MeVisLab Community Coding Style Guidelines

General rules

- **Write everything in English** (comments, names, variables, documentation...).
- **Be consistent.** Even if you decide not to follow some of the rules, do it at least consistently. Never produce code with different styles of programming.
- **Insert the license and copyright at the top of files.**

Naming conventions

Definitions

- **UpperCamelCase:** Upper case at beginning and new words
- **lowerCamelCase:** Lower case at beginning, upper case at new words
- **ALL_UPPER_CASE:** All upper case, words separated by underscore ("_")

Code

- **Class names:** UpperCamelCase
- **Typedefs:** UpperCamelCase
- **Defines:** ALL_UPPER_CASE
- **Enum values:** UpperCamelCase or ALL_UPPER_CASE
- **Methods, functions, variables:** lowerCamelCase
- **Private and protected member variables** are preceded by an underscore ("_")
- **"Guard defines" for header (*.h) files:** Starts with double underscore ("__"), followed by name of the file without ".h", ends with "_H".

Files

- **Do not use whitespace or other uncommon characters in file or directory names.**
- **Use identical upper/lower case spelling in header and source files.**
- **Use "/" instead of "\" as path separator in include statements.**
- **Use "ml" prefix for everything that links to the ML.**

MeVisLab/ML Identifiers

- **Field names:** lowerCamelCase with "Fld" at the end.

General Programming

- **Use spaces instead of tab characters for indentation.** The recommendation is two space characters for each level of indentation.
- **Comment your code using Doxygen-style formatting.**
- **Provide appropriate error handling.**
- **Use speaking and consistent variable names and other identifiers.**
- **Enclose code blocks in braces after all flow control statements,** even in one-line constructs.

C++

- **Never leave pointers uninitialized.**
- **Never allow dangling pointers.** If objects are destroyed or invalidated, all pointers to it must be cleared.
- **Use `NULL` for null pointers**, not the numerical value `0`.
- **Always initialize variables at declaration.**
- **Provide a virtual destructor if a class contains virtual methods.**
- **Define typedef and enumerated types in an appropriate class scope** to avoid name space conflicts/pollution.
- **Keep the scope of objects as small as possible.**
- **Use "guard defines" in header files** to avoid duplicate inclusion.
- **Initialize all member variables in the constructor.**