Contribution

Introduction

It is assumed that you know a little about node.js and git. If not, here's some help to get started with git and here's some help to get started with node.js.

- Install Node.js
- Install Git
- Fork three.js
- Open your OS's terminal
- Change into the directory you'd like
- Clone your forked repo

  ```
  git clone https://github.com/[yourgithubname]/three.js.git
  ```

- Go into the three.js directory.

  ```
  cd ./three.js
  ```

- Install the dependencies

  ```
  npm install
  ```

Next Steps

As per the npm standard, 'start' is the place to begin the package.

```
npm start
```

This script will start a local server similar to threejs.org, but instead will be hosted on your local machine. Browse to http://localhost:8080/ to check it out. It also automatically creates the 'build/three.js' and 'build/three.module.js' scripts anytime there is a change within your three.js directory.

The next most important script runs all the appropriate testing. The E-2-E testing is intended to be run by github actions.

```
npm test
```

The linting is there to keep a consistent code style across all of the code and the testing is there to help catch bugs and check that the code behaves as expected. It is important that neither of these steps comes up with any errors due to your changes.

Many linting errors can be fixed automatically by running

```
npm lint-fix
```

If you'd like to make a minified version of the build files i.e. 'build/three.min.js' run:

```
npm run build
```

Making changes

When you've decided to make changes, start with the following:

- Update your local repo

  ```
  git pull https://github.com/mrdoob/three.js.git
  git push
  ```

- Make a new branch from the dev branch

  ```
  git checkout dev
  git branch [mychangesbranch]
  git checkout [mychangesbranch]
  ```

- Add your changes to your commit.
- Push the changes to your forked repo.
- Open a Pull Request (PR)

Important notes:

- Don't include any build files in your commit.
- Not all new features will need a new example. Simpler features could be incorporated into an existing example. Bigger features may be asked to add an example demonstrating the feature.
- Making changes may require changes to the documentation. To update the docs in other languages, simply copy the English to begin with.
- it's good to also add an example and screenshot for it, for showing how it's used and for end-to-end testing.
- If you modify existing code, run relevant examples to check they didn't break and there wasn't performance regress.
- If you add some assets for the examples (models, textures, sounds, etc), make sure they have a proper license allowing for their use here, less restrictive the better. It is unlikely for large assets to be accepted.
- If some issue is relevant to the patch/feature, please mention it with a hash (e.g. #2774) in a commit message to get cross-reference in GitHub.
- If you modify files in `examples/jsm` directory, then don't perform any changes in the `examples/js`, non-module files are auto-generated by running `npm run build-examples`.
- If the end-to-end test failed in Travis and you are sure that all is correct, make a new screenshot with

```
npm run make-screenshot <example_1_name> ...<example_N_name>
```

- Once done with a patch/feature do not add more commits to a feature branch
- Create separate branches per patch or feature.
- If you make a PR but it is not actually ready to be pulled into the dev branch then please convert it to a draft PR.

This project is currently contributed mostly via everyone's spare time. Please keep that in mind as it may take some time for the appropriate feedback to get to you. If you are unsure about adding a new feature, it might be better to ask first to see whether other people think it's a good idea.