# Code Style

This file describes Gosu's code style.

If you are looking for instructions on how to customise and compile Gosu, see the [Hacking Gosu wiki page](https://github.com/gosu/gosu/wiki/Hacking-Gosu).

## rake format

Please run `rake format` before committing.

This is a Ruby script that checks all C++ headers and implementation files for deviations from Gosu's code style.

## Consistency Over All

Gosu does not an automatic formatter like clang-format because formatting code for legibility is highly contextual.

Instead, please ensure that code is consistent with its surroundings.

\*\*Example:\*\* When there are three `if` statements that fit on a single line, and you add a fourth `if` statement that requires a longer block, then all four `if` statements should be expanded to the multi-line form.

\*\*Rationale:\*\* A consistent (boring) code structure makes it easier to skim over code.

## Whitespace and Line Lengths

Do not use lines longer than 100 characters, except in Markdown files, where each line should be one sentence.

\*\*Rationale:\*\* Shorter lines introduce distracting line breaks.

Longer lines require horizontal scrolling in editors (depending on screen size) and on GitHub.

Trailing whitespace is allowed (but not required) in empty code lines when it matches the indentation of the surrounding code.

Please do not submit patches that only change files by removing whitespace.

\*\*Example:\*\*

```ruby

def this\_is\_fine

··puts "before empty line"

··

··puts "after empty line"

end

```

\*\*Rationale:\*\* This is the default indentation behaviour in TextMate and Xcode.

Worring about trailing whitespace is not worth it.

## C++ and Objective-C

\* Use C++11, not C++14 or later.

(What counts is whether your pull request compiles on Gosu's CI.)

\* Indent with four spaces, use C++ comments (`//`), use `int\* p` instead of `int \*p`, use `const int` instead of `int const`.

\*\*Rationale:\*\* Gosu has always followed these rules, and they match the [C++ Core Guidelines](https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md).

\* Use `CamelCase` for class names, `snake\_case` for variables and functions, and `SCREAMING\_SNAKE\_CASE` for constants (`const`, `#define`, `enum` values).

\*\*Rationale:\*\* This convention is dominant in the Ruby world, following it even in C++ keeps identifiers consistent in both languages.

\* Use C enums with a short prefix for enumerations.

\*\*Example:\*\* `enum SomeEnumeration { SE\_FIRST, SE\_SECOND }`

\*\*Rationale:\*\* This is much shorter than `SOME\_ENUMERATION\_FIRST` or `SomeEnumeration::FIRST`.

\* The opening braces `{` for classes, structs, enums, namespaces and functions should be on its own line.

The opening braces for `if`/`for`/`while` statements should never be on its own line.

\*\*Rationale:\*\* Compromise between brevity and readability.

\* Keep short `if` statements on a single line (usually: early exit).

\*\*Example:\*\* `if (argument == nullptr) return;`

\* Use a space between control flow keywords (and `assert`) and the argument.

\*\*Example:\*\* `return 5`, `if (condition)`, `switch (x)`, `assert (this\_is\_true)`.

\*\*Rationale:\*\* This makes it easier to distinguish control flow from function calls.

### #include order

Please try to follow this order for header files:

1. In implementation files, start with the header that corresponds to the source file, e.g. `#include <Window.hpp>` in Window.cpp.

2. In header files, `#include <Gosu/Fwd.hpp>` always goes first when it is needed.

2. Local headers in alphabetical order, e.g. `#include "WindowHelpers.hpp"`.

3. Gosu headers in alphabetical order, e.g. `#include <Gosu/Utility.hpp>`.

4. C standard library headers in alphabetical order, e.g. `#include <cstdint>`.

5. C++ standard library headers in alphabetical order, e.g. `#include <algorithm>`.

6. Platform-specific headers in alphabetical order, e.g. `#include <windows.h>`.

\*\*Rationale:\*\* The compiler will complain when header files rely on system headers being included first.

Also, when using `#import` to include Objective-C header files, please use `#import` for all headers instead of `#include`.

\*\*Rationale:\*\* Consistency.

## Ruby

\* Use simple (dumb, boring) Ruby code where possible.

\*\*Rationale:\*\* Ruby/Gosu is often used by novices.

\* Use `"` for strings, only use `'` when absolutely necessary.

\*\*Rationale:\*\* Consistency.

\* Do not replace all uses of `if not` by `unless`.

There is a time and place for both.

\*\*Rationale:\*\* The English word "unless" is [used to describe exceptional conditions](http://dictionary.cambridge.org/de/worterbuch/englisch/unless).

See also [this discussion on GitHub](https://github.com/bbatsov/ruby-style-guide/issues/329).

\* Use `and`, `or` and `not` in conditions.

\*\*Example:\*\* `if admin? and not undeletable?`

\*\*Example:\*\* `while object = stack.pop and object.valid?`

\*\*Rationale:\*\* Easy to read, consistency with other control flow operators.

\* Use `&&`, `||` and `!` in Boolean algebra.

\*\*Example:\*\* `should\_delete = is\_admin && !is\_undeletable`

\*\*Example:\*\* `puts(description || "no description")`

\*\*Rationale:\*\* Consistency with other binary operators (`|`/`&` and `||=`/`&&=`), no problems with operator precedence.

## Markdown and Comments

Please start each sentence on a new line.

\*\*Rationale:\*\* This avoids large diffs when a single word changes at the start of a long paragraph.