1 Collection of ideas

These are ideas to improve the program and add features. They are not sorted, but rather written down as they popped up in my head.

1.1 Automatic error correction

I have some mistakes i do alot, like forgetting to load a package or a tikz library or something. This is nearly all the time easy to fix and, more important, easy to detect automatically. I see two ways for automating this task, both should be used together: 1. check the difference in edits BEFORE compiling and add packages and libraries for inserted code. 2. parse error messages, insert missing code and recompile without showing the error to the user. There should be a message like "added package 'foo' because of line 13: '...'".

1.2 add a 'make minimal document to try something out' feature

For this, i would like to be able to provide templates, e.g. a tikzpicture. Later on, when i'm finished with the test, i would like to insert the written code (without preamble and stuff) into a content file (or make a new one)

`minimal tikz` opens a file with the contents of the preamble and a tikzpicture. try out, compile, edit, compile, insert into main project, delete test files

1.3 add options to commands, like:

`add -aux` calls aux, or calls a specific handler for aux files or something similar

1.4 add packages to the preamble without direct editing

`package add kantlipsum`

`package remove kantlipsum`

`package add -option package` (maybe different syntax)?

1.5 find a way to keep the current line number for every file, for quicker editing

First approach: use a diff after every edit to figure out the current line number (maybe only approximately)

1.6 add a command for ”adopting” projects

There could be several levels of adoption, like minimal (keep the structure but make it possible to use the command of the program) or full (restructure everything. This includes breaking up files into (possibly) smaller files. The people i know tend to write everything in one file (maybe seperating the preamble), this would be changed by this command).

1.7 add interactive ’wizards’ for project generation, loading, adopting etc.

Idea here is to ask the user if no option is given, or to explicitly use this by saying so.

For example, while adopting a project, the user could be asked for the filenames to generate, for boundaries where to break up the files, the level of adopting etc.

1.8 test for stability

Currently, a proper run of latex without errors should work, but i have run nearly no tests for failures.

We don’t want to see referencing of nil objects in user space. Check results!

1.9 further enhance parsing of the output.

Currently i’m using the output of the latex call, but better and more information is found in the log file. Use that as source.

Also, improve support for parsing of packages, latex messages and tex messages. Currently, no package is supported and i guess one error message

1.10 make it possible to react to errors and warnings

Figure out a way to find the source of the message (hard?) and open the corresponding file at that line

1.11 add option to temporarily exclude a content file from the project.

The file should just be ”commented out”, not deleted

This could be used together with a ”compile test recent changes” or something, which only tests to new edits

1.12 improve cleanup

Currently, this REMOVES all unwanted files, but these files mostly are needed files generated by latex. We just don't want to see them, so move them into a directory named by the user (default: something like .build)

1.13 argument parsing with options

There are a number of situations we need to solve:

- an option generates a number of arguments (-foo -> foo bar)
- an option causes a function to get called again, depending on the return value
- an option chooses the function to be called

Furthermore, it should be possible to specify the number of further arguments an option gobbles. We should be able to set a minimum, an exact or a maximum value.

2 Program flowchart