SOUND SPATIALIZATION WITH PARTICLE SYSTEMS

Dr. David Kim-Boyle

University of Maryland, Baltimore County
Dept. of Music, 1000 Hilltop Circle, Baltimore, MD, U.S.A.
`kimboyle@umbc.edu`

ABSTRACT

The author describes recent research in the use of particle systems for sound spatialization. A MaxMSP/Jitter patch is presented that maps the spatial trajectories of the individual particles in a particle system to the spatial movement of individual granular voices in a simple granular sampling patch. The ability to model the spatial trajectories of natural phenomena is also explored, as are applications outside granular sampling.

1. INTRODUCTION

Mapping spatial trajectories with particle systems can enable unique musical effects to be realized. While particle systems find their natural sonic realization in granular synthesis, the spatial distributive techniques typically employed in the latter are crude at best with grains most often randomly or stochastically distributed within predefined boundaries. [1] A more interesting spatial application of particle systems can enable more natural and complex types of movement, such as the movement of a cloud of smoke, the foam of a wave or the flight of a flock of birds, [2] to be realized.

Using the basic particle system objects of Cycling '74's Jitter, [3] the author will describe how these objects can be integrated into a basic granular sampling patch to enable distinct spatial trajectories to be realized. While these trajectories are naturally more suited to being heard on a multichannel, surround playback system, they can also, with slight modification, be mapped to a simple stereo system.

2. JITTER PARTICLE SYSTEMS

Jitter is a set of external graphical objects for the Max/MSP programming environment. It allows live video processing, OpenGL animations and other graphical effects to be integrated into the MSP audio platform. There are three Jitter objects used to model particle systems - jit.p.shiva, jit.p.vishnu and jit.p.bounds.

The jit.p.shiva object generates Jitter matrices where particles have unique ids, life expectancies, x, y and z coordinates and velocities. The number of particles emitted per frame and their life expectancies can be modified. The jit.p.vishnu object applies global forces to the matrix generated by the jit.p.shiva object modifying the x, y and z coordinates and velocities of the particles. The angle of particle emission relative to the horizontal plane and the speed of particle emission are among the variables that can be modified. The jit.p.bounds object confines the particles in a matrix to specific x, y and z limits and also determines the behavior of the particles when they reach the matrix boundaries - particles may either bounce, wrap around, remain at the matrix boundary or die. The jit.p.bounds object also allows the elasticity of the system boundaries to be modified. Complex patterns can be created when the particle system parameters are dynamically updated. OpenGL renderings of the particle systems enabled by these three objects are shown in Figure 1 with each particle rendered as a point.
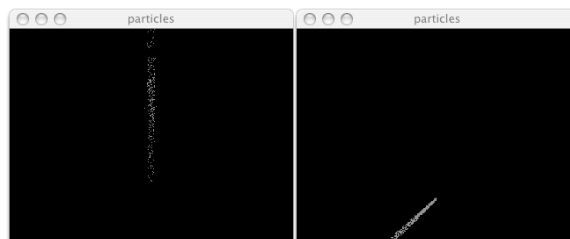

Figure 1: *A particle system with a narrowly bounded particle stream (left) and a particle system with a narrowly bounded particle stream that skews to the left (right).*

The x, y, and z coordinates of each particle in the generated matrix can be indexed with the jit.peek~ object and written to a buffer as illustrated in Figure 2. The particle values in a matrix are stored in the first row of a n column matrix where n is the number of particles, hence the use of a constant zero index. The second row of the matrix contains the value of the previous frame.
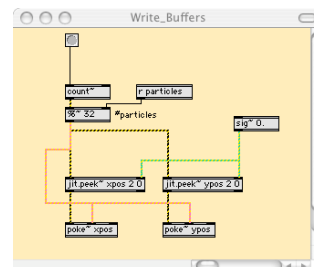

Figure 2: *Storing matrix values in signal buffers.*

Before the coordinates of the particles are written to buffers they can be further modified with basic mathematical functions. This can help realize some trajectories that are difficult to achieve with the jit.p.vishnu object. For example, by applying simple sine and cosine functions to the x and y coordinates, the particles can be made to rotate around the x and y axes, see Figure 3.
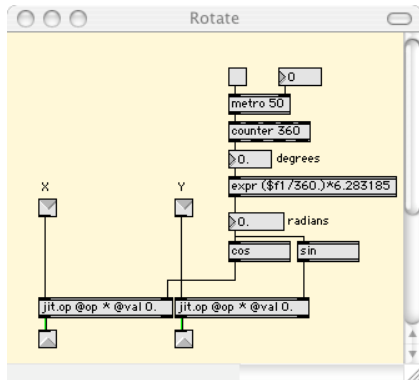

Figure 3: *Rotating particle coordinates around the x and y axes.*

3. INTEGRATION WITHIN A GRANULAR SAMPLING PATCH

The MaxMSP4.5 release contains a simple granular sampling patch designed by Les Stuck and Richard Dudas. This patch uses the Max poly object to perform dynamic voice allocation. Each voice reads a sample of a user-defined sound file with the number of grains, time between grains, duration, panning and transposition of each grain defined by the user. This basic patch has been modified by the author such that the panning of grains is determined by the Jitter particle system. The transposition functions have also been modified to simulate the doppler effect and the number of grains is defined by the number of particles. An excerpt from this patch is illustrated in Figure 4.
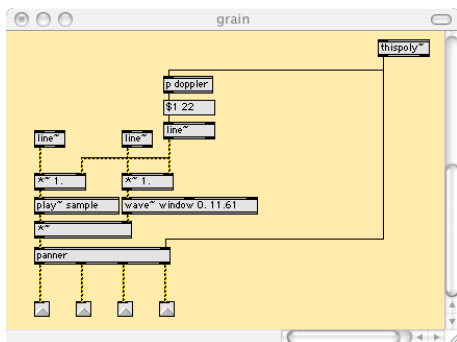

Figure 4: *Granular voice.*

Indexing the x and y buffers written to with the Jitter particle system patch gives the spatial location for each granular

voice. The x coordinate is mapped to the front left and right loudspeakers of a quadraphonic playback system while the y coordinate is mapped to the rear left and right loudspeakers. A simple sine function, from a panning patch originally designed by Les Stuck, is used to maintain a constant apparent distance of the sound source from the listener. This very simple patch is illustrated in Figure 5.
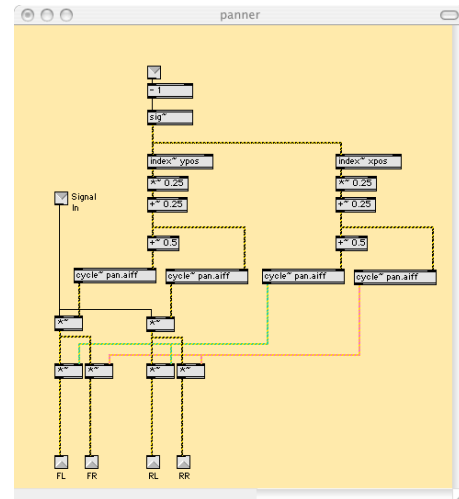

Figure 5: *Mapping particle coordinates to a quadraphonic spatial location.*

At this stage of its implementation the z coordinate of the Jitter particle is not mapped to any parameter although the author has achieved interesting effects by mapping it to a loudspeaker mounted above the listener. In this application, four small Genelec1029A studio monitors were positioned in a standard ITU 775 [4] front left/right, rear left/right surround configuration with a z coordinate monitor placed on a microphone boom directly above the listener's head, see Figure 6. While this application allows musically interesting trajectories to be defined above the listener, it is clearly limited by the very small listening space defined by the monitors.
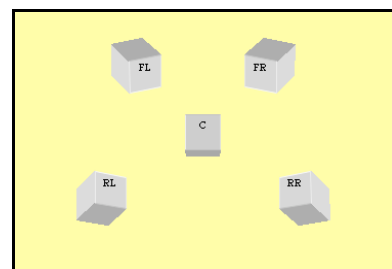

Figure 5: *A five channel playback system used for mapping the z coordinate of a particle to height.*

The author has also achieved interesting musical effects by mapping the z coordinate of each particle, or the y coordinate in a stereo playback system, to a global amplitude, inverse-square-gain control for each grain. This can give a crude impression of depth, particularly when each grain has a separate reverberant quality [5] although CPU usage quickly limits the

number of distinct reverbs that can be applied to each grain. Even with an outboard VST processor such as the TC Power-core, it is difficult to run more than 24 distinct reverbs. Despite these limitations, this has still proven to be a musically interesting application.

While including a dedicated reverb for each grain is too computationally intensive for most real-time applications, a simple doppler effect can be included which can help more realistically simulate the movement of the particles through space. [6] In this implementation, the velocity and distance of the particle from the front centre position is used to extend or compress the time it takes for a grain to be played. This results in a modification of the pitch of the sound.

4. MODELLING NATURAL SYSTEMS AND OTHER APPLICATIONS

One of the attractive qualities of particle systems is their ability to model or visually mimic natural phenomena. In order to model such systems with the Jitter particle objects, however, it is necessary to implement an additional logic whereby particles interact with one another. With the jit.p.vishnu object, all of the particles experience uniform forces. A more complex particle interaction, where particles might collide, have different masses and momentums, and experience forces from other individual particles can allow more interesting systems to be modelled.

While the author has not yet implemented all the above interactive possibilities, a simple proximity factor has been developed. This allows the spatial location of a particle to be modified if it occurs within a certain proximity of other particles. Such a feature is useful in simulating the clumping and separation forces in Craig Reynolds' well known Boids simulation. [2] Using a similar logic it is also possible to simulate an interaction with simple two dimensional objects. By placing such objects within the Jitter matrix, particles can be made to avoid certain spatial locations. This is illustrated in Figure 6. The ability to perceive the avoidance of these spatial locations is clearly affected by their size, the physical separation of the loudspeakers and the particle speed.
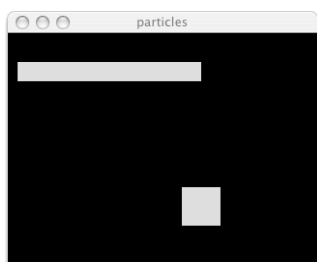

Figure 6: *Objects placed within a Jitter matrix.*

The author has also experimented with the use of Eric Singer's *boids* object, an early pre-Jitter particle system object. [8] Based on Reynolds original analysis of flocks,

Singer's object simulates the movement of a flock of birds in two dimensional space. Unfortunately, the object does not readily interface with the Jitter particle system objects as the particles in *boids* experience their own "vishnu-like" forces from parameters defined by the user. The location of the birds, or boids, is written to a simple two dimensional graphical object. By transferring the *x* and *y* coordinates to a Jitter matrix, using a similar process to that outlined eariler, and then performing simple math functions also like those outlined earlier, the flock's position can be compressed and made to rotate around the *x* and *y* axis. This is illustrated in Figure 7.
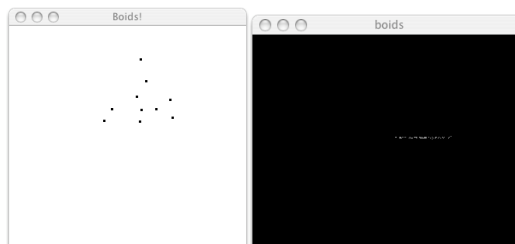

Figure 7: *Original Boids window (left), Rendered through Jitter's OpenGL objects and rotated around the x axis (right).*

Another interesting musical effect, although one that bypasses the granular implementation is achieved by mapping the movements of particles to the spatial location of individual bins in a short-time Fourier transform. With the number of particles equal to half the FFT length, the magnitude of each bin is multiplied by a coefficient read from the particle system matrix. The implementation, similar to that outlined in Torchia and Lippe's work on frequency-domain based spatial distribution, [7] allows unique spectral effects to be achieved.

5. FUTURE DEVELOPMENT

The author is extending the implementation to allow more sophisticated flocking patterns to occur. This will necessarily involve the development of more complex particle interactions. Of particular interest is the development of a three dimensional interaction with OpenGL rendered objects, where the *z* coordinates might be mapped to other musical parameters. The possibility seems rich in musical potential.

Considerable work is also taking place on the design of an interface that makes the spatial mapping possibilities more usable for live performance. Given the large amounts of information that needs to be easily accessed and controlled, especially with the FFT implementation outlined earlier, a musically well thought out interface is critical.

Finally, the work is also being extended and utilized in the composition of a new piece for flute with real-time audio and video processing.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1]   C. Roads, *Microsound*, Cambridge, MA: MIT Press, 2001.

[2]   C. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," in *Computer Graphics*, 21(4), July 1987, pp. 25-34.

[3]   D. Zicarelli, "An Extensible Real-Time Signal Processing Environment for Max," in *Proceedings of the 1998 International Computer Music Conference*, Ann Arbor, MI: International Computer Music Association, pp. 463-466, 1998.

[4]   International Telecommunication Union, <http://www.itu.int>.

[5]   J. Chowning, "The Simulation of Moving Sound Sources," in *Journal of the Audio Engineering Society*, 19, pp. 2-6, 1971.

[6]   B. L. Sturm, "Sonification of Particle Systems via de Broglie's Hypothesis," in *Proceedings of the International Conference on Auditory Display*, April 2000 <http://www.icad.org>.

[7]   R. H. Torchia and C. Lippe, "Techniques for Multi-Channel Real-Time Spatial Distribution Using Frequency-Domain Processing," in *Proceedings of the 2003 International Computer Music Conference*, Singapore: International Computer Music Association, pp. 41-44, 2003.

[8]   E. Singer, <http://www.ericsinger.com>, Accessed January 2005.