CT30A9300 Code Camp on Communications Engineering 30.10.2010

QT Code Camp

Group 1, Qute Translate

Authors:

Rostislav Malevich 0312328

Oskar Sonninen 0326770

Petri Ryhänen 0334856

email: firstname.surname@lut.fi

Qute Translate

Description:

Qute Translate is an application made in Qt framework. It is automatic sms translation program that translates incoming, send and receiver sms-messages from source language to target language. Qute translate does not disturb user if source- and target languages are same. Program can also translate already received messages for later examination. One can also answer to received text messages with preferred language as Qute translate can also translate users written text. Qute Translate is specifically targeted to Nokia N900, but as the program is made with Qt it should work also on devices like Nokia N8 which is a symbian³ smartphone.

Problem:

The idea for Qute translate came from exchange students who received text messages in a foreign language during their exchange time. They can of course copy their message to some translation application, but that is very time consuming and that opportunity is not always available .

Development:

Program can be further optimized by replacing the now used whole Google translate page fetching and parsing with javascript queries from Google translate by QTScript. That way would reduce lots of network traffic needed for translation. We dropped espeak but it was very easily implementable to Nokia N900. You just needed to install espeak to you N900 mobile phone and the program can use it through command line parameters. We were also researching the possibility to make the program a daemon but the gain didn't appear to be worth of the effort. If you want to make the program work properly with other mobile phones than N900 some user interface optimization is needed.

Non-technical description:

Qute Translate tires to provide one integration to sms translation problem so user can do all his sms related translations in one application. Qute translates interface is very similar to N900's so it very intuitive for former N900 users. It won't bother the user if target language and source languages of an incoming text message is the same. As Qute Translate is made with Qt it is portable to a variety of phones like Nokia N97, N900 and N8 so with one code developers can cover many phones.

Technical description:

Interface:

Qute Translates interface is done using Qwidgets which is a class provides by Qt framework. Qute Translate have four different widgets that make the whole programs UI. Main widget is created when program start. This main widget then creates other three widget and acts as a parent to all of them. Main widgets layout provides view screen to messages that is used to view received messages. Main widget contains buttons which are used to navigate to other widgets.

Translation:

Translation is implemented using Qt Qhttp library connections to Google translate web-service. Qhttp generates a appropriate request from received message and language inputs. This request is send to Google and return is parsed to fit Qute Translates purposes.

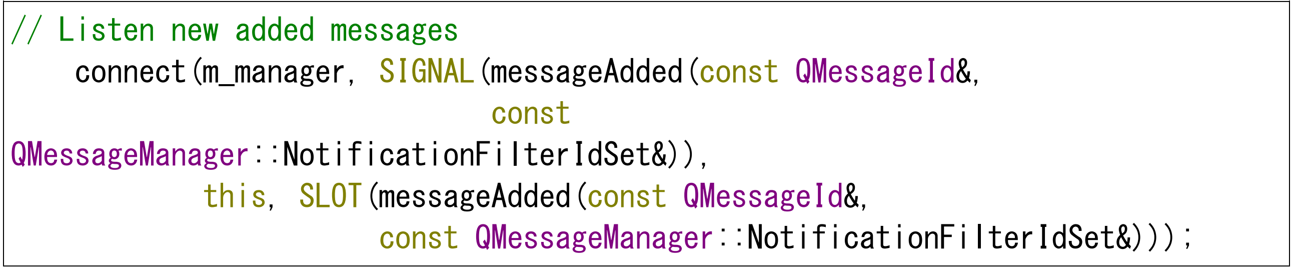
Translation is triggered when Qute Translate catches a signal that is emitted when a sms message is added to phone storage. Qute Translate opens this message and check if its not an the target language. If the message is same as the translation then Qute Translate won't do anything. If translation differs from the original message then the main widget launches to screen and shows the translated text to user.

If user wants to respond to received message he presses reply button and reply_message widget is opened and the senders name is filled automatically to correct field. Then user can type his message to text area and select to what language he wants to translates this message and press translate button to convert the text. When text is translated user will press the send button and the sms message is send.

Signals:

Signal are used to trigger actions in Qt. In our application signals are used in many buttons to trigger other widget. One example could be to change main widget view to message widget view. This happens when user clicks messages button in main widget. Messages button emits a signal that is captured to a specific slot that tell what function is to triggered. In this case changeViewToContact function is invoked and contacts widget is brought to front. Example of a signal and slot implementation is shown in Image 1.

Image 1:



```
// Listen new added messages
    connect(m_manager, SIGNAL(messageAdded(const QMessageId&,
                                    const
QMessageManager::NotificationFilterIdSet&)),
            this, SLOT(messageAdded(const QMessageId&,
                            const QMessageManager::NotificationFilterIdSet&)));
```

Contacts and listing:

Contact are managed in Qt through QMessaging library. List are handled with QList and QListWidget. Former is list specifically designed to contain Widget and QList to to contain other structures like QStrings and QMessageIds. In Qute Translate Contacts and lists are mainly used contacts.cpp where a list of received messages is generated

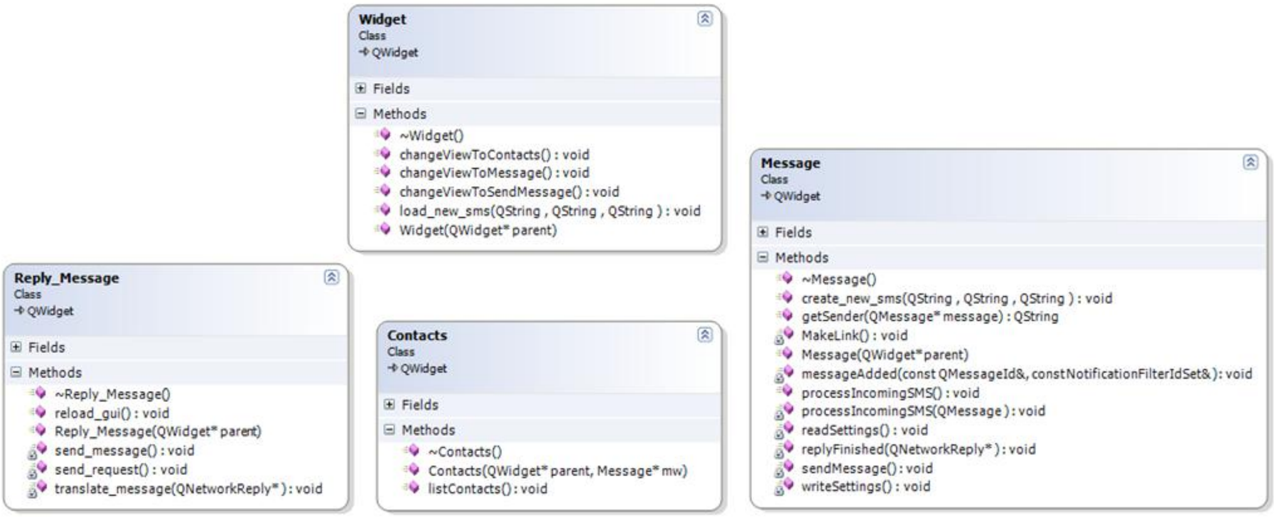
Used modules and configuration:

In Qute translate there is used many modules that Qt framework provides out of the box. Modules used are core, gui and network. Also mobility configuration is used a lot as the program is targeted for mobile phones. Mobility configuration uses messaging and contacts modules.

Involved technologies:

Qt 4.7 and Qt creator.

Class Diagram:


Widget
Class
→ QWidget
Fields
Methods
~Widget()
changeViewToContacts() : void
changeViewToMessage() : void
changeViewToSendMessage() : void
load_new_sms(QString , QString , QString) : void
Widget(QWidget* parent)
Reply_Message
Class
→ QWidget
Fields
Methods
~Reply_Message()
reload_gui() : void
Reply_Message(QWidget* parent)
send_message() : void
send_request() : void
translate_message(QNetworkReply*) : void
Contacts
Class
→ QWidget
Fields
Methods
~Contacts()
Contacts(QWidget* parent, Message* mw)
listContacts() : void
Message
Class
→ QWidget
Fields
Methods
~Message()
create_new_sms(QString , QString , QString) : void
getSender(QMessage* message) : QString
MakeLink() : void
Message(QWidget* parent)
messageAdded(const QMessageId&, const NotificationFilterIdSet&) : void
processIncomingSMS() : void
processIncomingSMS(QMessage) : void
readSettings() : void
replyFinished(QNetworkReply*) : void
sendMessage() : void
writeSettings() : void