Graphical GROOVE: memorial for the VAMPIRE, a visual music system

LAURIE SPIEGEL

E-mail: spiegel@dorsai.org

Once upon a time there was a computer music system called GROOVE (Generating Realtime Operations On Voltage-controlled Equipment, Bell Telephone Laboratories, Murray Hill, New Jersey) which outputted in the realm of sound, and was a wonderful and still-unique tool for the composition thereof. At that time a then-young composer who was using GROOVE for music got the harebrained idea that if she made a few minor changes here and there she could use it to compose images as well. This she did in 1974–6, and though the untimely demise of the system prevented creation of much documentation in the form of aesthetic works of its output, the system did function sufficiently to make some description worthwhile. While it is true that the mid-1960s DDP-224 computer on which GROOVE became a VAMPIRE (Video And Music Program for Interactive Realtime Exploration/Experimentation) was a massive room-sized computer, it has by now long been eclipsed in power by the constantly improving home computer. It is worth describing the concepts involved in part because there are by now many small computers capable of emulating its musical methods. Besides, I had a deep personal relationship with that computer, and wish to commemorate it. Here then follows the tale of Graphical GROOVE, aka the VAMPIRE.

1. WHAT WAS GROOVE, ANYWAY?

Before describing its visual applications, it may help to try to visualise the GROOVE system in its original form. It was a hybrid (digital–analog) computer music system, as developed by Max Mathews, Dick Moore and colleagues (Mathews 1963, Mathews and Moore 1970, Mathews, Moore and Risset 1974). The principle was both simple and general. A number of input devices (knobs, push buttons, a small organ keyboard, a 3D joystick, an alphanumeric keyboard, card reader, several console and toggle switches) and a number of output devices (fourteen digital-to-analog converters used for control voltages, thirty-six computer-controlled relays, a thermal printer, and two washing-machine-sized 1 MB hard disks) were connected to a room-sized 24 bit DDP-224 computer programmable by its users in FORTRAN IV and DAP 24 bit assembly language. Also accessible (as subroutines residing in FORTRAN IV libraries) were what might be called 'soft' or 'virtual' input devices (random number generators, attack–decay interpolators, and a sophisticated periodic function generator) and output devices (storage buffers, including arrays for logical switches and data of different types).

Users would write their own 'user programs' for their own purposes, to specify interconnections between inputs and outputs including the above-mentioned units. Since these connections could be complex transfer functions consisting of any kind of process the user could code up, this made the system ideal for the development of what we called 'intelligent instruments'. (These are essentially musical instruments for which the ratio of the amount of information the music system generates to the amount the musician plays, per unit of time, is greater than one to one.) It also made the system ideal for the exploration of compositional algorithms.

Aside from the creative freedoms and temptations inherent in the responsibility of each user to program their own 'patch', GROOVE had another cataclysmically important characteristic as a music system. It viewed everything as (perceptually) continuous functions of time. It did not think in terms of notes or other such 'events'. Instead, it required such entities to be programmed as processes or sampled as curves of change over time. The control sampling rate was 100 Hz, which meant that all perceptible control parameters were updated quickly enough to sound to the human ear like smooth, rather than stepwise, changes.

During each sample, the user program would be looped through once, all inputs referenced would be read, all programmed computations made, and the output channelled to DACs (digital-to-analog convertors) and disk files which could be edited later. An editing program would be just another interactive user program which each of us would write for a specific kind of modification of a particular work. The data being edited would be the stored time functions on disk instead of the live data coming into the computer from the various input devices. Editing programs could use all of the same devices (knobs, periodic function generators, etc.) that might be used in recording a first pass, and editing was often a real-time performance process in itself.

https://doi.org/10.1017/S1355771898003021 Published online by Cambridge University Press

The centrepiece of the design was the bank of 200 functions of time. All data was stored as a series of numbers that had no specific association with any parameter of sound or of musical composition, except what a user program might give it by connecting these numbers to a relay or DAC. The system allowed the composition of functions of time in the abstract.

The importance of being able to approach all parameters of sound, of composition, or of performance as perceptually continuous functions of time cannot be overstressed during this current period when music seems everywhere to be digitally described as entities called 'notes', and in which there are generally conceived to be differing necessary rates of change for different musical parameters. In our modern post-MIDI world, pitch is seen as changing at a rate of once per note whereas amplitude is updated 'continuously' at higher resolution (faster rate). GROOVE embodied a concept space shared with the old truly modular analog synthesizers of the 1960s, on which any pattern of temporal change could be applied to any parameter, and in which sound can really be treated as a multidimensional continuous phenomenon. GROOVE added to this the ability to use time functions computationally without direct connection to any input or output variable.

The software was able to handle several times as many of these simultaneous time functions as we had hardware DACs to use them on (200 functions for fourteen DACs). We therefore had many spare functions available to use for variables of any level of abstraction we might want, from recording actual knob or switch settings as we improvised or interpreted stored music, to global and profound compositional parameters such as probabilities, densities or entropy curves.

Each of us used these time functions in our own ways. In fact each of us freely used the entire system in entirely our own way because we each had an entire copy of our own with full source code. We could each change anything we wanted.

2. TEMPTATION

We often enjoyed just playing around with the system. The rate at which the computer ran through the user program loop, reading its inputs and writing to its outputs, was controlled by an external analog oscillator in this early hybrid system. So of course, we tried plugging in a voltage controlled oscillator so you could compose a time function which would create tempo changes by changing the sampling frequency of the computer itself. At one point Emmanuel Ghent had the computer control the speed of a variable-speed reel-to-reel tape recorder so that he could specifically compose pitch changes with the oscillations of a bank of fixed-frequency resonant filters he had built. In general, there were a lot of interconnections between the digital and analog domains and we played with them quite a bit.

This was often difficult because the analog audio hardware and the digital computer hardware were in separate labs at a cumbersome distance from each other, connected by several hundred yards of trunk cables. We all made many trips back and forth between the analog and digital ends of GROOVE to calibrate DAC output voltages or to change the configuration of the multicoloured spaghetti. (A typical patch consisted of hundreds of cables on a removable patch matrix board that each user could slide into a card rack full of audio modules too miscellaneous to describe here, so that each user could pursue a hardware configuration unlike anyone else's.)

During such trips down that long, long hall between the analog and digital labs, when not impatiently obsessed with an embarrassing desire for roller skates to shorten the long walk (in the era before roller skates were O.K. for grown-ups too), I began stopping to look through a glass window in a door to another computer room along the way. Strange abstract shapes could usually be seen evolving on a video monitor, growing and evolving, week after week, month after month. Eventually I got to know Dr Kenneth Knowlton, the computer graphics pioneer and master of evolutionary algorithms (Knowlton 1964, 1970, 1971), and we began to work together on various projects. After learning some graphics coding there, I became intrigued with the idea of trying to make musical structure visible and embarked on the strange mission of bringing GROOVE's compositional capabilities to bear on the frame buffer output, particularly the ideas of time functions, transfer functions, and interconnectable software modules.

This was back in the early 1970s, before digital synthesis of sound could be done in real time (computed at the speed that we hear it). In that era, apart from a hybrid system such as GROOVE, computer music could only be made noninteractively, by entering defining information into a computer, waiting for sounds to be computed, then retrieving, recording and listening to them later. Ken, working with film-maker Lillian Schwartz, was working in a similarly nonrealtime way, running image generation software all night, using a program that would compute a single frame of film, then open and close the lens of a computer-controlled film camera to expose and then advance the film.

I reasoned that just as GROOVE's computer control of analog modules had made interaction with relatively complex logic systems a realtime process, permitting realtime interactive computer control of musical materials for the first time, realtime interactive computer graphics should be possible as well by similar means. Instead of recording the image on film frame by frame, I should be able to code myself a visual musical instrument that would let me play and compose image pieces by recording the control data as time functions and playing back the time functions as visual compositions.

The idea of getting GROOVE running on this second computer in a different lab down the hall where it would output to a video monitor instead of to banks of equipment in an analog audio lab did not come to me all at once. Initially, I merely succumbed to the irresistible temptation of glowing colour and texture and movement and light, but for the next several years, I spent probably as much time working on this visual music system as on audible music, and realtime interaction with video images felt like playing music to me. The desire to compose music visually was an inevitable craving.

3. RTV (REALTIME VIDEO)

What later became VAMPIRE started relatively simply, as a program called RTV (RealTime Video). That, in turn, started even more simply, as a mere drawing program for creating still images. Using a routine that Ken Knowlton gave me which permitted me to address the 'frame buffer' (I believe this was just a dedicated area of memory in computer CORE) and a Rand Tablet, I wrote myself a 'drawing' program (similar to what we now call 'paint' programs, but in 1974 there was no terminology for this yet) and greatly enjoyed doing a long ongoing series of computer drawings, evolving and changing the way the drawing program elaborated an image from my motions over the Rand Tablet as time went on.

Ken and I also worked out an elaborate initialisation routine for an array of sixty-four definable, storable bitmapped textures which could be used as 'brushes' or letters of the alphabet, or whatever, and

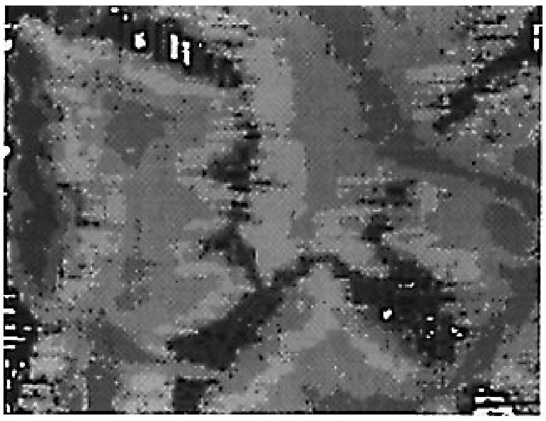

Figure 2

which made use of a box with ten columns of twelve push buttons each representing a bit that could be on or off, functioning as a means of entering these patterns. After consulting some of my old hand weaving books, I made a large deck of Hollerith cards, and shuffled them in different ways to be able to easily enter batches of patterns via the computer's card reader. (Ken did some truly amazing things with that ten by twelve button box that are beyond the scope of this article but nonetheless worth mentioning, such as projecting completely customisable virtual control surfaces for telephone-related jobs onto a half-silvered mirror above it (Knowlton 1977). But that's another story.)

As a composer of music, I soon found that I enjoyed playing the drawing parameters in real time like a musical instrument. I could move around in an image and change the size, colour, texture and other parameters in real time as I drew, using knobs and switches just like those on the GROOVE music computer down the hall. I would draw with one hand while manipulating the various visual parameters with my other hand using the 3D joystick, switches, push buttons and knobs.

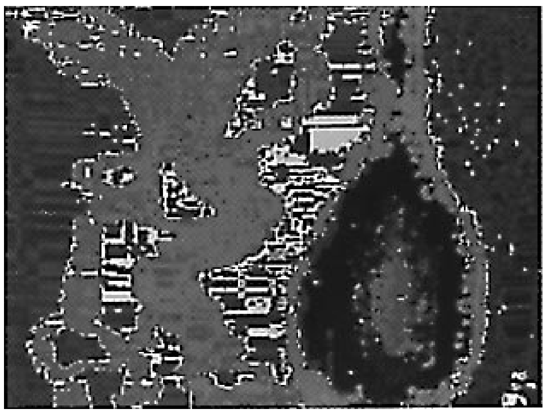

Figure 1

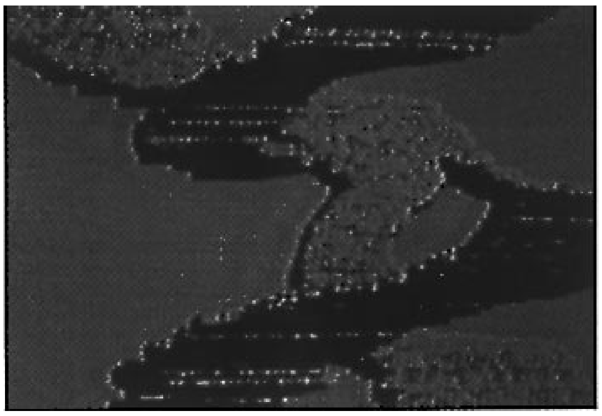

Figure 3

The movements of the object I dragged around the screen felt melodic, and I realised that I was not satisfied with just one 'melodic' line. In audible music I had loved counterpoint best, so I wrote in another realtime interactive device to play. It was a square box of sixteen push buttons for standard musical contrapuntal options. By now it was possible to interact with quite a number of visible variables in real time.

This was before 'menu-driven' human interface systems came into fashion. Even had it not been, however, I have always preferred random access parallel (equally reachable-for) controls to any kind of hierarchical or modal way of organising such a group of controls. The interfaces in traditional acoustic musical instruments are generally of parallel random access design. It may be more hardware intensive, but spontaneously grabbable controls are better for the music and art.

The simultaneous parallel inputs I had written into the system at this point, before interfacing it with the GROOVE music system, are as described below. This was when it was still just an unrecordable room-sized live-performance visual instrument.

- Rand Tablet:
  - x and y location being drawn
- Foot pedal:
  - Enable or disable drawing (writing to the display)
- Knobs:
  - Logical operation (write, and, or, xor)
  - Vertical size
  - Horizontal size
  - Colour number $1 \rightarrow 8$ for foreground
  - Colour number of background
  - Global colour parameters
    - Colour definition mode 1
      - Saturation
      - Value
      - Hue
      - Resolution of hue spectrum
      - 3D joystick for path through colour space of colour indices $1 \rightarrow 8$
    - Colour definition mode 2
      - 3D joystick axes
        - x = amplitude of green
        - y = amplitude of red
        - z = amplitude of blue
- Push buttons – contrapuntal options:
  - Single line (single sequence of time sequenced x–y locations, as drawn)
    - Contrary motions
      - Reflection on x axis added
      - Reflection on y axis added
      - Reflection on both axes
      - Sine–cosine relationship of 2 points moving
    - Parallel motions
      - Single line
      - Two parallel lines (for all above)
      - Three parallel lines
    - Oblique motions
      - Freeze x on reflection of drawn path
      - Freeze y on reflection of drawn path
    - Other push buttons
      - Clear screen to selected colour
      - Visible cursor by continually xor-ing twice
    - 10×12 button box – textures
      - Definition mode (not realtime)
      - Selection mode (textures 1 through 64, in real time)

4. VAMPIRE (THE VIDEO AND MUSIC PLAYING INTERACTIVE REALTIME EXPERIMENT)

With that many parameters to control in real time, I had arrived at the same difficult stage in visual improvisation at which I had found myself in audible music several years earlier, when I needed to switch over from improvising to composing. The capabilities available to me had become more than I could sensitively and intelligently control in real time in one pass to anywhere near the limits of what I felt was their aesthetic potential.

During this period I had increasingly come to compose my sonic music by writing algorithmic descriptions of logical processes with which I could interact in real time. The use of such processes within the GROOVE system allowed me to control far more parameters at once than did traditional mechanisms of musical performance. For example, instead of using a knob to change the pitch of a sound, I could use the knob to change the informational entropy (Shannon 1948, Pierce 1961) of an ongoing process that was selecting pitches for a whole texture of sounds. That GROOVE was designed for the composition and manipulation of functions of time in the abstract allowed me to store and later refine or reuse not only the knob's actual state and the entropy value calculated from it, but also the specific pitch content for which these values were ultimately used. All of these could subsequently be altered by editing or used for additional musical purposes.

Because I had been performing, improvising and composing in the sonic domain by realtime interaction with increasingly abstract and powerful sets of variables, the idea of creating and controlling algorithmically described visual processes in similar ways was irresistible. It made sense that similar processes in the visual domain could be incorporated into the same kind of general framework that GROOVE had provided for sound. Visual data could also be dealt with as abstract curves of change during time in all the same ways that sonic data could. An interface of the drawing system with GROOVE's time-function-oriented software and its libraries of compositional subroutines became inevitable. GROOVE's architecture would be able to support the composition of a single extended set of time functions that could ultimately be transduced into audio or video or both. Each time-varying component within such a single integrated set of processes could be interchangeably manifested in the human sensory world as amplitude, pitch, stereo sound placement, etc., or as image size, location, colour or texture, etc., or (conceivably, ultimately) in both sensory modalities at once. Within such a system, a temporal work could be represented as a single integrated supersensory structure or an ongoing process, or both.

Furthermore, such an audiovisual system would permit not only realtime interactive generation, storage and replay, but also the ability to edit, refine, append, overdub, filter and otherwise postprocess all composed time functions together, in relation to each other. Auditory and visual data could be directly interconnected with each other, interacting, interchangeable and interdependent. Despite the previous separation of these media, image and sound could now be simultaneously composed and synchronously cogenerated.

In a sense, I had begun to conceive of music not specifically as a sonic art, but as the art of composing abstract patterns of change within time. Such composed functions of time could be modulated variously onto either sonic or visual parameters as their carrier media. In essence, the digital computer had begun functioning like a Rosetta stone, allowing unprecedented ease of translation between the various sensory modalities that distinguish our time-based arts, and also permitting an individual's artistic skills and aesthetic understanding in either medium to be applied to both.

In fact, before its untimely demise, this system did ultimately provide an instrument for composing audiovisual phenomena as abstract changes over time. It did record human input into a computer via an array of devices for which the interpretation and use of each device was programmable. The data from such processes were able to be stored, replayed, reinterpreted and reused, and additional layers of data could be subsequently superimposed. Any time functions created could be further altered by any transformation programmable in FORTRAN and the result could actually be used to control any parameter of image or of sound, but unfortunately this was possible only by transferring the data back to GROOVE's audio-interfaced home computer by hand. Due to the use of separate computers in separate laboratories for visual and for acoustic research, it was not physically possible at that time to use a single set of time functions to control both image and sound simultaneously, though in principle this should have been possible and in spirit it had been the goal of much of my work on this project.

Like any other vampire, this one consistently got most of its nourishment out of me in the middle of the night, especially just before dawn. It did so from 1974 through 1979, at which time its CORE was dismantled, which was the digital equivalent of having a stake driven through its art.

REFERENCES

Knowlton, K. C. 1964. A computer technique for producing animated movies. *Am. Fed. Infor. Proc. Soc. (AFIPS) Conf. Proc.*, vol. 25, pp. 67–87.

Knowlton, K. C. 1970. EXPLOR – a generator of images from EXplicit Patterns, Local Operations and Randomness. *Proc. of the 9th Annual UAIDE Meeting*, pp. 544–83. Miami Beach.

Knowlton, K. C. 1971. TARPS – a Two-dimensional Alphanumeric Raster Picture System. *Proc. of the 10th Annual UAIDE Meeting*, Los Angeles.

Knowlton, K. C. 1977. Computer displays optically superimposed on input devices. *Bell System Technical Journal* **56**(3): 367–83.

Mathews, M. V. 1963. The digital computer as a musical instrument. *Science* **142**(3592): 553–7. Alternative version: 1972. The computer as a musical instrument. *Computer Decisions*, February issue: 22–5.

Mathews, M. V., and Moore, F. R. 1970. Groove, a program for composing, storing, and editing functions of time. *Communications of the Association for Computing Machinery* **13**(12): 715–21.

Mathews, M. V., Moore, F. R., and Risset, J. C. 1974. Computers and future music. *Science* **183** (January): 263–8.

Pierce, J. R. 1961. *Symbols, Signals and Noise: The Nature and Process of Communication*. New York: Harper and Brothers. Reprinted unabridged and revised as *An Introduction to Information Theory: Symbols, Signals and Noise*. New York: Dover Publications, 1980.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal*. Reprinted in Shannon, C. E., and Weaver, W. *Mathematical Theory of Communication* (University of Illinois Press, 1949).