Contributing to Terasology

==================================

If you would like to contribute code, documentation, or other assets you can do so through GitHub by forking the repository and sending a pull request (PR). You can also simply report issues (bugs), but please search for any previous reports first.

\*When submitting code, please make every effort to follow existing conventions and style in order to keep the code as readable as possible.\*

Read on for an overview and [check the wiki for more details](https://github.com/MovingBlocks/Terasology/wiki). For questions please join us in our [forum](http://forum.terasology.org/forum/) or on `#terasology` (irc.freenode.net).

File an Issue

-------------

You can report bugs and feature requests to [GitHub Issues](https://github.com/MovingBlocks/Terasology/issues). As mentioned please look for a similar existing issue before submitting a new one.

For finding easy to do issues to start with look at the [Good First Issue](https://github.com/MovingBlocks/Terasology/labels/Good%20First%20Issue) issues.

We prefer questions and support requests be posted in the [forum](http://forum.terasology.org/forum).

\_\_Please provide as much information as possible to help us solve problems and answer questions better!\_\_

Commits and Commit Messages

---------------------------

Follow these guidelines when creating public commits and writing commit messages.

1. If your work spans multiple local commits make sure they can be merged directly into the target branch. If you have some commit and merge noise in the history have a look at git-rebase. Ideally, every commit should be able to be used in isolation -- that is, each commit must build and pass all tests.

1. The first line of the commit message should be a descriptive sentence about what the commit is doing. It should be possible to fully understand what the commit does by just reading this single line.

1. \*\*Optional:\*\* Following the single line description (ideally no more than 70 characters long) should be a blank line followed by a list with the details of the commit.

1. \*\*Optional:\*\* Add `Fix/Fixing/Fixes/Close/Closing/Refs #issue` if you want to mark the issue mentioned as complete in the issue tracker when the change is merged.

License

-------

Submissions must be licensed under the [Apache License, Version 2.0](http://www.apache.org/licenses/LICENSE-2.0.html).

If you are adding a new file it should have a header like this (automatically available as a template in IntelliJ):

~~~

/\*

\* Copyright 2016 MovingBlocks

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

~~~

How to Submit a Pull Request

----------------------------

Pull requests are highly appreciated! Please follow the simple guidelines below.

1. Fork the repository to your personal GitHub account.

1. Create a topic branch for every separate change you make. The branch should have a short but explanatory name, such as "MakesGrassGrow".

1. Apply your changes, committing at logical breaks. Make sure your changes are well-tested.

1. If your code includes any strings shown to the user, make sure they're accessed through the translation system as opposed to hardcoded values.

1. Check any unit tests that cover the affected code and if at all possible include new ones for new code.

1. Push your branch to your personal account and [create a pull request](https://help.github.com/articles/using-pull-requests/).

1. Watch for comments or acceptance on your PR. The PR can be updated by just pushing to the original branch.

\_\_Please note:\_\_ if you want to change multiple things that don't depend on each other, make sure you create a new branch off the main develop branch before making more changes - that way we can take in each change separately.

How to Merge a Pull Request

---------------------------

For trusted contributors with push access to our root repos you are welcome to help merge pull requests.

1. Consider what scope is suitable for the change being PRed.

\* \*\*Trivial\*\*: If it is a single typo fix in a comment or other \*inactive text\* it can be merged without further review.

\* \*\*Patch\*\*: Small bug fixes. Be careful: even a string value changed can have unexpected impact elsewhere if it gets used as a key somewhere, involved in our internationalization, etc. Consider any possible integrations (like our translation server) and test the change locally. Report any concerns in the PR or merge if all appears well.

\* \*\*Minor\*\*: If substantial code is being changed/added please review as able and ping relevant contributors with `@username` to bring their attention to the PR for review. Probably one additional person should review then either person can merge.

\* \*\*Major\*\*: If any involved code is likely to cause a backwards incompatibility of some sort the PR needs the most possible review as it would require a major release. Consider if it is possible to deprecate the old version of the feature and provide a bridge to the new version to restore backwards compatibility. Alternatively should the change be parked in a longer term branch waiting for the next major release?

1. React appropriately to the scope and test what seems appropriate locally. Remember to run local multiplayer if it seems likely the change could cause an effect there.

1. If any issues are found or concerns noted mention them in the PR and ping anybody appropriate with a `@username` mention.

1. If sufficient review has been done and no issues have been noted push the merged code to GitHub to auto-close the PR as merged.

1. Assign the PR and any completed issues to an appropriate milestone if available for later release handling.