# Contributing to Goxel

Please read this file first before making a pull request to Goxel.

## CLA

In your first pull request, you must add your name in a CLA file in

'doc/cla/individual'. This gives me the right to release special version of

Goxel under a commercial licence. It doesn't change the license of the current

code. See:

https://github.com/guillaumechereau/goxel/blob/master/doc/cla/sign-cla.md

For more information about that.

## Coding style

Try to follow the code style I used for Goxel, as I am unlikely to merge a pull

request that doesn't. The coding style is almost the one used by the linux

kernel, but using four spaces for indentation (I also accept typedef). When in

doubt, just look at other part of the code. The most important rules are:

- Use 4 spaces indentations. No tabs characters anywhere in the code.

- 80 columns max line width.

- No trailing white space.

- function and variable names all in lowercase, with underscore to separate

parts if needed:

int nb\_block; // Good

int nbBlock; // Bad

- K&R style braces (opening brace on same line):

if (something) {

...

} else {

...

}

- Except for functions, where we put the opening brace on the next line:

int my\_func(void)

{

...

return 0;

}

- I also accept exceptions to the K&R braces for multi line conditions

(but that should be avoided if possible by making the condition shorter):

if (a\_very\_long\_condition ||

that\_uses\_several\_lines)

{

...

}

- No space between function and argument parenthesis:

func(10, 20) // Good

func (10, 20) // BAD!

- One space after keywords, except `sizeof`:

if (something) // Good

if(something) // BAD

- One space around binary operators, no space after unary operators and

before postfix operators.

x = 10 + 20 \* 3; // Good

x = 10+20\*3; // BAD

x++; // Good

x ++; // BAD

y = &x; // Good

y = & x; // VERY BAD

- Use 'C' style variable declarations:

int \*x; // Good

int\* x; // BAD

- Put all the variable declarations on top of the function, so that we can

see them all at the same place.

## Git commit style

- Keep the summary line under about 50 characters.

- The rest of the commit message separated by a blank line. Wrap lines at

72 characters.

- Try to separate the commits into small logical parts. For example if you

need to add a new public function in order to fix a bug, the first commit

should be about adding the function, and the second one about fixing the

bug.