Programming of Supercomputers Lab

Coding Guidelines

(1) Indentation should be done with 4 spaces per level and never with tabs. You can set your editor to emit spaces when you hit the tab key.
(2) Whitespace should be used liberally within expressions, but normally only one space at a time. For example, there should be a space before and after all operators and after every comma.
(3) Non-ASCII characters should be rare and must use UTF-8 formatting.
(4) Each line of text in your code should be at most 100 characters long. Be consistent in how you break up the longer lines.
(5) Use trailing braces everywhere (if, else, functions, structures, typedefs, class definitions, etc.). The else statement starts on the same line as the last closing brace.

```
if (x) {          if (x) {

}                 } else {

                  }
```

(6) Function names should be descriptive and should normally be a "command" verb, or a verb followed by noun. Avoid starting function names with nouns.
(7) Function names start with a lower case:

```
void function( int arg1, double r );
```

(8) In multi-word function names all words should be lower case, with underscores (_) between words:

```
void this_function_does_something(void);
```

(9) Write a function call on one line if it fits; otherwise, wrap arguments at the parenthesis.

```
bool retval = do_something(arg1, arg2, arg3);
bool retval = do_something(arg1,
                           long_argument_2,
                           arg3,
                           arg4);
```

(10) Always indent the following:
   a. Statements within function body
   b. Statements within blocks
   c. Statements within 'case' body of switch operators

```
int function(int bar) {
        switch (bar) {
        case 0:
                ++bar;
                break;
        case 1:
                // …
        default:
                bar += bar;
                break;
    }
}
```

(11) Within functions single empty lines should be used to add clarity, etc. A function or method should never have more than one consecutive blank line within it.

(12) Write small functions, no longer than 50 lines of code (around 1 page).

(13) Factor out common code into small functions but don't create too many tiny functions. Each function has to be a coherent unit of work.

(14) Use a standard header for all functions that is Doxygen compliant:

```
/**
* Calculate the distribution factor of a dataset.
*
* @param nintci start element
* @param cgup data array for which a factor should be calculated
*
* @return resulting distribution factor
*/
double calc_distribution_factor(int nintci, double* cgup)
```

(15) Variable names start with a lower case character.

```
float x;
```

(16) In multi-word variable names all words should be lower case, with underscores (_) between words:

```
double max_distribution_factor;
```

(17) Typedef, Struct and Enum names use the same naming convention as variables, however, they always end with "_t".

```
typedef int fileHandle_t;
struct renderEntity_t;
```

(18) Defined names and enum constants use all upper case characters with an underscore as a separator:

```
#define SIDE_FRONT 0
enum CONTACT_T {
        CONTACT_NONE,
        CONTACT_TRMVERTEX
};
```

(19) Use 'const' as much as possible. Use:

```
const int *p; // pointer to const int
int * const p; // const pointer to int
const int * const p; // const pointer to const int
```

However, don't use:

```
int const *p; // pointer to const int
```

Note: 'const' applies to whatever is on its immediate left (other than if there is nothing there in which case it applies to whatever is its immediate right)

(20) Avoid having more than 5 nesting levels. If your code needs more nesting, consider creating a function for some of the nested code.

(21) Comments should be brief and should explain WHY instead of HOW, i.e., don't restate code in the comments but give reasons why a particular algorithm was chosen, or why a particular data structure is used, or why a certain action must be taken.

References:

Google C++ Style Guide, http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml
id Software Coding Style, ftp://ftp.idsoftware.com/idstuff/doom3/source/CodeStyleConventions.doc