0.1 Packaging the program for distribution

Gunzipped tarball

This is the most simple format for a package. It only contains source code and files needed to build the program. Dependencies are often checked with a configure script which gives an error if unmet dependencies are found. The user is responsible for getting the libraries needed to build or run the program. In other words, the user needs to do everything by hand for getting it up and running. However, many peoples still prefer to do it this way, especially Slackware users.

In our case, this kind of package isn't difficult to build since Qmake already has added a section in the Makefile taking care of this. Running *make dist* will give us the gunzipped tarball, named *stopmotion.tar.gz*, which is ready to be shipped out to the people. To make it easier for the people to see which version the new package reflects, we repacks it with the name *stopmotion-x.y.z.tar.gz*, where *x*, *y* and *z* are the current version (e.g. 0.3pre.2). All this is done automatically with a script which will be further described below.

Debian package

One of the most fabulous things with Debian is the package management system. In contradiction to the simple tarball described above, everything is done automatically when installing a Debian package. Each and one of the dependencies to a package are downloaded and installed when installing the package itself. It can do a lot of more things too. However, to get such a complex package management system working well, packages have to follow strict rules; they have to be fully in line with the Debian policies.

When the 2.0 version of Stopmotion was finished in late of march, we decided to try getting this excellent software into Debian. The first step was to report a "Request For Package" (RFP) into the Debian Bug Tracking System (BTS). RFP means that someone has found and an interesting piece of software and would like someone else to maintain it for Debian. The bug report which was sent to the BTS can be read in appendix G on page **??**. We had to do this because no others than Debian developers can upload packages to the repository.

After two weeks nobody had responded to our RFP, but we could still see the light in the tunnel. We knew that it would be at least three official Debian developers at the upcoming gathering in Greece. The chance for catching one of them was therefore good. We succeeded with that. Andreas Schuldei, a German software developer, was willing to be a *sponsor* for the package. A sponsor is a Debian developer who acts as a mentor; they checks the package to see if it's packed correctly and uploads it to the Debian archive when they're satisfied with it. One of the group members therefore needed to be a maintainer for the package.

Building a Debian package is not a very complicated task if you have some experience with GNU/Linux and Unix programming. But, if you're a real novice, it's hard. An experienced Debian developer has stated the following [**?**]:

> One thing is certain, though: to properly create and maintain Debian packages you need man hours. Make no mistake, for our system to work the maintainers need to be both technically competent and diligent.

Since one of the project members already had builded few Stopmotion packages and became familiar with the most important Debian policies, it wasn't that hard getting the package correct. So, a few days after we had arrived Norway and was back in business, we had our first package into the Debian upload queue. After a while the package move on to the Debian repository[1] and now everybody who uses Debian can easily install our program by typing the following command in their consoles*:*

```
apt-get install stopmotion.
```

In other words: Our dream came true.

After the package had been built twice by hand, a bash script (see appendix **??** on page **??**) was created to automate the process as much as possible. The script creates a gunzipped tarball which contains all of the necessary files needed to build the application. The .pro file is also changed so compiler flags are switched from "debug mode" to "release mode", which means a more optimized executable. The tarball is then used to build the Debian package. When the building is finished and the output from lintian – a Debian package checker – is ok, everything is uploaded to one of the group members private Debian repository. Andreas is then given a hint about that a new package is available. He then gets the necessary files from the private repository and checks them. Everything is then signed with his GPG key and uploaded to the official Debian repository if the package is ok.

[1] http://packages.debian.org/stopmotion