SkAT Studio

User Manual

Table of contents

1 Introduction

This document presents the SkAT Studio software and its use. This software is used for sound design, and was developed in the scope of the SkAT-VG project.

1.1 The SkAT-VG project

SkAT-VG (**Sk**etching **A**udio **T**echnologies using **V**ocalizations and **G**estures) is a FET-Open European project which aims at developing sketching tools for sound design. Based on studies showing that people describe sounds better with vocal imitation and/or gesture than with words, this project explores the use of voice and gesture for the sketching process.

1.2 SkAT Studio

SkAT Studio is developed as a demonstrator and experimental tool for the technologies developed in the scope of the SkAT-VG project. Therefore, the software is thought and conceived as modular as possible, in order to allow the control of very diverse concepts with very diverse parameters extracted from voice and/or gesture.

In order to make the integration of new tools from the project partners as easy as possible, it has been decided to develop SkAT Studio with Max/MSP.

2 Installation

2.1 System requirements

SkAT Studio is running on a PC using the Windows operating system, or on a Mac using the MacOS operating system.

Max/MSP 6 is required, along with the **FTM&Co** library. FTM&Co may be downloaded at this address: http://ftm.ircam.fr/index.php/Download. Be careful to install the library in the Max 6 folder.

2.2 Installation procedure

SkAT Studio can be downloaded at the following FTP address:

ftp://ftp.genesis.fr

The credentials to access the FTP server are:
Login: genesis-svg
Password: SkATStudio

Copy the whole folder on your computer, then add this folder to the Max/MSP path.

For Windows only: add **fftw3.dll** (*externals/WinPitch~/fftw-3.0.1-w32*) to your windows path.

3 User interface

3.1 Overview

The framework is composed by a main GUI which can host and link together a collection of loadable ***modules***, each one taking care of a specific operation in the global process. Several modules can be loaded simultaneously, and signal and/or control data can be routed at will among different modules using ***patchbays***.

In order to simplify the use and comprehension of the software, the overall workflow is thought as a five step process, described with examples on Figure 1:

1. Inputs: get the control data (voice and/or gesture)
2. Analysis: get descriptors from Inputs
3. Mapping: transform the descriptors into synthesis control data
4. Synthesis: produce new sounds
5. Outputs: play/record the resulting sounds.

Each step is materialized as a ***group***. Several modules may be loaded in the same group.

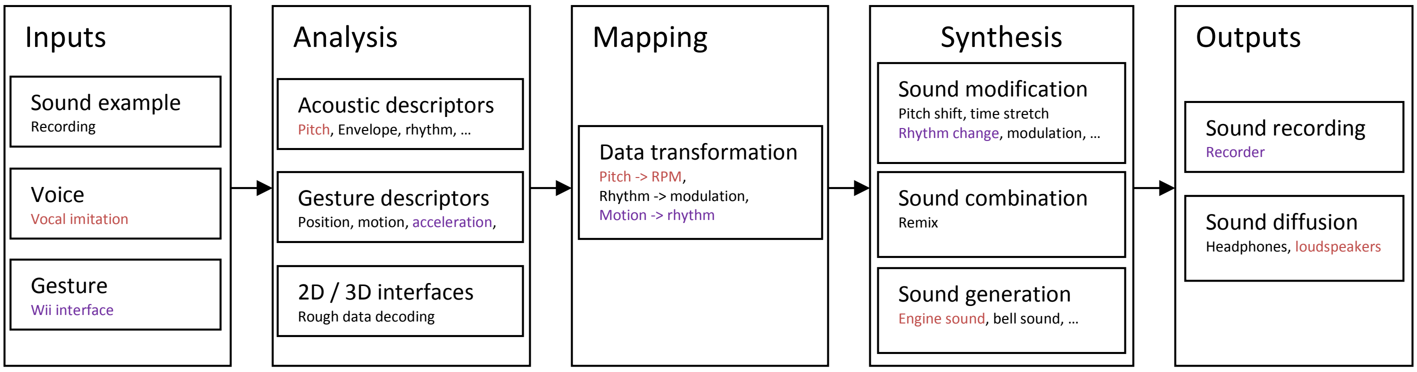


Figure 1: Overall SkAT Studio workflow. Two examples are highlighted in red (engine sound driven by voice) and in purple (rythm change driven by movement

As SkAT Studio is conceived to be very modular and customizable, data are routed manually between modules and between groups manually, using ***patchbays***.

The general GUI is presented on Figure 2. On this figure, each group is surrounded by an orange rectangle. In this example, 3 modules are loaded in the analysis group.

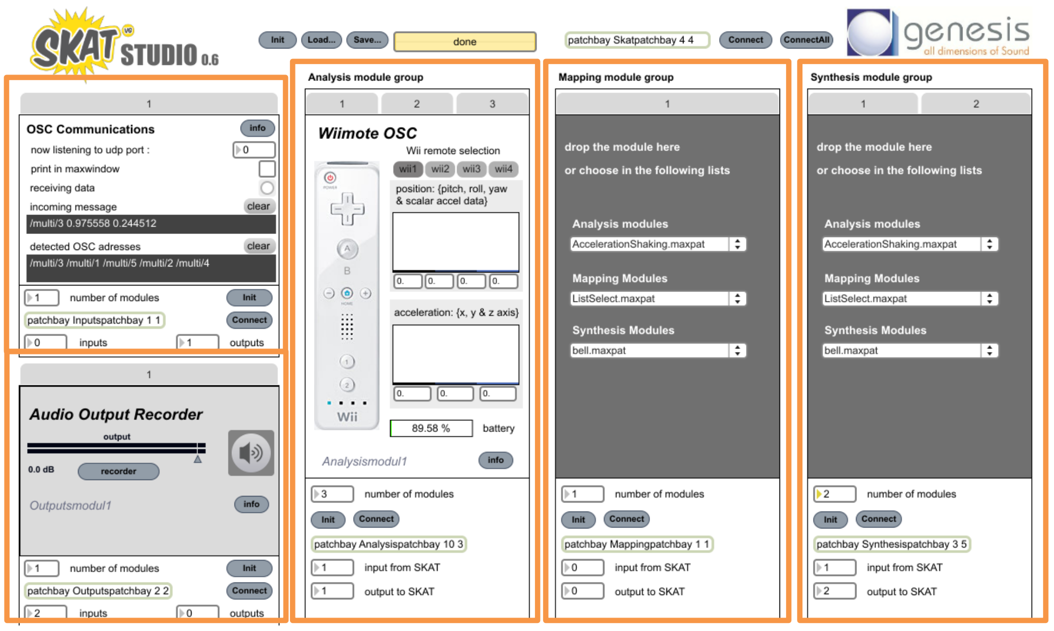


Figure 2: SkAT Studio user interface overview

3.2 Modules

Each module is realized as a Max patch and must adhere to a specific template. The SkAT Studio module template provides a common interface for back-end communication with the other parts of the framework and front-end integration into the main GUI. To comply with the template, modules must graphically fit a given area, and provide the following information: (highlighted in orange on Figure 3):

- Name of the module,
- Number of inputs and outputs,
- Input and output labels,
- Documentation (input/output data types, author, description of the underlying algorithms and so on).

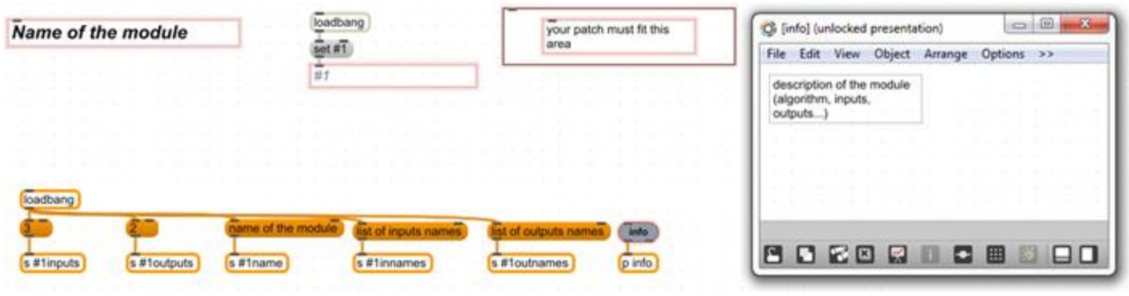
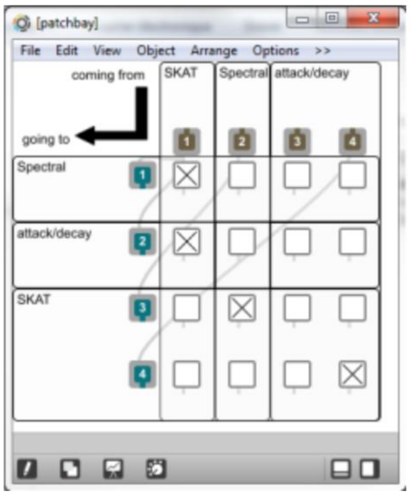


Figure 3: The SkAT-Studio module template

A wide variety of modules is already available in the SkAT Studio framework (see section 4 for more details), offering the basic building blocks for the composition of complex configurations. Future integration of new features is going to be an easy task, thanks to the capabilities of the Max patching environment and to the simple yet versatile module template.

3.3 Data routing

Data can be routed from any output to any inputs of modules. The communication is done using routing matrices called *patchbays*. A patchbay presents itself as a double entry table, as displayed on Figure 4, with all the module outputs listed on the top row and all the module inputs listed on the left column. A toggle matrix allows associating each output to one or more inputs, simply activating the appropriate toggles in the double entry table. To simplify the data routing process, modules are divided in *groups* reflecting the five stages of the process. Data is first routed among single modules inside the group, and then among different groups inside the main framework.



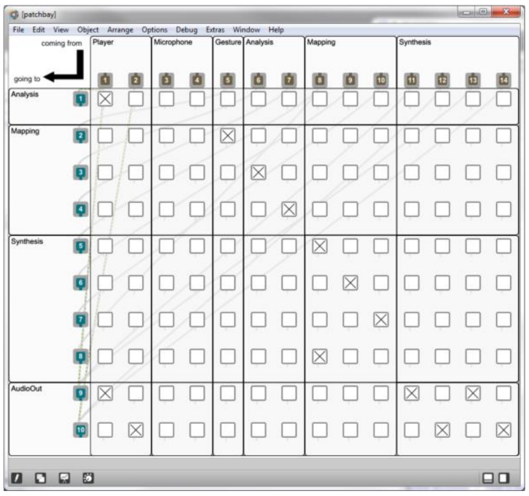


Figure 4: On the left, example of a patchbay for the analysis group. On the right, the global SkAT Studio patchbay.

3.4 Build a session

The procedure we describe thereafter is systematically illustrated with references to labels positioned on Figure 5.

To build a session, the successive steps are:

1. In each group, choose the number of modules.
2. This creates as many tabs as required.
3. In each tab, load the desired module. This may be done by drag and drop (from the Windows explorer) or by choosing the module in the list.
4. For each group, define the number of inputs and outputs. These group inputs/outputs represent the data that should be transmitted between groups.

   By default, the number of inputs (resp. outputs) of a group is the total number of inputs (resp. outputs) of all the modules in the group. Therefore, limiting these numbers limits also the number of signals and control data to route between groups (and the size of the corresponding patchbays.

5. Step 4 creates a patchbay, which will be used to route data in the group.
6. Click on connect to open the group patchbay and route data in each group.
7. Click on connect (top of the screen) to open the global patchbay and route data between groups.

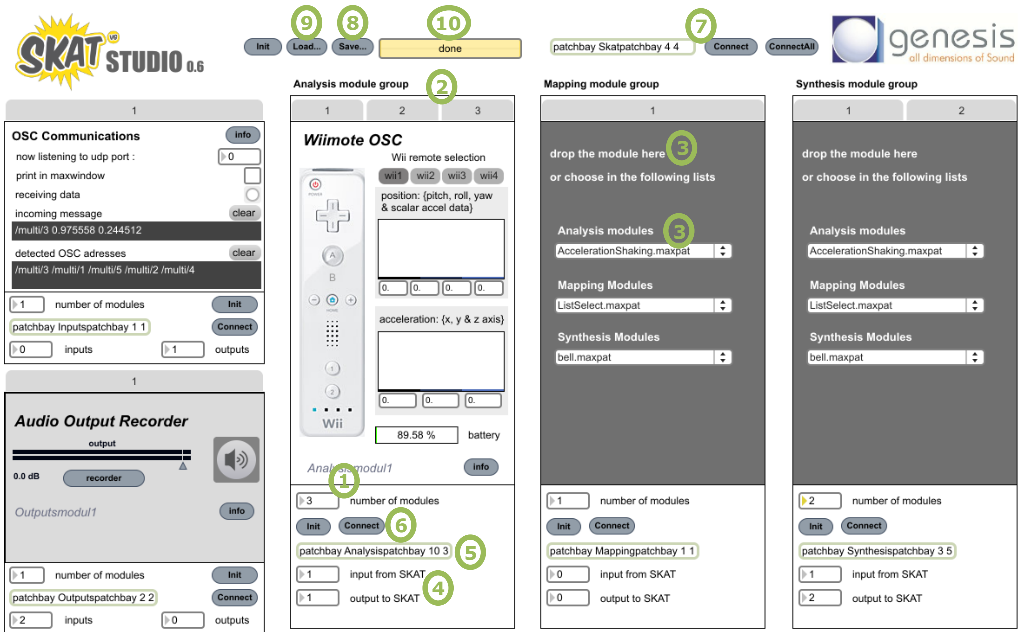


Figure 5: Illustration of the different steps of the use of SkAT Studio.

3.5 Save a session

Click on the "Save" button (labeled as 8 on Figure 5) to save a session. A dialog window allows choosing the saving folder and the name of the saving file.

A session is save as a .json file, which is the native Max/MSP save format.

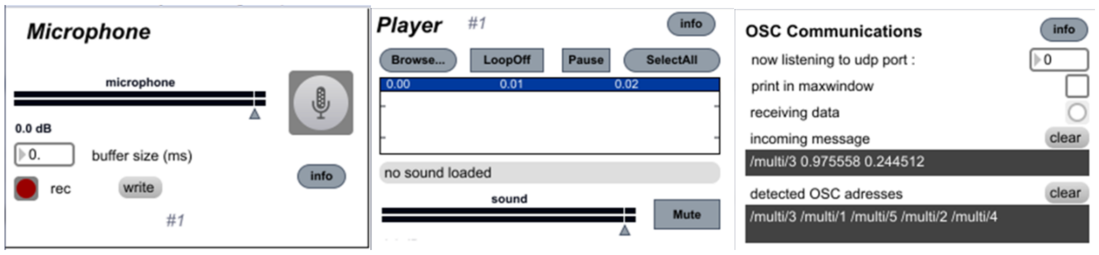
3.6 Load a session

Click on the "Load" button (labeled as 9 on Figure 5) to load a session, then choose the .json save file containing the session. The progress bar (10) shows the progression of the loading and prints "Done" when loading is finished.

4 Existing modules

4.1 Input modules

Microphone: Acquires vocal signals as real time control data.
Player: Plays back sound files, useful for offline vocal control.
OSC Communications: Acquires OSC data, such as sensor outputs in objects manipulated with hands, useful for live gestural control.



0.0 dB
0. buffer size (ms)

no sound loaded

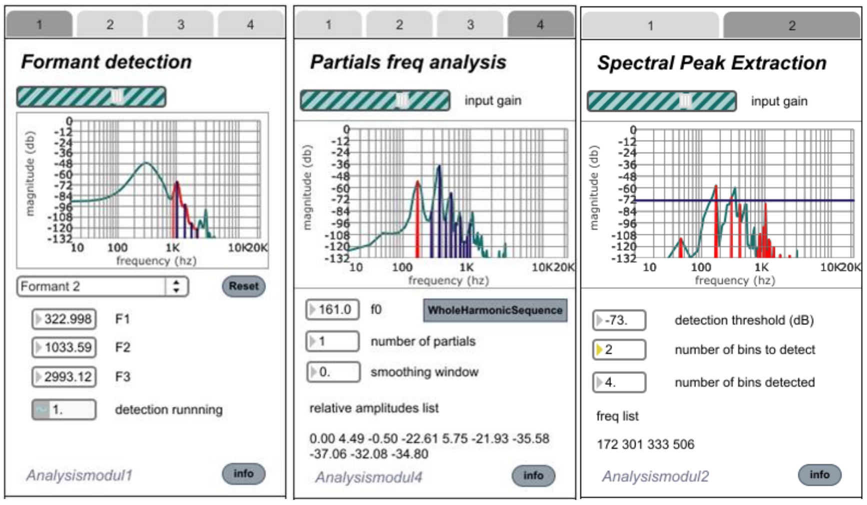
now listening to udp port : 0
print in maxwindow
receiving data
incoming message
/multi/3 0.975558 0.244512
detected OSC adresses
/multi/3 /multi/1 /multi/5 /multi/2 /multi/4

Figure 6: From left to right: The *Microphone*, *Player* and *OSC communications* input modules.

4.2 Analysis modules

4.2.1 Frequency analysis

Formant detection: Detects the center frequencies of voice formants.
Partials frequency analysis: Detects the partials of an input signal, namely its fundamental frequency and multiples.
Spectral peak extraction: Extracts spectral peaks from the input signal.



Formant 2
322.998 F1
1033.59 F2
2993.12 F3
1. detection runnning

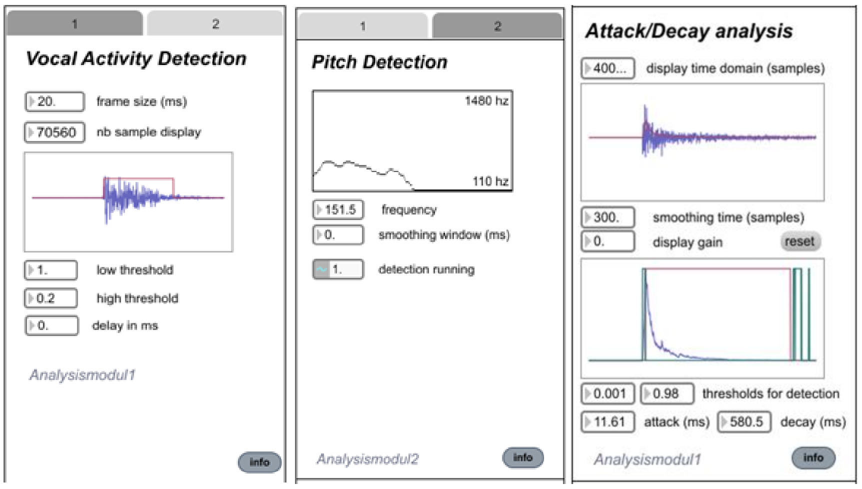
161.0 f0
1 number of partials
0. smoothing window
relative amplitudes list
0.00 4.49 -0.50 -22.61 5.75 -21.93 -35.58 -37.06 -32.08 -34.80

-73. detection threshold (dB)
2 number of bins to detect
4. number of bins detected
freq list
172 301 333 506

Figure 7: Frequency analysis: The *Formant detection*, *Partials frequency analysis* and *Spectral peak extraction* analysis modules.

4.2.2 Voice / music analysis

Vocal Activity Detection: Detects vocal activity in an audio signal, by means of calculations on signal energy in a specified temporal window.
Pitch detection: Detects the fundamental frequency of a signal. Based on the YIN algorithm[1].
Attack/Decay analysis: Detects attack, sustain, release and decay times of musical notes in the input signal. This module is particularly useful for music and singing voice.



Vocal Activity Detection
20. frame size (ms)
70560 nb sample display
1. low threshold
0.2 high threshold
0. delay in ms
Analysismodul1

Pitch Detection
151.5 frequency
0. smoothing window (ms)
1. detection running
Analysismodul2

Attack/Decay analysis
400... display time domain (samples)
300. smoothing time (samples)
0. display gain
0.001 0.98 thresholds for detection
11.61 attack (ms) 580.5 decay (ms)
Analysismodul1

Figure 8: Voice and music analysis: The *Vocal Activity detection*, *Pitch estimation* and *Attack/Decay analysis* analysis modules.

4.2.3 Gesture analysis

Gyro/Acc Control OSC, Multitouch Control OSC: Used with the *Control app*[2] on an android or IOS device, these modules detect either accelerometer data and gyroscope data(*Gyro/Acc*) or XY coordinates on the multitouch screen (*Multitouch*). In this last case, the coordinates can be offered in Cartesian or Polar form.
Wiimote OSC: This module has to be linked to the OSC input module of SkatStudio. Used with a *Wiimote*, it detects accelerometer data.
Shaking acceleration: Detects acceleration peaks. A threshold can be defined to avoid detection of small fluctuations.

[1] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *JASA*, 111(4):1917-1930, 2002
[2] http://charlie-roberts.com/Control/

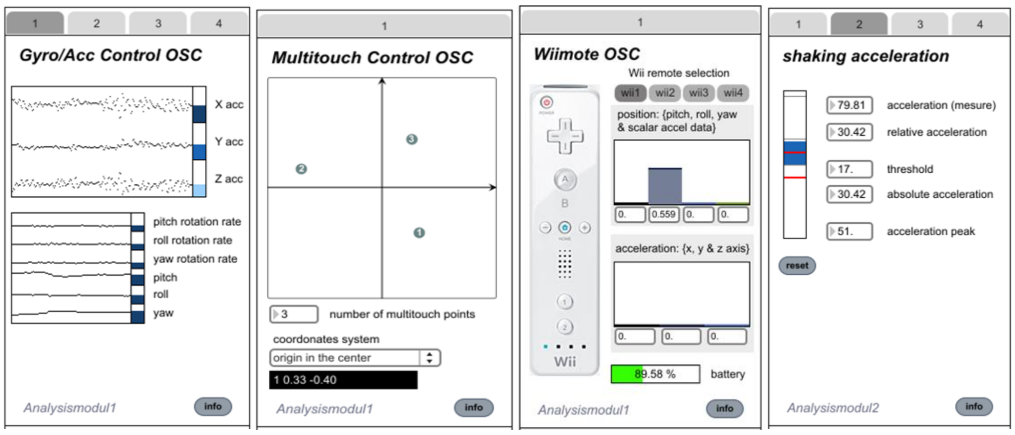


Figure 9: Gesture analysis: The *Control OSC* (sensors and touchscreen), *Wiimote OSC* and *Shaking acceleration* analysis modules.

4.3 Mapping modules

Mapping: Transforms data in three possible ways: graphical definition of a piecewise linear function (top), definition of a linear function by its coefficients (middle), definition of a custom function (bottom).
Scaling: Allows to intuitively defining a linear mapping function, providing two examples of input data and their corresponding desired output values.

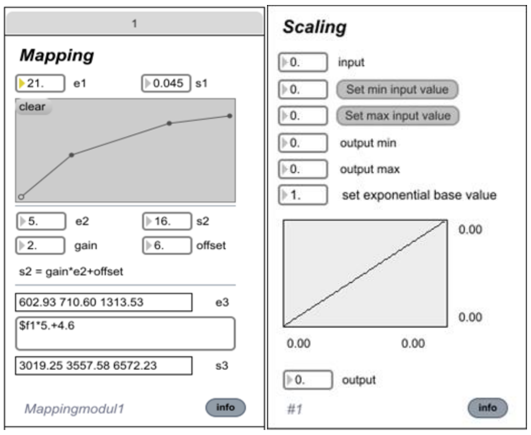


Figure 10: Mapping: The *Mapping* and *Scaling* mapping modules.

4.4 Synthesis modules

4.4.1 Sound generation

GeneCARS: Synthesizes engine sounds, using the GeneCARS technology.
IUAV Engine: Engine sound synthesizer, based on the *sdt.motor~* sound model available in the SDT.

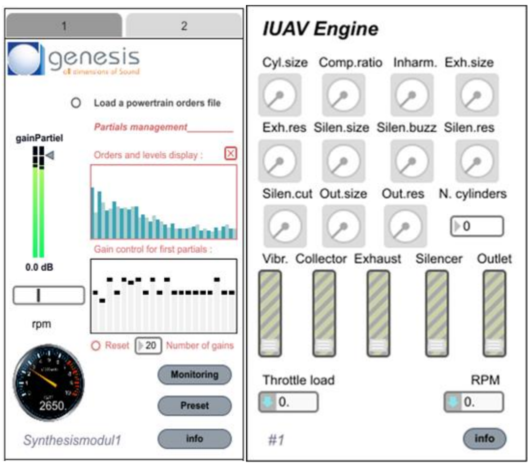

IUAV Engine
Cyl.size Comp.ratio Inharm. Exh.size
Exh.res Silen.size Silen.buzz Silen.res
Silen.cut Out.size Out.res N. cylinders
Vibr. Collector Exhaust Silencer Outlet
Throttle load
RPM
Load a powertrain orders file
Partials management
Orders and levels display :
Gain control for first partials :
Reset
Number of gains
gainPartiel
0.0 dB
rpm
Monitoring
Preset
info
Synthesismodul1

Figure 11: Engine sounds: The *geneCARS* and *IUAV Engine* synthesis modules.

Bell: Synthesis of bell-like sounds, based on the sdt.impact~ sound model, available in the SDT.
Impact: Synthesis of impact sounds, using the Sound Design Toolkit models for basic mechanical interactions between solids.
IUAVWater: Synthesis of water sounds, using the Sound Design Toolkit fluid flow model.
VST Shaker: This module is the integration of an external VST plugin, to demonstrate the feasibility of such integration.

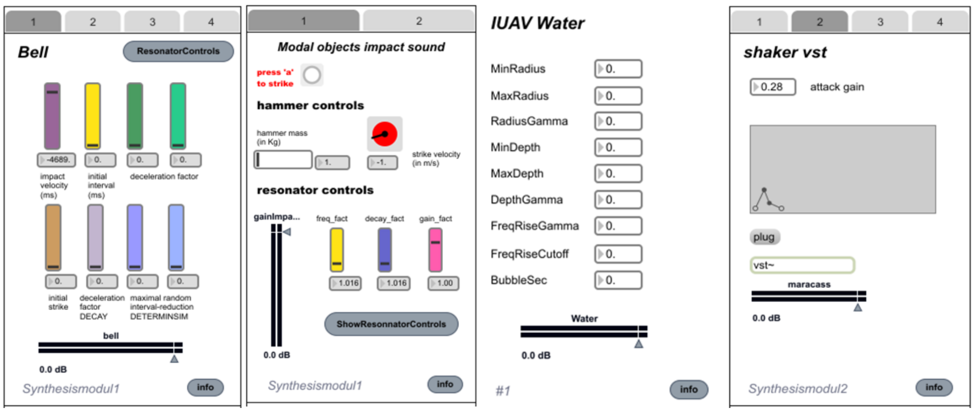


Figure 12: Sound generation: The *Bell*, *Impact*, *IUAV Water* and *Shaker VST* synthesis modules.

4.4.2 Sound modification

Remix: Four channel audio mixer.
Parametric EQ: Parametric equalizer.
Pitch shifter/Time stretcher: Applies a pitch shift and/or a time stretch to a given sound.
Doppler effect: Applies a Doppler effect and a left/right specialization to a sound, to simulate moving sound sources.

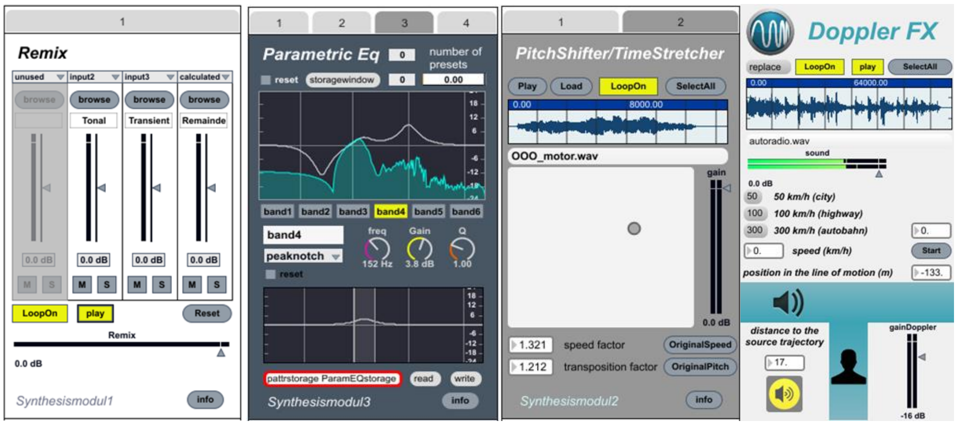


Figure 13: Sound modification: The *Remix*, *Parametric EQ*, *Pitch Shifter/Time stretcher* and *Doppler effect* synthesis modules.

4.5 Output modules

Audio output: Plays back the produced sounds on the default audio output device (loudspeakers, headphones, …).
Recorder: This module allows to record the produced sounds to an audio file.

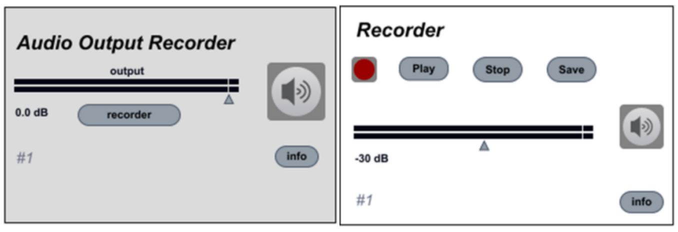


Figure 14: From left to right: The *Audio output* and *Recorder* output modules.

5 Examples

5.1 Engine sound controlled with voice

In this configuration, the user controls the RPM of a car engine with his voice pitch. The implementation in SkAT Studio is displayed in figure below.

Please note how the geneCARS configuration workflow reflects the general five-point structure of the SkAT Studio framework.

Note also that IUAV engine and geneCARS modules having the same control data (RPM, speed and load; only RPM used in this case), either one or the other may be used for the sound generation.

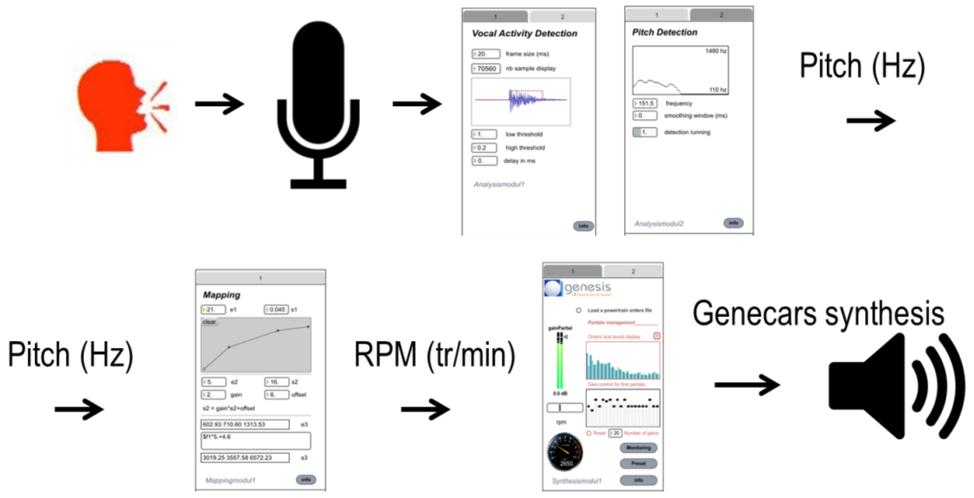


Figure 15: "Engine sound controlled with voice" scenario implementation

5.2 Bell sound controlled with 3D gesture

By shaking his/her smartphone or a Nintendo Wiimote, the user controls a bell sound as if it had a real bell in her hand. The implementation in SkAT Studio is displayed in figure below. The communication between the smartphone (or the Nintendo Wiimote) and SkAT Studio is done using OSC format, via bluetooth or wifi. OSC data are decoded to get the acceleration, which is analyzed to look for peaks. When a peak is detected, its value is used as input to control the strength of the impact on the bell.

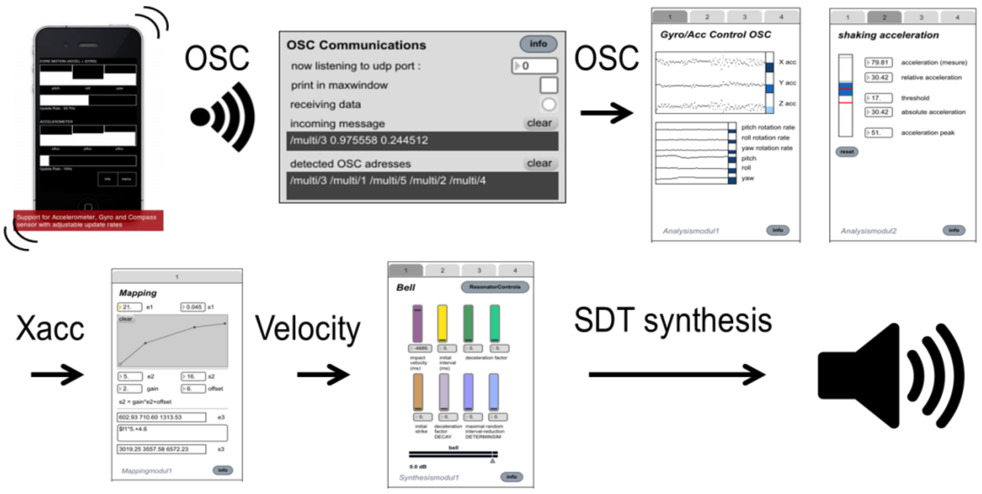

OSC
OSC Communications
now listening to udp port :
print in maxwindow
receiving data
incoming message
/multi/3 0.975558 0.244512
detected OSC adresses
/multi/3 /multi/1 /multi/5 /multi/2 /multi/4
OSC
Gyro/Acc Control OSC
shaking acceleration
Support for Accelerometer, Gyro and Compass sensor with adjustable update rates
Xacc
Mapping
Velocity
Bell
SDT synthesis

Figure 16: "Bell sound controlled with 3D gesture" scenario implementation

5.3 Control the remix of sound with 2D gesture

By moving his finger on a 2D device (for example a smartphone or a tablet), the user dynamically controls the mixing of three sounds: the 2D position directly controls the level of two sounds, while the third one is implicitly computed to keep constant the overall level. The implementation in SkAT Studio is displayed in figure below: the communication between the smartphone (or the tablet) and SkAT Studio is done using OSC format, via bluetooth or wifi. The 2D position (between -1 and 1) is directly used as a control for the mixing levels. The mixed sounds are the separation of the tonal, transient and noisy parts of a diesel engine with a turbo whistling.

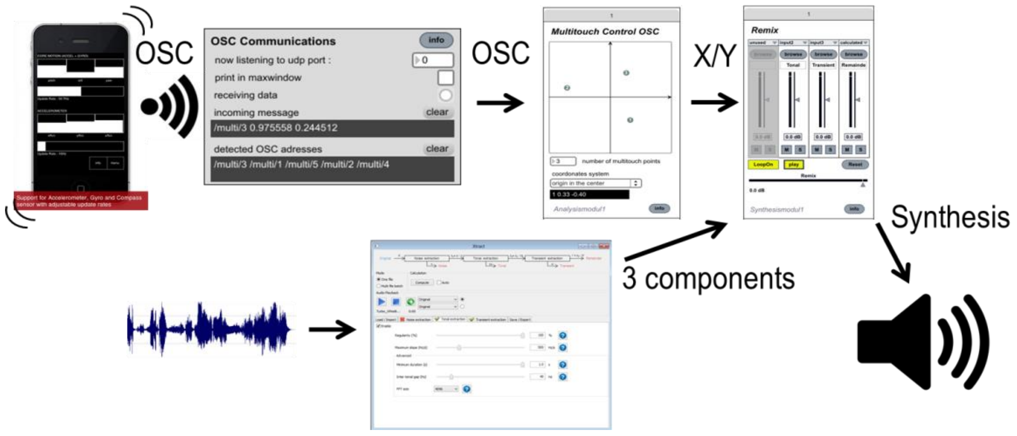

OSC
OSC Communications
OSC
Multitouch Control OSC
X/Y
Remix
Synthesis
3 components
Support for Accelerometer, Gyro and Compass sensor with adjustable update rates

Figure 17: "Control the remix of sound with 2D gesture" scenario implementation