Gesture Control of Sounds in 3D Space

Jan C. Schacher
Institute for Computer Music and Sound Technology,
Zurich School of Music, Drama and Dance
Baslerstrasse 30
8048 Zürich, Switzerland
++41 43 305 45 00
jan.schacher@hmt.edu

ABSTRACT

This paper presents a methodology and a set of tools for gesture control of sources in 3D surround sound. The techniques for rendering acoustic events on multi-speaker or headphone-based surround systems have evolved considerably, making it possible to use them in real-time performances on light equipment. Controlling the placement of sound sources is usually done in idiosyncratic ways and has not yet been fully explored and formalized. This issue is addressed here with the proposition of a methodical approach. The mapping of gestures to source motion is implemented by giving the sources physical object properties and manipulating these characteristics with standard geometrical transforms through hierarchical or emergent relationships.

Keywords

Gesture, Surround Sound, Mapping, Trajectory, Transform Matrix, Tree Hierarchy, Emergent Structures.

1. INTRODUCTION

When moving away from the score-based composition and timeline approaches to spatialisation we quickly realize that controlling a number of sources in a virtual acoustic space is a complex task, considering the amount of information needed to generate interesting motion patterns and source distributions. The necessity for a scoring language set aside, there is a need for a methodology describing the mapping of gesture actions onto source motion that includes the possibility to apply emergent behaviors to sound sources in a periphonic surround space.

The tools and methods presented here are based on the experience of many live-performances both in the audio and in the visual domains. The software-components for MaxMSP [18] and Pure Data [7] have been published since 2002 [12] and are widely used. [11]

What has been lacking is the definition of a comprehensive approach to linking all of the available elements in a way that is highly modular and not specific to one type of application. This approach shows a collection of elements and the corresponding framework for interfacing *gesture controllers* with *sound sources* in 3D surround.

The need for this systematical approach arises from the special situation of real-time musical performance using large numbers of sources in an open or improvised manner. In that context it is impossible to work from pre-edited motion sequences, but it's still imperative to build a control structure that permits access on different levels to the source's spatial properties. Often the mapping between the controllers and their point of application is structured in a dynamic layer of its own.

2. BACKGROUND

In the last few years attempts at unified mapping and input strategies have been made, most notably the MnM library from IRCAM [1], the hid project by H.C. Steiner at ITP/NYU [16] and the GDIF iniative by Alexander R. Jensenius in Oslo [5]. These projects all deal with the polymorphism found in today's physical controllers and motion tracking systems when used as musical interfaces. But they don't address the specific task of controlling and interacting with sounds in a hierarchically structured 3D surround space.

In his piece "Turenas" from 1972 John Chowning [3] sends his acoustical events onto carefully planned lissajou trajectories around the listener and space. More recently granular clouds have been injected with swarming behaviors to create emergent structures. [2] Thus the sounds have obtained object-status that gives them properties other than the purely acoustical ones that traditionally describe them.

In this project the sound-sources are considered as objects that can have a shape or "gestalt" and exhibit behavior by themselves or in relation to other objects, including the performer. In order to be able to spatially model such object characteristics they need to have the same properties we know from real physical objects, such as size, weight and position. The objects can also contain and encapsulate other objects in a parent/child relationship, permitting the building of hierarchical structures where the objects inherit spatial attributes from their parent. All of these concepts are clearly borrowed from 3D modeling and graphics but are less explored in the context of real-time interaction with sounds.

The author's user experience with this method have shown that careful planning of the object hierarchies can yield fairly simple control structures that already give many possibilities in their combinations. Typically this will include a number of sources, going through two levels of transformation, which are controlled, from simple MIDI-faders, joysticks and sensor-interfaces such as a glove.

3. INTERACTION MODES

Keeping in mind the notion of the physical presence of a performer in a space populated with sounding objects that can be "touched" and transformed, two modes of interaction can be distinguished. The top-down mode can be described as exerting direct control through a specific mapping onto certain parametric properties of an object. The bottom-up mode asserts that sounding objects exist in space with their own physical properties and

behaviors and all interaction happens through gesture actions in the virtual space. Both modes need specific mappings and different methods to interact with the objects. Higher level interaction models using for instance pattern recognition algorithms, such as neural networks or mappings based on psychological criteria through an 'affect space' can still be assigned to one of the two modes. This depends more on how the resulting data is applied than on how it was transformed before. What counts from the perceptual point of view is the self-awareness in the body image of a performer using gestures and the use of the perception of a physical presence of the sounds in the same space. The two interaction models lead to quite different results in that regard.

3.1 Top-down Control

When controlling an object directly through a gesture, the output of the physical interface is first abstracted, then translated to the appropriate scaling and dimension (one-to-many, many-to-one, one-to-one) and finally applied to the exact node(s) corresponding to the desired object's property in the hierarchical structure. The challenge here is to find an intuitive and direct way to route the dataflow to the appropriate transformation node. (see Figure 1.) Most real-time performances will use this interaction model, simply because it offers more direct control and finer mapping and scaling to the destination parameters. But the overwhelming number of routings and endless possibilities of changes in the mappings lead to mostly static setups, which develop only one form of gesture expression.

3.2 Bottom-Up Action

In a bottom-up interaction the gesture is abstracted from the interface, then translated to the appropriate scaling and dimension used for the control layer governing the sound-object's behaviors. This layer applies the transforms according to its implicit rules, be they physical, agent-based and emergent or arbitrary. The motion vector of a hand moving through space for example can be used as an impulse to propel an object along a prolongation of that vector, or the position of a tracked extremity can be used as attraction point to a group of objects controlled by a swarm algorithm. [2][9][14] Without going further into the details of these behaviors it is important to keep in mind their flat hierarchies or the complete absence thereof. Most of the time the agents (sound-sources) don't obey any other law than the one governing their behavior.

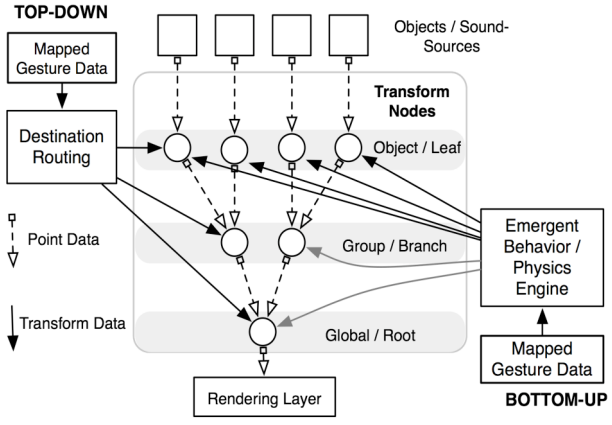

Figure 1. Diagram of *Top-Down* and *Bottom-Up* Interaction

Global and grouped controls can't be applied on a per-agent basis but only through the behavior algorithm's own variables.

4. STRUCTURE

The following elements comprise the complete control structure for gesture control of 3D sound. The modules are self-contained, presenting a unified interface and namespace. New modules, especially for emergent behavior, can easily be integrated into this design without an impact on the overall structure,

- the *Gesture Input Layer* inputs the information coming from the devices through their drivers and the respective software interfaces in the host software (serial, wireless, MIDI, hid). They are preconditioned, normalized and named according to a namespace using the OSC-syntax.

- the *Mapping Layer* translates from the dimensionality of the sensors/interfaces to a more general representation of data. Analysis on the raw data can be performed here to extract significant components and meta data describing psychologically more expressive aspects of the gesture. This data gets named independently and can be applied to any subsequent layer.

- the *Object Layer* contains the sound objects with their local behaviors

- the *Transform Layer* applies local, group or global geometrical transforms such as scaling, translation, rotation and skewing to a stream of point information coming from the object-layer. The parameters for the transformations are either routed directly from a gesture interface or remapped indirectly to data generated by a behavior algorithm.

- the *Rendering Layer* applies the resulting source positions to the chosen audio channels in the DSP processes.

- the *Storage Layer* handles either point-data, transform data or gesture data and provides facilities for playing them back. It also stores settings concerning the Meta-Layers themselves, for example the different routing settings used in a top-down interaction mode. The states of the Meta-Layers are stored in human readable XML-files, the more data-intense point and transform-data in a leaner binary format.

The sound-objects themselves are not just points describing the placement of a source in space but they can contain other local behaviors such as geometric patterns, random motion or predefined trajectories. Treating these behaviors as the object's *gestalt*-elements and manipulating their entire geometry in real-time permits complex motion patterns that depend on relatively little gesture information.

The use of hierarchical structures permits the control of all objects on several levels and creates the possibility of grouping them with varying scope. On the level of the leaf-node, geometrical manipulation only affects one object, the scope of the transform is local. On the level of a branch-node manipulations affect the properties of all of the branch's children, the scope of the transform is that of the group. The root-node transform affects all points in space and the scope for this can be considered to be global.

An important technique at an object's local or behavioral level is the use of a dual coordinate system using both the polar and Cartesian representations of position in space. When implementing orbital movements or collision detection, it's useful to change from Cartesian to polar coordinate representation where angular position and distance are decoupled. The use of relative

transforms originating from the current position of an object instead of from the global coordinate system makes the interaction of the top-down variants less dependent on the exact position of the performer's gesture. The gesture actions and their resulting transform data automatically get translated to the current point, permitting to switch the control quickly between objects that are not directly neighbors.

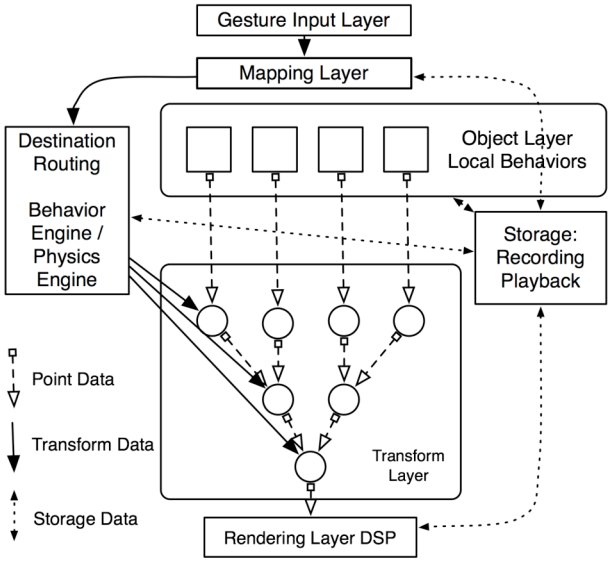

Figure 2. Control Flow and Modules

5. GESTURES

Depending on the kind of physical interface used, gestures can affect one or several parameters of a transform-matrix. When mapped onto a physics simulation, gesture actions such as grabbing, throwing, pushing or spinning objects become possible. Ideally the interface presents some kind of haptic feedback giving the performer a non-visual cue to the executed action. The phase project at IRCAM is a nice example of integrating a haptic feedback arm with a simulated terrain representing sounding objects. [10] Typical gestures range from one-dimensional fader-motion to multidimensional spatial gestures of the body measured by camera tracking, sensors worn on the body or embedded into physical devices with their own object properties. The author favors the use of small joysticks and graphics tablets with multiple pens to enter circular and planar gestures appropriate for horizontal source placements.

5.1 Controllers

The use of combined interfaces with extended dimensionalities makes a bottom-up interaction more intuitive and direct. Figure 3. shows an example of combined sensors on a glove. This gesture interface generates 9 dimensions of position data, and 5 streams of hand-shape data, making it ideal for the gesture actions mentioned above. The second example shows a mechanical arm that encodes the position of the tip with potentiometers in the joints. This type of interface has the great advantage of measuring a physical location in an actual space and giving a coherent sensorial feedback about motion in space. Other interfaces interesting for this kind of gesture mapping are surface controllers like the Lemur, a two dimensional, multi touch interface with visual feedback and the Wacom graphics tablets with two high resolution spatial dimensions and two auxiliary dimensions of pressure and tilt. For motion and gesture recognition video tracking has become one of the most commonly used technique even though it is fraught with its own problems and pitfalls.

5.2 Behaviors

Depending on the mode of interaction the sources may exhibit behavior that is completely independent of gestures or using gesture input only as impulses for physical behavior. Typical behaviors are random walks within boundaries, trajectory motion, flying within bounds, bouncing off walls with or without friction, gravity etc. The PMPD library by Cyrille Henry provides a few of these functionalities through physical modeling. [4] A future extension of this system by externals implementing the Open Dynamics Engine ODE [15] would increase the palette of behaviors considerably.

Figure 3. A gesture interface with flex- and pressure sensors, an accelerometer and an electro-magnetic tracker. The insert shows a simple 3D arm interface.

6. IMPLEMENTATION

The Software Components presented here were all implemented in C as externals for MaxMSP and are currently being ported to PD. Through the use of existing MSP libraries for multi channel audio spatialisation like the ICST Ambisonics Tools [13], VBAP [8] or VIMIC (for PD) [6] it has become easy to render multiple sound sources to multi channel surround sound on a laptop computer. Since all gesture interaction happens in the control-domain at rates rarely exceeding 100 Hz, the entire set of objects presented here runs on the message level and depends ideally on one central pulse or clock for animation.

6.1 Transform Matrix: *3Dmatrix*

The *3Dmatrix* external is the core of this system and executes the geometrical transformations on point data. It implements routines for all four standard matrix-operations: scale, translate, rotate and skew (also called shear). Since these operations are not commutable – the order of execution counts – several modes of operation are available. They range from single operations to chains of transforms. These sequences may even contain repetitions of an operation within the chain.

6.2 Curves: *Bezier and Bspline*

The *Bezier* external and its more general sibling, the *Bspline* external, are used to define and execute motion along curved trajectories. Splines are well known from vector graphics and DSP interpolations. These two externals execute their polynomial

calculations in various orders from either four (*Bezier*) or N (*Bspline*) control points and output the result either in a fixed number of steps or calculate the exact resulting point from the input of a fractional position along its length.

6.3 Visualization: *Gaze*

The *Gaze* external implements a perspective projection algorithm that allows calculating the projected position of a 3d point onto a 2D surface. It can be considered a 'virtual camera' for points, offering all standard camera controls known from 3D modeling and rendering applications. Its main use is in visualizing trajectories and source placements without the use of an OpenGL context, by simply generating the projection information needed to draw a 3D scene in the standard MaxMSP object *LCD*. Since there is no equivalent tool available in PureData the shapes should be displayed in the real 3D environment using GEM.

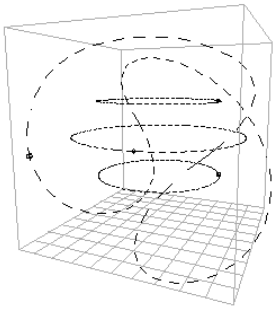

Figure 4. Visualisation of four object trajectories using the *Gaze* external to draw to *Lcd* in MaxMSP

6.4 XML Handling: *Intox, Detox and Toxine*

The preferred format for storing and transmitting gesture data and module settings is as a human-readable XML file. In keeping with the philosophy of small and portable modules written in C, the three externals *Intox, Detox* and *Toxine* were created. *Toxine* is the file I/O utility for reading and writing the XML files from or to disk. *Detox* is a simple streaming parser that outputs the XML entities tagged with their respective types and tag-tree. *Intox* is a simple entabbing utility, used for formatting the text lines before they are written out to disk. The *Detox* external is also being used by many people for parsing web pages, databases and other datasets encoded in XML.

7. EXAMPLE

The following example (Figure 5.) serves as an illustration of the proposed methodology. The four points are controlled bottom-up directly at their node by a classic Boids algorithm [2][9]. On the branch-level these four points are grouped into two groups of two. The branches are controlled top-down by a value extracted from a glove-gesture. This value describes the amount of bend on the fingers and is used to control the displacement properties of the transform. The root transform is controlled by the horizontal heading calculated from the angular vector obtained from a three dimensional accelerometer. This example shows the typical mixture between direct control and emergent algorithmic behavior. The Performer acts on the overall "gestalt" or positions of the points but not on their individual values thus producing a dynamic and fluctuating overall source behavior.

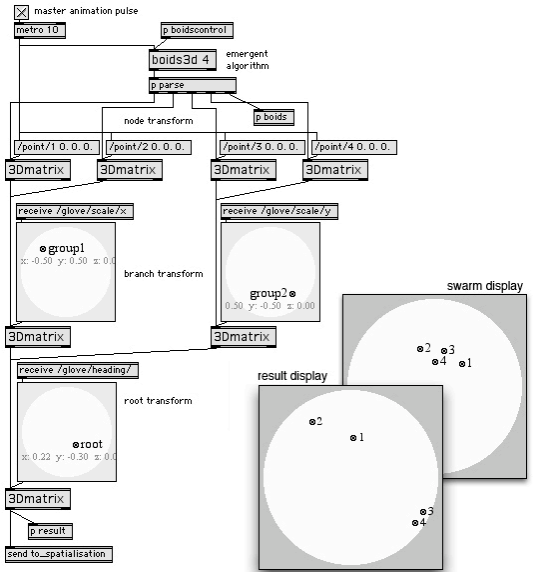

Figure 5. A simplified MaxMSP patch where four points are controlled by a swarm-algorithm at node-level and by gesture data at branch- and root-level.

8. CONCLUSION

This paper presents an approach to gesture control of sources in periphonic surround sound. The use of a few key software components in a modular structure should permit the construction of frameworks for gesture interaction that are simple to use and powerful in their expression. The main problem with this type of approach is that there is no real point of reference apart from individual experiences, neither in the literature nor in compositional practice. A language describing source motion and spatial placement is missing and has yet to be defined. While borrowing from 3D modeling and graphics as well as from choreographical notation and aeronautics gives us the tools for the transformations, there is no unified language for using them for musical expression. The proposed GDIF format and the modular approach promoted here, together with the use of OSC-namespaces [17] and XML files, are an attempt at providing a first rough sketch in that direction. This paper has voluntarily left aside the specifics of mapping since there is already a variety of an approach in use. The idea is to provide a framework in which to apply different mappings without enforcing one specific method. Possible drawbacks of this openness are the increased levels of indirection, especially when compared to more direct mapping strategies such as classic one-to-many connections normally used in pan-pots or pre-edited trajectories and the necessity to build a dynamic translation layer to adapt controller values to heterogeneous types of transform values.

Future work on the toolset should include a data structure to hold object-data and routines to execute spatial searches on large sets of points like collision/neighbor detection and different emergent swarming behaviors.

A collection of example patches for MaxMSP accompanying the publication of this paper is available for download. [11]

9. ACKNOWLEDGMENTS

I would like to thank Sergi Jordà of the IUA at Pompeu Fabra University, Barcelona for his support and for providing inspiration during my studies and degree work.

10. REFERENCES

[1] Bevilacqua, F. Müller, R. Schnell, N. *MnM: a Max/MSP mapping toolbox,* Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada pp. 85-88

[2] Bisig, D. *Interactive Swarm Orchestra* http://www.ifi.unizh.ch/ailab/people/dbisig/iso.html URI valid April 2007

[3] Chowning, J. M. *The Simulation of Moving Sound Sources*. J. Audio Eng. Soc.19, 2-6, 1971.

[4] Henry, C. *PMPD physical modelling for PD (also available for MaxMSP)* http://drpichon.free.fr/pmpd/ URI valid April 2007

[5] Jensenius, A.R.. Kvifte, T. Godøy, R.I. *Towards a Gesture Description Interchange Format,* Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France pp. 176-179

[6] Marshall, M.T., Peters., N. Jensenius. A.R., Boissinot, J. Wanderley, M.M. Braasch, J. *On the Development of a Gesture Control of Spatialisation.* Proceedings of the International Conference on Computer Music 2006 (ICMC'06) New Orleans, November 6-11, 2006

[7] Puckette, Miller *Pure Data*. Proceedings, International Computer Music Conference. San Francisco 1996: International Computer Music Association, pp. 269-272.

[8] Pulkki,V. Virtual sound source positioning using vector base amplitude panning, Journal of the Audio Engineering Society, 45(6) pp. 456-466, June 1997.

[9] Reynolds, C.W. *Flocks, Herds, and Schools: A Distributed Behavioral Model*, Proceedings SIGGRAPH 87 (Computer Graphics 21(4), July 1987, edited by Maureen C. Stone, pages 25-34).

[10] Rodet, X. et al. *Study of haptic and visual interaction for sound and music control in the Phase project.* Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada pp. 109-114

[11] Schacher, J.C. *Jasch Objects*, externals for MaxMSP http://www.jasch.ch/dl/ and ported partially to PD as *jasch_lib*, http://pure-data.cvs.sourceforge.net/pure-data/ URI valid April 2007

[12] Schacher, J.C. *Nomos V | Curved Spaces*, Dissertation, Master in Digital Arts, University Pompeu Fabra, Barcelona, Spain, (2002)

[13] Schacher, J.C. Kocher, P. *Ambisonics Spatialization Tools for Max/MSP,* Proceedings of the International Conference on Computer Music 2006 (ICMC'06) New Orleans, November 6-11, 2006

[14] Singer, E. Sier, A. Smith, W. Schacher, J.C. *Boids externals for MaxMSP and Jitter* http://www.ericsinger.com, http://www.mat.ucsb.edu/~whsmith/boids.html URI valid April 2007

[15] Smith. R. *Open Dynamics Engine ODE http://ode.org/* URI valid April 2007

[16] Steiner, H.C. *[hid] toolkit: a unified framework for instrument design* Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver, BC, Canada pp. 140-143

[17] Wright, M. Freed, A. *Open Sound Control : A New Protocol for Communicating with sound Synthesizers* In Proceedings of the International Computer Music Conference (ICMC), 1997.

[18] Zicarelli, David *An Extensible Real-Time Signal Processing Environment for Max.* In Proceedings of the International Computer Music Conference 1998. International Computer Music Association, Ann Arbor, 1998, pp. 463-466