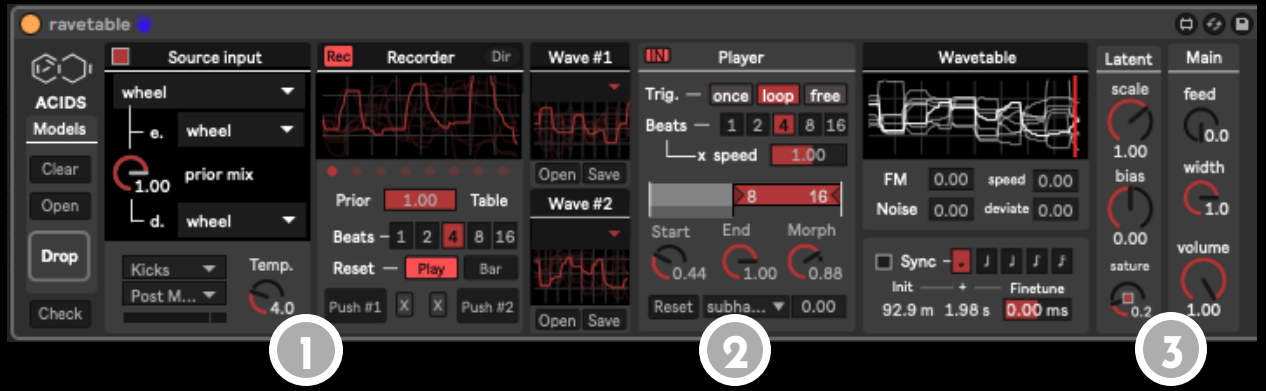
Ravetable

Joining traditional synthesis and neural audio

Philippe Esling, David Genova
{esling, genova}@ircam.fr

General interface

General interface of the ravetable prototype device



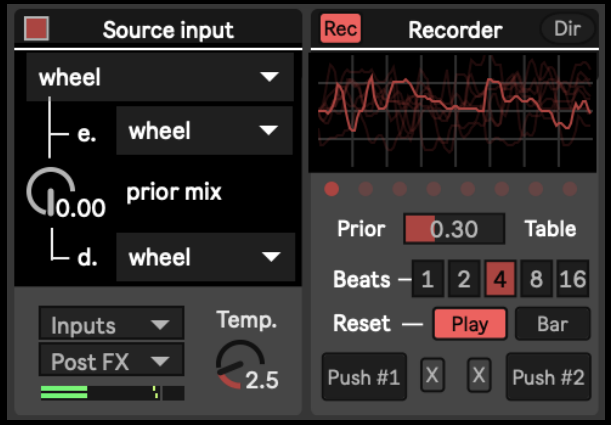
There are 3 major components to understand the use of Ravetable

1. **Source section:** creates new latent series with a generative model
2. **Player section:** allows to play those series similar to wavetable synthesis
3. **Effects and global:** modify the series and global behavior

Source section (creating tables)

General workflow for using the model and recording tables

The first section of the device aims to help in *creating new ravetables to play with*.
To do so, you can use input audio or prior models as usual neural audio synthesis.



Source section

Defines the *information generation stage*, which model are used and how they behave with a given input.

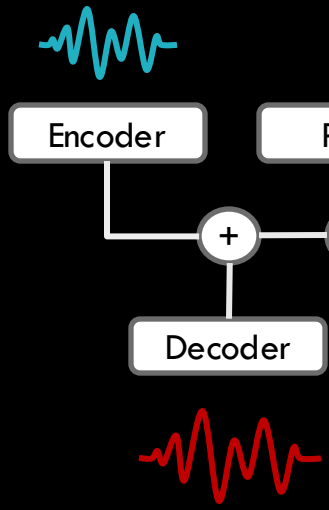
Recording section

Defines the *behavior of real-time recording*, which continuously fills a circular buffer with the current series.

Source input (latent model)

General workflow for creating new tables

The source section relies on an encoding / decoding generative model which will produce latent series that are then synthesized and heard.



Change all models

Change encoder and prior models

Change decoder model

Prior temperature

Defines the « entropy » of prior generation. Lower temperature imply simpler and more predictible generation

Note: the decoder is the same that will be used to play the latent tables

Audio sidechain input (feeds the encoder)

Recording latent tables

Activate recorder

Directory to find the saved tables

Current recording buffer content

Current output / record what is routed **to the recorder and output**

A value of 0.0 means that we only hear the source, while 1.0 is only the tables

Quantized length. The recording buffer is beat-synchronized

Beats defines the length of the buffer in number of beats

Resets synchronizes the reset position of the buffer

Push to current tables allows to transfer the current recording buffer to table #1 or #2

The crosses allow to clear each respective table

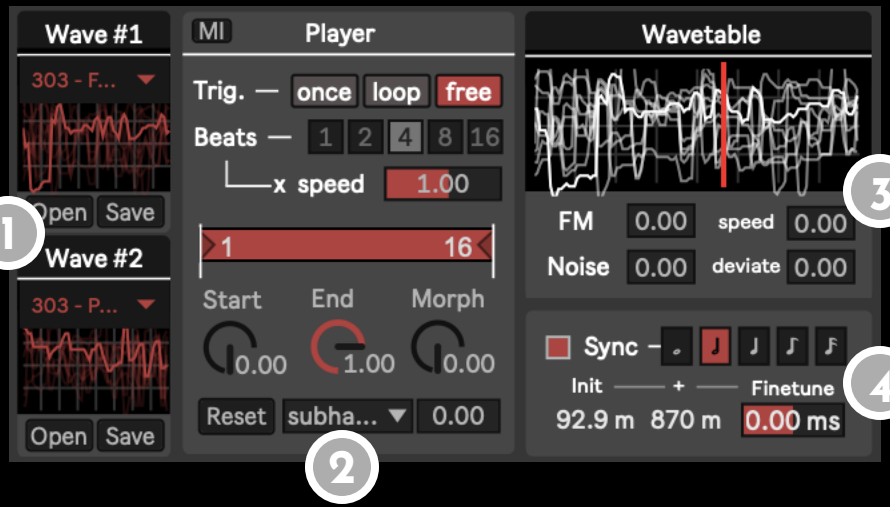
Important note on the recorder: The recorder works constantly on the **current audio output**, which means that even when you start meddling with existing tables and add effects or playhead morphing, you can also push those into new ravetables as new starting points !

Playing with ravetables

General workflow for playing and controling ravetables

The second section of the device aims to *play existing ravetables* and change their properties. This allows to refine the sound and its control

Current tables



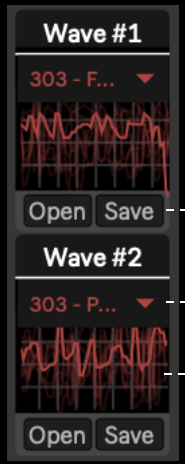
Playhead morpher

Audio post-sync

Tables player

Playing with ravetables

Ravetables loader



This section defines what tables are currently loaded for the player

Tables can be saved and loaded from the disk

Save / Open ravetable (saved as 16-channels .wav)

Quick load from the current tables folder

Display of current table

Playing with ravetables

Ravetable player

The player behave similarly to wavetable synthesis by reading, looping and morphing tables.

MIDI Activity indicator

Player controls

Trigger behavior. Defines how the player reacts to incoming MIDI

- **Once** – plays the table once at each MIDI onset
- **Loop** – restarts the table at each MIDI onset and then loop
- **Free** – does not take MIDI into account

Playback speed. Defines the beat-sync speed of the player

Defined as a number of **beats** and then multiplied by the **speed**

Quantized crop. Crop the tables with a 16th grid

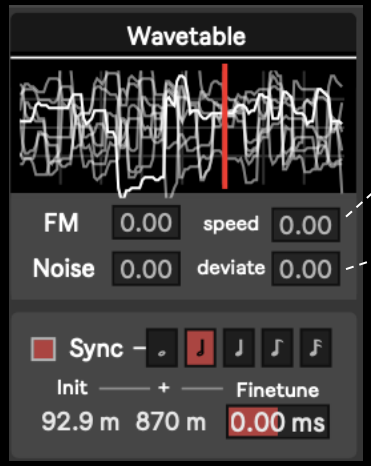
Table controls.

- **Start / End** – Defines the cropping points of both tables
- **Morph** – Morph between the two tables (0.0 = table #1 / 1.0 = table #2)

- **Reset** – Restart the playheads all at zeros
- **List** – Apply different types of variations to the playhead
- **Number** – Strength of application of the variation

Playing with ravetables

Ravetable display and effects



Playhead morpher

FM player – Adds frequency modulation to the playhead
Speed changes the speed variation across different dimensions

Noise injection – Adds noise to the latent playhead
Deviate changes the variation across different dimensions

Audio (post-) synchronisation

This section allows to ensure synchronization of the final output to the overall track by *enforcing a fixed delay*

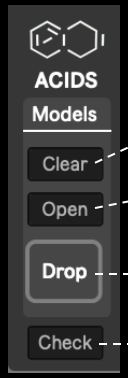
Activate / deactivate

Infered tempo-based values

Grid-based fixed delay introduced

Finetune delay in miliseconds

General sections



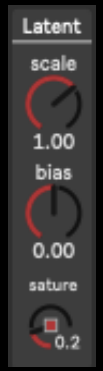
Model handling section

Clear the list of available models

Open dialog to set a specific folder of models

Drop zone for a folder of models (equivalent to open)

Open the Max console



Latent effects

Latent scale

Latent bias

Latent saturation

General audio section

Audio feedback (route output to the encoder)

Stereo width (latent noise injection)

General audio volume