# Orbitant Documentation

***Release 0.1.0***

**Matthew Scott**

November 07, 2016

Contents


**1 Overview of Metrasynth Orbitant** **1**
- 1.1 Purpose . . . 1
- 1.2 Status . . . 1
- 1.3 Requirements . . . 1

**2 Contributing** **3**
- 2.1 Bug reports . . . 3
- 2.2 Documentation improvements . . . 3
- 2.3 Feature requests and feedback . . . 3
- 2.4 Development . . . 3

**3 Changelog** **5**
- 3.1 0.1.0 (not yet released) . . . 5

**4 Authors** **7**

**5 Indices and tables** **9**

CHAPTER 1

# Overview of Metrasynth Orbitant

## 1.1 Purpose

MIDI clock and musical timing utilities for Python.

## 1.2 Status

**Warning:** This package is very experimental. Features and APIs may change a great deal before being considered stable.

## 1.3 Requirements

macOS:

```
$ brew install rtmidi
```

CHAPTER 2

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

## 2.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

## 2.2 Documentation improvements

Orbitant could always use more documentation, whether as part of the official Orbitant docs, in docstrings, or even on the web in blog posts, articles, and such.

## 2.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/metrasynth/orbitant/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

## 2.4 Development

To set up *orbitant* for local development:

1. Fork orbitant (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/orbitant.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

### 2.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`) [1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

### 2.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 3

# Changelog

## 3.1 0.1.0 (not yet released)

- Initial release.

CHAPTER 4

# Authors


- Matthew Scott

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search