Chapter 1

Requirements specification

For gathering and identifying requirements we choose to use usecases together with a short supplementary specification and a requirement risk list. This way of doing it were heavily inspired by the approach to collecting requirements in the RUP methodology framework as described by Craig Larman[**?**], most notably in chapter 6, 7 and 36.

The usecases were very good early on for identifying requirements, and the high level as well as the expanded written usecases were very good for getting started and for learning about the problem domain as we had hardly even heard of stop motion animations before this project started. Later when we knew more we didn't have to write out all the usecases anymore, but the usecases still drove our process and especially the requirement risk ranking list (see page 4 in this chapter) was very useful when planning new iterations.

1.1 Usecase Model

The usecases we have identified are summed up in a usecase-diagram shown in figure 1.1.

Written high level or expanded usecases for some of the identified usecases can be found in Appendix **??** on page **??**.

1.2 Supplementary specification

Functionality

- All buttons, except from trivial ones(ok, cancel, ...), shall have a tooltip and whatsthis text

Usability

- The application should be as simple as possible to use with most buttons visible to the user without having to access complex menues and submenues
- If a button can't be used at some point when using the program this button should be disabled. An example of this is that the copy button should be disabled if the animation is empty
- The application must be internationalized so that it can easily be ported to multiple languages

Reliability

- All errors and debug information should be logged through the same mechanism

Performance

- The application must be able to handle large quantities of pictures. At least ten thousand pictures should be possible

Supportability

- The application should be easy to port to new GUI-libraries and changing the GUI-library should not require changing many layers
- The application should work with all of the major GNU/Linux desktop environments / window managers (KDE, GNOME, Fluxbox, WindowMaker ...)

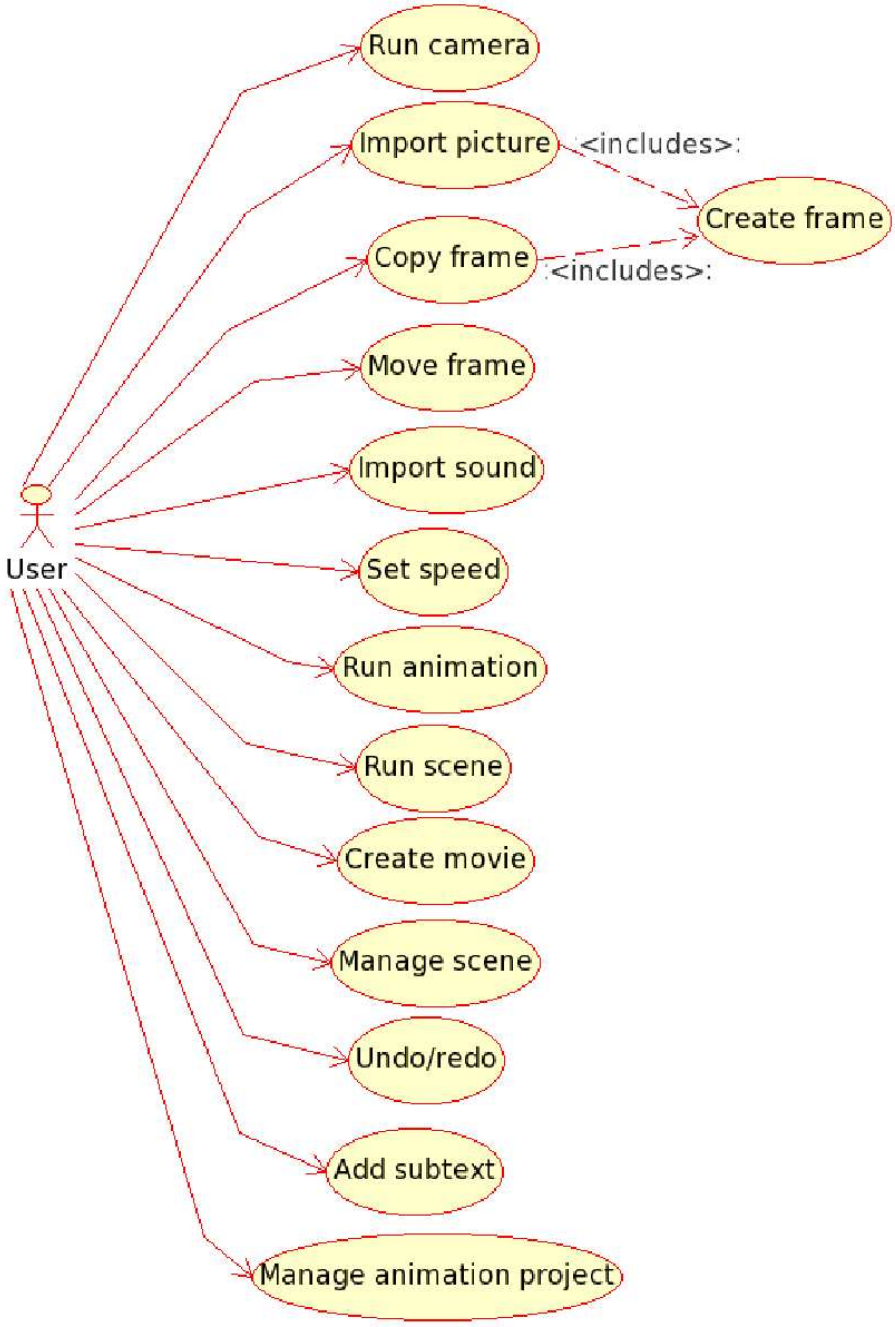

Figure 1.1: Usecase diagram

1.3 Requirement management

When deciding which feature to implement first it is valuable to rate the different features in relation to each other. This way one can better decide which ones are business critical or architectural significant and which ones are just eye-candy.

In table 1.1 the requirements for Stopmotion with values showing how heavily they influence the different areas of importance are listed. The requirements are ordered from most important to least important.

Table 1.2 shows the weights for each of the three areas the requirements are ranged by. As this is an open source project we felt it was sensible to give criticality or "early business value" the highest weight.

Requirement	Type	AS	Risk	Criticality	Sum
Import picture	UC	2	3	3	19
Create movie	UC	3	3	2	18
Manage animation project	UC	3	0	3	15
Internationalization	Feat	3	0	3	15
Create frame	UC	3	0	3	15
Run animation	UC	2	2	2	14
Setup camera	UC	1	2	2	12
Run scene	UC	2	2	1	11
Import sound	UC	2	2	1	11
Easily ported gui	Feat	3	2	0	10
Undo/redo	UC	1	1	2	10
Logging	Feat	3	0	1	9
Manage scene	UC	2	1	1	9
Copy frame	UC	0	1	2	8
Add subtext	UC	1	2	0	6
Set speed	UC	0	3	0	6

Table 1.1: The requirements with values ordered by importance

	Weight	Range
Architectrually significant	2	0-3
Technical risk/complexity	2	0-3
Criticality/Early biz. value	3	0-3

Table 1.2: The weights for the requirement factors