DMTPC C++ Coding Guidelines

March 25, 2013

1 Introduction

Until now, the DMTPC collaboration has had no formal coding conventions, which has at times resulted in chaotic code. This document seeks to fill that void by creating a set of conventions. As we work on restructuring our codebase, please try to keep this document in mind.

2 File structure

2.1 Header files

Our header files end with `.hh`. They should go in a separate include directory for most packages (except perhaps the most simple ones) to avoid clutter. Each header file should have a `#define` guard to prevent inclusion multiple times.

2.2 Source files

Generally source files should go in a separate src directory to avoid clutter.

3 Naming

3.1 Namespaces

Namespaces are lowercase. All code is to be in the `dmtpc::` namespace and and the subspace related to the directory structure (e.g. things in the DmtpcImage directory will be in the `dmtpc::image::` namespace.)

3.2 Classes

Class names should be in camel case starting with a capital letter. For example `DmtpcSkimDataset`.

3.3 Variables

Instance variables may have any convention that is convenient for the class in question. In some cases, a leading underscore may be useful for more complicated classes. No conventions are imposed on local variables, but it is useful to use descriptive names.

3.4 Functions and Methods

Functions and methods should be in camel case starting with a lowercase letter. For example `toPolarCoordinates`. Parameters to functions and methods should be in all lower case with underscores if needed for clarity.

3.5 Enums

Enum names and values should be in all caps. Underscores may be used to separate words.

3.6 Macros

Macros should be in all caps. Underscores may be used to separate words.

3.7 Constants

Constants should look something like `kPiOverTwo`. `???`

4 Code Formatting

4.1 Indentation

Spaces (NOT tabs) should be used for indentation. 2 spaces should be used in most cases.

4.2 Line length

Each line should fit in a fullscreen terminal at moderate resolution on a monitor 1280 pixels wide with reasonable resolution. This works out to a maximum length of 179 characters or so.

4.3 Braces

Please use a consistent brace style for each section.

5 Documentation

Every public method should have a doxygen style comment. Each class or namespace should have a short doxygen style comment describing its purpose in life. Code that does anything complicated may benefit from some explanatory comments.

6 Do's and Don'ts

6.1 Do

- Use inline methods for simple getters and setters.
- Use `const` whenever possible.
- Use the most general subclass whenever possible (e.g. `TH2` instead of `TH2F`).
- Try to minimize the use of printouts (except during debugging, of course).
- Make an attempt to spell everything correctly. Probably we slightly prefer American English?

6.2 Don't

- Put things, ESPECIALLY enums, into global namespace.
- Use operator overloading.
- Leak memory.
- Randomly Draw() things to canvases.
- Use asserts unless there's a very good reason (or they're only enabled conditionally for debugging).
- Use exceptions in your code.
- Use multiple inheritance unless you are forced to by external code.