Release 0.3, 15.02.1995 by Thomas Engel (tomi@shinto.nbg.sub.org)

1 *The ClassEditor*

ClassEditorRunning.tiff ¬

This editor allows you to modify class definitions on a per-method basis. In additon to offering a method browser it will make it very simple to keep the documentation in sync with your code. Both elements are shown side by side.
The ideal situation would be to maintain a documentation file from the first moment since creating a new class or method. ClassEditor is able to create the docu temples for you in this release. It also helps you to stick to NeXTs layout by offering a style menu

In the worst case you should think about this project as a RFD (request for discussion). Let's start a news thread on what you expect from advanced development tools. Maybe someone at NeXT might find new ideas for NeXTSTEP 4.0 in there.

What it does

ClassEditor opens a set of *MyClass.m*, *MyClass.h* and *MyClass.rtf* files and displays them. You can edit and view the class on a per-method basis or see all three files at once inside the "cheat

window".

Modifying existing classes that are not spread accross more then those three files is quite safe. Adding methods is possible from inside the "cheat window" but introduces some rough edges. See the bugs section for details.

The Package

Version 0.3 comes in a package that includes:

- ClassEditor.app. A FAT binary for NeXT, Intel, HP and Sparc hardware.
- An *Examples* directory containing working test setups of interface, implementation and documentation files for retain classes.
- Online help...minimal
- Full sourcecode
- The documentation and an ASCII version (README) of this introduction

Attention: You need gnutar and gzip und unpack those files. Both programs come bundled with NeXTSTEP since version 3.2.

On where to find the latest releases please see the '*About this Project'* section.

Features

Currently there are only very simple things this application will do for you. Now this is just an early alpha version but...anyway...don't expect it to do magic things anytime soon.

- Select methods and view or edit the description and implementation.

- Create missing pieces of class and method documentation using automatic text generation and useful ready-to-copy templates.
- Use the "Style" menu and panel to get fast access to the right fonts for the nasty documentation work.
- Undo the last changes inside a method.
- Use Emacs keybindings to navigate inside the text areas.
- Select the "Plain C-Stuff" entry from the mode popup and view (or edit) all three files inside the "cheat window". Add defines, instance variables or new methods here.

Some features will definitly be improved to help me get along with all those missing class docus (yes...MiscSwapKit docus are already waiting for too long).

Bugs and Birds

It is not fair to speak about them in a little sub paragraph. There are so many of them that it does justify a whole chapter.

This app is a piece of brain stroming ± a running ToDo list...not a product. Peter L. would call it another piece of German software: nice idea but bound to never get finished.

Well maybe he is kind of right. Now lets face the most serious problems of this app.

Attention: If you have done some changes to the Implementation or Documentation section inside the "cheat window" you **must** save the changes and reparse the methods (cmd-u) if this wasn't done automatically.
Otherwise you won't see added methods !

- RTF headers and source will only work if the entire method names are typeset with only one font. Take a look at the ExampleClass to see how rich sources should look like to work

with ClassEditor.

- You must follow a very strict coding style. Otherwise ClassEditor won't be able to parse your code and might even corrupt your documentation. (Again: see the Examples for working layouts)
- Many user interface controls are just fakes ! The split view might cause your window to turn into an ugly piece of GUI when used to their limits.

For a complete description of known bugs see the *Release Notes* in the second chapter.

What it should

Of corse this should become the killer application of the nineties. But I am lazy and I do believe in code reuse. The guys at NeXT have already done all the nasty work.

So I only want to build a minimal tool to fill the gap until NeXTSTEP 4.0 arrives. Perhaps this app will only lead to a font service application. Who knows. (btw. too many people use the wrong font for source code inside the docus..it is Courier 12pt...not 14pt)

Future plans

My plans are to enjoy NeXTs new ProjectBuilder that will come with NeXTSTEP 4.0. **But only** if it comes close to the ClassEditors idea.
As far as I can tell from all the published material, I'm not sure that it will really cover all the fruits of desire.

So lets see what we are really missing today (some points only valid for this app!) :

- A combination of a class browser with the freedom to write ugly and nasty C-style code.

- Use colored text even with plain ASCII sources. ClassEditor might include an one-the-fly ASCII to RTF converter that would take care of using the right fonts.
- Online source sanity checks. Checking braces is the first way to go.
- Grouping methods and adding descriptions to instance variables.
- Being able to create classes from scratch. With all the nice things (protocols, categories, protected & private methods etc. pp).
- The special categories called (private) and (protected) might be used in the headerfiles and other sections by default. Every category should follow the *ClassName_categoryName.h* (*.m* or even *.rtf*) convention.
  The editor should create those files for you if you don't want to organize it in your private style.
  This would keep the code clean for fast and easy distribution.
- Support for CVS. Should be quite simple to add check-in/out stuff.
- Maintaining a revision history. This could really go hand in hand with the CVS support.
- Have default directories for header, source and documentation files. Like HeaderViewer knows where to search for all the stuff. This is implemented to a very basic level. The dirs are hardcoded right now.

And now lets see what we still might be missing in the future. Most of these ideas have to be supported by the whole development system to work properly:

- Tracking bugs with the code and documentation.
- System support for "See also:" references. Perhaps HeaderViewer could be smarter and create link buttons infront of all those entries ? Or...as Marco suggested...have the possibility to dump the docus as HTML.
- The editor should be able to dump templates for get/set'er methods and other ever returning tasks.
- Being sure that classes inside NIB are always up to date with the code. I hate to have to

remember all those dependencies.

- Speaking about IB; one of the nasty *features* is that it won't drop back to palettalized objects if it can't find the code for a certain subclass of them. In testmode this only works for subclasses of window...why ??

It is true that many things can be done right now...but you have do them by hand. There often is no real support from the tools. It is time to change that.

About this Project

I don't see the need to rush for a much better version of this app until it is clear what NeXT will include with their NeXTSTEP 4.0 release. There are other projects I want to spend more time with then this one.

The latest version of this editor will be available at the ftp.informatik.uni-muenchen.de, ftp.cs.orst.edu or ftp.nmr.embl-heidelberg.de anonymous ftp servers. Soon there will be a WWW page were you can find out more about the status of this...and all the other projects I'm working on.
It will be under the projects section of:

http://wwwcip.informatik.uni-erlangen.de/user/tsengel/

Compiling

This version includes all the source code necessary to recompile the program from scratch. It does **not** come with all the libraries! You will need to get the MiscKit project (Version 1.4.0 or higher) from the archives.

For more details on the MiscKit collection see the common NeXT ftp servers or get the

Objective-C or NeXT FAQs. If you would like to submit something to this collection you should contact: Don Yacktman <don@darth.byu.edu>

In Case of Trouble

If you have any questions you can contact me.

Thomas Engel
Netpunstr. 9
D-90522 Oberasbach
Germany

E-mail: tsengel@cip.informatik.uni-erlangen.de
tomi@shinto.nbg.sub.org

(NeXTMail welcome)

Warranty and copyright

Copyleft

Source code that is not part of the MiscKit project - and therefore underlies the MiscKit distribution and copyright rules - is distributed under the GNUpublic license.

You are free to extend and modify this application. But don't redistribute a modified version under the same name unless I gave you the permission.
I don't want to have different, confusingly incompatible versions running around the world.

Anyway...comments are **highly** appreciated. Take this app as a request for discussion.

No Warranty

This software is provided '*as is*' and the programmer is not responsible for any harm this program may cause.
You - the user - are responsible for everything that may happen to your business, hardware, software, car, CD collection or what ever may be worth your attention or money.
Using this product is at your own risk and your private fun.

There should be no serious bugs inside but remember that a carbon-based unit did the coding.

TravelWare

Like all my projects this app is free and should be considered as travel-ware.

166998_12ptHevBlk.tiff ¬ Let it travel to as many people you know.

02_12ptHevBlk.tiff ¬ Send me a postcard or E-mail if you use it. I will try to keep you informed about new releases.

03_12ptHevBlk.tiff ¬ If you have a free bed or some free space on your floor¼ give me a hint. I might come and visit you on my trip to the US (locations on Hawaii, near Seattle or WhistlerMnt. (Canada) preferred)

Enjoy it.

.Unterschrift.tiff ¬