



COVER SHEET

Brown, Andrew R. and Jenkins, Greg (2004) The Interactive Dynamic Stochastic Synthesizer. In Norris, Michael, Eds. *Proceedings Australasian Computer Music Conference*, pages pp. 18-22, Wellington, New Zealand.

Accessed from <http://eprints.qut.edu.au>

Copyright 2004 the authors.

The Interactive Dynamic Stochastic Synthesizer¹

Andrew R. Brown †* & Greg Jenkins †

† Queensland University of Technology (QUT)

* The Australasian CRC for Interaction Design (ACID)

a.brown@qut.edu.au

g.jenkins@qut.edu.au

Abstract

Throughout musical history new sounds and instruments have opened new opportunities for music making. In this paper we outline a new interactive digital instrument that implements the dynamic stochastic synthesis algorithm devised by Iannis Xenakis. We discuss the history and operation of this synthesis process, previous implementations of it, and how our implementation is the first we know of designed specifically for live performance. Finally, the behaviour tendencies of the synthesis system and how these impact upon interactivity are discussed.

1 Introduction

The role of chance and random occurrence as a musical technique has played a significant part in Western compositional developments in the 20th century. Among the leaders of this development, that include John Cage, Gottfried Michael Koenig, Karlheinz Stockhausen, Lejaren Hiller & Leonard Issacson, and Luciano Berio, Charles Dodge and Brian Eno, is Iannis Xenakis.

Xenakis's interest in stochastic systems was informed by modern scientific theories including probability theory derived from quantum mechanics, and kinetic theories of gases. His conception of stochastic processes was as indeterminate functions with entropic tendencies or, more simply, processes that "evolve in different directions" and move between states of "order to disorder, or vice versa" [6:16].

Uniquely, Xenakis applied probability not only to the organisation of musical objects and structures but also directly to the generation of sound waves, with a process he called Dynamic Stochastic Synthesis (DSS). This synthesis process applies constrained random processes to the moment by moment fluctuations in air pressure that are fundamental to sound creation.

In this paper we outline an implementation of the dynamic stochastic synthesis algorithm that is intended for real-time performance, discuss some of the historical contexts that inform the design and discuss issues of interaction design that have arisen during its use in performance.

2 Dynamic Stochastic Synthesis

For most of his life Iannis Xenakis was concerned with the use of probabilities for creating music. He conceived of DSS as a way to apply stochastic functions to the generation of audio waveforms. His early compositional uses of stochastic processes were primarily in determining aspects of works at the level of note attribute selection and the organisation of musical form. It was not until later in his life, that he applied these processes directly to the dynamic stochastic synthesis process although he had previously worked with stochastic functions as elements of synthesis parameter control in processes including granular and FM synthesis. Xenakis termed the organisation of music at this sample-by-sample level, microstructure [7].

Dynamic Stochastic Synthesis involves multiple levels of probabilistic functions that determine the positions of break points in an envelope which in turn describes one cycle of an audio waveform, as shown in figure 1. The number of points in the waveform is predetermined by the user or programmer.

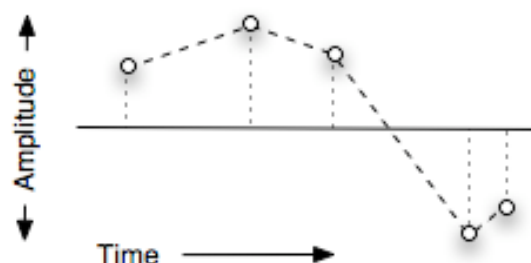


Figure 1. A breakpoint waveform description

¹ Brown, A. R. & Jenkins, G. 2004. "The Interactive Dynamic Stochastic Synthesizer" *Proceedings of the Australasian Computer Music Conference 2004*. Wellington, NZ: ACMA, pp. 18-22

The amplitude and time position of each break point are varied at each repetition by a random walk function. A random walk is a constrained stochastic function where each subsequent event is selected at random within a limited range above or below the previous value. The pitch (cycle duration) and dynamic (peak amplitude) are thus constantly varying from cycle to cycle. Each amplitude-time point varies independently and they all change concurrently. Slight variations produce a more stable ‘phasing pitch glide’ sound and larger variations result in a more ‘noisy’ timbre. The amplitude and pitch range can be compressed by setting maximum values for each, what Xenakis calls “elastic barriers.”

Sample values for the waveform are calculated by interpolating between the stochastically generated amplitude-time points. The resolution of the interpolation is traditionally quite coarse, adding to the ‘low tech’ character of the DSS sounds (Xenakis, I. 1991; [2]).

Part of the appeal of this process to Xenakis was its computational efficiency and the fact that it did not rely on a harmonic or Fourier conception of the sound world. He comments that “the challenge is to create music, starting, in so far as it is possible, from a minimum number of premises but which would be ‘interesting’ from a contemporary aesthetic sensitivity, without borrowing or getting trapped in known paths” (Xenakis 1991:295).

Xenakis had been explicitly exploring stochastic systems in his works as early as 1957 with *Achorripsis* and, not long after, with the *ST* (Stochastic) series of compositions. He had been using random walks in his instrumental compositions since the 1970s, the most paradigmatic of which was *Erikthos* (1977) where both note level and larger structures were heavily determined by random walks [1] [3]. The extension to use stochastic functions to control audio signal generation was an insightful one, even if it appears as a natural extension of his work in hindsight.

The use of non-linear processes at the level of microstructure and the constant updating of the waveform, means that DSS has an ‘organic’ quality because it is in a constant state of evolution. The dynamic qualities of continual development and the emergent nature of the sound’s pitch, loudness and timbre result in a system that can produce a wide variety of outcomes, but is not always easy to control. In the early 1990s when Xenakis was able to realise a complete DSS implementation, almost 20 years after the initial conceptions, the system was a non real-time computer assisted compositional tool. This paper outlines some of our attempts to enhance the performability of DSS.

3 Previous DSS Implementations

The original implementation of dynamic stochastic synthesis was in the *Gendyn* program,

written by Xenakis in BASIC with the assistance of Marie-Hélène Serra [4]. The program generated completed works including *Gendy3* (1991) and *S.709* (1992) by employing stochastic processes at both the macro and micro structural levels.

The *Gendyn* program introduces complexities into DSS beyond those described in the previous section. At the level of microstructure these include the cascading of two random walks for each of amplitude and time rather than one, and the use of “mirrors” for maximum boundaries that reflect values above the maximum back into the desired range, rather than “brick wall” limiters that result in significant areas of peak values.

At the level of macrostructure, the *Gendyn* program controls the form of the piece enabling it to produce a complete work, not just a continuous sound stream. The program allows for multiple voices, 16 in the case of Xenakis’ implementation. Short periods of activity or silence in each voice are stochastically determined which produces a kind of chaotic rhythmic counterpoint. Xenakis refers these short periods as ‘fields.’ A probabilistic choice about the number and start location of these fields determines the texture and duration of the piece overall.

A faithful re-implementation of the *Gendyn* program was undertaken by Peter Hoffman in the mid 1990s. This version, which Hoffman called *The New GENDYN Program* [2], was written in C++ and Visual Basic and was efficient enough to run in real-time. Hoffman’s intention was to gain a full understanding of the *Gendyn* program and his implementation even reproduced some of the programming ‘errors’ in the original implementation that had important sonic results. Our implementation, described later in this paper, while building on Hoffman’s work has a different purpose. It intends to make the opportunities of DSS available to the computer music performer, and to add some extensions that broaden the sonic potential even further.

The *Stochos* program [1] employs many functions used by Xenakis including elements of dynamic stochastic synthesis. The application, implemented in Max/MSP, is a computer-assisted composition tool that enables the user to apply its functions to a composition’s macro and micro structure and to work in real-time or render audio files if the processing load exceeds real-time performance limits. As well as enabling DSS as one of the synthesis process in *Stochos*, Bokesoy and Pape apply the DSS concepts of stochastic line segment envelope curves, and mirrors as reflective limiters of stochastic data, as general tools throughout their program. While *Stochos* has a broader range of stochastic sonic functions than *Gendyn* or *New GENDYN*, they each explore the features of sonic space opened up by dynamic stochastic synthesis.

4 Sonic Behaviour

In a stable state where the random walk variations of both amplitude and time are reduced to zero, the DSS process acts as a stable oscillator at a fixed pitch. The precise sonic spectrum depends on the wave shape but the tendency is for a bright ‘buzzy’ tone, not unlike a sawtooth wave. Given enough points in the wave envelope and a lot of luck in freezing the points, a simple tone close to a sine wave is theoretically possible.

The addition of a small amount of amplitude variation creates a warmer chorusing or phasing effect. Allowing slight variations to the time positions of the points results in a wandering pitch glissando, at times not unlike the sound of a Blow Fly. Larger random walk steps in amplitude and time positions cause the sound to become frantic, a result reminiscent of asynchronous granular synthesis with random pitch shifting or of waveshaping. This is not surprising given that Hoffman describes DSS as “a non-linear stochastic distortion of the shape of the waveform over time” [2:32]. Further increasing of the random walk step size causes the timbre to approach Brownian noise or, with tightly constrained time mirror boundaries, white noise.

While the shape of the waveform has a significant effect on the timbre of the sound, the characteristic behaviour of DSS derives from this non-linear change over time. Hoffman suggests that “it is this change that makes up the specific quality of GENDYN sound by continually transforming its spectrum” [2:36]. He goes on to explain that DSS is a dynamic system where the dynamic behaviour of the system is fractal in nature (like all random walks) and, further, that the cascading of random walks means that the sound is governed by ‘strange’ attractors. When using the system for performance the challenge is to balance the interest generated by instability with a degree of control that enables musical expressiveness. The performer of DSS systems will have better control given an understanding of dynamic system tendencies and the importance of the relationship between the successive random walk states in creating different attractor diffusion patterns.

Another variable in DSS is the choice of stochastic distribution. For example, random functions usually generate a linear distribution, but Gaussian, Cauchy, Logistic and Poisson functions were favourites of Xenakis who explored the different tendencies of each in some depth. The general sound of the DSS system is similar regardless of the distribution function, but the behaviour of the system over time can be significantly affected by changes in stochastic distribution functions and their parameters.

The role of the mirrors in DSS is to constrain the boundary limits of the sonic space. In the case of amplitude change the mirrors act as dynamic compressors and also as peak level limiters. The reflective nature of the mirror boundaries also means

that at more constrained amplitude ranges they have a timbral effect in producing jumps in the waveform’s fundamental pitch within the overtone series because the reflections cause “transient inner symmetries” in the waveform [2:37]. The time mirrors act to constrain the pitch range, with more restricted boundaries forcing the pitch range into higher and narrower frequency regions. When generating broad spectrum signals (noisy sounds) the time mirrors can act like a high pass filter.

Because the waveform is made up from line segments between points, the resolution of the rendering of those into sample values can effect the sound. Lower sample bit depths result in a ‘course’ sound, because of associated changes in quantisation noise. This will also effect dynamic range, of course. Changes in sample rate can act as a crude low pass filter by changing the upper frequency limit.

Hoffman’s analysis of the sonic behaviour of Xenakis’s original algorithm and code were incisive and although his implementation could be altered in real time, the system was focussed more on reflective composition and analysis than real time performance. The aim of our implementation of DSS is to make these behavioural characteristics available to the performing computer musician in a way that he/she can best exploit these tendencies.

5 An interactive implementation

In order to explore dynamic stochastic synthesis in live performance a real-time system was required. In this section we outline the considerations and design of an Interactive Dynamic Stochastic Synthesizer (IDSS) program.

The need for a new implementation was several fold. Xenakis’ original *Gendyn* software was not real-time, nor was it available. Hoffman’s *New GENDYN* operates in real-time but is not focused on performance control and was designed for the Windows 95/98 platform. The *Stochos* system is possibly more complex than required or even suitable for performance and in any case was not published until after our implementation was written. Finally, the benefits of fully understanding DSS through implementing it in software are significant in their own right, and an informative aspect of this creative/research project.

The IDSS is a real-time application developed in Java and uses the audio system from the jMusic library [5] which includes the DSS algorithm which was added to jMusic as a consequence of this project.

The IDSS program features a graphical user interface (GUI), one channel strip of which is shown in figure 2. The top area of the interface has start and stop buttons, below which are sliders for each of the random walks (1 & 2) that specify the step size (Step1 and 2) and mirror positions (Max1 and 2) for amplitude (A) and time (T).

There is a visual display of the animated waveform and the secondary mirrors are also displayed here. Sliders to the left of and below the display box control the secondary mirrors for amplitude and time respectively. In the bottom left corner of the wave display area are triangular buttons that move secondary time and amplitude mirrors concurrently.

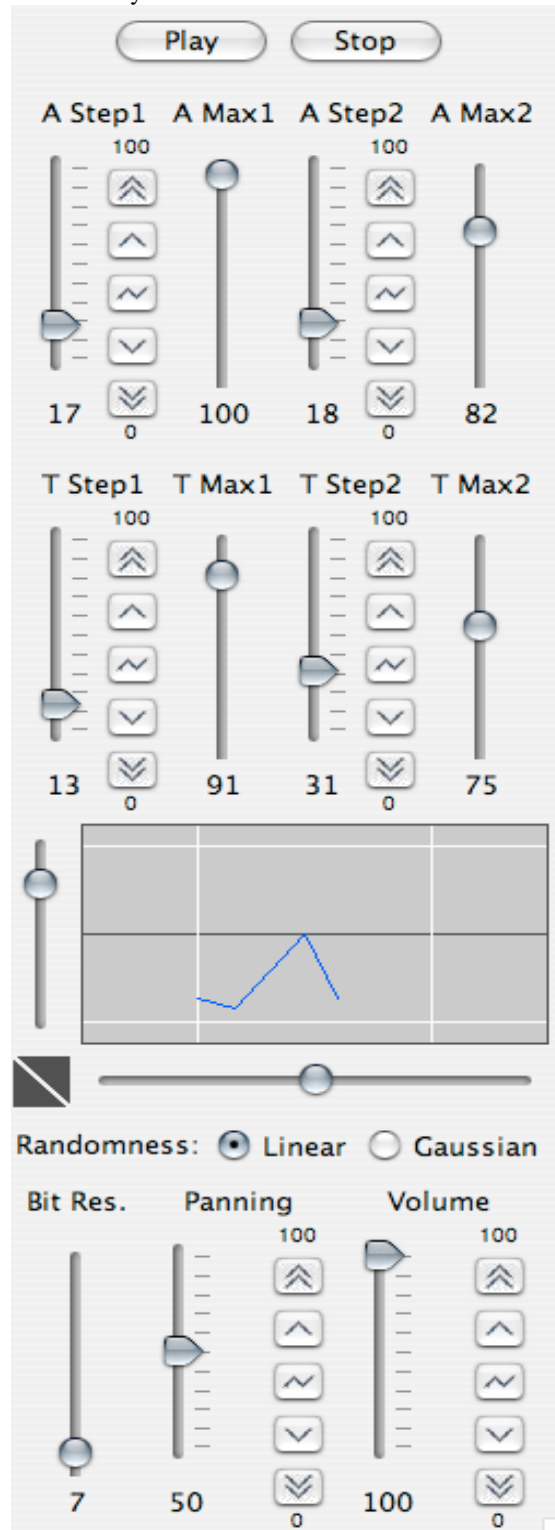


Figure 2. The graphical interface of one IDSS voice.

Below the wave display area are radio buttons for choosing the stochastic distribution type. Linear and Gaussian distributions are available. Below these are sliders to control sample bit depth, panning and volume gain for this voice.

To assist in the performability of the application, selected sliders have associated automated controller buttons. These buttons toggle fades (slow and fast, up and down) and the central button starts fader oscillation up and down repeatedly. The maximum and minimum values between which automation occurs are specified by numbers above and below the button array. These max and min values can be adjusted by clicking dragging on the number display. The automated faders are particularly useful in the control of multiple DSS voices as they offer the performer dynamic control over multiple parameters simultaneously when using the ubiquitous point and click controller (mouse/track pad). Computer keyboard shortcut commands for several functions also improve interaction in this regard. This level of multiple parameter control has been an important consideration of IDSS from its inception, in an effort to avoid the 'serial change' restriction often encountered in 'laptop' performances.

6. Interaction Considerations

During performance each IDSS voice produces alternate active and silent sound fields of variable duration, therefore the main task of playing the software is to vary parameters to stimulate and maintain an interesting polyphonic blend. Random walk step sizes and mirrors can be set so that a voice is quite stable or varies widely of its own accord. Each extreme has its own challenges for performance. More stable settings involve less risk of drift into inappropriate regions of sonic space, but require more tweaking to maintain interest. Unstable settings can almost create a piece by themselves, not unlike the way Xenakis used his *Gendyn* program, however, control over the outcome is transferred to the algorithm to a large extent and performance consistency can vary widely. Composers using other DSS implementations have the luxury of throwing away unsuccessful outcomes. For example, Xenakis' first DSS piece, *Gendy3*, was named after its full file name GENDY301, so we can assume that it took Xenakis more than three hundred attempts to arrive at a satisfactory outcome. The intent of our IDSS application is not a "sound-producing automaton creating complete composition 'out of nothing', purely through the structure-generating power of probabilities" [2:31]. but to enable a partnership between the processing algorithms and the live performer. Within this partnership, the processing algorithms determine the detail of the sonic microstructure while the performer is chiefly responsible for the macrostructure.

The live performance risk with generative music systems can be acute at times, and stochastic processes are often quite an imprecise tool. To help alleviate this risk the IDSS program can save and recall settings allowing the performer to act like DJ in scheduling and tweaking prepared sections of work. Unlike a pre-sequenced stream of MIDI information, or banks of sampled loops the generative nature of the real-time system maintains a degree of uncertainty even in this scenario. A skilled performer needs to not only understand every available parameter's operation and interrelation but also the degree of unpredictability likely to result from different relationships between settings.

Like other computer and electronic music performances the juggling of many sound sources is a critical skill. The software has been designed to operate with only the standard mouse and keyboard controllers. The automated controllers in the IDSS program assist the execution of several parallel tasks despite the often awkward one-mouse controller situation. The IDSS program supports assignable MIDI controllers, a feature that greatly enhances real time control and expressive capabilities by enabling the use of physical hardware controllers (knobs and/or slider boxes). With only two hands, such devices rarely permit simultaneous control of more than a few parameters so a combination of qwerty keyboard, mouse and MIDI control hardware is normally employed.

7. Conclusion

The application of a synthesis process in an interactive software environment requires a clear understanding of the operation, behaviour and potential of the process in order to adequately reveal the useful attributes to the musician and hide irrelevant details. We have outlined our approach to this task in relation to dynamic stochastic synthesis and its implementation in the Interactive Dynamic Stochastic Synthesizer. The design and development was informed by an understanding of the historical context, a familiarity with the technical aspects of the process, experimentation with DSS leading to analysis of tendencies in the synthesis process and methods of harnessing them for efficient use in live performance. However, as with all such efforts, our work is ongoing and further performances and exploration will suggest enhancements. Some already planned include stochastic macrostructure controls, meta controller assignment mapped to multiple parameters and an increased choice of probability distributions and interpolation functions.

References

- [1] Bokesoy, S. and G. Pape (2003). "Stochos: Software for Real-Time Synthesis of Stochastic Music." *Computer Music Journal* **27**(3): 33-43.
- [2] Hoffman, P. (2000). "The New GENDYN Program." *The Computer Music Journal* **24**(2): 31-38.
- [3] Matossian, N. (1986). *Xenakis*. London, Kahn & Averill.
- [4] Serra, M.-H. (1992). "Stochastic Composition and Stochastic Timbre: Gendy3 by Iannis Xenakis." *Perspectives of New Music* **31**(1): 236-257.
- [5] Sorensen, A. and A. R. Brown (2000). *Introducing jMusic*. The Australasian Computer Music Conference, Brisbane, ACMA.
- [6] Xenakis, I. (1971). *Formalized Music: Thought and Mathematics in Composition*. Bloomington, Indiana University press.
- [7] Xenakis, I. (1991). *Formalized Music*. New York, Pendragon Press.