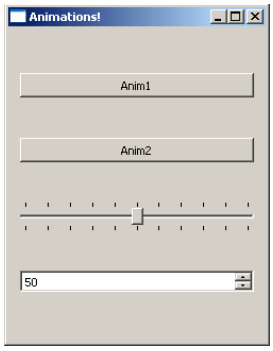
modifying the property "geometry" of one button and then later extend your animation to both buttons with a `QParallelAnimationGroup`.

Idea of the animation:





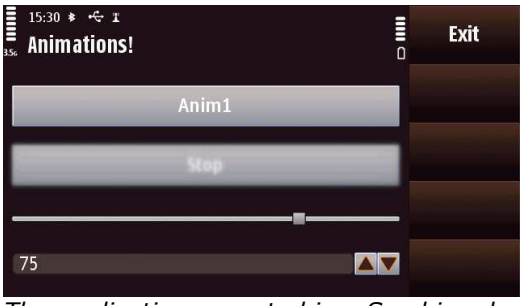
In the middle of the animation the buttons have switched places



And then return back

5. Try your animation with different target platforms, the desktop build, Qt Simulator, Symbian device etc.

N.B. If you want to execute your application in Symbian device, you need to first start the "TRK" application and connect with USB before Qt Creator can launch the application there.



The application executed in a Symbian device

6. Add a graphics effect (`QGraphicsBlurEffect` for instance) for one of the buttons.

7. Try animating the graphics effect itself (on/off during some time duration). If you want to make the animation endless, you can try function `setLoopCount()` with value -1.

8. Add a slider and a spin box to your UI layout (in the UI Designer), like in the screen shots. You can try connecting a signal directly here in the Designer between these two elements (`valueChanged(int)->setValue(int)`), but eventually remove the connection, as we are up to something cooler.

9. Make the slider move "smoothly" depending on what was set to the spin box.

a. Create a new slot function where you will animate the moving of the slider to a given new value (by modifying the slider's property "`value`").
b. Connect your spin box's signal `valueChanged(int)` to your slot function
c. Try your application and be amazed by *Your Ultimate Qt Animation Skillz* as the slider moves smoothly from a value to other!

10. Play around with different `QEasingCurves` for the animations (`setEasingCurve()`) and observe the effects