Solar Sails Documentation

Release 0.1.1

Matthew Scott

Feb 21, 2018

Contents

CHAPTER 1

Overview of Solar Sails

"SunVox Augmentation and Interactive Live-coding System"

Solar Sails is a cross-platform app that augments the SunVox modular music studio.

It combines Radiant Voices, Solar Flares, and sunvox-dll-python into a downloadable app that provides a workspace to make use of those tools.

1.1 Downloading and installing

Solar Sails is still alpha-quality software.

Please follow the steps outlined in the installation docs.

1.2 Requirements for downloadable app

- MacOS 10.11 El Capitan, or later
- Windows 7, or later, 64-bit install
- Linux, 64-bit install, Ubuntu 16.04 or equivalent

1.3 Requirements for building from source

- Python 3.6
- OS supported by PyQt5 and sunvox-dll-python.

1.4 Metrasynth

Solar Sails is part of the Metrasynth project.

CHAPTER 2

Installing Solar Sails

Follow the steps below to download and install Solar Sails for your platform.

Proceed to *Getting Started* after you've finished installing the app.

2.1 macOS

1. Visit the Travis CI build page for Solar Sails. Click on the build job that says "Xcode: xcode8".
2. Scroll to the very bottom of the page. Find the line that begins with `download_url: ....`
3. Highlight the entire URL starting with `https://s3.amazonaws.com` and ending with `.tar.bz2`.
4. Copy the URL to your clipboard.
5. Paste the URL into your browser location bar and press return. The download should begin.
6. After the download completes, open the downloaded `Solar Sails.app.tar.bz2` file to extract the app.
7. Move the resulting Solar Sails app to your Applications folder.
8. Open the app. You will likely receive a warning dialog. Refer to Apple's Open an app from an unidentified developer page, and authorize the Solar Sails app.
9. If the app closes immediately the first time you open it, open it a second time. This is a known issue, but only occurs once each time you install and authorize a new version.

If the app launches successfully, you'll see a small window containing large "Solar Sails" text. Now proceed to *Getting Started*.

2.2 Windows

1. Visit the AppVeyor latest build artifact page for Solar Sails.
2. Click on `dist.zip`. The download should begin.

3. After the download completes, reveal the file in explorer.
4. Right-click the file and click Extract All.
5. Choose the destination folder of your choice and click Extract. A folder named `solar-sails` will be created within the destination folder.
6. Open the `solar-sails` application inside the folder. Click Run when the security warning dialog appears.

If the app launches successfully, you'll see a small window containing large "Solar Sails" text. Now proceed to *Getting Started*.

2.3 Linux

It is difficult to provide a bundled package that works well across several Linux distributions. Therefore please follow the steps below for your distribution.

2.3.1 Debian variants without Python 3.6

Examples: Debian, Linux Mint, Ubuntu.

This assumes you cannot install `python3-dev`, or if you do, only Python 3.5 is available. The steps below help you use `pyenv` to install Python 3.6.

First-time setup

```
$ sudo apt install python-dev build-essential libasound2-dev librtmidi-dev libpcre3-
↪dev \
    zlib1g-dev libbz2-dev libreadline-dev libssl-dev libsqlite3-dev

$ git clone https://github.com/pyenv/pyenv.git ~/.pyenv
$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
$ echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi' >
↪> ~/.bashrc
$ exec "$SHELL"

$ pyenv install 3.6.4
$ git clone https://github.com/metrasynth/solar-sails ~/solar-sails
$ cd solar-sails
$ pyenv shell 3.6.4
$ python --version  # => Python 3.6.4
$ python -m venv venv
$ source venv/bin/activate
$ pip install -r requirements/app.txt
$ pip install -e $PWD
```

Starting Solar Sails

```
$ ~/solar-sails/venv/bin/sails-gui <optional-mmckpy-pathname>
```

Updating Solar Sails

```
$ cd ~/solar-sails
$ source venv/bin/activate
$ git pull
$ pip install -r requirements/app.txt
$ pip install -e $PWD
```

2.3.2 Solus

This is based on Solus 3 Budgie.

First-time setup

```
$ sudo eopkg install -c system.devel
$ sudo eopkg install alsa-lib-devel git
$ cd ~
$ git clone https://github.com/metrasynth/solar-sails
    # The following is optional, but contains many MMCK example scripts:
$ git clone https://github.com/metrasynth/gallery
$ cd ~/solar-sails
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install -r requirements/app.txt
$ pip install -e .
```

Starting Solar Sails

```
$ ~/solar-sails/venv/bin/sails-gui <optional-mmckpy-pathname>
```

Updating Solar Sails

```
$ cd ~/solar-sails
$ source venv/bin/activate
$ git pull
$ pip install -r requirements/app.txt
$ pip install -e $PWD
```

CHAPTER 3

Getting Started

3.1 First time configuration

Open the preferences for the app.

On MacOS, open the Solar Sails menu, then click Preferences.

On Linux and Windows, open the File menu, then click Settings.

3.1.1 SunVox tab

The first workspace path listed in this tab will be used for reading/writing SunVox clipboard files.

Click the `[+]` button at the bottom.

Navigate to, then choose, the folder containing your SunVox app.

SunVox and Solar Sails both use this location to store SunVox clipboard data.

3.1.2 Audio tab

This tab is not yet functional.

For audio playback, Solar Sails will use the default system output device at the time of application launch.

3.1.3 MIDI tab

Solar Sails will listen to MIDI note and CC events.

MIDI listening is omnichannel.

Solar Sails always listens to the interface named `Metrasynth Solar Sails`. On Linux and MacOS, Solar Sails will create this as a virtual interface. On Windows, use the free loopMIDI app to create a virtual interface with this name.

Interfaces

Available MIDI interfaces are listed here.

Solar Sails listens to selected interfaces.

Click on an interface to select or deselect it.

If you connect an interface while Solar Sails is running, click Refresh. If an interface was selected in the past, it will be automatically reselected.

Ignore MIDI in background

Turn this on if you want Solar Sails to ignore MIDI events when in the background.

This can be useful when you are using the same MIDI controller to input notes into SunVox.

CC Mappings

This is a list of CCs that will be auto-assigned to controllers in the *MetaModule Construction Kit (MMCK)*.

The format is `###=Label`. `###` is the decimal notation of the CC to listen to. `Label` is how you want the CC to be labeled in the user interface.

3.2 MetaModule Construction Kit (MMCK)

MMCK is a powerful tool for creating new SunVox MetaModules.

Find out more at *MetaModule Construction Kit (MMCK)*.

3.3 Polyphonist

This tool is currently broken.

These docs will be updated once it’s fixed.

CHAPTER 4

MetaModule Construction Kit (MMCK)

Solar Sails' MMCK lets you instantly create new SunVox MetaModules using scripts written in the Python language.

4.1 Workflow overview

1. Open a `.mmckpy` file.
2. Change parameters to your liking.
3. Audition sounds by sending notes from a MIDI controller. Adjust SunVox controllers on-screen, or using CC values sent from a MIDI controller.
4. Choose the SunVox controllers to assign to up to 27 user defined controllers.
5. Copy the MetaModule to the SunVox clipboard, or save the MetaModule to a `.sunsynth` or `.sunvox` file.

Note: If you don't have a hardware MIDI controller, try using SunVox itself. Create a MultiSynth module and set its MIDI output to `Metrasynth Solar Sails`.

4.2 Concepts

4.2.1 MMCK script

A Python script containing specifications for *parameters* and a *project builder*. Their filenames end with the `.mmckpy` extension.

4.2.2 Project

The SunVox project contained within the MetaModule.

4.2.3 Parameters

Values that control aspects of how a *project* is created. The values given by the user are passed directly to the *project builder*.

4.2.4 Project builder

The code that builds the actual project, making use of the Radiant Voices API. The resulting project is based on the *parameters* given by the user, processed by algorithms implemented by the *MMCK script*.

4.2.5 Controllers

These refer to SunVox module controllers, grouped and presented in a way that is specified by the *MMCK script*. As *parameters* are changed, controllers may be added or removed.

You can adjust the value of the controller using standard keyboard and mouse controls.

For each controller, you can specify a *user defined controllers* slot to assign it to.

You can also specify a MIDI CC number to use for adjusting the SunVox controller. CCs are automatically assigned to controllers in the order they are listed in Solar Sails preferences.

See *Getting Started* for how to make CCs available for use.

4.2.6 User defined controllers

These are the *controllers* exposed via the user defined controllers of the exported SunVox MetaModule.

Multiple controllers can be grouped into a single "macro" user defined controller. To do so, assign more than one controller to a single user defined controller slot.

4.2.7 Log

Textual feedback from the *MMCK script* and the Solar Sails app itself.

The log is cleared whenever a *MMCK script* is loaded, or whenever *parameters* are changed.

Here are some types of information you can expect to see in the log:

- MIDI "note on" events.
- Transport control events.
- Output and/or error details while the *parameters* are being defined.
- Output and/or error details while the *project builder* is running.
- "Finished loading" messages when the *project* is ready and *controllers* have appeared.
- "Exported synth" messages when a `.sunsynth` version of the *project* is saved.
- "Exported project" messages when a `.sunvox` version of the *project* is saved.

4.3 Workflow detail

4.3.1 Opening an existing MMCK script

Select the *File* > *Open* menu, then open a file ending with `.mmckpy`. After a script successfully loads, the MMCK main window will appear.

The initial *parameters* will be set if this is the first time you've opened the file. The previous *parameters* you used will be loaded if you've opened this file already.

MacOS keyboard shortcut: Command+O.

Linux and Windows keyboard shortcut: Ctrl+O.

Note: You can find an assortment of scripts at the Metrasynth Gallery GitHub repo.

4.3.2 Changing parameters

In the *parameters* window, change the values to alter how the *project* is built.

After a small delay, your changes will be processed by the *project builder*.

The *controllers* available for the *project* will appear.

Note: While many parameters have sensible boundaries, keep in mind that some parameter combinations can be potent! With some scripts, certain parameter combinations can result in unwanted behavior.

Some examples of unwanted behavior: volumes becoming too loud, harsh noises made by a *project builder* that utilizes randomness, or overloading the SunVox sound engine.

Just keep your master volume control nearby, and be familiar with your operating system's "force quit" features.

4.3.3 Auditioning notes

Use an available MIDI controller to send MIDI note on/off events to Solar Sails.

Notes will be sent to module `01` as created by the *project builder*.

What you hear comes directly from the SunVox audio engine and will sound exactly the same in SunVox itself.

When a note is played, the note and velocity are added to the *log*. Note off events do not appear in the *log*.

4.3.4 Auditioning patterns

Some scripts are designed to create patterns.

Use the *Transport* > *Play from beginning* and *Transport* > *Stop* menu items to control playback.

Play From Beginning keyboard shortcut: F11.

Stop keyboard shortcut: F12.

4.3.5 Adjusting controllers

Controllers are presented in "logical" groupings. This means a single group in MMCK could actually represent several modules.

Use the data entry widget provided to change a controller's value. If the value is numeric, use the slider or numerical entry widget to set the value. If the value is discrete, select the desired from the dropdown box.

Or, use a MIDI controller to send CC values to Solar Sails. The CC values received will be mapped linearly to the value range of the mapped controller_(s).

Use the second combobox to see which CC is mapped to a **controller_**, or to map it to a different CC. You can assign a single CC to multiple controllers. You can also disable a mapping by choosing the first entry, which is blank.

4.3.6 Selecting user defined controllers

To expose a controller in SunVox, it must be mapped to a **'user defined controller'_**. These correspond to controllers 6 to 32 (06h to 20h) in an exported MetaModule.

Use the first combobox to map a **controller_** to a **'user defined controller'_** slot.

You can map up to 16 controllers to a single slot. When doing so, a MultiCtl module will be automatically created to propagate value changes accordingly.

4.3.7 Exporting to SunVox clipboard

When you like a sound, use the *Edit* > *Copy to SunVox clipboard* menu item. Then, switch to SunVox, and use the *Paste* action in the module view. Your module is now ready for immediate use.

MacOS keyboard shortcut: Command+Shift+C.

Linux and Windows keyboard shortcut: Ctrl+Shift+C.

Note: Make sure you have added a workspace path as described in *Getting Started*.

4.3.8 Exporting to a .sunsynth module file

Use *File* > *Export MetaModule* menu item. A time-stamped `.sunsynth` filename will be created, and the *project* will be saved to that file as a MetaModule. The *log* will show the full path of the exported file.

MacOS keyboard shortcut: Command+E.

Linux and Windows keyboard shortcut: Ctrl+E.

4.3.9 Exporting to a .sunvox project file

Use *File* > *Export Project* menu item. A time-stamped `.sunvox` filename will be created, and the *project* will be saved to that file. The *log* will show the full path of the exported file.

Project files do not contain *user defined controllers*.

MacOS keyboard shortcut: Command+Shift+E.

Linux and Windows keyboard shortcut: Ctrl+Shift+E.

4.3.10 Restoring parameters from a .sunsynth file

(to be written)

4.4 Creating your own MMCK scripts

4.4.1 Creating a .mmckpy file

(to be written)

4.4.2 Specifying parameters

(to be written)

4.4.3 Building the MetaModule project

(to be written)

4.4.4 Auto-reloading

(to be written)

Note: Some kinds of errors cause the auto-reload to stop working. If this happens, quit the Solar Sails app and reopen it.

CHAPTER 5

Contributing

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

5.1 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.

5.2 Documentation improvements

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

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/metrasynth/solar-sails/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 :)

5.4 Development

To set up *solar-sails* for local development:

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

   ```
   git clone git@github.com:your_name_here/solar-sails.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.

5.4.1 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`.

5.4.2 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 6

Building Solar Sails

6.1 macOS

From a Python 3.6 virtualenv

```
$ pyinstaller -y specs/gui.spec
```

6.2 Windows

Install Anaconda Python 3.6, 64-bit.

conda create -n sv36 python=3.6 anaconda

activate sv36

python -m venv env

envscriptsactivate

for each repo, pip install -e ., in this order:

- sunvox-dll-python
- radiant-voices
- orbitant
- sunvosc
- solar-flares
- solar-sails

pip install pypiwin32 pyinstaller

6.3 Linux

Ensure python 3.6 is installed

kxstudio:

> apt-get -y install git curl build-essential
>
> use pyenv-installer to install pyenv

CHAPTER 7

Changelog

7.1 0.1.1 (under development)

- Add getting started and basic usage docs for MMCK.

7.2 0.1.0 (2016-11-09)

- Initial release.

CHAPTER 8

Authors

- Matthew Scott

CHAPTER 9

Indices and tables

- genindex
- modindex
- search