
Proceedings, 4th Chaotic Modeling and Simulation International Conference
31 May – 3 June 2011, Agios Nikolaos, Crete Greece

Lindenmeyer Systems and the Harmony of Fractals

Pedro Pestana

CEAUL — Centro de Estatística e Aplicações da Universidade de Lisboa
Portuguese Catholic University – School of the Arts, CITAR, Porto, and Lusíada University, Lisboa, Portugal
(E-mail: pedro.duarte.pestana@gmail.com)

Abstract. An interactive musical application is developed for realtime improvisation with a machine based on Lindenmeyer-systems. This has been used on an installation whose goal is to draw the attention of unexperienced users to the wealth of realtime applications in computer music. Issues on human computer interaction and improvisation grammars had to be dealt with, as well as probabilistic strategies for musical variation. The choice of L-systems as a basis for machine composition is a consequence of their ability to create results that easily have aesthetic appeal, both in the realms of sound and image.

Keywords: human-computer interaction, L-systems, fractals in algorithmic music composition, interactive composition, improvisation, computer music.

1 Introduction

Musical variation, and composition rules defined by Schönberg, exploit to a certain extent the self-similarity of fractals, and Lindenmeyer (cf. Rozenberg[11]) created algorithms (in biological research) that can be exploited fully using iteration in algorithmic music composition. But can fractals create harmony of sound and *cantabile* music as well as they create beauty for the eyes in graphical arts?

We present examples of an interactive algorithmic music composition system exploiting Lindenmeyer's technique, generating some forms of minimalist music based on user input, and further developments using the interaction of probability models, fractals and chaos.

Lindenmayer systems, or L-systems, are parallel formal grammars introduced in 1968 by the botanist Aristid Lindenmayer[3] as "a theoretical framework for studying the development of simple multicellular organisms" (Prusinkiewicz and Lindenmayer[10]). As such, in essence an L-system is a rule-based generative system that, drawing from a finite set of symbols, applies substitution schemes starting with an initial subset, called in Prusinkiewicz[9] an axiom. In Chomsky grammars, substitutions are made in series, with each pass focusing exclusively on a sole symbol, while L-systems are parallel, in the sense that all symbols are replaced within each iteration.

Extending the initial application of L-systems, developments were made in order to generate realistic computer images of plants and trees (Smith[15]), fractal curves (Prusinkiewicz[8]), and musical scores (Prusinkiewicz[9]).

Given words with a fair amount of complexity, an L-system will exhibit a noticeable degree of self-similarity over iterations, which makes its results memorable and pleasing when interpreted as musical height or visual branching, in the sense that there is an equilibrium of expected and unexpected developments. In other words, as Schröder[12], p. 109, boldly presents the key ideas of Birkhoff's *theory of aesthetic value*, the results are pleasing and interesting since they are neither too regular and predictable like a boring brown noise with a frequency dependence f^{-2}, nor a pack of too many surprises like an unpredictable white noise with a frequency dependence f^{-0}.

The remainder of this paper is organized as follows. In Section 2 we describe implementations of L-systems for the automatic generation of music. In Section 3 the focus is on the analysis of musical parameters from user input, such as pitch velocity and duration, and their mapping to L-systems. Section 4 deals with possible extensions of this work to polyphonic input and output, and Section 5 deals with the specific implementation of this project. Finally, in Section 6, we briefly discuss further issues and possible developments.

2 Construction of an *L*-system

L-systems come in several categories: context-free (OL-systems) or context-sensitive (IL-systems); deterministic or non-deterministic; propagative or non-propagative, and so on. The interested reader is referred to Manousakis[4] and to Rozenberg[11] for an extensive review of different types of L-systems. The present work uses non-deterministic OL-systems, as described below.

Let $\mathcal{A}$ denote an alphabet of letters ℓ, $\mathcal{V}$ the vocabulary, i.e. the set of words $w = \ell_1\ell_2\cdots\ell_n$ (strings of letters from this alphabet); $\emptyset$, the empty set, is considered a word.

A production $P:\ \mathcal{A} \longrightarrow \mathcal{V}$ is described by random variables associated with each $\ell \in \mathcal{A}$, i.e.

$$\ell \quad \overset{P}{\mapsto} \quad P(\ell) = X_\ell = \begin{cases} w_k \\ \\ p_k = \mathrm{P}[X_\ell = w_k] \end{cases},$$

and *j-letter* $\mathcal{L}_j:\ \mathcal{V} \longrightarrow \mathcal{A}$ selects the j-letter of any given word,

$$w = \ell_1\ell_2\cdots\ell_k \quad \overset{\mathcal{L}_j}{\mapsto} \quad \mathcal{L}_j(w) = \ell_j.$$

We assume that if $\ell_i \neq \ell_j$, then X_{ℓ_i} and X_{ℓ_j} are independent. If the actual result of $P(\ell)$ is w, we write $\ell \mapsto w$, and say that ℓ is the predecessor of w, or alternatively that w is the successor of ℓ.

If $w = \ell_1\ell_2\cdots\ell_k$, $\mathbf{P}(w) = P(\mathcal{L}_1(w))P(\mathcal{L}_2(w))\cdots P(\mathcal{L}_k(w))$. A *production* of size k with root w_0, $\mathcal{P}_{w_0,k}$ is

$$\mathcal{P}_{w_0,k}(\cdot) = \mathbf{P}(\mathbf{P}(\mathbf{P}(\cdots\mathbf{P}(\cdot)\cdots))),$$

and $\mathcal{P}_{w_0}(\cdot) = \bigcup_{k\in\mathrm{N}} \mathcal{P}_{w_0,k}(\cdot)$.

An OL-system is an ordered triplet $G = \{\mathcal{A}, w_0, P_{w_0}\}$, with $w_0 \in \mathcal{A}$ the starting point for the successive iterations, and P_{w_0} is a production of finite size with root w_0. In an OL-system the predecessor is a one-letter word whereas the successor can be of arbitrary length (it can even be an empty word). In a non-deterministic system, different successor words may occur according to a probabilistic distribution. Hence the production may be described in terms of a branching process, whose many possible trajectories are tied to the possibilities that actually do occur.

A very easy construction of a musical grammar (McCormack[5]) could be built by taking an alphabet $\mathcal{A} = \{C, D, E, F, G, A, B\}$ corresponding to the notes of a C major scale (or an even larger musical scale alphabet), an axiom that would be given by user input and a set of productions that may be arbitrary or may follow rules from common practice of harmony. Alternative constructions have been given by Soddell and Soddell[16], who map branching angles to changes in pitch, Prusinkiewicz[9] where a deterministic OL-system is used to generate a graphical turtle interpretation of the production, and then the resulting curve is traversed and the height of each line segment is interpreted as pitch among others. Most of the studied constructions have seamlessly resulted in pleasing musical results and in our approach we opted for the former, more literal one.

As an example, consider the alphabet $\{C, D, E_b, F, G, A_b, B\}$, the root $w_0 = DE_bCB$ (the celebrated Shostakovich signature, used in many of his mature works), and the stochastic transition matrix — a sparse matrix, so that the equilibrium of expected and unexpected generates aesthetic value — describing the probabilities governing the productions P:

	A_b	A_bE_b	A_bG	B	C	CFD	CFG	DC	E_b	F	G	GA_b	GF
C	0	0	0	0.7	0	0	0	0	0	0	0.2	0	0.1
D	0	0	0.8	0	0	0	0	0	0	0	0.2	0	0
E_b	0	0	0	0.8	0.2	0	0	0	0	0	0	0	0
F	0	0	0	0	0.2	0	0.7	0	0	0	0	0.1	0
G	0	0.2	0	0	0.7	0.1	0	0	0	0	0	0	0
A_b	0	0	0	0	0.2	0	0	0.8	0	0	0	0	0
B	0.2	0	0	0	0	0	0	0	0.7	0.1	0	0	0

Assume we get the sequence

$w_0 = DE_bCB$	1
$w_1 = A_bGBGE_b$	0.0896
$w_2 = DCCFDE_bA_bE_bB$	0.00896
$w_3 = A_bGBBCFGA_bGBDCBE_b$	0.078675968
$w_4 = DCCE_bE_bBCFGCDCCE_bGBE_bB$	0.004934557

with the probabilities indicated in the right column. So, in this example, with probability 3.11678×10^{-7} we get $\mathcal{P}_{DE_bCB,4} = DCCE_bE_bBCFGCDCCE_bGBE_bB$.

Observe that the rich theory of Markov chains, and concepts such as communicating evens, cyclicity, stationarity, can therefore be imported to analyse productions.

3 Analyzing user input

In the proposed interaction model, a user inputs a musical phrase which serves as the root (axiom), and given a significant pause the system reacts branching into the successive iterations given by the production set. At any point the user could feel inspired by the results and step in with a new musical phrase as a new root, stopping the automatic production, from which the computer draws new material according to the same set of productions or a revised version of it. The focus of this work is on the user-satisfaction with the musical results, and as such it was decided that the interface should not be a tried and tested one such as the music keyboard. This is also helpful in that it allows us to use a very robust MIDI communication, leading to a clear interpretation of pitch, velocity and duration.

The possibility of having the computer analyzing the intention of the musical input and generating different productions would be the first step towards a musical and engaging result. A first approach should consist on scale detection, and Chai and Vercoe's strategy based on hidden Markov models (see Chai and Vercoe[1]) was used in order to extrapolate the global outline of the production set, cf. also Noland and Sandler[6]. The set itself was constructed in strict adherences to classic common practice as described by authors such as Piston[7], as it was deemed that the musical results should be satisfying to a wide non-expert "random" audience.

An additional concern has been how to map user-inputted velocity and duration into the productions of the model. Three approaches have been considered and tested for note duration:

- Having an additional algorithm for tempo detection and building a parallel fixed set of productions for note duration.
- Keeping the duration that was given by user-input across successive generations of productions.
- Cycling through the set of user-inputted durations.

The first approach has been abandoned. Without further constraints forcing the user to adhere to a tempo it would have been unmusical to let the computer-generated productions have a strictly quantized feel as a result of the original input being free from adequate rules. The second approach has also been discarded, since after a few generations a pattern of unnatural repetitiveness would begin to emerge, creating unmusical productions. The

third approach has been, surprisingly, musically rewarding, as it potentiated the natural feel that resulted from the self-similarity of successive iterations. Consequently, it has been our choice to govern this parameter. The last member of the set needs to be automatically generated, as there is no way to infer the duration of the user's last note. For this we simply repeat the previous duration value.

It was also not clear from the start which solution would be better for velocity mapping and again different paths were evaluated:

- Quantizing the velocity to a set value given by the average value of the user input.
- Giving a fixed velocity to each of the words in the vocabulary, again averaging the user-inputted value for that word.
- Keeping the velocity that was given by user-input across successive generations of productions
- Cycling through the set of user-inputted velocities.

In fact, any of those solutions proved to be too mechanical, and we had to create a new rule that would allow for musical variety. We choose to create a set of user-inputted velocities, and to discard at random one value from the set in each iteration. The result is immediately more natural, since now there is a much longer period before any pattern of duration-velocity pairs can repeat.

4 Extending the system towards polyphony

The above discussion on analysis is straightforward for monophonic input and output, but the possibility of using multiple voices poses a string of new issues that are not so easily solvable. On the input side, making the distinction between harmonic movement and melodic movement is fraught with ambiguity and the allocation of each melodic movement to a unique voice is also a tremendous challenge. On the output side, decisions had to be made as to adherence to melodic rules and voice independence. Each problem has to be addressed in turn.

The distinction between harmonic and melodic movement cannot depend on simultaneity, when human input is considered. Users never perform with infinitesimal precision and we must therefore create time windows within which two events can be considered simultaneous. A sensible time window would be in the range of 30-50 ms, according to the Haas principle or precedence effect, that states that the human listener integrates all sound events that occur within that time frame. This is a very bold statement from a musical perspective as musical interpretation and style might at times dictate that events that are technically simultaneous should be performed with enough separation between them to clearly exceed the above-mentioned interval. One well-known and consistent example is the Flamenco's *rasgueado*, where the

harmonic intervals are always performed as a very quick succession. We must therefore agree on an extended interval based not on a Haas-inspired pursuit of simultaneity, but on the opposite idea of what would not be a melodic interval. With this in mind we can safely say that is untypical for a performer to go faster than a eighth-note on a 120 bpm tempo which would point us to a 63 ms window. This is of course ambiguous and might be prone to error on fast ornamentations.

Correctly distributing events between voices in a setting where different voices might have different musical durations and pauses is a subject that has not yet been successfully solved. Indeed, it is not clear whether the rules described in the previous section would work with multiple axioms as a starting point. Due to those yet unsolved questions, for the time being, the input side of polyphony has been dropped and the user would only be allowed to play monophonically.

It was however interesting from a musical standpoint that the output could be done polyphonically with the aid of an automatic accompaniment. A simplification of the model proposed by Schwarz *et al.*[13], based on HMM, has been used in order to extend the system, using a low and sparsely-generated voice.

5 Implementation

The system was implemented in Max/MSP, making use of the in-build Jitter object jit.linden. A first patcher parses the input and does the scale analysis, and feeds the finished list to the patcher responsible for the productions (shown in Fig. 1). The productions are fed to a third patcher that converts them to MIDI and sends them as UDP packages to SuperCollider, where a simple implementation of a quasi-sinusoidal synth that resembles a vibraphone is used as a sound module.

An example we fed the system with Shostakovich's aforementioned signature $DSCH$ (used musically as D, E_b, C, B) played as a pair of quavers followed by a pair of semi-quavers of equal velocity. The input patcher interprets the motif as played in C harmonic minor and constructs the set of productions already presented as a sparse stochastic transition matrix in Section 2, presented below in a more readable condensed form for those not wanting to dive in stochastic processes theory:

$$P = \left\{ \begin{array}{lll} P_{11} : C \xrightarrow{70\%} B & P_{12} : C \xrightarrow{20\%} G & P_{13} : C \xrightarrow{10\%} GF \\ P_{21} : D \xrightarrow{80\%} G & P_{22} : D \xrightarrow{20\%} A_b G & \\ P_{31} : E_b \xrightarrow{80\%} B & P_{32} : E_b \xrightarrow{20\%} C & \\ P_{41} : F \xrightarrow{70\%} CFG & P_{42} : F \xrightarrow{20\%} C & P_{43} : F \xrightarrow{10\%} GA_b \\ P_{51} : G \xrightarrow{70\%} C & P_{52} : G \xrightarrow{20\%} A_b E_b & P_{53} : G \xrightarrow{10\%} CFD \\ P_{61} : A_b \xrightarrow{80\%} DC & P_{62} : A_b \xrightarrow{20\%} C & \\ P_{71} : B \xrightarrow{70\%} E_b & P_{72} : B \xrightarrow{20\%} A_b & P_{73} : B \xrightarrow{10\%} F \end{array} \right\}.$$

The result can be heard at http://www.stereosonic.org/lindenmayer.

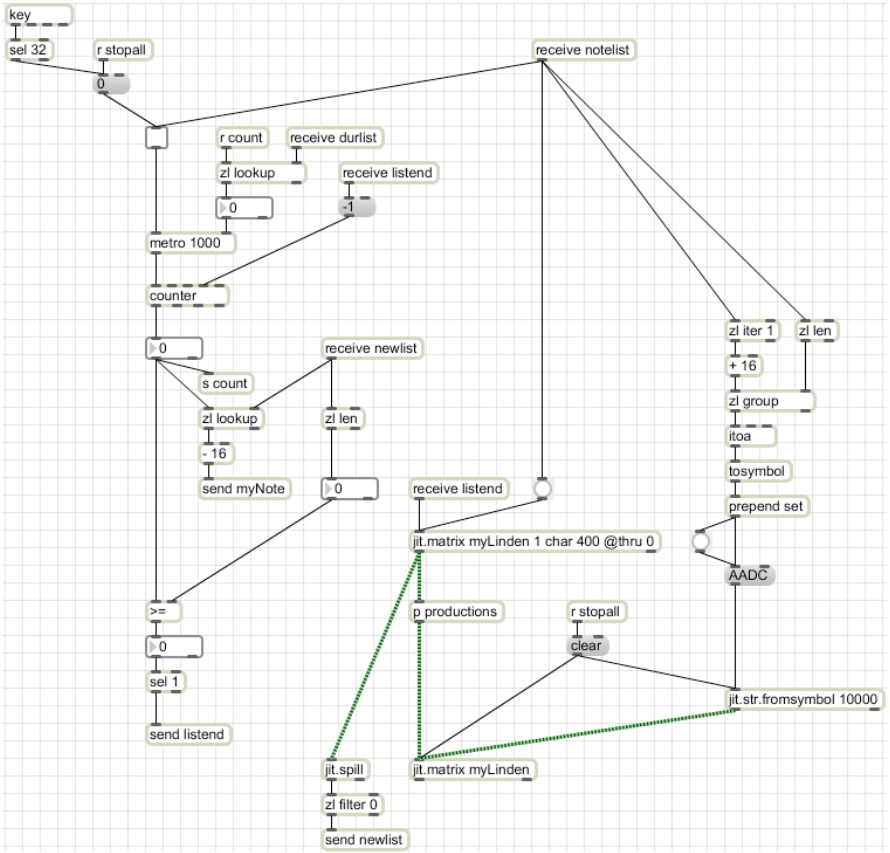

Fig. 1. Max/MSP main patcher

6 Concluding remarks

Many alternative ways do exist of music composition tied to fractals, cf. Johnson[2] and Skiadas[14], for instance. OL-systems as used in our examples generate appealing musical productions as far as letters map onto words of small size. Otherwise, the system must be interrupted by the user, since a rather small number of iterations generates a musical output that is too clumsy. The organisation of natural languages, and namely of the mating songs of birds and insects, seems to incorporate a strategy of long range dependence axed on a sequence of modulated shortcut Markov-type memories. Hence, for more elaborated vocabularies and mappings, it would be sensible to use only the r last letters from the (k-1)-th iteration to map onto the k-th iteration, instead of using all the letters as described for OL-systems. This is easily implemented using an *endletters* application $\mathcal{E}_r : \mathcal{V} \longrightarrow \mathcal{A}$ selecting

the r-endletters of any given word,

$$w = \ell_1\ell_2\cdots\ell_k \quad \overset{\mathcal{E}_r}{\mapsto} \quad \mathcal{E}_r(w) = \ell_{k-r+1}\ell_{k-r+2}\cdots\ell_{k-1}\ell_k,$$

so that the memory of the initial $k-r$ letters is erased and the musical composition will flow more naturally.

Research partially supported by FCT/OE. The author is grateful to Professors Álvaro Barbosa (UCP) and Joshua D. Reiss (QMUL) for generous guidance, stimulating discussions and encouragement.

References

1. W. Chai and B. Vercoe. Detection of key change in classical piano music. *Proceedings of the 6th International Conference on Music Information Retrieval*, 2006. London.
2. R. S. Johnson. Composing with Fractals. In J. Fauvel, R. Flood and R. Wilson, eds., *Music and Mathematics*, 2006. Oxford University Press.
3. A. Lindenmayer. Mathematical models for cellular interaction in development, *Journal of Theoretical Biology*, 18: 280–315, 1968.
4. S. Manousakis. *Musical L-Systems*. M.Sc. Thesis in Sonology, The Royal Conservatory, 2006. The Hague.
5. J. McCormack. Grammar-Based Music Composition. In Stocker et al., eds. *Complex Systems 96: from Local Interactions to Global Phenomena*, 1996. IOS Press.
6. K. Noland and M. Sandler. Key Estimation Using a Hidden Markov Model. Inn *International Society for Music Information Retrieval*, 121–126, Victoria, 2006. Canada.
7. W. Piston. *Harmony*. New York, 1941. W.W. Norton & Company.
8. P. Prusinkiewicz. Graphical applications of L-systems. *Proceedings of Graphics Interface'86*, 247-253, 1986a.
9. P. Prusinkiewicz. Score Generation with L-Systems. *Proc. Intl. Computer Music Conf '86*, 455-457, 1986b.
10. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*, 1990. Springer.
11. G. Rozenberg. *Lindenmeyer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, 1992. Springer Verlag.
12. M. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. New York, 2009. Dover.
13. D. Schwarz, N. Orio and N. Schnell. *Robust Polyphonic MIDI Score Following with Hidden Markov Models*, 2004. ICMC.
14. C.H. Skiadas. Exploring and simulating chaotic advection: A difference equations approach. In C. H. Skiadas, ed., *Recent Advances in Stochastic Modeling and Data Analysis*, pages 287–294, Singapore, 2007. World Scientific.
15. A.R. Smith. Plants, fractals, and formal languages. *Computer Graphics*, 18(3): 1-10, 1984.
16. F. Soddell and J. Soddell. Microbes and Music. In *PRICAI 2000 Topics in Artificial Inteligence*, pages 767–777, 2000. Springer.