The Spatial Sound Description Interchange Format: Principles, Specification, and Examples

Nils Peters,* Trond Lossius,† and Jan C. Schacher**

*Center For New Music and
Audio Technologies
University of California, Berkeley
1750 Arch Street
Berkeley, California 94720, USA
nils@icsi.berkeley.edu
†BEK, Bergen Center for Electronic Arts
C. Sundtsgt. 55
5004 Bergen, Norway
trond.lossius@bek.no
**Institute for Computer Music and
Sound Technology
Zurich University of the Arts
Baslerstrasse 30
8048 Zurich, Switzerland
jan.schacher@zhdk.ch

Abstract: SpatDIF, the Spatial Sound Description Interchange Format, is an ongoing collaborative effort offering a semantic and syntactic specification for storing and transmitting spatial audio scene descriptions. The SpatDIF core is a lightweight minimal solution providing the most essential set of descriptors for spatial sound scenes. Additional descriptors are introduced as extensions, expanding the namespace and scope with respect to authoring, scene description, rendering, and reproduction of spatial sound. A general overview presents the principles informing the specification, as well as the structure and the terminology of the SpatDIF syntax. Two use cases exemplify SpatDIF's potential for pre-composed pieces as well as interactive installations, and several prototype implementations that have been developed show its real-life utility.

Introduction

SpatDIF, the Spatial Sound Description Interchange Format, presents a structured approach for working with spatial sound information, one that addresses the different tasks involved in creating and performing spatial sound.

A major problem when working on spatial sound is that the methods and the resulting works are often tied to a specific system or infrastructure—for example, with regards to the software used for composition and reproduction or the available speaker arrangement and the characteristics of the physical venue. The lack of flexibility this produces impedes the exchange of pieces between different venues, the mixing of different tools for authoring or performing the piece, and ultimately the preservation of the work in a sustainable form that is independent of the technology used to create it.

The goal of SpatDIF is to simplify and enhance the methods of working with spatial sound content. SpatDIF proposes a simple, minimal, and extensible format as well as best-practice implementations for storing and transmitting spatial sound scene descriptions. It encourages portability and the exchange of compositions between venues with different surround-sound infrastructures. SpatDIF also fosters collaboration between artists such as composers, musicians, sound installation artists, and sound designers, as well as researchers in the fields of acoustics, musicology, sound engineering, and virtual reality.

SpatDIF strives to be human-readable, easily understood and unambiguous, platform- and implementation-independent, extensible, and free of license restrictions.

SpatDIF's applications are not limited to the sound-scene concept alone. With both its ability to communicate time-independent metadata and its extensibility with further types of data descriptors, the format is open to other related fields such as sound synthesis, compositional algorithms, and abstract spatial geometry.

SpatDIF is developed as a collaborative effort and has evolved over a number of years. The online community and all related information can be found at www.spatdif.org.

Computer Music Journal, 37:1, pp. 11–22, Spring 2013
doi:10.1162/COMJ_a_00167

History and Progress

The term SpatDIF was coined in 2007 when Peters, Ferguson, and McAdams stated the necessity for a format to describe spatial sound scenes in a structured way, since at that time the available spatial rendering systems all used self-contained syntax and data formats. Through a panel discussion (Kendall, Peters, and Geier 2008; Peters 2008) and other meetings and workshops, the concept of SpatDIF has since been extended, refined, and consolidated.

After a long and thoughtful process, the SpatDIF specification was informally presented to the spatial sound community at the International Computer Music Conference in Huddersfield in August 2011 and at a workshop at the Technische Universität Berlin in September 2011. The responses in these meetings suggested the urgent need for a lightweight and easy-to-implement spatial sound scene standard, which could contrast with the complex MPEG specification (Scheirer, Vaananen, and Huopaniemi 1999). In addition, many features necessary to make this lightweight standard functional were put forward—for example, the capability of dealing with temporal interpolation of scene descriptors. The feedback from these meetings and numerous discussions prompted the writing of this overview of the revised SpatDIF specification. This article is a revised version of an earlier conference paper (Peters, Lossius, and Schacher 2012).

Other Initiatives

Over the years several formats and frameworks have been proposed with the goal of platform-agnostic playback and re-usability of scene elements. With the introduction of MPEG-4 AudioBIFS (Scheirer, Vaananen, and Huopaniemi 1999) by the audio industry, a comprehensive format for sound scene description, multimedia content creation, and delivery was established. According to Geier, Ahrens, and Spors (2010), however, no complete implementation of the MPEG-4 system is available, because the MPEG-4 specification is large and hard to implement.

Spatial sound libraries such as OpenAL, FMOD, Wwise, and irrKlang primarily target the gaming market. They are easy to integrate, but lack a number of music-performance-related features and the flexibility necessary for artistic work.

Furthermore, proprietary object-based audio formats are also under development in the cinema industry, such as those by Dolby (2012), or Iosono (Melchior 2010). The specification by Creative Technology (2009) for controlling spatial audio content using the MIDI protocol also seems to be tailored towards cinema applications, as is discernible in the way it facilitates (for example) the "fly-by" trajectories (front-to-back and back-to-front movements) typical of action movies.

Partially inspired by the Virtual Reality Modeling Language (VRML) and X3D (Web3D Consortium 2004), several groups have presented XML-based scene description formats—for instance, Hoffmann, Dachselt, and Meissner (2003), Potard and Ingham (2003), or Geier, Ahrens, and Spors (2010).

Based on the binary Sound Description Interchange Format (SDIF), Bresson and Schumacher (2011) presented a workflow for interchange of sound spatialization data primarily used for algorithmic composition applications. Recently, Wozniewski, Quessy, and Settel (2012) presented Spat-OSC, a C++ library that circumvents the development of an interchange format altogether by communicating directly with a number of specific rendering interfaces through a dedicated Open Sound Control (OSC) syntax.

A Stratified Approach

When dealing with spatialization in electroacoustic composition or linear sound editing, the workflow comprises a number of steps necessary to construct, shape, and realize the spatial qualities of the work. After analyzing the current paradigms in spatial sound processing, a stratified approach to sound spatialization was proposed (Peters et al. 2009) that encourages the use of a clear structure, flexibility, and interoperability. We identified the underlying common elements present when sound spatialization is used, and structured them in a tiered model that comprises six functionally distinct layers, as will be later shown in Figure 3.

The topmost *authoring layer* encompasses the abstract creation processes that describe how and when audio content might be positioned and moved within

the space. A number of different processes and methodologies can be envisioned that avoid a need to directly control underlying audio processes. Some examples are symbolic authoring tools, generative algorithms, interactive processes, and simulations of emergent behaviors (e.g., swarms or flocks-of-birds). The *scene description layer* mediates between the authoring layer above and the *encoding layer* below through an abstract and independent description of the spatial scene. This description can range from a simple static scene with one virtual sound source up to complex dynamic sound scenes including multiple virtual spaces. In the proposed model, the actual spatial sound rendering is considered to consist of two layers. The *encoding layer* produces encoded signals containing spatial information while remaining independent of, and "agnostic" of, the speaker layout. Processing of sources to create an impression of distance, such as Doppler effect, gain attenuation, and air absorption filters, are considered to belong to the encoding layer, as is the synthesis of early reflections and reverberation, as demonstrated by surround effects that use Ambisonic B-Format impulse-response convolution. The *decoding layer* interprets the encoded signals and decodes it for the speaker layout at hand. Not every rendering technique generates intermediate encoded signals: Some can instead be considered to encapsulate the encoding and decoding layers in one process. Ambisonics B-Format, higher-order Ambisonics (HOA), and Directional Audio Coding (DirAC) (Pulkki 2007) are examples of spatialization techniques providing intermediate encoded formats, while wavefield synthesis (WFS), Vector Base Amplitude Panning (VBAP) (Pulkki 1997), Distance-based Amplitude Panning (DBAP) (Lossius, Baltazar, and de la Hogue 2009), Virtual Microphone Control (ViMiC) (Braasch, Peters, and Valente 2008) and Ambisonics equivalent panning (Neukom and Schacher 2008) are examples of techniques without an intermediate sound representation. The *hardware abstraction layer* provides the audio services that run in the background of a computer operating system and manage multichannel audio data between the physical devices and higher layers. Finally, the *physical device layer* defines the electrical and physical specifications of devices that create the acoustical signals, such as sound cards, amplifiers, loudspeakers, and headphones.

SpatDIF Structure

SpatDIF presents a hierarchical, unambiguous structure. The SpatDIF-syntax serves to structure information related to sound scenes.

The SpatDIF Philosophy

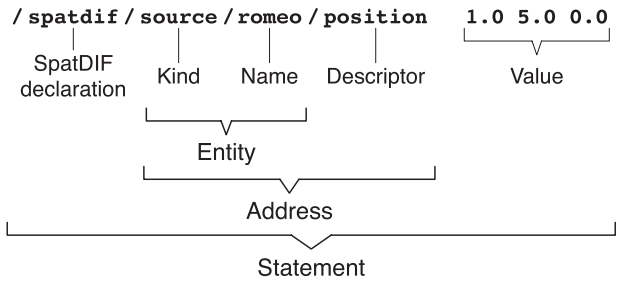
From the very beginning, one of the guiding principles for SpatDIF was the idea that the authoring and the rendering of spatial sound might occur at completely separate times and places, and might be executed with tools whose capabilities cannot be known in advance. The goal was to formulate a concise semantic structure that is capable of carrying the necessary information, without being tied to a specific implementation, thought model or technical method. SpatDIF is a syntax rather than a programming interface or file format. SpatDIF may be represented in any of the structured mark-up languages or message systems that are in use today or in the future. Examples of streaming SpatDIF data (via OSC) and storing it (via XML, YAML, or SDIF) accompany the specification in a separate document.

SpatDIF describes only the aspects required for the storage and transmission of *spatial information*. A complete work typically contains additional aspects that exceed the scope of SpatDIF. These are only addressed to the extent necessary for linking the elements to the descriptions of the spatial aspects (e.g., the Media resources).

Terminology

A SpatDIF *scene* is the combination of a space and the sounds and their behaviors that are unfolding within it. A scene consists of a number of SpatDIF *entities*. Entities are all objects that affect or interact with the sound of that scene. Entities can be of different *kinds* (e.g., sources or sinks). Each entity instance is assigned a *name*, so that it may be uniquely identified within the scene. The properties of entities are described and transmitted via SpatDIF *descriptors*. A complete SpatDIF statement consists of an *address* unambiguously identifying an entity and the desired descriptor, followed by the descriptor's associated *value*. The values associated with descriptors may

Figure 1. SpatDIF terminology.



change over time. All entities and descriptors are defined within the SpatDIF *namespace*.

OSC addresses, for example, need to comply with the SpatDIF namespace in order to be valid SpatDIF statements. An OSC message such as `/src/1/pos 1.0 5.0 0.0` is considered invalid, because neither the kind `src` nor the descriptor `pos` are defined within the SpatDIF namespace.

Figure 1 shows a valid SpatDIF statement in streaming OSC style: The entity is of kind `source` and is named `romeo`, the `position` descriptor is addressed, and the vector {`1.0 5.0 0.0`} is its value.

SpatDIF Specification

This section provides a brief overview of the current SpatDIF specification (Peters, Schacher, and Lossius 2010–2012).

Meta and Time Sections

A SpatDIF scene can consist of two sections: a *meta section* and a *time section*. The meta section serves to configure and initialize the system, and the time section describes the temporal unfolding of a scene.

Meta Section

The meta section contains meta descriptions, and is located at the beginning of a SpatDIF representation. It contains information that is not executed at runtime; timed events are therefore excluded from this section. The meta descriptions contain information about: extension setup (see the "Core and Extensions" section of this article), the organization of the subsequent time section, and general annotation and documentation, such as documentation about the technical setup of the original authoring situation.

The meta section can also be used to describe a static scene or the initial state of a dynamic scene. The meta section is mandatory for a SpatDIF representation.

Time Section

The time section holds information about entities and their descriptors as they unfold over time. Each statement is located at a specific point in time. If the scene to be described is static, no temporal data will be required. For this reason, the time section is optional.

SpatDIF does not enforce any particular system for ordering the statements within the time section. Standard musical notation shows that several ordering principles exist. Ordering by time is equivalent to an orchestral score and provides a complete overview, and ordering by entities groups the statements into individual parts or tracks. In the context of real-time streaming of scenes, ordering by time is necessary, although other ordering principles may be more appropriate in storage-type scenarios.

Core and Extensions

On the one hand, a standard for interchange of spatial scenes faces the challenge of having to offer a compact method for the description of works in a lightweight format; on the other hand, it has to cater to more-advanced techniques and various spatialization methods in an extensible way. SpatDIF solves this by defining a set of *core* descriptors and various *extensions*.

The SpatDIF Core

The most basic SpatDIF namespace is defined in the SpatDIF *core*. The core provides the most essential, yet extensible set of functionalities for describing spatial sound scenes. In terms of the layered model for spatialization as discussed later, the core only deals with the most fundamental descriptors required at the scene description layer (layer 5).

A SpatDIF compliant sound renderer must understand and interpret all core statements.

Table 1. Core Descriptors for Source Entities

Descriptor	*Data type*	*Default value*	*Default unit*	*Alternative units*
`type`	1 string	`point`	—	—
`present`	1 boolean	`true`	—	—
`position`	3 double	0. 0. 0.	`xyz`	`aed, openGL`
`orientation`	3-4 double	0. 0. 0.	`euler`	`quaternion, angle-axis`

Source entities are the most essential elements in sound scenes. As can be seen in Table 1, only the most basic source descriptors are provided by the core. This table serves as an example to show how entities are defined in the SpatDIF specification.

A sound source is further specified by the *type* descriptor. The core only describes point sources, therefore `point` is the default and only possible value. Extensions introduce additional types in order to describe more complex kinds of sources, as will be discussed in the section on extensions. A source can be dynamically added or removed from a scene by means of the boolean descriptor *present*. The six degrees of freedom of a point source are described by means of the *position* and *orientation* descriptors. Position as well as orientation can be expressed using a number of defined coordinate systems, thus allowing the description of the scene in a flexible yet unambiguous way. By default, position is described using Cartesian coordinates and orientation is described using Euler angles. Conversions between the different coordinate systems and units are provided with the specification.

The *media* resource provides descriptors to assign media content to the source entities. The SpatDIF core supports three types of media resources: live streams, live audio input, and sound files. In addition, the type can be set to `none`.

A number of meta-descriptors are defined in the core, primarily for use in the meta section. These include *annotation* for comments; *info* on author, session, location, etc.; and what *extensions* are used within the SpatDIF scene, as discussed later.

The core provides two *time methods* that simplify the description of common patterns of change over time: *Interpolation* and *Loop*. These two general methods may be applied to any descriptor that evolves over time (e.g., position, rotation, or even playback of a sound file). They can also simplify the process of describing a scene and can improve readability by thinning out data to reveal common underlying patterns. Interpolation enables up-sampling of temporally sparse information.

If a value can be described by several different units, interpolation is performed according to the unit of the target value. For example, if a target position is defined in Cartesian coordinates, the interpolation will be also performed in Cartesian coordinates. Based on five position statements, Figure 2 shows the differences in the trajectory resulting from linear interpolation in Cartesian vs. spherical coordinates.

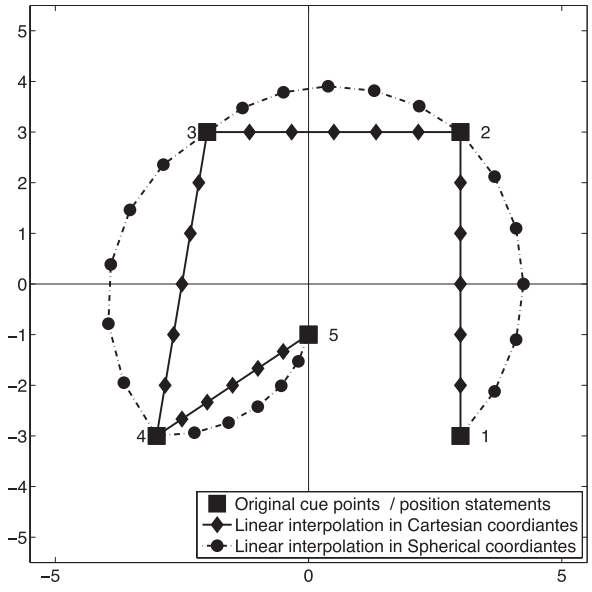
Extensions

When using only the SpatDIF core, information vital to a faithful reproduction of a spatial sound project may have been simplified or left out. For example, the SpatDIF core lacks support for directional sources and doesn't describe how spatial sound is rendered. It is important that additional information can be included and that details about the original performance and intentions can be stored for future reinterpretation or restoration of a work. For instance, maintaing precise information about all layers of the spatial workflow may be important for studying spatial sound performance practice (Harley 1994; Peters, Marentakis, and McAdams 2011).

SpatDIF extensions introduce additional descriptors in a modular way. The extensions expand the namespace and scope of SpatDIF in relation to authoring, scene description, rendering, and diffusion of spatial audio.

This permits the use of new descriptors addressing the remaining layers of the stratified model, and also enables a richer description of the spatial scene at layer 5.

Figure 2. Linear interpolation of five position statements in Cartesian and spherical coordinate systems.



Extensions might introduce new kinds of entities, expand the set of descriptors for existing entities, or augment descriptors defined by other extensions.

Extensions may also address meta-descriptors or extend and introduce new *time-methods*.

When a SpatDIF project makes use of extensions, the meta section is required to declare what extensions are present. Thus it becomes immediately apparent what rendering capabilities are necessary for a complete representation of the scene.

Support for extensions is optional: A renderer is not required to be able to act on extensions, and might only support a subset of all available extension information. If a renderer is unable to process the statements of a specific extension, it is expected to fail gracefully. A renderer without any extension support, for example, might treat all types of sources as the default point sources, and process only the core descriptors `present`, `position`, and `orientation`.

Figure 3 illustrates a number of extensions that are currently being considered, organized by the layer they belong to. New extensions will initially be developed and validated as a collaborative effort within the SpatDIF community, drawing on experts within the relevant fields. As the definition of an extension reaches maturity, it will be added to the SpatDIF specification.

To facilitate storage of data that is otherwise unsupported by SpatDIF core and extensions, a *private* extension is defined. It serves a purpose similar to that of System Exclusive (SysEx) messages within the MIDI specification. Because the syntax and semantics of the private extension are unspecified, its use severely hampers interoperability of SpatDIF scenes—a key goal of the SpatDIF project. It is therefore urgently recommended to abstain from using the private extension and rather make the effort to provide a generally useful new extension.

Additional Conventions

SpatDIF is governed by some additional general conventions. In the specification, a default state is defined for all relevant entities. An initial value may be explicitly set in the meta section, and it will override the default. This state can be further altered by subsequent statements in the time section of the scene. Entities maintain an internal state—when new statements are received, untouched descriptors remain the same. The *present* flag is an exception; please refer to the specifications for details.

Descriptors have a default unit, if applicable. Alternative units may be used as defined in the specification. For example, several alternative coordinate systems are supported and can be used interchangeably. The default system is the Cartesian Navigational System with x to the right, y in the forward direction, and z pointing upwards.

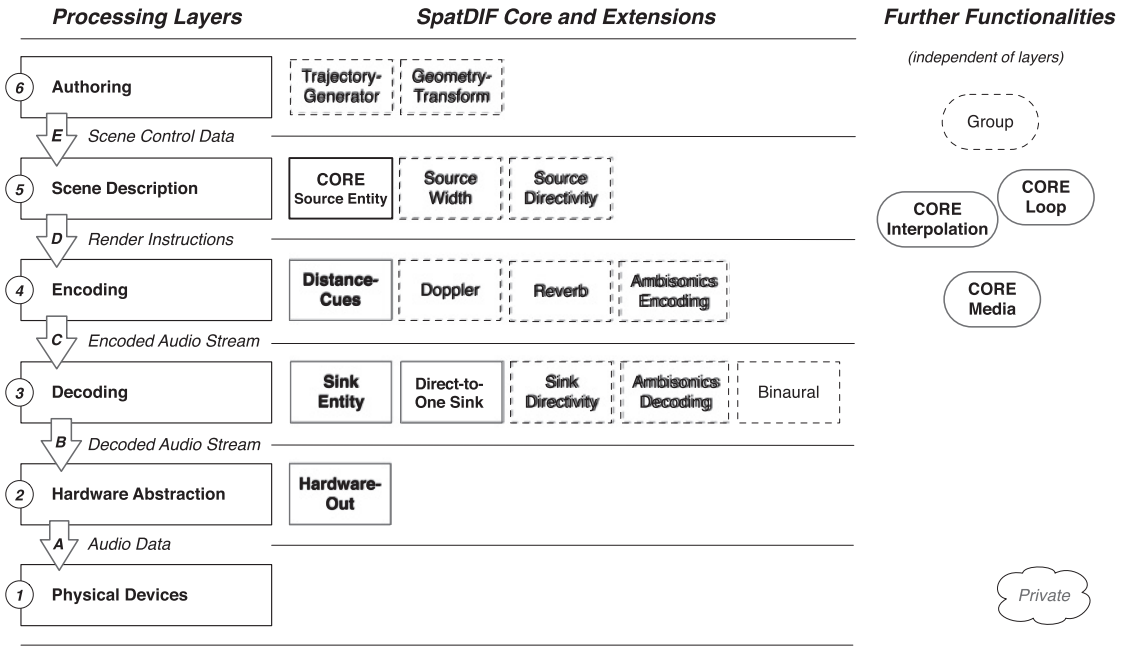
Use Cases

The following use cases illustrate two different applications of the SpatDIF standard.

File-Based Score: *Turenas*

In 1972 John Chowning completed *Turenas*, one of the first computer music compositions to create the impression of moving sound sources in a 360-degree space. A quadraphonic tape piece composed using the non-real-time sound-synthesis language Music IV, it is famous for its use of Lissajous figures as sound trajectories (Chowning 2011). In 2011 Laurent Pottier recreated it as a performance patch for the real-time processing environment Max/MSP. We

Figure 3. The layer model of the SpatDIF namespace. Extensions with a dashed frame are under development.



analyzed this patch to demonstrate how SpatDIF can be beneficial in this score-based context.

The main score of Pottier's rendition of *Turenas* is stored in a table containing sound-generation parameters for the patch's FM synthesis modules. There are additional trajectory cues that are triggered from the main score. Each point in these trajectories consists of: the gains of the four loudspeakers, to form the panning angle from which a sound source appears; a gain factor along with the relative contributions of the direct and reverberant signals, to form distance cues; and a pitch shift, to simulate the Doppler effect of a moving source. These eight values define the spatial properties of a source at any given point in time. Table 2 illustrates part of such a trajectory, in this case the beginning of the Lissajous curve used for spatializating an insect-like sound at the beginning of *Turenas*.

The trajectories consist of 60 to 120 discrete sampling points. At runtime, a linear interpolation in Cartesian coordinates is used for a smooth transition from one sampling point to the next. Because all sounds are synthesized by the performance patch, the tempo of the performance can be altered—for example, in order to accommodate the reverberant conditions of the venue.

Referring to our stratified approach in Figure 3, the *Turenas* score can be considered as channel-based rendering instructions for creating a decoded audio stream (stream B in Figure 3) on the hardware abstraction layer. Note the special requirements (in Pottier's version) of an external four-channel reverb unit. Such channel-based instructions pose a challenge to the adaptation of the piece to other loudspeaker configurations.

The score itself does not specify in which arrangement and sequence the four loudspeakers are to be placed. With the knowledge of the loudspeaker positions, however, and by applying an equal-power panning law to the gain values from Table 2, we were able to "reverse-engineer" the trajectory (see left plot of Figure 4).

Using the `position` descriptor of the SpatDIF core, the sampling points of Table 2 can now be described in the time section (using a stream-based OSC style):

```
/spatdif/time 0.0
/spatdif/source/insect/position  0.00  7.99  0.0
/spatdif/time 1.0
/spatdif/source/insect/position  2.92  6.96  0.0
/spatdif/time 2.0
/spatdif/source/insect/position  4.41  4.81  0.0
```

Table 2. Example of a Trajectory Template in the *Turenas* Patch

Cue Point #	*Loudspeaker gains*				*Gain factor*	*Direct amount*	*Reverb amount*	*Pitch shift*
	1	*2*	*3*	*4*				
1	0.7071	0.7071	0	0	0.2859	0.5347	0.4653	0.9773
2	0.7739	0.6333	0	0	0.2998	0.5475	0.4525	1.0314
3	0.8521	0.5234	0	0	0.3443	0.5867	0.4133	1.0884
4	0.9600	0.2801	0	0	0.4211	0.6489	0.3511	1.1109
5	0.8852	0	0	0.4652	0.4886	0.6990	0.3010	1.0660
6	0.6881	0	0	0.7256	0.4646	0.6812	0.3188	0.9800
7	0.5218	0	0	0.8531	0.3959	0.6292	0.3708	0.9347
8	0.4213	0	0	0.9069	0.3531	0.5942	0.4058	0.9454
:	:	:	:	:	:	:	:	:

Notated in the file-based YAML style and using spherical coordinates, the same description would be:

```
spatdif:
    version: 0.3
    meta:
        extensions:
        - distance-cues
        - doppler
        ordering: time
    time:
    - time: 0.0
      source:
       - name: insect
         position: -0.0 0.0 7.99 aed
    - time: 1.0
      source:
       - name: insect
         position: 22.8 0.0 7.55 aed
    - time: 2.0
      source:
       - name: insect
         position: 42.5 0.0 6.52 aed
```

Note that the resulting trajectory will be slightly different when interpolating in spherical instead of Cartesian coordinates (c.f. Figure 2). In this example, the source is moving along arced trajectories rather than straight line segments. Choosing the appropriate coordinate system can be a powerful yet simple means of describing a variety of trajectory patterns, which might run either in straight lines or in circular or otherwise curved arcs.

To encode the distance attenuation within the rendering process, the SpatDIF *distance-cues* extension is needed. Using this extension, the distance attenuation is computed based on the distance from the sound source to the origin of the coordinate system. At the same time, this distance information can be used for regulating the wet/dry ratio of the reverb. Similarly, using the *doppler extension*, a pitch shift can be described.

According to Chowning (2011), the sampling points shown in Table 2 were derived from the Lissajous curve in Equation 1 with additional scaling and translation (see right plot of Figure 4).

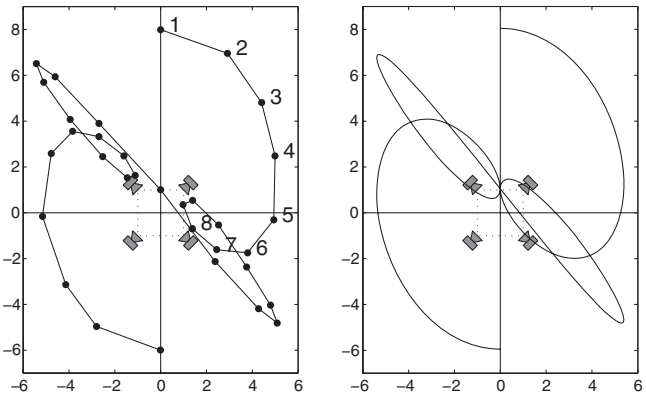
$$x = \sin(2\pi \cdot t) + \sin(6\pi \cdot t) \qquad (1)$$
$$y = \cos(3\pi \cdot t) + \cos(7\pi \cdot t)$$

By using the still unfinished *Trajectory-generator* and *Geometry-transform* extensions from the authoring layer, the Lissajous figures could be stored in an abstract mathematical representation and then rendered in any desired accuracy.

Real-Time Stream-Based System: *Flowspace II*

A number of scenarios for using SpatDIF in real time can be envisioned. The obvious use case is live control of spatialization during performance, using a joystick or a MIDI console. A further use case is a remote, co-located performance, where the state of two sound scenes is synchronized across a network. Finally, any system that generates control data on the fly can benefit from streaming the spatialization

Figure 4. Left: Reconstructed Lissajous trajectory from the cue points in Table 2. Right: The Lissajous curve based on Equation 1. The loudspeakers are placed in a square facing inward toward the listener. The virtual space is measured here in Cartesian coordinates, where the unit is the distance of each speaker from the x *and* y *axes.*



information using SpatDIF, especially when working in a modular fashion.

This last use case is exemplified by the interactive audiovisual installation *Flowspace II* by Jan Schacher, Daniel Bisig, and Martin Neukom. The work was shown in the fall of 2010 at the Gray Area Foundation for the Arts in San Francisco as part of the group exhibition "Milieux Sonores" (Maeder 2010). The installation presents the visitor with a dodecahedron of four meters in diameter. Located in its vertices are 20 inward-facing speakers, creating a regular spherical speaker array (Bisig, Schacher, and Neukom 2011), as can be seen in the video documentation of the work, available at www.jasch.ch/flowspace.html. The installation deals with simulation of lifelike autonomous systems (Schacher, Bisig, and Neukom 2011). Three different artworks are shown, each based on a different swarm simulation whose specific flocking algorithm controls both a musical and a visual composition. The swarm simulations themselves can be manipulated by the visitors using a large touch surface.

The interesting aspect in the context of this article is the system architecture used with the three simulations, each controlling a sonic and visual rendering (see Figure 5). The components of the system are written in different software environments and change depending on which piece is running. All control information flows through the network, encoded as OSC messages. The control and simulation parts fulfill the role of the authoring layer (layer 6) of the stratified approach. Control data from the intermediate interpreter layer are transmitted to the Synth and Audio-Renderer. The setting of state of the filter bank—a necessary component for driving the speaker array—is controlled by the master state-machine using SpatDIF layer 3 commands. The SpatDIF streams provide the common interface for these modules.

Figure 6 shows the SpatDIF stream with the two types of source which are derived from the flocking algorithm. The first type are members of the primary flock and are called "agent," and the second type belong to a secondary flock and are accordingly called "secondary." A piano sound is played when one agent bumps into another or reaches the edge of the flock.

Implementations

The definition of the SpatDIF specification is closely linked to simultaneous software developments in the community, which fully or partially support the SpatDIF namespace.

Jamoma is a layered architecture of software frameworks for interactive systems, mainly targeting the real-time processing environment Max (Place, Lossius, and Peters 2010). Jamoma offers extended support for sound spatialization and provides modules for a number of rendering techniques. A standardized interface of SpatDIF-compliant OSC messages is used for communicating source and speaker positions to all of these modules, making it easy to substitute one rendering technique with another when developing spatial sound content for artistic and research purposes. As previously discussed, the SpatDIF specification allows statements to be expressed using alternative units for values. Depending on the context, the choice of one unit or the other might improve readability or simplify the description of spatial movements over time. The Jamoma DataspaceLib implements this concept by allowing parameters to be described using a number of alternative units (Place et al. 2008b). Spatial positions, for example, can be described using Cartesian or spherical coordinates, as well as a number of other options. The RampLib enables *tweening* or gradual transitions to new values (Place et al. 2008a). A prototype implementation of the SpatDIF extension for interpolations is available by combining the DataspaceLib and the RampLib.

Figure 5. Flowspace II *system, structure, and data flow.*

Figure 6. SpatDIF excerpt, formatted in the OSC style, showing two types of source from Flowspace II.

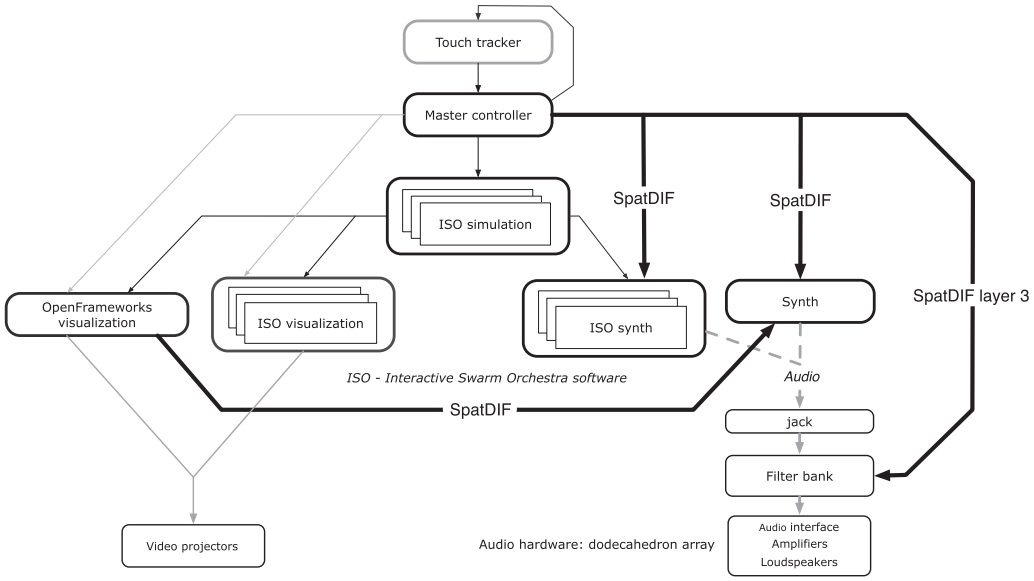


Figure 5

```
/spatdif/source/agent_01_impact/position 0.18 -0.56 0.5
/spatdif/source/agent_01_impact/media 067_piano.aif

/* at a later time */
/spatdif/source/secondary_17_impact/position 0.1 -0.31 0.48
/spatdif/source/secondary_17_impact/media 067_piano_gran.aif

/* at a later time */
/spatdif/source/agent_01_bounds/position 1.0 0.42 0.5
/spatdif/source/agent_01_bounds/media 084_piano.aif
```

Figure 6

This affords simple descriptions of a rich set of trajectories and interpolations. Interpolations between positions expressed in Cartesian coordinate can be used to create movements along straight lines in space, whereas interpolations between positions in spherical coordinates enables the description of circular and spiraling movements, as illustrated in the *Turenas* Lissajous example (c.f. Figure 4).

The Institute for Computer Music and Sound Technology (ICST) Ambisonics tools for Max (Schacher and Kocher 2006) can easily be adapted by users in order to apply OSC messages defined in the SpatDIF namespace to the control of the positions of point-sources and speakers in a sound scene. Authoring and rendering processes are both available in this tool set. The former generate, and the latter consume, messages that correspond to layer 5 for scene description, layer 4 for encoding, and layer 3 for decoding. This implementation of Ambisonics is used by a large community of musicians, researchers, and artists worldwide. The ease of use and simplicity of handling make it an ideal testbed for a typical SpatDIF workflow. In addition, the standalone digital audio workstation (DAW) editor for multichannel surround sound, "Choreographer"—also developed by the ICST—utilizes SpatDIF meta descriptors for the storage and playback of speaker settings.

For the computer-aided composition environment OpenMusic, from the Institut de Recherche et Coordination Acoustique/Musique (IRCAM), two libraries exist that provide SpatDIF support: "OMPrisma is a library for the control of sound source spatialization and spatial sound synthesis in [. . .] OpenMusic. All spatial sound rendering classes in OMPrisma share a common control interface in compliance with SpatDIF specifications"

(Schumacher and Bresson 2010). OM-Spat, a library for the creation and rendering of spatial scenes in OpenMusic, implements a storage of the source trajectories and spatial attributes using the SDIF format. Generated SDIF files can be rendered by Spat-SDIF-Player, a player application developed in Max for the representation and real-time streaming of SDIF spatial description data. As stated by Bresson and Schumacher (2011, p. 4), "The transmitted spatial description data are formatted in OSC following the SpatDIF specification and can be received, interpreted and applied to sound sources or signal generators by any external spatial sound renderer or environment compliant with this protocol."

Conclusion and Future Work

SpatDIF provides a concise syntax for describing spatial sound scenes and the additional information necessary for authoring and rendering spatial audio. A minimal, lightweight core is provided that addresses spatial scene descriptions. Although minimal, the core covers the fundamental descriptors, and is useful for a wide range of applications. For more specialized tasks such as authoring, extended descriptions, and special rendering techniques, SpatDIF is expanded by a growing number of extensions. The full specifications can be found at www.spatdif.org.

This article has presented the principles that guide the definition of SpatDIF, shown the structure of the syntax, and presented two use cases that illustrate the two main paradigms where SpatDIF is applied. Chowning's *Turenas* serves as an example of using SpatDIF in a file-based storage format. Composed prior to performance, this historical piece can benefit from the recasting in a SpatDIF description. This separates the interpretation from the original constraints and through generalization allows the piece to be played on larger speaker arrays and the use of different spatialization techniques. The stream-based transmission format, on the other hand, was discussed in connection with the audiovisual installation *Flowspace II*, where the utility of applying SpatDIF in a complex modular workflow was shown.

Future work will emphasize the development of additional extensions (see the dashed frames in Figure 3). This will be coordinated as a collaborative effort within the SpatDIF community. Interested parties are encouraged to implement the existing SpatDIF specifications within their own working environment and are invited to contribute to the process of formulating further extensions. As the number of extensions grows, it will be necessary to research in more detail how to deal with increasing complexity and to find an approach for handling the concept of "gracefully failing."

Currently, the SpatDIF workgroup is developing a reference application that implements parsing of the core descriptors and provides the basic functions required for rendering SpatDIF. This rendering engine will also serve to validate scenes and will provide a simple software solution for reproducing stored or streamed SpatDIF scenes. Furthermore, its source code will serve as a reference implementation of a SpatDIF parser.

Acknowledgments

Thanks go to the many people of the spatial sound community for their oftentimes passionate comments and suggestions, to McGill University's Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) for supporting the first author's work as a visiting researcher at IRCAM, to the Telekom Innovation Laboratories of the Technische Universität Berlin for hosting a workshop, and to John Chowning and Laurent Pottier for providing the *Turenas* performance patch. Finally, we thank the technical paper committee of the 2012 Sound and Music Computing Conference for granting an earlier version of this paper with a Best Paper Award.

References

Bisig, D., J. C. Schacher, and M. Neukom. 2011. "Flowspace—A Hybrid Ecosystem." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 260–263.

Braasch, J., N. Peters, and D. L. Valente. 2008. "A Loudspeaker-Based Projection Technique for Spatial Music Applications Using Virtual Microphone Control." *Computer Music Journal* 32(3):55–71.

Bresson, J., and M. Schumacher. 2011. "Representation and Interchange of Sound Spatialization Data for Compositional Applications." In *Proceedings of the International Computer Music Conference*, pp. 83–87.

Chowning, J. 2011. "Turenas: The Realization of a Dream." In *Actes des 17e Journées d'Informatique Musicale*. Available online at jim2011.univ-st-etienne.fr/html/actes/05_johnturenas2b.pdf. Accessed December 2012.

Creative Technology. 2009. "Three Dimensional Sound Controllers (RP-049)." Technical report, MIDI Manufacturers Association. Available online at www.midi.org/techspecs/rp49public.pdf. Accessed October 2012.

Dolby. 2012. "Dolby Atmos: Next-Generation Audio for Cinema." White paper, Dolby Laboratories, Inc., San Francisco, CA.

Geier, M., J. Ahrens, and S. Spors. 2010. "Object-Based Audio Reproduction and the Audio Scene Description Format." *Organised Sound* 15(03):219–227.

Harley, M. A. 1994. "Space and Spatialization in Contemporary Music: History and Analysis, Ideas and Implementations." Ph.D. thesis, McGill University, Montreal, Canada.

Hoffmann, H., R. Dachselt, and K. Meissner. 2003. "An Independent Declarative 3D Audio Format on the Basis of XML." In *Proceedings of the International Conference on Auditory Display*, pp. 99–102.

Kendall, G. S., N. Peters, and M. Geier. 2008. "Towards an Interchange Format for Spatial Audio Scenes." In *Proceedings of the International Computer Music Conference*, pp. 295–296.

Lossius, T., P. Baltazar, and T. de la Hogue. 2009. "DBAP—Distance-Based Amplitude Panning." In *Proceedings of 2009 International Computer Music Conference*, pp. 489–492.

Maeder, M., editor. 2010. *Milieux Sonores—Klangliche Milieus. Klang, Raum und Virtualität.* Bielefeld: Transcript Verlag.

Melchior, F. 2010. "Wave Field Synthesis and Object-Based Mixing for Motion Picture Sound." *SMPTE Motion Imaging Journal* 119(3):53–57.

Neukom, M., and J. Schacher. 2008. "Ambisonics Equivalent Panning." In *Proceedings of the International Computer Music Conference*, pp. 592–595.

Peters, N. 2008. "Proposing SpatDIF—The Spatial Sound Description Interchange Format." In *Proceedings of the International Computer Music Conference*, pp. 299–300.

Peters, N., S. Ferguson, and S. McAdams. 2007. "Towards a Spatial Sound Description Interchange Format (SpatDIF)." *Canadian Acoustics* 35(3):64–65.

Peters, N., T. Lossius, and J. C. Schacher. 2012. "SpatDIF: Principles, Specification, and Examples." In *Proceedings of the 9th Sound and Music Computing Conference*, pp. 500–505.

Peters, N., G. Marentakis, and S. McAdams. 2011. "Current Technologies and Compositional Practices for Spatialization: A Qualitative and Quantitative Analysis." *Computer Music Journal* 35(1):10–27.

Peters, N., J. Schacher, and T. Lossius. 2010–2012. "SpatDIF Specification Version 0.3, Draft Version." Available online at redmine.spatdif.org/projects/spatdif/files. Accessed October 2012.

Peters, N., et al. 2009. "A Stratified Approach for Sound Spatialization." In *Proceedings of the 6th Sound and Music Computing Conference*, pp. 219–224.

Place, T., T. Lossius, and N. Peters. 2010. "The Jamoma Audio Graph Layer." In *Proceedings of the 13th International Conference on Digital Audio Effects*, pp. 69–76.

Place, T., et al. 2008a. "Addressing Classes by Differentiating Values and Properties in OSC." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 181–184.

Place, T., et al. 2008b. "Flexible Control of Composite Parameters in Max/MSP." In *Proceedings of the International Computer Music Conference*, pp. 233–236.

Potard, G., and S. Ingham. 2003. "Encoding 3D Sound Scenes and Music in XML." In *Proceedings of the International Computer Music Conference*, pp. 65–74.

Pulkki, V. 1997. "Virtual Sound Source Positioning Using Vector Base Amplitude Panning." *Journal of the Audio Engineering Society* 45(6):456–466.

Pulkki, V. 2007. "Spatial Sound Reproduction with Directional Audio Coding." *Journal of the Audio Engineering Society* 55(6):503–516.

Schacher, J. C., D. Bisig, and M. Neukom. 2011. "Composing with Swarm Algorithms—Creating Interactive Audio-Visual Pieces Using Flocking Behaviour." In *Proceedings of the International Computer Music Conference*, pp. 100–107.

Schacher, J. C., and P. Kocher. 2006. "Ambisonics Spatialization Tools for Max/MSP." In *Proceedings of the International Computer Music Conference*, pp. 274–277.

Scheirer, E., R. Vaananen, and J. Huopaniemi. 1999. "AudioBIFS: Describing Audio Scenes with the MPEG-4 Multimedia Standard." *IEEE Transactions on Multimedia* 1(3):237–250.

Schumacher, M., and J. Bresson. 2010. "Compositional Control of Periphonic Sound Spatialization." In *Proceedings of the 2nd International Symposium on Ambisonics and Spherical Acoustics*. Available online at ambisonics10.ircam.fr/drupal/files/proceedings/presentations/O18_48.pdf. Accessed December 2012.

Web3D Consortium. 2004. "Extensible 3D (X3D)." Available online at www.web3d.org/x3d/. Accessed October 2012.

Wozniewski, M., A. Quessy, and Z. Settel. 2012. "SpatOSC: Providing Abstraction for the Authoring of Interactive Spatial Audio Experiences." In *Proceedings International Computer Music Conference*, pp. 512–517.