SpatDIF: Principles, Specification, and Examples

Nils Peters
ICSI, CNMAT UC Berkeley
nils@icsi.berkeley.edu

Trond Lossius
BEK, Bergen Center for Electronic Arts
trond.lossius@bek.no

Jan C. Schacher
ICST, Zurich University of the Arts
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 audio. A general overview of the specification is provided, and two use cases are discussed, exemplifying SpatDIF's potential for file-based pieces as well as real-time streaming of spatial audio information.

1. INTRODUCTION

SpatDIF, the Spatial Sound Description Interchange Format, is a collaborative effort [A] that aims to create a format (semantic and syntactic) as well as best-practice implementations for storing and transmitting spatial audio scene descriptions.

The goal of SpatDIF is to simplify and enhance the methods of working with spatial audio content in the context of authoring, storage of pieces and their distribution, as well as performance and study of spatial music. Typical users include composers, sound installation artists, sound engineers, acousticians, virtual reality researchers, musicologists and many more. SpatDIF strives to be human-readable i.e., easily understood and unambiguous, platform- and implementation-independent, extendable, and free of license restrictions.

SpatDIF's application is not limited to the audio-scene concept alone. The ability to communicate time-independent meta-data as well as the concept of extending SpatDIF with further types of data-descriptors opens up the format to other fields such as sound-synthesis, compositional algorithms or abstract spatial geometry developments.

1.1 History and progress

SpatDIF was coined in [1] where the authors stated the necessity for a format to describe spatial audio scenes in a structured way, since at that point the available spatial rendering systems used self-contained syntax and data-formats. Through a panel discussion [2, 3] and other meetings and workshops [B], 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 audio community at the ICMC 2011 and a workshop at the TU-Berlin in September 2011 [B]. Many of the responses in these meetings suggested the urgent need for a lightweight and easily implementable spatial audio scene standard, which could contrast the complex MPEG standard [4]. In addition, many features necessary to make this lightweight standard functional were put forward, such as the capability of dealing with temporal interpolation of scene descriptors. This feedback and numerous discussions prompted the presentation in this paper of an overview of the revised SpatDIF specification.

1.2 Other Intiatives

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 [4] by the audio industry, a comprehensive format for sound scene description, multimedia content creation and delivery was established. According to [5], no complete implementation of the MPEG-4 system is available, because the MPEG-4 specification is large and hard to implement. Aimed primarily at the gaming market, spatial sound libraries such as OpenAL [6], FMOD [C], or irrKlang [D] do also exist. They are easy to integrate, but lack a number of music-performance related features and flexibility necessary for artistic work. Furthermore, a specification for controlling spatial audio content using the MIDI protocol was also developed and published [7]. Partially inspired by VRML/X3D [8], several XML-based formats have been presented, such as [5, 9, 10]. Based on the binary SDIF format, Bresson and Schumacher [11] presented a workflow for interchange of sound spatialization data mainly for algorithmic composition applications. Recently, the Spat-OSC library [12] was presented, which circumvents the development of an interchange format altogether by communicating directly with a number of specific rendering interfaces through a dedicated OSC-syntax.

1.3 A stratified approach

A stratified approach to sound spatialization was proposed in [13], encouraging the use of a clear structure, flexibility, and interoperability. The tiered model comprises 6 layers, as shown in the leftmost column of Figure 2. The lowest two layers interface with and address physical hardware. Layers 3 and 4 deal with the actual audio processing such as encoding and decoding of spatial audio. The fifth layer serves for the description of spatial scenes and the uppermost layer represents the abstract processes and algorithms used for authoring the work.

2. SPATDIF STRUCTURE

SpatDIF presents a hierarchical, unambiguous structure. The SpatDIF-syntax serves for structuring audio-scene related information.

2.1 The SpatDIF Philosophy

From the very beginning, one of the guiding principles for SpatDIF was the idea that authoring and rendering of spatial audio might occur at completely separate times and places, and 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 now or in the future. Examples of streaming (OSC) and storing SpatDIF data (XML, YAML, 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 dimensions outside the scope of SpatDIF. These are only addressed to the extent necessary for linking the elements to the descriptions of the spatial dimension.

2.2 Terminology

A SpatDIF **scene** is the combination of a space and the actions that are unfolding within it. A scene consists of a number of SpatDIF **entities**. Entities are all objects that are affecting or interacting 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**, its **descriptor** and its associated **value**. The values of descriptors may 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, since neither the kind `src` nor the descriptor `pos` are defined in the SpatDIF namespace.

Figure 1 shows a valid SpatDIF statement in streaming OSC-style: the entity is of kind `source` and named `romeo`, the `position` descriptor is set by the vector `(1.0 5.0 0.0)`, which is its value.

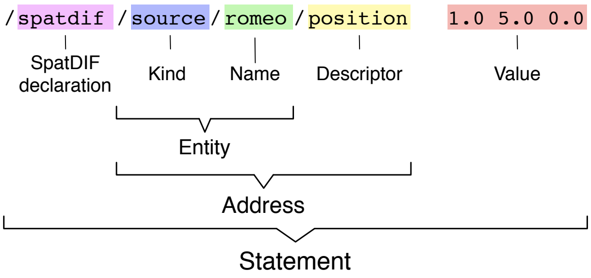

Figure 1. SpatDIF terminology

3. SPATDIF SPECIFICATION

This section provides a brief overview of the current SpatDIF specification, which can be found at [A].

3.1 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, while the time section describes the temporal unfolding of a scene.

3.1.1 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 extension setup information (see Section 3.2.2), general annotation and documentation information, information about the organization of the subsequent time section, higher-level process and compositional information and technical setup information referring to 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.

3.1.2 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, while 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, while in storage-type scenarios other ordering principles may be more adequate.

3.2 Core and extensions

A standard for interchange of spatial scenes faces the challenge of, on the one hand, having to offer a compact method for the description of works in a lightweight format, while on the other hand catering 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**.

3.2.1 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 discussed in Section 1.3, the core only deals with the most fundamental descriptors required at the scene description layer (layer 5).

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

Source entities are the most essential elements in audio 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 of how entities are being defined in the SpatDIF specification.

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`

Table 1. Core descriptors for source entities

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 discussed in Section 3.2.2. 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 different coordinate systems, thus allowing description of the scene in a more flexible, yet non-ambiguous way. Conversions between the different systems and units are provided as an addendum to the specification,

The *media* resources serve 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. This includes *annotation* for comments, *info* on author, session, location, etc., and what *extensions* are used within the SpatDIF scene, as discussed later in Section 3.2.2.

The default unit for *time* is seconds, with alternative units defined in the specification.

The core provides two *time methods* that simplify the description of common patterns of change over time: *Interpolation* and *Looping* are general methods that may be applied to any descriptor that evolves over time, e.g., position, rotation or even playback of a sound file. These time methods simplify the process of describing a scene and improve readability by thinning out data in order to reveal common underlying patterns.

Interpolation enables up-sampling of temporally sparse information.

3.2.2 Extensions

When using only the SpatDIF core, vital information to a faithful reproduction of a spatial audio 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 be included and details about the original performance and intentions be stored for future reinterpretation or restoration of a work. For instance, maintaing precise information about all layers of the spatial workflow is important for studying spatial audio performance practice [14, 15].

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

This permits the use of new descriptors addressing the remaining layers of the stratified model as discussed in Section 1.3, and also enables a richer description of the spatial scene at layer 5.

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. It thus becomes immediately apparent what rendering capabilities are necessary for the 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 *gracefully fail*. A renderer without any extension support, for example, might treat all types of sources as the default point sources, and processes only presence, position and orientation, which are core descriptors.

Figure 2 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 similar purpose to System Exclusive messages within the MIDI specification [16]. Since the syntax and semantics of the private extension are unspecified, its use severely hampers interoperability of SpatDIF scenes – a key goal of SpatDIF. It is therefore urgently recommended to abstain from using the private extension and rather make the effort to provide a generally useful new

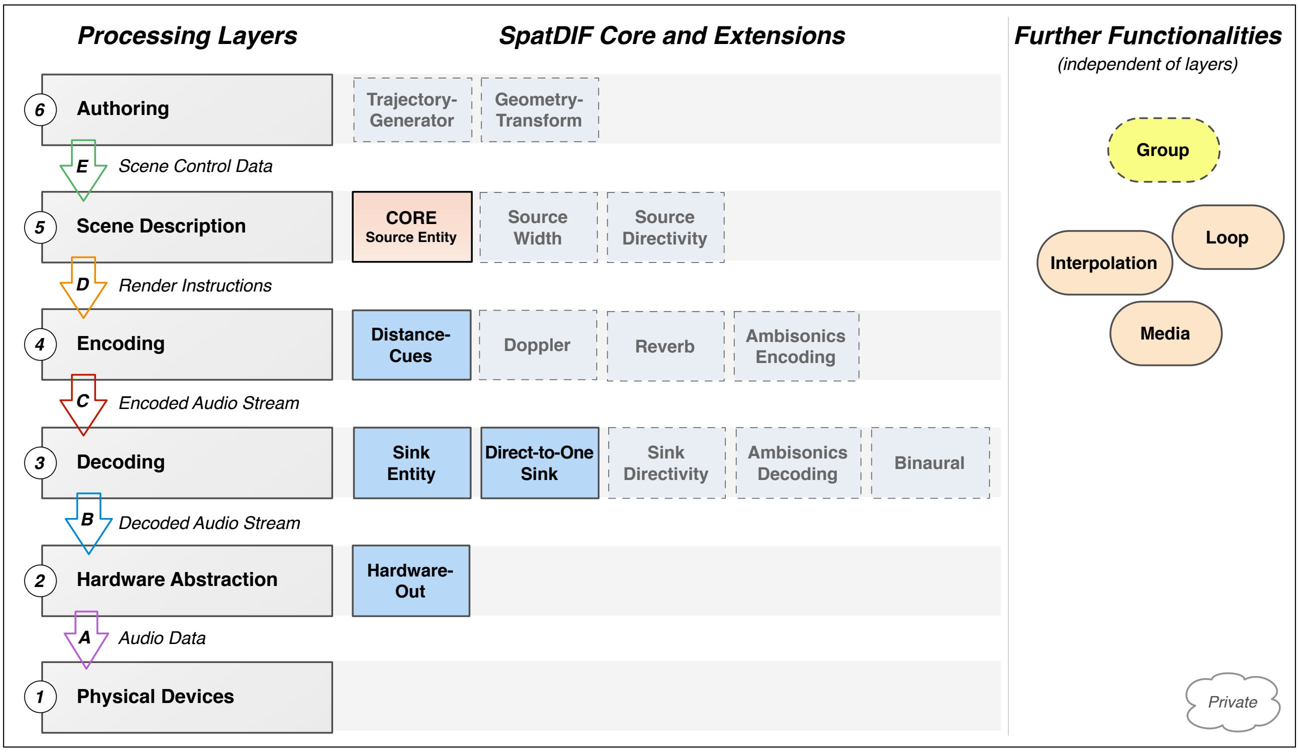

Figure 2. The layer model of the SpatDIF namespace. Extensions with a dashed frame are work-in-progress.

extension.

3.3 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 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, un-touched descriptors remain the same. [1]

All descriptors have a default unit. Alternative units may be used as defined in the specification. For example several alternative coordinate systems are supported that 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.

4. USE CASES

The following use-cases represent two applications of the SpatDIF standard.

4.1 File-Based Score: Turenas

In 1972 John Chowning completed Turenas, one of the first electronic compositions that created the impression of moving sound sources in a 360-degree space. It was composed for FM synthesizers, four loudspeaker channels and a reverberation unit. It is famous for its use of Lissajous figures as sound trajectories [15]. Although Turenas is a tape piece, it was recently recreated as a performance patch for MaxMSP [17]. We analyzed this patch to demonstrate how SpatDIF can be beneficial to this score-based context.

The main score of Turenas is stored in a table containing the sound generation parameters for the different FM synthesizers. There are additional trajectory cues which are triggered from the main score. These trajectories contain the gains of the 4 loudspeakers to form the panning angle from which a sound source appears, a gain factor and the direct-to-reverberant ratio to form distant 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 the beginning of such a trajectory, in this case the beginning of Turenas' Lissajous curve used for spatialization: an insect-like sound at the beginning of the piece.

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. Since all sounds are synthesized by the performance patch, the tempo of the performance can be altered, e.g. in order to accommodate the reverberant conditions of the venue.

	Loudspeaker gains				Gain	direct vs. reverb		Pitch
ID	1	2	3	4	factor	Ratio		shift
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,	0.96	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	1.066;
6,	0.6881	0	0	0.7256	0.4646	0.6812	0.3188	0.98;
:	:	:	:	:	:	:	:	:

Table 2. Example of a trajectory table in the Turenas patch

[1] The *present* flag is an exception, refer to the specifications for details.

Referring to our stratified approach in Figure 2, the Turenas score can be considered as channel-based rendering instructions for creating a decoded audio stream (stream B) on the Hardware abstraction layer (Layer 2)[2]. Such channel-based instructions pose a challenge to the adaptation of the piece to other loudspeaker configurations. It remains unclear from the score in which arrangement and sequence the 4 loudspeakers are to be placed.

With the knowledge of the loudspeaker positions and by applying an equal-power panning law to the gain values from Table 2, we were able to "reverse-engineer" the trajectory (see blue curve in Figure 3). 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
```

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

```
spatdif:
    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
```

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*, the amount of pitch shifting can be communicated to the renderer.

According to [15], the sampling points shown in Table 2 have been derived from the Lissajous curve in Equation 1 with additional scaling and translation. By using the *Trajectory-generator* extension and a *Geometry-transform* extension from the authoring layer, the Lissajous figures could be stored in an abstract mathematical representation and then rendered in any desired accuracy:

$$\begin{aligned} x &= \sin(2\pi \cdot t) + \sin(6\pi \cdot t) \\ y &= \cos(3\pi \cdot t) + \cos(7\pi \cdot t) \end{aligned} \tag{1}$$

[2] Note the special requirements of an external 4-channel reverb unit.

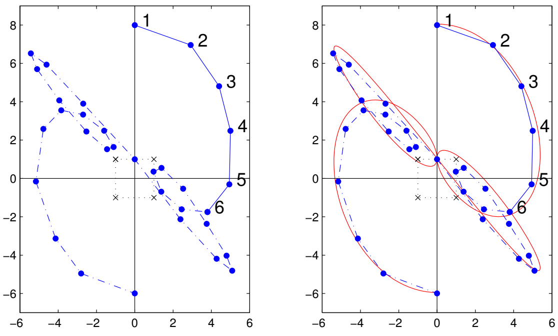

Figure 3. Left: reconstructed Lissajous trajectory from the cue points in Table 2. Black **x** mark the loudspeaker positions. Right: the Lissajous curve based on Eq. 1.

4.2 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 audio-scenes is synchronized across a network. Finally, any system that generates control data on the fly can benefit from streaming the spatialization information using SpatDIF, especially when working in a modular fashion.

This last use-case is exemplified by the interactive audio-visual installation Flowspace II by Jan Schacher, Daniel Bisig, and Martin Neukom. The work was shown in the fall of 2010 at the Grey Area Foundation for the Arts in San Francisco as part of the group exhibition *Milieux Sonores* [18]. The installation presents the visitor with a Dodecahedron of 4 meters in diameter. Located in its vertices are twenty inward-facing speakers, creating a regular spherical speaker array [19], as documented in [E]. The installation deals with simulation of life-like autonomous systems [20]. Three different art-works are shown, each composition based on a specific swarm simulation combined with a musical and visual composition, both controlled by a specific flocking algorithm. The swarm simulations themselves can be manipulated by the visitors using the touch interface.

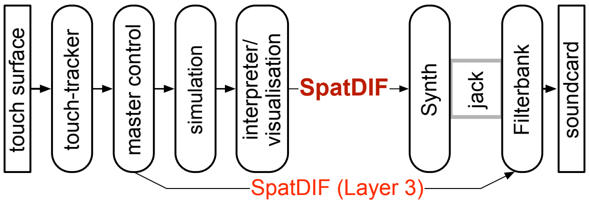

Figure 4. Flowspace II system, structure, and data flow.

The interesting aspect in this context is the systems-architecture used with the three simulations, each controlling a sonic and visual rendering (Figure 4). The components of the system are written in different software environments and change according to 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 of the stratified approach (layer 6). Control data from the intermediate interpreter-layer is transmitted to the Synth and Audio-Renderer, which stays the same. 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-stream provides the common interface for these modules

The following excerpt shows the SpatDIF stream, with two types of sources, agents and secondaries, located either at an impact-point between agents or at a point on the bounding volume of the flock. The example is formatted in the streaming OSC-style:

```
/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
```

5. CONCLUSION AND FUTURE WORK

In this paper we have presented the SpatDIF specification and the principles according to which it was designed. The full specification can be found at [A]. SpatDIF consists of a minimal, lightweight core for spatial scene descriptions. The scope of SpatDIF can be expanded by extensions.

Two use cases are presented, that illustrate the two main paradigms in which 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 permits the piece to be played on larger speaker arrays and using different spatialization techniques. The stream-based transmission format is discussed in the audio-visual installation *Flowspace II*, where the utility of SpatDIF in a complex modular workflow is shown, as well as the application of extended descriptors that address additional layers.

Future work will emphasize the development of additional extensions. This will be coordinated as a collaborative effort within the SpatDIF community, and interested parties are invited to contribute to this process. As the number of extensions grow, it will be necessary to research in more detail how to deal with increasing complexity and find an approach on how to handle the concept of "gracefully failing".

In brief, SpatDIF is ready to be implemented, used, and actively extended.

Acknowledgments

Thanks go to the many people of the spatial audio community for their often-times passionate comments and suggestions, to CIRMMT for supporting the first author's visiting research at IRCAM, to Deutsche Telekom Laboratories of the TU-Berlin for hosting a workshop and to Laurent Pottier for providing the Turenas performance patch.

References and Web Resources

[1] N. Peters, S. Ferguson, and S. McAdams, "Towards a Spatial Sound Description Interchange Format (SpatDIF)," *Canadian Acoustics*, vol. 35, no. 3, pp. 64 – 65, 2007.

[2] G. S. Kendall, N. Peters, and M. Geier, "Towards an interchange format for spatial audio scenes," in *Proc. of the International Computer Music Conference*, Belfast, UK, 2008, pp. 295–296.

[3] N. Peters, "Proposing SpatDIF - The Spatial Sound Description Interchange Format," in *Proc. of the International Computer Music Conference*, Belfast, UK, 2008.

[4] E. Scheirer, R. Vaananen, and J. Huopaniemi, "AudioBIFS: Describing audio scenes with the MPEG-4 multimedia standard," *IEEE Transactions on Multimedia*, vol. 1, no. 3, pp. 237–250, 1999.

[5] M. Geier, J. Ahrens, and S. Spors, "Object-based audio reproduction and the Audio Scene Description Format," *Organised Sound*, vol. 15, no. 03, pp. 219–227, 2010.

[6] G. Hiebert, *OpenAL 1.1 Specification and Reference*, 2005.

[7] Creative Technology, Ltd. / E-MU Systems, "Three dimensional sound controllers (rp-049)," MMA Technical Standards Board/ AMEI MIDI Committee, http://www.midi.org/techspecs/rp49public.pdf, 2009.

[8] Web3D Consortium, "eXtensible 3D (X3D)," 2004, http://www.web3d.org/x3d/.

[9] H. Hoffmann, R. Dachselt, and K. Meissner, "An independent declarative 3D audio format on the basis of XML," in *Proc. of the Int'l Conference on Auditory Display*, Boston, USA, 2003.

[10] G. Potard and S. Ingham, "Encoding 3D sound scenes and music in XML," in *Proc. of the International Computer Music Conference*, Singapore, 2003.

[11] J. Bresson and M. Schumacher, "Representation and interchange of sound spatialization data for compositional applications," in *Proc. of the International Computer Music Conference*, Huddersfield, UK, 2011.

[12] M. Wozniewski, A. Quessy, and Z. Settel, "SpatOSC: Providing abstraction for the authoring of interactive spatial audio experiences," in *to appear in Proc. International Computer Music Conference*, Ljubljana, Slovenia, 2012.

[13] N. Peters, T. Lossius, J. Schacher, P. Baltazar, C. Bascou, and T. Place, "A stratified approach for sound spatialization," in *Proc. of the 6th Sound and Music Computing Conference*, Porto, PT, 2009, pp. 219–224.

[14] M. A. Harley, "Space and spatialization in contemporary music: History and analysis, ideas and implementations," Ph.D. dissertation, McGill University, Montreal, Canada, 1994.

[15] J. Chowning, "Turenas: the realization of a dream," in *Proc. of the 17es Journées d'Informatique Musicale*, Saint-Etienne, France, 2011.

[16] The MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification. Version 96.1 Second Edition*. The MIDI Manufacturers Association, 2001.

[17] L. Pottier, "Turenas (1972) de john chowning, vers une version interactive," *Musimediane*, no. 6, p. http://www.musimediane.com/numero6/POTTIER/index.html, 2011.

[18] M. Maeder, Ed., *Milieux Sonores - Klangliche Milieus. Klang, Raum und Virtualität.* Bielefeld: Transcript Verlag, 2010.

[19] D. Bisig, J. C. Schacher, and M. Neukom, "Flowspace – a hybrid ecosystem," in *Proc. of the International Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011.

[20] J. C. Schacher, D. Bisig, and M. Neukom, "Composing with swarm algorithms – creating interactive audio-visual pieces using flocking behaviour," in *Proc. of the International Computer Music Conference*, Huddersfield, UK, 2011.

[A] `http://spatdif.org`

[B] `http://redmine.spatdif.org/projects/spatdif/wiki/Meetings`

[C] `http://www.fmod.org`

[D] `http://www.ambiera.com/irrklang`

[E] `http://www.jasch.ch/flowspace.html`

All quoted web resources were verified on July 6, 2012.