PyOracle + i-score

Jaime Arias

Laboratoire Bordelais de Recherche en Informatique
Inria - Bordeaux SudOuest

2016-09-19

Contents

1 Introduction

PyOracle[1] is a graphical interface developed in *Max/MSP* for the improvisation in real-time using the *Python* implementation of **Factor Oracle**.

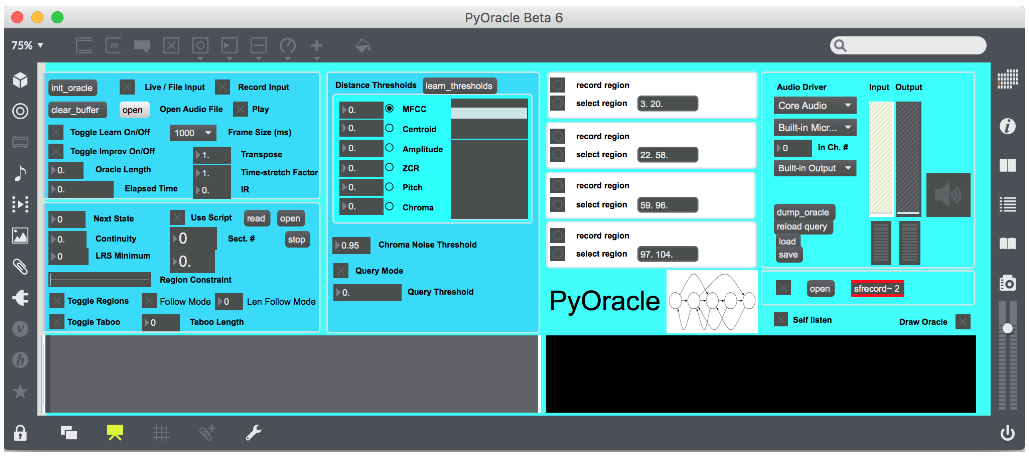


Figure 1: Main window of PyOracle

For more information, please refer to the following paper:

- Surges, G., & Dubnov, S. (2013). Feature Selection and Composition Using PyOracle. In AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.

1.1 Installation

In order to use PyOracle, we need to install the following dependencies:

- Max/MSP[2]: As we said before, PyOracle is a Max patch allowing to play and capture in real-time the audio that will be processed and generated by the python module of factor oracle.

- Py/pyext[3]: The interface between Max/MSP and the Python implementation of factor oracle is carried out by the Max package Py.

Currently, PyOracle supports the communication with the inter-media sequencer i-score[4] via the *minuit* protocol. For that, we need to install the following Max package:

- Jamoma[5]: Max package that allows to use the protocol minuit.

[1]https://github.com/himito/PyOracle_I-score
[2]https://cycling74.com/
[3]https://github.com/himito/py-mac
[4]https://github.com/OSSIA/i-score
[5]https://github.com/himito/Jamoma

2 Use

The use of PyOracle is divided into two parts: *training* and *improvisation*. The training can be done from an *audio file* or in real-time using an *input device* (e.g., microphone) of the computer.

2.1 Training from an audio file

To build an oracle from an audio file, we proceed as follows:

1. Create an empty factor oracle by clicking on `init_oracle`.

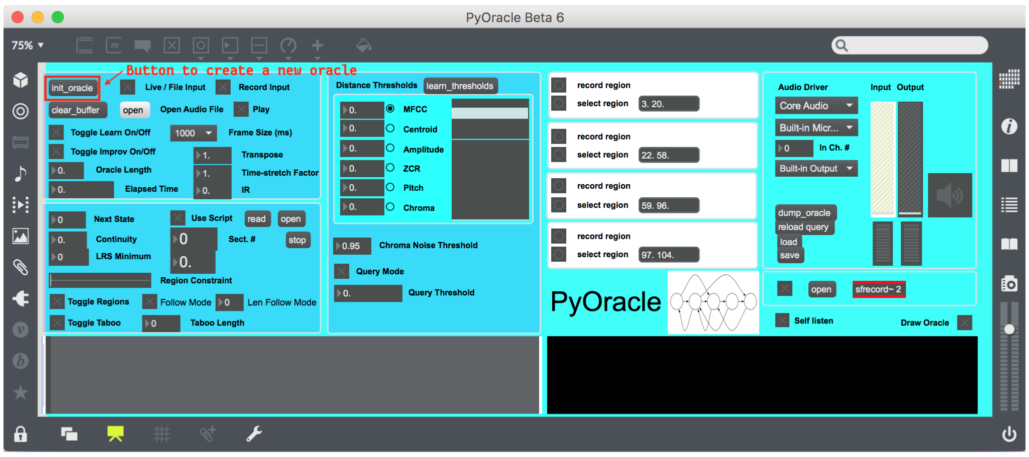


Figure 2: Button to create a new oracle.

2. Clear the audio buffer by clicking on `clear_buffer`.

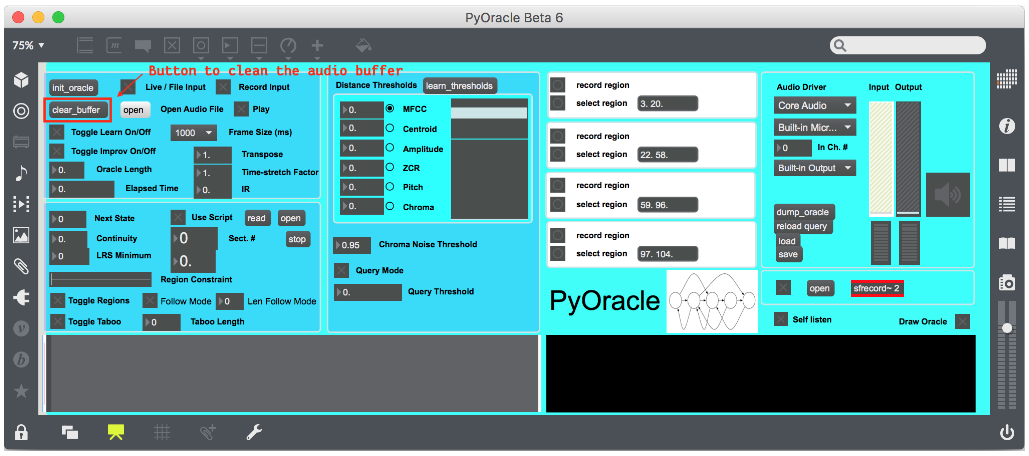


Figure 3: Button to clear the audio buffer.

3. Enable the audio and set the volume of the input (see Figure 4).

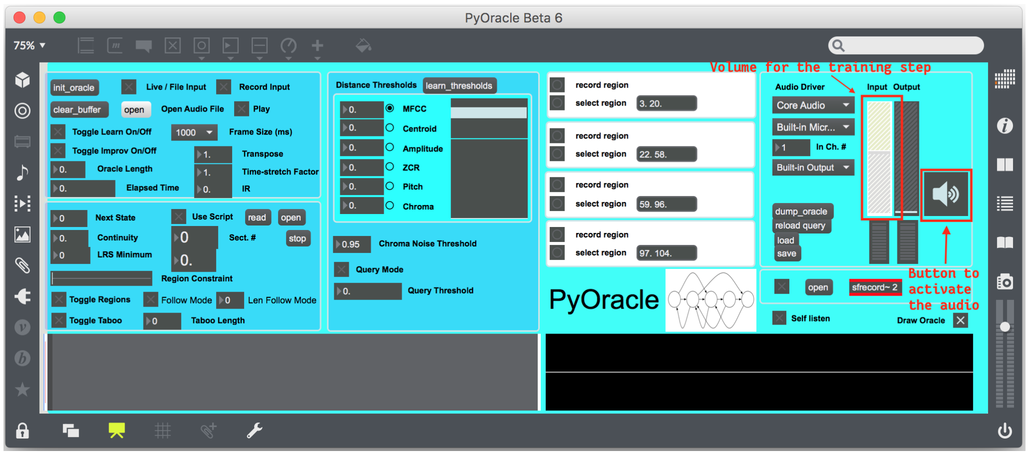


Figure 4: Configuration of the input audio.

4. Load the audio file by clicking on `open` (see Figure 5).

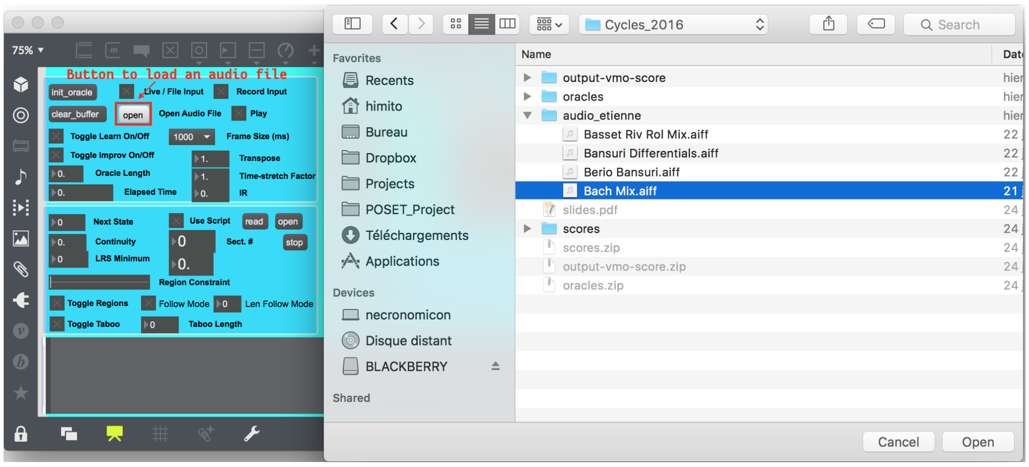


Figure 5: Button to load an audio file.

5. Click on `play` to start to play the audio file (see Figure 6). This button starts automatically the learning step (i.e., checkbox `Toggle Learn On/Off`).

6. Stop the learning process by clicking on `Toggle Learn On/Off` once you think that there is enough information for improvising. Also, stop the playing of the audio file.

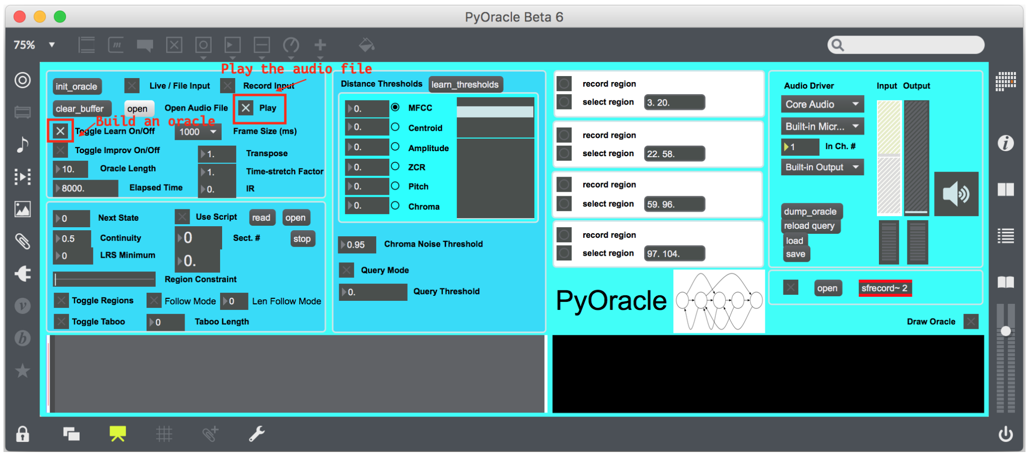


Figure 6: Button to play the audio file and the button to build an oracle.

7. Find the best threshold in order to build the "best" oracle for each audio feature by clicking on `learn_thresholds`. Briefly, during the construction process, an important parameter is the *threshold*. This parameter allows to define when two audio frames are similar.

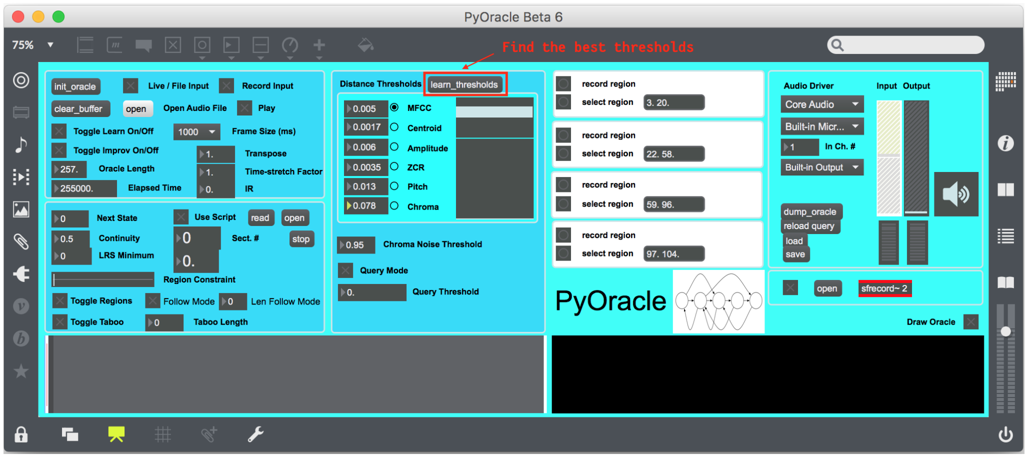


Figure 7: Button to find the best threshold for each audio feature.

8. Finally, repeat all the above steps with the new thresholds in order to build the best factor oracle for each feature.

2.2 Training from an input device

To build an oracle in real-time, we proceed as follows:

1. Follow the steps 1, 2 and 3 of Section 2.1.

2. Enable the live recording by clicking on `Live/File Input` and start the audio recording by clicking on `Record Input`. The latter starts automatically the learning step (i.e., checkbox `Toggle Learn On/Off`).

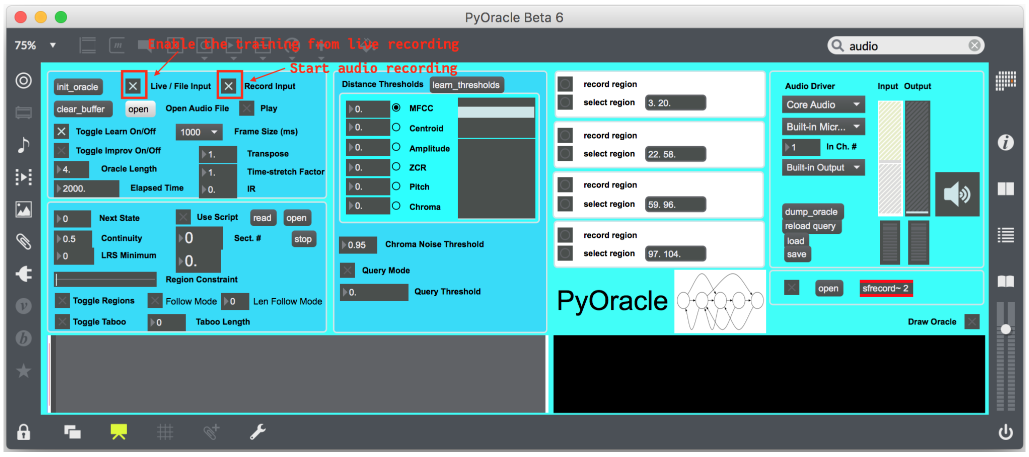


Figure 8: Button to enable live recording and start audio recording.

3. Stop the learning process by clicking on `Toggle Learn On/Off` once you think that there is enough information for improvising. Also, stop the audio recording.

4. Finally, follow the steps 7 and 8 of Section 2.1.

2.3 Improvisation with a trained oracle

Once we have built an oracle, we can use it for improvising. To do that, we proceed as follows:

1. Set the volume of the improvisation by using the slider `Output`.

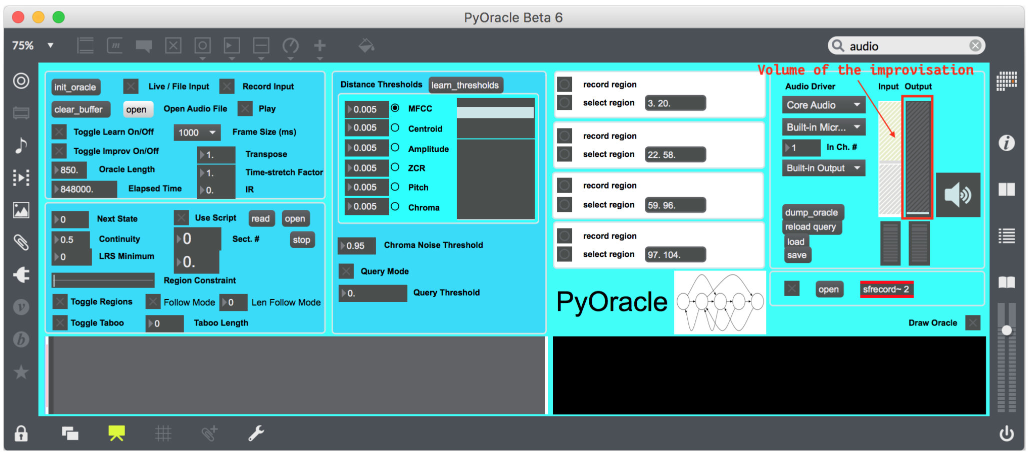


Figure 9: Slider to control the volume of the improvisation.

2. Select one of the available oracles for improvisation. Briefly, PyOracle builds an oracle for each audio feature.

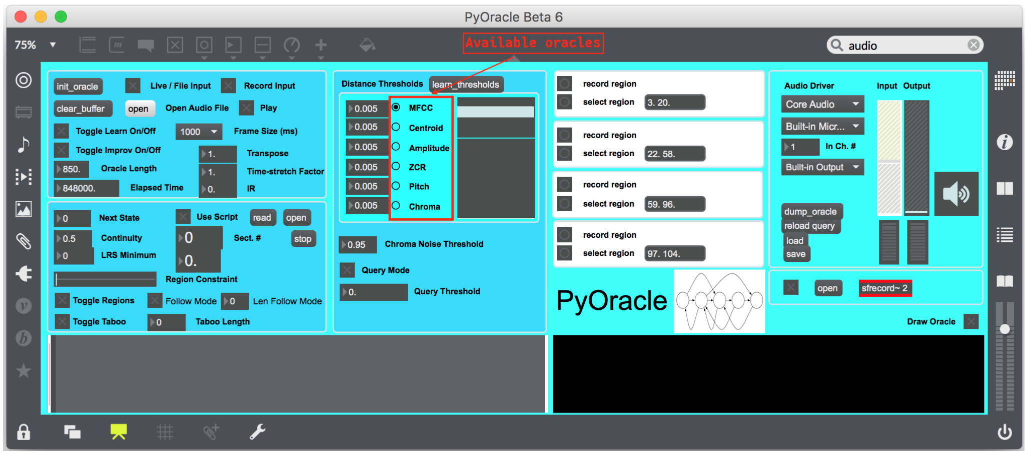


Figure 10: Available oracles for improvisation.

3. Start the improvisation by clicking on `Toggle Improv On/Off`.

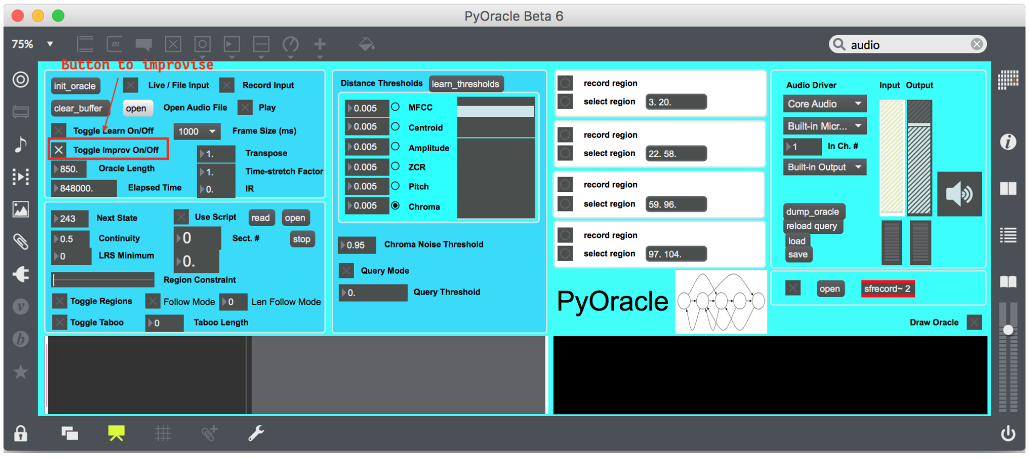


Figure 11: Button to start the improvisation.

2.3.1 Parameters of the improvisation

We can control the improvisation by setting the following parameters:

1. Continuity

   It is a value p between 0 and 1 controlling the navigation of the oracle structure. Forward linear movement (i.e., with probability p) corresponds to playback of the original audio, while jumping forward or backward along a suffix link (i.e, with probability $1 - p$) produce new variations on the original material.

2. Minimum LRS

   Setting a minimum LRS allows jumps to occur only when a certain length of context (i.e., the longest repeated suffix) is shared between states. This parameter allows the user to control the smoothness of the jumps, with shorter contexts producing less

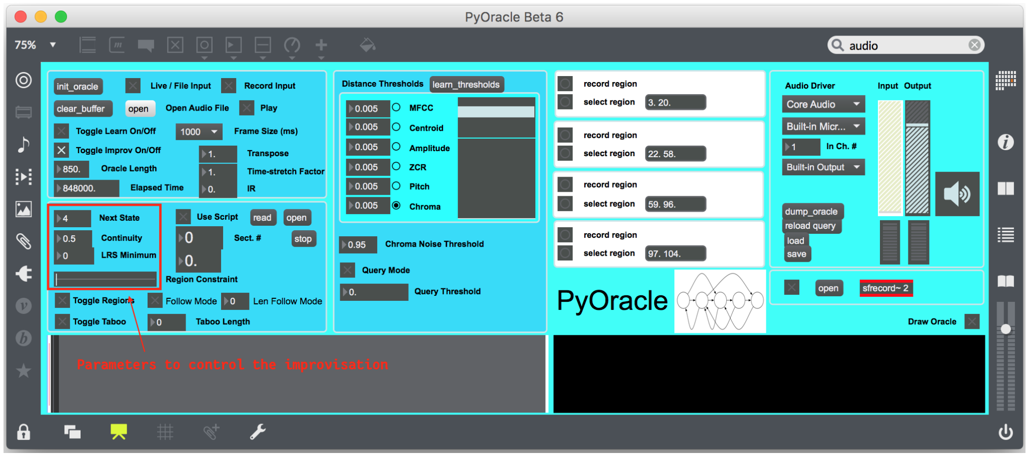


Figure 12: Parameters to control the improvisation.

smooth jumps. If a sufficiently long shared context is not found, the system defaults to a linear continuation to the next frame.

3. Regions

   The oracle navigation can also be limited to only certain areas of the data structure, that is, restricting the musical generation to specific materials. Transitions or jumps which would cause the navigation to fall outside this region are ignored. To activate this options is necessary to click on the checkbox `Toggle Regions`.

2.4 Extra Functionalities

In the following, some extra functionalities of PyOracle are described.

- It is possible to draw the current oracle by clicking on the checkbox `Draw Oracle`.

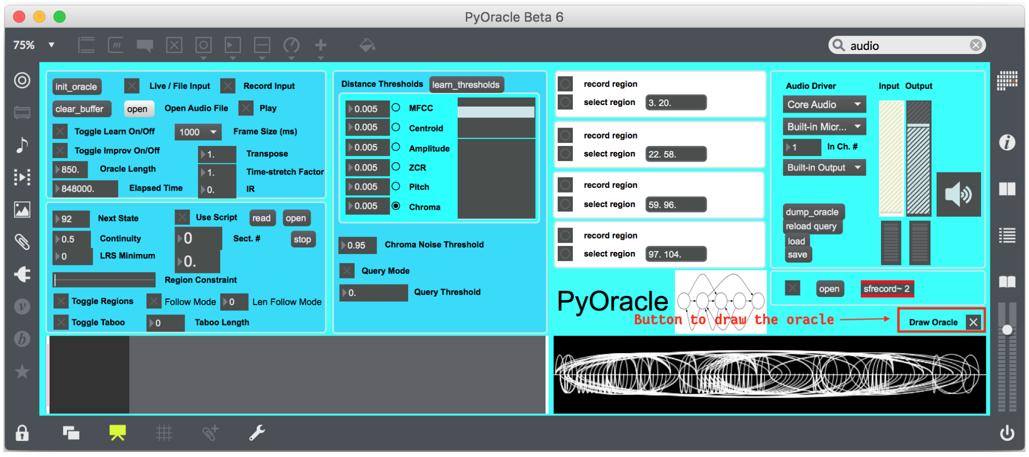


Figure 13: Button to draw the current oracle.

- It is possible to save all the created oracles by clicking on the button `save`. The oracles will be saved into the selected folder.

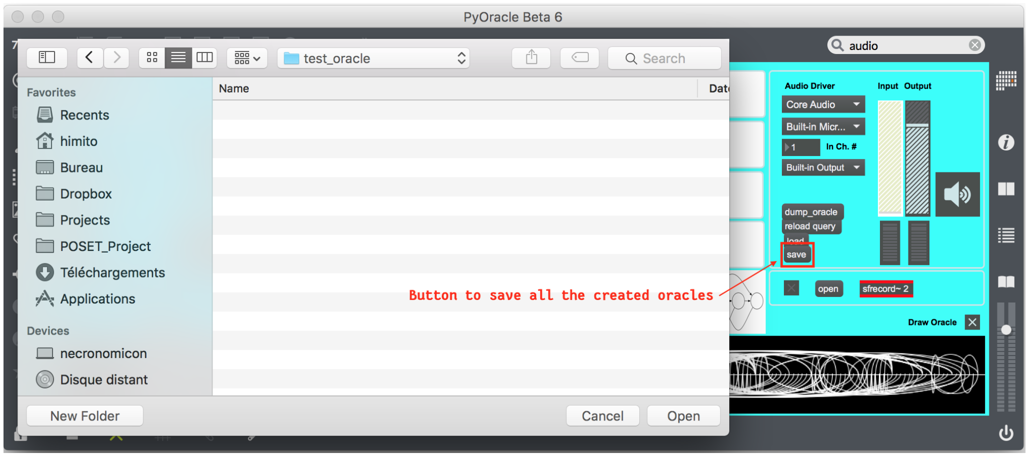


Figure 14: Button to save all the current oracles.

- It is possible to load oracles that have been saved before by clicking on the button `load`. It is only necessary to select the folder in which the oracles were saved.

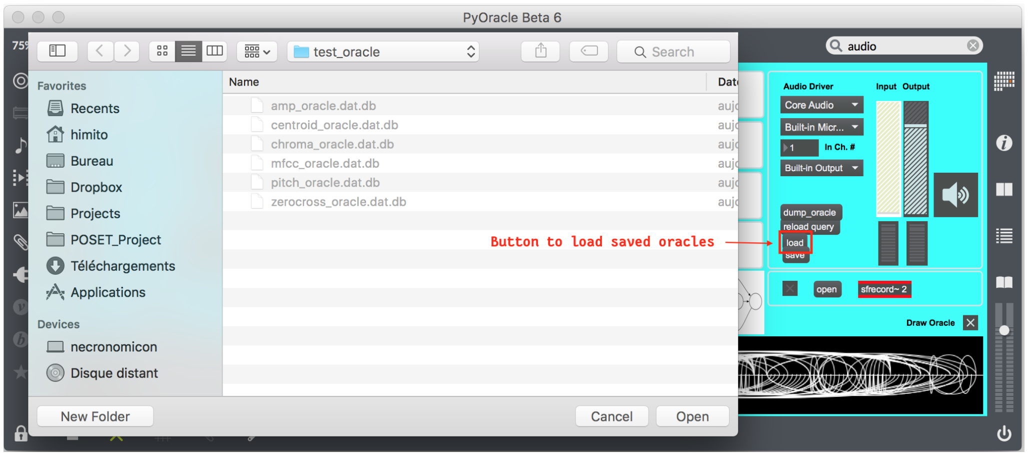


Figure 15: Button to load oracles.

3 Integration with i-score

i-score[6] is a free open-source inter-media sequencer that allows to orchestrate processes and devices by means of OSC messages.

Currently, PyOracle supports the communication with i-score via the protocol minuit, allowing the latter to discover automatically the parameters offered by the former. To do that, we then only need to add PyOracle as a minuit device in i-score as follow.

1. We add a new device by clicking on the button + found at the bottom left corner of i-score.

[6] `http://i-score.org/`

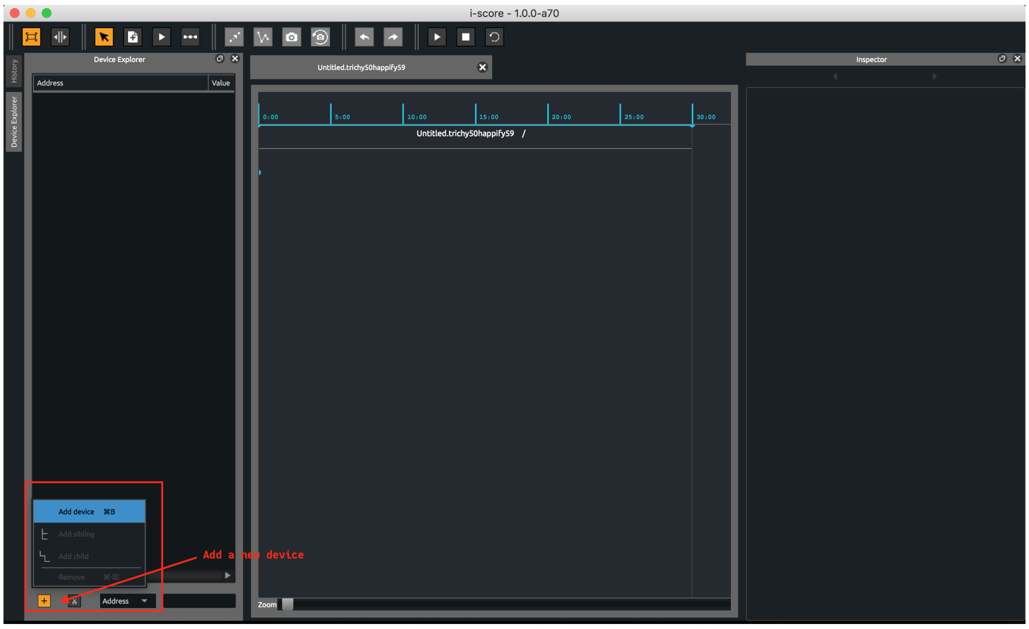
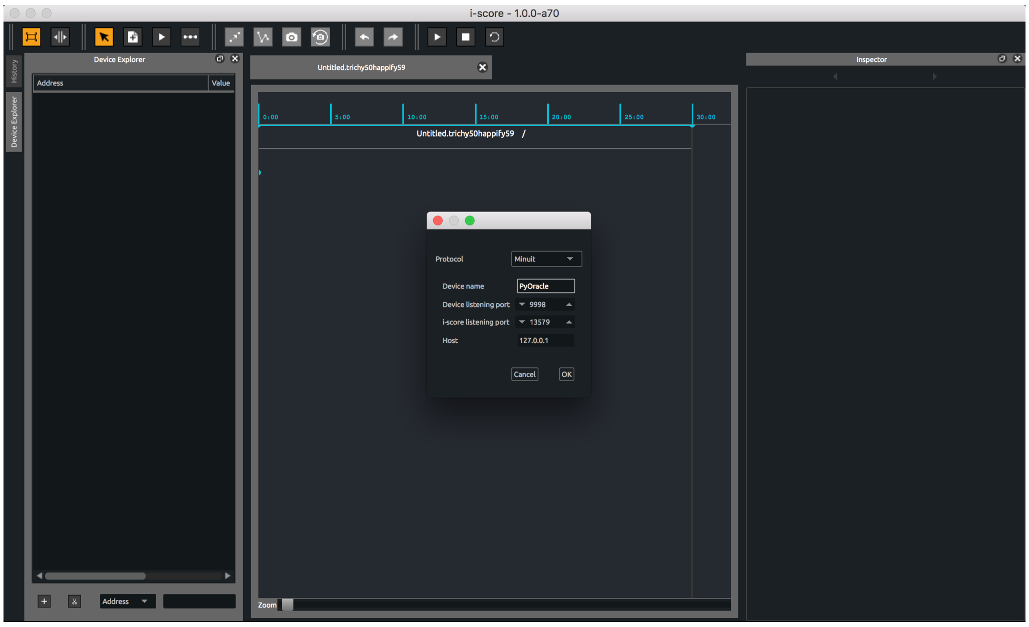


Figure 16: Button to add a new device in i-score.

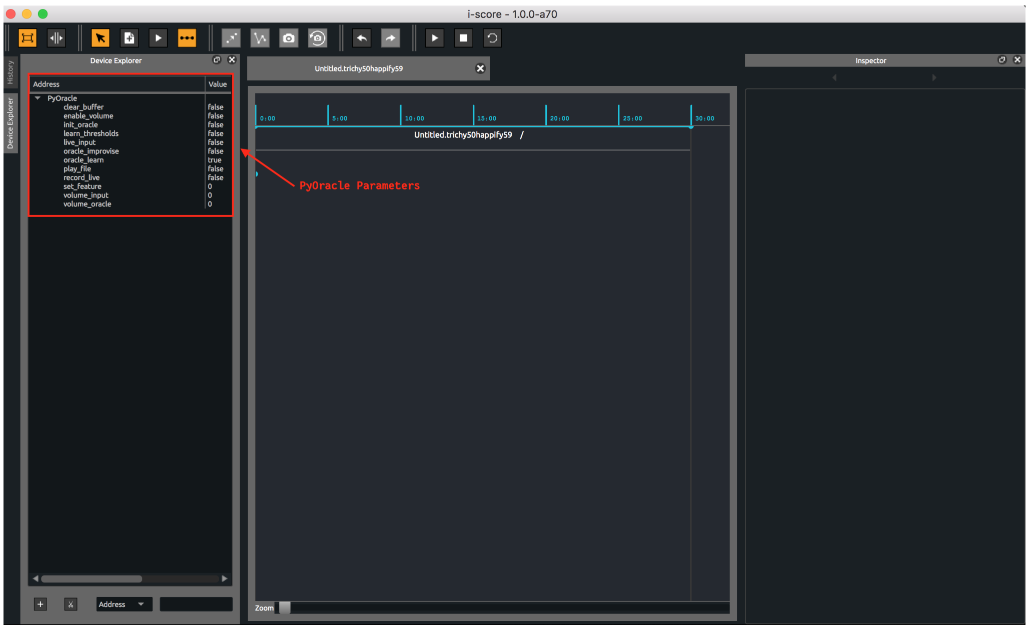
2. Add the PyOracle device by selecting `Minuit` as protocol and typing `PyOracle` in the name field. The other values remain the same.



Protocol	Minuit
Device name	PyOracle
Device listening port	9998
i-score listening port	13579
Host	127.0.0.1

Figure 17: Configuration of the PyOracle device.

3. Now, you can write i-score scenarios that control PyOracle in real-time.



Address	Value
PyOracle	
clear_buffer	false
enable_volume	false
init_oracle	false
learn_thresholds	false
live_input	false
oracle_improvise	false
oracle_learn	true
play_file	false
record_live	false
set_feature	0
volume_input	0
volume_oracle	0

Figure 18: PyOracle as an i-score device.