

Minutes SpatDIF level meeting

Location: CIRMMT conference room, Montreal

Date: Nov. 28th 2008

Duration 12.00 - ca. 17.00

Attendees:

- Mike Wozniowski, S.A.T.
- Jasch, ICST
- Marlon Schumacher, McGill
- Nils Peters, McGill

(Nils in black)

(Jasch additions in Purple)

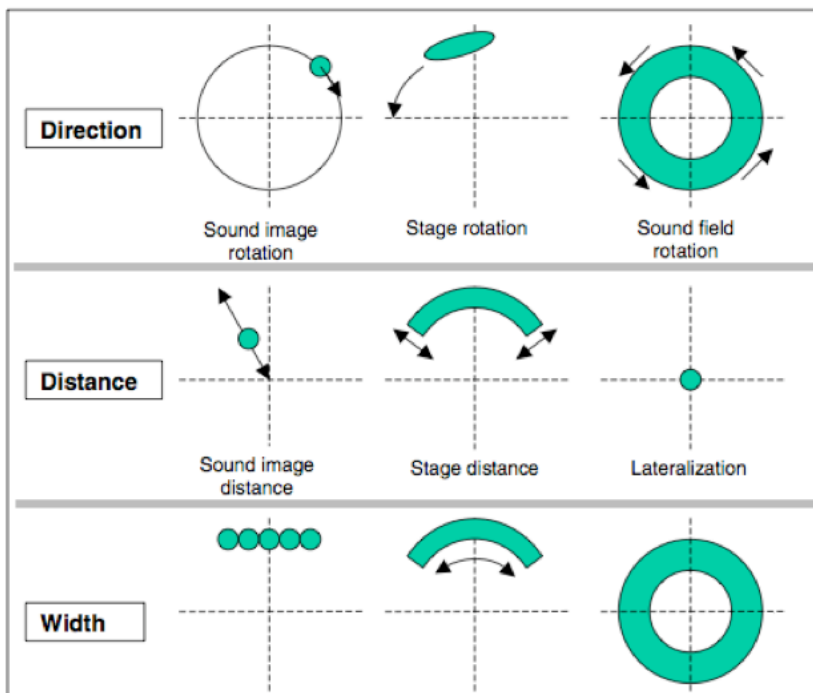
(pascal addition in green)

Main Points of Discussion

Who are the Users of the SpatDIF format?

We discussed the spectrum of users that would benefit from using a SpatDIF format.

- - Composers
- - Sound engineers
- - Scientists



<-- common effects in spatial sound mixing practise

Figure taken from

K. Hamasaki, H. Wataru, et. al.
"5.1 and 22.2 Multichannel

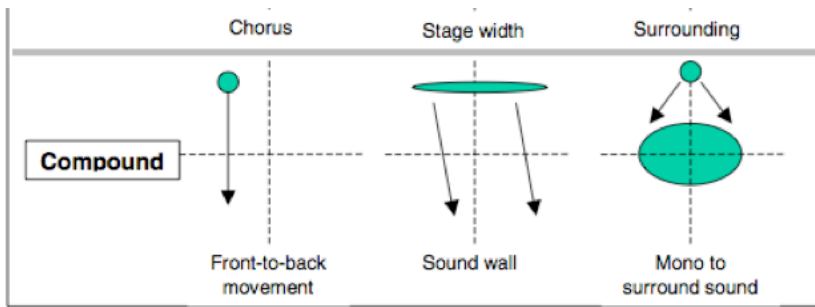


Figure 2: Surround sound effects required in 5.1 channel program productions

Sound Productions Using an Integrated Surround Sound Panning System", 117th AES Convention, San Francisco, 2004

What is the role of the SpatDIF format? (What paradigm does it express?)

- Transport a scene description.
- Represent abstract sound-objects in an abstract open euclidian space.
- Allow the decoupling of the spatialisation creation process from the actual rendering process.

What are the Core Descriptors of SpatDIF?

AXIOM: *The SpatDIF Format only allows for ONE type of unit or definition for each descriptor.*

The SpatDIF definitions should try to comprise only a small set of Core descriptors. These are mandatory / it's strongly suggested they be implemented in a SpatDIF compliant renderer. The Core descriptors reflect closely the main intent of Spatdif to serve as a spatilisation description transport format.

(euclidian/object attributes vs. perceptual psycho-acoustic elements ?) + the need to think about "acoustic space DIF" "electro-acoustic space DIF" (ASDIF :=)

- should be a set of fundamental descriptors which can be understood by most sound renderers
- A call for proposed core descriptors (and once that's collected a poll/ vote for descriptors) should help to defined this set and make it applicable.

SpatDIF and Time:

Time needs to be described in an explicit way in SpatDIF in order to be able to render a File/stored SpatDIF stream.

- Timestamps
- absolute time vs. relative time
- implicit time grid/explicit frame positions / dense vs. sparse description of evolving parameters

- the notion of frames. whenever a node in the scene changes at a new time point a frame message is generated to mark the new frame. everything happening at that time point from there on is considered part of the same frame.
- reference frames / Master frames. In order to be able to jump around in / have nonlinear rendering of a stream, mark certain frames as master frames and re-describe entire scene in this frame (hybrid between an implicitly state-less and fully state maintaining system.) (extension...)

SpatDIF Entities:

Objects in the Scene are abstract entities. Again we would need to define a set of **Core Entities**, and keep all other entities in the extensions.

Examples of basic entities (objects) would be:

- Source (aka voice, ...)
- Listener (aka target, sink, receiver ...)
- Speaker
- Diffuser (not sure what that is... Jasch? [NP])
- Obstacle
- ...

SpatDIF Descriptors and different Transport Protocols:

- The idea that SpatDIF should not be constrained into one unique transfer protocol.
- OSC, JSON, XML are all able to *transport* SpatDIF content
- XML
 - useful as a storage format
 - XML-RPC as streaming format? <http://www.xmlrpc.com/>
 - the XML hierarchy would represent the entire SpatDIF name-space. In analogy to the OSC name-space its root would be <SpatDIF version="1.0">...</SpatDIF>
- OSC
 - In order to limit confusion with existing OSC-streams, it was proposed to start OSC messages with /SpatDIF/.... (or /spatdif/ ? are we case sensitive...)
 - Jamoma nOSC proposal (e.g. to use dots for instances) is highly relevant, but as long as this is only supported in Jamoma, SpatDIF should not rely on it, because it will limit the exchangability with other platforms.

How intelligent are writer/encoder and reader/decoder?

It is the task of the SpatDIF writer (encoder) to convert from whatever other representation of a descriptor to the SpatDIF compliant one. The creation environment / controller might have extended functionalities (such as working with higher level hierarchies/ groups and allowing affine transforms of groups e.g. in local coordinate systems, different representation of a coordinate system) but its SpatDIF writer (encoder) outputs only standardized SpatDIF descriptors.

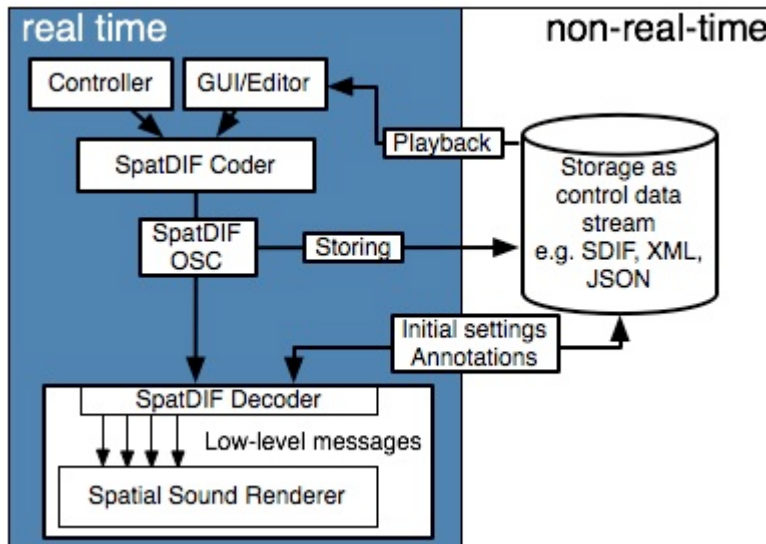
This protects all SpatDIF parsers (clients/interpreters) from having to deal with different definitions of descriptors.

(examples: coordinate systems, gain units, diffusion patterns).

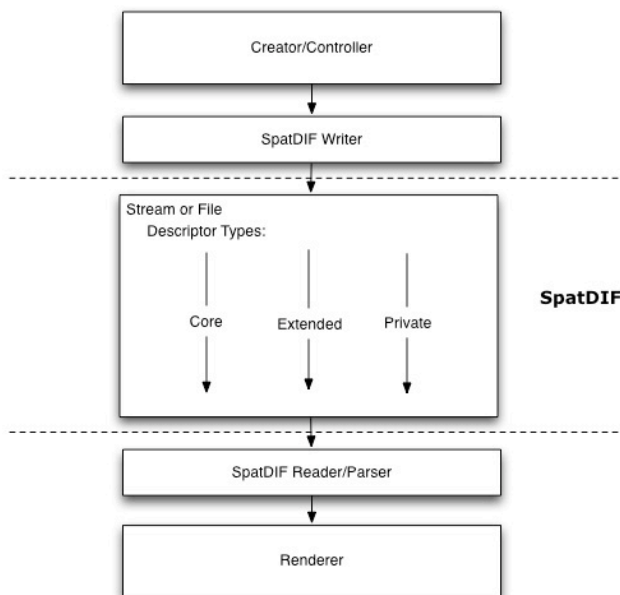
The view of the scene on the renderer side is non-hierarchical/flat, it's agnostic of the guiding principles that led to the generation of the information before the writer/encoder process.

Special information for specific rendering algorithms sent from the controller can be included in a SpatDIF stream/file but are not covered by core descriptors and should be considered private. A writer can be geared for a specific renderer using these private extensions. Other renderers ignore these messages but should maintain them in case of rewriting the SpatDIF file.

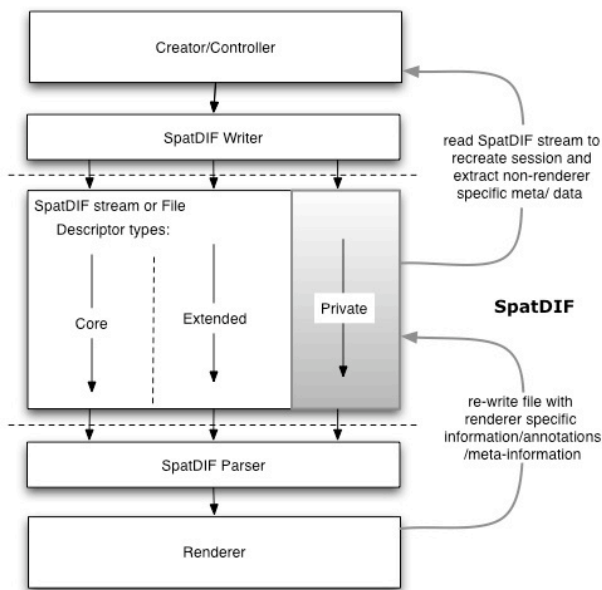
The interpreter/parser knows a minimum about SpatDIF but is an Expert in generating the low-level controls for it's rendering engine.



SpatDIF Basic Data Flow



SpatDIF Basic Data Flow with extended private Functions



Proposal for the use of Extensions:

- Extensions are not part of the core functionalities, so that renderers are not mandated to understand this information.
- They can be seen as a proposal. The most commonly used ones might become included into the core in future SpatDIF versions (via a consensus process of the community)

Media Extension

- the reason for this extension is to define not only where sources are spatialised, but also to assign content (media files, live inputs, Internet streams) to a virtual sound source position
- /source/1/type adc
/source/1/inputChannel 1
/source/1/type B-format
/source/1/type media
/source/1/path audiofile.wav
- Question: What to do with stereo and multichannel files ?
- research what other formats are doing in this regard (EDL edit decision lists in film/video or [broadcast wave format](#))

Event Extension

We need to think about how to deal with discrete vs. continuous parameter changes (event vs. state) what are the definitions?

- /source/*/start /* this is a discrete event */
/source/1/active 1 (inverse similar to mute) /* this looks more like a changing a state to me */

Ambisonics Extensions

a set of descriptors specific to ambisonic decoding,

- order,
- order-weights,
- bformat channel mapping,
- encoding formula used (Furse-Malham, SN3D, N3D)
- matrix manipulation of the ambisonics channels

Spatial Granular Synthesis Extension

this is a special case which we did not discuss in detail

Acoustic Spaces Extension

describing properties of acoustic rooms, as needed by renderers who model spaces.

- room size
- material/damping
- environmental variables (humidity)

General Question and unsolved Issues:

- Should speaker positions be included into this basic feature set?
- Definition of source directivity ?
 - compare definition of openAL, directX (cones) vs. definition of X3D (ellipsoids) vs. definitions in Spatialisateur (aperture)
- We should agree on one coordinate system (also for storage)
- What role plays time here ?
- What role plays a gain here ?
- How can conceptual ideas (i.e. groupings of sources) be preserved in the format without adding overhead to the decoder side ? (i.e. of solving complex tree structures -> adding group properties to source-objects would allow this)

Next Steps:

- Collect proposals for core descriptors
- Agree on unique standard units/definitions for the basic descriptors
- provide benchmark scenarios: (simple) compositions consist of a multichannel sound file (e.g. 4 tracks) together with a description how to spatialize these tracks. These benchmark tests could then be used to see how different renderers handle behave.

