Haptics effects in Qt

This article shows how to provide haptic feedback to users (either through vibration or audio) using the Qt Feedback API. This could be used in games or UIs to improve the user experience.

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee (02 Mar 2012)
Qt Quick should be used to demonstrate this API - UI development on mobile devices using Qt C++ is deprecated.

Before we start

First of all, it's important to understand the concept of actuators. An actuator is a device responsible for providing tactile feedback. It's like a tiny motor that produces a physical effect to a command sent by our application.

The Feedback API abstract those devices through the QFeedbackActuator class. Those devices are identified by the `id()` and have a `name()`. Please see QFeedbackActuator for the complete documentation.

The actuators are capable of providing different effects. For example, on the Nokia N8 the actuator named "Touch" is capable of providing feedback when the user touches the screen. The actuator named "Vibra" is the one that works when producing vibration (like the one that occurs when the phone is ringing). To know which actuators exists in the device, the API provides this method:

```
QList<QFeedbackActuator *> list = QFeedbackActuator::actuators () ;
```

With that list in hand you are able to query which actuators you can use.

Making it vibrate

Now we're going to show how to make the phone vibrate. The API provides a class named QFeedbackHapticsEffect that abstract the concept of “feedback effect”.

How it works

The effect is composed of three parts: start, middle, and end. The start part is called “attack”, and the end part is called “fade”. The API provides methods to configure all those parts. The end result is that the total time the effect lasts is “attack” + main part + “fade”. Let's see an example:

```
QfeedbackHapticsEffect effect;

effect.setIntensity (0.5);
effect.setDuration (1000);

effect.start();
```

This piece of code configures the effect to last for 1 second (1000 ms) and have an intensity of 50%. The intensity is a number between 0.0 and 1.0, with this last value meaning “full intensity”. The method `start()` starts playing the effect, and `stop()` is used to stop playing it.

What about the actuators?

The device plays the effect using one of its actuators. In the example, it uses the “default actuator”. In the N8, this happens to be the “Touch” actuator. So, how can we make the device vibrate? We need to change the actuator!

```
// get the actuator list
QList<QFeedbackActuator *> list = QFeedbackActuator::actuators () ;
```


(C) Copyright Nokia 2014. All rights reserved.

```
// get the Vibra
QFeedbackActuator * vibra = 0;

foreach (QFeedbackActuator * a, list)
{
  if (a->name() == “Vibra”)
  {
      vibra = a;
   }
}

// if vibra is not found, you device does not have it
if (vibra == 0)
 // handle error

// let's change the actuator
effect.setActuator (vibra);
```

And that's it.

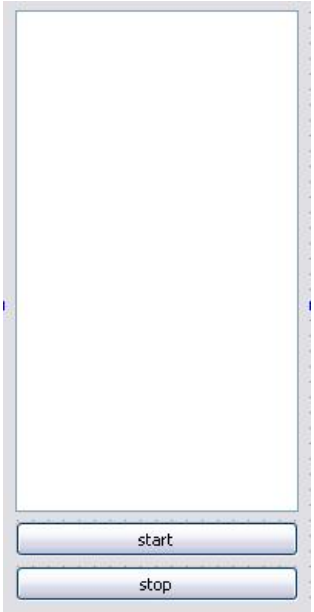
How about the other parts of the effect?

Use the following methods from QFeedbackHapticsEffect to change the other two parts of the effect. By default their duration is zero milliseconds.

- `void setAttackIntensity (qreal intensity)`
- `void setAttackTime (int msecs)`
- `void setFadeIntensity (qreal intensity)`
- `void setFadeTime (int msecs)`

Short example

Our example uses this user interface:



It has two button and a plain text box.

Here's our Widget.h file:

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
```


(C) Copyright Nokia 2014. All rights reserved.

```
#include <QFeedbackActuator>
#include <QFeedbackEffect>
using namespace QtMobility;

namespace Ui {
    class Widget;
}

class Widget : public QWidget
{
    Q_OBJECT

public:
    explicit Widget(QWidget *parent = 0);
    ~Widget();

   void put(const QString & msg);

public slots:

   void startClicked();
   void stopClicked();

private:
    Ui::Widget *ui;
    QFeedbackHapticsEffect rumble;


};

#endif // WIDGET_H
```

We need to include the files for the actuator and feedback effect. The classes of Qt Mobility are under the `QtMobility` namespace. We are declaring it in the file for convenience. The public slots are connected to the button events. The `put()` method is a convenience method to print text in the text box.

Now here is the implementation file:

```
#include "widget.h"
#include "ui_widget.h"

// convinience method to print text
void Widget::put(const QString & msg)
{
   ui->plainTextEdit->insertPlainText(msg);
   ui->plainTextEdit->insertPlainText("\n");
}


Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);

    // connect signals and slots
```


(C) Copyright Nokia 2014. All rights reserved.

```
    ...


    // here we list which actuators are there on the phone
    QList<QFeedbackActuator *> list = QFeedbackActuator::actuators();
    put (QString ("num actuators: %1").arg(list.count()));

    QFeedbackActuator* vibra = 0;


   // now we pick the vibra actuator
   foreach (QFeedbackActuator* a, list)
    {
      put (a->name());

      if (a->name() == "Vibra")
         vibra = a;
     }

    if (vibra == 0)
    {
      put ("vibra not found");
      return;
    }

   rumble.setIntensity(1.0);
   rumble.setDuration(1000);
   rumble.setActuator(vibra);
}

Widget::~Widget()
{
    delete ui;
}


void Widget::startClicked()
{
   rumble.start();
}

void Widget::stopClicked()
{
   rumble.stop();

}
```

Finally, here's the .pro file:

```
QT       += core gui

TARGET = FeedbackDemo
TEMPLATE = app


SOURCES += main.cpp\
```


(C) Copyright Nokia 2014. All rights reserved.

```
        widget.cpp

HEADERS  += widget.h

FORMS    += widget.ui

CONFIG += mobility
MOBILITY = feedback

symbian {
    TARGET.UID3 = 0xeb02a514
}
```

References

- Qt Mobility 1.2

(C) Copyright Nokia 2014. All rights reserved.