G E N E R A L   A R T I C L E

Computational Aesthetics: On the Complexity of Computer-Generated Paintings

Kang Zhang, Stuart Harrell and Xin Ji

A B S T R A C T

This article discusses how visual arts and computer technology could complement and assist each other in new and emerging interdisciplinary areas known as computational aesthetics and aesthetic computing. The authors present examples of computational aesthetics that demonstrate that modern computer technology can generate aesthetic forms of visual art. Several levels of complexity in computerized abstract paintings are discussed and explored. The authors recently experimented with encoding computational rules to automatically generate a particular style of abstract painting in an attempt to explore one of the levels. The preliminary results of this research are presented. A more systematic and grammar-based approach is discussed as a potential future direction of work.

Many argue that with the rapid advances of modern technology, particularly digital display and computer graphics, digital art is more expressive than traditional visual art. This increased expressive power, compounded with highly advanced artificial intelligence, has created tremendous opportunities for the realization of computational aesthetics and even simulated creativity [1]. If the potential of computational aesthetics is achieved, we would see a profound impact on various application domains where computer technology has traditionally played only an assistive role, such as graphic and industrial design. This is not to advocate the possibility of replacing human creativity with computational creativity; rather, the advancement of computational aesthetics would further extend human creativity by providing inspiration to artists and graphic/industrial designers.

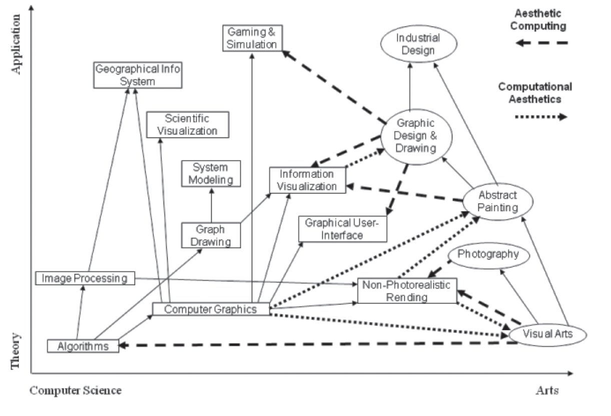
I. Emerging New Disciplines in Arts and Computing

When combining visual art with digital technology, two lines of questioning tend to emerge. On the one hand are questions such as: "How can the computer automatically generate various forms of visually aesthetic expressions?" On the other hand, some ask: "How can the theory and techniques of traditional visual art help beautify modern technology outputs and products and enhance their usability?" Addressing such questions, two interdisciplinary areas have emerged in recent years: *computational aesthetics* and *aesthetic computing* [2]. Both computational aesthetics and aesthetic computing aim at bridging computer science, philosophy, cognitive science and the fine arts through analytic and synthetic investigation.

Figure 1 illustrates a conceptual map of the relationships between various computer science areas (in rectangular nodes) and arts (in oval nodes), moving from theories to applications. The dotted arrows in Fig. 1 help to conceptualize that computational aesthetics aims at answering the first question above, that is, "How can the computer automatically generate various forms of visually aesthetic expressions?" In other words, computational aesthetics investigates how modern technology helps the arts. This technology in fact serves to create tools that can enhance the expressive power of visual art and heighten human understanding of aesthetic evaluation, perception and meaning. In the reverse direction, illustrated by the dashed arrows in Fig. 1, aesthetic computing addresses the second question—"How can the theory and techniques in the traditional visual arts help beautify modern technology outputs and products and enhance their usability?" This issue includes the aesthetic design of computer algorithms, simulation, visualization, human-machine interfaces and high-tech products, so that users are highly engaged and thus usability is enhanced. An interesting example of aesthetic computing is the application of Kandinsky's aesthetics to Java programming [3]. In 2006, Malina [4] highlighted the aesthetic computing activities published in *Leonardo* over the previous 40 years.

We have previously examined aesthetic computing, studying how information visualization using modern computer technology could benefit from the theory and practice of ab-

Fig. 1. A conceptual map of computational aesthetics and aesthetic computing. (© Kang Zhang)



Kang Zhang (educator), Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080-3021, U.S.A. E-mail: <kzhang@utdallas.edu>.

Stuart Harrell (student), Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080-3021, U.S.A. E-mail: <skh076000@gmail.com>.

Xin Ji (educator), School of Arts, Southeast University, Nanjing, Jiangsu, China. E-mail: <michelleji_seu@hotmail.com>.

See <www.mitpressjournals.org/toc/leon/45/3> for supplemental files associated with this issue.

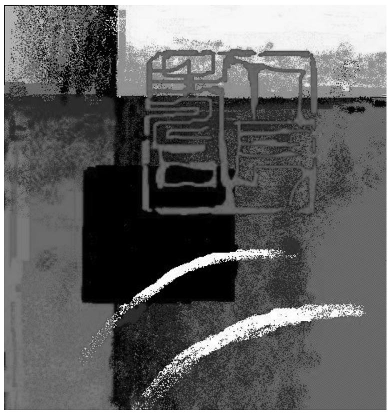


Fig. 2. Hand-drawn painting using a paint software program. (© Kang Zhang)

stract painting [5]. This paper will focus on computational aesthetics. Below we explore how to generate abstract paintings automatically using systematic and algorithmic approaches. First we classify various approaches based on the complexities of the computational intelligence utilized. Next we present our recent attempt at generating abstract works in the style of Kandinsky. Finally we explore more systematic approaches to the generation of computational aesthetics.

II. Computerized Abstract Painting

When considering computer-generated abstract paintings, we may consider four levels of sophistication based on the computational power utilized. Here we measure the sophistication of computer-generated abstract paintings by their computational complexity—that is, to what extent machine intelligence is utilized in generating the paintings—rather than by their visual complexity, as in Taylor's analysis [6]. At Level 1, one could use an existing painting software to draw paintings manually, as illustrated by the example in Fig. 2. At this level, the user may also select various visual components from a database of either manually generated or automatically generated components. With these works one can change visual attributes as needed. The computer provides digital brushes, a variable-sized canvas, a palette of colors and possibly a repository of commonly used visual elements that were manually drawn and saved in advance.

At Level 2, the user needs only to provide various attributes and styles, and possibly mathematic formulas, as inputs. A computer program can then automatically generate desired visual outputs. Fractal images, such as Fig. 3, are representative of works at this level. Research in fractals and fractal arts was extremely prolific from the late 1980s to the early 2000s [7,8]. One of the best-known researchers on this topic is Taylor [9,10]. Most other works at this level are freestyle fractal images (e.g. Fig. 3) automatically generated using adapted and randomized iterative formulas.

Two general approaches are used to characterize Level-3 computerized abstract paintings: generative and transformational. With the generative approach, artists' styles are encoded into computational rules and algorithms, so that it is possible to generate paintings of a particular style that mimic an original artist's paintings. One example at this level can be seen in automatically generated Mondrian-style abstract paintings, as shown in Fig. 4. We can trace these efforts back to the pioneering work of Noll [11].

The Level-3 transformational approach includes two-dimensional non-photorealistic rending (NPR) that is used to transform digital images into technical illustrations, cartoons, watercolor paintings and sketches, as well as abstract paintings. This approach applies image-processing techniques to an input image (usually a photo) to mimic brush strokes and texture patterns, such that the image is transformed into an abstract painting [12,13].

The main differences between levels 2 and 3 are that Level 2 is usually generated based on mathematical formulas augmented with certain degrees of randomness, while Level 3 uses knowledge-based machine intelligence and is typically heuristics-based. Level-3 research and practice has been most active in the last 10 years. The section below titled "An Attempt to Generate Kandinsky-Style Paintings" will further demonstrate a Level-3 effort. This is our recent attempt to generate Kandinsky-style paintings.

Level 4 is the ultimate aim of computational intelligence that is creative enough to generate highly aesthetic visual forms, such as abstract paintings or graphic designs. An abbreviated list of computational intelligence capabilities includes:

- automatic detection of specific styles from existing painting images
- objective aesthetic evaluation and measurements [14,15]
- adaptation to the social, emotional and cultural backgrounds of the audience
- automatic conceptualization of complex information into abstracted visual formats.

In Section IV below, "Toward More Systematic and Creative Approaches," we will explore a possible systematic approach as a Level-4 attempt and discuss future research directions in reaching the ultimate aim of maximal computational intelligence in enabling creativity. The four levels of complexity and their characteristics are summarized in Table 1. Human involvement in the Level 4 processes enables the combined and maximized creativity of human artists and computational generative power.

III. An Attempt to Generate Kandinsky-Style Paintings

We recently made an attempt to computerize Kandinsky-style paintings, based on Kandinsky's theoretical works [16–18]. The goal of this research was to experiment with the use of a rule-based approach to encode a specific style of Kandinsky paintings. Analyses of this nature fall in the spectrum of generative art, meaning that the art is constructed in a randomized autonomous manner. This independent generation of art is appealing, because traditional artwork has been constrained to the intuitive composition that is innate to humankind [19]. That is, the human artist does not compose stylistic elements without subjectivity, even when intending to. We proceed by synthesizing the two superficially divergent concepts of dissonance and consonance [20]. The construction begins by assembling a framework of aesthetic building blocks and then induces chaotic perturbations into the structure until harmony is achieved. This quality of measured discord combined with attention to aesthetics creates artwork that is fundamentally unusual yet pleasing to the eye.

Indeed, abstractionists' theories of form allow us to make an attempt to computerize Kandinsky-style paintings. Kandinsky wrote:

Fig. 3. Fractal art generated based on the Mandelbrot set. (© Kang Zhang)

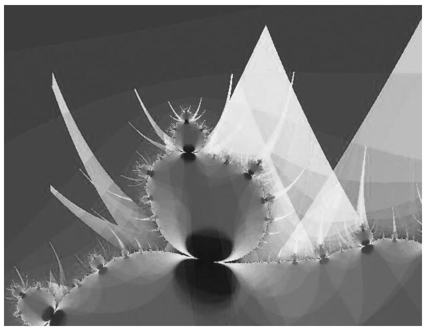


Table 1. Four levels of complexity in computerized abstract painting and design.

Level of Complexity	Human Participation	Means of Computer Support	Applicability	Example in This Paper
1	Full	Digital canvas and brushes, color palette	Painting and graphic design tools	Paint program (Fig. 2)
2	None	Mathematic forms, randomized	Fractal art, limited styles of paintings	Fractal art (Fig. 3)
3	None	Knowledge-based heuristic rules or patterns encoding given styles or domain knowledge	Domain-specific, or style-specific paintings and design	Program-generated Piet Mondrian (Fig. 4) and Section III
4	Minimal	Heuristic encoding aesthetic rules and multidisciplinary domain knowledge, customizable	Automated abstract painting and graphic design	Future direction

We must find, therefore, a form of expression which excludes the fable and yet does not restrict the free working of color in any way. The forms, movement, and colors which we borrow from nature must produce no outward effect nor be associated with external objects. The more obvious is the separation from nature, the more likely is the inner meaning to be pure and unhampered [21].

With the intention to leave the familiar world of objects behind, abstractionists prefer to use a small subset of primitive elements such as the circle, line and triangle, which can easily be generated by a computer. Consequently, it is not classical paintings, which depict landscapes and people with a strong emphasis on natural shapes, colors and refined skills, but the abstract arts that are more suitable for automatic generation using computational approaches.

The approach we have taken exemplifies the combination of aesthetic properties with induced discord. We can interpret this aesthetic appeal of pattern and randomness in terms of what Friedrich Schlegel described as “structured, artistic chaos”: the arabesque [22]. Although the phrase was used to analyze the aesthetics of literature, it vividly describes the aesthetic characteristics of our approach as a dynamic and complex system combining a variety of compositional elements. Three dominant stylistic elements are used in our approach to generate the paintings: the circle, the line and the triangle, as illustrated in Color Plate C No. 1. We advance by arranging these primitives in an aesthetic manner and then introducing dissenting alterations to the composition. The spirit of this construction is demonstrated by how the dimension, orientation and color of the elements are chosen. Each property has two associated sets of values it may assume, one corresponding to harmony and the other to dissonance. Several secondary elements that are derived from the primitive elements are also given prominence, such as the checker design and the intermingling of half-circles with straight lines. These are further combinations of the three primitive elements and have interesting properties. The checker design is a derivative of the equal squares of a checkerboard; the lines separating the equal squares are skewed and disfigured. The combination of the skewed and missing lines promotes discord, while the semblance to the unity of the checkerboard contributes to the aesthetic appeal. The generated paintings embody symbolists’ aesthetic-eclectic, ideational, “dematerializing” properties, which are characteristic of abstractionists.

The algorithm for composing elements on the screen follows our original blueprint of constructing an arrangement of primitive shapes and then disrupting it with random perturbations, as depicted in Fig. 5. We treat the line primitive as if it were an adhesive, that is, it associates the various elements with one another.

We begin by spawning a single line. Key points are picked along this line, including both ends and random points on the line. Furthermore, these key points, which act as connection junctures for primitives to attach to, are defined for every element. Next, for a specified number of iterations, a random element is placed at the randomly chosen key point of the previous element. The more iterations that are performed in this stage, the more harmoniously the elements appear together. The entire process, starting with the randomly drawn line, is then repeated for a pre-specified number of iterations. The more iterations that are performed in this stage, the more randomization or discord is induced. This enabled an easy method by which to manipulate the ratio of the combination of aesthetic properties with dissonance to find the optimum balance.

The composition approach outlined here reveals a number of similar properties between our computerized paintings and those of Kandinsky. They both “grow outwards from many points to envelop the whole (in a manner similar to the way new skin grows over a wound)” [23] rather than being developed from a single center as paintings of other genres are usually composed. The same forked approach was taken for color. The colors for each component of the painting are chosen randomly from a subset of either light or dark colors depending on what stage of the algorithm is being performed. If the stage of initial aesthetic construction is being performed, then the colors will be chosen to be complementary; otherwise, if we are in the final dissonance stage, random colors are used. Our simple contrast model could be improved by introducing color schemes, but this is not in play with the concept of a simple approach. Gradients were also introduced; for example, circles radiate from the light subset of color to the dark subset. This creates an interesting effect; it serves to distinguish the greater shape—the circle—from the rest of the greater shapes without any change in either dimension or composition. The balance and simplicity present throughout the specification and application of rules are what make this approach successful. Two more example drawings automatically generated are shown in Color Plate C No. 2(a,b).

In summary, mimicking Kandinsky’s style using a computational approach, as shown above, is extremely challenging. A deep understanding of the painter’s psychological and emotional expressions and much more sophisticated analysis are needed to properly encode his true

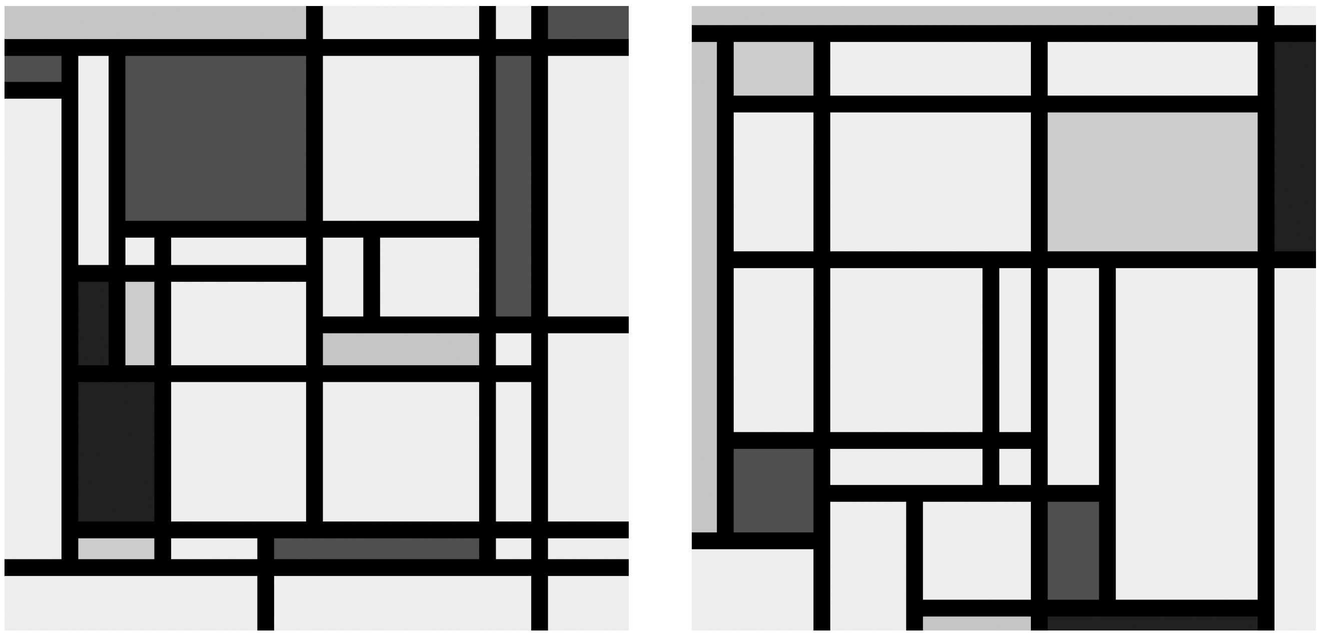


Fig. 4. Computer-generated Mondrian-style abstract paintings. (© Kang Zhang)

style into the generative machinery of rules and algorithms. Kandinsky once said "that one can feel the multi-sensory consonances and dissonances in simultaneously performed color movements, musical movements and dance movements" [24]. The aesthetic properties displayed in his paintings were perhaps profoundly influenced by his synesthesia [25]. Advanced artificial intelligence and social computing techniques at Level-4 complexity are required in order to generate any digital arts that may parallel Kandinsky's style.

IV. Toward More Systematic and Creative Approaches

An abstract painting or graphic design is often considered an aesthetic composition of various shapes, each with specific spatial and color assignment. The previous sections discussed the use of computational rules for encoding aesthetic principles and specific styles. Such rules, when expressed in graphs, shapes and spatial positions, are best represented as *grammars* that are programmable and executable by computers. A rule consists of two parts separated by an arrow pointing from left to right. The part to the left of the arrow is termed the Left-Hand Side (LHS). The part to the right of the arrow is termed the Right-Hand Side (RHS). The process of applying the rules to an existing graph to verify that the graph obeys the defined grammar is called *parsing*. The reverse process of applying the rules to generate all legal graphs is called *generation*. Each step during parsing or generation is called a *transformation*. The generation process is in the same spirit as Leyton's definition of aesthetics, based on the principles of maximization of transfer and maximization of reusability [26].

Graph grammars with their well-established theoretical background have long been used as a natural and powerful syntax-definition formalism [27] for graphical programming languages [28], which model structures and concepts in a 2-dimensional fashion. The parsing algorithm, based on a graph grammar, may be used to check the syntactical correctness and to interpret the language's semantics.

Different from all other existing graph grammar formalisms, the spatial graph grammar (SGG) [29] introduces spatial notions into the abstract syntax and is context-sensitive. In the SGG, nodes and edges together with spatial relations construct the precondition (as LHS) of a rule application. Figure 6 is a screenshot of the SGG tool called VEGGIE [30] during the process of constructing a grammar rule. Using spatial information to directly model relationships in the abstract syntax is consistent with the concrete representation, avoiding converting spatial information to edges. The distinct spatial capability in the spatial graph grammar makes the SGG an ideal specifying formalism to interpret abstract painting styles and graphic designs. SGG has been applied to mobile interface adaptation [31], UML diagram interpretation [32], reverse engineering [33] and Web interface interpretation [34].

Shape grammars [35], on the other hand, are a class of rule-based systems that generate geometric shapes. A shape grammar consists of *shape rules* and a *generation engine* that selects and processes rules. A shape rule defines how an existing part of a shape can be transformed. It depicts a condition in terms of a shape and a marker. The RHS depicts how the LHS shape should be transformed and where the marker is positioned. The marker helps to locate and orient the new shape [36].

Shape grammars can be used to form a basis for visual computation. The primitives in shape grammars are shapes, rather than symbols as in the SGG. The relationships and operations are all spatial (e.g. similarity, rotation) rather than symbolic, and thus complement spatial graph grammars. Shape grammars have been studied and applied in designs of mechanical parts [37], row-houses [38], floor layouts [39], decorative patterns [40] and interior layouts of buildings [41], as well as generating painting styles [42]. A shape grammar interpreter (SGI) developed by Trescak is available online [43,44]. It supports real-time sub-shape detection and labeled rules, as illustrated by an SGI screenshot in Fig. 7.

An interesting application of the spatial graph grammar formalism combined

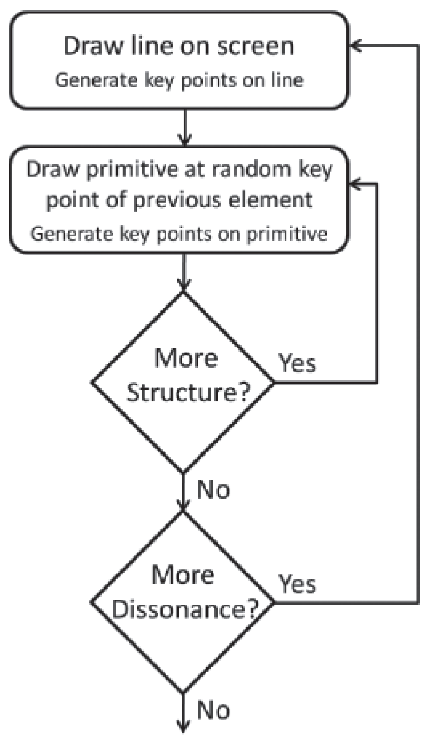

Draw line on screen
Generate key points on line
Draw primitive at random key point of previous element
Generate key points on primitive
More Structure?
Yes
No
More Dissonance?
Yes
No

Fig. 5. A high-level view of the drawing algorithm. (© Kang Zhang)

with shape grammar formalism is to generate shape-based abstract painting styles like that of Miró. Kirsch and Kirsch first proposed to model Miró's style by storing typical Miró shapes in a database and then manually analyzing the target composition using the stored shapes [45]. They did not encode the analysis into a shape grammar but offered some preliminary discussion. In our proposed approach, the shape grammar is responsible for generating singleton and composite shapes derived from a user-provided set of basic shapes. The spatial graph grammar is used to position these shapes in the provided 2-dimensional space. A combined parser for both grammars is able to generate an almost infinite number of visual images with different shape compositions and positioning. Creativity is thus demonstrated by the generation of many potential solutions toward specific designs [46]. From this point, two approaches to complementing computational intelligence with human creativity may be used to refine the solutions.

Using the first approach, the human designer would manually refine (possibly adding/deleting/modifying) the grammar rules to introduce more constraints in order to filter out less-desirable solutions. Then the grammatical system would be executed again with the refined grammars to generate better solutions. This iterative process could continue as many times as necessary until one or more satisfactory solutions are generated.

The second approach uses more computational intelligence. The human designer selects a subset of the solutions, based on which the grammatical system automatically refines the grammars itself by parsing the selected and unselected solutions and revising the grammar rules. The newly generated solutions are then selected by the human designer and fed back to the system again. This iterative process could continue as many times as necessary until one or more satisfactory solutions are generated.

Level-4 computational aesthetics poses challenging questions to artificial intelligence researchers, such as how to complement a human's creativity using computational creativity and whether machine intelligence could eventually become more creative than human beings. Aesthetic principles such as those of Gestalt theory [47,48] could be programmed into production rules.

In the domain of chess competition, a computerized chess player, IBM's Deep Blue, has already been able to beat all the world chess champions, including the Russian Grandmaster Gary Kasparov. Can the computer be an artist? [49] We believe that, given a constrained domain, such as graphic design for a specific application, computer-generated aesthetics in a 2-dimensional or 3-dimensional art form will soon be useful.

Acknowledgments

We would like to thank the anonymous *Leonardo* reviewers for their insightful and invaluable comments, which have helped us to improve the paper's presentation significantly. Stuart Harrell was partially supported by a UT-Dallas Undergraduate Research Grant.

References and Notes

Unedited references as provided by the authors.

1. M. Emmer, *The Visual Mind II*, MIT Press, 2005.

2. P. Fishwick (Ed.) *Aesthetic Computing*, MIT Press, 2006.

3. C.B. Price, From Kandinsky to Java (The Use of 20th Century Abstract Art in Learning Programming), *ITALICS*, Vol. 6, No. 4, October 2007, 35–50.

4. R.F. Malina, A Forty-Year Perspective on Aesthetic Computing in the Leonardo Journal, in: P. Fishwick (Ed.) *Aesthetic Computing*, MIT Press, 2006, 43–52.

5. K. Zhang, From Abstract Painting to Information Visualization, *IEEE Computer Graphics and Applications*, May/June 2007, 12–16.

Fig. 6. Composing an SGG rule in VEGGIE. (© Kang Zhang)

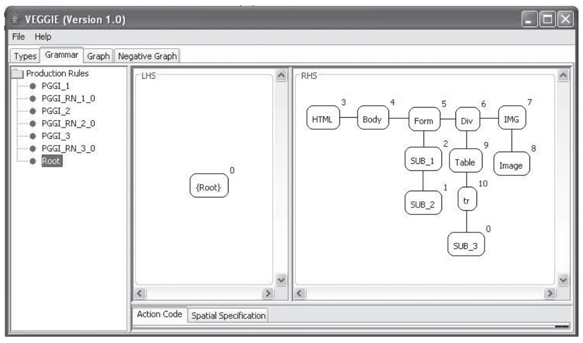
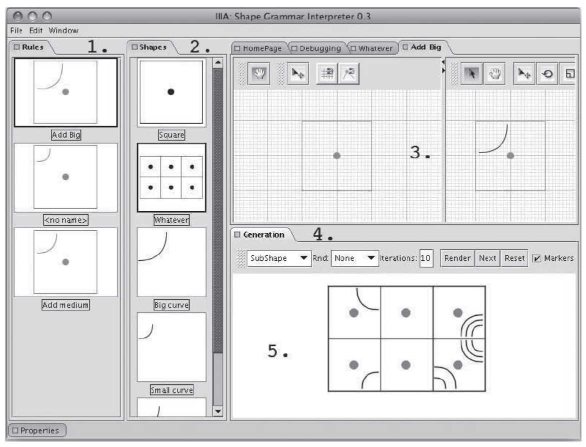


Fig. 7. A screenshot of Shape Grammar Interpreter. (© T. Trescak [50])


IIIA: Shape Grammar Interpreter 0.3
File Edit Window
Rules 1.
Shapes 2.
HomePage Debugging Whatever Add Big
Add Big
<no name>
Add medium
Square
Whatever
Big curve
Small curve
3.
Generation 4.
SubShape Rnd: None Iterations: 10 Render Next Reset Markers
5.
Properties

6. R.P. Taylor, Pollock, Mondrian and Nature: Recent Scientific Investigation, *Chaos and Complexity Letters* **1**, 2004, 29.

7. B.B. Mandelbrot, *The Fractal Geometry of Nature*. W.H. Freeman and Company, 1982.

8. M.F. Barnsley, *Fractals Everywhere*. Morgan Kaufmann, 1993.

9. R.P. Taylor, A.P. Micolich, and D. Jones, The Construction of Pollock's Fractal Drip Paintings, *Leonardo* Vol. 35, 2002, 203–207.

10. R.P. Taylor, Order in Pollock's Chaos, *Scientific American*, December 2002, 116–121.

11. A.M. Noll, Human or Machine: A Subjective Comparison of Piet Mondrian's 'Composition with Lines' and a Computer–Generated Picture, *The Psychological Record*, Vol. 16. No. 1, January 1966, 1–10.

12. P. Haeberli, Paint by Numbers: Abstract Image Representations, *SIGGRAPH'90*, Dallas, 6–10 August 1990.

13. M. Zhao and S-C. Zhu, Sisley the Abstract Painter, *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering (NPAR'2010),* Annecy, France, 2010, 99–107.

14. J. Nesetril, Aesthetics for Computers, or How to Measure Harmony, in M. Emmer (Ed.), *The Visual Mind II*, MIT Press, 2005, 35–58.

15. J. Rigau, M. Feixas and M. Sbert, Informational Aesthetics Measures, *IEEE Computer Graphics and Applications*, Vol. 28, No. 2, March–April 2008, 24–34.

16. W. Kandinsky, *Concerning the Spiritual in Art*, translated with an introduction by M.T.H. Sadler, Dover Publications Inc., New York, 1977.

17. W. Kandinsky, *Point and Line to Plane*, translated by H. Dearstyne and H. Rebay, Dover Publications Inc., New York, 1979.

18. W. Kandinsky, *Composición VII*. 1923. Solomon R. Guggenheim Museum, New York.

19. Zhang [5].

20. A. Ione and C.W. Tyler, Was Kandinsky a Synesthete? *Journal of the History of the Neurosciences,* Vol. 12, No. 2, 2003, 223–226.

21. Kandinsky [16].

22. G.R. Thompson, *The Art of Authorial Presence: Hawthorne's Provincial Tales,* Duke University Press, 1993, p. 21.

23. V.E. Barnett and P.H. Barnett, The Originality of Kandinsky's Compositions, *The Visual Computer*, Vol. 5, No. 4, July 1989, 203–213.

24. Ione and Tyler [20].

25. A. Ione, Kandinsky and Klee: Chromatic Chords, Polyphonic Painting and Synesthesia, *Journal of Consciousness Studies*, Vol. 11, No. 3–4, 2004, 148–158.

26. M. Leyton, The Foundations of Aesthetics, in Fishwick [4], 289–313.

27. G. Rozenberg (Ed.), *Handbook on Graph Grammars and Computing by Graph Transformation: Foundations*, Vol. 1, World Scientific, 1997.

28. M.M. Burnett, Visual Language Research Bibliography, <www.cs.orst.edu/~burnett/vpl.html, 2010> (up to date).

29. J. Kong, K. Zhang and X.Q. Zeng, Spatial Graph Grammar for Graphic User Interfaces, *ACM Transactions on Human-Computer Interaction,* Vol. 13(2), pp. 268–307, 2006.

30. K.L. Ates and K. Zhang, Constructing VEGGIE: Machine Learning for Context-Sensitive Graph Grammars, *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'07),* Patras, Greece, 29–31 October 2007, 456–463.

31. J. Kong, K.L. Ates and K. Zhang, Adaptive Mobile Interfaces through Grammar Induction, *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'08)*, Dayton, USA, 3–5 November 2008, 133–140.

32. J. Kong, K. Zhang, J. Dong and D. Xu, Specifying Behavioral Semantics of UML Diagrams through Graph Transformations, *Journal of Systems and Software*, Vol. 82, No. 2, 2009, 292–306.

33. Zhao and Zhu [13].

34. J. Kong, O. Barkol, R. Bergman, S. Schein, C.Y. Zhao and K. Zhang, Web Interface Adaptation Using Graph Grammars, IEEE Transactions on Systems, Man and Cybernatics—Part C (in press).

35. G. Sting, *Shape—Talking about Seeing and Doing*, MIT Press, 2006.

36. Sting [35].

37. K. Brown, Grammatical Design, *IEEE Expert*, March–April 1997, 27–33.

38. G. Cagdas, A Shape Grammar Model for Designing Row-Houses, *Design Studies*, Vol. 17, No. 1, January 1996, 35–51.

39. E. Grabska, K. Grzesiak-Kopec and G. Slusarczyk, Designing Floor-Layouts with the Assistance of Curious Agents, *ICCS'2006*, Part III, LNCS 3993, 2006a, 883–886.

40. E. Grabska, K. Grzesiak-Kopec and G. Slusarczyk, Visual Creative Design with the Assistance of Curious Agents, *Diagrams'2006*, LNAI 4045, 2006b, 218–220.

41. K. Yue, C. Hickerson and R. Krishnamurti, Determining the Interior Layouts of Buildings Describable by Shape Grammars, Proceedings of 13th International Conference on Computer Aided Design Research in Asia, Chiang Mai, Thailand. 9–12 April 2008, 117–124.

42. J.L. Kirsch and R.A. Kirsch, The Anatomy of Painting Style: Description with Computer Rules, *Leonardo*, Vol. 21, No. 4, 1988, 437–444.

43. SourceForge, Shape Grammar Interpreter, <sourceforge.net/projects/sginterpreter/>, 16 August 2010.

44. T. Trescak, I. Rodriguez and M. Esteva, General Shape Grammar Interpreter for Intelligent Designs Generations, *Proceedings of the 2009 6th International Conference on Computer Graphics, Imaging and Visualization (CGIV '09)*, Tianjin, China, 11–14 Aug. 2009, 235–240.

45. Kirsch and Kirsch [42].

46. E. Edmonds and L. Candy, Creativity, Art Practice, and Knowledge, *Communications of the ACM*, Vol. 45, No. 10, October 2002, 91–95.

47. W.D. Ellis, *A Source Book of Gestalt Psychology*, New York: Harcourt, Brace & World, 1938, 71–88.

48. M. Wertheimer, Laws of Organization in Perceptual Forms, First published as Untersuchungen zur Lehre von der Gestalt II, in Psycologische Forschung, 4, 1923, 301–350. Translation published in Edmonds and Candy [46] [available at <http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm>].

49. J. Lee, Goodman's Aesthetics and the Languages of Computing, in Fishwick [2], 29–42.

50. Trescak *et al.* [44].

Manuscript received 13 August 2010.

Kang Zhang is a Professor of Computer Science at the University of Texas at Dallas, U.S.A. His research interests include information visualization, visual languages and aesthetic computing.

Stuart Harrell is an undergraduate student in the Mathematics program at the University of Texas at Dallas, U.S.A.

Xin Ji is a lecturer in Aesthetics at Southeast University, China. Her research interests include aestheticization of everyday life, art history and computational aesthetics.