OM-Harmonist Tutorials

Written by Hangzhong Lian

Contact Email: hangzhonglian@gmail.com

(https://soundcloud.com/hangzhonglian)

Version: September, 2025

Contents

Chapter 1: Display Module
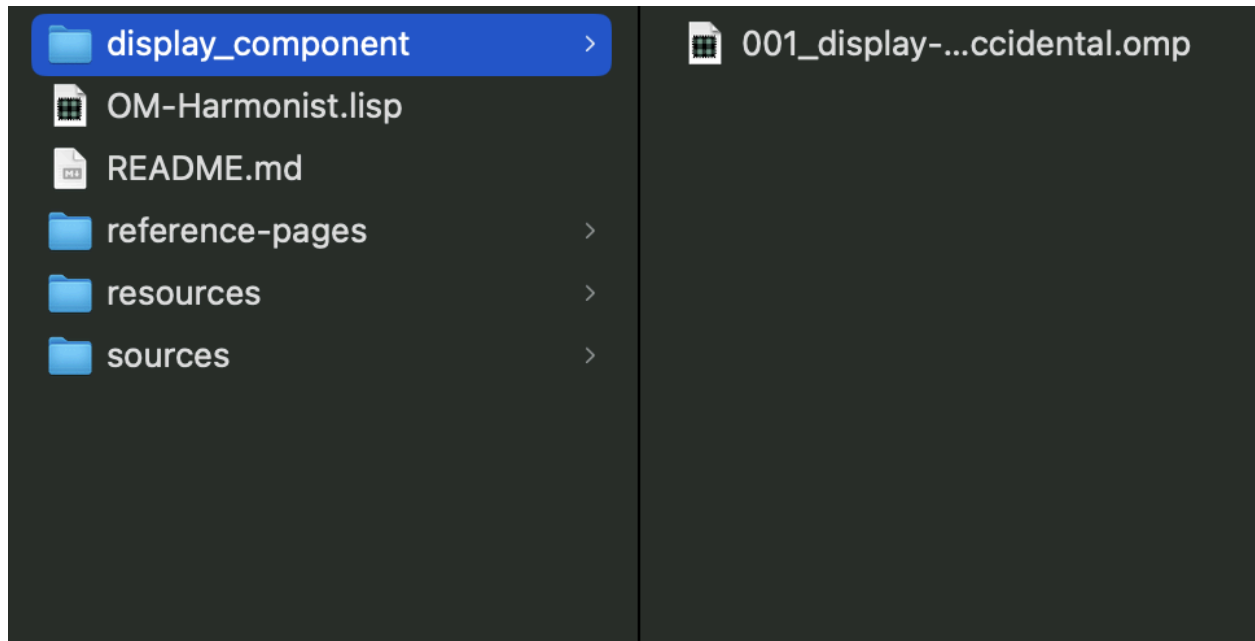
After you have downloaded, unzipped, and installed the OM-Harmonist library into the correct location, please follow these steps:

1. Inside the main `OM-Harmonist` folder, locate and open the `display_component` sub-folder.

2. Drag the Patch from this folder directly into your current OpenMusic workspace (as shown in the image below).

3. Follow the internal instructions within the patch itself to complete the configuration.

#P"/Applications/OM

install the patch

before use:
<-------------1. Make sure that the "scales.lisp" file is in THIS directory!
(it is the default directory of OM 6.10.1 in macOS10.10)
Otherwise you need to find it and write or choose the
correct pathname. (with -------->)

file-chooser

2. Evaluate the
sub-patch "install the
patch" with key "v" ,
or cmd + click

uninstall the patch

4. If you want to roll back the installation,
evaluate the sub-patch "uninstall the patch",
the display of the score object will be restored
in a few milliseconds.

3. double-click the Chord-seq (or other
score object), set the Approx(pop-up
menu) to "1/16#tonal"

(6000 6012 6038 6050 6062 6088
6200 6212 6238 6250 6262 6288)   (0 500)

"bb" = x + 12
"b" = x + 38
beca = x + 50
"#" = x + 62
"x" = x + 88

x must be a basic tone without accidental
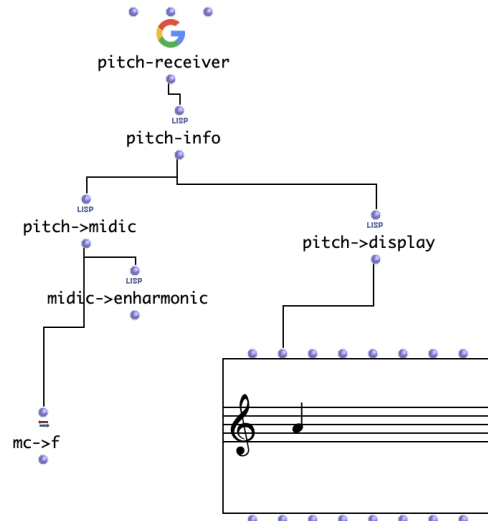**like C3(4800) G4(6700) D9(12200) B2(4700)**

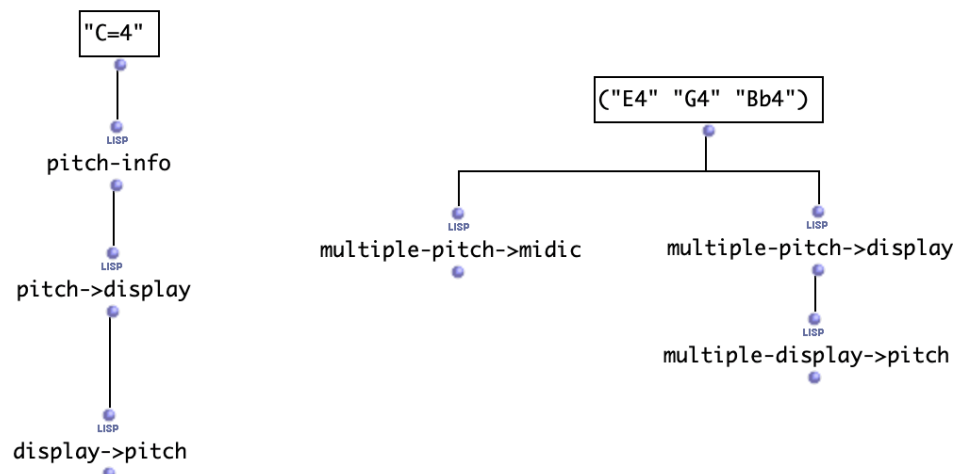**Please Note: This step is essential for the library's core functions to work correctly!**

This setup is required for the correct display of accidentals within a tonal framework, addressing a limitation in the native OpenMusic environment.

The algorithm for this component was developed by composer ©Jialin Liu (https://soundcloud.com/jialin-liu-457820276).

Chapter 2: Score Data Input/Output Module

pitch-receiver

LISP
pitch-info

LISP
pitch->midic

LISP
pitch->display

LISP
midic->enharmonic

=
mc->f

The patch shown above is a system for single-note input. To use it, select a pitch in the `pitch-receiver` object and evaluate the connected `chord-seq` to produce the output. Its primary function is to serve as a data entry tool for musical information.

"C=4"

LISP
pitch-info

LISP
pitch->display

LISP
display->pitch

("E4" "G4" "Bb4")

LISP
multiple-pitch->midic

LISP
multiple-pitch->display

LISP
multiple-display->pitch

The patch shown above is a system for outputting musical data. It is the main output utility for the `part-writing` module discussed in Chapter 6.

Please Note: To create a function box from this library in your patch, you can either type `om::harm` `function-name` directly into the workspace (replacing `function-name` with the specific function's name), or you can right-click on the patch background and select it from the **Libraries > OM-Harmonist** menu.
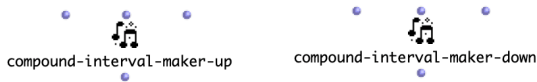
Chapter 3: Interval Module

This chapter covers the main features of the interval module. The best way to learn is by doing, so please recreate the tutorial patches shown in the images that follow. You'll find them to be very intuitive and easy to get started with!

1. construct the simple interval
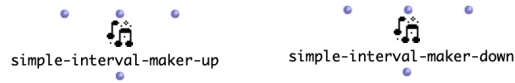
"d" = diminished, "m" = minor,
"p" = perfect, "M" = Major, "a" = augmented

simple-interval-maker-up   simple-interval-maker-down

2. construct the compound interval

interval-maker

compound-interval-maker-up   compound-interval-maker-down

3. construct the interval (including simple and compound)

interval-maker-up   interval-maker-down

The patch shown in the figure above is a utility for chord construction, one of the auxiliary functions provided in the OM-Harmonist library.

1. interval identification within two octaves

simple-interval-identify

interval-identify

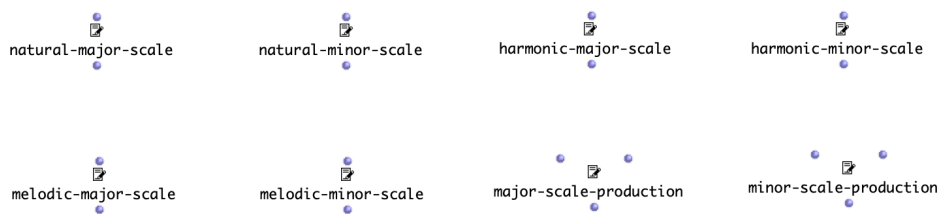simple-interval-identify-2

interval-identify-2
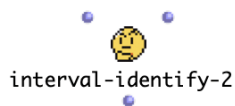
This patch is designed to identify the quality of chords (e.g., major, minor, etc.) within a two-octave range. To access detailed documentation for any function, select its corresponding box in the patch and press the **D** key on your keyboard.

Chapter 4: Tonality Module

This chapter covers the main features of the tonality module. We will begin with the `scale-producer` patch, a utility designed to generate a wide variety of musical scales. This function is primarily intended as a pedagogical tool for students learning about scale construction.
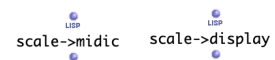
1. minor and major scale

natural-major-scale   natural-minor-scale   harmonic-major-scale   harmonic-minor-scale

melodic-major-scale   melodic-minor-scale   major-scale-production   minor-scale-production

2. Mode of limited transposition   3. Scale production midic

scale->midic   scale->display

mode-1   mode-2   scale-production-midic

The second patch shown below is a practical utility for melodic key analysis. It outputs a list of possible keys, ranked by probability based on the input melody. Please note that while this tool does not yet handle modulations, it correctly processes most chromatic events, such as secondary chords.
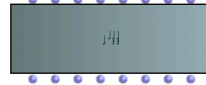
1. Determine the tonality to which the melody belongs

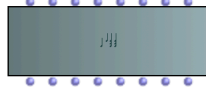("D4" "E4" "C#4" "D4" "F#4" "G4" "E4" "B4" "A#4" "A#4" "B4" "C#5" "D5" "B4" "G4" "F#4" "C#5" "B4")

LISP
multiple-pitch->midic

LISP
analyze-melody-key

("D#4" "F#4" "G#4" "F#4" "D#5" "C#5" "A#4" "B4" "G#4" "E5" "C#5" "A#4" "F#5" "F#5" "D#5" "E5" "D#5" "C#5" "A#4" "B4")

LISP
multiple-pitch->midic

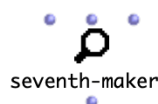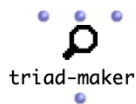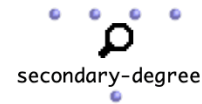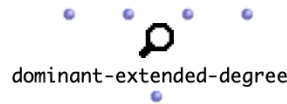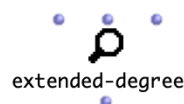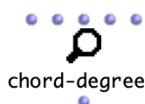LISP
analyze-melody-key

Chapter 5: Chord Module

In this chapter, we will explore three main patches that cover most of this module's functions. Please follow the images to build them and experiment on your own. As before, they are very intuitive to use!

1. chord-maker (don't need tonaliy information)

triad-maker   seventh-maker

2. Generate target chords using tonality and degree information

chord-degree   extended-degree   dominant-extended-degree   secondary-degree

3. Generate all common chords in the target tonality

generate-all-chords   generate-common-used-chords

The first patch shown above is the `chord-maker`. This utility is designed to quickly generate a target chord based on different inputs and conditions.

```
"C#4"   "B"   "minor"                "C#4"   "C"   "Major"


         LISP                                 LISP
    search-soprano                        search-soprano
```
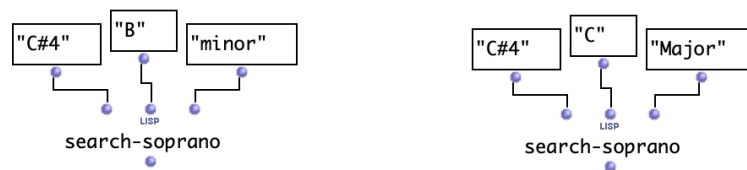
2. Analyze the degree of a chord belonging to this tonality, and what chord is it
(chord can only contain a maximum of 4 notes, and it could be any arrangement!)

```
        LISP
  chord-degree-analyze
```

The second patch is the main utility for chord analysis. The details of its functionality is

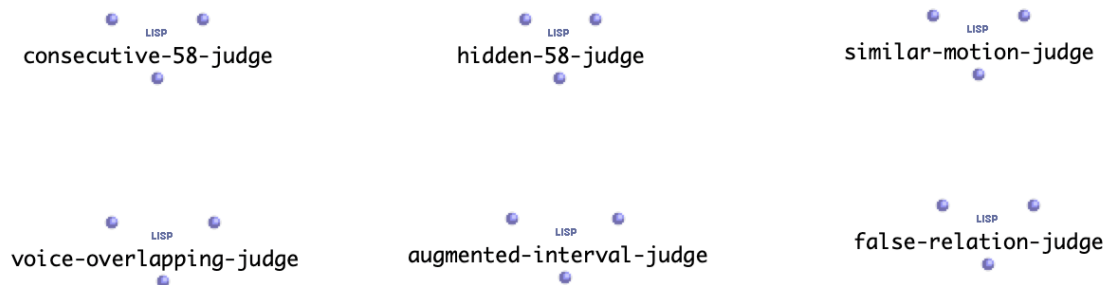provided in the text comments within the patch itself (shown above).

1. "SATB" voice leading

```
       LISP                    LISP                        LISP
  triad-arrangement       seventh-arrangement         chord-arrangement
```

The third patch shown above is a `chord-voicer`, a tool designed to generate various

voicings for a given chord in a standard four-part harmony setting. This is a

foundational function for the library's automatic part-writing capabilities.

Chapter 6: Part-Writing Module

This chapter introduces two primary utility patches that form the core of the library's four-part writing and analysis capabilities.

1. Determine if two chord progressions violate specific taboo

consecutive-58-judge   hidden-58-judge   similar-motion-judge

voice-overlapping-judge   augmented-interval-judge   false-relation-judge

2. Determine if two chords are well connected

smooth?   chord-connection-judge   valid?

Figure 1: The `voice-leading-checker` Patch

- **Function:** This patch evaluates the voice leading between two consecutive chords to determine if the connection complies with specific rules.
- **Output:** All functions within this patch return a boolean value: `t` (true) for a correct connection, or `nil` (false) if voice-leading errors are detected.
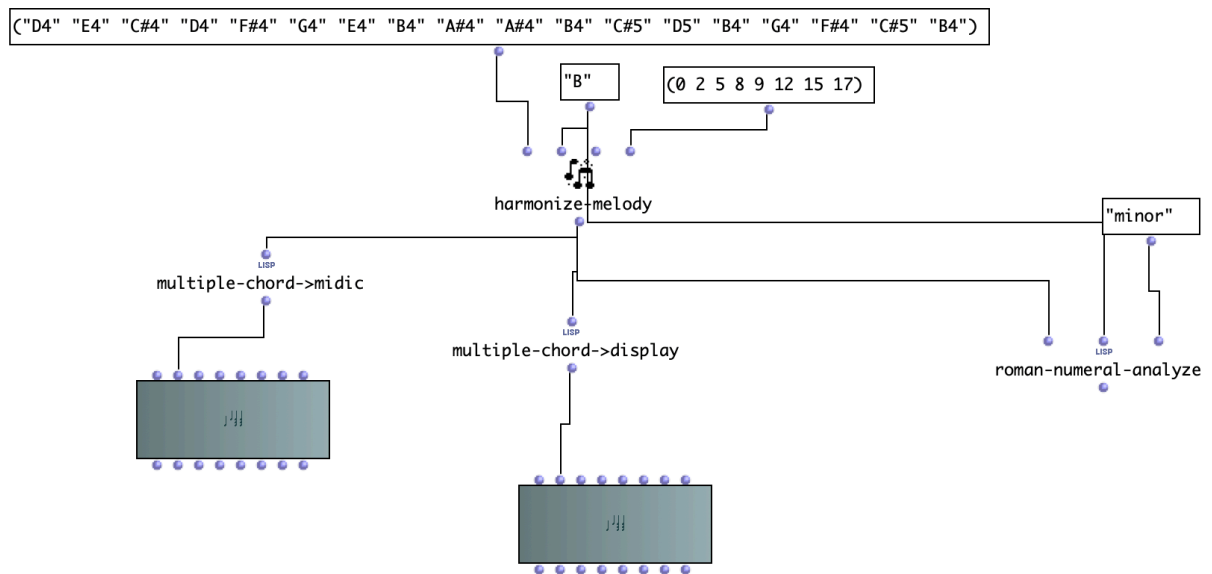
Figure 2: The `four-part-harmonizer` Patch

- **Function:** This patch automatically harmonizes a given soprano melody, generating a complete four-part chorale-style setting.

- **Output & Current Limitations:** The patch outputs a *single*, harmonically correct voicing. However, this is not guaranteed to be the most musically optimal solution. Due to its current architecture, the algorithm halts and returns the *first* valid result it finds.

- **Future Development:** An updated version is planned that will find and output *all* possible solutions, allowing the user to select the best option. Stay tuned!