
David Kim-Boyle

University of Maryland, Baltimore
County
Dept. of Music, 1000 Hilltop Circle
Baltimore, MD 21250
U.S.A.
kimboyle@umbc.edu

Spectral and Granular Spatialization with Boids

Abstract

The author describes applications of Craig Reynolds's boids algorithm for sound spatialization. A MaxMSP/Jitter patch is presented where the movement of individual boids in two dimensional space is rendered in OpenGL and is used to map the spatial trajectories of granular voices in a granular sampling patch and also the spatial location of a sound's spectral components. Musical possibilities of the technique are discussed and some performance considerations are also outlined.

Introduction

Spatialization techniques for granular synthesis are often quite primitive with grains usually distributed randomly or stochastically within predefined boundaries. (Roads 2001) In FFT based analysis-resynthesis, the spatialization of a sound's spectral components is often not explored despite its musical potential. Using Craig Reynolds's well known *boids* algorithm, (Reynolds 1987) as implemented in a Max object designed by Eric Singer, (Singer 2005) more complex types of spatial movement can be realized in both granular sampling and FFT analysis-resynthesis techniques. While Reynolds's work has been used in granular synthesis applications, (Blackwell and Young 2004) to the best of the author's knowledge it has been used on only one other occasion for sound spatialization. (Davis & Rebelo 2005)

The spatialization techniques to be outlined have been developed for realization on a four or eight channel playback system.

Boids

Reynolds's study of the movement of flocks of birds is well documented and has been implemented in a number of algorithms. (Parker 2002, Gombert 2005) Reynolds identified three primary factors that determined the steering behavior of the flock - *separation*, *alignment* and *cohesion*. These three parameters refer, respectively, to the preferable distance one bird would maintain from another, the tendency of a bird to fly towards the average heading of its local neighbors, and the tendency of a bird to steer towards the average position of its local neighbors. (Reynolds 1987) Using these parameters in computer animation applications,

Reynolds was able to realistically simulate natural flocking movements.

Based on Reynolds's original analysis, Eric Singer's *boids* object is an early pre-Jitter Max object which simulates flock movement in two dimensional space - Reynolds's original implementation was in three dimensions. Singer's object provides the *x* and *y* location of each boid as its flight is affected by various flock control parameters. While some of these correspond to Reynolds's original analysis, for example the centering instincts of the boids, the tendency of a boid to avoid its neighbor or the attraction of a boid to a particular point, other parameters are unique - the inertia of the boid or its willingness to change speed and direction and the number of neighbors a boid consults when flocking. The window displaying the boid's movements is shown in Figure 1.

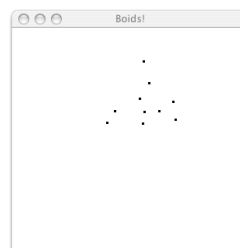


Figure 1. Singer's Boids window

In Singer's object, the trajectory of the flock is determined by a control device such as a mouse or trackpad. By changing the attraction, inertia and speed parameters, the tendency of the flock to respond to this movement can be radically affected. While defining trajectories with the mouse is useful at times, the author has also experimented with simple circular trajectories, the definition of trajectories with various alternate controllers, and the modification of trajectories with performance data.

Jitter Implementation

Interesting transformations of the boid's movements can be performed with standard Max and Jitter objects. Once the coordinates have been transferred to a Jitter matrix, they can also be rendered in OpenGL which provides immediate visual feedback of their spatial location. Rendering the boids in different colors is also useful for com-

posers or performers who need to quickly identify the spatial location of particular boids. This is also especially helpful in the spectral applications which will be outlined later. The x/y coordinates are written to buffers as illustrated in Figure 2.

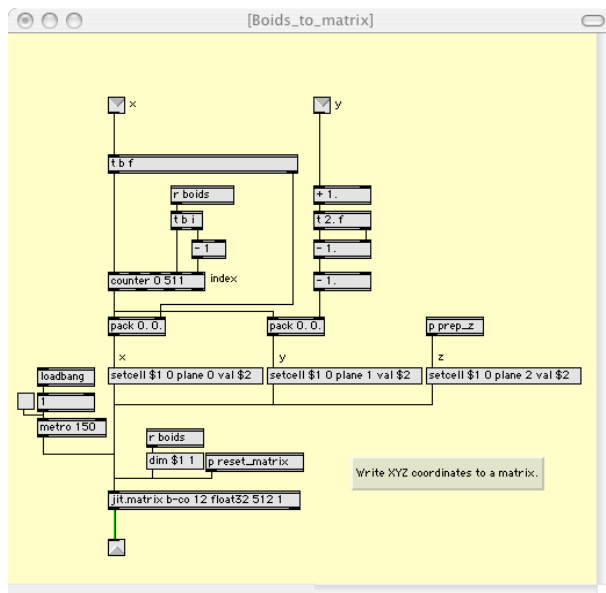


Figure 2. Storing boid coordinates in a matrix

The x/y coordinates are stored in the first row of a n column matrix where n is the number of boids. A dummy z coordinate, which is not provided by the *boids* object, is also mapped as this is required by the OpenGL rendering.

Before the coordinates of the boids are written to buffers they can be further modified with basic mathematical functions. This can help transform the boid's movements and subsequent sounds in ways that are not possible with the *boids* object alone. The author has implemented a number of useful transforms as illustrated in Figure 3. These transforms include the ability to add an offset to a coordinate, to squish the coordinates into a particular region in space, the ability to apply a separate rotational force to coordinates, and the ability to map regions in space into which the boids will not fly.

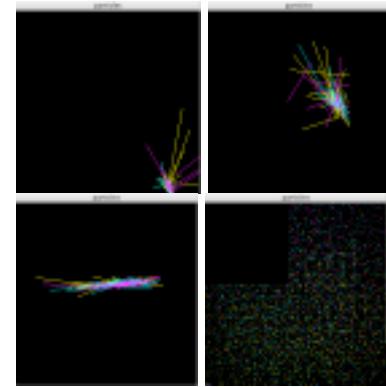


Figure 3. Transforming x/y coordinates. From top left, a) not modified (rendered as lines), b) with an x offset (rendered as lines). From bottom left, c) with squished y coordinates (rendered as lines), d) avoiding the upper left quadrant (rendered as points)

One obvious problem with the use of OpenGL renderings for visual feedback is that there can often be enormous amounts of data presented. Particularly in the spectral applications outlined later, it can become difficult to track the individual movements of the boids, even with discrete color mappings for each boid although in practice this is perhaps not as necessary as it is to be able to get feedback on the entire flock's trajectory. There are, however, interesting musical applications of the types of renderings shown in Figure 3, especially if a performer is asked to respond to particular spatial distributions.

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 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 patch. The transposition functions have also been modified to simulate the doppler effect and the number of grains is defined by the number of boids.

Indexing the x and y buffers written to with the Jitter patch outlined earlier gives the spatial location of each granular voice. The x coordinate is mapped to the front left and right loudspeakers of a quadrasonic 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 4.

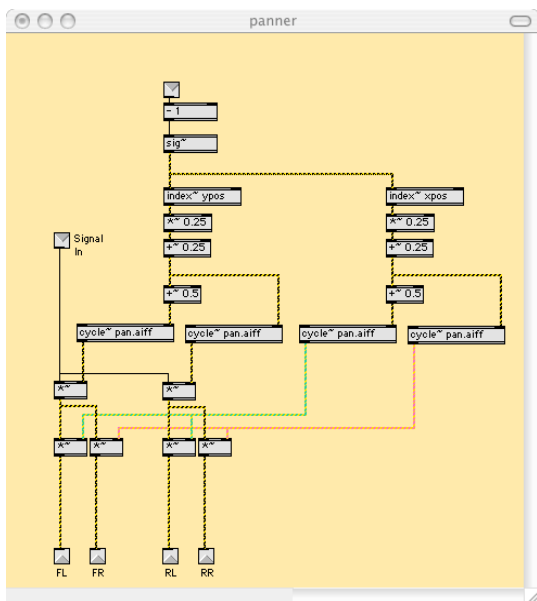


Figure 4. Mapping particle coordinates to a quadrasonic spatial location

In the granular implementation, the z coordinate is mapped to a global amplitude, inverse-square gain control for each particle. This can give a crude impression of depth, particularly when each voice has a separate reverberant quality (Chowning 1971) although CPU usage quickly limits the number of distinct reverbs that can be applied to each grain.

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 voices through space. (Chowning 1971) In this implementation, the velocity and distance of the boid 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 slight modification of the pitch of the sound.

FFT Implementation

Mapping the movements of the boids to the spatial location of individual bins in a short-time Fourier transform can produce some interesting musical effects. With the number of boids equal to half the FFT length, the magnitude of each bin is multiplied by a coefficient read from the Jitter matrix. The implementation is similar to that outlined in Torchia and Lippe's work on frequency-domain based spatial distribution, (Torchia & Lippe 2003) although the integration with the

boids object and the use of the z coordinates allows a more sophisticated level of control. The FFT patch is illustrated in Figure 5.

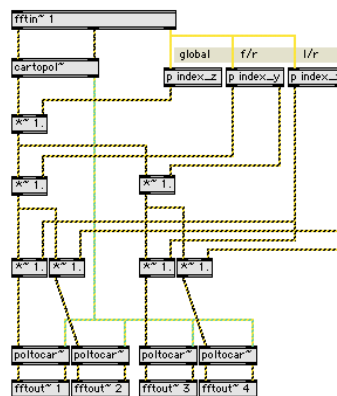


Figure 5. FFT Spatialization patch

As outlined earlier, the individual bins of the FFT are mapped to different colours in the OpenGL rendering - lower bins to reds, higher bins to violet and those between, the orange through indigo colour spectrum. While this is useful for tracking the movements of certain spectral bands it requires a 12-plane Jitter matrix to be used which can slow down performance - a significant consideration if the visual information is to be used for real-time performance.

The *squish* transformation outlined earlier also has interesting applications in the FFT implementation. With a high squish factor, all the bins of the FFT can be spatially distributed to one point source - or in other words the sound will simply appear to come from one location. As the squish factor is increased the spectrum can be dislocated in the *x/y* plane. This is illustrated in Figure 6.

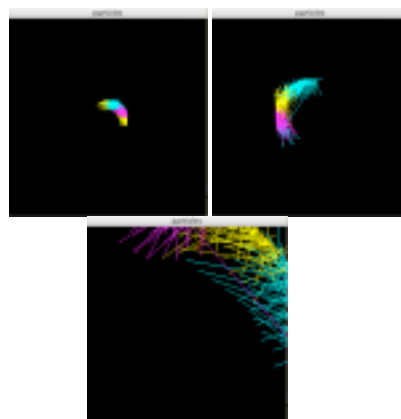


Figure 6. Dislocating a spectrum through the squish factor

The ability to perceive these spatial movements depends on many factors including the size of the FFT, the speed of the boid movement and the

physical size of the playback environment. For most practical purposes the author has found that there is little need to use FFT sizes much greater than 256 bins as the various individual trajectories cannot be easily perceived. This also has the benefit of simplifying the visual information presented.

Spectral delays, determined from the z coordinate of a boid, are also being incorporated into the patch. (Kim-Boyle 2004) Although the z coordinate is not provided by Singer's object, it can be generated from another source or randomly determined. When the squish and offset parameters are dynamically varied during performance in the x , y and z dimensions interesting timbral effects can be achieved with certain spectral components of the sound lagging behind others or preceding them. With natural sound sources whose timbre's are dependent on features such as attack transients, the transformations can be particularly arresting.

Performance Considerations

While the parameters that affect the flocking behaviors of the boids can all be easily controlled via qlists and other such compositional devices, a carefully considered interface is clearly required for applications in improvisatory style performances. Many of the concerns are common to all interface design techniques - what is the most useful way to present information, which information needs to be presented, how should that information be controlled etc.. (Tufte 2001) Who the information is intended for is also an important consideration. If the performer is to be able to musically respond to the spatial mappings a whole new range of issues is raised including the ability to track complex movements during performance. Another important consideration involves the musical control of FFT data. For most practical applications, there is probably little need to have direct control over every FFT-bin's spatial location. Indeed the concept of a bin is not especially helpful from a musical perspective. The ability to control the spatial location of fundamentals and harmonics and inharmonics would seem to be more useful.

The use of envelopes to control boid parameters has proven to be musically useful. Rather than having the various parameters remain fixed they can change as defined by a simple envelope generator. The author has also experimented with the manipulation of parameters from performance data. For example, it is fairly simple to change the speed that the boids fly around with data obtained from a real-time timbral analysis. Sounds with a high noise content might be mapped to low

speeds and sounds with a more oscillatory content might cause the boids to move more quickly.

Musical Applications

The application of Reynolds's *boids* for spectral and granular spatialization has many interesting musical applications. In the author's own work, the implementations discussed have been applied in a new work for bass clarinet and computer. In this work, bass clarinet multiphonics are analyzed with the strength of the harmonic components used to modify the spatial trajectories of the computer processed sounds. In this work the attraction of one boid to another is also decreased as the piece develops which results in a disintegration of various timbral structures. A similar technique is being employed in another work for cello and computer where sounds with greater noise delimit the spatial distribution of the computer generated sounds to a point location in space. As the cello sounds become more harmonic by varying finger and bow pressure, the spectral components of the computer generated sounds become more spatially scattered.

Acknowledgements

The author would like to thank Cort Lippe, from the State University of New York at Buffalo, and Chandrashekar Ramakrishnan from ZKM for their helpful suggestions during work on this project.

References

- Blackwell, T & Young, M 2004, "Swarm generator", in *Applications of Evolutionary Computing: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoASP, EvoMUSART, and EvoSTOC, Coimbra, Portugal, April 5-7, 2004*, eds G Raidl et al., Springer, Berlin, pp. 399-408.
- Chowning, J 1971, "The simulation of moving sound sources", *Journal of the Audio Engineering Society*, 19, pp. 2-6.
- Davis, T & Rebelo, P 2005, "Hearing emergence: towards sound based self organization" in *Proceedings of the 2005 International Computer Music Conference, Barcelona*, pp. 463-466.
- Gombert, J 2005, "Real-time simulation of herds moving over terrain", viewed December 2005, <http://consystlab.unl.edu/our_work/Papers/AIIDE05.pdf>.
- Kim-Boyle, D 2004, "Spectral delays with frequency domain processing", in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX-04)*, Naples, pp. 42-44.

- Parker, C 2002, viewed June 2005,
<http://www.vergenet.net/~conrad/boids/ps_eudocode.html>.
- Reynolds, C 1987, "Flocks, herds and schools: a distributed behavioral model", *Computer Graphics*, 21(4), pp. 23-34.
- Reynolds, C 2001, *Boids: Background and Update*. viewed November 2005,
<<http://www.red3d.com/cwr/boids/>>.
- Roads, C 2001, *Microsound*, MIT Press, Cambridge, MA.
- Singer, E 2005, viewed May 2005,
<<http://www.ericssinger.com>>.
- Torchia, R & Lippe, C 2003, "Techniques for multi-channel real-time spatial distribution using frequency-domain processing", in *Proceedings of the 2003 International Computer Music Conference*, Singapore, pp. 41-44.
- Tufte, E 2001, *The visual display of quantitative information*. 2nd edn, Graphics Press, Cheshire, CN.