EMACS Authoring Tool for Hyperties
Documentation and comments

General philosophy:

The authoring tool is a text editor of storyboards (to create, edit articles) as well as a previewer of Hyperties (you can follow links by clicking on ~strings~ and .target). It also calls associated tools like graphic editors and target editor.

Emacs creates one buffer per article (= per storyboard). Each new buffer appears in a new window overlapping the existing windows. The tab of the window displays the title of the article. Clicking on a tab brings the window on the top.

A lot of basic manipulation are provided directly by Emacs (window closing, stretching, bringing to the top, etc..). Clicking the right button on the tab brings menus.

Other functions are loaded with hyperties.ml. Those functions are specific for Hyperties (following a link, creating a new article, resolving references, etc...).

Today the functions are accessible by Esc-x + the function name. Later on, menus will be created to access all the commands (using the Emacs menus which are defined as xinfo nodes).

To run it:

nemacs
    Start the NeWS Emacs (the one with the tabs!)
    will also run from a standard terminal
    but of course without the tabs

Esc-x load /tumtum/don/emacs/hyperties.ml
    Load the lisp functions library written by Don.

Esc-x author <db directory>
e.g. Esc-x author /bensun/database/telescope
    Read the masterindex
    locate the !home article, open a new buffer with it.

The author is now ready to be used...

The way I think it works today:

There are two windows on the same buffer: the little one shows the definition, the large one the content.

The ".content" or ".definition" doesn't appear in the window (but remember the buffer still contains the whole storyboard. If you really wants to see the whole file use: Esc-x widdenregion.

In Emacs use Ctrl-n for next window, Ctrl-p for previous window, to change window on a same buffer (same storyboard).

When you click on a link (= a string surrounded by tildes), a new window appears with the in it the storyboard of the destination article.

A few commands exists like:
    ht-add-synonym
    create storyboard
    ...???

The only sure way to know what are the current available functions is to visit the file /tumtum/don/emacs/hyperties.ml. Emacs loads a ".ml" mode which allows you to get help on each function by clicking on the function with the middle button.

cps- May 8, 1989

Comments and suggestions :

I am not sure that the multiplication of the windows is a good idea. It become very difficult to recognize where an article starts and stops, and too much space is lost with the window bottom bars.

I would prefer to only have the content in the window, see the title in the tag (as now) and click somewhere to get the other fields. The other fields (definition, synonyms, group, notes,etc.) could appear in a transient window only when called. (RQ: A transient window appears as an overlapping window behaving like a Unix "more", and Ctrl-E start an edit mode on that window.)

Nevertheless it might be good to have a function which does "widenregion" for all storyboards (In case someone wants to see the complete storyboard all the time).

The masterindex should also be active. If you click on a title (or file), nyou get the storyboard (may be is it already true, I have to check.)

Possible functions:

see-definition
- show the definition in a transient window
(I think keeping the .definition is not problem, may be even better)

see-title

see-synonym

see-note

add-article <article title>
- check in index that it doesn't already exist
- create new buffer and go there
- create a file name for it
- insert article in the masterindex

add-synonym
- check in index that it doesn't already exist
- add a synonym in the storyboard
- insert it also in the masterindex

refered-by
- find all the articles which refer to the current article
-the list appears in a transient window

check-links

If called from a standard storyboard:
- find all the links and verify that the corresponding article exist (it can just check that the title exists in the master index, but it would be a lot better to check that the article actually exists: that it has a content or at least a definition).
- check that the picture and target names are valid
- When an orphan is found: stop with cursor on the orphan link.

If called from the masterindex:
- check all files (as described before)
- build a file called "checklink.aut" with all the encountered problems:
orphan link - article where first encountered
- build a crossreference table "crossreference.aut":
article title
"link to" list
"link from" list
- check that the pictures and targets exist

delete-article
- remove the window
- delete the file
- remove the title and synonyms from the masterindex

(If "refered-by" is fast enough, it would be good to call it before starting the deletion, to verify that no article was refering to the article that is to be deleted!)

Little problems:

Missing:
.title .TITLE
(should be case unsensitive)

So far the available menus are defined in Don "private" Emacs xinfo database. It should be made public!

Authoring the graphic

So far Hyperties accepts Sun raster, scalable raster and postscript files.
Emacs call the editors (or just display the picture) from the storyboard.

The hardest part is to define the area of the targets. A target-tool exists to create targets of various shapes (circles, rectangles, free shapes) but is still very clumsy to use.

This target tool will be made accessible from the Emacs author tool. The same way a click on a ~selectable string~ brings the corresponding storyboard in a new Emacs buffer, a click on a picture name will bring the picture loaded in the graphic editor, and a click on a target name will bring the target loaded in the target tool.

In some case Emacs may have a hard time finding exactly what you are clicking on (especially with macros...). Emacs has to know about the syntax of our Hyperties markup language. But the possible mistakes should be acceptable for the author who put that complexity himself!

Example:

.picture whale

A click on "whale":
loads the corresponding .pn0 in a buffer
Emacs read the type of the picture
If it is a raster: load the graphic editor
if it is a scalable picture: you should be able to strech it
If it is a postcript file: it appears in a window (with a refresh button to see the result of your editing).

If no .pn0 exists, Emacs prompts you for a type (raster, scaled raster or postscript) and for a file name and directory. The .pn0 is created, the name inserted in the master index. Then the graphic editor is called. Of course the user might only want to view the picture and quit right away.

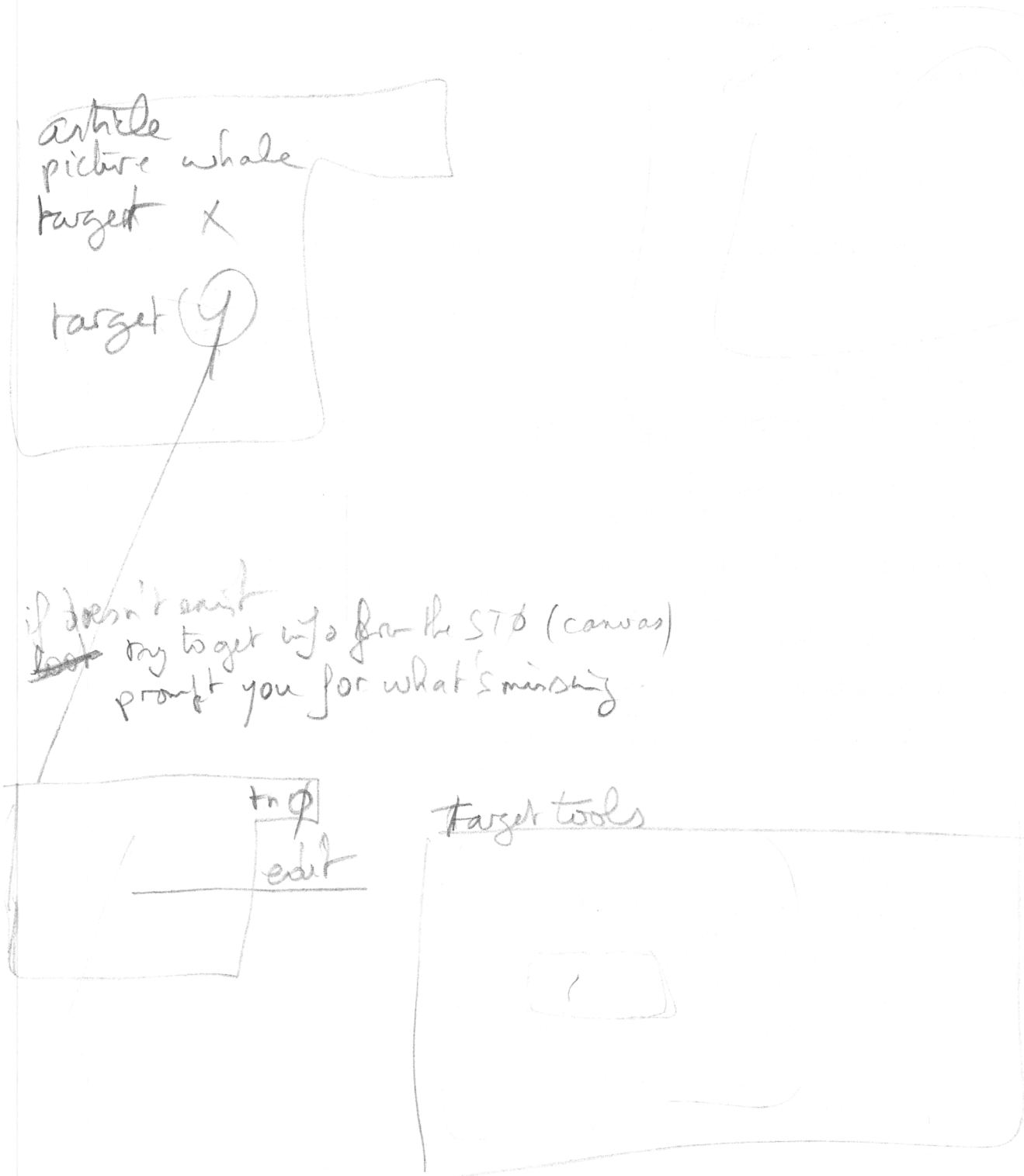
.target <tail of the whale><The article about the tail>

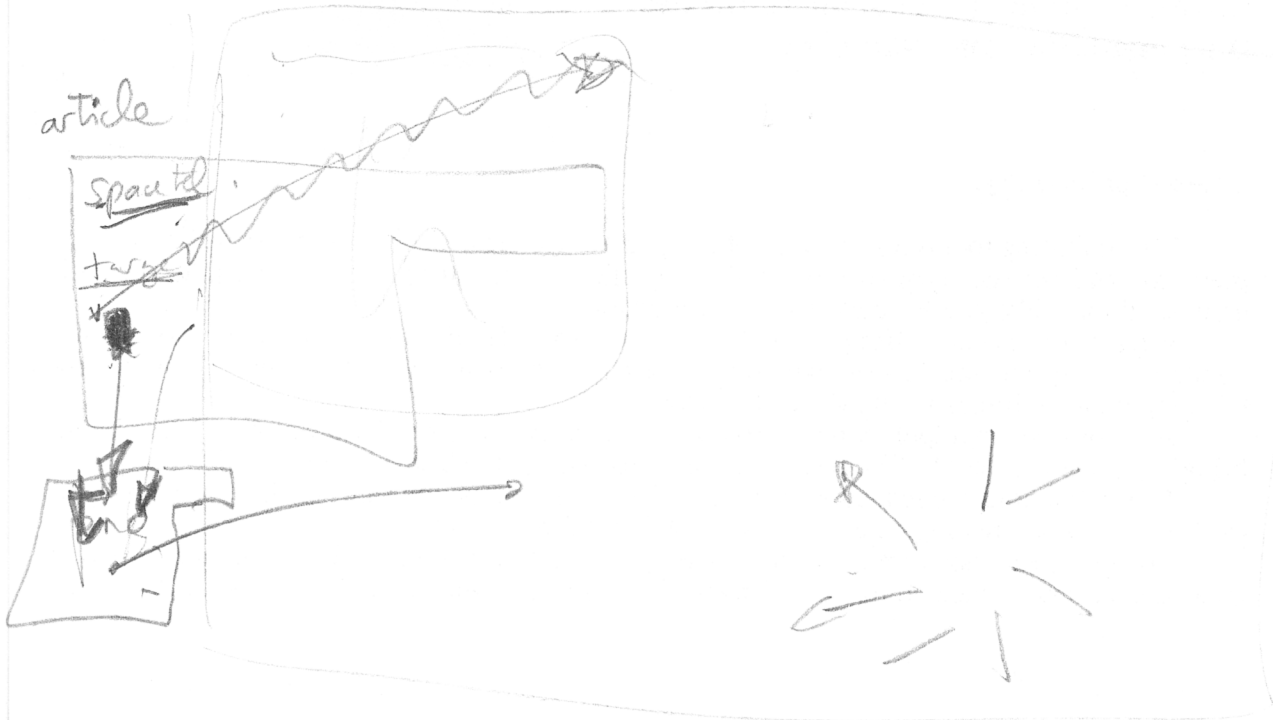
A click on "the article about the tail"
The storyboard is loaded in a new buffer (yes it is a link!)

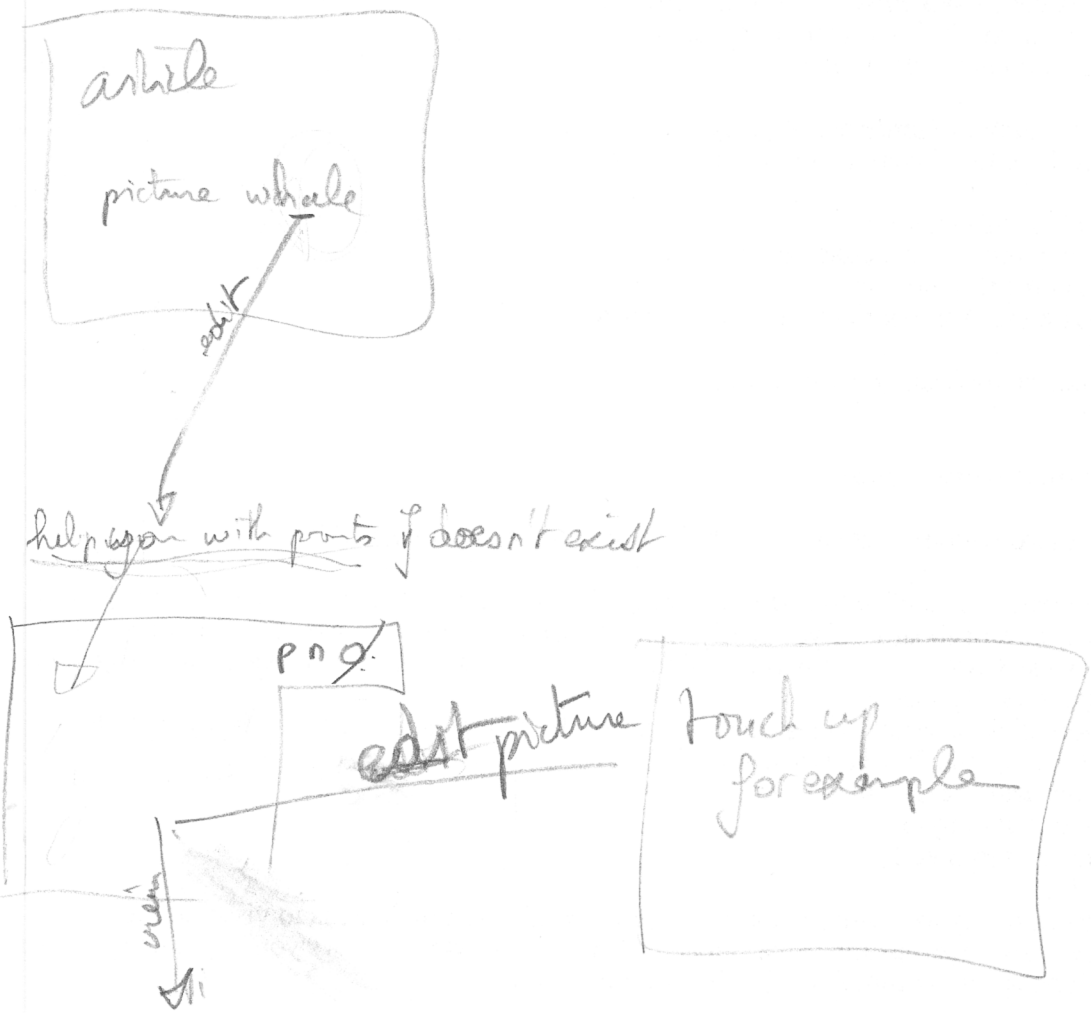
A click on "tail of the whale"
The .tn0 is loaded in a new Emacs buffer

Emacs backtracks to get the corresponding picture (here the whale)
The target tool is loaded with the whale as canvas, and the "tail of whale" target shape and attributes are shown.

If the target doesn't exist yet, the target tool is simply started.
If not picture can be found: a message tell you to first create the picture.


article
picture whale
target X
target
if doesn't exist
try to get info from the STD (canvas)
prompt you for what's missing
target
edit
target tools


article
Spawn
target
End


article
picture whale
edit
help pop-up with prints if doesn't exist
PNG
edit picture
touch up
for example