# Contributing Guidelines

Thank you for showing interest in the development of osu!lazer! We aim to provide a good collaborating environment for everyone involved, and as such have decided to list some of the most important things to keep in mind in the process. The guidelines below have been chosen based on past experience.

These are not "official rules" \*per se\*, but following them will help everyone deal with things in the most efficient manner.

## Table of contents

1. [I would like to submit an issue!](#i-would-like-to-submit-an-issue)

2. [I would like to submit a pull request!](#i-would-like-to-submit-a-pull-request)

## I would like to submit an issue!

Issues, bug reports and feature suggestions are welcomed, though please keep in mind that at any point in time, hundreds of issues are open, which vary in severity and the amount of time needed to address them. As such it's not uncommon for issues to remain unresolved for a long time or even closed outright if they are deemed not important enough to fix in the foreseeable future. Issues that are required to "go live" or otherwise achieve parity with stable are prioritised the most.

\* \*\*Before submitting an issue, try searching existing issues first.\*\*

For housekeeping purposes, we close issues that overlap with or duplicate other pre-existing issues - you can help us not to have to do that by searching existing issues yourself first. The issue search box, as well as the issue tag system, are tools you can use to check if an issue has been reported before.

\* \*\*When submitting a bug report, please try to include as much detail as possible.\*\*

Bugs are not equal - some of them will be reproducible every time on pretty much all hardware, while others will be hard to track down due to being specific to particular hardware or even somewhat random in nature. As such, providing as much detail as possible when reporting a bug is hugely appreciated. A good starting set of information consists of:

\* the in-game logs, which are located at:

\* `%AppData%/osu/logs` (on Windows),

\* `~/.local/share/osu/logs` (on Linux and macOS),

\* `Android/Data/sh.ppy.osulazer/logs` (on Android),

\* on iOS they can be obtained by connecting your device to your desktop and [copying the `logs` directory from the app's own document storage using iTunes](https://support.apple.com/en-us/HT201301#copy-to-computer),

\* your system specifications (including the operating system and platform you are playing on),

\* a reproduction scenario (list of steps you have performed leading up to the occurrence of the bug),

\* a video or picture of the bug, if at all possible.

\* \*\*Provide more information when asked to do so.\*\*

Sometimes when a bug is more elusive or complicated, none of the information listed above will pinpoint a concrete cause of the problem. In this case we will most likely ask you for additional info, such as a Windows Event Log dump or a copy of your local lazer database (`client.db`). Providing that information is beneficial to both parties - we can track down the problem better, and hopefully fix it for you at some point once we know where it is!

\* \*\*When submitting a feature proposal, please describe it in the most understandable way you can.\*\*

Communicating your idea for a feature can often be hard, and we would like to avoid any misunderstandings. As such, please try to explain your idea in a short, but understandable manner - it's best to avoid jargon or terms and references that could be considered obscure. A mock-up picture (doesn't have to be good!) of the feature can also go a long way in explaining.

\* \*\*Refrain from posting "+1" comments.\*\*

If an issue has already been created, saying that you also experience it without providing any additional details doesn't really help us in any way. To express support for a proposal or indicate that you are also affected by a particular bug, you can use comment reactions instead.

\* \*\*Refrain from asking if an issue has been resolved yet.\*\*

As mentioned above, the issue tracker has hundreds of issues open at any given time. Currently the game is being worked on by two members of the core team, and a handful of outside contributors who offer their free time to help out. As such, it can happen that an issue gets placed on the backburner due to being less important; generally posting a comment demanding its resolution some months or years after it is reported is not very likely to increase its priority.

\* \*\*Avoid long discussions about non-development topics.\*\*

GitHub is mostly a developer space, and as such isn't really fit for lengthened discussions about gameplay mechanics (which might not even be in any way confirmed for the final release) and similar non-technical matters. Such matters are probably best addressed at the osu! forums.

## I would like to submit a pull request!

We also welcome pull requests from unaffiliated contributors. The [issue tracker](https://github.com/ppy/osu/issues) should provide plenty of issues that you can work on; we also mark issues that we think would be good for newcomers with the [`good-first-issue`](https://github.com/ppy/osu/issues?q=is%3Aissue+is%3Aopen+label%3Agood-first-issue) label.

However, do keep in mind that the core team is committed to bringing osu!lazer up to par with stable first and foremost, so depending on what your contribution concerns, it might not be merged and released right away. Our approach to managing issues and their priorities is described [in the wiki](https://github.com/ppy/osu/wiki/Project-management).

Here are some key things to note before jumping in:

\* \*\*Make sure you are comfortable with C\# and your development environment.\*\*

While we are accepting of all kinds of contributions, we also have a certain quality standard we'd like to uphold and limited time to review your code. Therefore, we would like to avoid providing entry-level advice, and as such if you're not very familiar with C\# as a programming language, we'd recommend that you start off with a few personal projects to get acquainted with the language's syntax, toolchain and principles of object-oriented programming first.

In addition, please take the time to take a look at and get acquainted with the [development and testing](https://github.com/ppy/osu-framework/wiki/Development-and-Testing) procedure we have set up.

\* \*\*Make sure you are familiar with git and the pull request workflow.\*\*

[git](https://git-scm.com/) is a distributed version control system that might not be very intuitive at the beginning if you're not familiar with version control. In particular, projects using git have a particular workflow for submitting code changes, which is called the pull request workflow.

To make things run more smoothly, we recommend that you look up some online resources to familiarise yourself with the git vocabulary and commands, and practice working with forks and submitting pull requests at your own pace. A high-level overview of the process can be found in [this article by GitHub](https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/proposing-changes-to-your-work-with-pull-requests).

\* \*\*Double-check designs before starting work on new functionality.\*\*

When implementing new features, keep in mind that we already have a lot of the UI designed. If you wish to work on something with the intention of having it included in the official distribution, please open an issue for discussion and we will give you what you need from a design perspective to proceed. If you want to make \*changes\* to the design, we recommend you open an issue with your intentions before spending too much time to ensure no effort is wasted.

\* \*\*Make sure to submit pull requests off of a topic branch.\*\*

As described in the article linked in the previous point, topic branches help you parallelise your work and separate it from the main `master` branch, and additionally are easier for maintainers to work with. Working with multiple `master` branches across many remotes is difficult to keep track of, and it's easy to make a mistake and push to the wrong `master` branch by accident.

\* \*\*Refrain from making changes through the GitHub web interface.\*\*

Even though GitHub provides an option to edit code or replace files in the repository using the web interface, we strongly discourage using it in most scenarios. Editing files this way is inefficient and likely to introduce whitespace or file encoding changes that make it more difficult to review the code.

Code written through the web interface will also very likely be questioned outright by the reviewers, as it is likely that it has not been properly tested or that it will fail continuous integration checks. We strongly encourage using an IDE like [Visual Studio](https://visualstudio.microsoft.com/), [Visual Studio Code](https://code.visualstudio.com/) or [JetBrains Rider](https://www.jetbrains.com/rider/) instead.

\* \*\*Add tests for your code whenever possible.\*\*

Automated tests are an essential part of a quality and reliable codebase. They help to make the code more maintainable by ensuring it is safe to reorganise (or refactor) the code in various ways, and also prevent regressions - bugs that resurface after having been fixed at some point in the past. If it is viable, please put in the time to add tests, so that the changes you make can last for a (hopefully) very long time.

\* \*\*Run tests before opening a pull request.\*\*

Tying into the previous point, sometimes changes in one part of the codebase can result in unpredictable changes in behaviour in other pieces of the code. This is why it is best to always try to run tests before opening a PR.

Continuous integration will always run the tests for you (and us), too, but it is best not to rely on it, as there might be many builds queued at any time. Running tests on your own will help you be more certain that at the point of clicking the "Create pull request" button, your changes are as ready as can be.

\* \*\*Run code style analysis before opening a pull request.\*\*

As part of continuous integration, we also run code style analysis, which is supposed to make sure that your code is formatted the same way as all the pre-existing code in the repository. The reason we enforce a particular code style everywhere is to make sure the codebase is consistent in that regard - having one whitespace convention in one place and another one elsewhere causes disorganisation.

\* \*\*Make sure that the pull request is complete before opening it.\*\*

Whether it's fixing a bug or implementing new functionality, it's best that you make sure that the change you want to submit as a pull request is as complete as it can be before clicking the \*Create pull request\* button. Having to track if a pull request is ready for review or not places additional burden on reviewers.

Draft pull requests are an option, but use them sparingly and within reason. They are best suited to discuss code changes that cannot be easily described in natural language or have a potential large impact on the future direction of the project. When in doubt, don't open drafts unless a maintainer asks you to do so.

\* \*\*Only push code when it's ready.\*\*

As an extension of the above, when making changes to an already-open PR, please try to only push changes you are reasonably certain of. Pushing after every commit causes the continuous integration build queue to grow in size, slowing down work and taking up time that could be spent verifying other changes.

\* \*\*Make sure to keep the \*Allow edits from maintainers\* check box checked.\*\*

To speed up the merging process, collaborators and team members will sometimes want to push changes to your branch themselves, to make minor code style adjustments or to otherwise refactor the code without having to describe how they'd like the code to look like in painstaking detail. Having the \*Allow edits from maintainers\* check box checked lets them do that; without it they are forced to report issues back to you and wait for you to address them.

\* \*\*Refrain from continually merging the master branch back to the PR.\*\*

Unless there are merge conflicts that need resolution, there is no need to keep merging `master` back to a branch over and over again. One of the maintainers will merge `master` themselves before merging the PR itself anyway, and continual merge commits can cause CI to get overwhelmed due to queueing up too many builds.

\* \*\*Refrain from force-pushing to the PR branch.\*\*

Force-pushing should be avoided, as it can lead to accidentally overwriting a maintainer's changes or CI building wrong commits. We value all history in the project, so there is no need to squash or amend commits in most cases.

The cases in which force-pushing is warranted are very rare (such as accidentally leaking sensitive info in one of the files committed, adding unrelated files, or mis-merging a dependent PR).

\* \*\*Be patient when waiting for the code to be reviewed and merged.\*\*

As much as we'd like to review all contributions as fast as possible, our time is limited, as team members have to work on their own tasks in addition to reviewing code. As such, work needs to be prioritised, and it can unfortunately take weeks or months for your PR to be merged, depending on how important it is deemed to be.

\* \*\*Don't mistake criticism of code for criticism of your person.\*\*

As mentioned before, we are highly committed to quality when it comes to the lazer project. This means that contributions from less experienced community members can take multiple rounds of review to get to a mergeable state. We try our utmost best to never conflate a person with the code they authored, and to keep the discussion focused on the code at all times. Please consider our comments and requests a learning experience, and don't treat it as a personal attack.

\* \*\*Feel free to reach out for help.\*\*

If you're uncertain about some part of the codebase or some inner workings of the game and framework, please reach out either by leaving a comment in the relevant issue or PR thread, or by posting a message in the [development Discord server](https://discord.gg/ppy). We will try to help you as much as we can.

When it comes to which form of communication is best, GitHub generally lends better to longer-form discussions, while Discord is better for snappy call-and-response answers. Use your best discretion when deciding, and try to keep a single discussion in one place instead of moving back and forth.