How to contribute efficiently

Table of contents

Please read the first section before reporting a bug!

Reporting bugs

The golden rule is to **always open *one* issue for *one* bug**. If you notice several bugs and want to report them, make sure to create one new issue for each of them.

If you're reporting a new bug, you'll make our life simpler (and the fix will come sooner) by following these guidelines:

Search first in the existing database

Issues are often reported several times by various users. It's good practice to **search first in the issue tracker before reporting your issue**. If you don't find a relevant match or if you're unsure, don't hesitate to **open a new issue**. The bugsquad will handle it from there if it's a duplicate.

Specify the platform

Godot runs on a large variety of platforms and operating systems and devices. **In your bug reports, please always specify:**

- Operating system and version (e.g. Windows 10, macOS 10.15, Ubuntu 19.10)
- Godot version (e.g. 3.2, 3.1.2, or the Git commit hash if you're using a development branch)

For bugs that are likely OS-specific and/or graphics-related, please also specify:

- Device (CPU model including architecture, e.g. x86, x86_64, ARM, etc.)
- GPU model (and the driver version in use if you know it)

Bug reports not including the required information may be closed at the maintainers' discretion. If in doubt, always include all the requested information; it's better to include too much information than not enough information.

Specify steps to reproduce

Many bugs can't be reproduced unless specific steps are taken. Please **specify the exact steps** that must be taken to reproduce the condition, and try to keep them as minimal as possible. If you're describing a procedure to follow in the editor, don't hesitate to include screenshots.

Making your bug report easy to reproduce will make it easier for contributors to fix the bug.

Provide a simple, example project

Sometimes, unexpected behavior can happen in your project. In such case, understand that:

- What happens to you may not happen to other users.
- We can't take the time to look at your project, understand how it is set up and then figure out why it's failing.

To speed up our work, **please upload a minimal project** that isolates and reproduces the issue. This is always the **best way for us to fix it**. We recommend attaching a ZIP file with the minimal project directly to the bug report, by drag and dropping the file in the GitHub edition field. This ensures the file can remain available for a long period of time. Only use third-party file hosts if your ZIP file isn't accepted by GitHub because it's too large.

We recommend always attaching a minimal reproduction project, even if the issue may seem simple to reproduce manually.

Note for C# users: If your issue is not Mono-specific, please upload a minimal reproduction project written in GDScript or VisualScript. This will make it easier for contributors to reproduce the issue locally as not everyone has a Mono setup available.

If you've been asked by a maintainer to upload a minimal reproduction project, you *must* do so within 7 days. Otherwise, your bug report will be closed as it'll be considered too difficult to diagnose.

Now that you've read the guidelines, click the link below to create a bug report:

- **Report a bug**

Proposing features or improvements

Since August 2019, the main issue tracker no longer accepts feature proposals. Instead, head to the Godot Proposals repository and follow the instructions in the README file. High-quality feature proposals are more likely to be well-received by the maintainers and community, so do your best :)

See this article for detailed rationale on this change.

Contributing pull requests

If you want to add new engine features, please make sure that:

- This functionality is desired, which means that it solves a common use case that several users will need in their real-life projects.
- You talked to other developers on how to implement it best. See also Proposing features or improvements.
- Even if it doesn't get merged, your PR is useful for future work by another developer.

Similar rules can be applied when contributing bug fixes - it's always best to discuss the implementation in the bug report first if you are not 100% about what would be the best fix.

This blog post outlines the process used by core developers when assessing PRs. We strongly recommend that you have a look at it to know what's important to take into account for a PR to be considered for merging.

In addition to the following tips, also take a look at the Engine development guide for an introduction to developing on Godot.

The Contributing docs also have important information on the PR workflow and the code style we use.

Document your changes

If your pull request adds methods, properties or signals that are exposed to scripting APIs, you **must** update the class reference to document those. This is to ensure the documentation coverage doesn't decrease as contributions are merged.

Update the documentation template using your compiled binary, then fill in the descriptions. Follow the style guide described in the Docs writing guidelines.

If your pull request modifies parts of the code in a non-obvious way, make sure to add comments in the code as well. This helps other people understand the change without having to look at `git blame`.

Write unit tests

When fixing a bug or contributing a new feature, we recommend including unit tests in the same commit as the rest of the pull request. Unit tests are pieces of code that compare the output to a predetermined *expected result* to detect regressions. Tests are compiled and run on GitHub Actions for every commit and pull request.

Pull requests that include tests are more likely to be merged, since we can have greater confidence in them not being the target of regressions in the future.

For bugs, the unit tests should cover the functionality that was previously broken. If done well, this ensures regressions won't appear in the future again. For new features, the unit tests should cover the newly added functionality, testing both the "success" and "expected failure" cases if applicable.

Feel free to contribute standalone pull requests to add new tests or improve existing tests as well.

See Unit testing for information on writing tests in Godot's C++ codebase.

Be nice to the Git history

Try to make simple PRs that handle one specific topic. Just like for reporting issues, it's better to open 3 different PRs that each address a different issue than one big PR with three commits.

When updating your fork with upstream changes, please use `git pull --rebase` to avoid creating "merge commits". Those commits unnecessarily pollute the git history when coming from PRs.

Also try to make commits that bring the engine from one stable state to another stable state, i.e. if your first commit has a bug that you fixed in the second commit, try to merge them together before making your pull request (see `git rebase -i` and relevant help about rebasing or amending commits on the Internet).

This Git style guide has some good practices to have in mind.

See our PR workflow documentation for tips on using Git, amending commits and rebasing branches.

Format your commit messages with readability in mind

The way you format your commit messages is quite important to ensure that the commit history and changelog will be easy to read and understand. A Git commit message is formatted as a short title (first line) and an extended description (everything after the first line and an empty separation line).

The short title is the most important part, as it is what will appear in the `shortlog` changelog (one line per commit, so no description shown) or in the GitHub interface unless you click the "expand" button. As the name says, try to keep that first line under 72 characters. It should describe what the commit does globally, while details would go in the description. Typically, if you can't keep the title short because you have too much stuff to mention, it means you should probably split your changes in several commits :)

Here's an example of a well-formatted commit message (note how the extended description is also manually wrapped at 80 chars for readability):

```
Prevent French fries carbonization by fixing heat regulation

When using the French fries frying module, Godot would not regulate the heat
```

```
and thus bring the oil bath to supercritical liquid conditions, thus causing
unwanted side effects in the physics engine.

By fixing the regulation system via an added binding to the internal feature,
this commit now ensures that Godot will not go past the ebullition temperature
of cooking oil under normal atmospheric conditions.

Fixes #1789, long live the Realm!
```

Note: When using the GitHub online editor or its drag-and-drop feature, *please* edit the commit title to something meaningful. Commits named "Update my_file.cpp" won't be accepted.

Contributing to Godot's translation

You can contribute to Godot's translation from the Hosted Weblate, an open source and web-based translation platform. Please refer to the translation readme for more information.

You can also help translate Godot's documentation on Weblate.

Communicating with developers

The Godot Engine community has many communication channels, some used more for user-level discussions and support, others more for development discussions.

To communicate with developers (e.g. to discuss a feature you want to implement or a bug you want to fix), the following channels can be used:

- Godot Contributors Chat: You will find most core developers there, so it's the go-to platform for direct chat about Godot Engine development. Feel free to start discussing something there to get some early feedback before writing up a detailed proposal in a GitHub issue.
- Bug tracker: If there is an existing issue about a topic you want to discuss, just add a comment to it - many developers watch the repository and will get a notification. You can also create a new issue - please keep in mind to create issues only to discuss quite specific points about the development, and not general user feedback or support requests.
- Feature proposals: To propose a new feature, we have a dedicated issue tracker for that. Don't hesitate to start by talking about your idea on the Godot Contributors Chat to make sure that it makes sense in Godot's context.

Thanks for your interest in contributing!

—The Godot development team