The Evolving Role of the Artist

Prof. J. Sheridan, MFA
Chairman, Department of Experimental Animation
Maryland Institute College of Art, Baltimore, MD, USA.
e-mail: jamy@arthink.com

Abstract

For more than a decade the author has designed and used algorithmic systems to produce artworks that incorporate generative and evolutionary concepts, forms and processes. This work has demonstrated that algorithmic aesthetic processes and products can be effectively created and modulated by both human beings and non-human systems. However, this work has also raised important questions such as:

- What role can the individual human artist play in a cultural economy based upon industrialized generative processes and non-human systems?

- How can artists integrate standardized scientific languages and algorithmic processes into personal visions and expressive languages?

- How can artists capture their personal creative processes and encapsulate these processes in industry standard systems and software; and should they do so?

- How might the generative systems and products created by human and non-human artists function and evolve in the larger social context?

To address these questions, in this paper the author uses examples taken from his past and present artwork to illustrate the opportunities and pitfalls presented by computerized generative aesthetic processes and tools. In addition, the author offers a set of conjectures intended to help clarify issues such as: the evolving role of the artist as a producer of knowledge and form, and the value and appropriate structure of personalized computer languages for artists.

1. Introduction

During the past decade or so, I created a number of generative systems that I have used to produce computer-based, real-time, data-driven, human-scale, art installations and performances. These works embody concepts and images that grew from my eclectic synthesis of evolutionary concepts, algorithmic aesthetic processes, and art historical practices and precedents. John Dunn, an artist-programmer who I have had the good fortune to collaborate with for many years, created the computer tools and systems that I use to build and program my personal generative systems. He also composed the DNA-driven synthesized music that often accompanies my work.

It is worth noting here that John Dunn and I have both worked with Sonia Sheridan, Professor Emeritus of the School of the Art Institute of Chicago, and our work has been influenced by her approach to generative systems. For example, we share her belief that artists should be empowered to explore new forms of expression and production using contemporary industrial and scientific knowledge, systems and practices. We also share her conviction that artists must be able to develop unique personalized systems of practice and notation that complement the industrial practices and tools. We believe that artists can best explore and express their multidimensional, all-too-human, and often pre-verbal ideas by using multiple complementary creative systems that foster the integration of social scale and personal scale knowledge.

The purpose of this paper is to introduce the reader to some of this work and to illustrate some of the opportunities, pitfalls and questions that emerged from using computerized generative aesthetic processes and tools. To this end, I will discuss three different bodies of work, each built upon a unique computing platform and set of aesthetic concepts. I will also discuss the interaction between the concepts embedded in the tools, the concepts that the artist brings to the work, and the concepts that emerge from the interaction of the tools and the work. While some of these works and the computing platforms they were built on may appear visually archaic today, I find that many of the concepts and lessons they embody are still very relevant to contemporary generative art practice.

2. *Dark Matter* and *Tree of Life*

Dark Matter and *Tree of Life* are two installation artworks I created in 1994-1995 using the Vango software system created by John Dunn in 1991-1992. These works were presented as human-scale, immersive, real-time, computer animated installation performances with accompanying DNA-driven synthesized music [1].

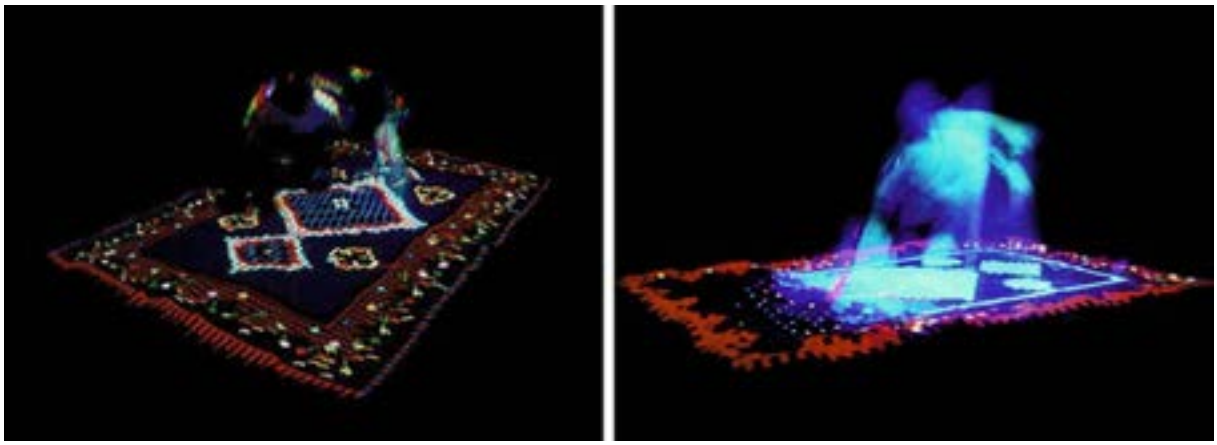

Figure 1. Still images taken during an animated *Dark Matter* installation-performance

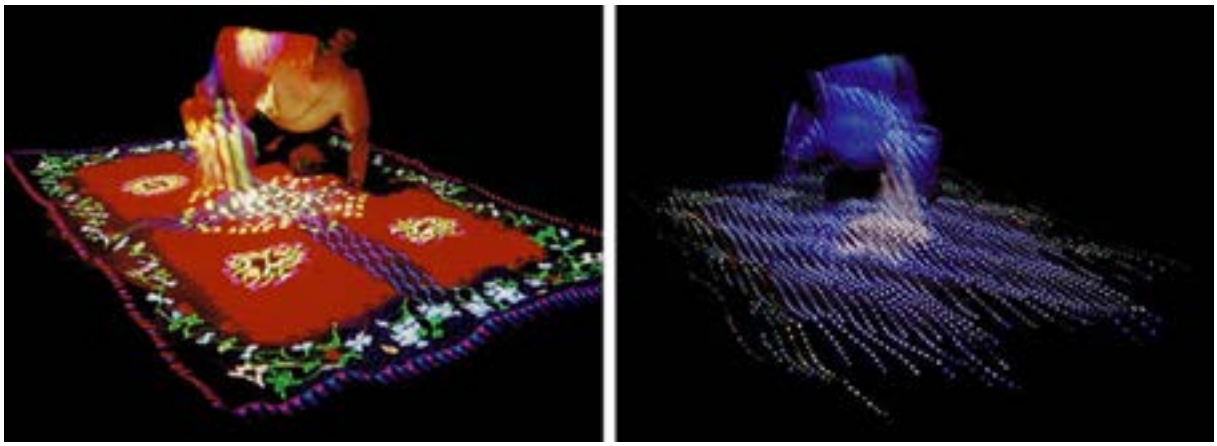

Figure 2. Still images taken during an animated *Tree of Life* installation-performance

Vango was a DOS-based, pre-web, real-time, animated hypertext system that used ASCII character graphics in innovative ways. John built Vango using character graphics because at that time he was constrained by a non-competition agreement that prevented him from developing 'real' graphics systems; i.e. pixel graphics systems like Lumena. Of course, it was precisely because Vango was not a 'real' graphics system that it was so interesting and revealing to work with.

For example, because Vango used character graphics, which were stored in a PC's hardware ROM, it was incredibly fast compared to pixel graphics systems of the time and therefore the creative feedback loop between the artist and the animated images was also very tight and fast. Of course, in addition to speed Vango had other unique structural and creative properties. Vango let an artist:

- 'Paint' in a 2 dimensional hyperspace using ASCII characters and using each character's CFB[2] attributes independently or in any combination.
- Combine multiple images in unique ways based on the CFB attributes of the characters that made up the images as well as on a transparent character attribute.
- Create user or system-activated buttons that jumped or slid the user to new locations in the 2D hyperspace while combining images in real-time.
- Create user or system-activated buttons that triggered MIDI events.
- Save and return to specific states of the system and therefore save and return to various states of the images, hyperspace trajectories, and CFB combinations.

After working with Vango for some time, I developed a whole set of generative processes that exploited Vango's capabilities and allowed me to develop a unique set of images and concepts. However, for the purposes of this paper, I think the system's great speed, its ability to combine sliding images in real-time, and its ability to individually set character CFB attributes are particularly worth mentioning.

Figure 3. Vango menus, color palette, CFB selector, navigation pads, and ASCII palette

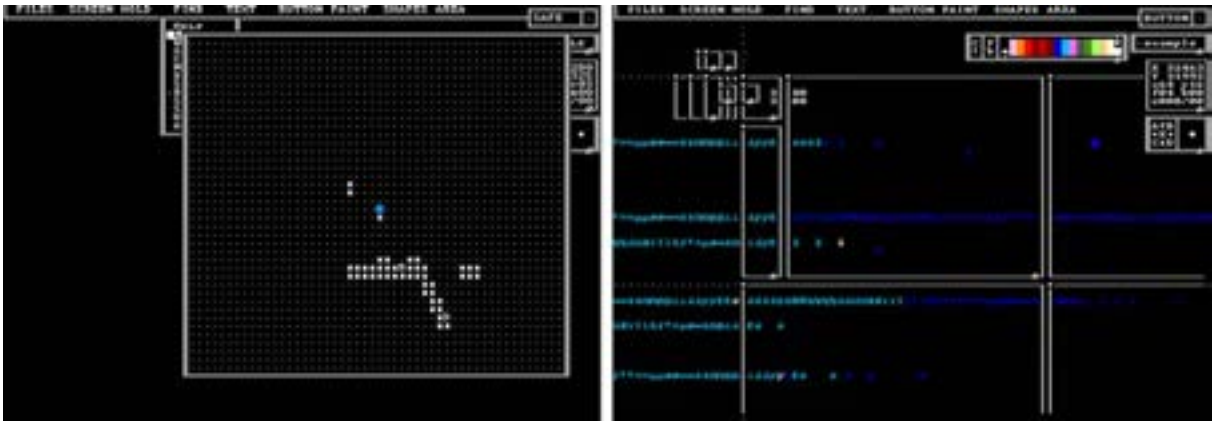

Figure 4. Vango hyperspace map and jump-slide-MIDI buttons

Because Vango could animate all 4000 characters on the 80 column by 50 row VGA screen at the rate of 50 characters per second when running on a 486 PC with local bus video, I was able to develop 'herds' of animated objects on my desktop PC. Each character position on the screen became an individual animation that was generated by sliding a series of characters past that position while 'fixing' the viewers eyes with stable color structures enabled by the CFB controls. This technique made it possible for a viewer to see animated characters rather than sliding characters. I could also group the sliding characters into visually meaningful units or herds that would, from the viewer's perspective, visually unite or break apart depending upon how I painted the interacting images and how I set up the hypertext button slide trajectories that combined the images. In any event, the animation effect was so strong that some viewers reported becoming seasick from the intense visual motion. (See the right hand image in figures 2 and 5 for static examples.)

Figure 5. Vango ASCII image structure examples

Having such a fast system also meant that I could use musical ideas to structure my visual processes and that I could involve viewers in the work in novel ways. For example, I used a

musical theme and variation structure to create the complex motion trajectories that animated the work's imagery. In addition, I used the fast graphics to create lines in the images that would scan and articulate the bodies of any viewers who were seated in the work. Vango's speed, coupled with its ability to change images, trajectories, colors, CFB interactions, and MIDI events on the fly, allowed me to produce a remarkably wide range of image structures and viewer experiences using a very limited set of generators.

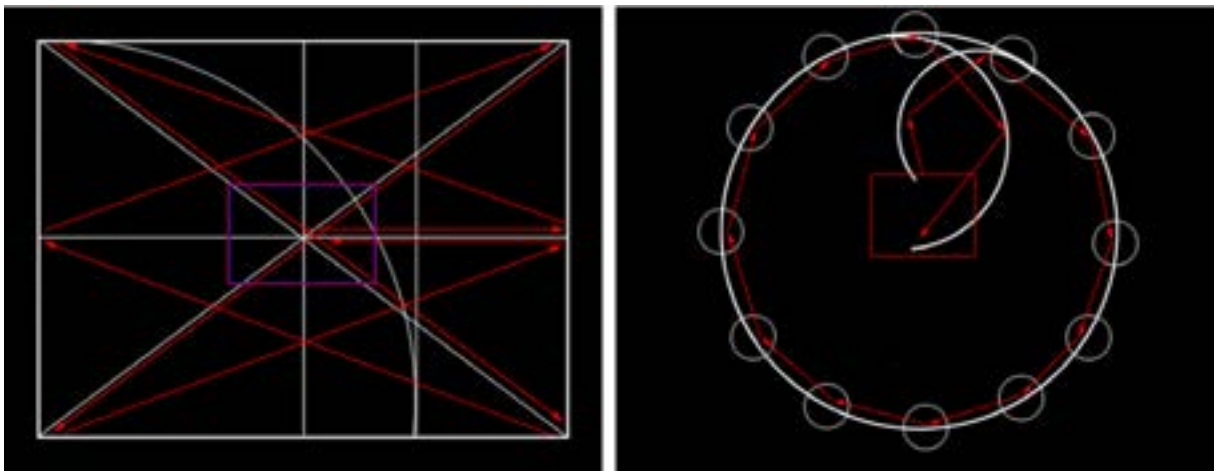

Figure 6. Hypertext sliding motion trajectory maps from *Dark Matter* and *Tree of Life*

While Vango certainly enhanced my creative process, the system also stimulated me to explore a number of genetic art ideas. For example, because Vango used the characters' CFB attributes when combining images as they slid or jumped through the hyperspace, Vango implicitly embodied a simple but dynamic genetic system. And, because Vango provided the ability to save and return to specific system states, it also provided a glimpse into an evolving genetic system in which my artistic decisions acted as the selective pressures. In essence, at any given point in the history of an artwork, I would use Vango to create a mother image and a father image. Then I would build the trajectories and assign the CFB settings that would combine the parent images' characters. The resulting real-time animation and combination process would produce a set of child images; or more accurately, a herd of child character sequences. I would repeat this process many times, then move back and forth in time to view the different genetic imaging systems I had created.

I found this whole process absolutely fascinating because I could use a simple, affordable, understandable, high-feedback, and animated computer imaging system to explore my multidimensional ideas about genetic imaging, structural transparency, fractal time, image evolution, process evolution, and the evolution of metaphor, to name a few. Moreover, at a

time when ‘real’ genetic animation systems could not realistically run on an artist’s desktop PC, I could actually work on my little 486 with herds of ‘object-oriented’ ‘genetic’ animations that lived in a large hyperspace and ‘interacted’ with the viewer. My imagination went wild.

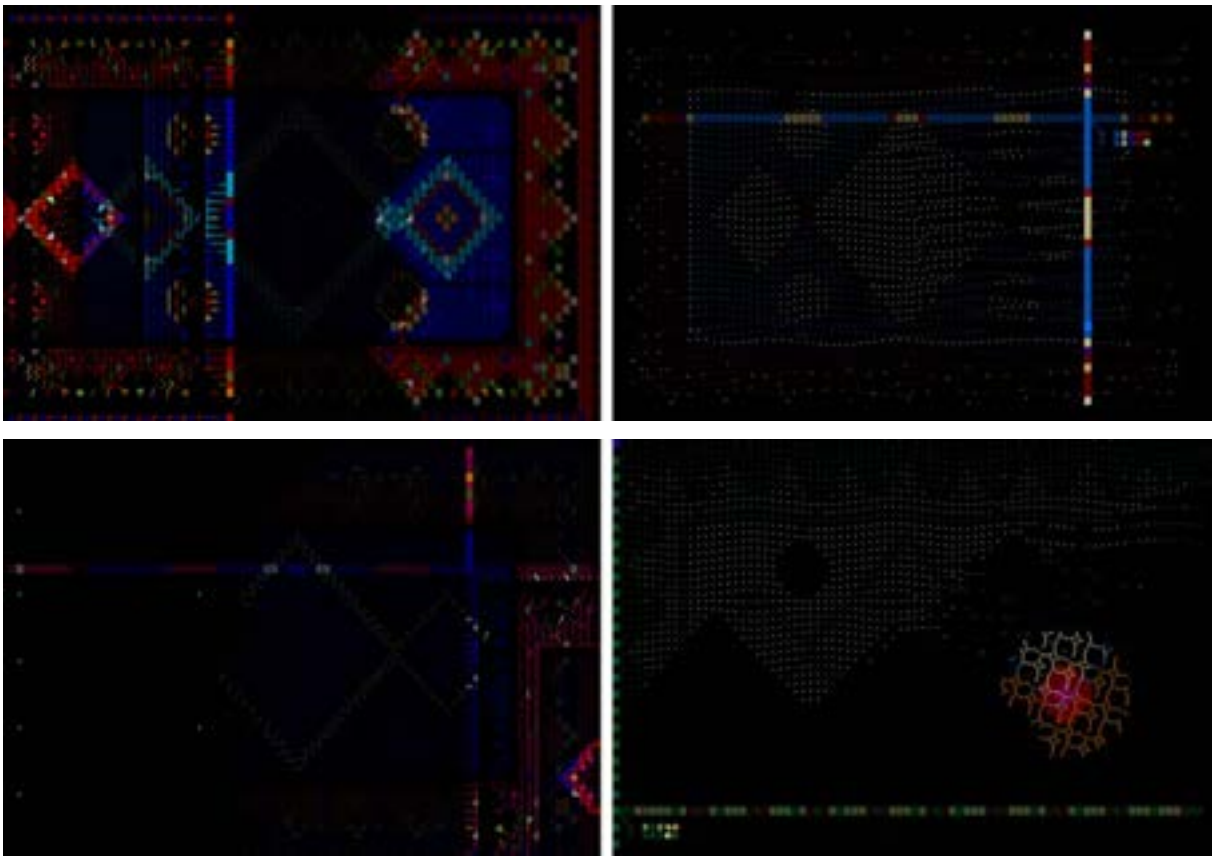

Figure 7. Various CFB 'genetic' interactions from *Dark Matter*

However, Vango had another structural property that made it interesting genetically; it used indexed characters and colors to create images. By using indexed entities to create images, Vango implicitly created a simple system in which there was a distinction between the image’s genotype and its phenotype. The genotype of a Vango artwork consisted of the character and color indexes that made up the images and the hard-coded button trajectories. The phenotype of the work was generated when the images were played and projected on a particular surface using a particular font and color palette. (See the right side of figures 1 and 7.) This aspect of Vango was also fascinating to me because I could create different ‘strains’ of an artwork simply by varying any combination of its CFB genetics, its font, and its projection environment.

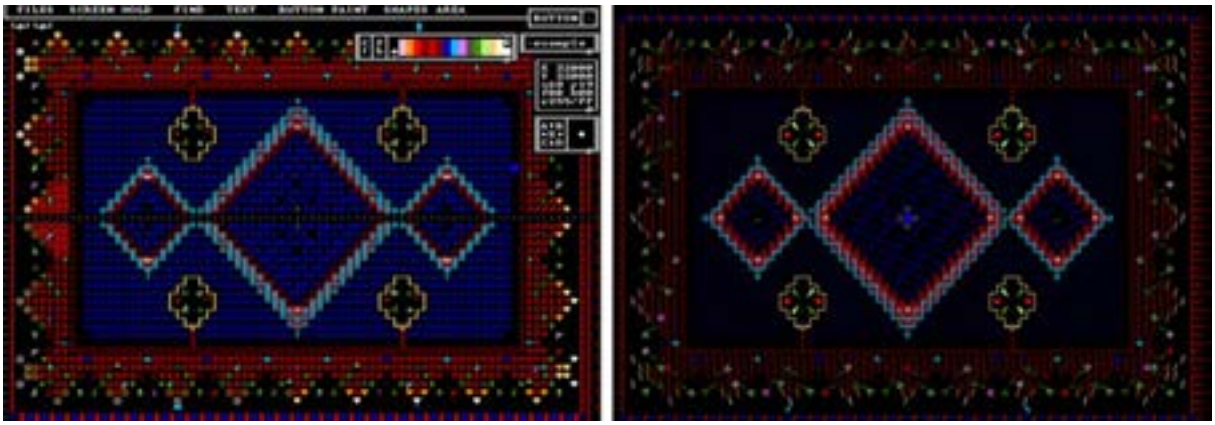

Figure 8. Genotype (visible as default phenotype) and screen phenotype from *Dark Matter*

Of course, using Vango also had its pitfalls, two of which deserve mention here. The first problem was that the more successful I became at producing interesting animations based on unique character fonts, the more difficult it became to actually use the system. This is because Vango, like most DOS ASCII applications, used the same character set for authoring the artwork and playing the artwork. As a result, the more I successfully modified the character fonts for aesthetic effect, the more difficult it became to read the Vango menus and tools, and the more difficult it became to read DOS screens. In fact, John and I used to joke that my pieces looked as if the were written in Klingon, an imaginary alien language used in the television series Star Trek. In the end, I had to memorize all the Vango menus because they had become unreadable.

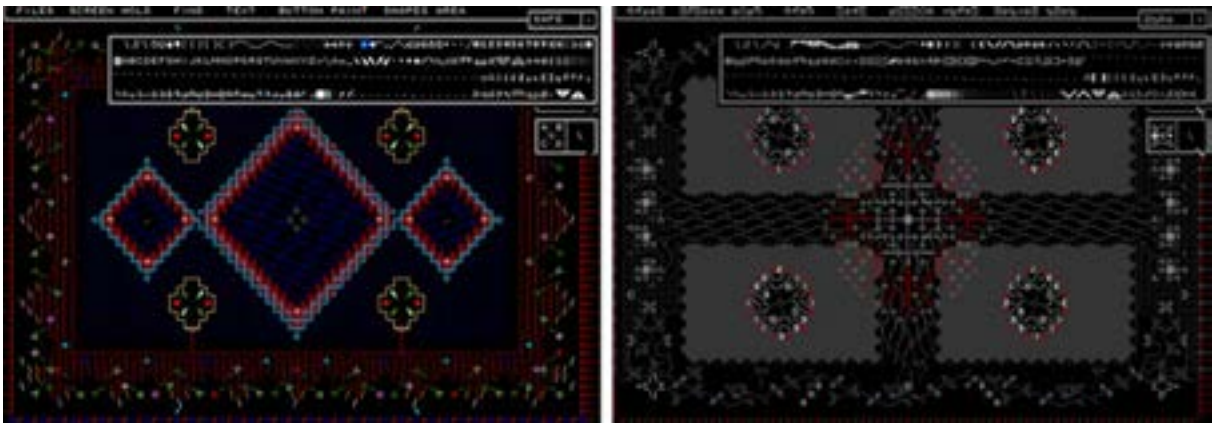

Figure 9. Progressively modified character sets from *Dark Matter* and *Tree of Life*

Figure 10. Progressively modified Vango menus and tools

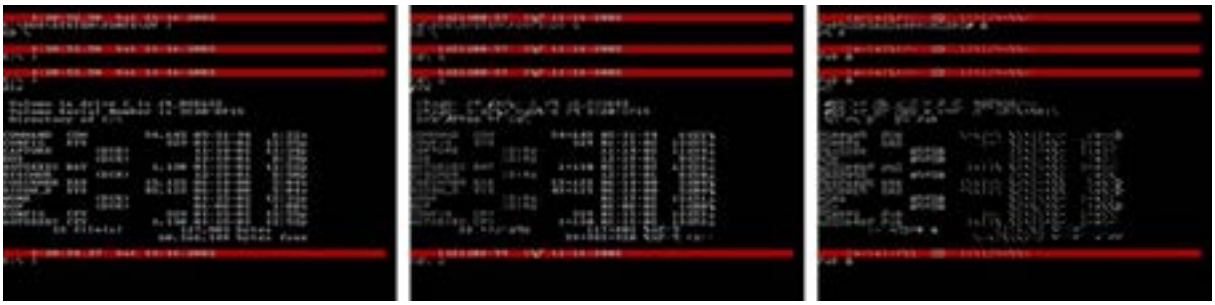

Figure 11. Progressively modified DOS screens

The second problem was that Vango was a unique world unto itself. It was character based, it could only handle pixel graphics in an extremely limited way, it could only generate a few of MIDI events, it could not cooperate with any of the other evolving pixel graphics systems, and its hypertext capabilities were about to be superseded by web tools. Vango had become too isolating and unique, so it was time to move on.

3. *Garden of Initial Conditions* and *Emerging*

Garden of Initial Condition and *Emerging* are two installation artworks I created between 1996 and 2001. As with my earlier work, these works were presented as human-scale, immersive, real-time, computer animated installation performances with accompanying DNA-driven synthesized music[3]. The evolving carpet-like images were projected on a dimensionalizing bed of sand in a totally dark installation space that made the images float dynamically. In addition, viewers were encouraged to play in the sand and become part of the work. The relaxed pace of these works in conjunction with the DNA music and the tactile projection surface gave the works a sensuous yet meditative quality.

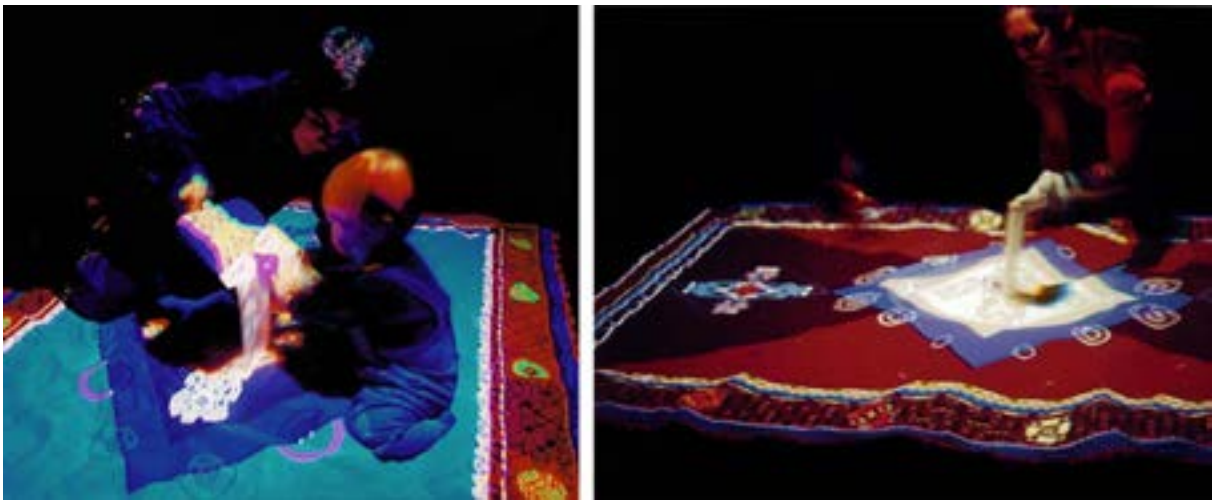

Figure 12. Still images taken from animated *Garden of Initial Conditions* performances

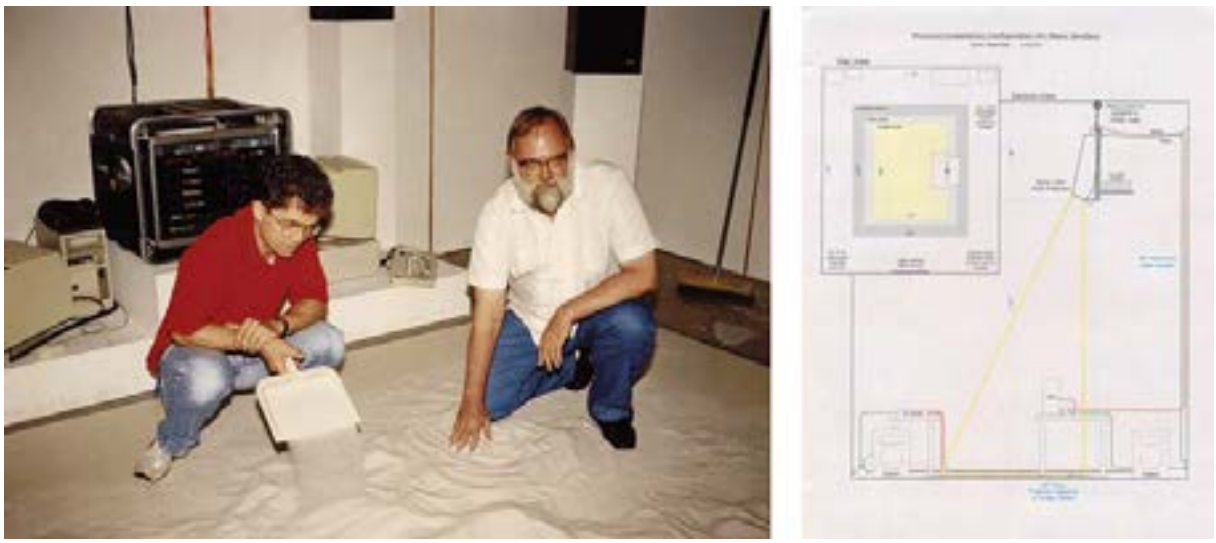

Figure 13. Jamy Sheridan and John Dunn preparing an installation

I created these works using Kinetic Art Machine (KAM), algorithmic multimedia software created by John Dunn in 1995-1996 with my assistance. KAM was a DOS-based software system that grew out of John's earlier Kinetic Music Machine (KMM), a MIDI composition and sequencing system he developed in the mid-1980's that enabled musicians to create sophisticated algorithmic compositions that would run in real-time on an IBM PC. KMM was the system John used to create his early DNA music, it was the system I used to create music for my first Vango works, and it provided a proof-of-concept for many of the ideas that were further developed in KAM. Some elements of KAM also grew out of an ongoing conversation about artists' computer systems and languages that John and I began in the mid-1980's.

Figure 14. KMM menus

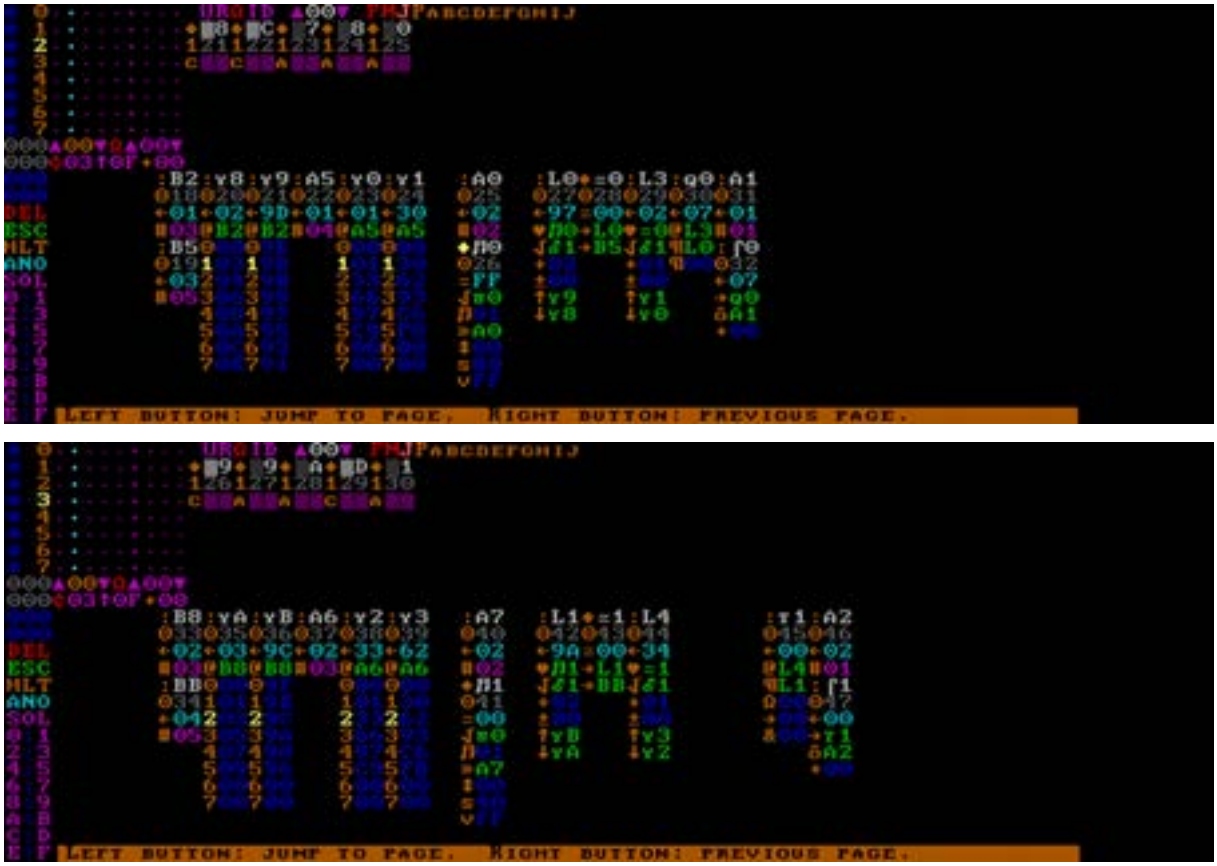

Figure 15. *Rug_27*; KMM code

KAM was conceived as a direct manipulation meta-language optimised for real-time MIDI and data-driven algorithmic visual processes. It was a meta-language in that it was a language to build languages, a general-purpose computer language that enabled artists to build personal languages of form and process. It was a direct manipulation language because KAM only provided functioning modules that the artist programmer could assemble into larger structures using a relatively metaphorless GUI. It was real-time system in that any changes to a KAM program structure or variables instantly affected the acoustical, visual, or

functional output of the program. KAM allowed the artist-programmer to work in a very fast, right brain feedback loop and thereby maintain a very direct relationship between the programming ideas and the aesthetic output.

KAM let an artist:

- Create complex real-time program structures that could be used to drive MIDI sound, generative raster graphics, and external processes.
- Explore unusual time structures and relationships by driving individual program processes with independent clocks.
- Save and index into specific states of the system and therefore save and return to various states of the aesthetic process.
- Drive evolving structures, images, sounds, and environments using real-world data, DNA and protein data, for example.
- Connect to the outside world by reading and writing to ports and memory addresses.

As with Vango, after I worked with KAM for a while, I developed a group of generative processes that exploited its capabilities and allowed me to develop unique concepts, mark making processes, and artworks. However, I found three of KAM's abilities particularly important: KAM could use data to drive audiovisual processes, it could embody aesthetic evolution in saved system states, and it allowed me to explore evolutionary concepts such as aesthetic genotype and phenotype.

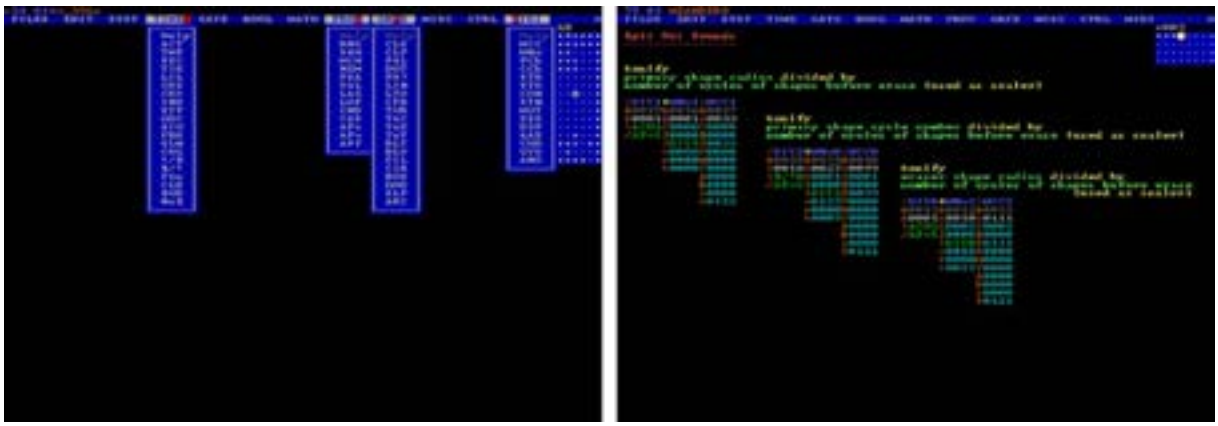

Figure 16. KAM menus and code example

KAM's ability to use DNA and protein data to drive both visual and musical processes was valuable for a number of reasons. First, John and I could use the same DNA-protein data to drive the music and the visuals for an installation. As a result, at a formal level the visual structures I produced with KAM worked very well with the music John composed; the music and images seemed to fit together effortlessly. I speculate that this because we both used the same source data and the same basic generative tools. In addition, using real DNA data allowed me to focus the viewer's mind on the general idea of code driven form and structural cascades; an idea that I believe has many implications in the fields of art, architecture, media production, and social theory to name just a few.

Using DNA data in the artworks also helped convey the idea that the artworks were alive. Physical genetic and cultural memetic processes became equated in the viewer's mind. DNA, at the root of a genetic tree of life, become associated with the 'tree of life' motif, the root of a memetic tree of metaphor. My 'DNA rugs' and the traditional 'tree of life' tribal rugs I referenced in my work were seen as part of the same evolutionary process. Moreover, since the artworks themselves had DNA, viewers thought of them more as living things.

Figure 17. ***Garden of Initial Conditions, strain 1*** **and** ***strain 2*****; KAM code and animation frames**

Conceptual issues aside, on a formal level using the DNA-protein data helped me to create very interesting evolving visual forms. For example, to create the central tree-of-life motifs seen in figure 17 above, I used DNA data to modulate the radii of a series of expanding, contracting and XOR'ing rectangles. The process was like a feedback loop, driven with an apparently random yet structured signal. In any event, by carefully tweaking generative parameters, I was able to get my code to produce some truly wonderful patterns that bore a remarkable resemblance to the familiar yet unique patterns found in best tribal carpets. I used this idea and related non-DNA strategies to produce many wonderful unique generative marks.

I also found KAM's ability to save and index into system states exceptionally useful. In *Garden of Initial Conditions*, I used system states to explore aesthetic genotypes and phenotypes. I created one basic code structure, i.e. genotype. I then copied the genotype, i.e. file, adjusted combinations of parameters such as color, timing, and limit value, then saved the new state of the work. I created multiple strains of the work by applying the selective pressures of my decision making process to copies of the genome; i.e. new files.

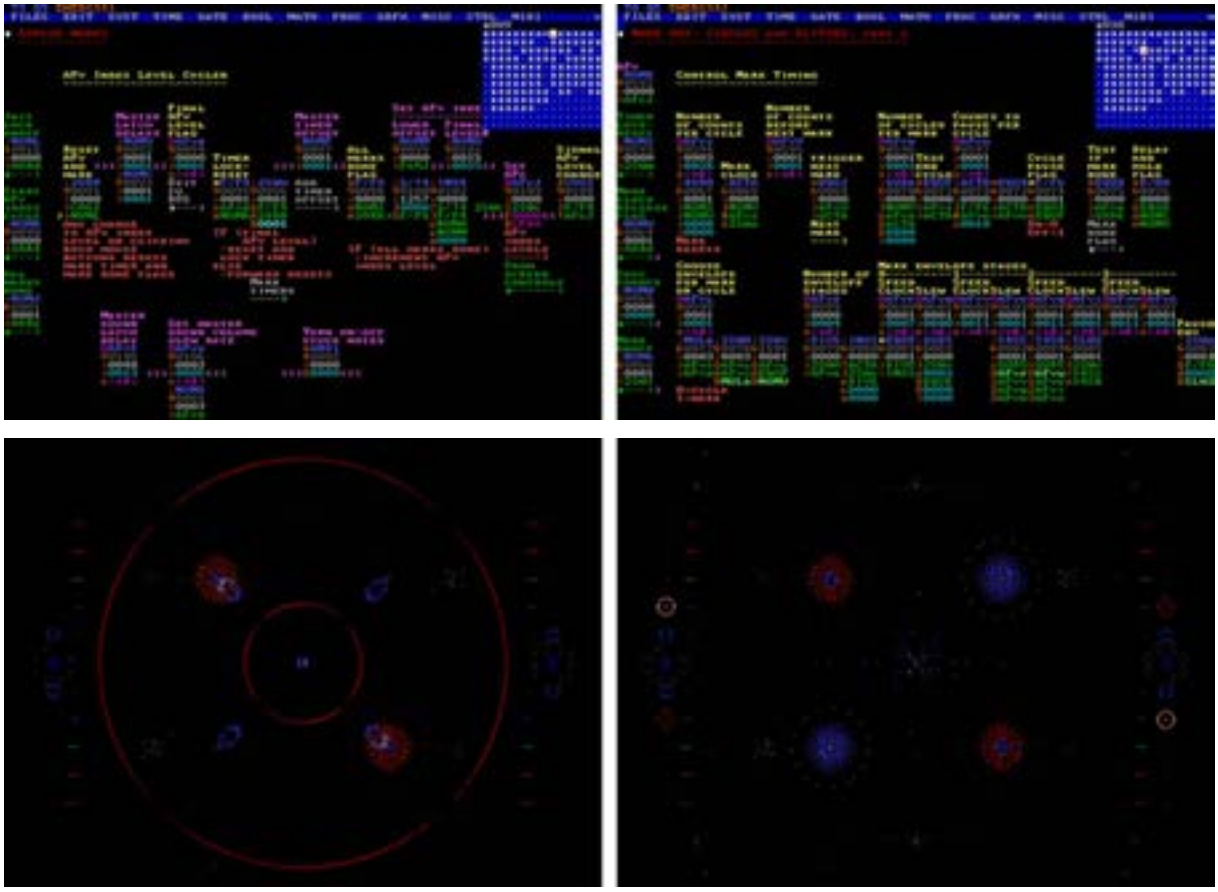

Figure 18. *Emerging;* KAM code and animation frames

The primary authoring and performance processes underlying *Emerging* were also based on evolving system states, but within a single file. More specifically, I built a simple mark-making engine driven by timed events, then created a series of marks and events which I saved as index level one. I then created another related set of marks and events which I saved as index level two, and so on until the work was 'complete'. This process worked very well, especially given the visual form of *Emerging*, inspired as it was by theories of cosmic and cultural evolution.

KAM certainly was a wonderful tool to use. But not surprisingly, it also presented its share of pitfalls. For example, like all young software, KAM evolved at a furious rate. When John first released a beta of KAM, I built 183 different versions of one piece before we uncovered all the bugs and unintended side effects within the system code and within my own artist's code. Ironically, just as we evolved KAM into being, the DOS world KAM was based on was nearing the end of its life cycle. DOS programs were becoming passé as the new GUI technologies began to dominate.

However, the worst problem associated with the demise of DOS was that, while KAM was a very fast DOS program, fast new graphics libraries and DOS drivers for new graphics cards became increasingly difficult to locate. This meant that John could not easily adapt KAM to take advantage of the ever-increasing speed and accelerated drawing functionality offered by the rapidly evolving graphics hardware systems. This also meant that I could never get my KAM programs to spit pixels fast enough to let my carpets fly in the way that I could with the hardware accelerated, character-based Vango. Another lingering side effect of this shift away from DOS is that I must still maintain seldom-used DOS computer systems just so that I can show my older work, much of which was tuned to a specific CPU.

There was also an architectural constraint built into KAM graphics because of its DOS roots. KAM could only draw on one graphics 'screen'. It was not possible to composite images nor was it possible to buffer images or processing of image components in the background. While a constrained creative system often drives its users to create well crafted and focused work, it also precludes many creative practices. In any case, in spite of our best efforts, the system had again become too isolating and unique. And again, it was time to move on.

4. Sketches for *Life by Analogy*

Fortunately for me, beginning in the late1990's John and I began to discuss how to translate the basic concepts of KAM, as well as portions of its precursors Vango and KMM, into the Windows GUI environment. As a result, and after an enormous amount of translation and new work on John's part, in 1999 he released the first version of SoftStep, a Windows-based, real-time, algorithmic MIDI composition and sequencing system.[4] In retrospect such a move seems obvious, but at the time the speed penalty exacted by the GUI itself was extremely significant. Even with the faster computers, newer graphics libraries, and accelerated graphics cards, it was very difficult to make a Windows system run as fast as the DOS-based KAM system; particularly when you remember that KAM was a real-time system. In fact, at that time it was so difficult to produce acceptable simultaneous real-time MIDI and graphics performance on an affordable PC that SoftStep did not support graphics when released. However, as SoftStep matured, processors, memory and graphics hardware got much faster, as did Windows itself. Accordingly, in 2001 John began to add graphics functions to SoftStep, specifically in the form of graphics libraries accessed through user programmable function modules.

Today, SoftStep provides a relatively metaphor-free direct manipulation GUI environment that enables the artist to do many things including:

- Create complex real-time program structures that can drive and manipulate MIDI sound, generative and sampled raster graphics, text, data, and various external processes.
- Explore unusual time structures and relationships by driving individual program processes with independent clocks.
- Build interactive real-time control surfaces that simplify the process of creating algorithmic music, imagery, and environments.
- Easily program specialized and reusable user function modules.
- Easily create custom scales and other data structures.
- Save snapshots of the system then return to specific states of the aesthetic process.
- Import numeric, text and genetic data sets using the DataBin and BioEditor.

- Communicate with micro-controllers like the Basic Stamp and EZ-IO.

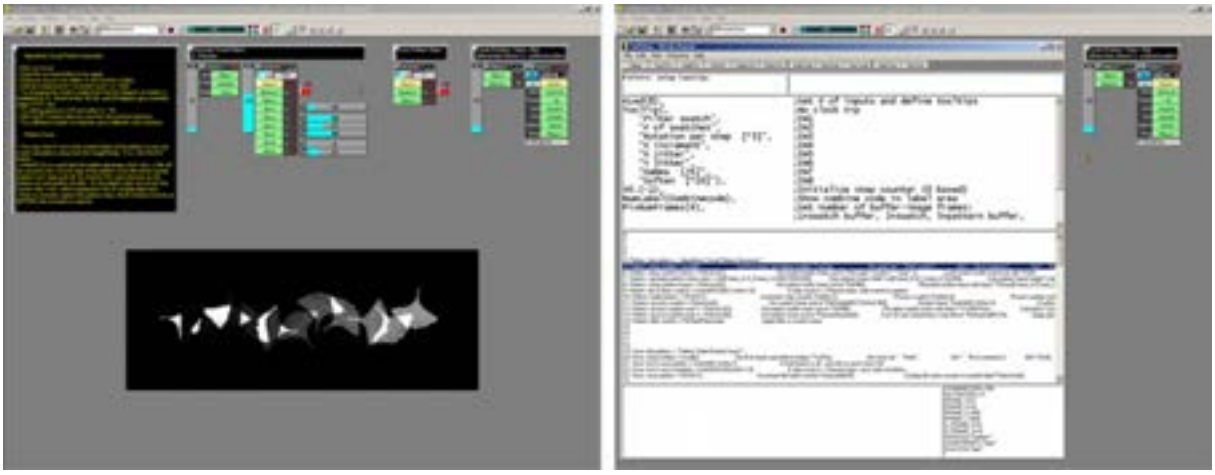

Figure 19. SoftStep code, animation frame and user function code examples

After working with SoftStep since its beta release, I have again developed a new set of generative processes. These processes add to those capabilities that carried over from previous systems such as system state snapshots and data-driven marks. A few of these new processes seem particularly worth noting in this discussion. To begin with, I am now able to use sampled graphics as part of my generative processes. For example, in addition to generating marks from simple patterns of interference or feedback or mathematical equations, I can now create marks by feeding sampled forms and captured picture elements to various convolution, combination and filtering processes. Having this new class of procedural marks greatly extends my expressive palette without requiring me to sacrifice my basic generative approach. I can also mix sampled representational images and generated marks at will, a capability analogous to easily mixing sampled and synthesized sounds. This ability has introduced me to new methods for generating and evolving complex time-based imagery.

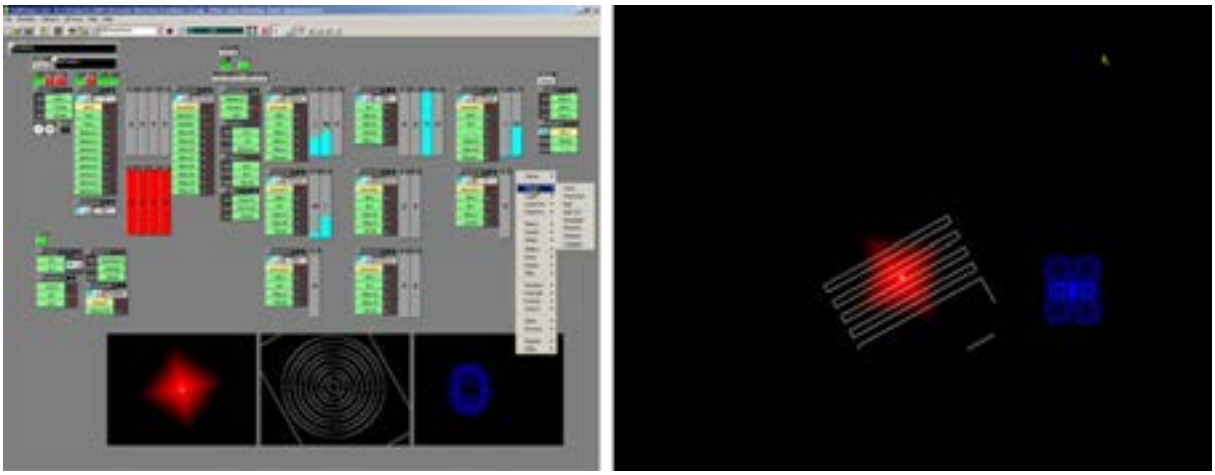

Figure 20. Sketches for *Life by Analogy*; SoftStep code, image components, and animation frame

SoftStep has also made it possible for me to continue and expand my experiments with time-enveloped marks, i.e. marks in which different visual elements or states of an entity are drawn at perceptibly different rates under program control. Time-enveloped marks are analogous to sonic notes that use ADSR envelopes, but visual marks usually run on a much slower time scale. Although I used this enveloping technique extensively when I used KAM, SoftStep's ability to manipulate high resolution sampled images using independent clocks gives me a whole new set of options to explore. I believe that manipulating the envelope structure of generative marks will prove to be a powerful way to develop interesting phenotypic variations or even new strains of an algorithmic artwork.

And finally, SoftStep has helped me to investigate a problem related to envelopes, i.e. how best to produce subtle and expressive generative instruments. A generative instrument is a set of processes and controllers that can be used in real-time to expressively create and modulate generative aesthetic systems. Examples of existing generative 'instruments' include: a brush with canvas, a keyboard with waveform synthesizer, a power carver with wood, a set of Buchla wands with MIDI lights, an Xacto knife with foam core, and a human-scale touch screen with mark synthesizer. However, none of these instruments easily supports the appropriate mixture of computer programming actions, visual imaging actions, sound and event composing actions, and expressive body actions that my working style demands.

SoftStep, on the other hand, begins to allow me to create new instruments by mapping input-process-output relationships in ways that the I/O device manufacturers and software library vendors never intended. I don't have to be a systems programmer to build these connections because John has provided me with an appropriate scale language that makes it relatively

easy. I can unite a physical interface to a micro-controller to a generative process running in SoftStep and output the results to audio-video projection systems while simultaneously controlling the installation environment. My instrument may not help me create a true virtual reality, but it will let me create a very expressive and immersive aesthetic experience while running on a PC an individual artist can afford.

Of course, having praised SoftStep's capabilities, it is only right to mention some of its shortcomings, two of which are especially challenging. First, because of various technical problems that arose from building a real-time system in the Windows environment, the graphics functions in SoftStep are not tightly integrated into the modular structure. As a result, I must use multiple programming paradigms to create my work. This makes the learning curve for programming SoftStep visuals much steeper and less intuitive than it could be. But more importantly, this lack of integration makes programming errors much more common and slows down the creative process. Mind you, the create process is still good, just not as good as it could be.

The second, and most serious problem for me with the original SoftStep architecture is the lack of a module encapsulation methodology. On the MIDI side, this problem is not as noticeable because John has already encapsulated so many algorithmic MIDI functions into the existing module structure. However, on the graphics side, while it is quite easy to build a chunk of imaging functionality and wrap it up for future reuse in the form of a user function module, it is basically impossible to encapsulate combinations of modules to make up a 'whole visual function'. This constraint makes it much more difficult to scale up the large mark and image structures associated with a complex artwork.

So, despite SoftStep's power and ease of use, it appears that it will soon be time again to move on to new system, one that has evolved to adapt to the latest needs of generative visual artists and musicians. And, of course, John has already developed the core of a new system designed precisely to meet these needs and more than a few future as-yet-unknown needs. A product of John and my decade-long conversation and John's three decades of programming experience, this new system will return to its KMM-KAM roots by providing visual artists with a very fast, highly scalable, direct manipulation meta-language. However, this language will run in real-time in Windows and be open and extensible, which will make a wide array of platforms, systems and libraries accessible to artists in unique ways.

I especially am excited by the following possibilities. First, I will be able to build very deep reusable hierarchies of marks, forms, and images, any part of which can be programmatically controlled. Second, I will be able to create my own personal appropriate scale language and create extensions to this language. That is, I will be able to build my personal language so that it embodies the level of abstraction that I personally need in order to be able to simultaneously work in the structured world of programming and the expressive world of fine arts imaging. However, I will not have to give up the ability to drill down into lower levels of abstraction when necessary, a sacrifice that many easy to use tools demand. Nor will other artists be forced to adopt my particular functional abstractions, they will be able to build their own. Not only will artists be able to build their own abstractions, they will be able to share and interoperate their functional thoughts with others. Moreover, I should be able to add new low-level functionality as it evolves.

Third, I am overjoyed that I may be able to create computer based generative systems using a language that evolves under me. In practical terms, that means that the system is being designed so that changes to the language interpreter do not require 'rewriting' user level code. This means that I can spend more of my time evolving my artworks and less of my time evolving myself to adapt to the changing systems.

Fourth, I look forward to the possibility that I may easily communicate with laboratory instruments, industrial I/O and control systems, and TCP/IP networks using my own personalized language. This is in addition to the existing MIDI and micro-controller based communications I can use to control lighting, sensing, and environmental systems.

Fifth, I anticipate that I can use the new language to drive other software systems, in essence treating them as specialized function libraries. This would allow me to develop my own non-standard ways to interact with standard systems. For example, 3D imaging, analysis and rapid prototyping systems are particularly suitable for this approach.

Before concluding, I should note here that other comparable languages for artists exist, such as the popular Max/MSP that runs on the Macintosh. However, for me personally, it is the ability to run in real-time on my existing PC's, the ability to interoperate with other PC applications, the ability to participate in the basic design of visual generative systems with John, and the fundamental elegance and openness of the system architecture itself, that keeps me coming back to John's systems to support my personal artistic evolution.

5. Conclusion

For more than a decade I have designed and used computerized algorithmic systems to produce artworks that incorporate generative and evolutionary concepts, forms and processes. When I first began this process, many felt that the major question was whether these generative processes could be used masterfully and whether the products should be recognized as Art.

My experience with the systems described above, and others, has convinced me that the answer to that question is obvious. Given enough time and energy, any process and product can be mastered and can become Art. The corollary is also obvious. Whether or not a creative work or process is recognized as Art at any particular time and place depends on social and historical forces that have little necessary relation to the original creative acts. Artists have no real choice but to just go ahead and explore what seems to them important. In fact, this question proved much too narrow to be helpful.

Instead, I have since found other questions to be more important. For example, I think it is important to ask:

- What types of new ideas can and should artists engage within the creative process and to what extent should these ideas be allowed to transform the creative process?
- What role can the individual human artist play in a cultural economy based upon industrialized generative processes and non-human systems?
- How can artists integrate standardized scientific languages and algorithmic processes into personal visions and expressive languages?
- How can artists capture their personal creative processes and encapsulate these processes in industry standard systems and software; and should they do so?
- How might the generative systems and products created by human and non-human artists function and evolve in the larger social context?

To confront these questions, but not necessarily answer them individually or completely, let me conclude this paper by proposing two conjectures based on my experience with the generative systems I described above.

Conjecture 1. Artists should focus attention on the structure of genetic and memetic systems evolving in time, the purpose being to make the behavior of these systems sensible to themselves and others. By extension, artists should help create and use generative systems that support this artistic focus but that also help existing creative processes adapt to the new circumstances.

I suggest this idea because I assume that human beings have become masters of large-scale industrial production, producing goods, knowledge, life forms, and even new production processes using large-scale social, scientific, and technical systems. I further assume that this productive activity exists in time and has significant impact upon human lives over time. And finally, I assume that some of the new core technologies of this productive onslaught will be genetics and memetics coupled to computer science.

If this is true, I believe it is essential that individual artists produce work by experimenting with simple computerized genetic and memetic systems and other ideas emanating from the world of science. Simple systems let artists see the essence of new forms and dynamics without inhibiting imagination. Sense based systems, ones that incorporate sight, sound, touch, and motion for instance, bring into play the enormous pattern recognition abilities of the human body and stimulate the human imagination. This is important because, as I believe Einstein said, “Imagination is more important than knowledge.”

To support their imaginations, artists should continue to rely in part on the power of a well-tuned intuition. Intuition, or pattern-knowledge, has served many human beings well, including both artists and scientists. However, all creative people should be aware of the limits of validity and functional limitations of any form of knowledge, whether it is subjective or objective, personal or public, intuitive or scientific. I believe it is a lack of clarity on this last point that forces many artists and scientists to misunderstand their own roles and contributions to knowledge and to society.

Artists should also use generative systems that foster experimentation with various time structures because varying the relative timing of the underlying generative processes often produces form variants and varying the timing of sense impressions often changes the meaning or implications of the experience. Stated another way, phenotypic expressions are often time sensitive. These kinds of tools can help artists and viewers develop sophisticated

structural and temporal intuitions and surface the new ideas that individuals and societies need to navigate into the future.

Conjecture 2. Since computers capture action in the form of language, a running program and its side effects can be considered the fundamental mark of a computerized generative system.[5] Therefore, artists should work with computer language based creative systems that enable them to directly handle these fundamental marks. Using these systems will also help artists better understand the structure, behavior, and creative potential of reified language, i.e. computer software. However, in the process of working with these systems, artists must also be careful lest their works quickly become extinct due to the rapidly changing computing ecology.

I make this proposal because each creative system is slightly different. Tools are slightly different from languages, clocked events are somewhat different from timelined events, genetic transparency is a little different than optical transparency, and computer marks are different from hand made marks. By implication, the insights each type of system provides and the products it produces are slightly different and effect humans in slightly different ways. Since exploring the subtle differences in life through word, image and music is the traditional purview of the artist, it seems clear to me that artists can naturally adapt to these new generative processes as long as they are empowered to the adjust the processes to their own needs while preserving the essence of the new relationships.

Furthermore, I believe it is very important that artists help explore and articulate these differences between systems. Subtle differences become the small changes that can make the big difference in the long run of cultural evolution. They are the butterflies in the weather pattern of global culture change. It is one of the artist's important social functions to catch, identify, and display these butterflies for all to view.

However, once artists capture the butterfly by creating artworks using the new systems, they should carefully consider how to preserve and share their processes and products. As an artist who has spent countless hours trying to recreate particular computer ecologies so I can show particular works that embody particular sets of non-verbal relationships that cannot be properly communicated in any other way, I can assure you that this is a big issue. There is an enormous difference between the documentation of an idea and the experience of the situation that generated the idea. There is an enormous difference between seeing something fixed into

a photograph and seeing the dynamics of that thing. There is an enormous difference between a genotypic abstraction and the phenotypic experience.

If artists actually hope to show their works to future generations, or even a few years from now, they must carefully conserve the entire generative system that they used to produce and show the work. In addition, they must meticulously document the work for themselves so they can remember how the work functioned and what its technical requirement were. Producing the illustrations for this paper forcefully reminding me of this reality.

6. References

[1] CD: *Magic Carpet Music*, 1. for *Tree of Life*: Alpha/Beta/Folding in Proteins, 2. for *Dark Matter*: DNA of HIV #7. by John Dunn

[2] Each ASCII character in a DOS-based PC has three attributes: a character(C) index, a foreground(F) color index, and a background(B) color index. In Vango each character's CFB indexes can be accessed independently.

[3] CD: *Algorithmic Music From DNA*, including HIV DNA #11, HIV DNA #39, HIV DNA #110. by John Dunn

[4] *SoftStep 3.0* by John Dunn. Available from Algorithmic Arts at: http://www.algoart.com

[5] From *Conjectures on Space*, by Jamy Sheridan and Peter Anders, published in *Minds, Machines, and Electronic Culture*, the proceeding of The Seventh Biennial Symposium on Arts and Technology at the Center for Arts and Technology at Connecticut College, New London, CT, March 1999.