Code guidelines for the `exbisim` source code

for internal usage only

Jelle Hellings

June 15, 2011

1 Introduction

These code guidelines do not describe how every aspect of the code should be formatted. This document only describes naming conventions; commenting conventions and some other conventions used in this project. On all other aspects we expect the usage of common sense and the goal should always be to write readable, portable and clear code.

This document is further expanded when more conventions or guidelines are introduced within the project.

2 File conventions

All header files have extension `.hpp`, all source code files have extension `.cpp`. Header files should be named to the principle component placed in the header file. Every function, type and member should be declared in a header file. Every function, type and member should be associated with a clear doxygen-like description in the comments. Implementations are placed in an accompanying source code file or implementation header file (templates, inline functions).

The suffix `_impl` is used for header files containing implementations of parts needed during compile time (templates, inline functions). These implementation header files are included by the declaring header files. The suffix `_pd` is used for platform dependent declarations (header files) or implementations (source files).

Header files containing declarations should have inclusion guards of the form `INCLUDE_PATH_TO_FILE_FILE_NAME_HPP_`. All generic library components are placed in namespaces reflected by the path of the file.

3 Naming conventions

All types, functions, methods and variables have comprehensive descriptive names. We do use short non-descriptive variable names for iterators (for example `i`, `j`, `it` and so forth).

1. All user defined types are in `CamelCaseWithLeadingUpperCase`.
2. All local variables are in `camelCaseWithLeadinglowerCase`.
3. Class members (data members and methods) are named in `lower_cases_with_underscores`.
4. Functions are named in `lower_cases_with_underscores`.
5. Macro constants and enumeration constants are in `UPPER_CASE_WITH_UNDERSCORES`.
6. Namespaces are short descriptive names in `lowercase`.

4 Other conventions

The following conventions are also used:

1. We use 4-spaces as the tab-indentation.
2. Indentation of multi-line declarations or expressions should lead to pretty formatted (aligned) or with an indentation of at least 8-spaces.
3. Every source line with comments has maximum length of 80 characters. Non-comment source lines are limited to a maximum length of 100 characters (whenever reasonable possible).
4. Functions should be short and simple enough such that in-function comments should be unnecessary.
5. For every function and type every parameter (template and function argument) is described. For every function with clear exception cases these cases are described.
6. All source code is ordered alphabetically. Ordering is types first; operators secondary; functions and methods last. Within classes and structs public members appear before protected and private members.
7. Each program has a `main` function declaring command line parser options; which are used by an entrypoint library. Each program has an `entrypoint` function called by the library. This function performs the actual tasks of the program.