Computational Expressionism, or how the role of random () is changing in computer art

Joanna Maria Berzowska and Walter Bender

Massachusetts Institute of Technology, Media Lab, Cambridge, MA 02139

ABSTRACT

The creative process can be described as a continuous feedback loop between the material and an artist's decision making process. A skill can be described as knowledge of a material that allows making more informed decisions and more controlled interactions. As the artist attains a deeper knowledge of a material, the cognitive process involved in creation diverges from technical considerations and concerns itself more with meaning and expression. With computation, the creative process is better described as an evaluative process. Computers allow a multitude of stored copies and variations and also permit the visual artist to create many compositions. The artist may subsequently choose the most appealing among them, refining procedures and algorithms through an evaluative process of trial and error. The traditional relationship between the artist and the computer has been one of artists exercising visual judgment in light of manipulation of material. In this paper, we contrast the extensive use of randomness with a more controlled expression given advances in our modeling of human vision and of imaging systems. The context for this discussion is Computational Expressionism, an exploration of computational drawing that redefines the concept of lines and compositions for the digital medium.

Keywords: computer art, human visual system, color science

1. ARTISTS AND ENGINEERS

Artists and engineers are separate individuals, and if they work together, something will come out of it that neither can expect. —Billy Klüver[1]

Is this a good thing? —John Maeda commenting on Klüver quote

Why do artists and engineers differ in regard to the consideration of expression in their work? Is it because engineers build tools that others (including machines) use in an unsupervised manner while artists typically operate in a continuous feedback loop between material and eye? Engineers leverage understanding of the human visual system to build more efficient systems and to provide maximum expression given available resources. Perhaps the paramount example is NTSC, where the combination of inspired signal processing and empirical understanding of the mechanisms of human color vision enabled engineers to unleash the dimension of color in the same amount of bandwidth previously devoted to black and white television[2]. More recent examples include the development of the JPEG and MPEG standards. Although there are some exceptions, e.g., Arnheim[3], Albers[4], Jacobson[5], and Swirnoff[6], artists rarely travel down the same path as engineers. While they certainly exploit available technology, decisions about expression are made by the artist's "eye" at the time of creation. Can (and should) the gulf between art and engineering be bridged? The remainder of this paper attempts to answer this question by examining the tools and models used by engineers and artists. Computational Expressionism, a potential synthesis, is also described.

1.1 Engineers, artists, and tools

Traditionally, computer engineers build tools and computer artists use tools to create art. Engineers build tools within constraints that are often driven by market demand and technical considerations, not aesthetic needs. Computational tools are not necessarily designed to maximize the breadth of expression. Artists use tools to restrict the aesthetic breadth and develop individual style. Artists take available tools and try to use them in new ways to achieve original results. With computational artistry, this becomes increasingly difficult as the algorithms that dictate form become more and more complex.

Artistic creation is predicated on the use of certain tools and thus working within the tools' constraints, but creation also needs elements of inspiration and unpredictability. The artist's work is such that its visual aspects should not be planned out through algorithmic means. Many artists believe that visual decisions should not be made by an external system. Interestingly, the materials themselves, paper, pigment, etc., routinely impose external limitations on the visual decisions that are made.

1.2 Metaphors and mappings

Computation is powerful in shaping the form and behavior of objects (as opposed to a piece of chalk whose behavior is well defined and readily learned). Consequently, computer artists are much more concerned with metaphors and mappings than with the interface. Perhaps this is because the computer's behavior appears almost arbitrary, without the justifications that physical materials offer. Definitions are abstract and conscious, not a natural outcome of the physical properties of a piece of chalk or a drop of oil. The medium demands an approach where the tools are fabricated and removed from the dictates of any physical constraints. Artists must rely on artifice and the conscious selection from within the multitude of possibilities that the medium allows. While they find this disturbing, often they do not feel a need to search for meaning in the tool itself. Where there is a direct metaphor to the physical world, there is little concern with the perceptual models or higher order mappings (rules) for composition. The metaphor is sufficient to enable the artist to reach for well-established methods in traditional media.

1.3 Chance and evaluative process of creation

Works of art are restored to life in our world, not their own. —Malraux[7]

Artistic expression is ultimately a matter of selection. A rock in a Zen garden is art not because it was crafted by human hands—it was shaped by nature—but because it was chosen by a human mind. Likewise, the relationship between the artist and the computer is most often one of the artist exercising visual judgment in light of random manipulation of process elements. In this manner, some artists explore the use of computation to generate two-dimensional, non-representational art. Algorithms are used to create visual shapes. This is referred to as the "open loop" process, where possibilities are generated and subsequently accepted or rejected.

The process of creating art with already available tools rests greatly on chance; mistakes made in the coding process; an arbitrary setting on a PhotoShop™ filter; and provides an experimental, evaluative approach to the creative process. When asked to produce a composition consisting of three lines that slice through a red square, the artist can write a program (Figure 1). The procedure is to subdivide each edge of a square into ten segments of equal length, and label the endpoint of each segment in a clockwise manner from 0 to 39. By asking the computer to generate three pairs of random integers in the range 0-39, we can connect these as endpoints of intersecting lines. Repeating the process one hundred different times produces one hundred compositions. The artist's role becomes evaluative rather than creative; the artist selects the most interesting creations.

The evaluative skills of the artist are important. The algorithm creation is a repetitive, iterative process that continues until the desired results are achieved. Similarly, the nature of computation is unique insofar as the digital medium has no concept of original or finished work. Copies and minute changes in the program itself are trivial to make and undo. The bits can be saved, changes undone, and the number of possible combinations of visual elements escalates. The artist must choose from many possibilities instead of creating in a single breath.

Computers are procedural beasts. The evaluative process of writing custom code, i.e., the decisions the artist makes in the act of programming, become the artistic procedures. The selection of procedures has to do with the individual artist's sensibility about the entire art-making process. "Each makes artistic choices as to which procedures he will articulate, what form the work will assume (size, materials), how the work will be presented—and, yes each one assembles the code to generate them—just so in terms of all their physical qualities whether in visual or sound arts forms."[8] The role of the artist, in response to new technologies, has been to shift towards a more evaluative role.

1.4 Perceptual models

We would like to contrast the extensive use of randomness with a more controlled expression given advances in our modeling of human vision and development of imaging systems. We would also like to discuss the use of computation as a tool for direct manipulation by artists.

Instead of using chance and evaluative models, why don't computer artists use more perceptual models in shaping the visual outcome? Why don't they demand that perceptually based tools be incorporated in the computer—tools that are defined by the artist and help make some of the crucial design decision. Instead of manipulating pixels, should artists manipulate high-level constraints and design decisions?

Perceptual models in computer art are constraints in an expressive space based on previous decisions regarding the final outcome. The artist can restrict some of the degrees of freedom and define a sort of personal iconography, personal aesthetic, by building tools that produce one aspect of the creative process. These are different levels of cognitive activity, where decisions made in an open loop process precede and shape the closed loop activity.

Perceptual models for most artists creating interactive work revolves around building a set of metaphors or mappings to use in the interaction with a computer. Artists construct "poetic worlds"[9] that invite an interactive participation from an audience or user. The artist provides "contexts"[10] of elements and construction rules. The user plays within these, altering elements and constructs a dynamic, aesthetic experience. The language of mappings is more suited to the artist's rhetoric, because perceptual models that assume a degree of correctness regarding the way things should look are perceived to rob the artist of some expressive freedom. This is ironic considering that most tools we use do just that. Computers themselves use perceptual models that dictate the way that color is represented on a CRT.

1.5 Computer artists should make their own tools

What we are describing is essentially the act of defining building blocks and generative rules for aesthetic expression, and their parameters within which the visual form is produced before proceeding with the closed-loop process of creating art. The parameters, in this case, are of a higher order and utilize perceptual models of vision theory.

For the remainder of this discussion, we narrow the definition of the "artist" to one who creates two-dimensional visual art that is composed on a computer monitor and either displayed on the monitor or printed. Temporal composition is also addressed. The work of such an artist belongs in one of three possible categories: (1) The creation of a static visual piece; (2) The creation of a visual piece that exhibits dynamism as one of the design elements; or (3) A visual interactive environment in which secondary users, or viewers, manipulate a set of carefully designed components. Computational Expressionism describes a method with which the artist makes drawings using self-defined drawing tools: lines that have customized shapes, movements, and behaviors. This example lends itself particularly well to the discussion to follow. The artist defines a set of parameters with which visual expression takes place.

Using traditional visual art tools such as pencils and paint, the creative process of composition is what we refer to as a closed-loop process. Visual decisions are made as the hand executes and the eye evaluates a work in progress. The artist can compensate for the failings of the medium. Computational Expressionism concerns a creative process that includes both the creation of tools (a set of elements and rules) and the closed-loop process of drawing, using the hand and an input device that records the hand's gestures.

2. ARTISTS AS ENGINEERS

Alberti was the first to suggest the artist's use of the pinhole in one-point perspective. Dürer's wood engraving *Draftsman Drawing a Vase* is concrete evidence that the Renaissance artists embraced such technology as a means to furthering their expression. These artists, along with other such notables as Brunelleschi and da Vinci are acclaimed for their engineering skills as well as their visual expression.[11]

The tradition of artist as engineer does not end with the 16th century. A recent example is the work of Jacobson, as described in *The Sense of Color.* [5] While his primary vocation was that of a painter, Jacobson developed the Modular Colors™ system in order to free the artist from the "drudgery and uncertainties of mixing colors with traditionally named paints." His goal was to expand the expressive reach of the artist by systematizing the relationship between imagination and realization. His system, well grounded in the science of color vision, helps the artist achieve a creative advantage by providing tools for thinking with color rather than tools for simply mixing colors.

Jacobson's system bridges the gap between color science and color communication. He builds upon the work of Munsell by adopting Munsell's three dimensions of color—Hue, Value, and Chroma. Jacobson begins by establishing a solid connection between Munsell and the human visual system. He goes on to describe how these dimensions and their interrelationships "can actually be put to work" towards the task of producing predictable color intermixtures, whether the medium is one of pigments, dyes, or emissive displays. Jacobson describes the transformation of the traditional "oval" palette to that of a grid in order to meet the demands of color in painting: color mixing and color composition. Finally, Jacobson makes a connection between the expressive quality of Modular Colors and the expression of artists as diverse as Piero della Francesca, Rubens, Cézanne, Matisse, van Gogh, Vasarely, and Indiana. The artist is supported by engineering, but not confined by it.

3. ARTISTS AS PROGRAMMERS

What art forms will emerge as artists start working with a programming language as a raw material? Designers of interactive digital forms who do not program must express form and interactivity with concepts derived from available software tools. Their image of the design space is fragmented and incomplete. It is shaped to a large extent by the tools they have used and by the solutions they have seen. Consequently, many possible solutions are unimaginable. In addition, because most tools with which one creates digital pieces have underlying metaphoric or stylistic structure, the content generated can be heavily influenced. By working in a programming environment, artists produce pieces that exist independently of the hand of other authors or programmers. Maeda, in *Design by Numbers*,[12] states that the core skill of a digital designer is the practiced art of computer programming, or "computation." Artists must design their own tools and their own interactive experience.

For a programmer, the artistic process is the act of writing the code. Different skills are involved in traditional art, great hand-eye coordination and intuitive faith in the hand, and computational art, concise, analytical thinking. Artists will choose one set of skills over another, depending on their level of comfort and practice. Programming languages can be designed for artists who necessitate a more bricoleur approach to producing algorithms and procedures. To support the bricoleur interaction designer, it is important that the tools and materials allow the designer to work fluently in an iterative and explorative modus operandi. It is important that the software environment enhances a "dialogue with the material" in the creative design process. [13]

In Svanaes' *Painting with Interactive Pixels*,[13] interaction designers are asked to construct GUIs by painting with pixels that have inherent behavior. Interaction is constructed directly with the brushes that create the lines and shapes on the canvas. Svanaes defines kinaesthetic thinking as corresponding to visual thinking for design, the modus operandi involved when interactive behavior is designed directly, without making use of abstract representations of behavior. For a visual artist the study of drawing primitives, the line, the circle, can be very instructive, especially those that deal with transformations such as translation, rotation, and scaling. To design a procedure for generating a new form one must think in primitive and analytical ways about the very nature of the drawing process.

3.1 Interaction model

Traditional drawing tools offer a narrow range of visual properties based on tangible, physical attributes of the materials and rely heavily on the artist's skills and abilities to produce compelling results. Because computationally based tools are more controlling visually and generally necessitate the insertion of levels of abstraction through interfaces and algorithmic representation, models of drawing in digital media suffer from inadequacies. While algorithms are very powerful in determining appearance, they must be chosen carefully, for specific aesthetic needs. However, custom algorithms can be created for executing individual drawings. Thus a model for drawing with computation includes the programming of specific tools for each composition. The bipartite process consists of first devising and implementing a set of visual elements and generative rules, and then engaging in the real-time closed-loop process of drawing.

The computer differs from other media in that it is both a constructive and a projective medium.[14] Computation invites us to represent, but also to interpret the world around us through experimentation and abstraction. In this context, artist-coded drawing tools are the programmed representations of what motion of the hand will produce on the display. They differ from the lines found in the physical world insofar as their appearance and behavior are entirely constructed, yet seek to illustrate, amplify, or abstract the properties of motion. In this model, the artist is able to pursue individual interpretations of her ideas about drawing and the meaning of gestures. This *conceptual* interpretation of gestures is a very personal part of the drawing process and a key element of the mental model of drawing with computation.

3.2 Skill

The art of drawing has historically been the domain of skilled and talented artists who devoted much time and effort to the creation of images. Skills combined with creativity were prized. Artists spent considerable time in the studio, working through interminable sketches, to master these skills.[12] Computers now provide a rapid and simpler process for creating many graphic representations that earlier demanded a high level of skill. Computers offer easy access to precise detail, high resolution, photo-realistic representations, stylized illustrations, and technical sketches. The importance of skill, of the mastery of a medium, is partially lost in commercial image software use.

The act of programming, however, provides an environment that must be learned and conquered to develop a closer relationship between the artist and the medium. This sort of complex understanding is necessary to move beyond a preoccupation with the medium, and allow a greater comfort that will cause the artist to concentrate on content and meaning instead of tools and process. The skills necessary to develop an expressive proficiency in the medium are those of thinking mathematically, programming, translating artistic vision into a concise, mathematical algorithm, and approaching composition in an explicitly procedural way.

4. COMPUTATIONAL EXPRESSIONISM

Computational Expressionism[15] presents an alternate model for drawing with a computer that demands a deeper involvement on the part of the artist in the creation of drawing tools. It describes a two-fold process, at two distinct levels of interaction with the computer. The artist has to program the appearance and behavior of drawing tools, and subsequently draw with these tools by dragging a mouse or other input device. Compositions incorporate higher-level conceptual and algorithmic design with the real-time gestural drawing process. We attempt to re-define the act of drawing in this medium and present a different creative model for visual composition, one that is more iterative and evaluative. The focus moves away from duplicating the methods and materials we know from traditional media, and moves towards developing a different perspective on visual thinking.

Our implementation consists of 24 Java-based drawing programs, each provides a selection of “computational lines” used to create images. We define the computational line as a sophisticated digital brush that incorporates elements of appearance, dynamism and behavior into the line definition, and responds to various parameters of a gesture, such as speed, direction, position or order. The marks rendered on the display are abstractions or representations of gestures.

A computational line has three attributes: (1) The computational line has physical appearance, which can be a set of points joining two endpoints to form a parametric curve, a shape, a pattern, or a color. The line can closely mirror a hand gesture with minimal embellishments, but it can also render a more abstract representation of the gesture. (2) The computational line has individual behavior or dynamism. Marks rendered on the digital canvas undergo some transformation over time: a shift in color, position, shape, or other attribute. (3) The computational line has behavior in its interaction with the other lines on the canvas. The line can sense the presence, proximity and topology of others, and respond in pre-determined ways. The line can push the rendered marks away with pseudo-magnetic forces, change their color, or affect their shape.

Computational lines are programmed by the artist as a necessary component of the drawing process. They are not created as tools for a secondary user. Each program is created with a particular aesthetic in mind, in order to produce a specific series of drawings.

4.1 First attribute: Appearance

We “draw” on the computer monitor by dragging a mouse over the table. The gesture is encoded as a vector of point object. Each point object is a Cartesian coordinate pair and a time parameter, which records the exact time of creation. The points the line traverses and their temporal ordering are factors in determining the line’s appearance. *Calligraphy 1*, shown in Figure 2, is an early example of a computational line, created to suggest the sort of difficult drawing that can be done with a fountain pen on a napkin. The fibers in the napkin interfere with the smooth motion of the pen. Blotches of ink can appear if the motion is paused. The thickness of the drawn line is determined by the speed with which it is drawn. The direction of the drawing also affects the texture of the line. The line becomes thicker, and generates blotches when the speed decreases considerably. To create shapes, or filled areas of ink, one must slow down the motion of the hand. The size of the shape is a function of the time the line is paused (with the finger on the mouse button), and the shape orientation is determined by the direction of drawing. A single computational line allows one to render marks of different thickness and to create shapes of various sizes.

Figure 3 shows some simple gestures drawn with the *Wheat* line, so named because of its appearance of wheat swaying in the wind. The first set of panels illustrates single gestures. The left panel represents a movement from the bottom right corner to the top left. The right panel is a visualization of the inverse movement. The following two panels illustrate slower, more detailed movements. The spacing between consecutive strokes varies with the speed of drawing. Tilting preserves the direction of the gesture; curved endpoints imply the direction of the hand’s motion. It is an intuitive, albeit more abstract drawing tool. Complex shapes can be created by a very simple gesture. In the second set of panels, modifications of the *Wheat* line algorithm are explored to achieve a variety of computational lines. Varying the length of the stroke, the amount of latency, the smoothness of the curve, direction, and other characteristics produces a wide range of computational lines.

The process of computational sketching consists of making changes in the code, compiling and running the drawing program, and making some sketches on the canvas. The drawings executed with the computational line instruct the next iteration of the algorithms. Changes can be saved to catalogue the progression of the computational line. The code evolves, and the algorithms are refined and varied. This iterative bipartite process of programming and drawing produces an individualized algorithmic style for this particular piece. The most successful algorithms are saved and incorporated into a drawing application. A color palette is selected by the artist for this particular computational line. The resulting drawing tool is used to create more complex compositions.

4.2 Second attribute: Dynamic behavior

The *Hairy* line shown in Figure 4 begins as a thick, black mark, which expands, grows in thickness, projecting hundreds of animated tentacles out of its spine. The color of later tentacles fades as they are drawn over previous ones that stretch away from the spine. As a result, the composition fades. Their dynamic life span ends when their color fades completely to white. The black edges remain, and become positive space. The different values produced by irregularly overlapping fading tentacles form a rich texture. The following sequence of stills illustrates the drawing process as several lines are added over time, building up a drawing. The left picture in each row depicts the state of the drawing as a new line is added. The right picture in each row shows the drawing after all animation has come to a standstill. Each line, although it is dynamic, comes to rest in a predictable final state. The animation does not continue indefinitely.

The interaction of lines and colors is particularly captivating and entertaining. Dynamic lines create a rich animated visual environment controlled directly by gesture. Unfortunately, the process risks becoming more engaging than the final composition. Designing dynamic lines is a difficult task. It is easier to devise algorithms when creating a static algorithmic line. With added dynamism it becomes increasingly more complex to conceptualize the connection between the gesture and the eventual composition. Sketching with the lines and altering their design becomes a fun but time-consuming process.

4.3 Third attribute: Interactive behavior

Emergence is a process in which a collection of interacting units, simple rules or behaviors, generates or acquires qualitatively new properties that cannot be reduced to a simple superposition of individual contributions. There is an element of emergence that governs composition in computational drawing, as computational lines are programmed with behaviors for interacting with one another to produce visuals that are more complex. In the *Stream* applet (Figure 5), the vertex of each set of flowing lines follows the movement of the cursor, modeled to evoke a feeling of fluidity. The lines respond to each other's presence with a simplified physics model, they sense proximity and push each other away. The compositions emerge from the interaction among the different dynamic behaviors of individual lines.

Even more so than with dynamism, it becomes increasingly difficult to think of a computational line as a composition element for the creation of static images. The interaction and animation becomes engrossing and more aesthetically fulfilling than the final piece. The focus of the creative process moves away from the desire to create a drawing and becomes more of an engaging interactive process, which has been described by Scott Snibbe as resembling that of a musical instrument.[16] We play with the lines not to achieve a final image, but to experience the pleasure of affecting a dynamic, responsive environment. The process, the experience, is of interest here. Just as it is pleasurable to elicit sound out of an instrument, we take great pleasure in extorting movement and reactions from the computational lines.

4.4 Computational drawing aesthetic

Traditionally, artists are trained in hand-to-eye coordination skills, not the more mathematical skills necessary to program a line from the point (90,30) to the point (10,140). Therefore, current digital drawing tools tend to shroud computational aspects and provide a graphical user interface (GUI) that inserts a layer of artifice between the user and the tools. This introduces stylistic constraints, both by pre-defining aesthetic choices and limiting the set of possibilities. Artists carry over skills from their previous work, and continue to use the new medium in the old way. To achieve an individual algorithmic style, the artist must customize the software used to create visual work. As it matures, the work will achieve aesthetic qualities that are proper to algorithmic style. This unique aesthetic does not mean that the algorithm will mimic work traced by the artist's hand in traditional media. Although it is just as individual, it is a separate form of thinking about visual representation.

Algorithmic lines do not lend themselves well to representational images. In particular, the human form is difficult to reproduce and interpret. The drawing on the left in Figure 6 is a quick sketch of a woman done with the *String* line. This is a very direct line, as it accurately traces the movement of the hand with only minimal embellishment. The sketch is neither evocative of human expression, nor computationally interesting. On the other hand, representational art need not be realistic.

A high level of stylistic abstraction is possible in the replication of human form and expression. The panel on the right shows a highly abstracted figure drawn with the *Calligraphy 1* line, introduced above, which produces line and shape from the temporal characteristics of a gesture. A quickly drawn line is represented as a ragged linear form. Pauses in the drawing process or a slower gesture produce shape artifacts whose size is proportional to sluggishness. In this image, the apposition of anthropomorphic cues and computational aesthetic produces a more intriguing image. The composition is interesting because it blends references to human form with a more abstract algorithmic drawing process. The algorithm, however, is very specifically suited for the task. It generates both direct linear forms that can define precise boundaries, and solid shapes of various sizes and orientations. The two can easily be combined to form compositions that appeal to our need to anthropomorphize our surroundings.

In general, the aesthetic qualities of an algorithm or a dynamic interaction outweigh representational potential. Thus, our work in computational drawing has been almost exclusively non-representational. We are interested in computational drawing primarily because of the gestural input, which we regard as a very powerful expressive device. The quality of lines is a strong aesthetic theme, together with the expressive potential of gesture, and. the textures and tones achieved through a layering of lines The resulting aesthetic highlights the tension between repetition and variation, the apposition of regular algorithmic patterns with irregular shapes and areas of tone and texture, the combination of mathematical order and gestural disorder. Both aesthetic qualities are produced with a single computational line, as illustrated by the two sketches in Figure 7. They are merely a function of the manner in which the line is used. It is a unique quality of computation that facilitates such visual constructions.

4.5 Why not perceptual models?

Most commercial software for visual arts is based on one of two paradigms: image manipulation and the replication of visual characteristics of preceding art forms and styles. The first is confined by the fact that image transformation tools and filters provided by software packages result in a look that is easily achieved and is consistent between users. The second revolves around direct metaphors of pencils, erasers, paint buckets and spray cans. The direct metaphors are restricted and thus restrict the range of expression we can produce. Computer paint cannot hope to compete with the real thing.

Computational Expressionism stresses the importance of programming custom tools for visual expression. Programming encourages the exploration of image through interactivity, dynamism, behavior, and other unique qualities of the computational medium. Artists who wish to use the computer as a drawing tool need to understand and master computation as a medium in order to produce art pieces that take full advantage of its unique characteristics.[12] Introducing artists to programming essentially means offering them a new model for thinking visually. This new model, which incorporates elements of higher level design, expands and transforms the possibilities for visual creativity.

The three attributes of computational drawing involve procedural definitions for the behavior of marks that the hand makes. Whether this behavior is defined as algorithmic representation, change over time, or interaction between distinct line objects, the design decisions are made primarily from an aesthetic viewpoint. What is interesting is that the natural restrictions of computation such as the speed of equipment and the structure of programming languages significantly influence design decisions. Still, aspects of visual expression such as contrast sensitivity or color balance are higher-level design decisions that are entrusted to the eye at the time of making the algorithms and the drawings are created. We accept the restrictions of machines constructed by engineers. We work around these restrictions in a closed loop, adjusting algorithms and methods until things look right, instead of programming perceptual rules. This element of chance, the random() function, dictates many design decisions made in this work. The evaluative process of creating programs and images allows computational artists to put less thought into the creation of tools and images, because decisions are easy to undo and multiple versions of programs are easy to save and modify.

Figure 8 shows the *Poppy* line, which can produce two very different visual styles, depending on how the drawing tool is used. The two images on the left show a very textured, tonal rendering of the algorithm. The image on the right shows a high level of contrast sensitivity, and results in a crisper image. It has been difficult to synthesize generalized tools or even a base set of classes that would follow perceptual models. Each drawing illustrates a substantially different visual concept. But while each composition is a singular expression, distinct perceptual rules may apply to each computational line. The line algorithm can be responsive to contextual factors such as spatial frequency, chromatic and achromatic contrast, masking, etc. The use of random() is necessary at some level, but more pixel-level decisions are made as a consequence of the deliberate application of perceptual rules.

5. CONCLUSION

There are two predominant reasons why artists are reluctant to relinquish more visual control to computers: (1) Regardless of the large number of decisions that computers already do make, artists are weary of the decisions that computers make for them; and (2) Artists place an emphasis on the use of "mappings," "metaphors," and "interfaces" while disregarding perceptual models. But the tyranny of trial and error and the romantic but ambiguous results of traditional practices can be replaced by employment of visually-based tools.[5] A better understanding of engineering by artists will produce richer expressive forms. Reestablishing the artist to the position of tool maker as well as tool user will lead to a new, *deliberate* expression. When the artist is the engineer, the outcome may still be something expected, but it will not be subject to the vagaries of random().

6. ACKNOWLEDGMENTS

This work was sponsored in part by the MIT Media Lab's News in the Future research consortium and by International Business Machines Corporation.

7. REFERENCES

1. B. Klüver, "On working with artists," IEEE *Spectrum* (July 1998).
2. V. K. Zworkyn and G. A. Morton, *Television: The Electronics of Image Transmission in Color and Monochrome,* John Wiley & Sons, Inc., New York, NY (1954).
3. R. Arnheim, *Art and Visual Perception: A Psychology of the Creative Eye*, Univ. California Press (1983).
4. J. Albers, *Interaction of Color*, Yale Univ. Press, New Haven, CN (1987).
5. N. Jacobson, *The Sense of Color*, Van Nostrand Reinhold Co., New York, NY (1975).
6. L. Swirnoff, *Dimensional Color*, Van Nostrand Reinhold Co., New York, NY (1992).
7. A. Malraux, *Museum Without Walls*, Doubleday & Company, Inc., Garden City, NY (1967).
8. R. Verostko, "Algorithms and the Artist," *ISEA'94 Proceedings* (1994).
9. B. Seaman, "Emergent Constructions: Re-embodied Intelligence within Recombinant Poetic Networks," *Proceedings of Consciousness Reframed: Art and Consciousness in the Post-Biological Era*, Wales (1991).
10. R. Ascott, "From Appearance to Apparition: Communications and Consciousness in the Cybersphere," *Leonardo Electronic Almanac*, MIT Press, Cambridge, MA (1993).
11. E. Renner, *Bibliography on Pinhole Optics in Science and Art from 5th Century BC to 1850 AD*, http://www.pinholeresource.com/books.html.
12. J. Maeda, *Design by Numbers*, To be published by the MIT Press, Cambridge, MA (1999).
13. D. Svanæs. "Kinaesthetic Thinking: The Tacit Dimension of Interaction Design," *Computers in Human Behavior* **13**, No. 4, 443-463, Elsevier (1997).
14. S. Turkle, *The Second Self: Computers and the Human Spirit*, Simon & Schuster, New York, NY (1984).
15. J. M. Berzowska, *Computational Expressionism: A Study of Drawing with Computation,* Masters Thesis, MIT Media Laboratory, Cambridge, MA (January 1999).
16. S. Snibbe, *Motion Phone*, http://www.snibbe.com/scott/mphone/index.htm.

8. FIGURES

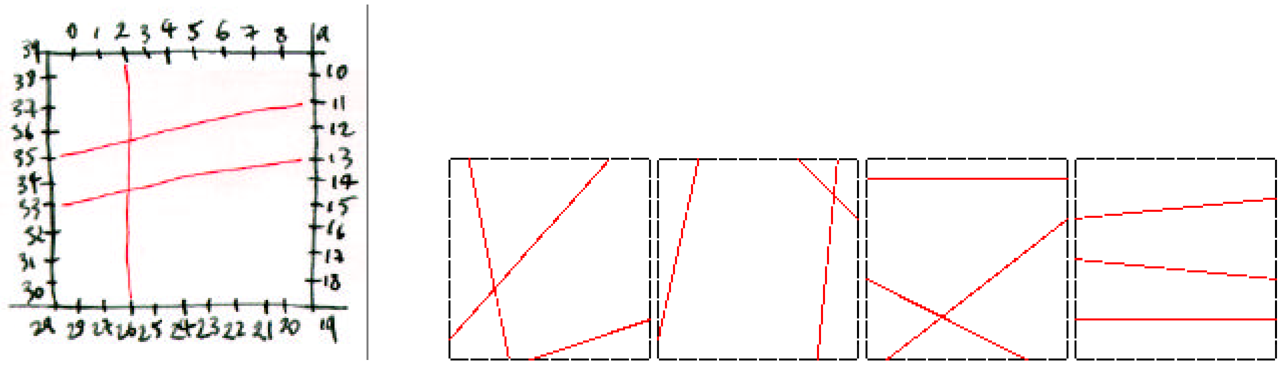


Figure 1: Three red lines.

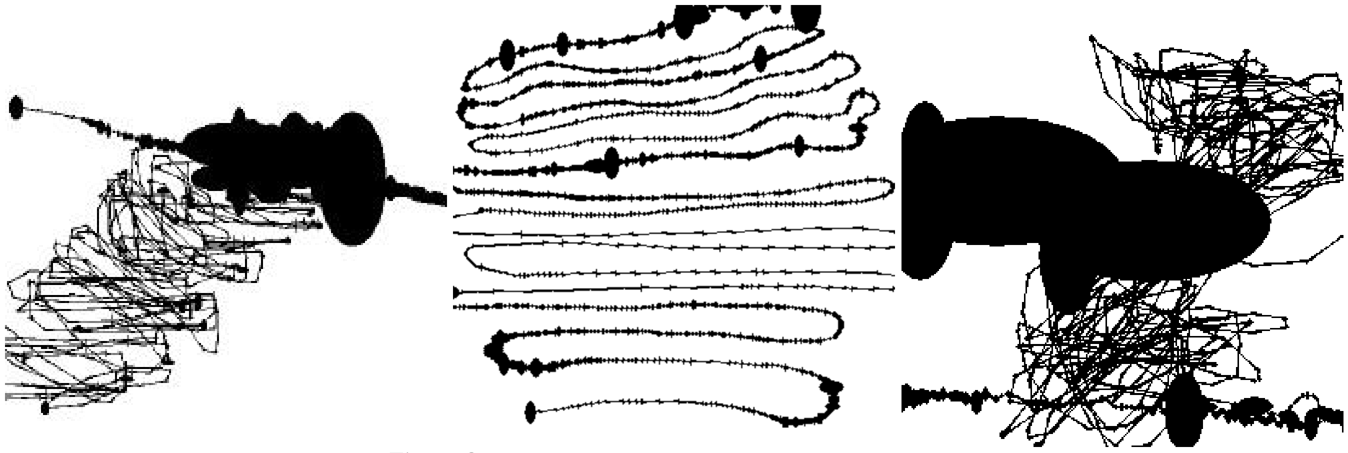
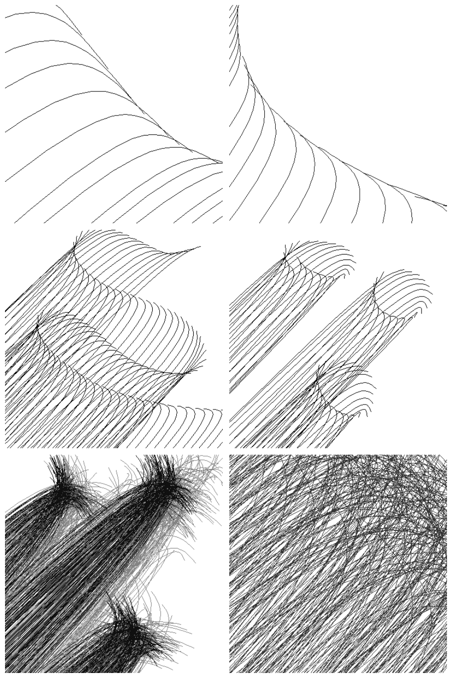
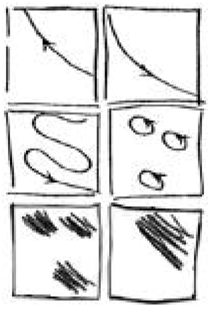
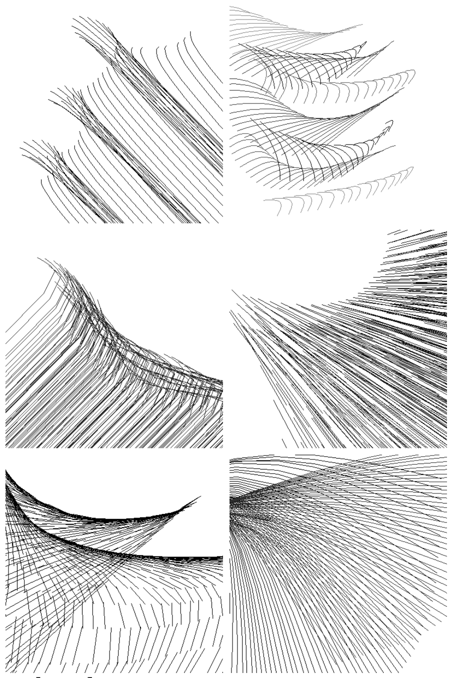


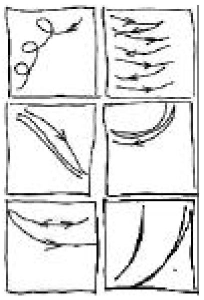
Figure 2: Three sketches made with the *Calligraphy1* line.





The six sketches on the left illustrate the *Wheat* line, programmed by the artist to reflect direction and speed of movement. The drawings were generated by the simple gestures shown above.





The six sketches on the left illustrate variations in the *Wheat* line. They were generated by the gestures shown above.

Figure 3: The Wheat line.

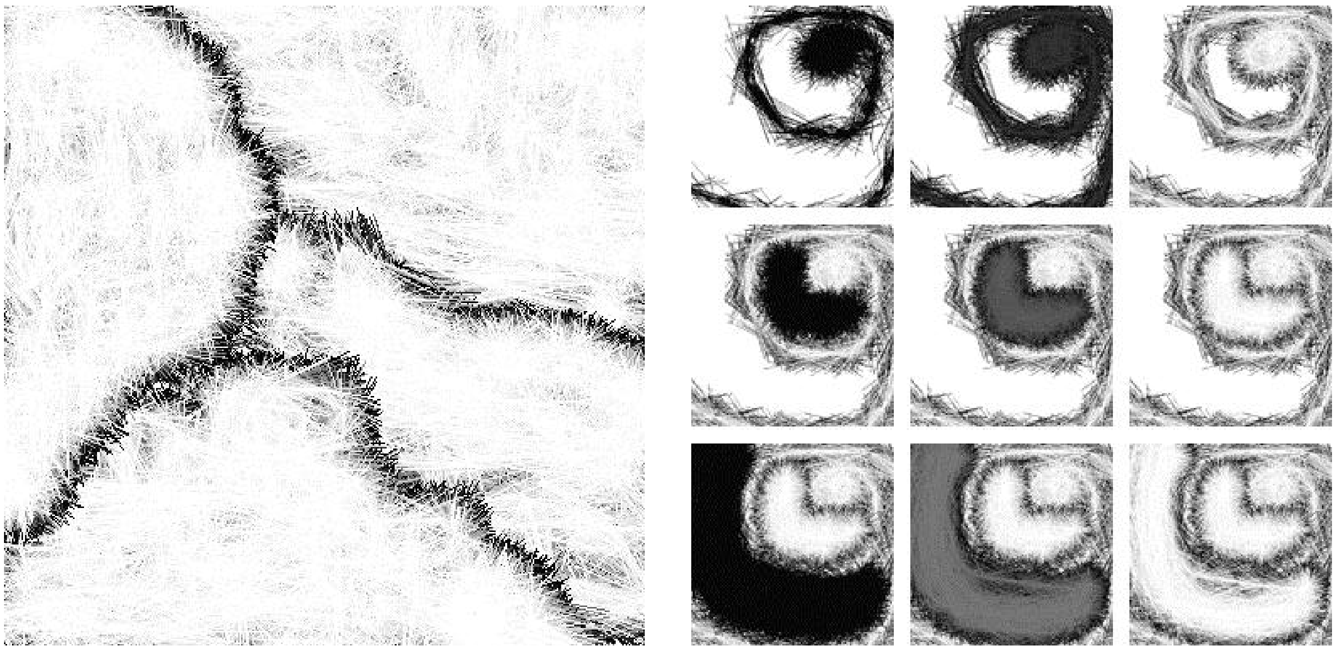


Figure 4: A drawing created with the *Hairy* line and consecutive stages in another *Hairy* drawing.

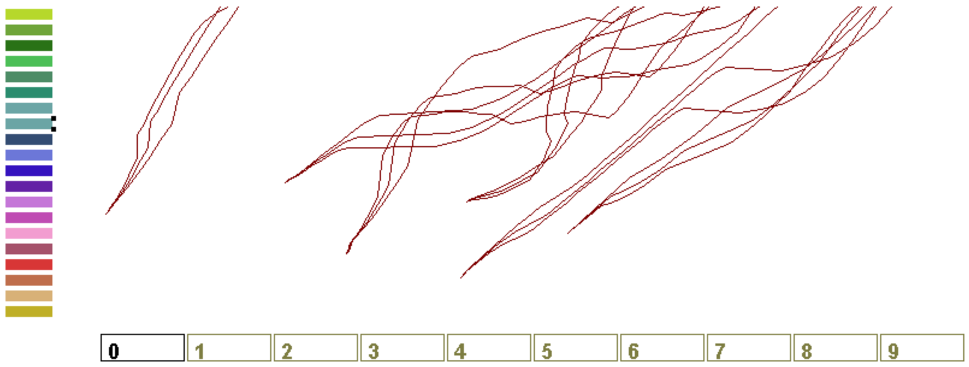


Figure 5: Interactive, dynamic representations of the *Stream* line.

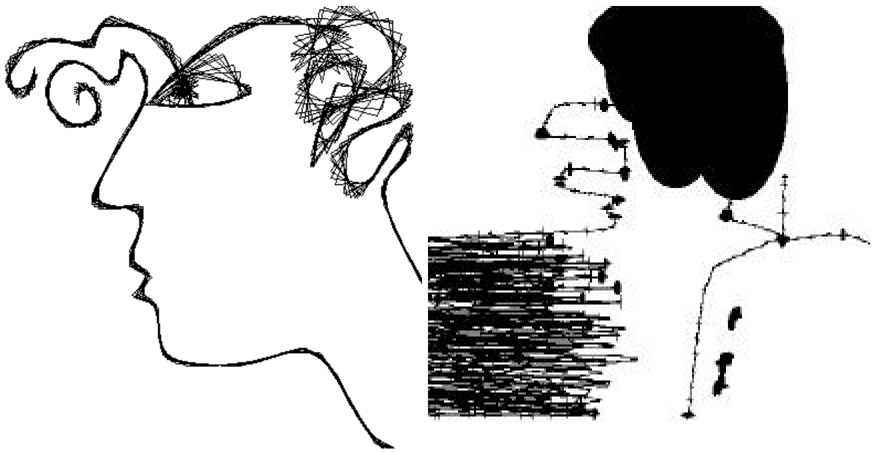


Figure 6: Girls drawn with the *String* line and *Calligraphy1* line.

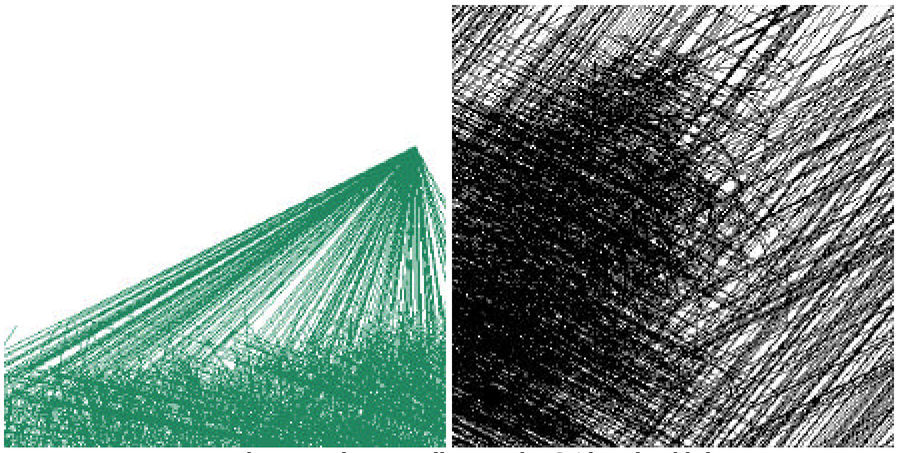


Figure 7: These two drawings illustrate the *Grids* and *Table* lines.

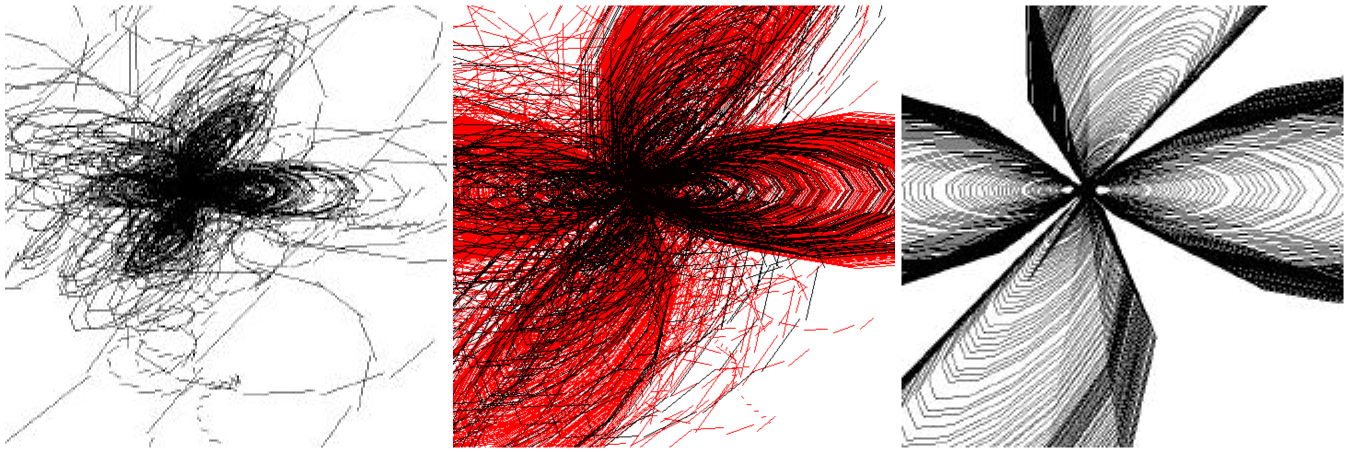


Figure 8: Three drawings created with the *Poppy* line.