AUTHORING SPATIAL MUSIC WITH SPATDIF VERSION 0.4

Jan C. Schacher
Institute for Computer Music
and Sound Technology, ICST
Zurich University of the Arts
Zurich, Switzerland
jan.schacher@zhdk.ch

Nils Peters
Centre for Interdisciplinary
Research in Music Media
and Technology, CIRMMT
Montreal, Canada
nils.peters@mail.mcgill.ca

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

Chikashi Miyama
Institute for Music
and Acoustics, ZKM
Karlsruhe, Germany
miyama@zkm.de

ABSTRACT

SpatDIF, the Spatial Sound Description Interchange Format is a light-weight, human-readable syntax for storing and transmitting spatial sound scenes, serving as an independent, cross-platform and host-independent solution for spatial sound composition. The recent update to version 0.4 of the specification introduces the ability to define and store continuous trajectories on the authoring layer in a human-readable way, as well as describing groups and source spreading. As a result, SpatDIF provides a new way to exchange higher level authoring data across authoring tools that help to preserve the artistic intent in spatial music.

1. INTRODUCTION

SpatDIF, the Spatial Sound Description Interchange Format [1, 2], is an initiative by musicians and researchers for the development of an industry-independent, light-weight, human-readable syntax for storing and transmitting spatial sound scenes. SpatDIF addresses the lack of an independent, cross-platform and host-independent solution for spatial sound composition. It is implementation-agnostic and not tied to a specific technical platform, programming language or file-format. Representing a high-level structure that embodies typical authoring and performance work-flows in spatial sound, it comprises a hierarchical syntax of descriptors, a set of basic definitions of units and coordinate dimensions, a number of methods and algorithms for spatial sound transformations, and comes with a set of best-practice examples in several markup-languages or network streaming protocols.

This results in a non-synchronous and potentially sparse description of spatial sound scenes, that is aimed at providing interoperability between different spatial sound rendering tools and spatial music venues. Additionally it serves as a storage format for archival purposes [3]

The SpatDIF syntax is implemented in a C/C++ software library that facilitates the integration in host environments [4], for example in MaxMSP and PureData as 3rd party externals, but also in other environments such as Supercollider [5] and Zirkonium [6], as well as Android mobile platforms [7].

While integrating SpatDIF into compositional environments and artistic workflows we realized that artistic intentions of spatial movements are not fully preserved without an ability to define and store continuous trajectory information. We decided to extend SpatDIF accordingly, because to our knowledge, such features are out of scope of other open formats tailored towards commercial or broadcast applications such as ITU-R BS.2076 [8].

1.1 The Stratified Approach

SpatDIF is a structured system for describing the different aspects of a spatial sound workflow where a number of different aspect come into play. An analysis of a variety of systems and tools, as well as similarly complex transmission systems, led to the grouping and organization of the different processes and tasks into a model that comprises many, but not all elements of such a work-flow.

In 2009 the authors of [9] proposed a layered model that describes the relationships and mediation processes between essential components in sound spatialization. This model comprises processing layers (see left column of Fig. 1) ranging from the low-level Physical Domain, which comprises devices that create the acoustical signals, such as loudspeakers, up to the highest-level description on the Authoring layer, which describes the organizing and dynamic processes that drive e.g., movement within the scene. SpatDIF is based on this model and with the latest iteration presented in this paper, SpatDIF defines descriptors from the processing layer two up to processing layer six.

The previously published SpatDIF version 0.3 [2, 10] describes the core elements of a sound scene as well as some of the lower level rendering and dispatching information. The information at that stage was oriented towards rendering of spatial sound content, carrying in a temporally quantized way the instructions necessary for the playback-system to trigger sound-files or route audio signals to sources within the scene and to subsequently position these source entities in the rendered sound scene.

SpatDIF version 0.3 enables the description of the appearance and disappearance of sound entities in the scene, the assignment of media content to the entities, and the evolution of the geometrical properties of those sound entities in the scene over time such as position changes. This discretized representation of a sound scene can be practi-

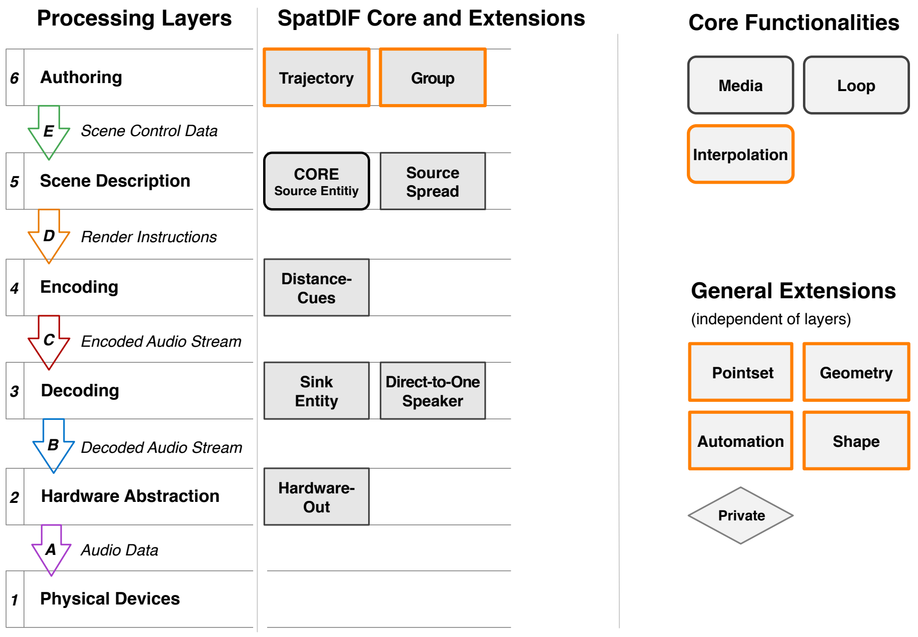

Figure 1. Left column: the processing layers and data streams in a spatialization workflow. Middle and right column: the categorization of SpatDIF extensions. New elements in SpatDIF version 0.4 are highlighted in orange and include the trajectory and group extensions on the authoring layer, a source-spread extension on the scene description layer, refined core functionalities, and four new general extensions.

cal for storing or transmitting a finished piece, where all artistic decisions have been taken and the piece just has to be rendered. It does not, however, keep any trace of how these artistic decisions and their manifestation in the scene, be it in position changes, loudness changes, etc., came to be. Also, because current discretization methods of trajectories and their perceptual artifacts is an ongoing research topic (e.g., [11]) and may improve in the future, it is desired to preserve the "master tapes" of sound trajectories along with their discretized version [12].

In other words, version 0.3 of SpatDIF is a rendering-centric syntax with a discretized representation of the audio scene.

1.2 SpatDIF Terminology

A SpatDIF representation 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** which consists of core descriptors, and descriptors organised in extensions that add functions. Finally, a SpatDIF representation consists 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.

2. NEW IN SPATDIF VERSION 0.4

To enable the authoring of spatial music with SpatDIF the new specification version 0.4 [13] extends the scope of SpatDIF to the sixth processing layer, the **authoring layer**. Layer six describes processes that drive changes, whereas layer five contains a discrete representation of the resulting state of a scene at specific times. On this sixth layer, compositional processes that build a sound scene take place. A vast number of operations or instructions can be imagined that manipulate sound entities in the scene. Of these many possible dimensions, the descriptors at this layer currently address motion.

In combination with the already defined appearance of source entities, their media assignment and an additional source-spread factor, a spatial sound composition using popular spatialization algorithms such as VBAP [14] or DBAP [15] can be fully represented.

2.1 The Trajectory Extension

The new authoring layer in SpatDIF is defined by the trajectory extension. Applying the trajectory extension to an entity in the time-section generates a specific spatial motion in time called trajectory.

A simple example for such a description would be the following instruction in natural language: "Move source N from point A to point B by following a straight line over 5 seconds with constant speed." In technical terms, a trajectory is a result of a combination of the following functionalities:

- A shape created by a **pointset** (see Section 2.2) or a predefined **shape** template (see Section 2.5).
- Affine **geometry** transformations to manipulate this shape (see Section 2.6).

- A spatial **interpolation** method describing how to get from point to point (see Section 2.3).
- An **automation** profile, describing how a trajectory proceeds in time (see Section 2.4).

These functionalities are addressed in the interpolation core functionality and in the layer-independent extensions pointset, geometry, automation, and shape.

Since the pointset extension can serve to predefine templates and is dependent on the interpolation functionality to define a shape it can be placed either in the meta or the time-section. The automation is applied to an entity in the time-section because it affects the temporal unfolding in the scene. Both the shape templates and the geometry transformations are used in the time-section to facilitate the repeated path-description of a trajectory from templates.

The following extensions may be used independently of the authoring layer for other purposes.

2.2 The Pointset Extension

A pointset describes a group of geometrical positions or points. These can be key-points on a path, shape, curve, or polygon of any kind, but also a collection of entities of the same kind, such as sink- or speaker-positions.

Points in a pointset can be of two kinds: actual points (default), i.e., *anchors*, and helper points, i.e., *handles*. The second kind is needed to describe cubic-bezier splines, but could also serve as reference-points in other functionalities (see Section 2.3).

2.3 Extended Interpolation Functionality

For the use in trajectories, the core functionality for interpolation were extended. An interpolation defines the method used for sampling between two defined values (points/positions).

The interpolation functionality now provides three methods:

The **none** method is used to stop a motion, for example when overwriting pre-existing layer five information.

The **linear** interpolation is the default method. It can be carried out in cartesian coordinates to obtain straight lines, while linear interpolation using polar coordinates leads to arc motions (this applies to entity positions only, for arc shaped trajectories, see Section 2.5).

SpatDIF version 0.4 features the newly defined **cubic-bezier** interpolation functionality [16] as its principal way to describe curved paths and automation profiles.

Contrary to other splines, a cubic-bezier curve is defined by four points; the first and last are the anchors points that bound the curve, the middle two points (P1, P2) are handles that are used to steer the tension or curvature (see Section 2.2). This makes the use of bezier curves unambiguous and has the advantage of generating a fallback polygon of anchor-points, thus producing a shape that still resembles the original intention. In the case of a multi-segment cubic-bezier curve, the last and first point of each contiguous segment are shared, thereby reducing the number of points of the pointset approximately by a quarter [17].

2.4 The Automation Extension

The automation timing function describes how the sampling cursor moves along the path over the duration of the trajectory. The control points for the functions are described in two-dimensional relative coordinates of time over value, abbreviated `tv`. They range from 0 to 1 and go from trajectory start to trajectory end in the time relative to the duration of the motion.

Typical automation movements begin with a slow acceleration and end in a deceleration to mimic the physical behavior of objects with mass; these time profiles are called *easing* curves (see the ease-in-out function in Fig. 2). The default easing function is linear with constant speed over the entire path. A number of standard easing functions are provided that mirror the timing functions of CSS transitions [18].

The addition of multi-segment polygons or cubic-bezier curves completes the selection of functions and caters to almost all imaginable curves (bottom of Fig. 2).

By using oscillatory, zig-zag or rectangular shapes in the automation timing function, motion patterns such as palindrome looping, jumps and other interesting behaviors along the trajectory may be produced.

With these three *essential* general extensions, a trajectory can now be fully articulated in the time-section. As with any other element in a typical compositional method, however, shapes and trajectories want to be re-used and modified again and again. To avoid having to repeat a shape definition every time it is applied as an entity's trajectory, the capability to pre-define and recall shapes, pointsets and other elements is crucial. For this purpose two additional extensions have been defined.

2.5 The Shape Extension

To facilitate describing the most common trajectories, SpatDIF provides a few basic shape-primitives in the shape extension. These default shapes are defined with standardized size and orientation (see Fig. 3).

The **point** primitive serves to stop or fixate a trajectory; the **line**, **triangle** and **rectangle** primitives provide standard shapes. The **circle** primitive consists of a closed multi-segment cubic-bezier spline with predefined tension points. Note how the rectangle and circle primitive share the same anchor points, this serves as a bridge between the two shapes. The **arc** primitive is a special case, since it is defined using only the starting point, angle and radius to the centre point and the arc angle. This function is mainly intended for spiraling motions.

The addition of an arbitrary pointset in the shape extension completes the elements needed to predefine paths or timing functions in the meta-section.

To reuse shapes within a scene, shape templates can be stored in the meta section and applied to a trajectory. The pointset defining the control polygon, as well as the interpolation method are thus predefined in a standard size, to be resized when applied as a trajectory to a specific entity in the scene. When assigning the shape-template to the entity, the first point of the pointset is attached to the current position of the entity. The use of the unique name in the

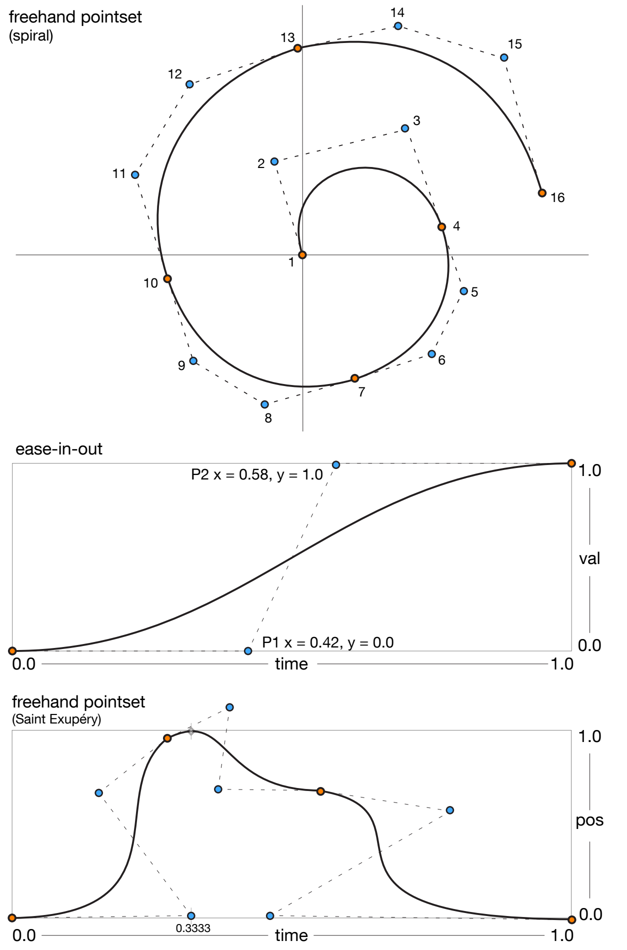

Figure 2. A trajectory consisting of a 2D spiral path (top) and two possible automation curves. The path consists of a pointset with a cubic-bezier interpolation; anchor points are marked in red, handle points in blue. The first automation curve is a predefined ease-in-out function smoothly moving from start to end of the path. The second automation curve is a free-hand multi-segment pointset with cubic-bezier interpolation. It moves from the start to the spiral's end in a third of the duration with a steep acceleration, then returns gradually to the beginning of the path in a retrograde motion.

id descriptor allows to reference a standard or pre-defined shape in the time-section [17].

When applying these shapes to an entity in a trajectory, the standard geometric properties may be transformed using the geometry extension (see 2.6).

The same pre-definition and referencing mechanism can be used for a pointset defining an automation curve.

2.6 The Geometry Extension

When applying a pre-defined shape to a trajectory, the geometric properties, such as size and orientation may need to be modified. For this purpose *affine geometrical transformations* [19] can be applied. The set of transformations includes **scale**, **translation**, **rotation**, **skew** and **mirror** and can be applied in any combination. Because the results of such transformation sequence may be order-dependent, the order of these operations is defined explicitly [17]. A

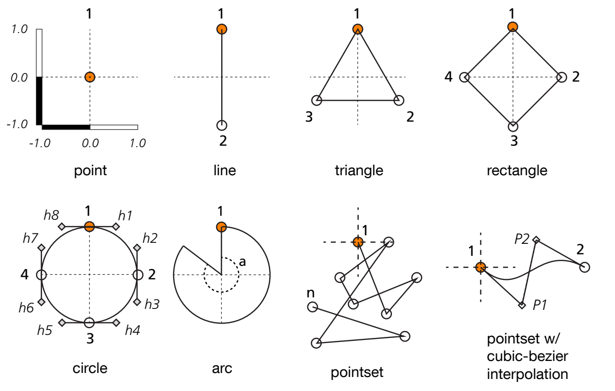

Figure 3. Predefined shape primitives, an arbitrary polygon, and a cubic-bezier curve based on the pointset.

transformed shape's first point overwrites the current position of an entity in layer five description; the entity position needs to be updated when storing the file or scene.

2.7 The Group Extension

To affect several entities at the same time the group extension is introduced on the authoring layer. This functionality can be used to apply the same behavior to a collection of entities.

This is used for compositional processes where several voices are conceptually treated as a single unit. An example might be a scene in which a vehicle is modeled that consists of the sounds of the four wheels and the engine placed at the appropriate positions in relation to each other. In order to displace the vehicle in the scene only the group's 'handle' is displaced (for example located in the driver's seat), and all other sound entities (e.g. the wheels and the engine) move by maintaining the relative position to the group (driver) (see top left of Fig. 4).

A SpatDIF group is identified and linked to by its unique name. The group represents an abstract entity and possesses the same properties and functionalities as a basic entity. For instance, it has a reference-point with a position and orientation, and this point serves as a 'handle' point for geometrical operations on the group (see top of Fig. 4). Groups can be statically defined in the meta section and/or dynamically created in the time-section. At the time of their creation, groups are initially empty. In order to populate a group, entities need to become members of a group. They attach by setting the group's unique name in their `group-membership` descriptor. An entity can only belong to a single group at a time. As long as an entity is attached to a group, the group's behavior overrides the member's behavior with respect to all descriptors that are explicitly described by the group. As a consequence attempts to change the same properties of single group members will be ignored.

When an entity joins a group, the relationship to the group is established by calculating the entity's relative (delta) value to the group's descriptor value. If a change in relationship is desired, the entity first needs to be detached from the group by setting the `group-membership` to `none`, so that it can be addressed individually, and then reattached to the group with a changed relationship, for

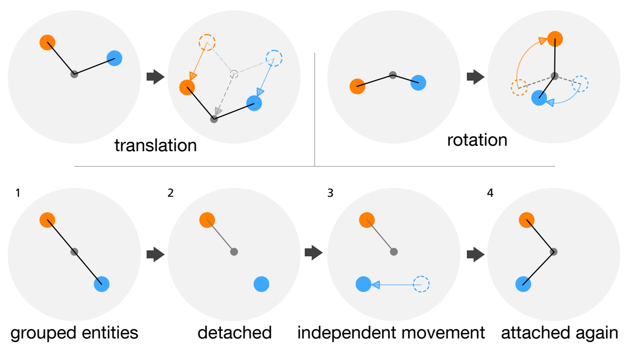

Figure 4. The group extension: A group is moved (top left) or rotated (top right); a group's member entity detaches to execute an independent movement, then attaches again (bottom).

example a shifted position or orientation, gain or spread factor (see bottom of Fig. 4). At the time of detaching from a group, the entity keeps the most recent value, including any changes that have happened as consequences of changes to the group's descriptor values. In other words, detaching from a group introduces no discontinuities in descriptor values at the fifth layer, unless a new value is set explicitly.

For descriptors affecting sound, i.e., directly active in domain of acoustics, the descriptor's unit is used to determine the manner in which to combine the group's descriptor value with the member-entities' descriptor value. When the value of a group's descriptor is changed, the change in the member-entities' descriptor-value happens according to a simple rule: If the change is expressed using a *logarithmic* unit, the group's value will be **added** to the current value of each member; if the change is expressed using a *linear* unit, the group's value will be **multiplied** with the current value of each member. For example, if the gain-level of a group is increased by 6 dB, this will increase all member-entity gain levels by 6 dB. In contrast, if the gain change is 2.0 linear units, this will double the gain levels of all member-entities.

For descriptors affecting the scene geometry, the group descriptor's value is **added** to the descriptor value of each member (see top of Fig. 4).

Core functionalities and general extensions such as trajectories, affine geometrical transforms, and automations can be used to change the group's properties instantaneously or over time. These operations can be applied to the same descriptors and in the same manner as for the entities contained within the group. As seen already, a group could be used to control the gain of several entities, rather than their position or orientation. Or the geometrical transformation could be used to rotate the entire group around the handle-point, or to shrink or expand the group by changing the scaling factors, or a combination thereof [17].

Currently, in SpatDIF v.0.4, the group extension does not support nested *hierarchies*, i.e., a group cannot contain another group [20]. Although planned for a future version, a number of open questions pertaining to the definition of this functionality still need further research and clarification.

As with any of the operations in extensions of the sixth layer, when modifying the scene, the group behaviors need to be propagated to the individual member-entities in the fifth layer description (see Section 3.1).

2.8 The Source Spread Extension

SpatDIF version 0.4 introduces one additional extension that addresses in a simple and general way the perceived spatial extent of a source.

Many rendering techniques offers methods for making the spatial localization of sources less distinct, leading to the perceptual illusion that the extent of the source spreads out. In the widely used VBAP-algorithm, for example, a width factor determines the spreading or smearing of the source across part of the sound sphere [21]. The planar DBAP algorithm has a similar blur parameter [15]. In other more advanced spatialization algorithms, source widening and diffuseness can be generated using small source motions around the position [22] or by lowering the spatial resolution in Ambisonics and other techniques to reduce directness and generate diffuseness in the sound source [23, 24].

The Source Spread Extension offers a simple and shared minimal description of the amount of spread, expressed as a percentage. The different spatial rendering processes will need to interpret this accordingly, each in relation to its spatialization principles and abilities. In general a spread of 0% should be rendered with a spatial localization that is as precise as the process is able to produce, while a spread of 100% should result in the sound being rendered in a manner that is as spread-out and non-localized as the algorithm possible can achieve.

3. DISCUSSION

Currently the descriptors on sixth layer work together with general extensions to describe source trajectories in the sound scene as well as the grouping of entities. A few fundamental rules of how to deal with this new type of representation need to be discussed.

3.1 Complementary Representations

With the introduction of the sixth layer for authoring instructions in SpatDIF version 0.4, events within the unfolding scene can be represented in two parallel ways; as layer six trajectories and as a layer five discretized, time-sampled representation. The two representations are complementary and serve slightly different purposes. The difference between them is analogues to the difference between vector-based graphics and bitmap images. Trajectories express processes, relationships and tendencies over time, and software tools for spatial composition such as Zirkonium [6] may provide intuitive graphical user interfaces for visualization and interaction with the trajectories. The ability to store trajectory information makes the resulting spatial composition more robust to future transformations such as geometric or time-related modifications, as the time-sampled representation can be recalculated to ensure adequate temporal resolution.

When making use of the trajectory extension, it may seem tempting to simply replace the layer five representation by the much more economical trajectory representation. SpatDIF requires that the layer five representation is always present in the resulting description because of the following reasons:

Direct use of the trajectory representation for rendering may impose the heavy burden of continuously interpreting the trajectories in the scene for *every* SpatDIF-compliant rendering process. The inclusion of the layer five time-sampled representation caters to relatively simple playback and spatialization processes, eliminating the need to continuously recalculate all sound properties from high-level instructions and this minimizes computation load. This also ensures that scenes authored using SpatDIF version 0.4 that make use of the trajectory extension remain backward compatible. Additionally it supports the goal of interoperability and reproduction in future software tools of unknown capability.

This does not prevent a capable software of rendering directly from a sixth layer representation. However, it imposes the presence of fifth layer information in the exported files or transmitted streams.

If a file contains sixth layer information, as indicated by the mandatory extension declaration in the meta section, a *rendering* process disregards the authoring information, whereas an *authoring* or editing process that modifies the scene's animation processes *supersedes* the simpler scene rendering information. In order to maintain the two representations in synchrony, when storing the scene to file, the modifications of the 'blueprint' of the scene in the authoring layer, i.e., of shapes that describe the evolution of scene, are always propagated down to the simpler representation, thus potentially altering and updating existing rendering instructions in layer five descriptors [17]. A further consequence of this is that if for some reason a conflicting discrepancy has emerged between the layer five and six representations of a spatial event, the layer six representation takes precedence, provided that the software reading the file is able to deal with trajectories.

3.2 The Quest for Efficiency

In his seminal analysis of sonic art Wishart provides an extended chapter dedicated to spatial motion [25, pp. 191–235]. Taking these reflections and his many concrete geometrical shape examples as a reference point, the challenge for the definition of the authoring layer descriptors lies in the bound-less variety of systems, functions and models that are capable of generating motion. Much as Wishart aims for a qualitative understanding of sound-motion in space, the SpatDIF authoring layer aims at *describing* rather than *formalizing* the resulting shape of motion generated by an algorithm or by free-hand drawing by a composer. It does *not* transport a possible formalized, mathematical representation of a source motion, such as for example the Lissajous formulas that generate the repeated figures in the canonical piece 'Turenas' by John Chowning [26]. The final resulting trajectories, however, are what the sixth layer aims at representing in their most detailed form.

This choice is done in the spirit of achieving the most with the least elements, and thus enables the methods for describing curved shapes (see Fig. 2). However, as shown in other standards such as Postscript (for example used in eps/pdf vector graphics) [27] and CSS used for rendering graphics on webpages [18], these few elements prove to be sufficiently flexible and powerful to cover all but the most exotic cases; even circular shapes can be approximated to a very high degree using multi-segment cubic-bezier curves [28].

Composers may think that SpatDIF's description of trajectories are counter-intuitive, but authoring is expected to be done with software tools that provide graphical user interfaces; hence there should be little or no need to interact directly with cubic-bezier parameter values.

In SpatDIF version 0.4 a trajectory is expressed as the combination of a spatial path and a time-based automation function. This may seem counter-intuitive compared to simply describing position in space as a function of time. In authoring tools it is however easier to access and author trajectories and movements graphically when organized in this way. In addition, this separation reflects spatial movement as it occurs in everyday life: The spatial shape, with its additional geometric transforms and spatial interpolations, describes the pathway to be followed, whereas the time-based automation function expresses how an entity or group moves along this pathway.

4. CONCLUSIONS

The recently updated version 0.4 of the SpatDIF specification [29] addresses the ability to define and store continuous trajectories on the authoring layer in a human-readable way. As a result, SpatDIF provides a new way to exchange higher level authoring data across authoring tools that helps to preserve the artistic intent in spatial music. Trajectories are described using cubic-beziers curves. With a minimum amount of functions this enables a high degree of flexibility in terms of what curves can be realized.

The new group extension enables multiple entities to be addressed collectively. In combination with trajectories this enables coordinated movements of multiple sources. Groups may also be used for mixing, where the gain level of a group can be adjusted relative to other parts of the scene while maintaining a consistent mix within the group.

SpatDIF version 0.4 compliant files that contain trajectories and groups also need to include the layer five, discretized scene description information. This ensures backwards compatibility with version 0.3 SpatDIF-compliant rendering software.

Support for SpatDIF version 0.4 authoring has already been implemented in the Zirkonium spatial audio authoring tool. Work on archiving and restoring older spatial compositions from the ZKM archive is ongoing, and the restored compositions are being saved to SpatDIF version 0.4 files [6].

Finally, in addition to improvements on the authoring layer, SpatDIF version 0.4 adds a simple description for source spread.

5. REFERENCES

[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] 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, pp. 11–22, 2013.

[3] C. Miyama, G. Dipper, R. Krämer, and J. C. Schacher, "Zirkonium, SpatDIF, and mediaartbase.de: an archiving strategy for spatial music at ZKM," in *Proceedings of the Sound and Music Computing Conference*, Hamburg, Germany, 31. August – 3. September 2016.

[4] J. C. Schacher, C. Miyama, and T. Lossius, "The SpatDIF library – Concepts and Practical Applications in Audio Software," in *Proceedings of the joint International Computer Music and Sound and Music Computing Conference (ICMC|SMC|2014)*, Athens, Greece, 2014.

[5] A. Pérez-López, "Real-Time 3D Audio Spatialization Tools for Interactive Performance," Master's thesis, Universitat Pompeu Fabra, Barcelona, 2014.

[6] C. Miyama, G. Dipper, and L. Brümmer, "Zirkonium Mk III - A Toolkit for Spatial Composition," *Journal of the Japanese Society for Sonic Arts*, vol. 7, no. 3, pp. 54–59, 2015.

[7] R. Diaz and T. Koch, "Live Panorama and 3-D Audio Streaming to Mobile VR," in *AES Conference on Headphone Technology*, Aalborg, Denmark, 2016.

[8] ITU, *ITU-R BS.2076: Audio Definition Model*. Geneva, Switzerland: International Telecommunication Union, 2015.

[9] N. Peters, T. Lossius, J. C. 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.

[10] N. Peters, J. C. Schacher, and T. Lossius, "SpatDIF specification version 0.3, draft version," Specification of the Spatial Sound Description Interchange Format (SpatDIF) v. 0.3. 2010–2012, http://redmine.spatdif.org/projects/spatdif/files.

[11] N. Hahn, K. Choi, H. Chung, and K.-M. Sung, "Trajectory sampling for computationally efficient reproduction of moving sound sources," in *Audio Engineering Society Convention 128*, May 2010.

[12] G. Boutard and C. Guastavino, "Archiving electroacoustic and mixed music: significant knowledge involved in the creative process of works with spatialisation," *Journal of Documentation*, vol. 68, no. 6, pp. 749–771, 2012.

[13] N. Peters, J. Schacher, T. Lossius, and C. Miyama, "SpatDIF specification version 0.4, draft version," Specification of the Spatial Sound Description Interchange Format (SpatDIF) v. 0.4. 2010–2016, http://redmine.spatdif.org/projects/spatdif/files.

[14] V. Pulkki, "Virtual sound source positioning using Vector Base Amplitude Panning," *J. Audio Eng. Soc.*, vol. 45, no. 6, pp. 456–466, 1997.

[15] T. Lossius, P. Baltazar, and T. de la Hogue, "DBAP - Distance-Based Amplitude Panning," in *Proc. of 2009 International Computer Music Conference*, Montreal, Canada, 2009, pp. 489–492.

[16] M. Sarfraz, M. Asim, and A. Masood, "Capturing outlines using cubic bezier curves," in *Proceeding of the International Conference on Information and Communication Technologies*. IEEE, 2004, pp. 539–540.

[17] N. Peters, J. C. Schacher, T. Lossius, and C. Miyama, "SpatDIF Example Files," http://www.spatdif.org/examples.html.

[18] WC3 Editors, "CSS Transitions." [Online]. Available: https://drafts.csswg.org/css-transitions-1/

[19] K. Nomizu and T. Sasaki, *Affine differential geometry: geometry of affine immersions*. Cambridge University Press, 1994.

[20] J. C. Schacher, "Gesture Control of Sounds in 3D Space," in *Proceedings of the Conference on New Interfaces for Musical Expression*, New York, USA, 2007.

[21] V. Pulkki, "Uniform Spreading of Amplitude Panned Virtual Sources," in *Proc. 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, Oct. 17-20 1999.

[22] F. Zotter, M. Frank, and M. Kronlachner, "Efficient phantom source widening and diffuseness in ambisonics," in *Proc. of the EAA Joint Symposium on Auralization and Ambisonics*, Berlin, Germany, 3-5 April 2014.

[23] T. Lossius and J. Anderson, "ATK Reaper: The Ambisonic Toolkit as JSFX plugins." in *International Computer Music Conference— Sound and Music Computing*, 2014, pp. 1338–1345.

[24] A. Sèdes, P. Guillot, and E. Paris, "The HOA Library, Review and Prospects," in *International Computer Music Conference— Sound and Music Computing*, 2014, pp. 855–860.

[25] T. Wishart and S. Emmerson, *On sonic art*. Amsterdam: Harwood Academic Publishers, 1996.

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

[27] G. Farin, *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[28] J. J. Chou, "Higher order bézier circles," *Computer-Aided Design*, vol. 27, no. 4, pp. 303–309, 1995.

[29] N. Peters, J. C. Schacher, T. Lossius, and C. Miyama, "Specification of the Spatial Sound Description Interchange Format (SpatDIF) V. 0.4," http://spatdif.org/specifications.html, 2010-2016.