```
TRANSLATIONS += translations/stopmotion_no_nb.ts \
                translations/stopmotion_no_nn.ts \
                translations/stopmotion_no_se.ts \
                translations/stopmotion_de.ts
```

Figure 1: Translation fields in a .pro file.

## 0.1 Internationalization

Internationalization is important for any software, perhaps even more so for an open source project. We aim at having users in a variety of countries in all parts of the world and internationalization and translation of the application is therefore very important. We can't expect everyone to read English, especially when our software is primarly aimed at students in primary and secondary schools.

Qt supports internationalization and provides some functions and tools which eases this work. All text in a Qt application which should be translated to various languages have to use QStrings and has to be processed with the `tr()` function.

Here is an example of how some user visible text (a tooltip using HTML formating), which needs to be translated are entered using the tr function:

```
infoText =
        tr("<h4>Remove Selection (Delete)</h4> "
        "<p>Click this button to <em>remove</em> "
        "the selected frames from the animation.</p>");
```

As mentioned above Qt has several tools to ease the translation process.

### Ts files

The *lupdate* program takes the stopmotion.pro file and the source-code as input. It then scans through the source code looking for instances of the tr() function. All of these instances are added to XML based files with the extension .ts (eg. stopmotion_no_nb.ts, stopmotion_gr.ts, ...). This file contains the English text, from the tr() functions as well as the translation in the language for that file. The translators can then use a tool called *Qt linguist* to translate the .ts files.

Before using *lupdate* on the .pro file however, some fields have to be added to this describing which translation files to make as shown in figure .

When some or all of the text have been translated one can create a .qm file using lupdate program, again with the stopmotion.pro file as input. The .qm files can then be loaded in the source code using the QTranslator class and used to translate strings.

## Po files

The system with the .ts files is all one should need for translating the application, but as we are working on an open source project we can't tell people to do things our way. If we want people to use their spare time for free to help us translate our application it has to happen on their terms.

The translators in the Skolelinux didn't like the .ts files as they were accustomed to a file type called .po. To recruit translators and to be able to upload our files to the Skolelinux i18n repository we needed .po files of the strings to be translated.

Qt used to use this format in older versions so we were able to find some tools called *findtr*, *msg2qm* and *mergetr*. findtr will scan through the inputed source files and produce a .po file. msg2qm produces .qm files from the .po files and mergetr will merge to .po files.

One problem we had with the findtr command was that it was less intelligent than the equivalent lupdate command. This meant that we had to change some parts of the source code in order to allow it to find the strings to translate. We also made some small scripts to ease the use of the findtr command to create po files for many languages. These scripts can be found in section **??** on page **??**.

To translate the .po files there are several tools out there. They are in ASCII format so one could use any text editor, but if one want a graphical environment for the translation one can for example use the KBabel program, which is very popular in the KDE world.

## Implementation

Blanchette and Summerfield points out in chapter 15 of their excellent book on Qt programming[**?**] that for most application it is enough to set the language at startup. We wanted the user to be able to switch language at runtime so we had to go through some extra effort to do this. When the user selects a new language we have to retranslate all the strings in the program so that Qt updates them.

The user can change language by going to the Settings->Languages menu. Stopmotion dynamically detects and loads all the translation .qm files which have been translated or partly translated and adds them to this menu. With Qt program it is customary to place the translation files in the catalogue /usr/share/<programname>/-translations and the translation files for Stopmotion is thus placed in the catalogue /usr/share/stopmotion/translations when one installs it from the Debian package.

Our module for creating language menus and loading translators (LanguageHandler) was another piece of code which proved to be very general and as such our sister Skolelinux project, the gnup project, was able to use the class directly in their AdminWorm application[**?**].

Stopmotion has currently only been translated to English and Norwegian no_NB, but now that we have .po files and now that the strings have stabilized we think we should get some speed in this process.