Coding Conventions

This page is **stale** and needs to be reviewed. It may be deleted or radically changed in the near future.

Overview

- Due to this being a collaborative software project among new developers, it's important to make sure that the code you write is clean and usable by others. We predominantly use two languages in this project, C++ and Python. Before contributing to the project make sure you understand the language you are working with completely so as to not make glaring mistakes that could be overlooked by the team.
- Below are a list of coding conventions for each respective language in the project. There may be exceptions to these rules, but otherwise they should be followed.

Programming Practices

- Before writing ANY code, know what you want to accomplish. Whether it's fixing a bug, adding a new feature, or cleaning stuff up, have a mental map to avoid spaghetti code and wasting your own time.
- Know the system you are working on BEFORE you touch it. Some portions of the software on the sub are extremely complex and require knowledge of the whole system before you can effectively contribute to it. Ask around for help if you are editing a portion of the codebase you have not touched before.
- Plan with others before you make major changes to something. You may be editing code that someone else had branched awhile ago. Try to isolate your development from everyone else to reduce merges and implementation conflicts.
- For software design refer to this wikipedia article. Software Design

Naming Style

File / Class Names - Words in file / class names should start with an uppercase, be concise, and should not conflict with any other classes.

Target Names - Target names should be lowercase, concise, and should not conflict with any other targets.

Class Members / Functions - Class members and functions should also start with uppercase letters unless you are calling derived members of the same name in which case you can use the lowercase in order to not override the function.

Constants / Macros - Constants an macros should be all caps, concise, and should not conflict with

other macros.

Style

- 1TBS should be followed as closely as possible for all C++ code. Some exceptions may occur when you have functions with lots of parameters or a call requiring multiple lines to accomplish. When it doubt, just make it look CLEAN.

Optimizations

- For most contributions to the codebase, DO NOT WORRY about optimizing your code. Obviously utilize algorithms and logic that is efficient for your use case, but do not write cryptic code for the sake of speed. Also, DO NOT use 'compiler tricks' either. Compilers nowadays are very good at creating fast code. Sacrificing readability for that 0.01% increase in speed is not worth it.

Commenting

- COMMENT YOUR CODE. If you are writing a class or a function give a simple, one line explanation of what it does and how it should be used. For algorithms, explain major steps that cannot be easily deduced just by looking at the code. If you write a line of code that is a bit 'hacky', that needs to be changed in the future, or has possible bugs in some circumstances, **Label It** with a comment explaining the problems.