SATIE: a live and scalable 3D audio scene rendering environment for large multi-channel loudspeaker configurations

Nicolas Bouillot
Society for Arts and Technology
1201, Boulevard Saint-Laurent
Montréal H2X 2S6, Canada
nbouillot@sat.qc.ca

Zack Settel
Society for Arts and Technology
1201, Boulevard Saint-Laurent
Montréal H2X 2S6, Canada
zack@sat.qc.ca

Michal Seta
Society for Arts and Technology
1201, Boulevard Saint-Laurent
Montréal H2X 2S6, Canada
mseta@sat.qc.ca

ABSTRACT

Recent advances in computing offer the possibility to scale real-time 3D virtual audio scenes to include hundreds of simultaneous sound sources, rendered in realtime, for large numbers of audio outputs. Our Spatial Audio Toolkit for Immersive Environments (SATIE), allows us to render these dense audio scenes to large multi-channel (e.g. 32 or more) loudspeaker systems, in realtime and controlled from external software such as 3D scenegraph software. As we describe here, SATIE is designed for improved scalability: minimum dependency between nodes in the audio DSP graph for parallel audio computation, controlling sound objects by groups and load balancing computation of geometry that allow to reduce the number of messages for controlling simultaneously a high number of sound sources. The paper presents SATIE along with example use case scenarios. Our initial work demonstrates SATIE's flexibility, and has provided us with novel sonic sensations such as "audio depth of field" and real-time sound swarming.

Author Keywords

Audio Spatialization, Scalability, 3D engine, Control Protocol, SATIE, OSC, Blender, Unity

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing, I.3.8 [Computer Graphics] Applications.

1. TOWARD REAL-TIME HIGH-DENSITY 3D AUDIO SCENES

A growing number of computer music performance venues are now equipped with large loudspeaker configurations, and therefore provide new opportunities for artists using 3D audio scene environments for composition and sound design. Such spaces tend to be wide, offering improved spatial resolution, compared to more traditional venues typically offering planar audio displays ranging from stereophonic to octophonic. While historical composers such as Varèse, Xenakis, Stockhausen, and others pioneered spatial composition, composing computer music for "space" remains a challenge in terms of the heterogeneity of speaker configurations. Our recent work in spatial composition uses a "volumetric" approach [10], where the 3D audio scene is composed within a 3D scenegraph authoring environment, such as Unity3D or Blender, simplifying the control of spatialization parameters, thanks to behavioural descriptions of sound object location, physics in the 3D audio scene, and other scripting abilities provided by the 3D scenegraph environment, that are used for calculating the many values of spatialization parameters.

Wide listening spaces with improved spatial resolution are able to offer novel sonic sensations when displaying dense audio scenes containing hundreds of sound objects. To date however, the rendering of such scenes to high-resolution spatial audio displays was not possible in realtime–only attainable for applications using pre-rendered, off-line audio. Existing real-time 3D audio scene rendering systems, such as COSM [12], BlenderCAVE [9], Spatium [6], Zirkonium MK2 [11], CLAM [5] and 3Dj [7] do not explicitly mention audio scene density (number of simultaneous sources), and are not developed with sound object scalability in mind.

In this paper, we present SATIE and explain how it is designed to handle dense audio scenes. In its current state, albeit nascent, SATIE offers artists a new possibility to compose real-time audio/music scenes that can operate on a "symphonic scale", consisting of hundreds of simultaneous sources, and the ability to render these dense scenes to loudspeaker configurations of 32 channels or more.

The development of SATIE (with the SuperCollider language [3]) was first motivated by the need to render dense and sonically rich audio scenes for the Satosphere, a large dome-shaped audiovisual projection space at the Society for Art and Technology [SAT] in Montreal. Nearly 13 meters high and eighteen meters in diameter, the Satosphere is equipped with 157 loudspeakers grouped into 31 adjacent clusters on the dome's surface, and with 8 video projectors that likewise distribute the video image across the dome's surface. A 3D audio renderer for such a venue must produce 31 channels of spatialized output, consuming significantly more CPU processing power than systems for venues with fewer speakers (clusters). In the Satosphere, with its 31 output channels, prior to the development of SATIE, we were limited to sparse audio scenes of no more than 32 sound sources using our previous system. From a musical and qualitative point of view, this limiting factor of polyphony also limited musical range–much in the same way that you can't write a symphony if all you have is a string quartet. Rendering now with SATIE, our 3D audio scenes can be 10 or more times more dense, thus greatly expanding the potential musical range.

NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.

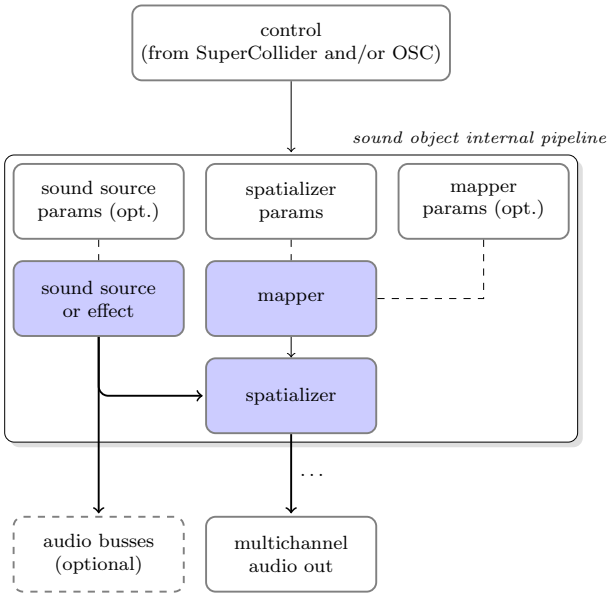


Figure 1: Sound object internal pipeline. When SATIE is factoring the sound object type, control parameters from the selected plugins (sound source, spatializer and mapper) are exposed. The audio source's mono output is made available to the spatializer, which in turn, sends multichannel audio to the audio output. The source's mono audio output can also be sent to one or more busses, providing input to particular effect nodes. Spatializer parameters can optionally be preprocessed by a mapper that modifies parameter values, before forwarding them to the spatializer.

2. IMPLEMENTATION CONCEPTS IN SATIE

SATIE is a sound server that can be controlled by the OSC protocol [16] or from a SuperCollider client. During runtime, the use of SATIE consists of the following basic tasks:

- creating/deleting groups of audio sources, or group of effects.
- creating/deleting sound objects (audio source or effect). Each sound object is named and belongs to a group, as specified when sound object is created
- controlling sound object parameters using key/value pairs, applied by name to sound objects, individually, or in groups.

Before (or during) runtime, the user specifies several named sound object and effects types to be used in a given 3D audio scene. This specification includes a sound source plugin, one or more spatialization-type plugin(s), an optional audio bus for non-spatialized audio source signals, optional spatialization parameter mapper plugin, and optionnal preloaded audio buffers for audio file sources. The sound object code (a SynthDef in the SuperCollider language) is assembled and made available for future instantiation. While we will describe these plugins in more detail later, note that plugin-specific parameters are made available as sound object parameters, and then controllable via OSC messages.

Once sound sources types are defined, they are used during runtime for the creation of sound objects[1]. This configuration pass has the advantage of enabling the reuse of all plugins in various contexts, without the need of changing runtime behaviour. For instance, loudspeaker configurations can be interchanged, allowing the same 3D audio scene to be rendered to suit the current rendering environment, at home in stereo, or elsewhere, using a large speaker configuration.

[1]SupperCollider is very good at dynamically instanciating DSP nodes without generating glitches or drop outs, probably thanks to extensive use of memory pool at the server side. This is well illustrated in SuperCollider documentation that employ this feature extensively.

2.1 Factoring sound objects from plugins

During initialization, SATIE scans plugin folders, to create and maintain dictionaries of named plugins. A plugin is actually a SuperCollider file that defines a named function. Then, a sound object type is defined by the user, who specifies a sound source (effect), and a spatializer. From this description, SATIE factors a monolithic sound object type with a pre-defined internal pipeline presented in Figure 1. Then, when requested to create a sound object, SATIE instantiates a node in the DSP graph that computes the sound and its spatial rendering. Accordingly, there is no de facto dependency between sound objects in the DSP graph, thus, effective parallel computation can be carried out using Supernova, SuperCollider's scalable parallel audio synthesis server [2].

2.2 Audio source plugins

Audio plugins have parametric control and output a mono audio signal. Currently, available plugins include physical modeling for plucked strings, various test generators, live audio input, and sound file sampling and streaming. Control parameters are specific to each plugin. In order to create a new audio source plugin, the user creates a file, and adds it to the audio source plugin folder. This plugin file is written in the SuperCollider language and requires two definitions: a name and a function. The name allows for selecting the audio source plugin when creating a new sound object type, as described in Section 2.1. The function's arguments define the audio source's control parameters (for instance frequency, trigger, position in the sound file, etc), that will be exposed to the user for control.

2.3 Spatializer plugins

Spatializer plugin inputs are pre-defined, consisting of a mono audio signal and an expandable set of low level para-

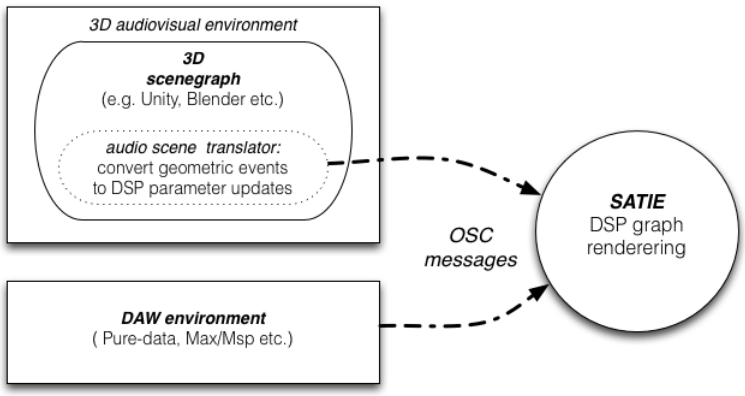


Figure 2: OSC messages control sound object parameters (individually or by group) at different levels. For controlling SATIE, 3D scenegraph can convert geometric events to lower level spatializer parameters. Geometry-related computation can be added to SATIE though the use of mappers than can provide higher level parameter control, such as for DAW environment.

metric controls, allowing for the option to process geometry from a higher level, such as an external process, or a mapper plugin (described below). All spatializer plugins define the following standard controls: azimuth, elevation, gain, delay, low pass cutoff frequency, high pass cutoff frequency, distance and spread.

SATIE comes with set of plugins for standard renderer output formats, as well as custom formats for the SAT's 24 and 31-channel dome playback systems. These plugins use SuperCollider's VBAP for the panning element, and are thus, easily modified to render to other loudspeaker configurations. Ambisonics and Wave Field Synthesis are also available in SuperCollider, and can be easily integrated as SATIE plugins.

2.4 Mapper plugins

Mapper plugins perform control parameter conversion. The mapper preprocesses input parameters (standard and custom) which are, in turn, applied to a corresponding spatializer plugin's input. A mapper plugin does not process audio. These plugins can be used for provide for non-generic control and geometric computation of lower-level spatialization parameters. For instance, the gain of a sound source may be modified as a function of azimuth, elevation and distance parameters, in order to obtain a custom radiation pattern.

2.5 Effect plugins

Effect plugins input consists of a mono audio SuperCollider bus, and custom control parameters. The plugin outputs a mono audio signal. Effect plugins thus read raw audio from other sound objects, process that audio, and generate an output signal that will be spatialized. Examples include spatially located echoes or reverberating zones.

Though effect object types are built from the same SATIE function as source objects, effects objects depend on input from source objects. For this reason, care must be taken with the order of signal computation. Thus, effect object creation must be done in a group whose member objects are added at the tail of the DSP graph, while source objects must be added to the head of the DSP graph. To facilitate this, two options (`addToHead` and `addToTail`) are available when crating a SATIE group, as it is already the case for SuperCollider groups.

2.6 Self-destructing sound objects

SATIE offers the possibility to create self-destructing sound objects (kamikaze). The creation of many kamikaze sound sources avoids the need for the SATIE client to maintain a state for monitoring these sound objects, and accordingly reduce the number of messages required for killing sound objects.

Implementation is based on SuperCollider `DetectSilence` done-action that is able to free the enclosing sound object (a synth in the SuperCollider language) when reaching a silent state. These kamikaze are particularly effective for numerous sound objects that are associated with localized events, such as a particles in a swarm, as opposed to singular virtual objects.

2.7 Controlling SATIE from OSC

Runtime management of sound objects consists of a very small set of SATIE operations, i.e. creation/deletion and setting named parameters. Only the parameters themselves are specific to a sound object. Accordingly, when a sound object's parameters are known, it is straightforward to send OSC messages that will control them, as we do from our 3D graphic authoring/rendering engines, such as Blender and Unity3D [10], for each of which, we have implemented a particular OSC protocol.

Currently we are creating more generic OSC protocol, based on the design of SATIE (SATIE/Blender protocol) and the porting of our previous work with the Soundscape [14] and SpatOSC [15] projects (SATIE/Unity3D protocol), which was inspired by the SPATdif [8] project.

In our protocols, 3D objects with sound attributes in the scenegraph are associated with SATIE's sound objects. During runtime, the synchronization of 3D objects in the scenegraph with their corresponding sound object in SATIE is accomplished via SATIE-bound OSC messages from the scenegraph for real-time creation/destruction/control/grouping of sound sources, and their parameter updates for nodes in SATIE's DSP graph, for panners, variable delays, attenuation, filtering, or synthesis parameters.

3. LOAD BALANCING PHYSICAL COMPUTATION

Today, 3D audio scene composition is broad and includes: A) music whose electronic source sounds are distributed among loudspeakers which are strategically localized in a given space (Edgard Varèse, Stockhausen and Xenakis), and B) music whose electronic sounds are rendered as sources in a virtual 3D audio scene, using physics simulators to localize each source from a unique perspective. To avoid confusion, we will refer to the former as "parametric", and the latter

	number of simultaneous live sound objects	
sound object types	8 ch. ring	our 31 ch. dome
physical model (plucked string)	337	288
sine wave	4288	1984
sound file	2964	1520

Table 1: Performance achieved with our machine (15.6GB RAM and 8x3.6GHz Intel Xeon CPU), creating and then controlling simultaneously a large number of same type sound objects.

as "volumetric" [10].

Traditional composition techniques for spatial sound are really related to the parametric approaches: creating trajectories of sound objects in three dimensional space [1], allowing reproduction of choreography of sounds among arbitrary speaker systems, thanks to reproduction technology that is computing the complex mapping between speaker placement in the space and virtual trajectories.

In the last ten years, the availability of 3D graphical has permitted the automated, or semi-automated control of trajectory, simulating spatial motion behaviour possibly affected by physical simulation of gravity, collision [13, 12, 4]. Additionally, the use of 3D engine facilitates audiovisual choreography, including distances related sonic relations among sound objects, such as Doppler shift and room acoustics. Today, 3D engines are able to scale to many 3D virtual objects, coming with authoring tools that allows for specifying trajectory motion, or high level motion specifications based on parameters such as initial speed, other object to follow, etc. This is offering a very powerful way of automating the control of spatialization parameters, defining higher level of choreography between sound objects. It is also offering the possibility to experiment interactions among a high number of sound objects and possibly with live performer interaction with the 3D environment that will be not achievable drawing trajectories in an editor.

In order to provide for both "parametric" and "volumetric" approaches, SATIE's spatializer plugins handle only low-level DSP parameters (see Section 2.3), delegating the computation of physical state to external software, and/or to mapper plugins.

However, there are cases when computation of geometric scene-graph state within SATIE is preferable, if not essential. One such case is when we wish to generate a large group of localized sound nodes, who's positions must be calculated relative to a particular location in the scene-graph, at the framerate. The reception of DSP parameter updates for each sound node, via OSC could create significant bottleneck. To avoid this situation, SATIE offers the possibility to load balance the geometry computation using a mapper plugin, which can, for example, calculate it's source node's position in the scenegraph, based on one or several of the node's group attributes, and in turn, generate the DSP parameters for the spatializer it is coupled to. In the following section, we will demonstrate how a swarm of sound objects can be managed, using a minimum of SATIE-bound scenegraph OSC update messages.

4. USE CASE: A PARTICLE SWARM

Musicians and sound artists have only recently started using 3D scenes for the computation of audio and music. This approach has largely been limited to sonic interaction between singular sound source objects and listeners. Using SATIE to render audio scenes can extend sonic interaction to include 3D audiovisual particle systems that are animated by the physics engine, containing hundred of particles that can sonically interact with scene geometry (i.e. flow, collision, bouncing etc.). This possibility opens up the door to the use of swarming and other such algorithms as a way to organize sonic material for creating music or sound design.

3D scenegraph environments such as Unity3D, provide access to individual particles, and thus allow for spatialization and event triggering of each particle. The environment can operate on potentially dense scale–a scale that can quickly exhaust the computing resources of a given audio renderer, particularly if it is running as a separate process and spatializing to 32+ loudspeakers.

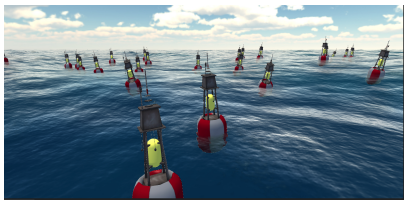
Rather than limit the number of particles to render individually, we render small groups of particles, and thereby increase our particle density by an order of magnitude. Thus, in the scenegraph, we define particle systems of few particles. Each particle belongs to a corresponding group in SATIE, containing 10 or 20 surrounding clones, each clone statistically deviating from the group's "master" particle. This deviation is applied to spatialization and temporal event scattering, and can be dynamically controlled as a function of the particle's context in the 3D scene, such as its distance from the listener, normal of impact, or collision with another body in the 3D scene. In this manner, we can render audiovisual clouds of hundreds of particles that still can be controlled with a high degree of granularity, and remain spatially coherent from a listening perspective.

Particle collisions with resonant bodies is of particular interest, as it allows us to *musically* explore the particle/collision dynamics of things like rain falling on a tin roof, or maracas. For the implementation of particle clone groups, it is necessary to extend the SATIE spatializer to provide for dynamically controlled random deviation. The mapper is loudspeaker configuration agnostic, and allows for parametric control of the deviation that is generated and applied to the spatializer's DSP parameters for incidence, gain, delay and filtering; for example, the mapper can use the distance of the group's "master" particle to the "listener" when calculating these parameters. When that parameter is quite large, the amount of random spatial deviation can be relatively small, and vice versa. Inside the mapper, we can calculate the amount of deviation for azimuth, by using the distance projected on the horizontal plane. Thus, the mapper can provide a solution for cases for spatialization that uses arbitrary scenegraph geometry.

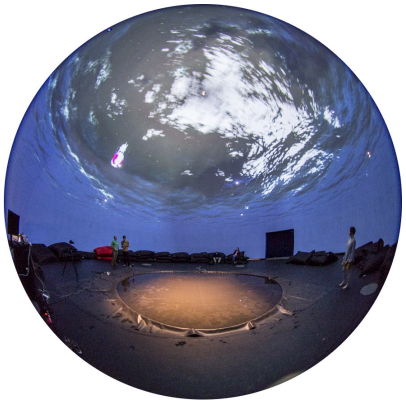
Using the SuperCollider interpreter, and its object oriented programming environment, classes for sound node control processing can be defined and deployed. Incoming node messages are received, action is taken, and associated nodes in SATIE's core are updated. For example, the management of the particle node clone groups, described above, is handled by processes that act on messages to a given group's "master" particle, and subsequently update clones of that group accordingly.

5. PERFORMANCE

Mainly thanks to supernova, the scalable parallel audio synthesis server for SuperCollider [2], running on a standard multi-core computer, SATIE's audio scene can scale to in-



(a) Initial scene with ambient sounds and ringing buoys.



(b) Fish-eye view of the Aqua Khoria stage. The top of the picture shows the projection of the water top when viewed from underwater. The white object floating on the water is one of the buoys seen in Figure 3a. The circular pond can be seen on the ground. Photography by Sébastien Roy.

Figure 3: Visuals from the Aqua Khoria piece. Figure 3a is a rendered view of the beginning scene and Figure 3b shows a photography of the stage taken latter in the piece, when audience dive into the see.

clude several hundred simultaneous and independent sound sources (physically modeled, synthesized, sound file and live inputs), of which, more than three hundred may be independently controlled externally at an update rate of 10Hz. (these limits are based on our observations in practice). To date we have tested SATIE on Linux and OSX platforms, with similar performance.

SATIE benefits from the many optimization included in the SuperCollider language and runtime such as the internal use of memory pool, parallel processing with the supernova server. SATIE has been developed trying to follow to SuperCollider philosophy, encouraging buffer and bus allocations at initialization. It also encourages audio linking among audio processing units (unit generators) only at sound object creation, allowing optimization by the server.

Table 1 presents measurements when trying to increase the number of sources instantiated without starting to encounter issues like dropping audio buffers or reaching the maximum CPU usage. The measurements have been made on a Linux computer with 15.6GB RAM and an 8x3.6GHz Intel Xeon CPU. Each value in the table is obtained creating multiple sound objects of the same type. For instance, we were able to render 288 independent sound objects (each running a physical model for a plucked string) spatialized for a 31 channel dome. The results below shows an important improvement over similar measurements we have done in [10]. The differences are due to improved performance of supernova in SuperCollider version 3.7 (versus 3.6.6 used in our previous tests) and removal of JITlib from SATIE's architecture. However, we were unable to perform the test with 128 speakers using the current version of SuperCollider.

6. AQUA KHORIA

Aqua Khoria [2] is a spatial audiovisual immersive/interactive dance piece where the dancer performes in a shallow pond, located in the middle of the Satospher, our 31 audio channel dome with 360° projection (see Figure 3). Position and gesture tracking allow for the dancer to interact with a synchronized surrounding virtual 3D environment. The show was developed in Unity3D and in SATIE, where physical models for bells were implemented as SATIE plugins, generating sound in over 100 audiovisual objects in the Unity scene. On a larger movement scale, covering the inside of the dome (250 square meters) where the dancer roamed during the performance, dome-top camera was used to track an infrared light source worn by the dancer. Thus, the dancer's position in the performance space was captured and used to animate objects in the virtual scene, such as light sources, or the virtual camera. A particularly successfull result was acheived by mapping the dancer's position to that of a candle-like light source in the surrounding virtual scene; objects in the virtual scene were "lit up" as the dancer moved in their direction.

[2] `http://aquakhoria.com/`

7. DISCUSSION AND SUMMARY

We have presented various aspects of SATIE, our Spatial Audio Toolkit for Immersive Environments. Its use is based on creation, control and deletion of sound sources. The sound source parameter set is defined as the union of source (audio or effect) parameters, spatialization parameters and possible extra parameters defined by specific spatializers and/or mappers. The sound objects can be grouped, allowing for parameter updates by group, as well as group deletion. SATIE can be controlled from a SuperCollider Client or from external processes, such as Unity3D and Blender, via OSC messaging.

In this paper, particular emphasis is made about design strategies that scale well, to support hundreds of simultaneous sound objects on a single SATIE server. One such strategy is to eliminate direct dependencys on the DSP graph. We accomplish this by generating sound object types by assembling plugin code, enabling parallelization of audio computation. We also introduced a mapper plugin that allows for balancing geometry computations, reducing the number of inbound parameter updates needed to update groupes of independent but related sound sources. To illustrate this, we provide an example implementation of a particle swarm of sound objects, using low-bandwidth control from a game engine.

Our experience with SATIE's ability manage dense 3D audio scenes with hundreds of sound sources, and render them to large multi-channel hemispheric playback systems has serendipitously offered us a new and unexpected listening sensation, which we describe as a heightened sense of aural depth of field effect. While this effect could be the object of separate study, it would seem that this effect is probably linked to the perception of a listening space of great

dimension, delineated by numerous sound sources at various directions and distances (intensities) from the listener, giving rise to a well-defined aural foreground, background and space in between. We suspect that this effect depends on the audio scene density, and the number of channels of surrounding loudspeaker diffusion. Finally, with a large amount of perceivable sonic detail available, we are encouraged to experiment with the complex musical forms of a larger polyphonic scale that are associated with symphonic music. This effect is particularly apparent when rendering swarm system simulations, such as rain, and in particular, when rendering dense musical material where harmony comes into play.

From a musical point of view, the ability to organize sound using particle dynamics holds great potential–particularly when coupled to dense multi-channel audio displays. Pitch material can be assigned to each particle, giving rise to symphonic textures of complex and shifting harmonies, a la Debussy. The flow of pitch-sequenced particles in the air, and their multiple collisions with particular resonant surfaces in space offer great potential for spatial counterpoint, a la Charles Ives or Steve Reich. From a performance point of view, the flow of particles in the 3D scene can be redirected to collide with this surface or that, and thus, sound pitches or timbres pertaining to the particular surface of collision.

Stemming from our need to manage spatialized particle systems, we have added an experimental new type of object to SATIE: the "process" object, which, like source and effects objects, can be localized in space. Containing user-programmable behavoirs (running on the SuperCollider Client) for local managemet of sound or effects objects, process objects respond to inbound OSC messages and can spawn and animate the sound nodes they are associated with. Typically, a process object is defined and updated in the same way that sound or effects objects are; however, when created, a process object will automatically be assigned to its own unique group.

We are currently working on the definition of one single unified OSC protocol, and the addition of a query-based "resource" agent that can provide clients with a description for SATIE's capabilities (available plugins, audio files, etc), and better control over pre & post spatialization and effects. Currently, preparing SATIE with custom sound object types is written in the SuperCollider language.

8. ACKNOWLEDGEMENT

This work has been done at the Société des Arts Technologiques and funded by the Ministère de l'Économie, de l'Innovation et des Exportations (Québec, Canada). Thanks to the people involved into the Aqua Khoria piece: Peter Trosztmer, Susan Paulson, Osman Zeki, Romain Tavenard, Olivier Bradette, Luc Courchesne, Thea Patterson, Lee Anholt, Pierre Corsy, Olivier Rhéaume, Louis-Philippe Saint-Arnault, Joseph Lefevre, Guillaume Bourassa, Guillaume Raymond, Sean Caruso, Carl Lavoie, Emmanuel Durand, Sebastien Gravel, Martin Lapointe.

9. REFERENCES

[1] M. A. Baalman. Spatial composition techniques and sound spatialisation technologies. *Organised Sound*, 15:209–218, 12 2010.

[2] T. Blechmann. Supernova, a scalabale parallel audio synthesis server for SuperCollider. In *Proceedings of the International Computer Music Conference 2011*, University of Huddersfield, UK, August 2011.

[3] J. McCartney. Rethinking the computer music language: SuperCollider. *Computer Music Journal*, (26):61–68, 2002.

[4] R. McGee and M. Wright. Sound element spatializer. In *proceedings of the ICMC conference*, Huddersfield, UK, 2011.

[5] N. Olaiz, P. Arumi, T. Mateos, and D. Garcia. 3D-audio with CLAM and blender's game engine. In *proceedings of The Linux Audio Conference*, Parma, Italy, 2009.

[6] R. Penha and J. P. Oliveira. Spatium, tools for sound spatialization. In *Proceedings of the Sound and Music Computing Conference*, Stockholm, Sweden, 2013.

[7] A. Perez-Lopez. 3dj: a supercollider framework for real-time sound spatialization. In *Proceedings of the 21th International Conference on Auditory Display (ICAD–2015)*, Graz, Austria, July 2015.

[8] N. Peters, T. Lossius, and J. C. Schacher. The spatial sound description interchange format: Principles, specification, and examples. *Computer Music Journal*, Vol. 37(No. 1):Pages 11–22, May 2013.

[9] D. Poirier-Quinot, D. Touraine, and B. F. Katz. BlenderCAVE: A multimodal scene graph editor for virtual reality. In *Proceedings of the 19th International Conference on Auditory Display (ICAD2013)*, Lodz, Poland, July 2013. Georgia Institute of Technology & International Community for Auditory Display.

[10] Z. Settel, N. Bouillot, and M. Seta. Volumetric approach to sound design and composition using SATIE: a high-density 3D audio scene rendering environment for large multi-channel loudspeaker configurations. In *15th Biennial Symposium on Arts and Technology*, Ammerman Center for Arts and Technology at Connecticut College, New London, February 2016. 8 pages.

[11] D. Wagner, L. Brümmer, G. Dipper, and J. A. Otto. Introducing the zirkonium MK2 system for spatial composition. In *proceedings of the ICMC/SMC conference*, Athens, Greece, September 2014.

[12] G. Wakefield and W. Smith. COSM: a toolkit for composing immersive audio-visual worlds of agency and autonomy. In *Proceedings of the International Computer Music Conference 2011*, University of Huddersfield, UK, August 2011.

[13] M. Wozniewski, Z. Settel, and J. R. Cooperstock. A framework for immersive spatial audio performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 144–149, Paris, France, 2006.

[14] M. Wozniewski, Z. Settel, and J. R. Cooperstock. A paradigm for physical interaction with sound in 3-D audio space. In *Proceedings of International Computer Music Conference (ICMC)*, 2006.

[15] M. Wozniewski, Z. Settel, A. Quessy, T. Matthews, and L. Courchesne. spatosc: Providing abstraction for the authoring of interactive spatial audio experiences. In *Panel session at ICMC*, Montreal, Canada, 2012.

[16] M. Wright. Open sound control 1.0 specification. Published by the Center For New Music and Audio Technology (CNMAT), UC Berkeley, 2002.