# How to contribute to jMonkeyEngine

First and foremost, you have to familiarize yourself with Git & GitHub. Dig through

[help.github.com](https://help.github.com/), [try.github.io](http://try.github.io/) and the [gh cheat sheet](https://github.com/tiimgreen/github-cheat-sheet/blob/master/README.md) if these are new topics for you. If you'd like to contribute with something other than code, just tell us about it on the forum.

## Communication

Communication always comes first. \*\*All\*\* code changes and other contributions should start with the [forum](http://hub.jmonkeyengine.org/). Make a thread to explain your change and show us the important bits of your code. If the code is too long to be posted within the forum’s code tags, please paste your code in a Gist or pastebin and link to the submission in your thread. You are required to register on our website in order to create threads. (We do support login via GitHub though).

### New Contributors

Check out the [Projects](https://github.com/jMonkeyEngine/jmonkeyengine/projects/1) tab, where the team has prioritized issues that you as a new contributor can undertake that will familiarize you to the workflow of contributing. This highlights some issues the team thinks would be a good start for new contributors but you are free to contribute on any other issues or integration you wish.

When you're ready to submit your code, just make a [pull request](https://help.github.com/articles/using-pull-requests).

- Do not commit your code until you have received proper feedback.

- In your commit log message, please refer back to the originating forum thread (example) for a ‘full circle’ reference. Also please [reference related issues](https://help.github.com/articles/closing-issues-via-commit-messages) by typing the issue hashtag.

- When committing, always be sure to run an update before you commit. If there is a conflict between the latest revision and your patch after the update, then it is your responsibility to track down the update that caused the conflict and determine the issue (and fix it). In the case where the breaking commit has no thread linked (and one cannot be found in the forum), then the contributor should contact an administrator and wait for feedback before committing.

- If your code is committed and it introduces new functionality, please edit the wiki accordingly. We can easily roll back to previous revisions, so just do your best; point us to it and we’ll see if it sticks!

p.s. We will try hold ourselves to a [certain standard](http://www.defmacro.org/2013/04/03/issue-etiquette.html) when it comes to GitHub etiquette. If at any point we fail to uphold this standard, let us know.

#### Core Contributors

Developers in the Contributors team can push directly to Main instead of submitting pull requests, however for new features it is often a good idea to do a pull request as a means to get a last code review.

## Customs around integration, branching, tagging, and releases

- Most pull requests are integrated directly into the master branch of the repository.

- Integrators should note, unless the history of the pull request is important, it should be integrated to a single commit using “squash and merge”. If the history is important, favor “rebase and merge”. Don’t create a merge commit unless GitHub cannot rebase the PR.

- For each major release (such as v3.0 or v3.3), an appropriately named release branch is created in the repository.

- For each minor (or “dot-dot”) release (such as v3.2.3), an appropriately named tag is created in the repository.

- In general, library changes that plausibly might break existing apps appear only in major releases, not minor ones.

## Building the engine

1. Install [Gradle](http://www.gradle.org/)

2. Navigate to the project directory and run 'gradle build' from command line to build the engine.

## Best Practices

### Git essentials

- [Creating good pull requests](http://seesparkbox.com/foundry/creating\_good\_pull\_requests)

- [How to write the perfect pull request](https://github.com/blog/1943-how-to-write-the-perfect-pull-request?utm\_content=buffer0eb16&utm\_medium=social&utm\_source=twitter.com&utm\_campaign=buffer)

### Testing

general testing tips? WIP

### Code Quality

We generally abide by the standard Java Code Conventions. Besides that, just make an effort to write elegant code:

1. Handles errors gracefully

2. Only reinvents the wheel when there is a measurable benefit in doing so.

3. Has consistent naming conventions.

4. Has comments around ugly code explaining why it is ugly.

5. Compiles (or runs if interpreted) without warnings.

## Reporting bugs

1. Start by searching the [forum](http://hub.jmonkeyengine.org) and GH issue tracker for duplicates.

2. Create a new issue, explaining the problem in proper detail (templates pending).

## Documentation

- How to edit the [wiki](https://github.com/jMonkeyEngine/wiki).

- How to edit JavaDocs - WIP