Volume: 3 Issues: 8 [June, 2018] pp. 45-50]
Journal of Information System and Technology Management
eISSN: 0128-6666
Journal website: www.jistm.com

A DATAFLOW-BASED METHODOLOGY FOR SOUND RENDERING USING L-SYSTEMS AND VISUAL LANGUAGE FRAMEWORK

Chen Kim Lim[1]
Kian Lam Tan[2]
Hazwanni Yusran[3]

[1,2,3]Faculty of Art, Computing & Creative Industry, Sultan Idris Education University, 35900 Tanjong Malim, Perak, Malaysia

Accepted date: 3 May 2018 **Published date**: 2 July 2018

To cite this document:

Lim, C. K., Tan, K. L., & Yusran, H. (2018). A Dataflow-Based Methodology for Sound Rendering Using L-Systems and Visual Language Framework. *Journal of Information System and Technology Management*, *3*(8), 45–50.

Abstract: *Lately, computer-music learners have been attracted to the applications of L-Systems in rendering harmonious music. However, the current available systems such as LMUSe, could only render harmonious sound if the L-Systems grammars are known by the user. Besides, in producing harmonious sound in LMUSe, stochastic and context-sensitive L-Systems are combined in a single step and therefore, it is hard for the computer-music learners to apply L-Systems. In this paper, we propose a 6-phases simple and flexible visual language programming for sound rendering so that the harmonious sound can easily be generated compared to LMUSe. Furthermore, with this framework, the computer-music and L-Systems learners do not need to have prior knowledge in both areas. Moreover, the stochastic and context-sensitive grammars are applied step-by-step as part of the methodology through the data-flow process to enhance the understanding of the process involved. Based on the subjective evaluation, it was found that the users could easily understand, appreciate and learn the process of applying stochastic and context-sensitive grammars in producing harmonious sound.*

Keywords: *Visual Language, Harmonious Sound, L-Systems, LMUSe*

Introduction

There exists a comprehensive visual language framework based on icons for L-System sound rendering by [12, 13] that improves an expressive music environment and the system was inspired from an earlier system by [25] which is a visual language framework for modelling the plant using L-System. The system [12, 13] generates visual language grammar models for L-System music rendering that are to be imparted into a system for music rendering for a more effectual enhancement of plant model of the L-System to music rendering. Other existing systems such as LMUSe [24] can only be used to render harmonious sound if the user knows in advanced the L-Systems grammars that need to be keyed in. In this paper, we propose to use visual language programming to render harmonious sound by also applying stochastic and context-sensitive grammars through a dataflow-based methodology. The process of rendering of the harmonious musical sound is then enhanced significantly.

Previous and Related Work

There are works done concentrating in modeling plant digitally. One of the recent work by [5] managed to reproduce the key features of grasstree (Xanthorrhoea) bark patterns. However, the grasstree was not modelled based on the L-system algorithm. [1] proposed an application named L-Py to teach plant modeling in the classroom. In order to quickly generate and model plant in an easier and more interactive way, [14] proposed TreeSketch which uses motions of brushes to instantaneously generate tree branches on a tablet. This work could be further extended to interpret tree branches as music.

The L-systems has started off with the idea of making improvement of best mimic the development of biological plant in reality. This area has been further extended from plant modelling to music rendering. There are a few examples showing that the L-system is capable of traversing plant to music and vice versa as depicted in Figure 1. Within the L-system, context-sensitive and stochastic are the two most outstanding musical rendering method which helps in generating more “pleasing” music melodically.

[16] proposed an extensive framework for designing L-systems that generate musical structures from the macro-level to the sample level. Besides, [6] also proposed to create and explore potential taxonomies using algorithmic string-substitution systems to generate music. However, these works are more beneficial to users who are already in musical industries such as singers, composers and musicians. Lejaren Hiller (1924–1994) is widely recognized as the first composer to have applied computer programs to algorithmic composition with Fibonacci and Markov chains [7]. The algorithmic composition consists of the deterministic (fixed or cellular automata) and indeterministic (stochastic, random or Markov model) procedures.

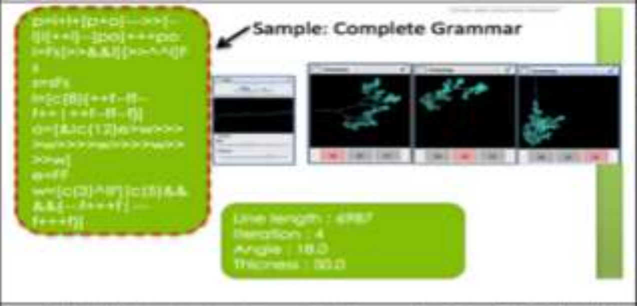
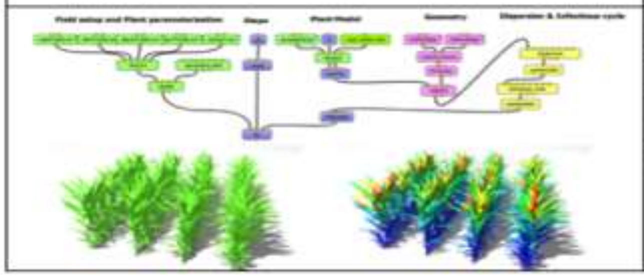
Images	Explanation
	Examples of trees created with TreeSketch. Arrows indicate the motions of the brush that determined the corresponding tree forms. The trees were generated instantaneously while brushing (Longay *et al.*, 2012).
	Plants are drew and music is rendered through MIDI when a set of complete grammar named Algae Beauty is keyed into LMUSe (Sharp, 2009)
	OpenAlea Visual Modelling Environment provides a graphical user interface (GUI) which is VisuAlea in order to make it possible to access easily the different components and functionalities of the system (Pradal *et al.*, 2008)

Figure 1. The Improvement of Work Done from Plant Modelling to Music Rendering

[11] has further extended L-System with Finite L-systems (FL-systems) that uses Genetic Algortihms (GA) to expand the adaptability to certain rhythmic tasks such as density, pauses, self similarity, symmetry and syncopation. In order to handle temporal aspects of music, [22] proposed a Probabilistic Temporal Graph Grammars (PTGG) which is able to handle harmonic and metrical structures. Then, repetitive visual patterns are developed with function composition, recursion and lazily evaluated lists that are inspired by the concept of using traditional textile and wallpaper designs [10].

Our work is inspired by the previous work of [25] namely Visual Language Plant Modelling system (VLPM) which is a visual language framework for plant modelling using L-Systems. Besides, [12, 13] has designed and developed a visual language framework for music rendering using L-System called Visual Language Music Rendering (VLMR). [12, 13] have also further improved the system. As for LMUSe [24], it is a complete and sophisticated system that is compatible with L-System grammars specifically for music rendering. However, music cannot be rendered if one does not know the L-System grammar that is supposed to instruct the direction, movement, operations of the stack, increment and decrement of musical pitch, and tempo. In LMUSe, as proposed by [28], it is possible to produce more 'pleasing' or harmonious musical sounds by using stochastic and context-sensitive grammars. However, it requires the users to key in the L-System grammar and to have extensive knowledge in L-System. In this paper, we propose a visual language approach used by the users who have little or no knowledge on L-Systems. By chance, L-Hasr integrates parts of VLPM with pieces from LMUSe since VLPM provides iconic representations which match the musical notes with the attributes of plant models [25].

Methodology

In the visual programming domain, the emphasis of the research depends on the application of visual formalism. From the view of programming it is regarded to be more effective than textual formalism.

The overall implementation approach is designed based on the data flow programming model as shown in Figure 2.

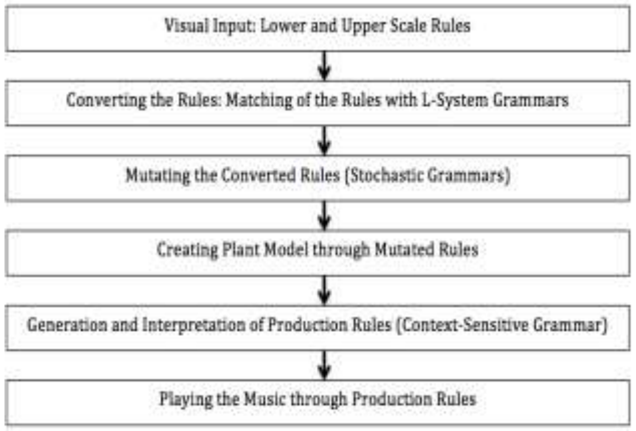

Visual Input: Lower and Upper Scale Rules
Converting the Rules: Matching of the Rules with L-System Grammars
Mutating the Converted Rules (Stochastic Grammars)
Creating Plant Model through Mutated Rules
Generation and Interpretation of Production Rules (Context-Sensitive Grammar)
Playing the Music through Production Rules

Figure 2. The Overall Method: Data Flow in L-Hasr

In the method, firstly, the user needs to input the rules (lower and upper scales) using the visual language framework by selecting, and dragging and dropping the icons on the window for defining the rules. The selected icons are read by the L-System editor that runs behind the interface. The editor then converts the rules by matching them with the L-system grammars and stores the converted rules in an array. As the rules are too short to generate harmonious musical sound, the rules can be mutated using the stochastic L-System grammar. Rewriting the converted rules (mutation) then takes place (if required) so that a longer piece of rules is generated [15, 19, 21]. The mutated rules are also able to create a plant model. The L-system editor is then pass the rules to the string generator. Then, the string generator which follows the context sensitive L-system grammar, output (interprets) the production rules based on the mutated rules [23]. Finally, harmonious musical sound can be played with the aid of MIDI library [2, 3]. The input and output strings are summarized as in Table 1.

Implementation

Based on L-System, the plant modelling can then be interpreted for music rendering [28] through the interpretation of the L-Systems grammars. The visual language framework is implemented with musical icons that can be pulled and placed flexibly and is written in Java [9] using Netbeans IDE in order to integrate with both the context-sensitive and stochastic L-Systems grammars that were partly adopted from LMUSe system.

Table 1. A Sample of Input and Output Strings

Input String	Output String
Original Rules	X=f[+X][-X]fx
Converted Rules (Stochastic Mutation)	X=[X[X]]f\-+X/\-+[X]-[X]X/fX
Production Rules (Context-Sensitive Mutation)	=X/\-+[X]-[X]X/fX/f[X[X]]f\-+X/\-+[X]-[X]X/fX
Angle	25-degree
Left Branch's Size	Small
Right Branch's Size	Large

The L-Hasr system consists of the Rules (Upper Scale and Lower Scale), Music Toolbox (Key, Transpose Up, Transpose Down, Play, Push and Pop), Tree's Attribute (Angle, Left Branch's Size and Right Branch's Size), Converted Rules and Production Rules. The rest of the method is implemented under Functions Menu of the interface namely ConvertRule (Converting the Rules: Matching of the rules with L-System Grammars), Mutate (Mutating the Converted Rules (Stochastic Grammars)), Make (Creating Plant Model through Mutated Rules), Interpret (Generation and Interpretation of Production Rules (Context Sensitive Grammar)) and Play (Playing the Music through the Production Rules). The music can be easily rendered by clicking 'Play' button on L-Hasr.

The array of the lower and upper scale is meant to store the icons which can be dragged and dropped from the Music Toolbox to the Rules section. By default, each scale can store up to fifteen icons in the array which can be extended if necessary. The inputs to the array are treated as a string and to be passed for string translation later in the next stage of the data flow model. The purpose of passing the array is to match the rules with the predefined set of grammars of L-System. Each and every icon on the visual framework represents an attribute of the L-System Grammar. For example, for the function name 'Key', the string 'Key' is passed for translation of the lower and upper scale strings that correspond with the rule symbol of 'X' as the L-System attribute of an 'axiom'. On L-System Harmonious Sound Rendering (L-Hasr), the button 'ConvertRule' is clicked. Rewriting the converted rules undergoes five types of mutation processes. Each of the mutation is based on one-point swap mutation randomly. The mutation rules consist of shuffling of icon keys, inserting a randomly picked rule and replacing the new rules with the rules on the right octave, exchanging fragments among the existing rules on the right side randomly, picking a random rule in order to reverse the directions, and switching the pop command to the push command randomly. These mutations of rule allow various pieces of L-System grammar to produce different musical sound. On L-Hasr, the button 'Mutate' is clicked to rewrite the previous converted rules.

In order to model the plant, the tree's attribute needs to be filled entirely. The attributes consist of the angle, and the size of the left and the right branch of the tree. The sizes are small, medium and large. The user is encouraged to select a small size in order to reduce the computational time and increase the efficiency of plant modelling. If the user does not key in the angle, the plant is not modelled and vice versa. In order to make the music sound more harmonious, three iterations are fixed. More iteration can be defined if required. However, the rendering of the musical sound is slow. On L-Hasr, the button 'Make' is to be clicked to generate production rules. The interpretation of the output string which is the production rules can be done by clicking the button 'Interpret' on L-Hasr. From the perspective of the backend engine, the production rules have to check with some predefined cases to execute different actions. An example pseudocode of how production rules are interpreted is as follows:

```
Begin
    1.  Initialize the counter variable integer I to zero
    2.  While 'i' is less than the length of the rules
        3.  Read rules character and check
        4.  If character is "+" and angle is more
              than 360$^0$, turn the cursor left
        5.  Else 360-angle and turn left
        6.  If character is "-" and angle is more
              than 360$^0$, turn the cursor right
        7.  Else 360-angle and turn right
        8.  If character is 'f' move forward one
              line length and draw make a branch
        9.  If character is "!", swap directions of
              all the following cursor turns
        10. If character is "@", scale the line length
              up or down
        11. If character is "[", store the current
              graphics state
```

```
            12.    If character is "]", recall the previous
                   graphics state
            13.    Else print out line "Missing ] in line: "
                   + rule
        14. End while
        15. Interpret the rules for variables
        16. Increase counter variable by 1
    End
```

Music Rendering is the final stage of the Implementation stage from the L-Hasr Data Model. The music can be easily rendered by clicking the 'Play' button on L-Hasr as shown in Figure 3.

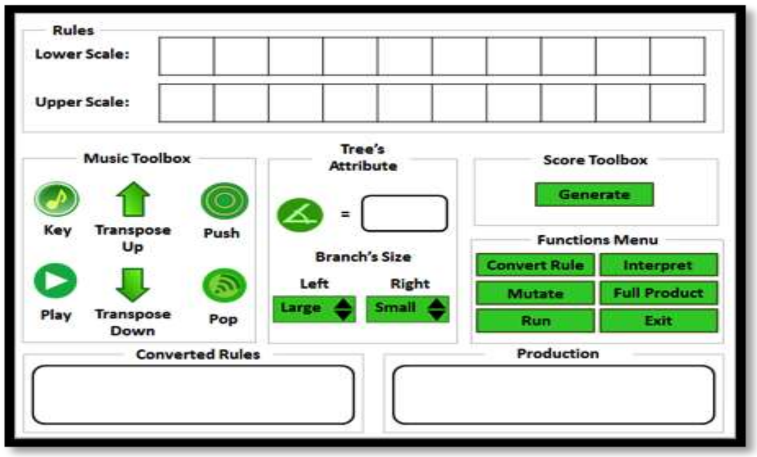


Figure 3. A Snapshot on the Music Rendering Process

Result and Discussion

Open feedbacks are carried out among a hundred respondents who do not have prior knowledge in L-Systems and music. Some of the technical feedbacks were compiled. The respondents are first exposed to other L-System related application LMUSe, OpenAlea, TreeSketch, LScore, OpenMusic, L-Py, L-studio and L-Hasr before the feedbacks are gathered. LMUSe could model the plant and render the music accurately through MIDI only when a complete set of grammar such as Algae Beauty is keyed into it. OpenAlea is a user-friendly modeler with complex components and functionalities in the system based on FSPM but it is more suitable for advanced deployments. TreeSketch could generate plants when the motion of the brush is detected instantaneously. Besides, the sketching of the plants is efficiently and accurately generated but no music could be rendered. LScore is opposite of TreeSketch where it supports deterministic context-sensitive and stochastic parametric with crossover and mutation operators to generate the MIDI files from the description of L-Systems but no plant is modelled. OpenMusic is more for specialised level where it has a set of classes and libraries for music composition through visual programming language based on Lisp. However, OpenMusic concentrates on music and no plant is generated too. L-Py is an L-System simulation framework for modelling plant architecture based on a dynamic language and thus only expert in computer science and botanist would appreciate this framework. L-Studio has an integrated editor to edit almost all types of music files and no plant modelling is involved. Finally, L-Hasr is a simple visual programming framework where music and plant both can be generated by assembling and connecting icons that represent their functions and data structures. LHasr is the most suitable application for non-expert in both music and L-Systems.

Conclusion

Compared to the previous research on L-Systems music rendering, this research has provided through a much simpler and more flexible visual language framework a variety of L-Systems grammars by applying separately step-by-step stochastic and context-sensitive L-Systems. This hybrid model allows rendering of a more harmonious musical sound for music rendering. It was also found that the L-Hasr is easy to use in producing harmonious musical sounds. Thus, the visual language framework used in L-Hasr is well-suited to allow non-experts and experts to understand, learn and use L-systems in particular for L-system music rendering and producing more harmonious music (Menzies, 2000). In order to further harmonise the sound of the music, harmony search algorithm in evolutionary computing can be examined to generate a more advanced L-system syntax and semantic and they should also be realized within a visual language programming to retain the simplicity and flexibility of the framework.

Acknowledgement

This work presented is supported by the KPT FRGS Grant (No. 2015-0163-1090-02). The authors would like to acknowledge the many helpful suggestions of the anonymous reviewers. Thanks to Then Siew Tong for new the GUI.

References

[1] Boudon, F., Pradal, C., Cokelaer, T., Prusinkiewicz, P., & Godin, C. 2012. L-Py: an L-system simulation framework for modelling plant architecture development based on a dynamic language. *Journal of Frontiers in Plant Science*. 3(76).

[2] Bresson, J., & Agon, C. 2006. Sound writing and representation in a visual programming framework. *In DMNR-06 Doctoral Research Conference, Digital Music Research Network*.

[3] Bresson, J. 2004. OpenMusic MIDI Documentation. *Ircam Software Documentation*.

[4] Bruno, F., Jose, C. L., & Marcio, C. P. 2009. L-Systems, scores, and evolutionary techniques. *In SMC 2009: Proceedings of 6th Sound and Music Computing Conference*. 113 - 118.

[5] Dale, H., Runions, A., Hobill, D., & Prusinkiewicz, P. 2014. Modelling biometrics of bark patterning in grasstrees. *Journal of Annals of Botany*. 114(4), 629 - 641.

[6] DuBois, R. L. (2003). Applications of generative string-substitution systems in computer music. *Thesis of Doctor of Musical Arts in the Graduate School of Arts and Sciences, Columbia University*. 1 – 164.

[7] Edwards, M. 2011. Algorithmic composition: Computational thinking in music. *Journal of Communications of the ACM*. 54(7), 58 – 67.

[8] Ijiri, T., Owada, S., & Igarashi, T. 2006. The sketch L-System: global control of tree modelling using free-form strokes. *International Symposium on Smart Graphics*. 138 - 146.

[9] Java Technology. 2010. Oracle and Sun Microsystems. Retrieved July 5, 2015, from website: http://www.sun.com/java

[10] Jones, P. 2014. [Demo abstract] patterning: Repetitive and recursive pattern generation using Clojure and Quil. *Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling and Design*. 71 – 72.

[11] Kaliakatsos-Papakostas M. A., Floros, A., Kanellopoulos, N., & Vrahatis, M. N. 2012. Genetic evolution of L and FL-systems for the production of rhythmic sequences. *Proceedings of the 14th Annual Conference Campanion on Genetic Computation*. 461 - 468.

[12] Lim C. K. & Talib A. Z. 2011. Improving L-Systems music rendering using a hybrid of stochastic and context-sensitive grammars. *Proc. of 2nd Int. ICST Conf. on Arts and Technology*. 46 – 53.

[13] Lim, C. K. & Talib, A. Z. 2010. A visual language framework for music rendering using L-System. *In Proc. of the 3rd WSEAS Int. Conf. on Visualisation, Imaging and Simulation*. 47 - 52.

[14] Longay, S., Runions, A., Boudon, F., & Prusinkiewicz, P. (2012). TreeSketch: Interactive procedural modelling of trees on a tablet. *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling*. 107 – 120.

[15] Majherová, J. 2007. Virtual plants in high school informatics – L-systems. *Conference ICL*. 1 - 7.

[16] Manousakis, S. (2006). Musical L-Systems. *Master's Thesis – Sonology, The Royal Conservatory, The Hague*. 1 - 133.

[17] McCormack, J. 1996. Grammar based music composition. *In Stocker et al, Eds. Complex Systems 96: from local interactions to global phenomena, IOS Press*. 321 – 336.

[18] Menzies, T. 2000. Evaluation issues for visual programming language. *Handbook of Software Engineering and Knowledge Engineering*. 93 – 101.

[19] Pradal, C., Dufour-Kowalski, S., Boudon, F., Fournier, C., & Godin, C. 2008. OpenAlea: A visual rogramming and component-based software platform for plant modelling. *Journal of Functional Plant Biology*. 35(10), 751 - 760.

[20] Prusinkiewicz, P. (1986). Score generation with L-Systems. *Proceedings of the International Computer Music Conference*. 455 - 457.

[21] Prusinkiewicz, P., & Lindenmayer, A. 1990. The algorithmic beauty of plants. *Springer-Verlag New York Inc*.

[22] Quick, D., & Hudak, P. 2013. Grammar-based automated music composition in Haskell. *Functional Art, Music, Modeling and Design*. 59 – 70.

[23] Sato, K. 2004. Design information framework, context-sensitive design and human-centered Interactive systems. *Conference on Human Factors in Computing Systems*. 1588 – 1589.

[24] Sharp, D. 2009. LMUSe. Retrived January 28, 2010, from website: http://www.geocities.com/Athens/Academy/8764/lmuse/lmuse.html

[25] Siew, B. H., & Talib, A. Z. 2009. Visual language framework for plant modelling using L System. *LNCS, Springer Berlin/ Heidelberg*. 5857: 696 – 707.

[26] Stepney, S. & Beaumont, D. 2009. Grammatical evolution of L-Systems. *In Proc. of 11th Conf. On Congress on Evolutionary Computation*. 2446 - 2453.

[27] Teresi, S. 2010. Scott Teresi's Website. Retrieved January 25, 2010, from website: http://teresi.us/html/main/programming.html

[28] Worth, P., & Stepney, S. 2005. Growing music: musical interpretations of L-Systems. *Lecture Notes in Computer Science, Springer Berlin/ Heidelberg*. 3449: 545 - 550.