Spat : a comprehensive toolbox for sound spatialization in Max

Thibaut Carpentier

Spat~: a comprehensive toolbox for sound spatialization in Max

Thibaut Carpentier
STMS Lab — CNRS — IRCAM — Sorbonne Université
1, place Igor Stravinsky, 75004 Paris
thibaut.carpentier@ircam.fr

1. INTRODUCTION

Ircam's Spatialisateur [1, 2], frequently dubbed spat~, is a suite of audio tools dedicated to real-time sound spatialization, artificial reverberation, and sound diffusion. It has been developed at Ircam since the early 1990s, and it primarily operates in the Max [3] environment. It is packaged as a comprehensive toolbox[1] of audio processors, control objects, and graphical user interfaces. Typical fields of application include concerts, mixing, post-production, virtual reality (VR), sonic installations, sound design, etc. The software suite is developed through Agile methods [4], and it continuously integrates state-of-the-art technologies, research outcomes from the Acoustics and Cognition Team (formerly Room Acoustics Team), and feedback from users – sonic artists and sound engineers. As the application is built as a long-term project, its development roadmap tackles multiple concerns: improvement of existing features, implementation of new functionalities, maintenance (adaptive, corrective, and preventive), optimization, architecture refactoring (in order to improve maintainability and extensibility of the code base), etc.

This paper offers an overview of the toolkit, with an emphasis on the current version, spat~5, released in 2018. First, we present the framework and underlying space-time-frequency model of spat~ (Section 2). This framework has been established since the inception of the tool, and has been perpetuated over the releases. Next, we detail the primary features of the toolbox (Section 3), with a highlight on Ambisonics technology as this has received significant attention in recent years. Section 4 briefly enumerates a number of handy tools, also part of the spat~ package, that form a complementary apparatus for the production of spatialized media.

The foundation of spat~ is a powerful C++ library that is mostly host- and platform- independent. This software library is indeed embedded into several environments such as Open Music [5], o7 [6], Flux:: Spat Revolution[2], Matlab[3], or PureData[4].

Section 5 outlines some specifics of its integration within Cycling'74 Max[5] host framework.

This article is an extended and updated version of the work previously published in [7, 8].

2. CORE FRAMEWORK: A SIMPLE SPACE-TIME-FREQUENCY MODEL

At the core of the spat~ library is the spat5.spat~ external object[6]. It is an all-encompassing audio engine that handles the necessary operations for creating spatialization effects. Its processing architecture is depicted in Figure 1.

(1) The input signal, assumed devoid of reverberation, first goes through a pre-processing module that can simulate air absorption and Doppler effect.

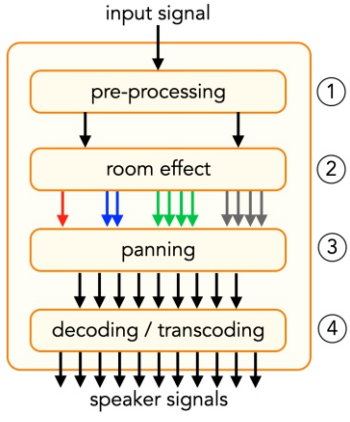

Figure 1: Signal processing architecture of the spat5.spat~ object.

1 The library features more than 250 external objects, along with example and tutorial patchers.
2 https://www.spatrevolution.com
3 https://www.mathworks.com
4 https://https://puredata.info
5 https://cycling74.com

6 In order to avoid confusion or conflicts with other toolboxes, all entities of the spat~ library are named with the "spat5." prefix. This explains the seemingly redundant naming of the spat5.spat~ object.

(2) Then, the room module produces an artificial reverberation effect. This is implemented with a scalable, multichannel, reverberation unit using feedback delay networks (FDN) [9]. The unit is designed to produce a natural-sounding reverb effect, following a simplified model of a room impulse response (IR) (see upper part in Figure 2). The reverberation engine delivers independent signals for direct sound, early reflections (discrete echoes, typically spanning approximately 20 msec), late reflections (a denser pattern of discrete reflections, also referred to as "cluster" in the spat~ dialect), and the late, exponentially-decaying, reverberation tail. These four temporal segments –respectively depicted in red, blue, green and grey in Figure 1 and Figure 2– constitute an object-based model of the room effect that is independent of the reproduction system (loudspeaker layout or headphones).

When multiple sources have to be rendered simultaneously, it is possible to share the late sections ("cluster" and tail), in order to save computational resources.

(3) The streams are later filtered with a three-band filter bank (low/medium/high frequency range), and sent to the panning module that distributes the signals in space (lower part in Figure 2); by default, direct sound and early reflections are precisely localized, while the late reflections and late tail are spatially diffuse (decorrelated sound field with isotropic energy distribution). The panning module can render various spatialization algorithms, later discussed in section 3.2.

(4) Finally, an optional module can decode/transcode signals for specific use cases such as Ambisonics playback.

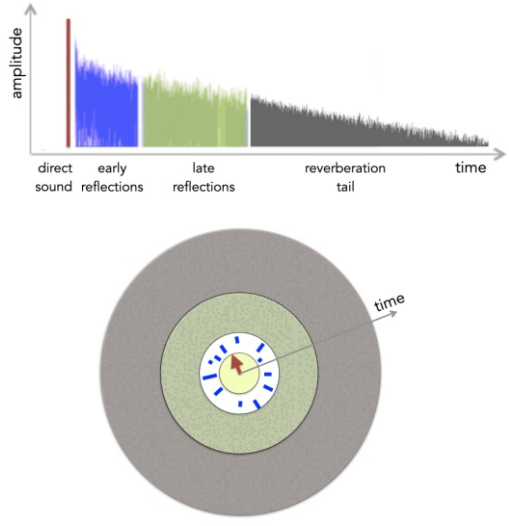

Figure 2: Simplified space-time model used in spat~. Top: temporal profile of the impulse response. Bottom: schematic space-time distribution of reflections.

Thanks to this architecture and to the underlying space-time-frequency model, the desired temporal and directional effects can be specified irrespective of the reproduction setup: a sound scene can be rendered for arbitrary loudspeaker layouts, and only the panning and decoding modules have to be configured appropriately.

The spat5.spat~ processor can be parameterized by a high-level control interface[7], which allows to specify and modulate the acoustical quality of the synthesized room effect according to perceptually relevant criteria [10, 11]. With four temporal segments controlled over three frequency bands, the time-frequency model of spat~ essentially offers twelve degrees of freedom; the high-level control interface provides an intuitive way to navigate into this parameter space, and allows to seamlessly interpolate between different acoustic qualities.

In addition to the all-in-one spat5.spat~ object, the toolbox also features separate externals for each module of the framework (e.g. spat5.room~, spat5.pan~, spat5.decoder~, etc.), allowing for modularity, flexibility, and granularity. Therefore, users are not tied to the proposed space-time-frequency model, as they can combine the low-level components to their needs.

3. PRIMARY FEATURES

This section, without purporting to do so exhaustively, will consider the principal characteristics of the spat~ toolset.

3.1 Artificial reverberation

As mentioned in the previous section, one fundamental element of spat~ is its reverberation engine. It consists of an efficient and scalable FDN unit: the number of internal feedback channels is adjustable, typically ranging from 8 to 16, but higher values can also be used when high modal density or long reverberation times are desired. For natural-sounding effects, the FDN embeds lowpass filters simulating air absorption. Control of the decay rate (i.e. reverberation time) is achieved by proportional parametric multi-band equalizers [12], usually operating in three frequency bands, although finer spectral resolution is also possible.

Besides algorithmic reverberation, it is also possible to use convolution-based techniques, as they have become widespread during the last decades, thanks to the increase in available processing power. spat5.conv~ is an efficient multi-channel convolution engine based on overlap saved

7 This so-called "perceptual operator" is implemented in the spat5.oper object.

block-partitioned FFT algorithms [13]. The computation of the high latency blocks is handled in a background thread in order to take advantage of modern multi-core processors. Users can adjust the minimum block size which offers a tradeoff between audio latency (which can be as low as zero) and computational load. spat5.converb~ is another convolution processor based on similar DSP algorithms. spat5.converb~ extends spat5.conv~ by further slicing the impulse response in four temporal segments (with adjustable lengths) according to the spat~ paradigm (Figure 2); parametric filters are additionally applied to each segment of the IR. One can control the four filters either at the low-level (gains and cutoff frequencies) or by means of the spat5.oper high-level perceptual approach presented in section 2.

Convolution-based reverberators relie on IR obtained through acoustical measurements; the spat~ library provides a toolkit to do so, and this is later described in section 4.1.

3.2 Panning

Panning is performed by the spat5.pan~ external, a polymorphic object that supports a wide range of spatialization algorithms such as:

- Traditional stereo techniques, emulating the properties of AB, XY, or MS microphones.

- Binaural synthesis, including simulation of nearfield effects (see section 4.2 in [8] for details). Custom sets of HRTF can be loaded, as spat~ implements the "Spatially Oriented Format for Acoustics" (SOFA) standard [14].
  It is also possible to transcode binaural streams to two-channel loudspeaker system with cross-talk cancellation (CTC). Various CTC implementations are available (see also [15] and section 4.3 in [8]), offering a tradeoff between localization accuracy and spectral coloration.

- Amplitude panning techniques, such as vector base amplitude or intensity panning (VBAP, VBIP) [16, 17] in 2- or 3D, distance-based amplitude panning (DBAP) [18], layer based amplitude panning (LBAP) [19], Ambisonics equivalent panning (AEP) [20], speaker-placement correction amplitude panning (SPCAP) [21], etc.
  We have also developed a K-nearest neighbors amplitude panner, which, similarly to DBAP is a distance-based approach. However, it offers more flexibility in choosing the number of contributing loudspeakers. This number can either be set by the user, or adjusted on the basis of a maximal distance (with the respect to the position of the virtual source) criterion.

  For triangulation of 3D layouts – as required e.g. for VBAP – spat~ uses robust implementation from the qhull library [22], and it is further possible to introduce imaginary loudspeaker(s) (see for instance chapter 3 in [23]).
  Additionally, users can control the spread (perceived width) of the virtual sources. This is achieved through multiple-direction amplitude panning (MDAP) [24] wherein several virtual sources are distributed around the main panning direction.

- Wave-field synthesis (WFS) [25], currently restricted to linear loudspeaker arrays.

- Higher-order Ambisonics (HOA) [23, 26], which is further detailed in the next paragraph 3.3.

Note also that arbitrary loudspeaker layouts can be monitored over headphones, by using spat5.virtual speakers~ which relies on a virtual speakers paradigm to "down-mix" any multichannel stream into a binaural stereo track preserving the spatial image of the original sound scene.

3.3 Higher-Order Ambisonics

Ambisonics technologies have recently gained great interest from both the academic world and the media industry. We have consequently integrated in spat~ state-of-the-art tools to leverage the HOA production workflow. Besides classical Ambisonics encoder (spat5.hoa.enco-der~), this notably includes:

- Tools for converting between HOA conventions, such as spat5.hoa.sorting~ and spat5.hoa.converter~. In particular, the various normalization schemes in used for HOA (FuMa, MaxN, SN3D, N3D, etc.) are a frequent source of confusion for the users, and they may lead to compatibility issues between rendering tools. A formalization effort has been proposed [27], and the spat~ documentation has been significantly improved in order to clarify the impact of normalization schemes in the Ambisonics production workflow, and to ease interoperability.

- State-of-the-art HOA decoders, supporting sampling decoder (SAD), mode-matching (MMAD) [26], energy-preserving (EPAD) [28], All-Round Ambisonic Decoding (All- Rad) [29], Constant Angular Spread (CSAD) [30], AllRAD+ [31], most VBAP-like Ambisonic decoder (MVLAD) [31], etc. spat5.hoa.decoder~ can further perform phase-matched dual-band decoding [32], with adjustable crossover frequency, and in-phase or max-re optimizations can be applied in each sub-band.

- A-to-B format encoders that transcode microphone signals into the Ambisonics domain. First order microphones (such as Sennheiser Ambeo, Soundfield, Tetramic, etc.) as well as higher-order compact spherical arrays (e.g. MH Acoustics EM-32 or Zylia ZM1) are supported.

- A number of frequency-independent Ambisonic effects (FX) that can globally transform the sound field in a linear or non-linear way. This encompasses: spat5.hoa.rotate~, performing 3D rotations (yaw, pitch, roll) of the sound field; spat5.hoa.mirror~, re-mapping the sound scene with regard to planes of symmetry; spat5.hoa.blur~ is a tool for manipulating the "spatial resolution" of an encoded HOA field. It allows to continuously vary the order of the HOA stream (i.e. simulating fractional orders), while preserving the overall energy [33]. It can be used to adapt the order of existing content, or as a creative FX (typically by varying the "blur" factor dynamically); spat5.hoa.focus~ is another effect operating in the HOA domain. Inspired from [34], it allows to synthesize virtual directivity patterns and apply them to a HOA stream. Orientation and selectivity of the pattern(s) can be edited in an intuitive graphical user interface (Figure 3). The tool is most useful during post-production stage, i.e. when applied to recorded HOA materials, as it allows to directionally "zoom" into the sound scene; spat5.hoa.warp~ which distorts the spatial image with respect to a given direction, e.g. squeezing or stretching the content towards or away from the horizon. The warping processor somehow extends first order dominance effect [35] to HOA.

- A number of other tools operating in the HOA domain: spat5.hoa.beam~ applies beamforming with adjustable beampattern and steer direction(s) in order to extract mono signals from the sound scene; spat5.hoa.scope~ is a metering interface, displaying in 2D or 3D the RMS or peak value of the HOA stream sampled on a t-design grid [34]; spat5.hoa.intensity~ computes the sound inten-sity vector [36], from which is derived a realtime estimate of the direction-of-arrival (DOA) and diffuseness of the sound field.

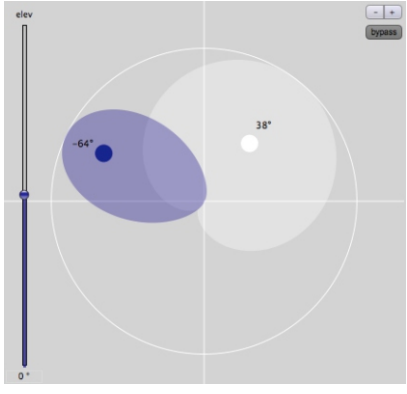

Figure 3: spat5.hoa.focus interface for synthesizing virtual patterns in the HOA domain.

4. OTHER FEATURES

4.1 Acoustical measurements and calibration

Spat5.smk~ is a tool for measuring the impulse response of linear time-invariant systems, such as concert halls, using the swept-sine technique [37]. After configuration of the sweep signal (duration, waveform, frequency range, etc.), the object performs the measurement of the system (which can be multichannel), deconvolves the raw signals, and saves the data as well as relevant metadata to disk. After each measurement, the object estimates in real-time various criteria (signal-to-noise ratio, harmonic distortion ratio, reverberation time, etc.) for controlling the quality of the measure. In addition to that, spat5.smk~ can be linked to other acoustical tools such as spat5.edc which computes and displays the energy decay curve of the IR, or spat5.ir.analysis which calculates a number of standard acoustical criteria (clarity, central time, early decay time, etc.).

The spat~ toolbox also proposes tools for calibrating a reproduction system, a crucial step when dealing with panning and spatialization on irregular loudspeaker layouts. spat5.align~ performs delay and level alignment of an arbitrary set of loud-speakers, according to their geometrical coordinates. Similarly, spat5.calibrate.delay~ and spat5.calibrate.gain~ can be used to quickly and automatically calibrate a loudspeaker setup and adjust the respective gain and delay of each channel. Unlike spat5.align~, these objects rely on in-situ acoustical measurements.

4.2 Geometrical operations

The production of spatialized media most often involves the manipulation of geometrical data, be it for the configuration of the reproduction system or for more creative purposes. In particular, multichannel composition frequently uses the paradigm of spatial trajectories, wherein motions of sound sources are apprehended as paths, curves, geometrical patterns, etc. Therefore, the spat~ library comes with a large set of utility tools to generate, transform, and manipulate geometrical data e.g. 2D or 3D coordinates. These externals facilitate standard operations such as translation, rotation, scaling, mirroring, distance normalization, calculation of nearest neighbors, barycenter, convex hull, Delaunay triangulation, etc.

In addition to that, spat5.grids can generate a wide range of static distribution of points in space, according to various mathematical constraints.

In contrast, spat5.trajectories produces dynamic (in motion) trajectories, based on parametric equations of curves (e.g. ellipse, cycloid, Moebius patterns, etc.). A large set of predefined equations is available, of which the user can change speed and the properties. Finally, spat5.simone is a creative software tool that metaphorically uses the concept of vector field in order to produce fluid, lifelike, motions of autonomous agents. A vector field is generated as a grid of arrows, each with a given length and direction (see Figure 4). When objects or "particles" are thrown into this field, they become animated, and they navigate through the domain, under the influence of the vector field (which can also dynamically vary). See [38] for additional details and examples.

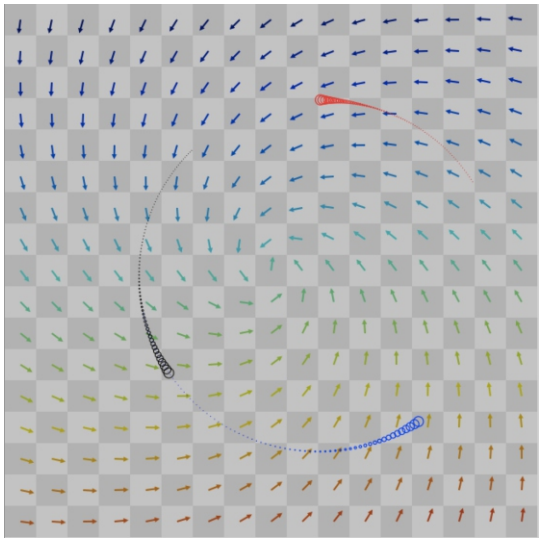

Figure 4: Graphical user interface of spat5.simone. In this example, the vector field has a vortex-like shape. The "strength" of each arrow is color-coded, and it is mapped to the speed/acceleration of the particles. In this example, three trajectories are generated, and depicted by the red, blue, and black dots.

4.3 Quaternions

With the democratization of VR devices, spat~ is more and more used for rendering audio scenes in immersive multimedia applications, typically presented with binaural over headphones. These virtual environments require the manipulation of 3D geometrical data. In particular, controlling the orientation of entities (either audio objects or the listener in the scene) is a frequent source of confusion for the users, as various conventions are being used (and they are not intercompatible). To alleviate this problem, we have developed a library of externals for the manipulation of quaternions, Euler angles, and 3D rotation matrices. These tools allows for converting between the different representations. The spat~ audio engines (for instance spat5.binaural~) can be controlled by either quaternions or Euler angles, therefore facilitating the cross-operability with VR SDKs.

4.4 Panoramix

Panoramix is a full-featured versatile workstation for sound spatialization and artificial reverberation, primarily intended for 3D mixing and post-production scenarios. The tool has been presented in previous publications [39, 40]. It offers essentially the same functionalities as spat5.oper and spat5.spat~, however with an ad-hoc front-end (see Figure 5), especially designed for mixing heterogeneous multichannel content, seamlessly combining object-, scene-, and channel-based paradigms.

Even though it is distributed separately as a standalone application, panoramix is completely built upon modules included in the spat~5 package. It is therefore possible to integrate spat5.panoramix~ in larger Max projects (also in Max for Live devices), to cope with non-conventional mixing scenarios or to better communicate with other multimedia tools. The interested reader can refer to [39, 40] for further details about the design and usage of panoramix.

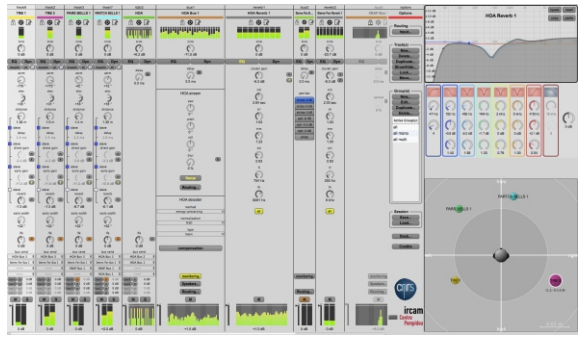

Figure 5: Overview of the spat5.panoramix mixing workstation.

4.5 Object-based audio

In recent years, there has been renewed interest in the object-based paradigm for producing and broadcasting multichannel audio. Several inter-exchange formats have proposed; in particular, the Audio Definition Model (ADM)[8] is an open standard, published by the ITU and EBU[9], for the description of object-oriented media encapsulated

8 ITU-R BS.2076 (ADM Audio Definition Model) https://www.itu.int/rec/R-REC-BS. 2076/

9 International Telecommunication Union and European Broadcasting Union

in a Broadcast Wave Format (BWF) container. ADM prescribes a set of metadata (such as time-varying position and gain of audio objects) encoded in a XML chunk. spat~is one of the first toolbox offering a complete production chain for BWF-ADM files: spat5.adm.record~ allows for the creation of BWF file with embedded spatialization metadata, and spat5.adm.renderer~ copes with the real-time rendering of ADM media over an arbitrary reproduction setup (headphones or loudspeaker layout); other externals also allow to handle objects' interactivity. These externals are presented in greater details in [41]. Note however that only a subset of the ADM specifications is currently supported (although covering most typical usages), and a tighter integration of the format within the spat~ architecture remains to be done (e.g. direct import/export of ADM files from processors such as spat5.spat~). This is part of on-going development work.

4.6 Multichannel tools

When working with massively multichannel data, users often face inappropriate or cumbersome tooling, even for basic operations. The spat~ library has thus been supplemented by a number of simple – yet efficient – tools for dealing with these common tasks. It is beyond the scope of this article to examine them exhaustively and we just mention here a few examples:

spat5.sfplay~ and spat5.sfrecord~ supersede the Max built-in sfplay~ and sfrecord~ objects: while they exhibit similar functionalities and messaging, they are especially designed for massively multi-channel streams. They can easily handle hundreds of audio channels, and they support a wide range of audio formats, in particular WAV RF64 which allows the usual 4GB file size limit to be overridden.

Mixing and producing spatial audio usually requires multichannel FX, therefore spat~ comes with a tool-chain of processors such as compressor, limiter, noise gate, graphic EQ, parametric EQ, etc., especially optimized for applications with high-channel count. Facilities for multichannel routing and signal matrixing are also provided, complementing the built-in Max features.

4.7 Time Code

Spat~ is also frequently used for audio-visual productions; in such contexts, it is necessary to synchronize the audio and video streams. One of the most popular technique to do so, is to use a *Linear Timecode* (LTC) [42] which encodes SMPTE frames, and is transmitted as a longitudinal audio signals.

Unfortunately, this standard is not natively supported in Max. We have therefore developed tools for receiving (spat5.ltc.decode~) and generating (spat5.ltc.encode~) linear time codes.

Furthermore, the spat5.ltc.trigger~external can be used as a cue manager as it triggers actions at specific (user-defined) time stamps; the temporal granularity is rather low (typically 30 fps≈33 milliseconds), but sufficient for most spatialization use cases.

4.8 Control interfaces

The spat~ package contains more than thirty graphical user interface (GUI) control objects. Some of them are particularly tied to a given processor, e.g. we have mentioned before spat5.oper (see Figure 6) the control interface bound to spat5.spat~.

On the other hand, some of these GUIs are completely generic, and highly customizable, and they can be employed to control a broad range of processors (not necessarily from the spat~ library) in various contexts.

As will be discussed in Section 5, user-friendly mechanisms are offered in order to edit (via Max messages), store, recall, interpolate, the parameters exposed by these GUIs. The graphic components can be temporarily disabled –while maintaining all other functionalities alive– in order to optimize performances. This is especially useful during showtime, when only minimal graphical feedback is needed/wished.

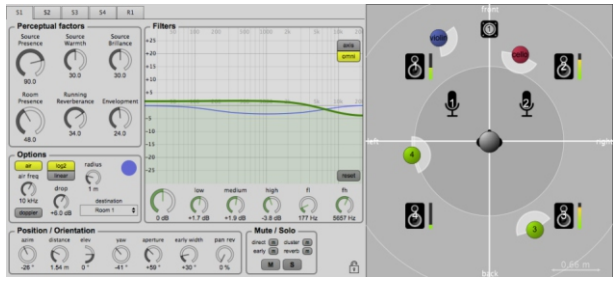

Figure 6: Graphical user interface for spat5.oper (high-level perceptual control for spat~). Perceptual factors for controlling room effect (left); filtering (centre); 2D view of the sound scene (right).

5. INTEGRATION IN MAX

5.1 OSC syntax

Starting with version 5 of spat~, all external objects natively support the Open Sound Control (OSC) [43] protocol and syntax. The rationale behind this choice has been explained in [7].

At the Max interface level, OSC messages are converted to/from Max native format: atoms. Such conversion is trivial as atoms and OSC arguments have very similar data type (int, float, symbols, etc.). OSC bundles are transmitted as *FullPacket* that only convey a pointer to a memory address (similarly to Max dictionaries). This allows for the very efficient transmission of large amount of data (the Max scheduler service is triggered only once per bundle, and not for each individual message contained in the bundle).

The syntax we have adopted is inspired from the REST (Representational State Transfer) style [44]. For instance, to control the Cartesian position of a sound source in spat~5, one can use the following message:

/source/1/xy [float][float]

External objects support pattern matching semantics, which facilitate the grouping of multiple elements:

/source//mute [boolean]*
/source/[2-5]/mute [boolean]
/source/{3,6,7}/mute [boolean]

Routing and dispatching OSC address patterns in Max may require the manipulation of regular expressions *(regexp)*, which is usually cumbersome and inefficient; we have thus developed a toolbox of handy objects (approximately 35 externals) to ease usual operations (see Figure 7).

The most frequently used OSC address patterns are stored in a hash table at compile-time. This avoids CPU-intensive string operations during runtime, and guarantees efficient dynamic lookup (similar to Max static symbol tables).

5.2 Inter-operability and compatibility

One potential advantage of the OSC interface is that users can benefit from existing tools and libraries, such as:

- The *odot* package [45], which provides a powerful expression language for the manipulation of OSC bundles in a variety of programming paradigms. Typically, *odot* might be used for the algorithmic generation and transformation of spatialization data such as trajectories.

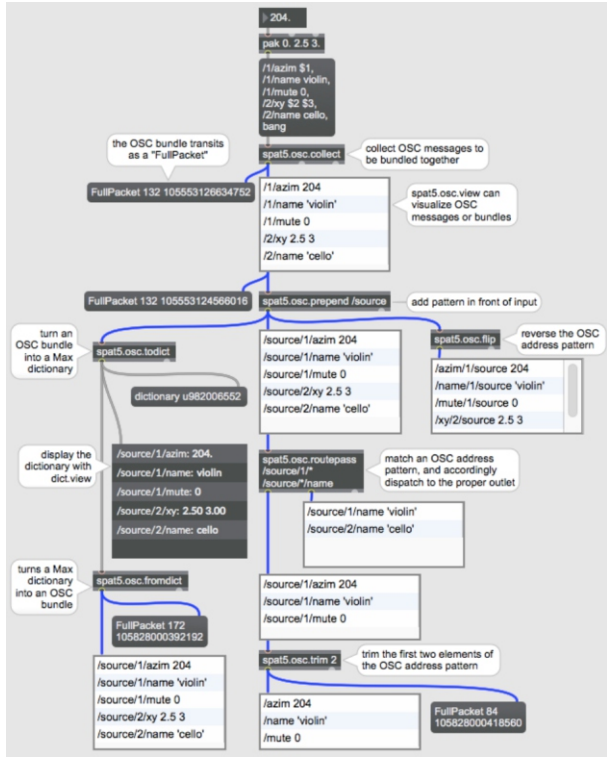

Figure 7: Examples of usual OSC manipulations. OSC bundles are conveyed (as FullPacket) through the blue-colored patchcords.

- *ToscA* [46] (and its successor *OSCar*), a DAW plugin that allows the transmission of automation data over OSC. Although *ToscA* is designed as a completely generic tool, it has been thought, from its inception, for the remote control of object-based spatialization processors such as spat~.

- *IanniX* [47], *o7* [6], *Antescofo* [48], *Vezér*[10], *Qlab*[11], *Holo-Edit* [49], etc., are other examples of OSC compatible software tools with powerful features for generative compositional processes.

10 https://imimot.com/vezer/
11 https://qlab.app

5.3 Usability

With very few exceptions, spat~5 external objects do not use Max built-in attributes. As a consequence, they cannot benefit from Max built-in features such as the inspector or automatic documentation hints. We have thus introduced ad-hoc mechanisms that serve as a replacement: each object has its own status and help window (see Figure 8). The status window displays the current state of the object, similar to the Max inspector; it comes with a search filter, and one can copy/paste messages from this window to the patcher. The help window displays a text description of all supported OSC messages. Reference pages are also proposed and can be accessed via the standard Max documentation browser.

The state of each object is internally represented as an OSC bundle. Convenient mechanisms are offered to export/import this bundle, either as human-readable file on disk, or as in-memory snapshot. This provides a simple process for creating, recalling, and even interpolation presets. The bundle can also be stored (i.e. embedded) into the host patcher, or via the Max snapshots window ("Parameter Enable Mode"); in these cases, the OSC bundle is converted to a binary blob, and saved within the patcher file.

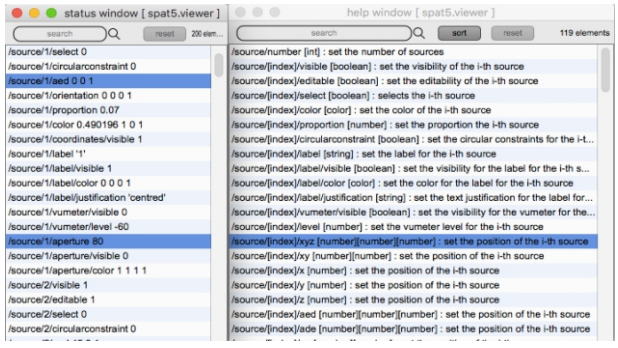

Figure 8: Status window (left) and help window (right) for spat5.viewer.

5.4 Multichannel audio streams

Introduced with Max version 8, MC –acronym for "Multi-Channel"– is a new feature that, among other things, enables the use of multi-channel signal patchcords, conveying an arbitrary number of channels.spat~5 has been one of the first toolbox made compatible with MC as this tremendously eases the development, readability, and maintenance of massively multichannel applications for high-density loudspeaker arrays, such as presented in [50].

This feature is disabled by default, for backward compatibility with previous Max versions. It is simply enabled via the *@mc* attribute of spat~externals. spat~objects support an arbitrary number of input/output audio channels, and the only limiting factor is the available computing capacity of the host processor.

5.5 Scheduling and thread-safety

Most audio software applications, including Max, involve multiple concurrent threads such as the audio thread, message thread, high-priority events thread, etc. Proper communication and synchronization between these processes, under real-time constraints, is essential for the integrity and efficiency of the program. spat~ uses a thread-safe non-blocking FIFO[12] queue (detailed in [51]) wherein incoming events (OSC messages or bundles) are stored, and later processed, in due time and in the appropriate thread.

For DSP objects, the FIFO is dequeued in the audio thread, at the beginning of the rendering callback (see Figure 9). Such behavior is similar to the Max scheduler in audio interrupt mechanism. By default, all spat~5 audio objects operate as such, regardless of the overdrive or interrupt settings of the host application. Admittedly this strategy does not guarantee perfectly accurate timing, as events might be delayed until the beginning of the next audio callback. However, such temporal granularity is believed to be sufficient for most sound spatialization applications. When a faster automation rate is needed, dedicated spat~ externals supporting signal-rate (i.e. sample-accurate) control can be used (see e.g. spat5.pansig~).

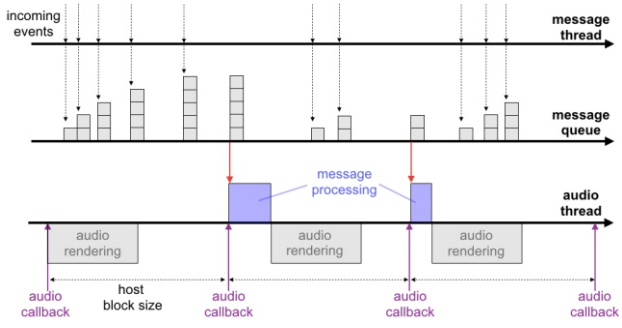

Figure 9: Scheduling of events according to the scheduler in audio interrupt procedure.

12 First-In First-Out

6. CONCLUSION

We have presented a broad overview of the spat~5 framework for sound spatialization and reverberation, implemented in the Max environment. The library contains a modular, flexible, scalable, comprehensive set of more than 250 processors, covering a large scope of multichannel multimedia activities. The toolkit is accompanied by rich documentation and tutorials.

Driven by research outcomes, technological innovations, and artistic challenges, spat~ remains an ever-evolving environment.

The short-term prospects notably focus on the extension of the framework to 3D spatial room impulse responses manipulated in the HOA domain, and the integration of an hybrid reverberation engine allowing to seamlessly combine algorithmic and convolution-based approaches.

The author would like to thank all researchers, composers, sound artists, sound engineers, etc. who contribute to improving spat~ with priceless ideas and discussions.

7. REFERENCES

1 J.-M. Jot, "Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces," ACM Multimedia Systems Journal (Special issue on Audio and Multimedia), vol. 7, no. 1, pp. 55 – 69, 1999.

2 J.-M. Jot and O. Warusfel, "A real-time spatial sound processor for music and virtual reality applications," in Proc. of the 21st International Computer Music Conference (ICMC), Banff, 1995, pp. 294 – 295.

3 M. Puckette, "The Patcher," in Proc. of the 14th International Computer Music Conference (ICMC), Köln, Germany, Sept 1988, pp. 420 – 429.

4 C. Larman, Agile and Iterative Development: A Manager's Guide. Addison Wesley, 2003.

5 J. Bresson, C. Agon, and G. Assayag, "OpenMusic – Visual Programming Environment for Music Composition, Analysis and Research," in Proc. of the 19th ACM International Conference on Multimedia (OpenSource Software Competition), Scottsdale, USA, 2011.

6 J. Bresson, D. Bouche, T. Carpentier, D. Schwarz, and J. Garcia, "Next-generation Computer-aided Composition Environment: A New Implementation of OpenMusic," in Proc. of the International Computer Music Conference (ICMC), Shanghai, China, Oct 2017.

7 T. Carpentier, "A new implementation of Spat in Max," in Proc. of the 15th Sound and Music Computing Conference (SMC). Limassol, Cyprus: https://doi.org/10.5281/zenodo.1422552, July 2018, pp. 184 – 191.

8 T. Carpentier, M. Noisternig, and O. Warusfel, "Twenty Years of Ircam Spat: Looking Back, Looking Forward," in Proc. of the 41st International Computer Music Conference (ICMC), Denton, TX, USA, Sept. 2015, pp. 270 – 277.

9 J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in Proc. of the 90th Convention of the Audio Engineering Society (AES), Paris, France, Feb 1991.

REFERENCES

10 J.-P. Jullien, "Structured model for the representation and the control of room acoustic quality," in Proc. of the 15th International Congress on Acoustics (ICA), Trondheim, Norway, June 1995, pp. 517 – 520.

11 E. Kahle and J.-P. Jullien, "Subjective listening tests in concert halls: Methodology and results," in Proc. of the 15th International Congress on Acoustics (ICA), Trondheim, Norway, June 1995, pp. 521 – 524.

12 J.-M. Jot, "Proportional parametric equalizers – application to digital reverberation and environmental audio processing," in Proc. of the 139th Convention of the Audio Engineering Society (AES), New York, NY, USA, Oct 2015.

13 W. G. Gardner, "Efficient convolution without input-output delay," in Proc. of the 97th Convention of the Audio Engineering Society (AES), San Francisco, CA, USA, 1994.

14 P.Majdak,Y.Iwaya,T.Carpentier,R.Nicol,M.Parmentier,A.Roginska,Y.Suzuki, K. Watanabe, H. Wierstorf, H. Ziegelwanger, and M. Noisternig, "Spatially oriented format for acoustics: A data exchange format representing head-related transfer functions," in Proc. of the 134rd Convention of the Audio Engineering Society (AES), Roma, May 2013.

15 A. Baskind, T. Carpentier, J.-M. Lyzwa, and O. Warusfel, "Surround and 3D-Audio Production on Two-Channel and 2D-Multichannel Loudspeaker Setups," in Proc. of the 3rd International Conference on Spatial Audio (ICSA), Graz, Austria, Sept. 2015.

16 V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," Journal of the Audio Engineering Society, vol. 45, no. 6, pp. 456 – 466, June 1997.

17 J.-M. Jot, V. Larcher, and J.-M. Pernaux, "A comparative study of 3-d audio encoding and rendering techniques," in Proc. of the 16th Audio Engineering Society International Conference on Spatial Sound Reproduction, Rovaniemi, 1999.

18 T. Lossius, P. Balthazar, and T. de la Hogue, "Dbap - distance-based amplitude panning," in Proc. of the International Computer Music Conference (ICMC), Montreal, 2009.

19 I. I. Bukvic, "3D time-based aural data representation using D4 library's layer based amplitude panning algorithm," in Proc. of the 22nd International Conference on Auditory Display (ICAD), Canberra, Australia, July 2016.

20 M. Neukom and J. C. Schacher, "Ambisonics Equivalent Panning," in Proc. of the International Computer Music Conference (ICMC), Belfast, Ireland, 2008.

21 R. Sadek and C. Kyriakakis, "A novel multichannel panning method for standard and arbitrary loudspeaker configurations," in Proc. of the 117th Audio Engineering Society Convention, San Francisco, 2004.

22 C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," ACM Trans. Math. Softw., vol. 22, no. 4, pp. 469 – 483, Dec. 1996. [Online]. Available: https://doi.org/10.1145/235815.235821

23 F. Zotter and M. Frank, Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality. Springer, 2019.

24 V. Pulkki, "Uniform spreading of amplitude panned virtual sources," in Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 1999, pp. 187 – 190.

REFERENCES

25 S. Spors, R. Rabenstein, and J. Ahrens, "The theory of wave field synthesis revisited," in Proc. of the 124th Convention of the Audio Engineering Society (AES), Amsterdam, The Netherlands, May 2008.

26 J. Daniel, "Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia," Ph.D. dissertation, Université de Paris VI, 2001.

27 T. Carpentier, "Normalization schemes in Ambisonic: does it matter?" in Proc. of the 142nd Convention of the Audio Engineering Society (AES), Berlin, Germany, May 2017.

28 F. Zotter, H. Pomberger, and M. Noisternig, "Energy-preserving ambisonic decoding," Acta Acustica united with Acustica, vol. 98, pp. 37 – 47, 2012.

29 F. Zotter and M. Frank, "All-round ambisonic panning and decoding," Journal of the Audio Engineering Society, vol. 60, no. 10, pp. 807 – 820, 2012.

30 N. Epain, C. Jin, and F. Zotter, "Ambisonic Decoding With Constant Angular Spread," Acta Acustica united with Acustica, vol. 100, pp. 928 — 936, 2014.

31 F. Zotter, M. Frank, and H. Pomberger, "Comparison of energy-preserving and all-round Ambisonic decoders," in Fortschritte der Akustik, AIA-DAGA, March 2013.

32 A. J. Heller and E. M. Benjamin, "Design and implementation of filters for Ambisonic decoders," in Proc of the 1st International Faust Conference (IFC-18), Mainz, Germany, July 2018.

33 T. Carpentier, "Ambisonic spatial blur," in Proc. of the 142nd Convention of the Audio Engineering Society (AES), Berlin, Germany, May 2017.

34 M. Kronlachner and F. Zotter, "Spatial transformations for the enhancement of Ambisonic recordings," in 2nd International Conference on Spatial Audio (ICSA), Erlangen, Germany, February 2014.

35 T. Lossius and J. Anderson, "ATK Reaper: The Ambisonic Toolkit as JSFX plugins," Ideas Sonicas/Sonic Ideas, vol. 8, no. 16, pp. 9 – 19, 2016.

36 J. Ahonen and V. Pulkki, "Diffuseness estimation using temporal variation of intensity vectors," in Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New York, NY, USA, Oct 2009, pp. 285 – 288.

37 A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in Proceedings of the 108th Convention of the Audio Engineering Society (AES), Paris, France, 2000.

38 T. Carpentier and A. Gerzso, "Steering Behaviors for Spatial Sound Authoring," in Proc. of the International Computer Music Conference (ICMC), New York, NY, USA, June 2019.

39 T. Carpentier, "Panoramix: 3D mixing and post-production workstation," in Proc. 42nd International Computer Music Conference (ICMC), Utrecht, Netherlands, Sept 2016, pp. 122 – 127.

40 ——, "A versatile workstation for the diffusion, mixing, and post-production of spatial audio," in Proc. of the Linux Audio Conference (LAC), Saint-Etienne, France, May 2017.

REFERENCES

41 M. Geier, T. Carpentier, M. Noisternig, and O. Warusfel, "Software tools for object-based audio production using the Audio Definition Model," in Proc. of the 4th International Conference on Spatial Audio (ICSA), Graz, Austria, Sept 2017.

42 J. Ratcliff, Timecode: A user's guide, 3rd Edition. Focal Press, 1999.

43 M. Wright, "Open Sound Control: an enabling technology for musical networking," Organised Sound, vol. 10, no. 3, pp. 193 – 200, Dec 2005.

44 A. Schmeder, A. Freed, and D. Wessel, "Best Practices for Open Sound Control," in Proc. of the Linux Audio Conference (LAC), Utrecht, Netherlands, May 2010.

45 A. Freed, J. MacCallum, and A. Schmeder, "Dynamic, instance-based, object- oriented programming in Max/MSP using Open Sound Control message delegation," in Proc. of the 37th International Computer Music Conference (ICMC), Huddersfield, Aug. 2011, pp. 491 – 498.

46 T. Carpentier, "ToscA: An OSC Communication Plugin for Object-Oriented Spatialization Authoring," in Proc. of the 41st International Computer Music Conference (ICMC), Denton, TX, USA, Sept. 2015, pp. 368 – 371.

47 T. Coduys and G. Ferry, "Iannix – Aesthetical/Symbolic visualisations for hypermedia composition," in Proc. of the of the 1st Sound and Music Computing Conference (SMC), Paris, France, Oct 2004.

48 A. Cont, "Antescofo: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music," in Proc. of the 34th International Computer Music Conference (ICMC), Belfast, Ireland, Aug 2008, pp. 33 – 40.

49 C. Bascou, "Adaptive spatialization and scripting capabilities in the spatial trajectory editor Holo-Edit," in Proc. of the 7th Sound and Music Computing Conference (SMC), Barcelona, Spain, July 2010, pp. 21 – 24.

50 T. Carpentier, N. Barrett, R. Gottfried, and M. Noisternig, "Holophonic sound in ircam's concert hall: Technological and aesthetic practices," Computer Music Journal, vol. 40, no. 4, pp. 14 – 34, Winter 2016.

51 T. Carpentier, "Synchronisation de données inter-processus dans les applications audio temps réel: qu'est-ce qui débloque ?" in Proc. of Journées d'Informatique Musicale (JIM), Amiens, France, May 2018, pp. 35 – 44.