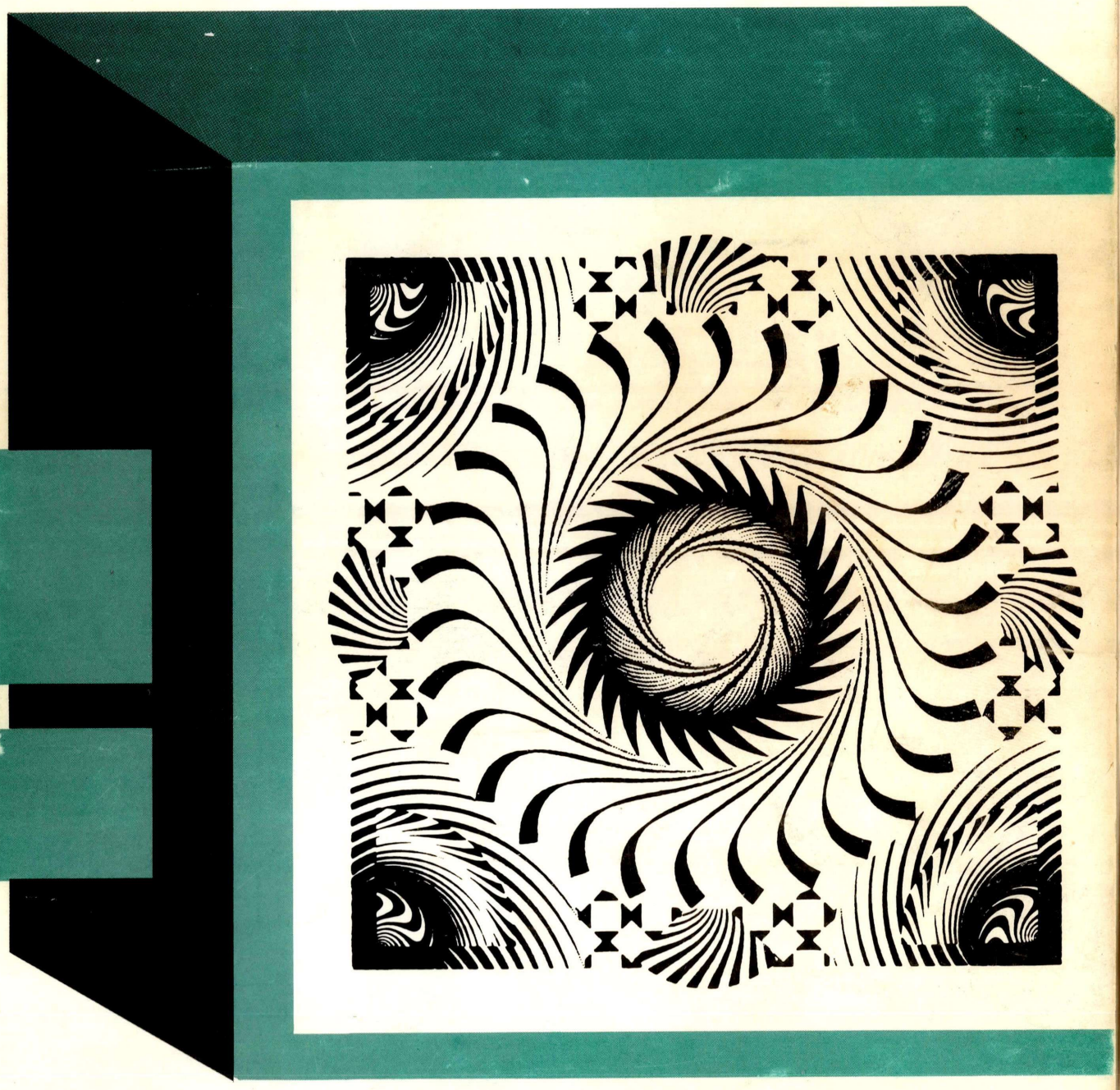
PROCEEDINGS

7th symposium on small computers in the arts

OCTOBER 8-11, 1987
PHILADELPHIA, PENNSYLVANIA

SPONSORED BY:
SMALL COMPUTERS IN THE ARTS NETWORK, INC.
COMPUTERS SOCIETY OF THE IEEE
PHILADELPHIA AREA COMPUTER SOCIETY



ISSN 0748-2043

KOLOMYJEC

1987
Proceedings of the Seventh Annual Symposium on Small Computers in the Arts

October 8-11, 1987

Sponsored by:

Small Computers in the Arts Network, Inc.
IEEE Computer Society
Philadelphia Area Computer Society

Organized and Produced by:

Small Computers in the Arts Network, Inc.

The papers appearing in this book comprise the proceedings of the Symposium on Small Computers in the Arts. They reflect the author's opinions and are published as presented and without change in the interest of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors, officers, or board members of the Small Computers in the Arts Network, Inc.

Symposium Chairman: Dick Moberg

Symposium Organizing Committee: Alan Cosgrove, Trip Denton, Carol Flaherty, Nancy Kimmons, Perry Leopold, Bill Mauchly, Dan McCormack, Des McLean, Dick Moberg, Rob Morris, Tom Porett, Tom Rudolph, Mark Scott, Anne Seidman, John Senior, Jennifer Wolf.

Art Show Committee: Alan Cosgrove, Dan McCormack, Des McLean, Anne Seidman, Jennifer Wolf.

Proceedings: Editors, Mark Scott and Dick Moberg; Production, Robin Snyder and Jennifer Wolf.

SCAN Staff: Robin Snyder

Published by
Small Computers in the Arts Network, Inc.
Box 1954, Philadelphia, PA 19105

ISSN 0748-2043

Cover Art: Christa Schubert, Untitled, © 1985.

The SCAN Story

The Small Computers in the Arts Network (SCAN) originated from a computer music concert held in 1978 in Philadelphia as part of a Personal Computer show and sponsored by the Philadelphia Area Computer Society, one of our sponsors this year. The concert was so successful that its organizers have repeated the event every year. It is now the Annual Philadelphia Computer Music Concert, presently in its tenth year. We realized at that first concert that the audience was not there because of the high quality music (it wasn't), but because of the novelty of the techniques involved in producing the sounds. Since some people didn't quite understand what they were listening to, during the second year of the concert, lectures on computer music were added for those interested in the technology inovlved. Another realization at that first concert was, what does the audience look at during the performance? Normaly an audience follows the movements of a musician or singer on stage....but here was a little box sitting on a front table with, at best, a few blinking lights. So, for the next year's meeting, we added talks on computer-generated graphics and video and, since then, have incorporated them into the evening concert.

Interest continued to grow and, in 1981, the organizers decided to hold their own three-day event called the Symposium on Small Computers in the Arts which consisted of lectures, demonstrations, exhibits, and workshops. The subject matter was expanded to include all the arts and included some talks on dance and sculpture. The event was co-sponsored by the Institute of Electrical and Electronic Engineers (IEEE) Computer Society which published the Proceedings of the event. To keep in touch with interested persons the organizers formed an informal non-profit, group, the Personal Computer Arts Group and published a small, irregular newsletter. As interest grew further, it became clear that the group could better serve its purpose by incorporating as a non-profit, educational organization. This led to the formation of the Small Computers in the Arts Network, Inc., or SCAN, in August of 1985 and the obtaining of its tax-exempt status in 1987.

Today SCAN is an organization which provides information for those interested in using computers and related technology in the arts. This includes music composition, music synthesis, music performance, graphics design, animation, video, textile design, sculpture, dance, writing, and other aspects of the arts. SCAN performs its services partly by means of a 100-page resource guide to the field published twice a year entitled Small Computers in the Arts News (also called SCAN). Current subscribers are located across the country, in Canada, and in several European countries. In addition, the annual Symposium on Small Computers in the Arts is organized by SCAN and provides a forum for those working in the field to present new works, developments, and techniques. Courses and workshops are offered at both beginning and advanced levels. Demonstrations, film and video shows, a computer art gallery, and computer music/video performances are also part of the Symposium.

An important service which SCAN performs is to provide information for educators who have been charged with setting up computer-based music or graphics programs in their respective schools and institutioins. Approximately half of our Symposium attendees and newsletter subscribers are connected with educational institutions. A speakers and consultants bureau has recently been established by SCAN to connect those who have experience in the field to those who need it.

SCAN has operated for the past nine years as a volunteer organization with a minimal budget and has been entirely self-supporting. We have a mailing list and data base of over 1800 people across the country and internationally working in the arts with new technology or interested in learning about the field.

Introduction

We start this year's Proceedings with a paper by David Cook providing a concise and informative overview of current graphics hardware (sigh!) capabilities. Dialecticians will not be disappointed, we follow with Stella Russell's Critic's View of recently produced electronic images. Bruce Bubacz then proposes a widening of our view of an aesthetic to include (and criterion by which to judge) interactive on-screen activity. Leslie Darby warns that we not assume too much from advances towards desk-top publishing, and Frank Kulesa provides solid advice for commercial electronic artists in Presentation Graphics.

Applications of hardware and software are described in Kathleen Stumpf's academic architecture curriculum proposal, Vernon Reed's Cybernetic Jewelry, and Donna Cox's description of ICARE, a function-based color palette editor for hi-res graphics systems.

Algorithmic art-production (image creation by programming as opposed to paint system) is well represented by Bill Kolomyjec, and the paper by William Dudley and Marie Ozmon. One method of sparking the interest of artists in this area is shared by Duane Palyka in Visual Metaphors Used in Teaching 'C' to Artists.

With our three final graphics-related papers: Paul Rutkovsky's inventive Insect Technology, my attempt at partial exegesis, and a wonderful text/graphic work by Walter Wright; we find ourselves already inside the process and realize that everything written and designed with electronic assistance is subject to aesthetic critique. Has the medium always been the message?

Electronic music is represented as David Lawrence provides a thought-provoking essay from a classicist's point-of-view, and K.S. Gugel and R. Sudhakar discuss the digital representation and manipulation of a complex sound: the 'acoustic' piano note.

Mark W. Scott

Contents

PC TOOLS FOR ARTISTS

David Cook
Computer Graphics Consultant

6727 Leonard Road
Eastmanville, MI 49404

Abstract

This paper attempts to educate the reader in the capabilities and types of computer graphics hardware and software available today for the personal computer system. The user is given a brief tour of computer graphics history, and then introduced to hardware and software concepts including some of the problems associated with computer graphics. Finally, there is a discussion of types of available equipment and the proper places to purchase such equipment.

INTRODUCTION

The advent of the personal computer has placed advanced, yet fairly inexpensive tools in the hands of everyday people. Artists are finding the new tools to be useful for visualization and creation. However, picking the proper equipment and coping with the new terminology and technologies can prove to be a frustrating experience, even for the computer literate. This paper is a foundation for the artist to use in picking graphic tools. It covers terminology, computer graphics hardware, and the latest in display technology.

Caveat

The following discussion mentions specific products. I have choosen these paticular products because of my familiarity with them. This does not mean that these products are necessairily the best to buy. It is up to the reader to take the information presented in this paper, and apply it to **all** products before purchasing.

A Brief History

Graphic design was an early use of computers, first realized in 1950 at MIT. The computer's ability to perform powerful mathematical computations at a high rate of speed made it especially useful for science and statistics. However, even the earliest uses included art. Many pioneers in computer graphics art made their start by using the scientific systems after hours. The early output device of choice (and necessity) was the pen plotter. This simple device consisted of one or more colored pens attached to a corkscrew drive. Beneath the pen was a piece of paper attached to a drum. As the drum rotated, the pen could be instructed to move back and forth. In this way, the artist could render complex geometric designs. This was the beginning for the computer graphic artist.

HARDWARE TERMINOLOGY

The earliest systems were known as **vector** display systems. To draw a line, the user specified the coordinates for the start and end of the line. The display would then trace the space between the endpoints, making the line. The lines were clean and crisp, not '**jaggie**'. This was because the electron beam in the display was drawing the line, from endpoint to endpoint, and not approximating the line with individual points. However, three problems were soon discovered with this type of display. First, it was very difficult to edit the picture. One could not simply turn off portions of a line. Instead, the lines had to be recalculated and the result redisplayed. Thus, a long line with a section edited from the middle became two short lines. The second problem came with very complex drawings. Since the electron beam was tracing the image, as the image became more complex, the beam took longer to complete the display. This resulted in images becomming dimmer and dimmer as the complexity increased, to the point where the image would flicker. The third problem was trying to portray 'realistic' images. For example, it is very difficult to express a scene such as a

photograph of a lake, using lines, circles and other geometrics. The very subtle details, shadows and textures of real life could not be expressed via vector graphics. It was soon realized that this approach to hardware would not suffice for all applications.

To solve the problems associated with vector hardware, raster equipment became popular. Though vector systems are still used today (mostly in CAD, Computer Aided Design, areas), raster systems are almost always the display of choice. In a raster system, the image to be produced must be stored in computer memory. Memory which stores such images is called frame buffer, or bit plane memory.

Basically, raster displays work much like a television. In a television, the electron beam 'paints' the screen by starting at the upper left corner of the display, and moving horizontally. At the end of each line, the beam returns to the left side, and moves down one line (this may be somewhat different on interlaced displays, but for our argument, it is safe to assume this is what happens). At the end of the last line, the beam returns to the upper left corner. The amount of time that it takes the beam to move from the bottom of the display, back to the top, is called the vertical retrace. This is seen on home television sets as the black line which sometimes scrolls through the image. This black line is the amount of time that it takes the electron beam to move from the bottom back to the top (usually this is below the image on your screen and is invisible). To make the image seem continuous to your eyes, the electron beam must repaint the entire screen 60 times per second (30 times per second for interlaced displays).

The frame buffer contains the image to be displayed on the screen. Display hardware is designed to read the memory in synchronization with the location of the electron beam on the screen. The number stored at each position in memory indicates the intensity for the electron beam at the corresponding location on the display. These discrete positions in memory are termed pixels (for picture elements). Common television contains about 512 pixels horizontally, by 400 pixels vertically (displayable). If you wish to display more pixels, you will need more memory and better hardware capable of reading memory and displaying it fast enough. Current technology limits the effective resolution of display tubes to about 2000 pixels horizontally by 1500 pixels vertically.

So far, we have only looked at monochrome graphics. In a monochrome system, only black & white are displayable. The Apple Macintosh is such a system. On machines capable of more complex graphics, intermediated grey levels are specifiable. The memory requirements for a monochrome display are fairly easy to compute. If a display is only capable of showing BLACK and WHITE, then each bit in memory (either a 1 or 0) indicates the color for a particular pixel. Therefore, a 512 x 400 resolution display contains 204,800 individual pixels or bits. Since there are 8 bits in one byte of computer memory, this would equate to 25,600 bytes of memory (in computerese, this would be 25K, the 'K' means 'times 1024'). However, if we wish to display intensities between black and white (ie greys), we need more bits per pixel. A good system would display 256 shades of grey. The value ZERO (0) would correspond to BLACK, the value 255 would correspond to WHITE. A value of 128 would be a middle grey. Values greater than 128 would gradually move towards white, and values less than 128 would gradually move towards black. Representing the numbers between zero and 255 require 8 bits of memory (one byte). Therefore, if our monochrome system is to display a 256 level greyscale, one pixel will be one byte in size. Our display size, at 512 x 400, would then grow from 25,600 bytes of memory, to 8 times that number, or 204,800 bytes of memory (200K).

Black and white (or grey) images are nice, but can get boring quickly. What most artists desire is to design in COLOR. Color raster technology is as old as the color television industry. Before we look at color display systems, we should first understand how a raster display makes color. Basically, ALL colors on a color television are created by placing three colors together: red, green and blue. This scheme is known as the RGB system. Red, green and blue are the primary colors of the additive/transmissive light system (as opposed to subtractive/reflective light primaries of yellow, cyan and magenta with which artists are more aquainted). To create other colors, we combine various intensities of the red, green and blue colors. The following chart shows the secondary colors obtainable from combining equal amounts of primaries:

Red and Green	Make	Yellow
Red and Blue	Make	Magenta
Green and Blue	Make	Cyan

To create other colors, each of the three guns (RGB) will vary in their intensities. A good display system allows 256 shades of RED, 256 shades of

GREEN and 256 shades of BLUE. A value of ZERO (0) for the gun indicates that the gun is off, or not displayed. A value of 255 for a gun indicates the full (brightest) intensity of that gun. The value 128 indicates a median intensity. The following chart gives an example of RGB colors:

Red	Green	Blue	Resulting Color
255	0	0	RED
0	255	0	GREEN
0	0	255	BLUE
255	255	0	YELLOW
255	0	255	MAGENTA
0	255	255	CYAN
0	0	0	BLACK
255	255	255	WHITE
128	128	128	MEDIUM GREY
128	255	0	LIME
255	128	50	BRIGHT ORANGE
200	190	200	PINK
50	30	10	DARK BROWN

Since the three guns each contain 256 levels of intensity, the display is capable of showing 256 (R) x 256 (G) x 256 (B) colors, or a total of 16,777,216 colors (this would be 16 Meg, the 'Meg' indicates 'times 1,048,576').

The storage memory required for a display containing 204,800 pixels (512 x 400) with 16,777,216 colors available for each pixel would be a total of 614,400 bytes of memory. Each pixel would contain a byte of RED, a byte of GREEN and a byte of BLUE. As we increase the resolution, the memory requirements also increase. A display of 'acceptable' resolution for professional work would be around 1024 x 768 pixels. This display contains 786,432 pixels or a total of 2,359,296 bytes of memory (2.25 Meg). Thank goodness memory is cheap!

A word should be mentioned about the display processor itself. This remarkable device is responsible for reading memory and converting each red, green and blue byte into the proper signal at the proper time. In the case of a 1024 x 768 resolution full color display, the entire 2,359,296 bytes of memory must be scanned and converted to a video analog signal 60 times per second. This is no trivial task for in one second of time, the processor must convert a total of 141,557,760 bytes of memory. To aid in this conversion, frame buffer memory is often composed of special, more costly memory chips. The chips have a much faster access time than normal computer memory. Memory of this type is often termed **VRAM**, which stands for Video Random Access Memory.

Many vector and raster systems are what is called **double buffered**. This term simply means that there are two frame buffers. When one is displayed, the other is not. The ability exists for the display processor to switch between the displayed and non-displayed frames. There are two major reasons for double buffering. First, with a double buffered system, it is possible to give the user an **undo** capability. Undo allows the user to reverse the last action performed on the frame buffer. For example, if the user draws a red line onto the display, and then a blue line, the blue line could be removed if it was a mistake. To accomplish this, the display processor (or computer hardware) copies the displayed frame into the second frame between each operation. If, after the operation is finished (such as drawing the blue line), the operation was deemed improper, the second frame is copied back to the original frame and the line is in effective erased.

The second use for double buffering is in animation. For example, to show an animation of a ball bouncing, the ball must be drawn, erased, redrawn at the next position, erased, redrawn at the next position, etc... However, the action of erasing and redrawing is visible to the user, and appears as a flicker, or **windowshading**. To overcome this, the computer only draws into the frame buffer which is not currently being viewed. After the draw is complete, the frame which is being viewed is switched with the frame which was not being viewed and the process continues. In this way, the computer only draws into the non-viewed frame, and the user never sees the drawing or erasing take place.

Finally, there are two types of frame buffers: **color mapped** frame buffers, and **bit mapped** frame buffers. We have already dealt with bit mapped frame buffers, which can display an extreme number of colors on the screen at one time. Color mapped systems were devised to make computer graphics more affordable in applications not demanding lots of colors. In cases where the user only needs 256 or 4096 different colors on the screen, color mapped buffers are more advantageous. Instead of writing the actual RGB color into each pixel, a number is written which represents a **index** into a **color map**. For example, in a system which has a maximum of 256 colors displayable at one time, out of a palette of 16 million colors, each pixel is one byte in size (as opposed to the three bytes necessary for the full color system). This byte represents values between zero and 255. The **color map** is another area of memory

seperate from the frame buffer. It contains 256 x 3 bytes. Each set of three bytes indicates a true RGB color. Therefore, there are 256 RGB colors in the color map. As the display processor reads the frame buffer, each pixel (byte value) found is looked up in the color map and the corresponding RGB value is output.

A major benefit of this type of display, besides lower memory cost, is the fact that by changing the color map RGB values, ALL pixels on the screen written with the color map entry are instantly changed. For example, if 2000 of the pixels are written with the value 100, indicating color map entry #100, and entry #100 is associated with a RGB value of 255, 255, 0, then those 2000 pixels are YELLOW. If, however, we suddenly change color map entry #100 to be a RGB of 255, 0, 255, then all 2000 pixels written with a value of 100 are instantly changed to magenta. This allows a very simple (and cheap) form of animation known as **color map animation**, or **color cycle animation**. To continue our bouncing ball example, a ball could be drawn in 10 or 12 positions of the bounce, each ball being drawn with a different color map entry. Each color map entry RGB value is set to BLACK except the color or the first ball (usually the first entry in the color map). To make the ball bounce, the computer simply makes the first ball black, and turns on the color of the next ball (for the sake convenience and efficiency, this is usually the next color map entry). Since this occurs at a very fast rate of speed, the ball will appear to bounce in realtime. Although this is fairly powerful, it is limited in the complexity and quality of the finished animation sequence.

SOFTWARE TERMINOLOGY

Just as there is vector and raster hardware, there is vector and raster software. Software which is vector oriented is geometry based and is often termed **object oriented**. This term stems from the fact that all images are defined as objects such as a **point** (pixel), **line**, **circle**, **polygon** etc.. CAD systems are usually vector software programs running on raster, or vector hardware. Each object is defined as a set of coordinates and a color, and may have other attributes, such as scale, rotation and depth. Vector oriented software is usually expressed in two catagories: two dimensional and three dimensional. In a three dimensional system, each pixel has an additional value of depth. The depth indicates the distance from the observer. The depth may be used to move the line through the three dimensional field of view, and/or color the line based on the depth from the viewer. In this last case, as the line moves 'away' from the viewer, the color of the line decreases in intensity, adding dramatically to the 3D effect. Very often, vector based software systems are designed around special vector hardware. An excellent example of this is the Silicon Graphics IRIS line. These systems are designed to manipulate complex vector scenes in realtime. On the IRIS system it is possible to create fairly detailed 'objects' and move them in realtime using dials. For example, a sculptor could create the entire sculpture on the computer, turn and view the sculpture from all angles, simulate different colored surfaces and light sources, and then create the finished piece from the computer supplied data. Most of the flashy graphics used in commercials and introducing the nightly news are generated by these types of systems.

Raster based software attempts to deal with reality. In raster software, images are created by painting, sampling via a video input source such as a camera or scanner, or are created by algorithms. An interesting field of raster technology is that of **image processing**. Image processing allows us to take video signals and modify them to our desires. For example, an image could be sharpened, blured, recolored, bent, warped, rotated, rescaled, and merged with other images. Raster software is the newer of the two fields. The major reason for this is the expense and complexity of processing a large number of pixels quickly. In raster systems, the artist can deal with the image at the pixel level. A picture of a human face is too complex to express with vectors. However, to an image processor, each pixel is an object and may be manipulated in any way. Raster processers are finding their uses in such applications as medical imaging, satellite imaging, recoloring of movies and special effects.

The ideal system is a marriage between **vector** and **raster**. A good design allows the artist to manipulate both complex raster objects, and simple vector objects. The ability to place raster objects onto vector objects (called **texture mapping**) is important. For example, the ability to place real pictures on the face of a cube and rotate the cube in realtime would require a high-speed vector generation of a cube, along with high-speed manipulation of raster data onto the face of the cube, at the proper rotation and scaling to preserve the feeling of proportion.

WHAT'S AVAILABLE

The bottom line for any system is the cost. How

much does all this wonderful technology cost the individual? The answer depends on the needs of the artist. What is the artist's use for the system? If the artist is going to use the system as an idea generating system, and not for final artwork, then quality can be sacrificed and the cost kept very reasonable. If the artist is going to publish/display/sell output from the system, then quality is a concern, and the price goes up. Additionally, the final output must be considered. Some artists may require going to 35mm slide, others may require going to video tape, or movie footage, large format photographic print, weaving looms, or painting systems. The list of output capabilities is limitless, and often expensive. For the sake of time and space, we will limit our discussion to the two most popular mediums -- 35mm slide and video tape.

The VIDEO Artist

The video artist is just comming of age due to the availability and increasingly low prices of the technology. To match the quality of television, the hardware must run at a resolution of at least 512 x 400 pixels. A low-end system should have at least 256 colors out of a maximum color palette, and should be color mapped. Medium and upper level systems should be bit-mapped. To match color quality of a full-color image, the medium and high ended systems should allow a minimum of 15 bits per pixel (32,768 maximum colors). A good system will allow 24 bits per pixel (16 million colors). The frame buffer should be able to **sample** (or **digitize**) a video image in real time. Good systems should allow non-video and video information on the screen at the same time with control on a pixel by pixel basis. The output of the frame buffer must be broadcast quality and should contain seperate RGB outputs as well as a composite video output. Seperate RGB outputs are higher quality than a composite video output. Additional hardware is available for artists interested in full animation production which allows the computer to control single-frame video decks. This allows the designer to create images during the day, and to have the computer record them, unattended, at night.

The 35mm Slide and Print Artist

Many artists render their works in 35mm slide and high quality print. Even for print output, the choice for the 'master' is 35mm slide, and the artist converts the slide to print via their local photographic service. For these artists, quality is everything. 35mm slides are almost always viewed LARGE. If the original display is at a low resolution, projected, a single pixel may become as large as a fist. Therefore, the higher the resolution, the smoother the enlarged display. 35mm slide film has a maximum resolution of about 6000 x 4000 pixels. Movie film has a resolution in excess of 8000 x 8000 pixels. Computer displays used for 35mm film should be no lower than 512 x 400, preferably 1024 x 768 or higher. The number of colors available is up to the artist and the particular application.

The major problem is in producing the output. 35mm output cameras fall in two catagories: **video cameras**, and **data cameras**. Video output cameras take a picture of the actual video signal. The quality of the picture is in direct relationship to the quality of the frame buffer signal. Therefore, the resolution of the final picture is identical to the resolution of the frame buffer. Luckily, video cameras usually contain electronics to remove the scanlines associated with a tv picture (look at the tv closely if you have never seen scanlines). A data camera is a special type of video camera. Data cameras convert the image to numbers and send these to high resolution cameras (data cameras are available with resolutions from 1024 up to 8000 lines). With the data converted to numbers, special software in the computer, or hardware in the camera can take lower resolution images and improve the quality (through bilinear interpolation) to create better quality images. The major differences between data and video cameras are the cost and speed of picture taking. Video cameras typically cost between $1,000 and $10,000, and take a picture in under one minute. Data cameras cost between $5,000 and $100,000 and may take from several minutes to 20 minutes to take a picture.

ISSUES IN QUALITY

What are some of the factors determining quality in computer images? What are the problems with using an electronic medium?

Color Versus Resolution

There is much talk these days about higher resolution. Issues in resolution often deal with the term **jaggie**. The jaggies appear on low resolution displays when lines and other shapes are drawn which are not horizontal or vertical. Since raster displays deal in discrete dots of light called pixels, objects drawn on those displays are only approximations of the true object. If you were to take a grid and lay it over a photograph, and then paint each square with the average color of all the colors in

the photo under that square, you would have created a real-world equivalent of a raster display. Each colored square in the grid would REPRESENT the true image, but would not be as acourate. A line drawn in such a grid would not appear totally smooth, depending on the angle of the line and the density of the grid. Because the pixels are square (or sometimes round), stacking them at angles causes an effect known as **aliasing**. Aliasing has a real-world equivalent. When a wagon wheel turns at a fast enough rate, the spokes will suddenly appear to move backwards. Everyone has seen this effect somewhere. In actuality, what is happening is the raster display in your head (your eyes and brain) have a maximum resolution and speed of operation. The speed of rotation of the spokes *exceeded* the processing capability of your brain/eyes. The result was for your brain/eyes to use a sub-set of the actual reality, thus they appeared to move backwards because your sample was out of sync with reality. To correct this effect on raster displays a method called **antialiasing** is employeed. This method consists of drawing the line with varying intensities of color, depending on the angle. At the point of a jag, the computer will place less intense pixels. At points of visual smoothness, the computer places brighter pixels. The overall effect on your brain is that of a smoother line. With these types of software (and sometimes hardware) algorithms, lower resolution displays can approximate higher resolution displays. Another interesting phenomon is that of continuous tone images versus computed images. Take a 512 x 400 display. A circle drawn on this display will not appear totally smooth, and will exhibit some aliasing. However, a picture of a person (video captured) will equal or exceed the quality of television. The reason for this is that the video image is **continuous tone**. This means that the color from pixel to pixel varies continuously and evenly. Colors and images blend into each other. For example, a video image of a red ribbon on a white background will contain perhaps thousands of red, red-white and white colors. Where the edge of the ribbon overlaps the white background, light pink to pink-white colros will be found. This natural 'blurring' causes your brain to see almost infinite resolution (ie, your brain fills in the missing details). In the case of a pure red circle drawn on a pure white background using objects with no antialiasing or bluring, the critical edge between the circle and white background is an abrupt change. Your eyes immediatly pick out this edge and all the jaggies.

The meaning of all of this is that COLOR means more than RESOLUTION. With many colors, it is possible to make images appear to have more resolution by tricking the brain into making assumptions. This is why some systems have over 16 million colors. Most of them are used to add the textures and colorations which blend (antialias) the objects, making the resulting image appear much higher in resolution.

Color Matching

Color matching is a significant problem in computer graphics, and one without a simple solution. The problem occurs in attempting to match the colors one gets in the output, to those seen on the screen. Take the example of an artist who starts by digitizing a real scene into the frame buffer, modifying the imagery and recoloring some of it, and then shooting the final image on a video output camera. First, the original colors are modifies slightly by the video input camera which digitized the image. Next, the artist is picking colors on a monitor with a particular ambient room light. Not all monitors display all colors correctly. Most monitors have a tendency to show reds slightly oranger than they should. Also, sometimes the green gun is slightly strong and the blue gun especially recessive. Additionally, the brightness and contrast the artist has choosen for the monitor are subject to change. Finally, the output to 35mm film also changes the colors. The output monitor usually takes the picture by seperating the RGB components and passing them through color filters. The film itself is more sensitive to some colors than others. The bottom line is that color matching throughout the process of design is a complex and tedious job. This is the most negative aspect of electronic design. However, new techniques in color correction allow the user to specify correction tables which will correct the image upon output. Many video output cameras allow the user to modify each and every color in the image to be the desired color. If blues are recessive, the entire blue table can be shifted in intensity. These steps help, but tuning a system for exact color output still remains an intensive job.

AVAILABLE EQUIPMENT

Let's look at some of the equipment available today. We will review their specifications, some in detail, others in passing.

The Apple Macintosh

The Macintosh computer by Apple has its own built-in monochrome (one bit per pixel) 512 x 342

high definition display. Because the display is limited to black and white graphics only, it is generally used for publication type applications. In fact, the Macintosh is a popular choice of low-cost publication users. Some color vendors have produced add on boxes for the Macintosh line which give color graphics output. But these are generally not widely used. Current prices for Macintosh computer systems run between $1,500 and $3,000.

The Apple Macintosh II

This is the latest release from Apple. The Macintosh II is a color version of the Macintosh. The screen resolution is the same (512 x 342) but supports 256 colors out of a total of over 16 million (this makes the display color mapped). The software, however, computes images at a full color resolution of 24 bits per pixel (a full 16 million colors). It is conceivable that future equipment from Apple and other vendors will take advantage of this capability. However, being a new system, not much is available which uses these capabilities at the time of this article. Systems start at around $3,000.

The Amiga 2000

This is the latest release from Commodore Computers. The Amiga is a multi-tasking (meaning that it can run many programs at one time) color graphics computer system. It has variable screen resolutions and color resolutions and has specific hardware for the quick generation of graphics. In particular, animation is an exciting possibility on the Amiga series computers. Several good 'cellular' animation systems exist which display their results in real-time. Additionally, music graphics and music midi systems have found their home on the Amiga computers. However, perhaps one of the nicer features of the Amgia 2000 is an option of an IBM PC compatible interface. This allows the Amiga to 'plug into' the IBM line of graphics add-on boards, which offer more sophisticated graphics then the Amiga. The bottom line is that the Amiga system is fairly inexpensive, with a starting price of around $1,300. This coupled with fast hardware, an eye to music and graphics, and a growing graphic software base, makes Commodore a company to watch.

The IBM PC and Compatibles

This is the computer of choice for the serious computer artist. The IBM PC line of computers has been extremely sucessful. There are more companies building compatible IBM PC computers than any other computer system. It is the most widely purchased and supported of all personal computers. The IBM in itself does not contain exceptional graphics capabilites. But hundreds of vendors create thousands of graphic add-on products for the IBM line. The IBM line itself contains many compatible models. The PC is the lowest of the line, followed by the XT, AT and compatible versions of the 386 system. Prices start from $500 for the low end, up to $5,000 for the high end. However, mail order shoping has made the IBM PC very affordable. The wise shopper can purchase a top of the line system for as little as $3,000. As stated, the sheer number of add-on cards for the IBM line of computers is staggering. However, I have listed some of the more popular, truly professional display companys, their products and their properties:

AT&T EPICenter

The AT&T EPICenter group is a dedicated team of creative people based in Indianapolis, Indiana. The EPICenter group has continuously redefined state-of-the-art for computer graphics, not only for the area of personal computing, but for raster graphics in general. Four basic product lines make up their TRUEVISION line. These products range in price from $600 to $6,000. All the products take up only one slot in the IBM system, making them very space efficient. At the low end is the VDA (Visual Display Adapter) board. This board runs at 256 x 256 resolution with 256 colors at one time from a color map of 32,768 colors. Next comes the ICB (Image Capture Board). This is a 256 x 256 by 15 bits per pixel frame buffer. 32,768 colors are placeable at one time on the screen. Additionally, the ICB board can sample a full color composite video camera or video disk/tape signal in real time. Each pixel contains a bit which specifies whether the pixel is viewing live video or displaying a graphics pixel. This allows computer graphics and live video to display at the same time. The ICB board costs around $1,300. Next in line is the Targa series. These range in price from $3,000 for a board with 32,768 colors at one time, up to a 32 bit per pixel (16 million colors with one byte of overlay) for $5,000. Other intermediate boards exist in the Targa line, including a lower cost 8 bit board. These are all 512 x 400 resolution and have features such as hardware pan and zoom. Additionally, each pixel has the same video/graphics control as the ICB board. The Targa32 has the added capability of one byte per pixel of overlay information. This byte specifies the amount for MIXING of the live video information with the stored graphics. Since each pixel has its own mixing byte, full color graphics may

be faded into live video. Additionally, text with smooth edges, glows and other effects may be added to video. The Targa series will digitize a RGB image in both RGB and composite video mode. Additionally, the Targa boards may be reconfigured on the fly to act as four pages of ICB resolution. This is especially nice for multi-buffered animation. Finally, EPIC's latest entry to the graphics market is the Vista board at $6.000. Vista can be configured to be 32 frames of ICB resolution, or 16 frames of Targa resolution, or one frame of 1024 x 768 by 32 bits per pixel or one frame of 2048 x 1024 resolution at 16 bits per pixel. Over 16 million colors are available at one time as well as an additional byte of configurable data per pixel. The Vista board represents another landmark in raster graphics with the concept of a display which is both color mapped and bit mapped at the same time. Individual pixels may be assigned 32 modes including . . . bit mapped graphics, live graphics, color mapped graphics . . . all on the screen at the same time. This allows one screen to have live video, realtime animation and color map animation all occuring simultaneously. Additionally, Vista contains its own computer (Texas Instruments' famous TMS34010) and requires no processing from the PC itself. Therefore, the PC may be freed up to perform other tasks while the Vista is rendering graphics. Additionally, both the PC and the Vista computers may write into the frame buffer at the same time.

Many vendors support AT&T graphics cards with a wealth of software. From sophisticated paint programs such as TIPS from Island Graphics, to fully professional 3D single frame animation systems such as the West End Films Animation Package. Virtually any application is met with a software solution from various venders which support the EPICenter line.

Number Nine Computing

Number Nine makes less sophsicated graphics boards than the AT&T line. However, Number Nine is also cheaper. Their new PEPPER series of graphics cards offer high resolution, starting from the PEPPER SGT card which offers 800 x 600 resolution with 16 colors out of a palette of 16 million viewable at one time, to the PEPPER Pro1280 board which offers 1280 x 1024 resolution with up to 256 colors out of a total of 16 million. If your application is not color intensive, but requires high resolution output, then these boards represent a lower cost alternative. Prices range from under $1,000 to around $2,000. Another interesting feature of these boards is the use of a graphics 'operating system' called NNIOS. NNIOS allows applications to run on the entire PEPPER graphics line. This is important for upward mobility.

WHAT TO LOOK FOR

What should the buyer be looking for in a system? What are the pitfalls and where is the best place to shop without getting ripped off?

What To Buy

No matter what brand of computer you buy, you should have at least 524,288 bytes of memory (512K) and preferably one megabyte. Though you can add memory on, many manufactures offer these sizes standard. You will also need at least one hard disk drive and one floppy disk drive. The floppy disk drive allows you to buy off-the-shelf software and load it onto your system. The hard disk drive allows you to organize your files, keep large amounts of information always accessible, and allows the computer to run faster, not to mention saving you all the time of switching floppy disks. Additionally, images on large resolution devices may be very big. An image from the AT&T Vista card can be as large as three megabytes. This is significantly more than can even fit on a floppy, so a hard disk drive for some graphics cards is the ONLY way to save an image (without splitting the image into lots of smaller images). Your hard disk drive should be at least 30 Megabytes for a professional graphics system, if not more. Hard disk drives are very inexpensive and a good shopper can find fast 30 megabyte drives from $400 to $700. Almost all computer manufactures offer multiple systems with different 'standard' amounts of disk storage from 10 megabytes up to 80 megabytes.

You may, or may not require a math co-processor. Most of the IBM PC line, and many of the other lines of computer contain an empty socket near the CPU. A math co-processor may be purchased, sometimes as low as $100 to plug into the socket. Some (not all) software packages take advantage of the math co-processor if they detect it in the computer. These packages may offer significant speed improvements in calculating graphics or other numeric intensive functions.

Make sure you pick a high quality monitor. Many systems come 'bundled' with a monitor. However, some distributors like to offer cheap monitors. Eyestrain is a significant problem with continual computer use with these monitors. The monitor should have clean and crisp typeface and should contain standard features like vertical and horizontal

positioning, contrast and brightness controls. Often it is better to purchase the system without the monitor and pick a monitor yourself. or simply junk the monitor if it was provided as part of the system and proves not to be of a high enough quality. If you are purchasing an IBM PC or compatible, get a monitor with at least a CGA capability, if not EGA capability. Briefly, the IBM line may be purchased with one of three types of monitors and display cards, MDA (Monochrome Display Adapter), CGA (Color Graphics Adapter) and EGA (Extended Graphics Adapter). While these graphics adapters are not high enough quality for serious artistic work, they do display a higher quality image than standard monitors, and cost very little more. Since you need a standard computer monitor anyways, the small extra money you would spend on EGA graphics and monitors is worth the headaches you will get later.

With add-on boards such as the AT&T and Number Nine boards, you need to purchase a secondary monitor to display your image. This monitor MUST be high quality and may cost between $500 and $1,500. Some of the newer monitors are termed **multisync** which means that they can hook up to a multitude of outputs. These monitors offer you an alternative, in that they can be used for both the computer display, and the graphics display (not at the same time however). If your software packages do not require both monitors at the same time, this might be a good path. Additionally, beware of the termonology of **digital RGB** versus **analog RGB**. Most MGA, CGA and EGA display cards put out a digital RGB signal. These signals are only capable of a few colors (16 is average). However, professional display cards such as the Number Nine and AT&T cards put out high quality **analog RGB** video. You can not view digital RGB image on an analog monitor, or an analog image on a digital monitor. Most of the multisync monitors have a switch which allow both digital and analog signals.

Next, your system needs some type of input device such as a **mouse**. This device is used to select commands from menus, and paint or design graphics. It is a small device which fits in the hand, and rolls on a surface. Rolling the mouse causes the cursor/paint-brush etc. to move on the screen. If you purchased a system such as an Macintosh, or Amiga, then the computer comes with a mouse. However, if you purchase an IBM or compatible, the mouse will cost you extra. Mouse input is the most popular type of input currently used, and virturally all graphic programs allow this type of input. However, many artists prefer to work with a graphics tablet; some prefer light pens. Many manufactures offer several types of inputs, but their use depends on the software packages. Check the software first. When buying a mouse for the IBM PC and compatible line, make sure it is a Microsoft compatible mouse. Any mouse non-Microsoft compatible may not work with your software, however most all mice are Microsoft compatible.

Most computers come with Input/Output ports (RS232) which allow you to connect your computer to printers and phone modems. If your computer does not come with I/O ports, you can add these on later at a minimal cost. In fact, you can add almost any component to the computer after the fact. In general, if you are unsure that you need a particular option, wait, it will be cheaper later.

The software you buy will be determined by your application. Before buying a graphics board, contact the manufacturer and request a list of all products relating to the graphics board. Most manufactures will be more than happy to suggest products and second party vendors who support their equipment. Make sure your application is covered from more than one source. It is important to have a choice when picking software directly geared to your desires.

Picking good software is an art in itself. Much of the software designed today is of poor quality. Shrewd shopping can pay off when buying software. The quality of the documentation should be high. Is there an index or table of contents? Are all the features spelled out with examples? Does the software manufacturer have a 'hotline' that you can call with questions? Try out the program whenever possible. Is it comfortable to use? Are the menus easy to read and understand? Programming houses which design in ASSEMBLY LANGUAGE generally produce faster and tighter products than those programming in higher level languages such as 'C', FORTRAN or BASIC. If the software package you are purchasing is significant in price, it may be wise to request the name and telephone numbers of some of their users. Call the users and ask them how they like the program. First hand advice from users may often save you hours of frustration.

Where To Buy, And Why

There are three places where computer equipment may be purchased. The first is from your local computer dealer. The second is from mail order warehouses. The third is directly from the

manufacturer. Each source has its own pros and cons.

Buying from your local computer dealer will probably cost you 50 to 100 percent more than you would pay from mail order. However, you are also buying (hopefully) peace of mind and support. If the product fails, you have a local 'friend' to which you can turn. If you go this route, be aware that you are going to pay more no matter to what store you go. So, pick the store based on other factors. Do they carry a good line of add-ons? Are the salesmen helpful and knowledgable? Does the store have their own service department? If so, what are their terms and turn-around times? How long the store has been in business is a good indication of their staying power. Going with a computer chain store overtop of a truly local store does not always mean better service as both types of stores hire local people to staff the establishment. Make sure the establishment you choose has technical as well as sales persons employeed.

If you choose to buy your computer through mail order, make sure to read several periodicals. Paticularly good periodicals for computer mail order advertising are BYTE Magazine, and COMPUTER SHOPPER. Both have an incredible number of ads and prices vary throughout the entire range from ad to ad. Literally, from one page to the next, you may see as much as a $500 difference in price for the same product. Picking the proper company to avoid rip-offs is tricky. Most companies are good, and some of the best deals are real. However, false advertising by dishonest people have been know to occur. Both BYTE and COMPUTER SHOPPER take some steps to ensure valid advertising, but proper screening of all ads is impossible. If you see an ad you like, make sure it is in more than one magazine and/or issue. How do they want payment? Do they offer a toll free 800 number? Call the company and inquire about the product. Were they knowledgable and helpful? Finally, if you have real doubts, call the Better Business Bureau for the town shown in the company's address. Have them check for you as to the validity of the company. 99.9 percent of the time, ordering mail order is not only cheaper, but often indistinguishable from the service you would recieve locally. Lastly, remember that what you bought today for $500 will be advertised next month for $399.

The final method of purchasing is directly from the manufacturer. Sometimes, the manufacturer will pass you along to dealers or representativites in your area. Ask the manufacturer for mail order houses which also carry their products. In the case of high resolution graphics boards for the IBM PC line, you most always need to contact the manufacturer to receive the address of the dealer near you. In any case, ask the manufacturer to send you the specification sheets, and to place you on their new product mailing list. You will often pay the highest price from the manufacturer. Even local stores buy in quantity, offering substantial discounts. Mail order houses buy in very large quantity, and have a low overhead and thus can offer better deals.

Caveats

1) Make sure any company you buy from has a return policy and a warranty period.

2) Make sure the company you purchase from can offer service, or suggest another vender to supply service on your product. Most computer equipment will work with no problems, but occasionally, parts fail, or lightning strikes.

3) Make sure the company you purchase from can offer help should you be unable to get your system running properly.

When buying software, try to stay away (if possible) from the company which copy protects its disks. Copy protected software means that the manufacturer has physically made sure that you will be unable to run the software on more than one machine. With many of these packages, if you should lose the data on your hard disk drive, you would have to repurchase the program again. Additionally, some copy protection schemes cause the equipment to operate in a non-standard manner, and can even be harmful to your system. Copy protection is not professional and shows a company which is more concerned with their pockets than a company which is concerned with the user. For many computers, lock-smith programs exist which can unprotect some of the more popular copy protection schemes. If you must buy copy protected software, use these programs to unprotect the disk and immediatly make backups for your archives, in case of data loss, or loss of the original disk.

If you are buying an IBM PC compatible, make sure that the manufacture guarantees that the computer is 100% IBM compatible. If it is not 100% compatible, not all add-on boards, or software programs may run. Any serious artist should not even consider the IBM PC or XT, but should consider instead the more powerful IBM AT or compatible 386

machines. These computer are extremely high speed and can be purchased through mail order from $1,000 to $5,000. One of the fastest, the 386, can be purchased mail order for $3,000. The local distributor price for an identical system (different manufacturer) was $9,000.

Additionally, be wary of any new computer equipment. An example of this would be the new IBM professional line. Until 2nd party software and hardware manufacturers begin to support the new lines, they are risky.

THE ULTIMATE SYSTEM

The ultimate system, and approach, is to buy the hardware which satisfies your needs, and have the software custom designed for you. The major problem with most software is that it rarely does all that you want. The manufacturers can not satisfy all people at all times. The 'if only it did that' syndrome can only be solved by having your software custom designed. There are two approaches, the expensive and the cheap. The expensive approach is to hire an experienced computer graphics programmer and enter into a contract for services. Usually, the price of such a development means that you are going to market the product to re-coop some of the initial cost (marketing a desireable product is a very lucrative sideline).

The cheap approach is to inquire at the computer graphics laboratory of the college or university near you. Even high schools are fair prey for such ventures. If you have purchased some software to use temporarily, and are able to wait the time for these less experienced, but very eager and cheap (if not free) programmers to finish your dreams, the benefits can be very rewarding.

CONCLUSION

The final result of your computer venture is up to you. Get seriously involved. Read magazines to find out what's hot and what's not. Attend computer conventions when they come to your area. See what type of equipment agrees to your eyes and what doesn't. Go around to local stores and ask to see and use various systems. As you get more familiar, read more technical information to remove some of the mystery of operating and using the machinery.

Finally, even though computer prices are dropping every day, do not wait too long to get involved. Avoid the deadly virus of computeritis where you find yourself saying 'if I just wait another day/month/year, it will be cheaper'. This deadly ring of procrastination will see to it that you never have a machine. Pick a machine which offers the best in future expandability so that as new technologies come out, they can be immediatly used by you. Good luck!

REFERENCES

Computer Periodicals:

For Ads and Information...

1) Byte Magazine:
   PO Box 328,
   Hancock, NH 03449
   (603) 924-9281
2) Computer Shopper:
   407 South Washington Ave.
   PO Box F
   Titusville, FL 32781
   (305) 269-3211

For the latest graphics information:

1) Computer Graphics World:
   PO Box 122,
   Tulsa, OK 74101
   (617)486-9501
2) IEEE Computer Graphics and Applications
   Computer Society of the IEEE
   Circulation Department
   10662 Los Vaqueros Cir.
   Los Alamitos, CA 90720-9970

Computer Graphic Societies:

1) The ACM/SIGGRAPH
   11 W. 42nd Street
   NYC 10036
   (212) 869-7440
2) The IEEE Society
   PO Box 80452
   Worldway Postal Center
   Los Angeles, CA 90080

Books on Computer Graphics:

1) Berger, Marc
   Computer Graphics With Pascal
   Menlo Park, CA.
   Benjamin/Cummings Publishing

2) Rogers, David F.
Procedural Elements For Computer Graphics
New York, Mc Graw Hill
3) Foley, J. D. and Van Dam, A.
Fundamentals of Interactive Computer Graphics
Addison-Wesley Publishing

Addreses of Vendors Mentioned:

1) Commodore International
1200 Wilson Drive,
West Chester PA, 19380.
(215) 436-4200
2) Apple Computer Inc.
20525 Mariani Avenue.
Cupertino, CA 95014
(408) 996-1010
3) For the address of IBM and IBM compatibles, look up the many makers of IBM and compatible equipment in issues of *BYTE* and *Computer Shopper* magazines.
4) AT&T EPICenter
7851 Shadeland Station Way,
Suite 100,
Indianapolis, Indiana 46256
(317) 841-0332
5) Number Nine Computer Corporation
725 Concord Avenue,
Cambridge MA. 02138
(617) 492-0999

Additional, unlisted vendors:

1) AT&T Graphic Software Laboratory
10291 North Meridan,
Indianapolis Indiana 46290
(317) 844-4364
This group of people are the software equivalent of the EPICenter group. Major products include RIO, an object oriented, very high resolution, paint package. Supports the EPICenter cards.
2) MATROX
1055 St. Regis Blvd.
Dorval, Quebec, Canada H9P2T4
(514)695-2630
MATROX makes a series of graphics boards for the IBM PC line of computers. One, in particular, contains some of the power of the Silicon Graphics IRIS system in manipulating objects at high speed for animation.

A CRITIC'S VIEW of Computer Graphics

STELLA PANDELL RUSSELL Ph.D, M.C.A., M.A.
Professor of Art

NASSAU COMMUNITY COLLEGE
AUTHOR - HOLT, RINEHART and WINSTON

ABSTRACT

The diversity of computer derived imagery runs from commercial to fine arts. Six criteria concern most art critics in evaluating computer graphics as well as all other forms of art. A few unique aspects, little explored as yet, may determine masterpieces of computer art in years to come.

INTRODUCTION

No doubt about it....

The art world is ready for a new idea in art, and it just might be art work derived from a computer. Yet, how much computer output is a mathematical product of that tool and not art at all? If so much of the art form depends upon technology, aren't the results gimmickry instead of art?

In our time we have come to realize with Marshall McLuhan "that societies have always been shaped (more) by the nature of the media by which they communicate" [1] than by anything else. By extension, the computer art we create --if it is art--is shaped as much by the machine as by what the artist demands from it. We will quickly establish today that imagery generated, or enhanced, by computers may indeed be valid as an art form, but we must add that not everything that appears on a screen will make it in the art world.

DIVERSITY

Diversity in computer graphics ranges from the ad art marketplace, the television studio, the movie theater, the science lab, the school, the hospital, the hotel lobby penny arcade, and even in the games on low-end home computers. But validation for the fine artist can be found in none of those places. What concerns us here are the evaluative criteria that admit our computer art to the art gallery, to the museums, and most important of all, to a place in history.

Michael Brenson, New York Times art critic, informed us not so long ago [2] that the current art scene demonstrates no particular focus. "There is a sense that anything can happen", well, anything is an open-ended term that includes computer art when the images look just right to an art critic (or art historian). Is that label of "good art" only a matter of opinion to a critic? Not really. There are grounds for judgments.

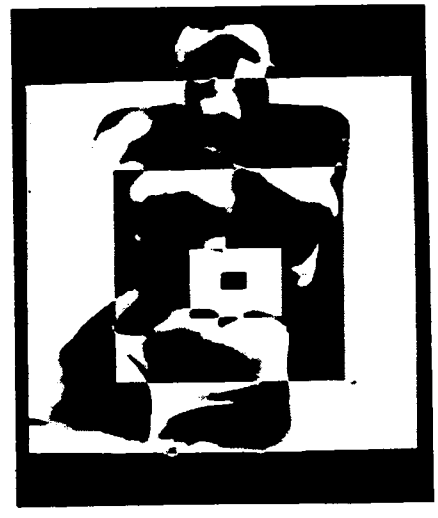


Figure 1. "Stella Russell's computer piece, derived from the Bible, 'Eve Was the Mother of All Living', is particularly successful. The strange nonreality of undefinable electronic space is astutely handled by designing the figure to fall into four superimposed geometric areas. The equally unreal electronic light has been chaneled to give the appearance of multiple illumination sources, as if a different time and place is being shown simultaneously. Remarkable too, for computer art, is the way an emphatic glow plays over fully rounded, fleshy body contours. The piece manages to be sophisticated and at the same time frankly machine-assisted and modern."
Phyllis Braff, NY Times Art Critic
February 15, 1987

[1] McLuhan, Marshall
The Medium Is the Massage , p.8

[2] Brenson, Michael
New York Times - December 23, 1984

ART CRITICISM

Art criticism has a venerable history that goes back at least to Plato. It seems to be doublefold, consisting of a sense of art history combined with a trust in intuitive judgment. There are no permanent and irrefutable critical viewpoints on art, but there is something like a consensus, which becomes evident over a period of time, about the value of individual works or a body of work. There is also a procedure for making critical judgments which has evolved out of Renaissance beginnings.

Today, while the criteria for judging art are continuously evolving, art criticism plays an important role in the dynamics of cultural life, with its main goal educational. Hilton Kramer, Editor-in-Chief of "The New Criterion" and Chief Art Critic of The New York Times for many years describes his job quite simply:

> The first task of the critic is to see the object for what it is, and to describe its qualities and attributes as clearly and concretely as possible. This is the task of elucidation...
>
> Another important function of criticism is evaluation...best performed as frankly as possible with reasoned arguments and detailed examples and comparisons.

The key to the process, of course, is "to see the object." Professionals approach critical assignments by asking questions like these:

1. How does this work fit into the history of art? In what way does it remind us of other works we have seen? Are there connections to be made between this piece and those that preceded it? For instance, does a 20th century work imitate a masterpiece produced 100 years ago? If so, its unique value is questionable.

2. Is there anything known about this artist that might influence our responses to the work? For example, is this the work of an unknown, or a famous artist whose works are all highly regarded?

3. When and where was this work created? What was the context in the artist's time which inspired the work? You might ask, for instance, "What induced Picasso to create "Guernica" within a scant 2 month period?", and learn it was horror at an event in Spain's civil war.

4. How has the artist used the medium that presumably explains the reasons for its choice? Does watercolor, for instance, reveal freshness of hues and, perhaps, transparent layers of colors and the crisp highlights of unpainted paper seen through the paint? Or does the wood-grain of a block enrich a woodcut?

5. How has the artist's vocabulary--form, techniques and organization--contributed to the total effect? Does the work evoke feelings and memories that you may associate with the artist's inspiration, like Van Gogh's "Starry Night"?

6. Does the content of the work reveal the artist's goal in creating the piece? If the theme is important, is the message clear?

Armed with only these six steps to art appraisal--1) art history 2) the artist 3) the art era 4) the art medium 5) the language of art and 6) the content, if the theme is vital to the work--should allow almost anyone to look at art like a critic.

For many artists, intellectual games, as they may consider this format, is foreign to the way they work, but if an artist's goals are recognition as much as personal satisfaction, there's nothing to lose and much to gain from this approach.

ART CRITERIA

Let's begin by setting computer art into an historic framework. Like photography in the 19th century, computer graphics adds a new medium of expression important to our contemporary world, but like the critics (and public response) a hundred years ago, few viewers are equipped to understand or judge this new art. Each time a new art form is introduced, it is compared to its disadvantage, with its antecedents--just as impressionism was contrasted with academic realism, and photography with painting, many of us see the computer today as an electronic paintbrush, instead of applying to it a new aesthetic. Another example, neither cubism nor abstract expressionism could be perceived at the time as the interpretations of nature and reality people were familiar with. From today's perspective, we understand cubism as a natural response to African

art, the space-time theories of Einstein, and an interpretation of the multiple images shown by the newly invented motion picture projector. We also can now accept Pollock's drips and splashes as the historical evolution of Monet's large field painting paintings of waterlilies, combined with the action painting emotional techniques that revealed Van Gogh's agitated state of mind when he worked. Perhaps, using a computer to simulate familiar forms of art may be the route for acceptance with art critics who will reject anything else they do not yet understand, but, in the long run, critics will not use painting as its major point of comparison. Computer art need not emulate traditional printing, but, as a legitimate product of the electronic age would probably do best to demonstrate what these remarkable attributes may be. If we are still too close to do that now, we can be sure that historians who follow will not have those difficulties. They will pinpoint the pioneers who have intellectually determined what is special in the machine or have simply blundered upon it.

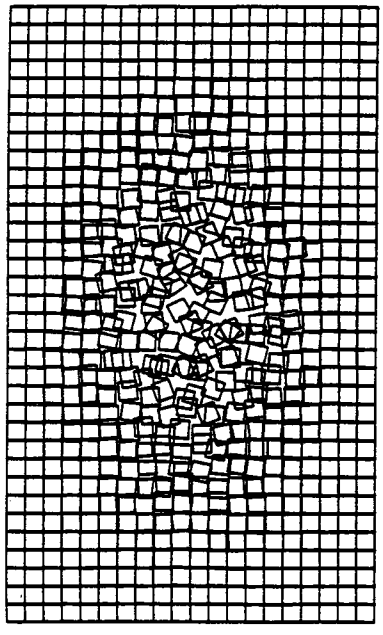
Next, which computer artists will a critic review? Certainly, those artists who have a secure reputation. Critics will go for the people they know even if those names are famous for something else. There can be no doubt that a comtemporary Leonardo da Vinci or Michelangelo Buonarroti would be experimenting with computers since in their own times both were attracted to the avant garde in their crafts. As artists familiar to us, their works would be always reviewed. Critics who know Duane Palyka, as an artist-scientist or Ed Emschwiller in video would look with interest on their newest work. Unknown artists rarely command the attention of a reviewer, although of course on rare occasions a critic seems "to discover" a latent genius perhaps who has been carefully "groomed" by a collector or a dealer for that publicity. An artist who is truly an unknown quantity can expect little from a professional critic unless the work is extraordinary in itself, -- always a possibility.

The context of a computer work, whether it be static imagery or animation, will certainly affect how that work is evaluated. An episode in "Star Wars" or a feature film like "Tron" has an enormous built-in science-fiction audience, anticipating almost any art of a fantasy nature. On the other hand if our computer art deals with women's issues, a show devoted to that theme, while attracting a much smaller audience, will be entirely receptive to most work submitted. "New Technology of the Nineties" would be an appropriate arena for xerography, holography and of course computer graphics. A professional reviewer will be prepared to deal with art according to its connection with the culture from which it originated. No artist can afford to ignore this aspect of the art world any more than an artist can overlook well-established idiosyncratic responses of the critics, curators or journalists who help determine the canons of art taste. While there are legions of judges who evaluate local events, the numbers of reviewers in the country are finite. It is possible to consider their orientations to specific contexts, if it is important enough to do so.

In evaluating computer art, its most significant feature, no doubt, is the medium itself. The tool is particularly unique in separating memory from intelligence. The computer permits among other things: serial repetition, distortion, almost infinite manipulation of colors, cut and paste facility, enlargements and reductions, etc. Whatever expressive possibilities computer art offers, its special characteristics seem to be intellectual above and beyond emotional impetus. The viewer is most likely to perceive the artist as creating with the brain, more indeed than with the hand or the heart. In a related way, people reacted to cubism. No one was able to project searing emotion into a cubistic work other than Picasso who deliberately chose cubism to depict fractured aspects of human feelings to demonstrate his horror at the senseless bombing of the sleepy city of "Guernica". This work, perhaps his greatest, was rendered in a medium not thought to be appropriate for his message. The potential of the computer for emotional expression it seems, has not yet been reached. Since so much of the art form depends upon the technology, the thrust of the art has appeared scientific. While audiences have been somewhat

intrigued with geometric art, for example, the appeal of a Mondrian is far from universal despite its brilliant colors. Aesthetic appreciation of subtle differences in line such as can be seen in a Paul Klee drawing are not for everyone. Therefore, the patterns generated by Bill Kolomyjek are beloved art to the computer artist cult, but not necessarily for the masses--or the critics. Is it possible that memorable art requires a tug at the heart strings? What we admire most seems to be human-oriented experiences. The infusion of life from God, the Father into Adam by Michelangelo is one of those arresting images of a theme that thousands of artists before him had never fully exploited. Nor, it would seem, has been the medium of computer graphics.

What about the artist's vocabulary? Have the visual elements been effectively explored in computer art? Each offers expressive possibilities in all media. Line, value, color and texture are easily available to the computer artist in static imagery, while time and motion are surely the basis of computer animation. The principles of design can serve the computer artist just as they operate in all the arts, that is, unity and variety, balance, rhythm, proportion and scale, thrust, dominance and subordination. Since so many computer artists have come to the ranks from science instead of art, few seem to have the background to fuse some of these expedients into a computer work of art but when it happens, the images are as arresting to critics as to computer hackers.



2. William J. Kolomyjec Bones I
By permission of William J. Kolomyjec. Copyright © 1989 Kolomyjec Design

Finally, we should consider the criterion that the lay public will use to gauge computer art, and a factor inescapable even to the museum official. We must now address the possible content of the work. So much of the computer graphics we see demonstrates the intellectual prowess of the designer. What was really the artist's reason for creating the work should come across as we view it. The complexity of hidden lines should not concern the viewer any more than we must consider the hours Baryshnikov spends to permit him to appear to effortlessly soar across the stage. The omission of content in much computer art is likely to be the major factor governing the responses of critics to the art. The exciting new territory offered by the computer is for most of us sufficient to draw us to the medium, but the public likely enough is not ready for art that may only express line, color and texture.

Those criteria the computer exclusively offers to artists were expressed perhaps best by New York Times art critic, Phyllis Braff, a pioneer in its evaluation, quoted freely:

1. The computer monitor tends to lend credibility to its imagery, since we see the screen as a source of information at the stock market, the airport, etc. Its measurements in part dictate the proportions of our work.

2. Public perception of the computer as science closely relates the art to be a product of logic as well.

3. The computer screen offers a new kind of space, without gravity or landscape connotation.

4. Perceptions of mass, even in art created by fractal geometry, need not duplicate reality, and for many will not simulate real volume.

5. The CRT offers a different kind of light that is electro-magnetic and compelling.

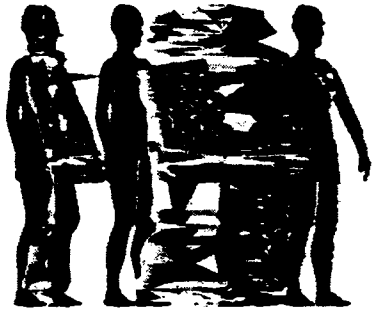
6. The potential for interpolating scanned images has hardly been explored at all.

7. Access to grid and grid-like imagery is undeniable, as is....

8. ...the capture of motion and flux in the world around us.

These last eight aspects of computer art may offer a whole new perspective into the medium and the substance of yet another paper.

In conclusion, perhaps, computer imagery will ultimately be evaluated in history by the creativity of the computer artist, utilizing criteria that the 20th century has yet to identify. For this reviewer, much of the work of Duane Palyka succeeds in meeting most of today's and tomorrow's evaluative concerns, "Picasso Two" is classic art.



SUMMARY

In general, art critics and historians consider six factors in evaluating art: 1) art history 2) the artist 3) the art era 4) the art medium 5) the language of art and 6) the content, if the theme is vital to the work. These criteria are literally extrapolated from my text Art In The World to be reissued, third edition, Fall 1988 by Holt, Rinehart and Winston, Inc. Eight additional aspects unique to the computer: the screen, computer logic, computer space, mass connotations, computer light, interpolation, the grid, motion and flux, along with other issues will be examined in depth in THE IMAGE REVOLUTION in Computer Graphics, possibly released by Holt, Rinehart and Winston in 1989.

BIBLIOGRAPHY

Brenson, Michael "New York Times" 1984

McLuhan, Marshall
"The Medium is the Massage"
Random House, N.Y. 1967

Russell, Stella Pandell
"Art In the World", 2nd ed.
Holt, Rinehart and Winston, N.Y. 1984

A REVEALING NEW ART FORM: SOME IMMEDIATE BENEFITS OF AI

Bruce Bubacz

University of Missouri-Kansas City

ABSTRACT

Increases in RAM and developments in interactive programming have made a new art form possible. This new art form engages the artist and audience in new and intimate ways. Some forms of this new relationship are proposed. It is argued that as a consequence some old aesthetic problems will be resolved. However, new problems relating to the integrity of both artists and the art establishment will be raised.

"Where do birds come from? How did that tree get there? Do rainbows come from the air?" Children are fascinated by origins and, if this fascination survives the madness of adolescence, a lucky child might become a historian. Historians trace origins systematically, beginning with a definition then searching for the earliest example that satisfies the definition. Occasionally the historian's task is simplified by the importance of an event, ("Heavier than air flight began at ...." "Caesar crossed the Rubicon on...."). However, most important matters — for example, cultural or political or philosophical developments — begin unheralded. Thus, if we ask, "Who wrote the earliest novel?" a search is necessary. Apuleuis did not know that he was a literary pioneer.

The historian's task in searching for origins in the arts is complicated by the disputatious impulse of aestheticians and art historians, an impulse which is especially violent when applying definitions. Apuleuis's The Golden Ass may be denied the status of earliest novel because (1) there is an earlier work that meets the proposed definition or (2) The Golden Ass does not fulfill the criteria mentioned in the definition or (3) the definition was defective so the search was flawed.

Sometimes disputes turn on definitions. Photography, it is said, is an art form, but what about performance art? Some would claim that public speaking is an art form, but what about following strangers? In 1917 Marcel Duchamp signed a porcelain urinal "R. Mutt" and placed it on display. Was the selection and signing of the urinal an art form? Was it still an art form when he did it again in 1951? And then again in 1964? Cliff Westermann's "The Last Ray of Hope" is a pair of shoes. Was labeling the shoes an art form?

Most people would characterize Duchamp's signed urinal and Westermann's shoes as anything but art, and it is ironic that those who would dismiss the views of "most people" on most things are prepared to agree with mass reactions to the arts. Even educated people would sneer at signed bathroom fixtures, including Joseph Beuys' "Child's Bathtub" of 1960. Of course these silly things aren't works of art and selecting them for display is not an art form. Most people can see that.

In fact, there is a lot here that "most people" cannot see, for these issues are part of an intellectual briar patch and, although Br'er Art Historian may be quite happy there, for the rest of us it is an uncomfortable and confusing place. We read about Chris Burden's riding around tied to the front of his car, we see Vito Acconci's "Gang Bang," we hear John Cage's "HPSCHD" and we sputter, "That's not art!"

Now, critics, and other aesthetic souls spill much ink trying to convince the rest of us that something which seems just silly or self-indulgent or downright stupid is, in fact, art and that if we had open minds and souls we could appreciate it. Since many of us are timid in the face of true believers, we are frequently browbeaten into accepting the ironic or the academically titillating in place of the merely beautiful, and thus our public and private spaces become encrusted with works that have no effect on us, leaving us empty of any reaction or insight. And it isn't very helpful to be told that the treatment of art-as-form necessitates "non-beautiful" art.

In light of these incredibly complex issues, what happens when someone proposes that something counts as a new art form? We shall see, for I will argue here that recent technological developments have made possible a new sort of creation that would count, by any sane criterion, as a new art form.

The technological developments that make this new art form possible involve, of course, microcomputers. Specifically, they are enormous increases in RAM and the increased sophistication of interactive programs and expert systems.

RAM is the memory that a computer can access immediately. As RAM increases, so does the amount of a program that is instantly available to a user.

A program is a set of instructions given to a computer. Most of us are familiar with the way that simple programs work, those that run automated bank tellers, for example. Typically, the user confronts a screen that asks for a specific entry. Is this a deposit or a withdrawal? Choosing "deposit" calls up a new screen which asks whether the deposit is to checking or savings. A third screen then asks for the amount of the deposit. At each step a simple prompt leads the user to choose from a limited number of options. Only a small part of the program is in use at any one time.

The limited scope and machine-like delay in response which is characteristic of ordinary programs is absent in interactive programs. Interactive programs respond immediately to a user's inputs, giving the impression of a conversation with the computer, immediately processing a user's input and responding to it. As an interactive program increases in size the number of possible responses for each possible input increases. As such a program increases further, the number of possible inputs expands, adding more possible responses.

Expert systems may be thought of as software managers. They oversee other programs, allowing many different programs to be used simultaneously. In use an expert system gives the impression that several diverse programs are actually a single integrated program. Such systems are the most useful results thus far of research into artificial intelligence. Expert systems are an important step toward multi-tasking, the ability to perform more than one operation in the same part of a computer's central processing unit.

Notice that both interactive programs and expert systems are sophisticated deceptions — they both convince a user that something is happening in the computer that, in fact, is not happening. The computer is not conversing with the user. There are many programs in use, not just one.

Filmmakers have exploited the possibilities of computers for several years. John Dykstra used computers to control the miniatures in the various "Star Wars" films, and Harrison Ellenshaw's striking computer generated graphics are the best thing about "Tron." Other artists have begun to investigate the potential for computer programming. As Stephen Wilson tells us in his excellent Using Computers to Create Art (Prentice Hall, 1986), that potential is nearly limitless. Consider what a microcomputer can do. Computer outputs may be visual, aural, literary or cinematic. Outputs may be printed to a video screen, a printer a loudspeaker or all three. Images may be static or moving, representational or abstract or both. Any sound may be reproduced and, of course, any written work may be read from a screen.

This in itself provides new opportunities for combining old media in interesting ways, and there should be no argument that the result would be an art form. If novels are works of art, then so are novels presented on computer screens. If a print is a work of art, then so is a print reproduced by a graphics printer. However, novel writing and print making are established forms of expression and I promised a new one. We will find it in the interactive character of new programming techniques.

Interactive programs must be written in such a way as to maintain the illusion that they are in communication with a user. The better the programmer, the more convincing will be the apparent conversation with the computer. This illusion of conversation and the ability to immediately access many different programs leads to the possibility of a new art form. It is now possible to create a work of art that reacts in complex and diverse ways to an audience response, changing either to reflect or to modify the response of an audience. If the audience reacts in the way that the artist expects, then a particular path in the program is accessed. If the audience does not respond as expected, then a different program is accessed that (perhaps) will bring the audience back on track. Here are some suggestions for interactive art.

Neither authors nor readers of mystery stories want "Whodunit?" to be answered too early. Consider an interactive mystery story — a novel or a film or a narration or a combination of all three. The audience is asked at an early stage to guess whodunit. If the audience guesses correctly, the story begins to change, presenting new evidence that either leads away from the original guess or confuses the audience by pointing in a new direction. If the audience guesses incorrectly or is unable to determine who the villain is then new evidence is presented that leads the audience down the intended path.

Consider an interactive musical composition that has as its purpose inducing the audience to relax. The audience indicates that it is not in the least relaxed (through a simple input or through something more complex, perhaps a system that monitors biological processes, including brainwave activity). As a result, the composition changes as it is presented to the audience to bring about the intended response.

An artist creates a series of interactive prints titled, "Evocations of Childhood." She wants her work to appeal to any viewer so she begins by asking about the viewer's childhood. Where did you grow up? Did you have a dog? What were your favorite games? This information is read into the program which modifies what will be printed on the screen or the printer. (Note that this need not be a representational work and that even if it is, it need not represent the viewer's childhood if that is not the artist's intention.) Of course, responses to further prompts as the images unfold may be used to modify new images.

Audience reaction can have a direct and immediate effect on a work of art — a work can change to reflect the response of the audience. It would be like reading a novel with the author sitting by your side, changing future chapters as you talked about the ones you were reading. Interactive art would serve as the realization of efforts that have only achieved a very primitive level of audience/artist interaction. Happenings may have been art forms that included the activity of an audience, but what difference does the artist make in the happening? It is true that cabarets have traditionally included "improvi-

sations" that were inspired by audience suggestions, but there is often something fraudulent, something studied and rehearsed about the results. Interactive art would involve genuine responses, anticipated by an artist, to audience reactions.

Much of the ferment in the Twentieth Century art world has centered on such problems as the intent of an artist, the determination of authorship, and the significance, if any, of audience reaction in judging a work. The simple fact that such problems are difficult to resolve keeps this debate alive for most art forms. However, many of these problems would be easily resolved if a work is claimed to be interactive. Consider the question of authorship. Is the artist the designer or the craftsman who produces the actual object? In some cases the fact that a work has not been touched by an artist raises no problems. Composers need not perform their works and architects need not construct buildings. Other cases are a bit harder. Who is the artist when one person gives another vague instructions? ("Nail these boards together any way you wish and spray them blue.") Is it the person who had the idea or the person who did the nailing? And whom are we to blame for a musical composition consisting of four radios playing simultaneously?

Such problems should not arise for interactive art because of the nature of the medium. In order to see this it is necessary to understand how the components in the interactive art experience are classified and interrelated. The technique used by the artist is interactive programming. It can be learned by anyone with moderate intelligence and skills. The art form is any use of such programming to create an interactive environment for an audience; the art form is flexible in application and may in fact cross the boundaries of traditional art forms. In this regard it fits well with contemporary developments. The creation is the program, which is static (in the same way that a musical composition is static.) The actual experience of the art form involves audience reaction to the output of a computer, reaction which is entered into the computer and accesses different parts of the work to change the form and content of the audience's experience. Absent this interaction, then, the work only exists potentially; there is no work until there is an interaction.

We may now ask whether writing such a program really be the work of an artist, or is this merely a technical task akin to the work of a printer who pulls an artist's etching? Of course, anyone who is trained to program a computer can write an interactive program, just as anyone can daub paint on canvas or write notes on a page. But in the case of interactive programming it is easy to separate the artist from the craftsman.

One way to see the level of creativity necessary to write a program that successfully anticipates the potential inputs of a user/audience is to compare such a program with an ordinary case of communication. Imagine two bored teletype operators on a slow night. They begin a conversation over the teletype system, each responding to the other's queries and remarks in an ordinary and unexceptional way. One types, "Do you have any kids?" The other types, "Two, a boy 10 and a girl 13." "Oh brother, a teenager. You're in for some interesting years." "Yeah, but she's a good kid. How about you, have any kids?" And so on. Of course, a computer programmed to seem like a flesh and blood teletype operator could respond, too. The responses of such a program would be convincing to the degree that the programmer anticipated questions. There would have to be a default response for surprises — an all purpose, vague, but convincing reply.

Now, consider a game (called the Turing Test) in which a teletype operator is asked to determine whether the conversation is being held with a computer or with a human being. One obvious strategy would be to formulate questions that would force the default response. "Do you ever visit with your grandmother's next door neighbor?" or "When you were in high school were penny loafers in fashion?" A number of "Gee, I don't remember" responses would begin to make the operator suspicious. As the sophistication of the program increases, however, so does the range of possible replies to potential conversational gambits. The default response occurs less frequently, making detection of the computer more difficult. A program that is so sophisticated that the player of this game could not distinguish the computer's responses from those of a human being represents the level of sophistication needed for interactive art. Preserving the illusion of conversation in an interactive program is no more a "mere technical skill" than are orchestrating, writing convincing dialogues or to handling perspective. Interactive programming can become a part of any artist's repertoire and, just as with any other technique, such programs can be used to express an artist's intentions. Further, using such techniques can also help to realize an important artistic goal. As Stephen Wilson puts it, "Contemporary art has for many years been exploring ways of breaking down the barriers between artwork and viewer and between performer and audience."

But it is here that the possibility of interactive art presents an interesting perspective on some ongoing debates in aesthetic theory. It is one thing to agree that something counts as an art form, it is quite another to agree on appropriate critical standards. How would we evaluate an interactive piece? If I am an artist using interactive techniques my intention cannot be ignored or obfuscated. If my goal is to write a tragedy and my audience is amused, then I must anticipate this possibility and write the program in such a way that the amusement is changed to sympathy or sorrow or dread. And my skill as an artist will be reflected in the subtlety with which I can bring about this change. Further, although I may be able to dismiss a noncooperative audience as a pack of soulless, bourgeois philistines, if nobody responds appropriately to my work, then I am incompetent. (This result would be especially powerful if we introduce the biofeedback component mentioned above — if actual physiological responses to the work could be measured and if the results of this measurement could be read into the computer. Note that this would also reveal both audience perversity and pretentiousness, for it would no longer be possible to pretend indiffer-

ence when genuinely engaged or to feign rapture when experiencing idiocy.)

One standard of criticism, then, would have to be the intent of the artist. An incompetent artist would be unable to hide behind academic blather when a work fails. If my tragedy amuses or bores everybody and I have failed to write my program in such a way that this amusement or boredom is dispelled by subtle modification in the story line, then I ought not to be writing tragedies. If my drawing is intended to conjure the emptiness of modern life and everyone who sees it is sexually aroused or angered or embarrassed then only some fancy verbal footwork ("That's what I intended all along, no matter what I said earlier.") will keep me from looking silly.

There is a way to avoid this intrusion of the real world. As an artist I might say that I don't care about the emotional state of my audience -- that the reaction of the audience is irrelevant to my work. In creating my work I had no intention, no goals, wished to express no emotion, had no interest in the response of my audience. It is just a work of art. To place so much emphasis on my intentions and the audience's response badly misrepresents the true nature of true art.

Now, this may well be an appropriate attitude for any artist in any medium, but it is a bit harder to maintain for a work that depends upon a relationship with an audience in order to exist. It doesn't take much imagination to see that the solipsistic conception of art is irrelevant to both the audience and the artist. If any position about art needs to be argued for it is the view that anything at all counts as a work of art when someone who calls himself an artist has created it or someone who calls himself a critic praises it. This attitude trivializes the arts and it is this trivialization that makes the scorn of the ordinary observer -- of "most people" -- seem justified.

Of course, there are important non-artisic considerations that play a role in artistic judgments, and the powerful illusions possible with interactive art may not appeal to those in the art establishment who have an interest in preserving other illusions. For such people, perhaps computers are best left to bankers and engineers.

[My appreciation to Erika Doss of the University of Colorado and Katherine Varady of Johnson County Community College for their helpful suggestions on art and computers respectively. They both tried to keep me from writing anything too patently foolish.]

AN APPROPRIATE TOOL AT THE APPROPRIATE TIME

LESLIE B. DERBY III

IMAGEBASE Design Systems

Graphic designers, who manage a process, are dependent upon a range of trade services. The introduction of easy to use personal computers lead to the advent of "desktop publishing," and the mentality that design is the product of the machine. The problem of desktop publishing is not with its product, but with its process. While non-designers are wondering why their printed pages don't communicate as well as those produced by graphic designers, the computer is becoming accepted as a practical design tool. Those designers who would thus disclaim computers as valuable should realize that they are merely tools.

Introduction

Designers, in all disciplines, are not mere manipulators of form and space, nor are they purveyors of a personal style. Designers are managers of a process. It is their responsibility to continually examine the methodology supporting this process and to choose tools for design tasks which will be most effective in their problem solving. Any designer should at least investigate using a computer as a process tool. Otherwise, they limit their ability to keep pace with the current state of design technology.

Technological evolution has always impacted very heavily on the communication design process. The odyssey of an idea from its inception to its distribution as information has been continually re-routed throughout the history of visual communication. Initially, the printer practiced graphic design as an integral part of his craft. As printing technology advanced, the various tasks involved in the preparation of printed communications became segregated into areas of specialization.

The graphic designer has become dependent upon a wide range of specialized trade services. These components are assimilated by designers to create the end product. For example, in order to produce a publication, the designer must commonly coordinate the efforts of typesetters, photographers, color separators, plate makers and others. This coordination requires knowledge of a complex set of specifications and working nomenclature. These skills have allowed graphic designers to remain an essential part of the communication design process.

The Estrangement of the Designer from Desktop Publishing

The introduction of easy to use personal computers, having the capability to generate and manipulate type and graphics, has lead to the advent of "desktop publishing." Suddenly, anyone with enough dexterity to push keys or slide a mouse is capable of graphic design. This misguided perception has been encouraged by the marketing campaigns of computer manufacturers and software publishers. In many cases, organizations that once relied on design services now belive that design is a product of the machine. By owning that machine, they would have a designer at their service.

In understandable retaliation, many designers shunned the computer and its product. Those who chose to explore this new medium were viewed as traitors to the cause by other designers, whose philosophical stance was initially reinforced by the computer's limitations and the poor quality of its output. These are still very worthwhile considerations in choosing the computer as a design tool, but as the technology has evolved, the limitations have disappeared, and the output now rivals the quality of traditional methods.

The problem of desktop publishing is not with its product, but with its process. High quality output alone is not justification for using the computer for design. The users of desktop publishing systems often lack sufficient design background to fully utilize the applications that the computer has to offer. Conversely, designers who are not familiar with the properties of electronic data, fail to see the benefits this technology offers their process.

Magnetic media and random access memory are environments in which design can be conceived, grow, and evolve endlessly. Because of the inexhaustible malleability of electronic data, there is a greater amount of equity in each step of the electronic design process. Even the fainthearted can ask "what if," take the most outrageous chance, and still return to the previous state. This ability to explore with minimal risk allows designers to experiment with many permutations, thereby increasing the number of possible design solutions. The investment in the computer-aided design process will be multiplied further by design expert systems.

Design expert systems will provide designers with the ability to impart their aesthetic sensibilities, thought processes, and philosophies to the medium. A design can live on, impervious to tampering by the production worker lacking the skills to make appropriate decisions about its final form. This concept is now manifested in the form of electronic templates. One software publisher now markets a set of templates for newsletters along with a manual of procedures. This is a rudimentary form of expert system.

The Computer as an Integrated Design Tool

The computer has the potential to be the focal point of the design process. In this information age of data communication standards, a computer terminal can link to a myriad of resources. On a local area network, designers can share information easily. Most importantly, they have, at one location, a comprehensive tool that allows extensive manipulation of raw visual and textual information and also replaces several outside services that have traditionally been necessary to execute design.

Information gathering allows the designer to gain a clearer picture of the scope of a visual communications problem, and can now be accomplished through more than just the keyboard. Optical character recognition, image scanning and telecommunications are important channels through which information can be obtained, and made part of the decision making process that a designer must take part in.

Manipulation of the acquired data into a well-defined hierarchy is the next phase of the design process. Software for generating new visual and verbal ideas, using previously captured information, are abundantly available to the computer-aided designer. Word processing has been a staple of the computer user's repertoire since its inception. Now this application, as well as the text files it creates have new importance to the designer. A standard for communication of text allows for its transportation without re-typing and the designer has the opportunity to influence the editing process. Graphic oriented software gives the designer more direct control over the manipulation of visual information, without the logistical problems of using outside services.

The generation of final art for publication becomes the last step of a streamlined process. Page composition software, and its ability to combine several forms of visual/textual information is the root of the desktop publishing revolution. Again, standards for the communication of a variety of data have been essential to the practicality of this type of software. Full-featured typesetting capabilities and the equivalent of a photostat camera on the screen enable the designer to create and edit a nearly final design without time-consuming cycles of procuring these services. Another emerging standard, the page description language, allows artwork to be viewed as a proof on a plain paper output device, or sent as a text file over phone lines to a remote device.

The Symbiosis of Design and Desktop Publishing

The computer companies still shout their message of "design in a box," and some users are still listening as they wonder why their printed pages don't communicate as well as those produced by graphic designers. Designers are having the last laugh though. The visual

garbage that has been flooding the mainstream of printed communication is serving as divine counterpoint. Those who cannot afford to be party to poorly designed communications are realizing that the services of the design professional are essential to their efforts.

The "look what I can do with a computer" mentality of desktop publishing has nearly run its course, and as the computer is becoming accepted as a practical tool for the design process, graphic designers can now safely consider its use without being ostracized. No longer are their motives for using such an unorthodox tool being questioned.

To those graphic designers who disclaim computers as valuable to the design process: If an accountant refused to use anything but a pencil and manual ledger, or a writer, nothing but a quill pen, their purist's stance may seem quite noble, but can they be expected to produce at a rate that approaches today's standards for the work they do? This is not to say that an accountant cannot function without an electronic spreadsheet, or that one cannot write without a word processor. They are merely tools.

Presentation Graphics

Frank Kulesa

School of Art, Northern Illinois University

Abstract

Within the last several years a large market has developed for using the computer to generate presentation graphics. Unfortunately may users of presentation graphics software do no possess graphic art skills, mistakenly assuming that the software will produce high quality graphics without much effort on their part.

This paper hopes to give the untrained person in graphic design some introduction into basic guidelines for producing effective presentation graphics. This paper is only a start, hopefully the reader will do more research into the rich area of graphic design.

Introduction

Due to technological innovations in graphic versatility and sophistication of computers, video, and television; there has been an increased emphasis and influence on using images for communication. Graphics and pictorial images are beginning to hold an important position in our society. Unfortunately the typical audience is impressed more by the medium than the message. New software and hardware has made the availability of images easier, allowing graphics to be created in minutes instead of hours or even days. Using the computer to assist in preparing presentation graphics, offers great potential for improving communication of complex material. There is the danger that the user of this new tool may develop a presentation that the viewer may not understand, or even worse, be repelled.

Unfortunately some of the best examples of what not to do, come from the software vendors of presentation graphics. The images are usually too complex, taking away or hiding the presentation's desired message. Vendors emphasize ease of use, special effects, or just useless capabilities, that distract from the real issues of output quality. The user of presentation graphics software must be able to judge its graphic art qualities of typography, color, and composition. The software should allow easy movement from one type of format to another, allowing the user to display data in professional looking hardcopies or animations.

The computer offers the ability of a medium that is cognitive to both verbal and visual output. The designer must work in harmony with the computer to produce clean and effective graphics; graphics that will not create obstacles to effective communication.

Many individuals using graphics software have little experience turning computer data into effective hardcopy. One must know what the graphical goal of the work should be. To create effective presentation graphics, one has to have some knowledge of design and color principles. Typically the user is not a trained graphic artist, and the software will not turn the user into one. Although the computer has automated certain design tasks, it does not mean that design principles have changed. Besides learning hardware and software operations, one must be able to judge the capabilities of a graphic system to decide if it produces the desired output.

General Graphic Uses

The main reasons for using presentation graphics is to support the speaker by emphasizing important points, and to persuade the viewer to the speaker's viewpoint. Basically presentation graphics takes volumes of data or verbal arguments (intangibles and abstractions), and compresses them into a graphic form that makes information understandable and acceptable to the viewer. Presentation graphics attracts the viewer's attention and guides the viewer through a desired path. A well thought out presentation will help the viewer make a decision from the presented information.

Some of the main presentation formats are; charts, graphs, flow diagrams, text list and tables.

There are many other applications that are too numerous to mention here.

Design

Text, images, numbers, and relationships are the content elements of presentation graphics.

The first steps that should be done in organizing these elements is to analyze, and prepare an outline to identify the key ideas that relate to the information to be presented. For better communication, organize the information into a logical pattern. This will help show a visual correspondence between ideas and their order of arrangement.

Too much information leads to a misinterpreted message. Do not overcrowd the composition, (especially slides and videotex). Trim down the information to its essentials, and adapt a formal guideline with the information organized into an order of importance. A common misuse is to try to show all of the data relationships within one composition, break up the information into subcategories.

Design presentations that respond to the problem and fit the purpose, so that the level of the graphics is equal to the level of the decisions that will be made from the presentations.

Maintain a consistent graphic design through out a presentation. Color, text, shapes, and the overall position of items should have a continuity, but be careful each frame of information does not look the same. This will cause the presentation to become boring. Elements of size, proportion, thickness, color, dimensions, as well as spacing are some of the variables that can be used to enhance and vary the message.

Use specific formats to describe specific data (grids, visual hierarchy and focal points) to make the graphics and text have a consistent appearance. Some type of visual guide can be setup to build continuity. Items could be positioned on some vertical and horizontal axis. Another method used widely is to develop a grid structure for the overall presentation. Common graphic elements (titles, logos, company name, symbol, graph legend, etc.) can be placed in specific locations though the entire presentation.

Emphasis

To emphasize a statement, make that statement a different color. This statement will stand out because of the color differentiation. Color differentiation is also used widely for emphasizing graphic elements in charts. The viewer's attention can also be directed along a visual path by using color differentiation through the presentation.

Place a statement or shape by itself, any item surrounded by a large area of negative space will command attention. If space is at a minimum, then the use of ruling will help distinguish shapes and text.

Do not get carried away, only emphasize what us needed. There is a limit on how many items can be emphasized is a single frame of information.

Accuracy

People generally make an assumption of trust in presentation graphics. Unfortunately you can mesmerize an audience with flash and color to distract them from your true meaning. There is a trade-off between accuracy and special effects; 3D, shadows, size and color can mislead an audience. Graphics are deceptive, a small change in scale can accentuate a big change in the actual data. Lying with graphics by distorting scales, and by chosing data selectively is quite possible. Credibility is based on certain psychological factors, (layout, size, color, type of chart, type of font, etc.). Accuracy of data is number one in importance, and the designer must work with only creditable data.

Presentation graphics should accomplish certain functions. The presentation makes the key point or answers the key question that the audience is asking. It does not use out-of-context data or color. The information is highly legible, and data is offered as well as graphics.

Consistency

Provide a common ground of expression by working with elements that are the same or similar in structure. This will produce a consistency in the way elements are presented. Use the same symbols for particular objects or ideas. With text, use the same spatial indicators and font families.

Text

The major portion of all presentation graphics is in the form of text. It is necessary that a text arrangement is easy to read and understand.

Text is generally used to describe or explain a process, concept, or observation. Text is used with or without graphic aids. When text is used without graphics, it is referred to as a Visual. Visuals are generally reduced into a few key words or lines arranged into an outline form of lists or tables.

Horizontal aligned type is the most readable. If type has to be placed vertical, tilt a line by 90 degrees so that it reads from bottom to top. Do not stack letters, this is the least attractive arrangement.

Line length affects readability, and eye flow. A good rule to follow is about 1 1/2 times the alphabet. This will allow about 5 to 7 words per line. Use upper and lower case letters. Do not use type that is too large; this will cause the type to become a separate shape, distracting from the information.

For emphasis typographical arrangement may be done by indentation, line separation, bullets, stars,

checks, and boxes. A change in text size, weight and color, or the use of iconic or graphic devices will also emphasize recognition.

Text used in flow diagrams, organizational charts, or instructions, will describe a process flow or an order of importance. Diagrams and charts can describe activities that occur through time (temporal information) or over space, (spatial information).

Use clean and simple font faces. Stay away from text that have fancy or intricate designs. Use no more than 2 to 3 different font faces in a presentation. If boldface text is used, make sure it does not distract from the information. Sometimes italics can be used to accent a text item, but be careful that it can be read in smaller point sizes. The largest text should be used for the titles.

When dealing with the same subject matter use the same font faces, sizes, colors, and format. The viewer's attention and reaction will change if fonts change through a presentation.

Color

How Color Improves Readability

1. It improves visualization by accelerating communication.
2. Adds impact, and clarity.
3. Provides atmosphere and continuity.
4. Increases the speed of comprehension.
5. Improves the aesthetics of the layout.
6. Direct the viewer's eye.

General Rules

Use color with restraint and reason to emphasize important elements, do not use it as a decorative device. Keep colors consistent and distinguished. Be aware that color affect size relationships to other surrounding colors and to the background. Warm colors cause shapes to appear larger and closer than cool colors, which seem smaller and appear to recede into the background. Color will make the audience focus on structure, movement and details. Because color has a high memory value, we remember information by associating a color to it, therefore a composition's most important element should have the brightest color, with the greatest contrast to the background. The number of colors must be restricted to the least required to represent the message. The more color used, the more chance for confusion. Keep colors consistent between presentations. By using similar colors, parallel links of information can be encapsulated.

It is possible to have five to six colors in a presentation. Color can be used as a design element and for coding. You can highlight, separate, define, and associate images to organize information. Do not select colors on your intuition or personal taste. Be sure to avoid colors that cause optic vibrations along text and graphic edges.

Attention

The main use of color is to attract the viewer's attention. This can be accomplished by contrast between design elements of the greatest significance. Discretion needs to be practiced when using color contrast, cluttered or loud colors will distract from the message. Contrast in color values (light versus dark) is greater than contrast in hues (blue versus yellow). Warmer colors (red, yellow, orange) are higher in visibility than cool colors (blue, purple, violet). The darker the background, the brighter a color appears against it. The repetition of certain colors will produce a rhythmic sequence to help guide the viewer's eye through a presentation.

Text

The colors chosen for titles and text should be among the brightest colors. Incompatible colors make text hard to read and images hard to distinguish from each other. White, light blue, yellow or green works best on a black or very dark background. Use black, red, or blue only against a light background. The larger the text size, the lighter the color it should be.

Background

Black, blue black, or dark gray work best, (especially slides). Use only light colored details against dark backgrounds. The background color is the single most important color in determining a color scheme.

Slides

Black, dark blue, dark green, and dark gray are the most effective background colors. Yellow, light blue, and white are recommended for the text. Stay away from pinks, reds, and any gratuitous colors. For images or charts, oranges and cool colors are preferred.

Present the essential message first, a slide should have a single message that is easy to grasp, use more than one slide for complex data.

Keep the title as short as possible and place it in the composition so it is not too large or in an awkward location.

Text slides should be in an outline format, use standard upper-and lower-case text with consistent color and format.

Transparencies

Light-colored backgrounds work best, while images, details, and text should be dark. Transparencies should closely follow the design priciples used to design slide presentations.

Associations

One should remember that a color scheme is an abstract concept, and that certain colors emit different emotions from different people and different professions. Choosing the wrong colors to represent certain information will repel the viewer. Also the meaning and purpose of a color will change significantly from audience to audience, and differences are not just a matter of personal taste. What will influence color moods or color connotations is the fundamental purpose of a presentation. The below table gives some of the basic moods and connotations associated with different professions.

	Movie Audience	Financial Managers	Control Engineers
Blue	Tender	Reliable	Coolant
Green	Playful	Profitable	Safe
Yellow	Happy	Important	Caution
Red	Exciting	Unprofitable	Danger
Purple	Sad	Wealthy	Hot
Light Green	Leisure	Subdued	Steam

Designing For the CTR and Slides

Organization

Because most CTR screens have low or limited resolution, maximum viewing distance is about twelve times the width of the screen. Therefore text and shapes need to be simple and bold for strong contrast and readability. Consider the target audience, and choose only pertinent material to work with. The designer must decide what the precise objective of the information is, and structure the information into a logical order to convey an order of importance. The designer must decide how well the text and graphics communicate the information, not how pretty they are. Use simple graphics that convey a single idea at a time, brief supporting points are the most effective. Try to express only one idea at a time with no more than six lines, with six words per line.

Structure

Work within the limitations of the medium. Selection of the presentation medium will depend upon the nature of the graphics, audience size and the viewing distance. Break a large subject into main categories, then sub-divide each subject into smaller categories if needed. Be concise, focus upon simple words, short sentences, and paragraphs, describe (who, what, why, how and when). To produce a consistent presentation, items should be like in style (shapes, colors, fonts, size, etc.), this will allow for easy identification of the subject matter and message.

Typography

Text will probably carry the bulk of the information. Use upper and lower case text placed on simple backgrounds. Differentiate between main topics and sub-topics by changing the weight and size of the text. Headlines and capitalizing can be used to emphasize the subject matter.
Do not forget to use correct spacing between headings, body copy, column spacing, and justification. Stay away from small type, and do not put black type on colored backgrounds.

Animation

The computer itself can be used for a presentation. An animated self running slide show on the computer screen, or on a large projection screen will capture an audience's attention.

Charts

When property constructed, charts, graphs, and diagrams are an excellent tool for presenting quantitative facts quickly. They are used to tabulate data for analysis, comparison, prediction, and presenting information. Charts, graphs, and diagrams fall into two categories: those used for purely technical and scientific purposes, and those used as advertising or presenting information for influencing a decision.

It is up to the designer to choose the correct format for the right purpose. Otherwise the audience will not receive the correct clues to make a decision.

Composition

A chart should be a harmonious and balanced composition of shapes, titles, labels, legend, color, and text. Relate all elements so that charts function as a whole. Give the most important element special emphasis, without disrupting the chart's equilibrium. A chart should be clear of any element that is not essential to its message.

Text and numbers should be positioned to be read from the bottom and from the right side of the graphics.

Large graphic areas and dark colors should be at the bottom, moving up to lighter colors and smaller areas.

The title of a chart is important, and its wording has to be clean and concise with a description telling what the chart does.

Axis lines that serve as guides, should be thin and darker than the text or graphics. Always place the independent variable (time) on the horizontal axis, its scale should increase from the left to the right. The dependent variable (data being studied) goes on the

vertical axis, and its scale increments should increase from the bottom to top. Axis lines should be drawn heavier then grid lines, but lighter than lines used to plot the data.

Keep measurements even when choosing scales and grid divisions.

Place the legend where it complements the graph and is effectively color coded. For readability, the colors should not be too close in hue and intensity.

Care has to be taken in using the added perspective of 3D, otherwise data becomes distorted.

In multi-chart presentations, use the same size fonts for charts of equal importance, and for elements of equal importance within a chart. At times more than one type of chart can be combined for explanatory functions. Placing two line charts to show comparative trends or combining charts such as using pie charts on a map can also be effective. Each complicating factor however must be evaluated to avoid ambiguity or inaccuracy.

Major Types

Organizational Charts

Used for updating and reorganizing information. These charts can be produced into a variety of formats showing a relationship of parts to a whole.

Column Charts and Bar Graphs

Used to show the change of a single or similar items over a period of time.

Pie Charts

Used to show percentages between similar items in terms of their relative contribution to a total.

Area or Volume Charts

Used to show change over a period of time. This type of chart is the most deceptive among the other methods of representation.

Line Charts

Used to show developments of multiple items of data over time. Very useful for showing trends.

Scatter Charts

Used to show a relationship or correlation among data that coincide. Usually two sets of data is plotted and a line is drawn that shows the highest correlation between the data.

Pictorial Charts

Used to add interest but usually distracts from accuracy. A pictorial chart shows comparisions among products, population, costs, etc. Usually silhouettes of the items representated are shown is heights or areas.

References

Bair, Turnbull, McDonald. The Graphics of Communication. 5th ed. New York, New York: Holt, Rinehart and Winston, 1987, pp. 128-135.

French and Vierch. Graphic Science and Design. 3rd ed. New York, New York: McGraw-Hill Book Company, 1970, pp. 709-719.

Mathews, Eleanor., " Design Guidelines for Information Graphics," *Proceedings of the Fifth Annual Pacific Northwest Computer Graphics Conference*, University of Oregon, Eugene, Oregon, October, 26-28, 1986.

Wyman, Peggy. " Effective Business Graphics." Computer Graphics World. June, 1984, pp. 39-42.

Morris, Gillian. " Database Standards For Vidotex." Computer Graphics World. May, 1984, pp. 21-26.

Keller, Kevin. "Using Color to Best Advantage." Computer Graphics World. May, 1984, pp. 53-54.

James, Gerald. " Color Use Abuse in Presentations." Computer Graphics World. May, 1986, pp. 119-122.

Samit, Morris. " Presentation Perfect." Computer Graphics World. April, 1987, pp. 37-39.

INTEGRATING COMPUTERS INTO THE INTERIOR ARCHITECTURE CURRICULUM

Kathleen R. Stumpf

Northern Illinois University

Abstract

This paper presents several areas of concern that control an effective integration of the small computer into the Interior Architecture curriculum. Underlying constructs of design methodology, the role of education, and directions for integration are considered within a holistic context of design research and application. A model is given for the use of the computer within the core courses of interior architecture.

Introduction

This paper provides a holistic view for integrating the small computer into the interior architecture curriculum. Marginal use of the computer in interior architecture has been explored. Educational institutions are in a position to expand the use through research of design methodology and the effects that the technology information of the computer may have on the design process in education and in practice.

At the basic level of thinking the computer can be a creative aid to the designer. To realize this potential, the designer must consider the capabilities of the computer beyond the standard accepted ways of mere production and analysis of preconceived drawings. To do this however, one must be familiar enough with the computer to feel comfortable to go beyond the accepted usage and invent new ways that it can be used. For developing this familiarity, students who have been exposed to the computer from the beginning of their design learning experience may be in a better position to explore the higher orders of the design process using the small computer. A computer integrated curriculum must be in place for this development to happen.

The Computer as a Conceptualizing Tool

"Drawing as thinking" is the process that designers use in conceptualizing new projects. Pencils, pens, papers, restaurant napkins all become spontaneous tools in this process. Yet, this attitude has not transferred well to the use of the computer as another type of drawing tool for creative thinking. Essentially its use has limited as an aid in drafting the finished drawings and in analyzing the effects of the design especially in the areas of energy and lighting. The premise for this paper is that this need not be the case. There is a strong need to recognize and understand the cognitive and affective consequences of the new information technologies on the design process. More exploration must be made on what the human mind/computer relationship can do that the human mind/hand relationship cannot.

In order to broaden the use of the computer to include the subjective "gray" stages of the design process where the actual concept development is taking place, the mind frame for perceiving the potential use of the computer must be enlarged. This is one area where the design profession could take greater advantage of computer technology than it currently does. The environmental designer is, in the twentieth century terms, a technological anachronism. Even though Eric D. Teicholz[1] from Harvard University discussed this deficiency in an article published in 1968, twenty years later the observation still holds true for interior architectural designers.

As fine artists realize the creative aspects of the computer including the serendipities that enhance the creative process, designers too must realize the computer's potential in the schematic conceptual process. Artists using computers as sources of imagery, the theme of the exhibition for this conference, may work this imagary into other media. This approach for computer usage would be similar to that of designers. Schematic imagery developed by designers on the small computer would be translated into visualization drawings and then refined into construction documents (both of which are presently done on the computer). The final step would be the realization of the original schematic computer imagery as a three dimensional environmental space.

The Design Process and Methodology

At present the small computer is used in three areas of the design process: project management and administration, design analysis, and production of technological drawings. In order to bring the use of the computer into the conceptual stage of the design process, design methodologies must be reviewed and revised. The philosophical issues underlying design methodologies should be examined in relation to the use of the computer.

George Banz[2] proposes that an initial effort would involve an attempt to modify some of the programming design activities for a project to fit the data-processing capabilities of the computer. The design determinant and constraint variables of a project could be redefined as the components combining to create a conceptual pattern from which a schematic image could be made.

Following Christopher Alexander's proposal for a "pattern language"[3] developed in 1964 at the Center for Environmental Structure at Berkeley, an ordered collection of these patterns would be incorporated into a "design unit" defined in terms of dimensions and relative location. Its

coherence would be derived from a complex network of links among individual patterns, resulting in a structure with the characteristics of a relational base. The computer would have a program to work with these relationships. The designer would make the creative choices between the patterns and design units presented.

An example where a type of these patterns already exists is in systems office planning. A design unit, or a kit of parts for a work area, is standardized for various levels of employee classifications for a corporation ranging from the secretarial work area to the managerial work area to the executive suites of the partners or owners. This data base of design units is used by designers for the space planning and specificationof large office spaces. Development in this area of programming could be expanded to create more "design form" data banks to be shared with the design community.

The Role of Educational Institutions

The university affords an environment conducive to the research necessary for the optimization of the use of the small computer in the design process. Traditionally universities have taken this role of being at the cutting edge in other fields. One of the responsibilities of an educational institution is to provide the necessary conditions for actively pursuing this role and for advancing the field of study, which in this case would be advancing the creative use of the computer in the design process.

This leadership role for the profession has not been fully realized in interior architectural design education[4]. The ability of future designers and their creative use of the computer is greatly affected by their design education. As a relatively new field, interior architecture is still searching for the best way to educate competent designers in both theoretical issues and pragmatic concerns of creating quality environments for people. A more concentrated effort should be made toward strengthening the leadership role of interior architectural design education.

Direction for Curriculum Integration

This leadership role would be enhanced by the integration of computers in the interior architecture curriculum. To support this role, the educational institution must provide for: a) the necessary planning for a computer-based design curriculum; b) financing (internal and external) for appropriate computer workstations and research areas to support this curriculum, c) the educational opportunities to build a computer literate faculty to offer this computer-based design instruction, and d) the time for faculty to do research, development, evalution, and dissemination of the information on the creative use of the computer in design. In the end, it will be the sharing of the information and understanding developed at many universities that will advance the greater integration of computers into the design process.

Graduate level programs in interior design and interior architecture are at a minimum in art schools. The graduate program can be a strong mechanism for advancing the creative use of computers in interior architecture. If this mission is left solely to the undergraduate programs, at best the educational institution can only meet the current demands in the field while trying to keep up with the established use of computers by design firms.

Computers are no longer a trend in society, but a technological reality that interior architecture programs cannot ignore. The traditional means of dealing with information is no longer adequate when designing for the large corporate environments of today. At present most interior design/interior architecture programs look to the field to find answers for how to deal with the changing design process. The present design profession is meeting this changing process with answers demanded by the situation rather than planned answers based on a body of shared critical research. Part of the objectives for research is to predict future needs and to explore the best ways to meet them. Interior architecture programs in art schools lack this research endeavor.

A more innovative and creative use of the computer involves both the undergraduate and graduate programs. It becomes a cyclic process where a student who has had a computer integrated undergraduate design experience can then explore its greater potential at the graduate research level which then in turn influences the instruction at the undergraduate level.

A Proposal for N.I.U.

Integration of the small computer into the curriculum of Interior Architecture at Northern Illinois University should take place across all levels of instruction and content areas in order to take advantage of building creative technological skills. The main computing facility available to the Interior Architecture students is a small MacIntosh laboratory. Other support facilities include an Apple II laboratory and a few I.B.M. computers. A phased integration will be necessary due to the limited number of both computers and computer literate faculty. Immediate integration will also be in those courses with methodology and patterns most related or sympathetic to the computer process, such as office planning and store planning.

The proposal presented here involves the full use of the MacIntosh laboratory in the Interior Architecture core courses. Each core course is listed below with suggestions for how the use of the computer can be immediately integrated. The new design concepts introduced in each course would be taught with an appropriate computer application on MacDraft C.A.D. system, MacDraw graphics, MacWrite wordprocessing, Microsoft Excel spreadsheets, Schema perspectives, and others that become available.

The design learning experience with the computer would be sequenced to build on previous skills. Course outlines would be coordinated with faculty cooperation and support. As funding permits, the computer laboratory facilities will grow larger and more comprehensive C.A.D. systems and data-processing programs will be acquired.

Art 217 Technical Drawing for Designers. New design concepts introduced: Multi-view drawings, orthographic projections, perspectives.

In this course, the student would be exposed to MacDraft C.A.D. program and the construction of two-dimensional and three-dimensional views of environmental space. While the students are first learning the concepts of graphic representation of space using plans, elevations, axonometrics, plan obliques, and perspectives, they will also have the computer as an aid in visualizing and constructing

these drawings. The learning of technical hand drafting skills will still remain an integral aspect of the course, but the computer will play an important role in the vizualization process of design concepts.

Art 303 Interior Architecture. New design concepts introduced: Programming, adjacency and other relational matrices, schematic space diagrams.

This course would introduce the conceptualization process of design along with the use of the MacWrite word processor and MacDraw graphic program. Those programs would be used to explore the relationships between determinant and constraint variables of the design form.

Art 304 Office Planning. New design concepts introduced: Corporate images and analogies in design methodology, sociograms, office space standards and planning, specification, and construction documents.

This course covers the problems of designing large office spaces. MacDraft would be used to set up a data bank of space standards and work stations to be used in the space planning process. The repetition of basic design units applies well here. Area calculations would be made on MacDraft for block diagrams and space planning. Microsoft Excel Basic would be used for setting up the specifications for the components of the office system used.

Art 305 Store Planning. New design concepts introduced: Store space standards, planning, and design, merchandise fixturing and display.

MacDraft would be used again for setting up a data bank of design units. This time it would involve design fixtures and feature areas that would be repeatedly used in the space planning process. The fixtures, other furniture, and area finishes would be specified on the spreadsheet of Microsoft Excel.

Art 306/Art 307 Architectural Design I and II. New design concepts introduced: Large scale building space planning and design with structural and mechanical considerations.

MacDraft would again be used to do the initial schematic design drawings including the area calculations and the construction details. We are looking into obtaining various analysis programs for energy and lighting considerations.

Art 308 Architectural Detailing. New design concepts introduced: Working drawing specifications and interior construction detailing.

Repetition of similar details would be done on the MacDraft program with analysis done in schematic drawings on MacDraw. Door, window, and room finish schedules would be done on the Excel Basic spreadsheets.

Art 309 Human Behavior in Interior Architecture. New design concepts introduced: Analysis of the behavioral aspects of environmental space.

A program for using a data bank of behavioral concepts is being developed at the University of Illinois. When this is completed we would like to obtain it for this course.

Art 407 Special Problems in Interior Architecture. New design concepts introduced: Bank design, restaurant and hotel design.

Advance use of the computer would be explored along while continuing to use the previously learned computer skills for design conceptualization, programming, and drafting.

Art 408 Lighting Design. New design concepts introduced: Analysis of lighting and color in interior environments.

We are currently considering purchasing a program for lighting analysis and calculations to be used in this course. A data bank of schematic drawings of lighting effects would be made using MacDraft.

Conclusion

In proposing a holistic approach to the integration of computers into the interior architecture program at Northern Illinois University, an acknowledgement has been made of the limited resources available for implementation. The attitude taken is not to become discouraged by this, but to maximize the use of the existing computer facilities within all levels of the curriculum. This integration will establish the importance of the small computer as an essential creative drawing and analysis tool for interior architecture education. This visibility will make it easier to seek out ways to continue that momentum through new funding (internal and external) for hardware and software and through promoting the development of computer literate faculty.

The issue of implementation is a critical one. Several complex problems must be solved. Appropriate computer work areas must be made available. Scheduling and controlling access to the computers and the software need to be worked out. Providing user-assistance to faculty and students also should be considered.

Computer integration and implementation is necessary if interior architecture and interior design programs in art schools are to take a leadership role within the design profession. In meeting the changing needs of a technological society, the design methodlogy of the past is not adequate. It must be incorporated with the data processing capabilities of the computer.

References

[1] E. D. Teicholz. *"Computer Graphics"*, Architectural Forum, Sept. 1968.

[2] G. Banz. *"An iterative approach to computer-aided architectural design"*, Journal of Architecture and Planning Research, 2, pp.187-200, 1985.

[3] C. Alexander. *Notes on the Synthesis of Form.* Cambridge, MA: Harvard University Press, 1964.

[4] K. Stumpf. *"Future role of Interior Design Education"*, Journal of Interior Design Education and Research, Vol. 8, No. 2, pp.38-39, 1982.

CYBERNETIC JEWELRY: WEARABLE MICROSYSTEMS

VERNON REED

4407 SINCLAIR AV
AUSTIN TX 78756

ABSTRACT

The author presents a revolutionary new kind of jewelry which employs an on-board microcomputer to generate images in an LCD panel or other display device, making it unique in being defined by software, rather than hardware, considerations. This new jewelry form is analyzed in terms of nested systems, with discussion of each system's organization and integration with other levels. Finally, current trends and possibilities are extrapolated to conjecture about possible future evolution.

INTRODUCTION

Since the dawn of history, it has been possible to characterize virtually all jewelry in terms of hardware of some kind. This hardware has taken a tremendous number of forms, from seashells strung on grass to diamonds set in platinum. We have seen jewelry objects made in every metal imaginable; in cast plastic and folded paper; in ivory, slate and wood. Lately, we have seen electronic jewels made of integrated circuits, powered by batteries and driving liquid crystal displays and light-emitting diodes.

What all these various kinds of jewelry have in common is that in each the aesthetic entity is defined by configurations of hardware, and changing the visual aspect of that entity requires reconfiguring that hardware in some way. This reconfiguration can involve substitution of materials, rearrangement of parts, alteration of surface texture, etc. However, recent developements in microcomputer technology allow for a radical new possibility: cybernetic, or software-intensive, jewelry. In this scheme, a significant part of the aesthetic entity is defined by a software program running on the jewel's internal microcomputer and output to an appropriate display device. This software component is the primary feature which distinguishes cybernetic jewelry from all other jewelry forms.

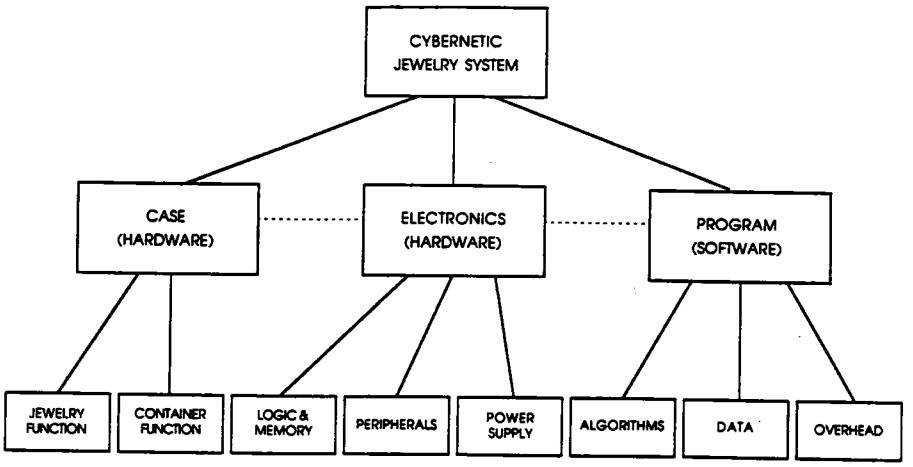
If we consider the aesthetic entity which comprises the jewel to be determined at least to some degree by software, rather than hardware considerations, then such a jewelry piece can assume some rather radical and revolutionary properties. Most obviously, if a software program is determining how a jewel appears, then the appearance of that jewel can be altered by loading a different program into the jewel's computer memory, without changing the hardware aspect in any way. Alternatively, if the software is written in a modular form, it is possible to achieve totally different effects by directing the computer to execute at different locations in memory. I have used this strategem often in my first generation of cybernetic jewels. Also, the interactive nature of computer technology makes it fairly simple to use data input from external sources, such as switches, keyboards or sensors, to modify the execution of the program, and thus the visual appearance of the jewel. Visual output in such a system could be determined by brain waves and barometric pressure, for instance, combining to alter program execution.

Thus, we see that the application of computer graphics technology to electronic jewelry systems allows a unique redefinition of what constitutes the jewel. This new definition points the way to an information age jewelry aesthetic, where software becomes the primary medium.

The decision to employ microcomputers in jewelry requires the artist to adapt to a new way of working, and to tailor his/her aesthetic vision to the capabilities of the medium. Inevitably, as with any medium, much time and effort will be required just to find what those capabilities are, as well as any limitations. One very basic factor limiting system complexity is the wearable nature of the end product, which means that component size and parts count are always critical issues. Due to the complex nature of microcomputer technology, I cannot overemphasize the importance of finding qualified people to form a developement team; I usually work with one or two engineers, when I am able. Time spent "reinventing the wheel" is time wasted, but often economics force such waste.

A SYSTEMS ANALYSIS

It is useful to think of cybernetic jewelry in terms of nested and interrelating systems, a mode of thinking which provides us with a way to manage the compexities involved in analyzing and creating such work. All cybernetic jewelry systems are made up of three main subsystems, two implemented in hardware and one in software; these can be further divided into simpler subsystems (Fig. 1). There are tools and techniques necessary for operating at each level.



(Fig. 1) Systems organization of cybernetic jewel.

Case system (hardware)

The first, and most basic, subsystem is the hardware system which I call the case, to borrow a term from watchmaking. This system would constitute the entire jewelry object in standard terms, in a cybernetic jewelry system, however, it serves primarily to contain the other subsystems and to mediate their interactions with the wearer's body. The case is made using materials and techniques widely familiar to metalsmiths and jewelers, but probably less so to computer specialists.

The case has two levels of function: as a jewelry object in its own right, and as a container for the electronic and software subsystems. The physical configuration of the electronic circuits can have a great effect on the size and shape allowable for the case, and preconceived aesthetic qualities of the case can influence the nature of the subsystems which it can contain.

The case can be made of almost any material, but regardless of its composition, certain minimum requirements are necessary to insure proper functioning of the other subsystems. It is important to maintain tolerances on interior cavities, so that electronic components fit without binding or shifting. It is often necessary to route wires through the body of the case, and care must be taken to provide holes and channels for this purpose; when using metal in a case, be careful not to short-circuit electronic components or wires. Access must be provided for such things as changing batteries and repairing or replacing electronic components. And all the while, keep in mind that the final result is meant to be at least nominally wearable.

After years of experimentation, I have evolved a method of case fabrication which easily meets these criteria. This is a plastic and metal laminate, in which all interior cavities are cut from acrylic sheet before the metal is laminated to the front with silicone adhesive. The metal plates which cover the back are held on with screws to allow easy access to the interior. The metal I use is titanium, because it is light and is ideally suited for producing graphic images. I use silicone adhesive to prevent the electronic components from shifting, and to cushion them against shock; it, as opposed to epoxy, can be easily cut if removal of parts is desired.

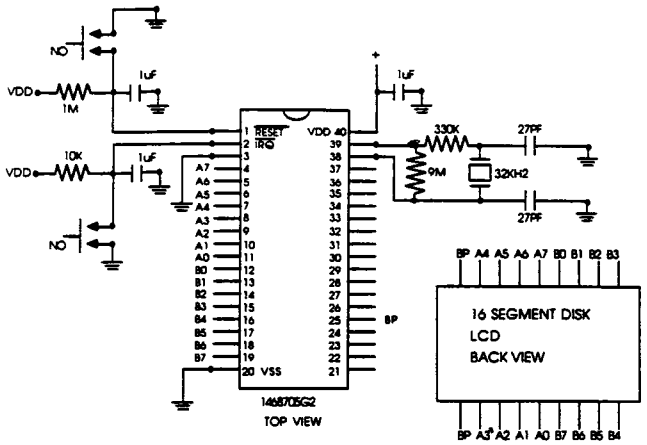
Any materials-working technique which allows for creating internal cavities would be suitable for creating a cybernetic jewelry case. Possible candidates include: die-formed, machined, electroformed or cast metal; vacuum-formed, machined or cast plastic; cast paper; carved ivory or wood; etc. Information on all such techniques is widely available through the Society of North American Goldsmiths and other sources.

Electronic system (hardware)

The point at which cybernetic jewelry diverges widely from standard jewelry techniques is at the second main subsystem, also in hardware. This is the electronic system (Fig. 2) and it is composed of three main subsystems: logic and memory circuits; peripheral devices; and power supply.

Logic circuits, as I use the term here, are the electronic circuits which perform calculations and generate signals. They are at the heart of electronic, as well as cybernetic, jewelry systems. In microcomputers, the logic circuits have the added advantage of being programmable, so they can be configured to perform a wide variety of tasks.

Memory circuits are used to store the software program and any data needed to generate a desired output. Memory circuits are of two kinds: volatile memory or RAM (random access memory), which retains information only as long as power is applied; and non-volatile memory which retains memory even if power is removed from the circuit. The kind of non-volatile memory we shall concern ourselves with in this paper is EPROM (erasable, programmable read-only memory). This kind of memory circuit is



(Fig. 2) Schematic of MCU/LCD module.

necessary for jewelry systems, as it lets the artist enter the desired program into the memory and also deliberately erase and modify it, if need be. Once stored in EPROM memory, however, the program cannot be lost due to removal of battery power.

The fact that cybernetic jewelry is a practical possibility is due to the advent of one very specific microcomputer chip— the MC1468705G2. This microcomputer unit (MCU) is the first one to combine on one chip the following properties necessary for use in jewelry systems: low power CMOS technology, which allows it to run for months on a set of batteries; EPROM memory (see above); and a sufficient number of output lines to allow driving LCDs without external driver circuits. This MCU, made by Motorola Semiconductors in Austin TX, is truly the key that unlocks the potential of cybernetic jewelry.

The next subsystem of the electronic system is the peripherals subsystem. Peripherals are the devices used to channel data into and out of the MCU; as such, they are the interfaces between the human and machine worlds. The MCU itself has 4 bit-programmable input/output ports (I/O ports), whose 32 individual lines can be programmed to send or receive data. Most peripherals interact with the MCU through these channels.

Input devices are used to enter data into the MCU to alter the execution of a program already in place. One input device familiar to most people is the keyboard commonly seen connected to computers, other possible input devices are switches and various kinds of sensors. Due to the limited memory available on the 1468705, it is impractical to interface a typewriter-style keyboard; a more reasonable way to enter data would be via a hexadecimal keypad. Key entry systems of any kind, however, require extra overhead in software and hardware. An even simpler and easier way to enter data (from the MCU's viewpoint, certainly not the human operator's!) would be an array of switches, each connected to a line of an I/O port.

Sensors responding to environmental change can also be interfaced to the MCU through the I/O ports, but the output from these sensors must typically be first processed by circuits which make the data compatible for input. These extra circuits increase the dreaded parts count.

There is one way to modify the execution of a program without going through an I/O port. The 1468705 has an external connection called the interrupt-request switch. When this switch is closed, it forces the MCU to go to a particular location in memory and begin program execution there. Modular software, in combination with this switch input, provides a powerful and relatively simple way to give the cybernetic jewelry wearer control over the functioning of the system.

The peripheral subsystem of most importance to cybernetic jewelry systems is the output system, the system which displays to the outside world the changes which are taking place inside the MCU as the program executes. This is the system which converts the abstract notion of some aesthetic software entity into moving shapes and images which engage the human eye and imagination. It is the window, as it were, into the computer.

My primary concerns have been with visual output, although it is equally possible to have devices generating audio, tactile or other kinds of output. Video-quality output is a desirable, but presently unobtainable goal, so we are left to find the most suitable compromise we can. There are several possible technologies which can be considered for output devices, but only two are really practical: the light-emitting diode (LED) and the liquid crystal display (LCD).

LEDs are semiconductor devices which emit light when a current is passed through them. The fact that LEDs emit light is both an advantage and a handicap, but mostly a handicap. The obvious advantage is that LED output would be quite impressive in low-light situations, such as a bar or disco. In real life, however, low-light situations are far outnumbered by instances in which high ambient light levels make it difficult or impossible to see LEDs. Just ask any electronic jeweler who has had a piece displayed in a gallery with a nice, bright spot shining directly on it! An even more severe handicap of LED technology is the amount of power it consumes to emit all that light. It is a widely known fact among electronic jewelers that battery life of LED jewels is measured in days, or weeks at most, if the output is anything beyond the most banal flashing light. This makes it very hard, for instance, to properly display jewels which will be on view for long durations in gallery or museum situations.

LEDs have two advantages, however, which typically overrule all other considerations: they are readily available and they are relatively simple to work with. Discrete LEDs, which look similar to tiny light bulbs, can be obtained at any electronics store, where it is also possible to find arrays of LEDs in different configurations. Unfortunately, however, the 1468705 MCU will not drive LEDs

directly from the output ports. It is necessary to employ one or more external chips to generate the proper signal levels.

The other viable technology for output systems is the liquid crystal display (LCD). LCDs are glass sandwiches containing an organic chemical which acts as a light valve when electronically stimulated. As compared to LEDs, the advantages and disadvantages of LCDs are completely reversed; they do not emit light at all and have an extremely low power consumption. The more ambient light, the better LCDs look, and battery life of LCD jewels is measured in months or years, depending on the kind of circuit driving the display. There are not many aesthetically pleasing LCDs commercially available, however, and making them is difficult, albeit possible. They are not as easy as LEDs for the MCU to control, either.

The main advantage of LCDs, apart from low power drain, is the mysterious, video-like quality of the images flickering on a blank screen. LCDs are, in fact, now being used in Japan to make tiny video screens, and it is only a matter of time until this technology will be available to jewelry artists. In the meantime, LCDs are the closest we can get to the ideal of video output for our cybernetic jewels. I fabricate my own LCDs in my studio, and I use these exclusively for the visual output of my cybernetic jewels.

In keeping with the jewelry artist's constant concern with size and parts count, I have so far stressed those peripherals which can be hooked directly to the MCU. There is another class of circuits, called device drivers, which interface between the MCU's I/O port and an output device. Typically the output devices driven by these circuits have much more inherent resolution, and thus image complexity. For instance, an unadorned 1468705 chip can drive an LCD panel having 31 independent picture elements, or pixels; a multiplexed LCD driver, however, can control an LCD with 256 pixels, and do it with only 32 control lines. The fly in this ointment is that the LCD driver is in a package the same size as the MCU. At this time, this is an intractable problem.

An attractive alternative or complement to visual output is audio output. This is another relatively power-hungry output system, but one which could be simple to implement. One possible device to use is the kind of solid-state transducer which is used for the beeper in digital watches. The MCU cannot directly drive a device of this kind, so external chips would be required to provide the proper signal levels.

The final electronic subsystem to consider, and by far the simplest, is the power supply: basically this is going to be one or more batteries. The 1468705 microcomputer is designed to run on a voltage as low as 3 volts and, since power drain goes down as voltage drops, it would be unwise to power it with a higher than necessary voltage. When choosing batteries, the main choice is between rechargeable and non-rechargeable; there are advantages and disadvantages to each kind, depending mainly on the kind of output device used.

The best batteries for LCD jewels are the non-rechargeable lithium batteries used in calculators and watches. They produce 3 volts per cell and have more power per unit volume than any other battery; if very long battery life is desired, then it is possible to use more than one battery in parallel. The disadvantage of lithiums is that they are not designed to provide the kind of power output needed for driving LEDs. Rechargeable nickel-cadmium batteries are designed for high power output, and they do not have to be replaced, both desirable features for a system which uses relatively large amounts of power. They do, however, require more space in the jewel to store an equivalent amount of energy.

If one does decide to use rechargeable nickel-cadmium batteries, then there are several options for recharging those batteries. They can be charged in situ in the jewel by plugging a charger into the back. They can be removed and recharged in an external charger, after placing freshly charged batteries into the piece. Another alternative is to employ solar cells to constantly charge the batteries when light levels are sufficient. If this last option is employed, then the solar cells must somehow be integrated into the design of the jewelry object, a process I have never been completely satisfied with.

Program system (software)

The two subsystems of cybernetic jewelry covered so far would be sufficient to completely define an electronic jewelry system. For cybernetic jewelry, however, they only serve as a matrix or substrate for the third, and most important, subsystem- the program, or software, system. If the electronic system is the heart of the cybernetic jewel, then the software system is its mind. This is the system which embodies the radical new jewelry aesthetic I am working toward.

Like the other two main subsystems, the software system can itself be roughly divided into subsystems. These are the algorithms subsystem, the data subsystem and the overhead subsystem.

Algorithms are rules for manipulating data to achieve a particular outcome; as used in creating cybernetic jewels, they define a way for the MCU to display the desired patterns in the output device. These algorithms are programmed as sequences of instructions which direct the MCU along predefined paths as it processes data. The data subsystem consists of direct pieces of information which act on the program in an immediate way. Examples of data are pattern information and control information; when these are plugged into the logical structure defined by the algorithms, they are manifested as visual output.

The overhead subsystem encompasses all the repetitious background activities necessary to keep the cybernetic jewelry system functioning. Examples included the various timing loops used to keep track of program execution, and such features

as device drivers implemented in software, rather than hardware. The software which I use employs several nested levels of timing loops, each associated with a particular aspect of the program.

There are some significant differences between the program system and the other two cybernetic jewelry subsystems. Most importantly, while the other two can be completely described in terms of hard, physical objects, the software system consists entirely of coded numbers stored in the computer's memory and circulating through its processing unit. Computer programs are written in synthetic languages which correlate specific numerical codes with specific internal processes of the computer. At the most elemental, or "machine language" level, the program is expressed solely in terms of these numerical codes. A language level which preserves the close coupling between numerical code and processor function, but still provides for a minimal level of human-readability, is the "assembly language" level. This is the level most reasonable for developing software for the 1468705 microcomputer. Higher level computer languages, such as BASIC, Pascal, or C cannot be used, because the 1468705 lacks enough memory to hold such complex instructions.

I develope the software for my jewels using a cross-assembler running on an IBM PC. A lot of artists, myself included, abhor the kind of nit picking specificity required to program at the assembly language level; however there is a way around this problem. A clever computer programmer can use the macro function of a good cross-assembler to define macros as modules having specific effects desired by the artist. A framework for organizing these modules can then be programmed, and the result is essentially a custom "language" in which the "instructions" correspond to manipulations of the output device. This modular software system, which I use, would be a lot more intuitive for most jewelry artists than machine or assembly language.

The software system I use is also modular at a higher level. All my programs are organized as groups of macros, and each group is a self-contained entity which produces a particular sequence of patterns in the LCD. One radical result of such modular organization is that the visual quality of the jewel can be instantaneously reconfigured by going to a different group of macros and executing the instructions contained there. The interrupt-request switch is used to cycle from one group to another; typically my programs contain three to five such groups, with varying visual characteristics.

Systems Organization

The organization of these basic systems is somewhat more hierarchical than is indicated by (Fig. 1). The case system could stand on its own as a standard piece of jewelry, and could combine with the electronic system to become an electronic jewel, finally becoming truly cybernetic with the additio of the program system. The electronic system on its own, however, lacks sufficient capabilty to function effectively as jewelry, even when combined with the program system. This fact is a result of the peculiar requirments imposed by the wearable nature of jewelry, and it places final constraints on any attempt to produce a "pure" cybernetic jewel without hardware components.

The key to effective integration of the component systems is modular organization, wherever possible. This allows great apparent variety to be achieved from a relatively small number of basic elements. For instance, at the case level, the plastic container for the electronic system could be standardized, but variations of the metal lamination would be sufficient to produce a large number of very different effects. The electronics system is essentially transparent and can be totally standardized, with all variations in the display device being determined by the logical arrangements of modular software configurations.

The evanescent nature of these software contructs stands in marked contrast to the rigidly defined forms inherent in "normal" jewelry, and provides an avenue for evolution with possibilities that we can only guess at today.

THE FUTURE

It is, of course, impossible to predict future developements, but we can extrapolate based on current trends and anticipated breakthroughs. Probably the most important trend for cybernetic jewelers will be the incorporation of ever increasing amounts of non-volatile and volatile memory on the MCU chips. This increased memory space will allow for greatly increased sophistication of software. As memory size approaches the 8K mark, it should be possible to program jewels in inefficient, but easier, high level languages, such as BASIC or Forth. With a really dramatic breakthrough in memory capacity, we might see the possibility of employing sophisticated image-generating algorithms, such as fractal and recursive procedures.

Another path of MCU evolution will see the incorporation of more device drivers and peripheral interfaces onto the chip. One very desirable outcome of such developement would be provision of on-chip multiplexed LCD drive circuitry. Also desirable would be the inclusion of circuitry to transform analog sensor data into digital form suitable for computer processing. Output devices and external device drivers will grow in complexity, as well. High on any wish list for cybernetic jewelers is a driver which will produce true video and a panel which will display it.

At the far reaches of our conjecturing, we may anticipate such things as the application of artificial intelligence principles and non-invasive nerve signal receptors to create a "bionic paradigm", which will blur the distinction between the jewel and its wearer/operator. It is important to think about such things, because the future happens faster and faster, and we as artists must be able to interpret and moderate the changes when they come.

BIBLIOGRAPHY

1) ------, Single-chip Microcomputer Data, Austin TX: Motorola Inc, Second printing, 1984

2) ------, M6805 HMOS, M146805 CMOS Family Microcomputer/Microprocessor User's Manual, Englewood Cliffs NJ: Prentice-Hall, 2nd Edition, 1983

3) ------, M1468705 EVM Evaluation Module User's Manual, Phoenix AZ: Motorola Inc, 1983

4) ------, MC1468705G2 Advance Information, Austin TX: Motorola Inc, 1983

5) ------, LCD Technical Manual SI-100, Torrance CA, Seiko Instruments USA Inc.

6) ------, 6805 Macro Cross-assembler User's Manual, Aurora CO, 2500 AD Software Inc.

7) Bylander, E.G., Electronic Displays, New York: McGraw-Hill, 1979

8) Kallard, Thomas, editor, Liquid Crystal Devices, New York: Optosonic Press, 1973

INTERACTIVE COMPUTER-ASSISTED RGB EDITOR (ICARE)

Donna J. Cox, Visiting Assistant Professor
in the School of Art and Design
Adjunct Professor at the National Center for Supercomputing
Applications (NCSA), University of Illinois, Urbana-Champaign

Abstract

ICARE is a simple, interactive software tool created by an artist to generate and manipulate large subsets of RGB color space. This tool provides a systematic color mapping technique for computer-generated or digitized images. The user controls color through input parameters. Interactive control allows the user to create complex waveforms of color "sounds." Interesting applications include unique color maps for scientific visualization (SIGGRAPH '87 Film and Video Show) and an real-time graphics performance (International Computer Music Conference, 1987). ICARE was originally created and displayed on a high-end workstation and has been translated to a personal computer where it will be a part of the "Interactive Image" exhibit in the Museum of Science and Industry, Chicago. The basic concepts used for color generation and exploration are presented here.

Introduction

Since the advent of color monitors for computer graphics, scientists have concentrated on software or hardware dependent color models [1,2]. Historically, these models represent color in three-dimensional space, and a considerable body of knowledge has been accumulated concerning the psychological and physical responses associated with color interpretation [3, 4, 5]. However, this technical information is generally opaque to the artist.

Computer paint programs typically allow one to choose a single color and mix between two colors. This approach ignores a fundamental concept in color research: color and surrounding field interaction. Computer palettes should allow the user to dynamically create and visually examine the color palette as a gestalt background/foreground perceptual relationship. Another shortcoming of most paint programs is the fact that the artist is unable to interactively and systematically control the color space with continuous, mathematical functions. While most artists prefer that the numbers be transparent, many artists wish to systematically control the numerical input in much the same way as computer musicians numerically generate envelopes of synthesizer sounds . The typical paint program precludes mathematical functional input to control large array output [6].

ICARE is a simple software package which was created by the author for pseudo color mapping, art theory education, and scientific visualization. This software tool has been useful in creating unique color palettes and instructing artists in color theory. The approach is unique because the user systematically creates and manipulates, large ordered subsets of red, green, and blue (RGB) color space with simple parameters in a real-time interactive session. The numbers do not remain transparent to the artist; rather, they become a conceptual framework for aesthetic decisions in the creation of the pseudo-color maps. The color map (or palette) can be stored or immediately displayed in 256 pixel values of a full screen, high-resolution, 8-bit image. The user has the option to see graphs of the RGB intensities for extensive analysis.

Until recently, this type of color control has been relegated to expensive 8-bit color frame buffers and workstations. Now it is possible to purchase many color graphics boards which allow this RGB color capability on an inexpensive personal computer.

This dynamic interaction between user and computer exploits many concepts which bridge computer music and visual art. This color map manipulation is analogous to "creating a movement" through color space. The user metaphorically "maps a movement" from color space to the digitized or computer-generated image. This unusual capability is unique to the computer medium as a coloring palette of the 20th century.

Color Mapping and Digitized Images

Most computer frame buffers afford digital control of color. Cathode ray tubes create color through red, green, and blue phosphoric excitation. The intensity or brightness of the color

phosphors can be software driven in most of these display devices. RGB can be represented and stored as a triplet of numbers in computer memory registers. These triplets are arranged in sequential order and are referred to as color tables (also known as color map). Color table array elements are mapped to the raster computer screen coordinates through an indexing procedure within computer memory. Color table indices correspond to pixels (picture elements) of the raster computer display device. Each pixel addresses the RGB color table elements which control the red, green, and blue color components at that pixel location on the computer screen.

This color mapping is called pseudo color because the RGB values can be arbitrary; thus the color can change without changing pixel values. For example, Figures 1A and 1B are identical pixel values. The only difference between the images is the color mapping. Likewise, Figures 2A and 2B are identical pixel values. Only the color maps have been changed.

A computer graphics image is created with pixel values. One way to create an image on the screen is to write pixel values to the frame buffer which access a color table. Many times the pixels are written into the frame buffer as mathematical functions. Figures 1 and 2 were calculated on a supercomputer as numerical simulations of astrophysical phenomena [7]. Simulations generate images which reflect solutions to mathematical equations. The arrangement of the pixel values create the "form" or the shape of these images.

Another method to create pixel values is to digitize an image. Digitizing cameras (often referred to as "frame grabbers") transform a video signal into digital information which can be displayed on raster computer displays. This digital information can be represented as a pixel value range. Thus, a digitizing camera can capture an image through a camera lens and partition the light intensities into an ordered array of pixel values. This process is analogous to dividing a photograph into continuous tones and placing these tonal values into the pixel matrix of the computer screen. The computer manifests color when the color tables are mapped to the pixels on the screen. Each of the 256 tones of an 8-bit image has a corresponding pixel value which has an index to a color map. Figure 3 is a digitized photograph which has been mapped with color.

Interactive Computer-Assisted RGB Editor

ICARE provides a convenient way to apply simple mathematical functions to the red, green, and blue values in the color arrays and to view the results when this pseudo color is mapped to the screen. The frame buffer may hold a digitized image (Figure 3) or pixel values generated by other mathematical functions (Figures 1 and 2). A composite of the RGB color tables is mapped over the entire image as a single array of color. The user inputs simple parameters such as amplitude, phase, and frequency which control the intensities over each of the red, green, and blue color tables. At the end of each loop, the user is prompted with an extensive menu. This menu has many options which range from beginning to advanced commands, including Save, Dump, or Help.

The user creates complex waveforms by controlling a smooth, periodic function which is applied to each of the RGB color tables. Each RGB color table will graph a separate waveform of intensites over the pixel values. The algorithm is presented for 8-bit computers which can simultaneously display 256 different pixel values composed of red, green, and blue components (2^8= 256 possibilities).

x = integer pixel value
$0 \leq x \leq 255$

c[256] = red, green, or blue array
f=frequency
a=amplitude
v=vertical placement
p=phase (in radians)
scale = f * 6.283/256

PROCEDURE (calculate color array)

Repeat for x = 0 to 255
    c[x] = a * sin (scale+p) + v
    scale = scale + scale
    x = x + 1
End

Figure 4 shows the ICARE display screen where each of the RGB array values are plotted in the upper right quadrant of the display. The lower right quadrant shows four horizontal bars which display the separate RGB color table intensities, and the composite color map (lowest horizontal bar). The upper left quadrant is the text where the user is prompted and gives keyboard commands. The lower left quadrant is an image to which the color is mapped. Any image file can be copied into ICARE and mapped with color.

The composite color table in Figure 4 is the combination of the red, green, and blue color arrays. Such that,

r [ ] = red color array
g [ ] = green color array
b [ ] = blue color array
C[ ] = composite color array
x = pixel value

and

$$C[x] = r[x] + g[x] + b[x]$$

Thus, the composite color map is a 256-element palette which represents the addition of primary colors from the red, green, and blue arrays. The user can independently adjust any variable for each array while other variables remain constant.

The input variables are simple; yet this visual and numerical information is effectively used for instruction in additive color theory. Students with little or no trigonometric knowledge can quickly grasp the functional control of the arrays. This procedure provides many possibilities to use color as a interactive teaching aid [8]. The student's intuition is reinforced by immediate visual feedback and the innate attraction to color provides motivation. ICARE is a part of the "Interactive Image" 1987 exhibit in the Museum of Science and Industry, Chicago, Illinois, where museum attendees can interactively create and map color tables into NCSA images. This exhibit was organized by Tom DeFanti, Dan Sandin, and Maxine Brown, Electronic Visualization Lab (EVL), University of Illinois at Chicago.

Color "Envelopes" and High Frequency Maps

There is a metaphorical relationship between the rise in amplitude of a musical note and the rise in amplitude of the color array. In music, the amplitude relates to the loudness of the note. Likewise, in the above algorithm, the intensity or brightness of the color relates to the height of the amplitude: the higher the amplitude of the RGB color array values, the brighter the color. In other words, the ICARE sine wave generator controls the RGB intensities, and one can create color "envelopes" or "sounds" in an image.

Figure 1A reveals a high-amplitude, low-frequency color map. The amplitude peaks and reveals a brightness which is sustained for most of the image and decays to darkness at the edge. Likewise, Figure 1B reveals a high amplitude; however, the frequency has been increased by a factor of ten. This increase in frequency coupled with a high amplitude has created a series of light-to-darks in Figure 1B. Remember that Figure 1A and 1B are identical mathematical computations, and the difference which we perceive relates to the frequency change in the color maps.

Figure 2A has been colored with complex waveforms over the RGB arrays. These waveforms were generated using the addition of sine functions during an interactive session with ICARE. The amplitude and frequency combination in this color map reveals the form or morphology of this computer simulation of an astrophysical jet. The same image file is used in Figure 2B. However, the color map frequency has been increased by a factor of 16. The high frequency creates an interference pattern throughout the image.

Figure 3 is a digitized image which has a maximum color frequency of 128. The high frequency stratifies the color map into light and dark dots. This color map also has a linear trend

which has been applied to the RGB color arrays as follows:

```
c[ ] = precomputed color map

PROCEDURE (apply linear trend to map)

Repeat for x = 0 to 255
    c[x] = c[x] * x/255
    x = x + 1
End
```

Thus, the color map in Figure 3 is at minimum intensity where pixel values equal zero. The brightness of the color map increases as pixel values approach 255. This linear trend from low intensity to high intensity of the RGB values creates contrast in the photograph. This contrast retains the shape and form of the image. At the same time, the high frequency palette creates a "pointillist" visual effect within the photograph. Without the linear trend, the facial form would be lost in a field of "noise."

One can imagine a transition from a low frequency color map to a high frequency color map as a function of time. Thus, one might see a change in the color which "ripples" the images. A transition from Figure 1A to Figure 1B can be a continuous real-time animation change. Likewise, other variables such as amplitude and phase can be changed in real-time. These concepts were used to control color and perform a real-time computer graphics color "score" as part of the International Computer Music Conference, August 23-26, 1987, Urbana, Illinois, Krannert Center for the Performing Arts Center. The performance was an improvisation between two musicians, a dancer, and the author. The computer graphics images were projected on the stage with a light valve during the live performance with Paul Zonn, Music Composer, University of Illinois at Champaign-Urbana.

From Supercomputer to Personal Computer

ICARE was originally written in C-Programming Language for the Silicon Graphics 2400 Workstation. The simulations in Figures 1 and 2 were computed on the CRAY XM/P by research scientists at the National Center for Supercomputing Applications, and ICARE was used to pseudo color the images. Figure 2A is a single frame from a computer animation which was shown as part of the SIGGRAPH '87 Film and Video Show in Anaheim. This animation reflects interdisciplinary research between artist and scientist working toward a common goal - scientific visualization [9, 10].

A version of ICARE is now operating on the IBM AT with Number Nine Card or M8 TARGA Board using RT/1 Language [11]. The author thanks two former students, Steve Curtin and Brian Evans, for developing this version of ICARE.

The RT/1 Programming Language is designed by Professor Tom DeFanti (EVL). This graphics language has been distributed since 1986 and is an expanded version of the Zgrass language which was distributed and documented from 1981-1986 [12]. RT/1 language is public domain and distributed for a small documentation fee.

The author was inspired by early work at the EVL. In 1973, Professor Dan Sandin created the Sandin Image Processor, an analogue colorizer and IP for video and computer images. Later, DeFanti and Sandin interfaced the IP with Zgrass to develop low-cost, real-time graphics image generation and color manipulation [13]. ICARE is a colorizer which allows real-time, systematic digital control and mapping of color.

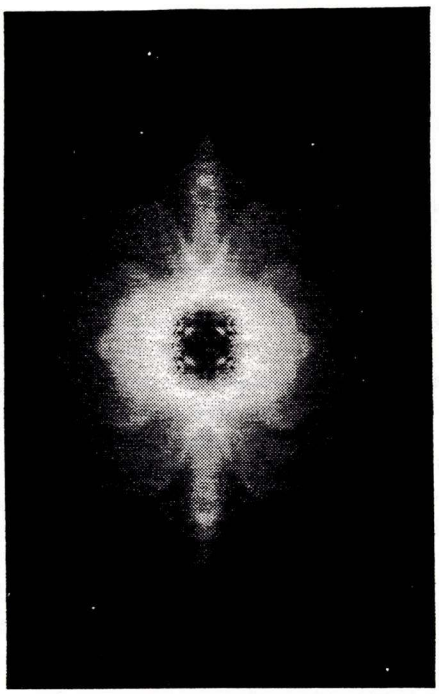


Figure 1A : CRAY XM/P simulation of a collision between two neutron stars, by Chuck R. Evans, visiting research scientist at the NCSA. The color map here is high-amplitude, low-frequency and is sustained over the pixels until it decays at the image edge.

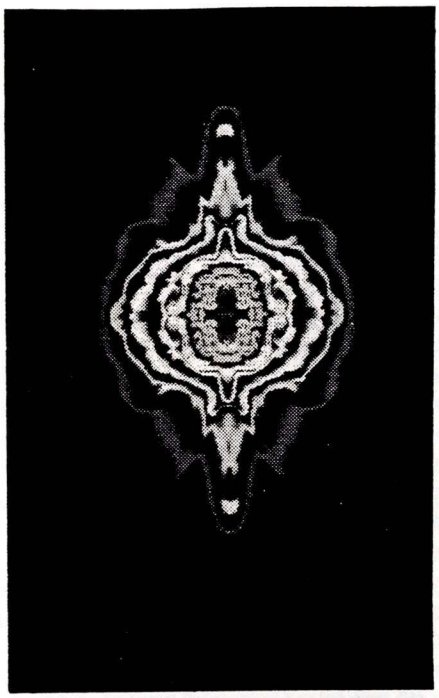


Figure 1B: Same pixel values as Figure 1A. However, the color map is high-amplitude and high-frequency such that a "rippled" effect is perceived. This figure was published in Time, "Pictures Worth a Million Bytes,"May 18, 1987.

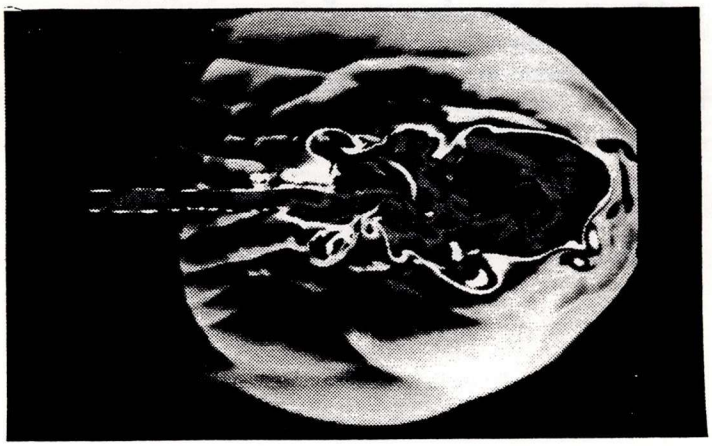


Figure 2A: CRAY XM/P simulation of the turbulent flow of an astrophysical jet spanning more than a million light years in length, by Michael Norman, research scientist at the NCSA. This figure is a frame from an animation which was shown at the SIGGRAPH '87 Film and Video Show, Anaheim.

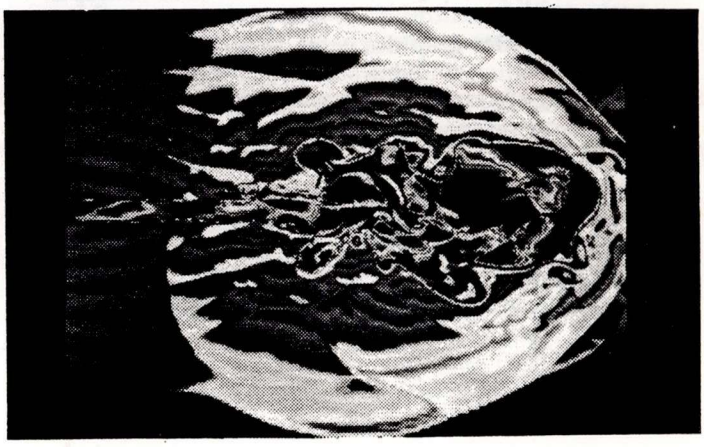


Figure 2B: Same numerical data as Figure 2A; however, a high-frequency color map creates an interference pattern over the morphology (or form) of the astrophysical jet by Norman.

Figure 3: The "pointillist" effect in this digitized, pseudo-colored photograph is created by a maximum-frequency color map. A linear trend has been applied to this map to increase overall contrast since extremely high frequencies degenerate the image into "noise."

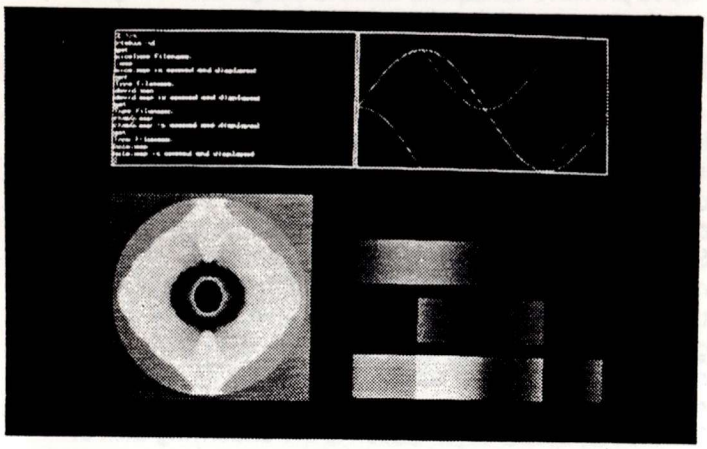


Figure 4: ICARE display menu and graphs. The RGB values are plotted in the upper right quadrant. The composite color table is the lowest horizontal bar in lower right quadrant. This composite is mapped to the image file shown in lower left quadrant. User input and prompts are in the upper left text window.

References

[1] Joblove, G.H., Greenberg, D.; Color Spaces for Computer Graphics, Computer Graphics, Vol. 12 #3, Aug. 1978, 20-25.

[2] Foley, J.D., Van Dam, A.; Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Co., 1982.

[3] Montalvo, F.S., Special Session on Vision, Computer Graphics (Siggraph '79), Vol. 13 #2, Aug. 1979, 121-124.

[4] Fish, J.C.; Color Sensation in Visual Art and in Science, Leonardo, Vol. 14 #2, 1981, 89-98.

[5] Kaiser, P.K.; Physiological Response to Color: A Critical Review, Color Research and Application, Vol. 9 #1, Spring 1984, 30.

[6] Giloth, C., Veeder, J.; The Paint Problem, IEEE Computer Graphics and Applications, July 1985, 66-75.

[7] Cox, D.J.; "Using Supercomputer to Visualize Higher Dimensions: An Artist's Contribution to Scientific Visualization", to be published in Leonardo, Volume 21, 1988.

[8] Meier, B.; BUCOLIC: A Program for Teaching Color Theory to Art Students, IEEE Computer Graphics and Applications, Vol. 246 #2, Aug. 1979, 121-124.

[9] DeFanti, T.; B. McCormick; M. Brown, "Visualization in Scientific computing," report to NSF, 5/18/87, to be published in ACM Computer Graphics Quarterly.

[10] ACM SIGGRAPH '87 Film and Video Show Catalogue, "Instabilities in Supersonic Flows," M. Norman, D.J. Cox, p. 12, p. 25.

[11] DeFanti, T.; Tanaka, K., "RT/1 Glossary, Lessons, and C Coding Techniques," self-published, distributed with RT/1 language, 1987, University of Illinois at Chicago, EE/CS, Box 4348, Chicago, IL 60680.

[12] DeFanti, T., Fenton, J., Donato, N.,; "Basic Zgrass," Computer Graphics, Vol. 12 #3, Aug. 1978.

[13] DeFanti, T., Sandin, D.,; "The Digital Component of the Circle Graphics Habitat," Proceedings of the National Computer Conference, 1976.

Smart Art:
Thematic Algorithms that Underlie Algorithmic/ Process Art

Dr. Bill Kolomyjec

Northern Illinois University

Abstract

An analogy between the components of algorithmic/ process art and fundamental design theory is presented. Graphics primitives are compared to primary visual elements. The video display screen is discussed relative to the two dimensional frame of reference. Rudimentary drawing algorithm are compared with spatial forces; main programs are compared with forms of spatial organization. The concept of Smart Art is introduced and discussed. Smart art embodies "criteria for rationalization" which in turn becomes the thematic algorithm that is responsible for the optical quality of an algorithmic/ process program. These programs are time based and have performance implications. Two smart art program examples for the Apple® Macintosh™ written in the C programming language are included.

Introduction

At the Sixth Annual Symposium on Small Computers in the Arts I described a style of computer graphics/ computer which I call algorithmic/ process art[1]. After another years' work in that style I propose to continue with that theme and discuss how algorithmic / process art has evolved into what I now call Smart art. However, it may be important to re-present a few key definitions before going on.

Algorithmic art refers to that distinctive category of aesthetic computer imagery generated by some special method or algorithm, i.e., via computer program. I showed how it was possible to use the software and hardware aspects of a small computer to continually generate computer imagery; imagery that can neither be classified as "static" nor as single frame animated (true animation) this is process art. Process art is computer art in the course of being done; the emphasis is on process not artifact.

In the paper associated with the presentation I gave three C language programming examples for the Apple® Macintosh™ to illustrate algorithmic art/ process art. First, I generated a static algorithmic image consisting of a"tiled" or tessellated surface using a static module. Second, I generated a static algorithmic image consisting of a tessellated surface using a module that varied with randomness. Lastly, I generated a motion graphic or dynamic image where a row of "bitmapped balls" was rendered on the bottom of the Macintosh™ display screen which then scrolled upward in such a way that the next row could be rendered. This algorithm was placed inside an infinite loop which would allow these operations to continue virtually forever. This example typified an algorithmic/ process art piece. In my conclusion I stated that in this manner it was possible to produce perpetual imagery; imagery where product and process had become integrated.

Since that conference I have had some time to experiment with this form image generation. In that time I found the need to examine the visual aspects of algorithmic/ process art in an objective and rational manner. I would like to share with the reader what I have discovered concerning algorithmic/ process art relative to formal design. I will use two reference books to substantiate my discussion: Gyorgy Kepes' book Language of Vision[2] and another that he edited entitled: Module, Proportion, Symmetry and Rhythm[3]. In the first, Kepes provides an outline for discussing visual imagery and, in the second, he chooses the writings of several individuals who give their views on broader aesthetic notions.

In Language of Vision Kepes outlines four broad areas into which aspects relating to plastic image design can be grouped: 1) Primary elements. 2) The two dimensional field of reference or the picture frame. 3) Spatial Forces. And, 4) Spatial Organization. In my own mind I find many analogies in algorithmic/ process art to these areas. For example: Primary elements are the graphics primitives. The two-dimensional field of reference is the raster screen. Spatial forces are rudimentary drawing algorithms in subroutine/ function form. And, spatial organization is the underlying task of the main or driver program.

Primary Elements

Most fundamental textbooks on design address the notion of primary visual elements. De Sausmarez[4], for example, using an approach similar to Kepes. He talks about how points or "spots" effect space and how lines

can be formed from points and how areas can be formed from both points, lines and their combinations. By analogy, the elements of computer graphics/ computer art can be thought of in the same manner. Our most fundamental visual element is the pixel. However, the placement of a pixel on a raster display is more restrictive or "discrete."

If the pixel is the most primitive of all computer graphics visual elements then from it can be constructed other primitives. I propose to list and briefly describe what I consider to be a fundamental set of primary elements for the purpose of generating algorithmic/ process art. To a designer/ artist such as myself machine level code is generally opaque; it is only necessary to know a primitives' usage and to properly define associated parameters.

```
/* erase the screen to a specified color */
ERASE (color);
        color = color value

/* return the color value of a specified pixel */
GETPIX ( x_screen, y_screen );
        x_screen = x - axis screen coordinate
        y_screen = y - axis screen coordinate

/* turn on a specified pixel to a specified color */
PUTPIX ( x_screen, y_screen, color);
        x_screen = x - axis screen coordinate
        y_screen = y - axis screen coordinate
        color = color value

/* draw a line between two points with options */
LINE ( x1, y1, x2, y2, color, width);
        x1 = x - coordinate of initial end point
        y1 = y - coordinate of initial end point
        x2 = x - coordinate of final end point
        y2 = y - coordinate of final end point
        color = color value
        width = line thickness

/* draw a rectangle by specifying opposite corners with
options */
RECTANGLE ( x1, y1, x2, y2, color, fill_flag);
        x1 = x - coordinate of initial corner
        y1 = y - coordinate of initial corner
        x2 = x - coordinate of opposite corner
        y2 = y - coordinate of opposite corner
        color = color value
        fill_flag = 0 (outline) or 1 (fill)

/* draw a circle about a specified screen coordinate with
a specified radius with options */
CIRCLE ( xctr, yctr, radius, color, fill_flag);
        xctr = x - coordinate of center
        yctr = y - coordinate of center
        radius = pixel length of radius
        color = color value
        fill_flag = 0 (outline) or 1 (fill)

/* draw an ellipse about a specified screen coordinate
with specified radii with options */
ELLIPSE ( xctr, yctr, a, b, rot_ang, color, fill_flag);
        xctr = x - coordinate of ellipse
        yctr = y - coordinate of ellipse
        a = x - radius
        b = y - radius
        rot_ang = rotation angle in radians
        color = color value
        fill_flag = 0 (outline) or 1 (fill)

/* draw an elliptical arc about a specified screen
coordinate with specified radii and end points with
options */
ELARC ( xctr, yctr, a, b, x1, y1, x2, y2, color);
        xctr = x - coordinate of center of elliptical arc
        yctr = y - coordinate of center of elliptical arc
        a = x - radius
        b = y - radius
        x1 = x - coordinate of starting point
        y1 = y - coordinate of starting point
        x2 = x - coordinate of ending point
        y2 = y - coordinate of ending point
        color = color value

/* copy a portion of the screen to another portion of
screen (same size) */
COPYRECT ( x1, y1, x2, y2, dest_x, dest_y);
        x1 = x - coordinate of initial corner (lower-left)
        y1 = y - coordinate of initial corner (lower-left)
        x2 = x - coordinate of opposite corner
        y2 = y - coordinate of opposite corner
        x_dest = x - coord. of destination (lower-left)
        y_dest = y - coord. of destination (lower-left)

/* draw a predefined polygon with options */
POLY ( poly_dscrpt, color, fill_flag);
        poly_dscrpt = polygon description (array)
        color = color value
        fill_flag = 0 (outline) or 1 (fill)
```

Graphics primitives are extremely machine dependent. This group of graphic primitives comes fairly close to the set I have available to me; I did not and could not write them myself. The set I use were written by a software engineer, David Cook[5], in assembly language and they are available to me through the C programming language. For the record: I use an MS-DOS™ machine (IBM® PC/AT™), a Microsoft® C Compiler and the AT&T® TARGA16™.

The catalog of primitives given above represent the building blocks for graphics programming, specifically algorithmic/ process art programming. I suggest that a tool box consisting of these or a similar set of graphics primitives would allow any programming artist a reasonable tool kit of primary elements to produce an endless range of static and/or dynamic computer artworks.

The Two-dimensional Frame of Reference

Much has been written and theorized about the image plane or two-dimensional frame of reference. Kepes called it the "picture field[6]" and de Sausmarez labeled it the "space frame[7]." These authors of basic design theory and others have made reference to it as a window into a world where the designer/ artist establishes the rules and controls what happens accordingly in that space.

In computer art/ computer graphics, we are forced to look at algorithmic/ process art on the raster or video screen. This is both a blessing and a curse! It is a blessing in the respect that video technology is pervasive and economical. Video Cassette Recorders have almost provided a quality means of recording and disseminating imagery. The curse is that video is cumbersome, not very portable and, it restricts the size and spatial resolution of the image. The larger we enlarge the video image the bigger the pixels become etc.

I feel any designer, computer artist or algorithmic/ process artist must remember three important notions about the medium: 1) In general, visual theory still applies to the raster screen. 2) Video has technical and broadcast aspects and that certain rules and conventions apply. And, 3) Video imagery has a unique potential to affect the viewer.

A person who designs for video should be aware of the role of proportion (or lack thereof) in video imaging. Although the raster screen is of a fixed proportion; it need not always be used in total. I would refer the reader to Arnheim's "Review of Proportion.[8]" In this excellent article we are reminded that "the sense of proportion is inherent in the in the experience of perception.[9]"

A person who designs for video should be aware of the general aspects of video as it is used in "The Media." On one hand, there are the technical aspects such as scanning, interlacing, color balancing and transmission. On the other, there are the design aspects such as safe title area, raster interference, placement of text on graphics, live video windows, and a whole plethora of electronic graphics considerations. Let it be sufficient to say that design for electronic media is different than design for print media.

Lastly, a person who designs for video should be aware of video's potential to affect the view through sublimation, i.e. the affect that television has on human beings. It has been demonstrated that we perceive video imagery in a manner different from other image delivery systems. It has been said that video produces a low-level hypnotic state[10] and, it has been stated that our brains perceive a different kind of meaning due to the nature of the media[11]. What this implies is, that in additional to an awareness of the rules of plastic design and attention to technical aspects, the algorithmic/ process artist must also be aware of the way electronic media itself will potentially affect the viewer.

Our two-dimensional space frame, the raster screen, is more than meets the eye. It provides an unique context in which to create and present visual aesthetic experiences. I would suggest to anyone who designs imagery for this medium to be aware of these notions.

Spatial Forces

Spatial forces are broad generalizations for elemental relationships. Placement of primary elements in a picture frame effects the picture field as well as other primary elements. In plastic design these relationships can be generalized into categories, such as: figure-ground, size, subgrouping, attraction and repulsion, visual balance and weight, linear relationships such as perspective etc. Spatial forces create movement and spatial tension. This is what gives the overall optical quality to an image.

To continue with my attempt to make analogies between design theory and algorithmic/ process art, I suggest that subroutines/ functions can be written to produce unique (well-defined) optical qualities. These subroutines/ functions make use of the primary elements (graphics primitives) in algorithms, in a variety of ways. In my way of thinking, each subroutine/ function may produce a modularized visual effect. Transformation data and other information may be determined at a higher-level and passed to each module through a parameter list. Each module may be used singlely, with itself (in repetitious fashion) or with other routines. Thus, a variety of visual statements can be made manifest using top-down programming.

For my own experimentation I have written a series of "spatial forces modules." To support interchangeability each module has and identical parameter list. For example:

```
module_number (xtr, ytr, x_size, y_size, color);
        xtr = x - coordinate of translation (lower-left)
        ytr = y - coordinate of translation (lower-left)
        x_size = width, in pixels
        y_size = height, in pixels
        color = color or tone numerical value
```

By providing for the interchangeably of modules the overall optical effect of the image can changed quickly and dramatically. Furthermore, modules may be picked randomly once a set of parameters is defined. Below are the verbal descriptions of eight spatial forces modules that I have used. Notice that most modules employ controlled randomness to make trivial decisions.

module0 - Holes. Draw a filled circle not to exceed the limits of a prescribed area in a prescribed color.

module1 - Crossing lines. Draw line segments whose

end points are randomly located on opposite sides of a prescribed area in a prescribed color.

module2 - Concentric hollow rectangles. Draw concentric unfilled rectangles in a prescribed area in a prescribed color at a random but uniform interval.

module3 - Family rectangles. Draw 3 randomly placed and sized rectangles in a prescribed area. Rotate the red, green and blue components of the color value allowing each rectangle to be related by color.

module4 - Parallel lines. Draw parallel lines oriented randomly either vertically or horizontally and at a random interval in a prescribed area in a prescribed color.

module5 - Concentric hollow circles. Same as module2 except use the circle primitive.

module6 - Converging lines. Randomly pick a location on one of the four sides of a prescribed rectangular area and divide the opposite side into some random number of uniform intervals, connect these points with the single point on the opposite side. Draw all lines in a prescribed color.

module7 - Polygon. Randomly pick a location on each of the four sides of a prescribed rectangular area and form a polygon. Fill the polygon in a prescribed color.

The Frame Buffer

One dramatic "spatial force" that I use a great deal on the Macintosh™ is screen inversion. The Apple® Macintosh™ has a one bit frame buffer (each pixel is either off or on, black or white.) Once an area is defined these bits can be inverted by a single command. On the AT&T® TARGA16™ a variety of different effects are possible because16 bits of color resolution associated with each pixel. The frame buffer is a relatively new feature in small computers and is therefore worthy of further discussion.

On the IBM® PC AT™/ AT&T® TARGA16™ that I use I have additional control of what goes on in the frame buffer with Cook's Logical Function Mode Set[12]. This programming option effects the whole drawing environment (the raster display via the frame buffer) and, as such, it should be considered as a spatial force. Specifically, the 'mode' setting allows the user to tag a special logical function to occur in the frame buffer during a write. The syntax is as follows:

flogic (mode); where, when mode equals
0, color gets 'ADDed' to pixel in the frame buffer.
1, color gets 'ANDed' to pixel in the frame buffer.
2, color gets 'ORed' to pixel in the frame buffer.
3, color 'REPLACEs' pixel in the frame buffer.
4, color gets 'SUBTRACTed' ...
5, color gets 'XORed' to pixel in the frame buffer.

Put another way: As you are drawing you can significantly change the visual effect of what you see relative to a particular mode setting. Usually, when you draw into a frame buffer you are actually writing into a special segment of memory that is also scanned by the video processor. Hence, you normally replace (redefine) what is already there. With an *a proiri* call to Cook's flogic function using a legal value other than 3, that particular boolean operation will be performed at the pixel level. Thus, the color value you are drawing (writing) into the frame buffer operates with the color information that is already there. The results can be spectacular! For example, to XOR Erase the screen with a particular color dramatically changes the visual quality of the entire image.

Randomness

A few words about randomness. There are many kinds of randomness (pseudo, Gaussian, Brownian, fractal etc.) To use or rely on randomness too much is not Smart art. I use randomness to make trivial decisions, i.e., to add visual texture. Over the years I have come to the conclusion that the excessive use of randomness in a program will not generate a consistently beautiful or visually satisfying work. Therefore I believe that randomness must be bridled and used in moderation.

Let me cite a ridiculous example that typifies the overuse of randomness: It is conceivable that all known visual imagery can be generated by writing a simple algorithm that will loop around the construct of illuminating random pixels in random color. It could be argued that this is a rational approach but, who is going to sit in front of a CRT display and wait for this to happen? Ideally, we use randomness to provide for the unexpected but it should be "parametized" and used with moderation. I do not rely on serendipity to produce aesthetic effects. (I admit it occurs.) Rather, I try to achieve variety and balance in my algorithmic imagery by using randomness in premeditated manner.

Spatial Organization

In the following discussion of spatial organization I intend to clarify the concept of "plastic image" and "visual whole." Respectively, Kepes states "The plastic image is an organism that reaches out of the dimensions of understanding beyond the sensory radius." and, that we have a "dynamic tendency to organize the optical [spatial] forces into a unified whole;" that this happens within the psychological field [emotionally and intellectually] against a background of a readiness to perceive - a field of attention.[13]"

We tend reduce complex visual stimuli into a limited number of optical elements based on our individual experiences or, we tend "to find the most economical spatial unity ... to form the shortest time interval in the shortest spatial span.[14]" Forms of spatial organization

can be categorized as: nearness, similarity or equality, continuance and closure and that these and other organizing mechanisms produce rhythm and temporal order in imagery.

Unlike the majority of aesthetic media, computer graphics/ computer art inhibits the notion of spontaneous image generation. For example, visual feedback in process/ algorithmic art is not as immediate as it is in painting and other hand oriented production media. In other words, it is very difficult to be subjectively involved with the image when employing the computer. Based on my own experience, I suggest that a specific rationality precedes the algorithmic/ process art image generation process.

Arnheim explains rationality in the context of image making (as opposed to the context of mathematics) as "the extent to which visual structure of a pattern and the parts that make it up are simple clear-cut, identifiable[15]." Rationality can be based on specific notions such as: numerical or geometric relationships. It is my observation that the process of programming computer graphics/ computer art has an inherent rationality built into it and that this rationality cannot be separated from the work itself. As an image making endeavor computer graphics/ computer art presupposes rationality and as such may be called Smart art.

Smart art is the orchestration for computer art/ computer graphics in the time dimension. It facilitates unity, continuity and variety, the key components of time art. In a more specific sense, smart art is the term I give to the embodiment of rationality in thematic algorithms or driver programs; that quality which gives spatial organization to visual output, or to a series of visual events. (Programs that employ artificial intelligence constructs are even smarter!) Smart art is the all encompassing term for the timing, order and control of any number of spatial forces and how spatial forces are manipulated within the space frame.

Examples

By way of example I am appending to this article two smart art program examples. Both illustrate the algorithmic/ process art format. For those that are interested, the "C" programming language listings of the Macintosh™ versions of both of these smart art programs are included following this article. (I translated versions initailly programmed for the IBM PC/AT - AT&T TARGA16 to the more popular machine.)

Both examples use what Arnheim refers to as a "criteria of rationalization.[16]" The first uses a model for "geometric simplicity" employing the lissajous waveform. The second uses a process of "defining parts as fractions of a whole and therefore making it the whole"; a construct for the orderly subdivision of a space frame into a dynamic grid or areas for modular activity.

Smart art example 1 - Lissajous waveform.

The lissajous is a natural "rationalization" for an algorithmic/ process art piece. It is easy to program and has an underlying time component as noted in the following description. "The lissajous is formed by the superimposition of two sinusoidal motion components in the x- and y- directions. The phase shift [rotational offset] is some part of a complete oscillation and the overall complexity depends on the ratio of frequencies.[17]"

The main program of the first example contains the aesthetic variable assignments that give character to the image. It provides the means of assigning bridled random values to the frequency, phase and value (tone) variables. Given these definitions a mathematical algorithm may then proceed to generate a position on the waveform at which a graphic primitive or spatial force module may be located. A key component of the main program is the while-loop construct that operates as an infinite loop, repeating the process, until the user presses the mouse button. At the end of each iteration of the infinite loop, based on a fixed probability, the screen is either erased or inverted. See figures 1 and 2 for representative graphic output.

Smart art example 2 - Parts as fractions of the whole.

A second visual rationalization involves dividing vertical and horizontal picture space into modules of various size. In essence, the screen is subdivided based on a series of whole number relationships. This "rationalization" is not new or unique. Similar algorithms were devised by members of ancient civilizations, such as the Pythagoreans and Vitruvius, and other throughout the history of western art. A relative contemporary, Le Corbusier, wrote a treatise on harmonious measure based on the proportions of the human body entitled: The Modular[18].

In the second software example provided, the main or driver routine incorporates the same infinite loop construct as in the first example. The initial operations performed inside this loop are to assign bridled values to the random variables: number of divisions in x and y, tone, the scanning variables and a variable that will determine which spatial forces module is be used per display. Next, a function called "split" is evoked. It is passed the number of prescribed divisions in x and y and returns appropriately calculated values in arrays. These values contain the translation coordinate data of the lower-left corners of each subdivision, i.e., each modularized area.

A double nested loop construct follows which allows indexing translation coordinate information which in turn is modified by the scanning variable information.

Modules can the draw from left to right or top to bottom or in various combination. The argument of a multiple branch construct keeps the module of choice consistent during the execution of the nested loops. Then, based on a fixed probability, the screen is either erased or inverted. This process repeats indefinitely. See figures 3 and 4 for representative graphic output.

Conclusion

In my attempt to gain a greater understanding about algorithmic/ process art I discovered that what I was doing was unique only in the sense of the medium. One cannot ignore the fundamental aspects of plastic design any more than one can ignore the fundamental aspects of the medium. These factors remain consistent due to the visual nature of the endeavor.

Hopefully the concepts of "smart art" discussed above will give additional credence to algorithmic/ process art. For me it is a legitimate way to use the medium of small computers and small frame buffers. Small computers are thinking machines that require the user to be rational. The designer/ artist is forced to think in a logical and orderly fashion in order to produce an aesthetic image. The "smart artist" will have better control and will get the most out of an electronic media.

It is my opinion that to use electronic media to imitate other media is a step in the wrong direction. Computer graphics/ computer art and consequently algorithmic/ process art have a tremendous potential for producing new and unique image forms. Imitation will bring scorn and rebuke and, if this is the case, then it is rightly deserved. Therefore, a strong case for programming can be made. When a designer/ artist has a programmer's grasp on a thinking medium s/he is in control and not vice versa. Furthermore, with a programmer's control any "criteria for rationality" ought to become a visual reality. This is the correct use of the electronic media and the correct course to follow into the future.

Finally, making algorithmic/ process art has made me aware of the role of computer graphics/ computer art, in electronic media, in general. For instance, I recognize the need to more closely investigate other related electronic media formats, such as video and sound. I have come to this conclusion after careful analysis and critique of my own work. I feel that the imagery I am generating would be enhanced, and the experience made more complete, if accompanied by sound track or musical score. Media synthesis appears to be the ultimate goal.

References

1. Kolomyjec, W.J. "Algorithmic Art and Process Art." Proceedings. 6th Annual Symposium on Small Computers in the Arts. IEEE Computer Society Press. 1986.
2. Kepes, G. Language of Vision. Paul Theobald and Company. Chicago, IL. 1944. (out of print.)
3. Kepes, G. ed. Module, Proportion, Symmetry, Rhythm. George Braziller, Inc. New York, NY. 1966.
4. de Sausmarez, M. Basic design: the dynamics of visual form. Studio Vista: London / Van Nostrand Reinhold Company: New York, NY. 1970.
5. Cook, D. "PCX-STANDARD 'C' INTERFACE, Users and Technical Manual." unpublished. 1986. np.
6. Kepes, G. op. cit. 2. p. 19.
7. de Sausmarez, M. op. cit. 3. p. 32.
8. Arnheim R. "A Review of Proportion." From Kepes, G. ed. Module, Proportion, Symmetry, Rhythm. George Braziller, Inc. New York, NY. 1966. pp. 218-230.
9. Ibid. p. 218.
10. Mander, G. Four Arguments Against Television. Australia.
11. McCullan, M. The Media is the Message. Canada.
12. Cook, D. op cit. 5. np.
13. Kepes, G. op. cit. 2. p. 44.
14. Kepes, G. op. cit. 2. p. 45.
15. Arnheim, R. op. cit. 8. p. 224
16. Arnheim, R. op. cit. 8. p. 224
17. Franke, H.W. Computer Art - Computer Graphics. 2nd edition. Springer-Verlag Inc. Berlin. 1985. pp. 12-13.
18. Le Corbusier. The Modular - A Harmonious Measure to the Human Scale Universally Applicable to Architecture and Mechanics. Cambridge, MA. 1954.

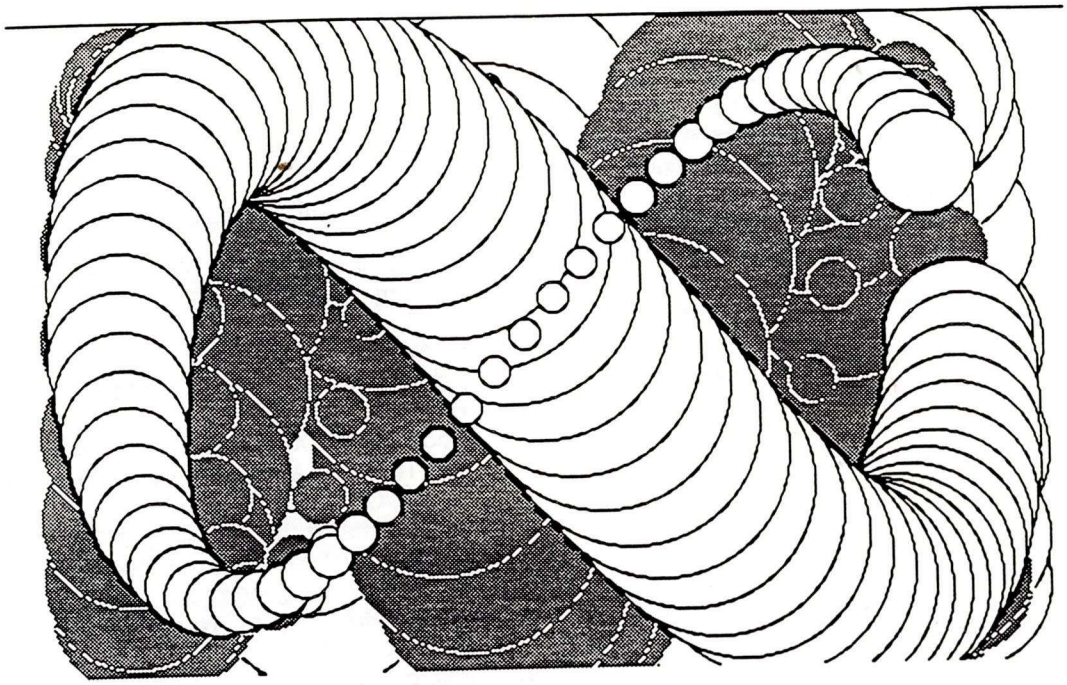


Figure 1. A frame from Smart art example 1 - Lissajous waveform.

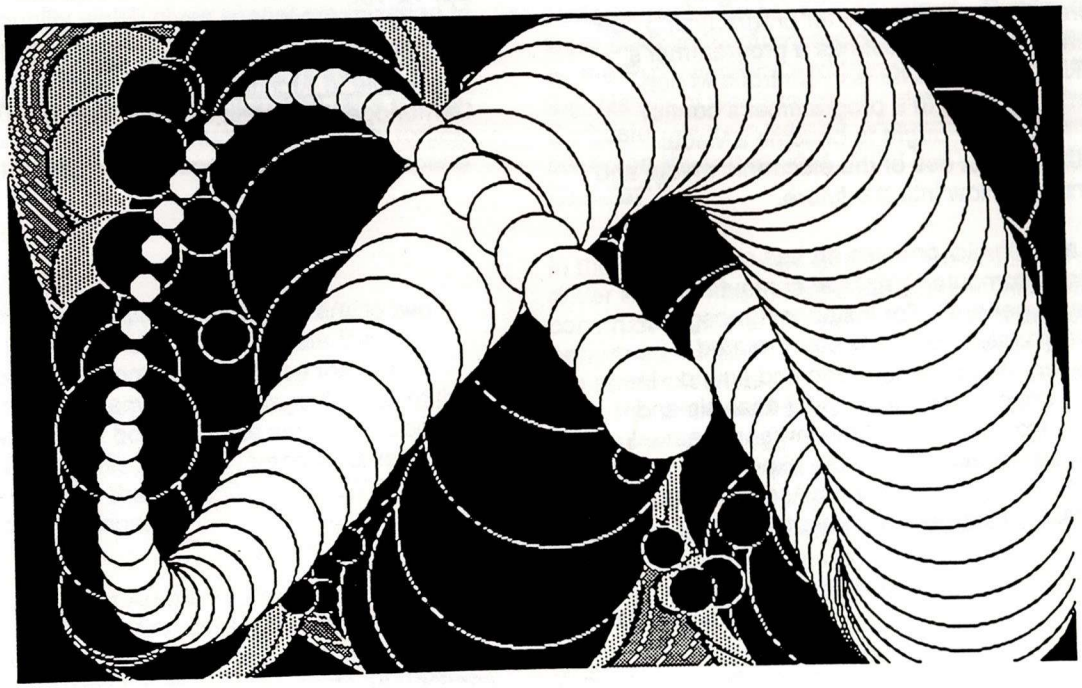


Figure 2. A frame from Smart art example 1 - Lissajous waveform.

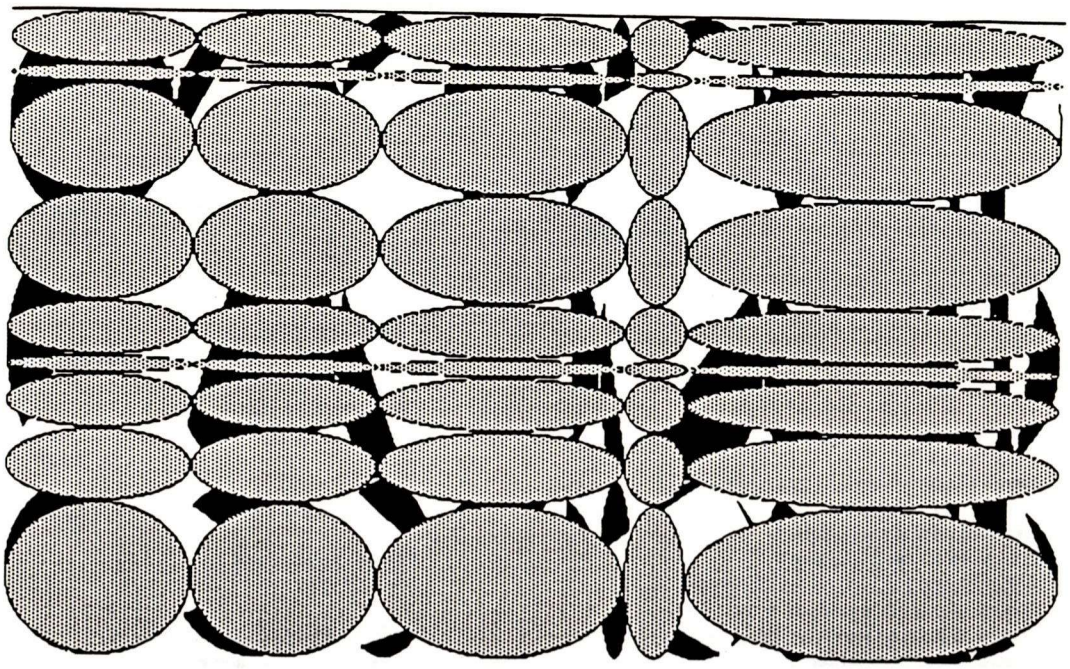


Figure 3. A frame from Smart art example 2 - Parts as fractions of the whole.

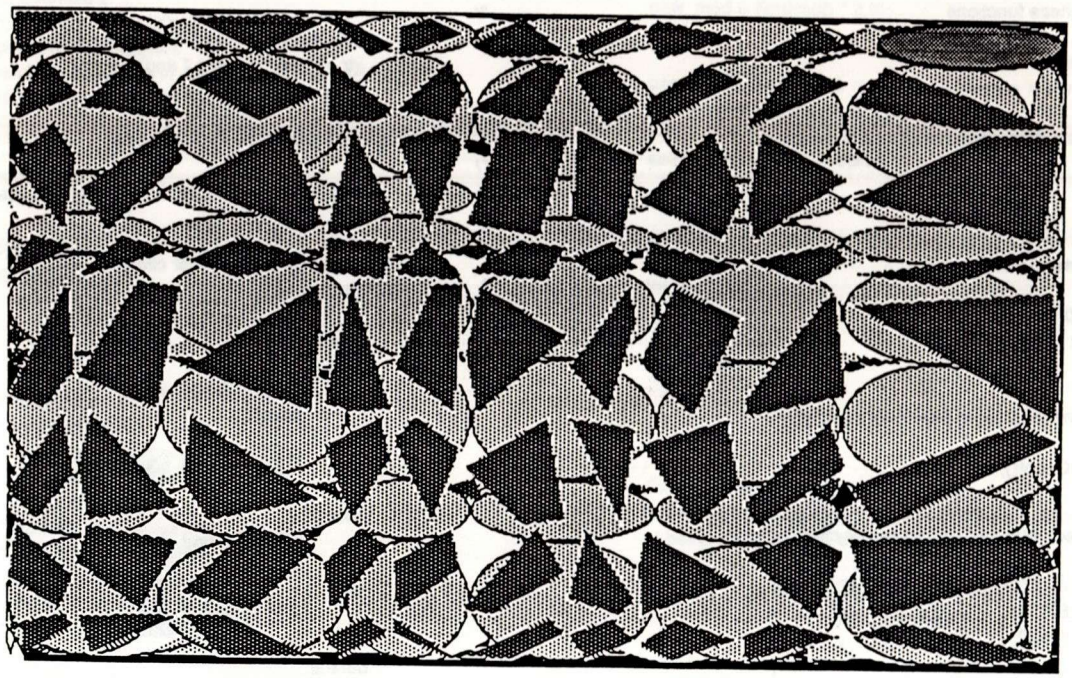


Figure 4. A frame from Smart art example 2 - Parts as fractions of the whole.

```
/*
 *      Smart Art Example 1 - Lissajous waveform
 *      Macintosh Version written in Aztec C version1.06H
 *      by Dr. Bill Kolomyjec for SSCA87
 */

#include <math.h>
#include <quickdraw.h>
#include <event.h>

#define X_SCREEN 512
#define Y_SCREEN 320
#define TWOPI 2.0 * 3.1415926

main() {

int
j,              /* loop index */
num,            /* sampling rate */
l_rad,          /* overall waveform radius */
tone,           /* random tone variable */
r_max,          /* max. radius for circle primitive */
r_min,          /* min. radius for circle primitive */
rad,            /* relative radius */
xtr,            /* x-coord. of point on waveform */
ytr,            /* y-coord. of point on waveform */
x_mid,          /* x-coord. of center of screen */
y_mid;          /* y-coord. of center of screen */

double
ang,            /* relative angle of waveform in radians*/
amp,            /* amplitude or half-height */
aspect,         /* aspect ratio correction factor */
f_max,          /* max. allowable frequency */
f_min,          /* min. allowable frequency */
freq,           /* frequency or ratio of waveform */
pct,            /* relative percent */
phase;          /* phase shift or offset of waveform */

/*
 *      declare these functions
 */
double rnd(), sin();

/*
 *      QuickDraw data types (name two rectangles)
 */
 Rect sr, mod;

/*
 *      define screen and erase it
 */
SetRect (&sr, 0,0,512,342);
EraseRect (&sr);

/*
 *      use internal counter to provide a random seed
 */
randSeed = TickCount();

/*
 *      calc. aspect ratio and screen center coords.
 */
aspect = (double)X_SCREEN / Y_SCREEN;
x_mid = X_SCREEN / 2;
y_mid = Y_SCREEN / 2;

/*
 *      assign values to aesthetic variable names
 */
r_max = 64;
r_min = 8;
f_max = 7.0;
f_min = 1.0;
num = 90;
l_rad = 127;

/*
 *      process loop, hold down mouse button to exit
 */
while (!Button()) {

/*
 *      define random variables
 */
        phase = rnd() * TWOPI;
        freq = rnd() * f_max + f_min;
        tone = (int)(rnd() * 4.0);

/*
 *      one waveform loop
 */
        for (j = 0; j<num; ++j)  {

/*
 *      calc. relative percent thru loop
 *      determine relative angle
 */
                pct = (double)j / (num-1);
                ang = pct * TWOPI;

/*
 *      amplitude must always be positive
 *      scale circle accordingly
 */
                amp = (sin(ang) + 1.0) / 2.0;
                rad = (r_max - r_min) * amp + r_min;

/*
 *      calc. screen coords. of relative point on waveform
 */
                xtr = aspect * l_rad * sin(ang + phase) + x_mid;
                ytr = l_rad * sin(freq * ang) + y_mid;

/*
 *      set up circle primitive's drawing area
 */
                SetRect(&mod, xtr-rad, ytr-rad, xtr+rad, ytr+rad);

/*
 *      use switch construct to render circle
 *      in random tone
 */
                switch (tone) {
                case 0:
                        FillOval(&mod, white);
                        break;
                case 1:
                        FillOval(&mod, ltGray);
                        break;
                case 2:
                        FillOval(&mod, gray);
                        break;
                case 3:
                        FillOval(&mod, dkGray);
                        break;
                default:
                        printf("illegal tone value!\n");
                        exit(0);
                } /* end switch */
```

```
/*
 *      always outline the circle in black
 */
                FrameOval(&mod);

/*
 *      abort the mission if mouse button pressed
 */
                if (Button())
                        exit(0);

        } /* end for */

/*
 *      erase the screen 20 percent of the time
 *      else invert it
 */
        if (rnd() < 0.2)
                EraseRect(&sr);
        else
                InvertRect(&sr);

        } /* end while */

} /* end main */

/*
 *      random number function which
 *      returns a value between 0.0 and 0.999...
 */
double
rnd() {
        return(( Random() & 32767) / 32768.0);
}/* end - rnd */




/*
 *      Smart Art Example 2
 *      Parts as Fractions of the Whole
 *      with 2 spatial forces modules
 *      Macintosh Version written in Aztec C version1.06H
 *      by Dr. Bill Kolomyjec for SSCA87
 */

#include <quickdraw.h>
#include <event.h>

#define X_SCREEN 512
#define Y_SCREEN 320

main() {

int
j,              /* loop index */
k,              /* loop index */
xtr,            /* x screen translation coordinate */
ytr,            /* y screen translation coordinate */
nx,             /* number of modules in x direction */
ny,             /* number of modules in y direction */
x_size,         /* module size in x (pixels) */
y_size,         /* module size in y (pixels) */
pick_mod,       /* holds random pick module value */
tone;           /* holds random tone choice */

static int
x_split[11],    /* array of fractional values in x */
y_split[11];    /* array of fractional values in y */

double
lr,             /* left/right scan choice */
tb;             /* top/bottom scan choice */

/*
 *      declare these functions
 */
void module0(), module1();
int split();
double rnd();

/*
 *      QuickDraw data type
 */
Rect sr;

/*
 *      define screen and erase it
 */
SetRect (&sr, 0,0,512,342);
EraseRect (&sr);

/*
 *      use internal counter to provide a random seed
 */
randSeed = TickCount();

/*
 *      process loop, hold down mouse button to exit
 */
while (!Button()) {

/*
 *      define random variables
 */
        nx = rnd() * 10 + 1;
        ny = rnd() * 10 + 1;
        tone = (int)(rnd() * 5.0);
        lr = rnd() - 0.5;
        tb = rnd() - 0.5;
        pick_mod = (int)(rnd() * 2.0);

/*
 *      pass function split nx & ny and it returns the lower-left module
 *      coordinates in the x_split, y_split arrays
 */
        split(nx, ny, x_split, y_split);

/*
 *      vertical
 */
        for (j = 0; j<ny; ++j)  {

/*
 *      horizontal
 */
                for (k = 0; k<nx; ++k)  {

/*
 *      Render modules on screen with a scan algorithm.
 *      Based on coin toss algorithm (lr & tb have a 50% chance
 *      of being negative, see their assignments)
 */
                        if(tb < 0.0) {
                                ytr = y_split[j];
                                y_size = y_split[j+1] - ytr;
                        } else
                                ytr = y_split[ny - j - 1];
                                y_size = y_split[ny - j] - ytr;
                        } /* end if */
```

```
                if(lr < 0.0) {
                        xtr = x_split[k];
                        x_size = x_split[k+1] - xtr;
                } else {
                        xtr = x_split[nx - k - 1];
                        x_size = x_split[nx - k] - xtr;
                } /* end if */

/*
 *      render module determined by pick_mod variable
 *      (note: consistant through nested loops)
 */

                switch (pick_mod) {
                case  0:
                        module0(xtr, ytr, x_size, y_size, tone);
                        break;
                case  1:
                        module1(xtr, ytr, x_size, y_size, tone);
                        break;
                default:
                        printf("illegal module choice!\n");
                        exit(0);
                } /* end switch */

/*
 *      abort the mission if mouse button pressed
 */
                if (Button())
                        exit(0);

            } /* end k for */

        } /* end j for */

/*
 *      erase the screen 20 percent of the time
 *      else invert it
 */
        if (rnd() < 0.2)
                EraseRect(&sr);
        else
                InvertRect(&sr);

        } /* end while*/

} /* end main */

/*
 *      function split - contains an algorithm that picks a prescribed number of
 *      integer values sums them and sizes them to fit the whole (screen.)
 *      Values are then placed into two arrays which together contain
 *      the lower left screen coordinates of each module.
 */

int
split(nx, ny, x_split, y_split)
int nx, ny, x_split[11], y_split[11];  {

int
j,              /* loop variable */
sum,            /* summing variable */
temp,           /* temporary value holding variable */
max_value;      /* maximum part size value */

/*
 *      declare this function
 */
double rnd();

/*
 *      this value is based on an aesthetic judgement
 */
max_value = 6;

/*
 *      split y - axis
 */
sum = 0;
y_split[0] = 0;

for (j = 1; j<=ny; ++j)  {
        temp = rnd() * max_value + 1;
        y_split[j] = temp;
        sum += temp;
} /* end for */

temp = Y_SCREEN / sum;

for (j = 1; j<=ny; ++j)
        y_split[j] *= temp;

for (j = 1; j<=ny; ++j)
        y_split[j] += y_split[j-1];

y_split[ny] = Y_SCREEN - 1;

/*
 *      split x - axis
 */
sum = 0;
x_split[0] = 0;

for (j = 1; j<=nx; ++j)  {
        temp = rnd() * max_value + 1;
        x_split[j] = temp;
        sum += temp;
} /* end for */

temp = X_SCREEN / sum;

for (j = 1; j<=nx; ++j)
        x_split[j] *= temp;

for (j = 1; j<=nx; ++j)
        x_split[j] += x_split[j-1];

x_split[nx] = X_SCREEN - 1;

} /* end - split */

/*
 *      module0
 *      Use a prescribed rectangular area as a bounding box for an Oval
 */
 void
 module0(xtr, ytr, x_size, y_size, tone)
 int xtr, ytr, x_size, y_size, tone;  {

/*
 *      define QuickDraw data type
 */
Rect mod;

SetRect(&mod, xtr, ytr, xtr + x_size, ytr + y_size);
```

```
/*
 *      render an Oval based on tone designation:
 *      for case 0, frame it only,
 *      for cases 1 - 3, fill it, then frame it
 *      for case 5, fill it only
 */
switch (tone) {
case 0:
        FrameOval(&mod);
        break;
case 1:
        FillOval(&mod,ltGray);
        FrameOval(&mod);
        break;
case 2:
        FillOval(&mod,gray);
        FrameOval(&mod);
        break;
case 3:
        FillOval(&mod,dkGray);
        FrameOval(&mod);
        break;
case 4:
        FillOval(&mod,black);
        break;
default:
        printf("illegal tone - module0\n");
        exit(0);
} /* end switch */

} /* end - module0 */

/*
 *      module1
 *      Use a prescribed rectangular area to form a polygon by randomly
 *      picking a point on each side and connecting them in an orderly manner
 */
void
module1(xtr, ytr, x_size, y_size, tone)
int xtr, ytr, x_size, y_size, tone; {

int
x1,y1,x2,y2;    /* random variables */

/*
 *      define QuickDraw data type
 */
PolyHandle p;

/*
 *      declare this function
 */
double rnd();

/*
 *      random points on each side of module
 */
x1 = xtr + rnd() * x_size;
x2 = xtr + rnd() * x_size;
y1 = ytr + rnd() * y_size;
y2 = ytr + rnd() * y_size;

/*
 *      the QuickDraw way of defining polygons
 */
p = OpenPoly();
        MoveTo(x1, ytr);
        LineTo(xtr + x_size - 1, y1);
        LineTo(x2, ytr + y_size -1);
        LineTo(xtr, y2);
        LineTo(x1, ytr);
ClosePoly ();

/*
 *      render a polygon based on tone designation:
 *      for case 0, frame it only,
 *      for cases 1 - 3, fill it, then frame it
 *      for case 5, fill it only
 */
switch (tone) {
case 0:
        FramePoly(p);
        break;
case 1:
        FillPoly(p, ltGray);
        FramePoly(p);
        break;
case 2:
        FillPoly(p, gray);
        FramePoly(p);
        break;
case 3:
        FillPoly(p, dkGray);
        FramePoly(p);
        break;
case 4:
        FillPoly(p, black);
        break;
default:
        printf("illegal tone - module1\n");
        exit(0);
} /* end switch */

KillPoly(p);

} /* end - module1 */

/*
 *      random number function which
 *      returns a value between 0.0 and 0.999...
 */
double
rnd() {
        return(( Random() & 32767) / 32768.0);
} /* end - rnd */
```

TEACHING THE TURTLE C

William F. Dudley
Electrical Engineer/
Consultant
186 Holbrook Road
Holbrook, (Long Island) N.Y. 11741

Marie Morello Ozmon
Artist/Programmer
P.O Box 181
Hopewell, N.J. 08525

ABSTRACT

Teaching the Turtle C describes the development and use of a set of C functions which emulate a turtle programming environment. The functions can be used to develop an understanding of C programming, and to draw textile simulations, simple geometric shapes, and space filling constructions.

Why C?

A previous article by Ozmon "Digital Textile Construction: Using Logo," described how she used Logo to develop textile simulations. While Logo provides the user with a highly interactive and user friendly environment, Logo interpreters remain system dependent. Within the past year the need for access to more computer memory and code which could be ported to many different machines led to the Ozmon's purchase of XT - clone parts. Along with the purchase and assembly of the new machine came the requirement to learn a different operating system and the challenge of learning a language which would allow access to large areas of machine memory, be system independent, permit access from within commercial graphics software applications, and permit recursion. William Dudley, suggested C.

How To C

A quick look at Kernighan and Ritchie convinced Ozmon that learning C would require more time than she had alloted for a proposed graphics project. The question arose of how a beginning C programmer could use the algorithms she had used in Logo and transfer them into C code. The answer seemed to be in conceptualizing the transfer process as teaching the turtle C code. The task at hand became that of creating a turtle which would understand C and function in the same way as the Logo turtle. In Logo the turtle responds to very simple commands which can be combined to form complex procedures. In C the turtle is non-existant and must be defined. In an on-going dialogue Ozmon shared with Dudley her description of an ideal turtle world environment. Dudley wrote the low level C routines that look like the turtle primitives forward, back, left, and right. The code for these routines is given in Listing 1 Turtle.c. The C turtle control options can be customised to fit the needs of the programmer/user. In Video.c*, Listing 2, Dudley composed the low level routines which control the pc hardware video display i.e. draw lines, set pixels on/off and determine colors. These functions were written in such a way as to allow Ozmon to think in conventional turtle world terms and thus to convert her Logo algorithms into C functions. For example, Listing 3 is a Logo routine for drawing a Hilbert curve. Its C counterpart is hil.c, Listing 4.

By changing the variable inputs in hil.c images may be drawn which range from approximations of the Hilbert curve in Plate 1 to the textile simulation of Plate 2. In addition, with small line length increments, Hil.c serves as a space filling rountine as in the rectangular areas of Plate 3.

Utilizing the format of hil.c any number of interesting functions may be written by changing the command lines from hil (s,l,p) up to and including left((int) (90*p));. Thus it is possible to take the Logo code given in "Digital Textile Construction" and convert it into C functions.

*The V_Draw routine in video.c is taken from the Dr. Dobbs article listed in the Bibliography.

C - ing Beyond The Turtle World

Initially it took several months for Ozmon to began feeling some confidence in writing, reading, and using C functions. Compiling the C code seemed to be the biggest hurdle in the whole process. In the beginning stages of learning C, she would panic when confronted with the many error messages resulting from the misplacement of one brace. She soon learned to utilize the editor's capabilities for matching braces before venturing into the mire of error messages.

In addition to converting her Logo procedures to C functions, Ozmon experimented with combining simple line routines, as in listing 5, with more complex shapes via the batch file process. Thus the operating system was used as an interpreter. (See Plate 3). While composing images in this way was somewhat easier than coding directly in C, Ozmon soon realized that the her use of C code was greatly limited by the batch process. Recently she has been exploring more complex routines which are combined with with paint system generated images in her textile and video work.

A Selected Bibliography

Barkakati, Nabajyoti. "Simple Plots with the Enhanced Graphics Adapter." Dr. Dobb's Journal of Software Tools. #115, May 1986, pp. 74 - 75.

Kernighan and Ritchie. The C Programming Language. Prentice-Hall, 1978.

Ozmon, Marie Morello. "Digital Textile Construction: Using Logo." Proceedings of the 5th Symposium on Small Computers and the Arts. IEEE. October 1985.

Plate 1

Hilbert Curve

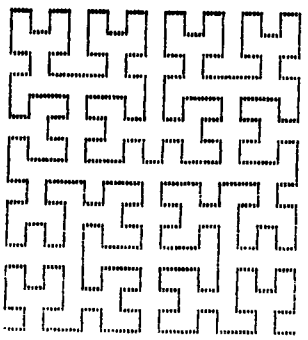


Plate 2

Textile Simulation

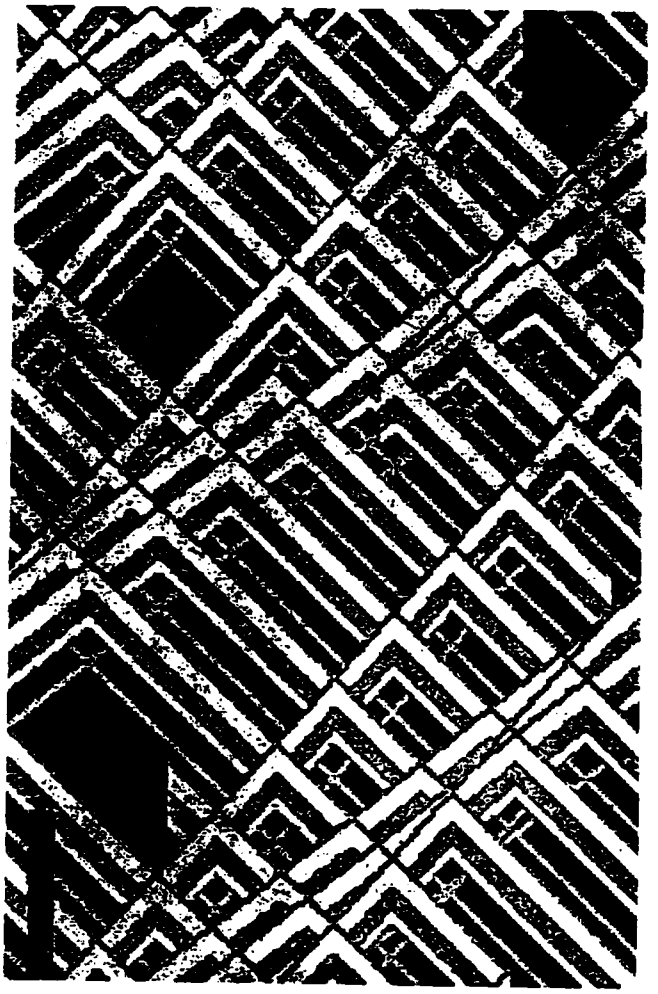
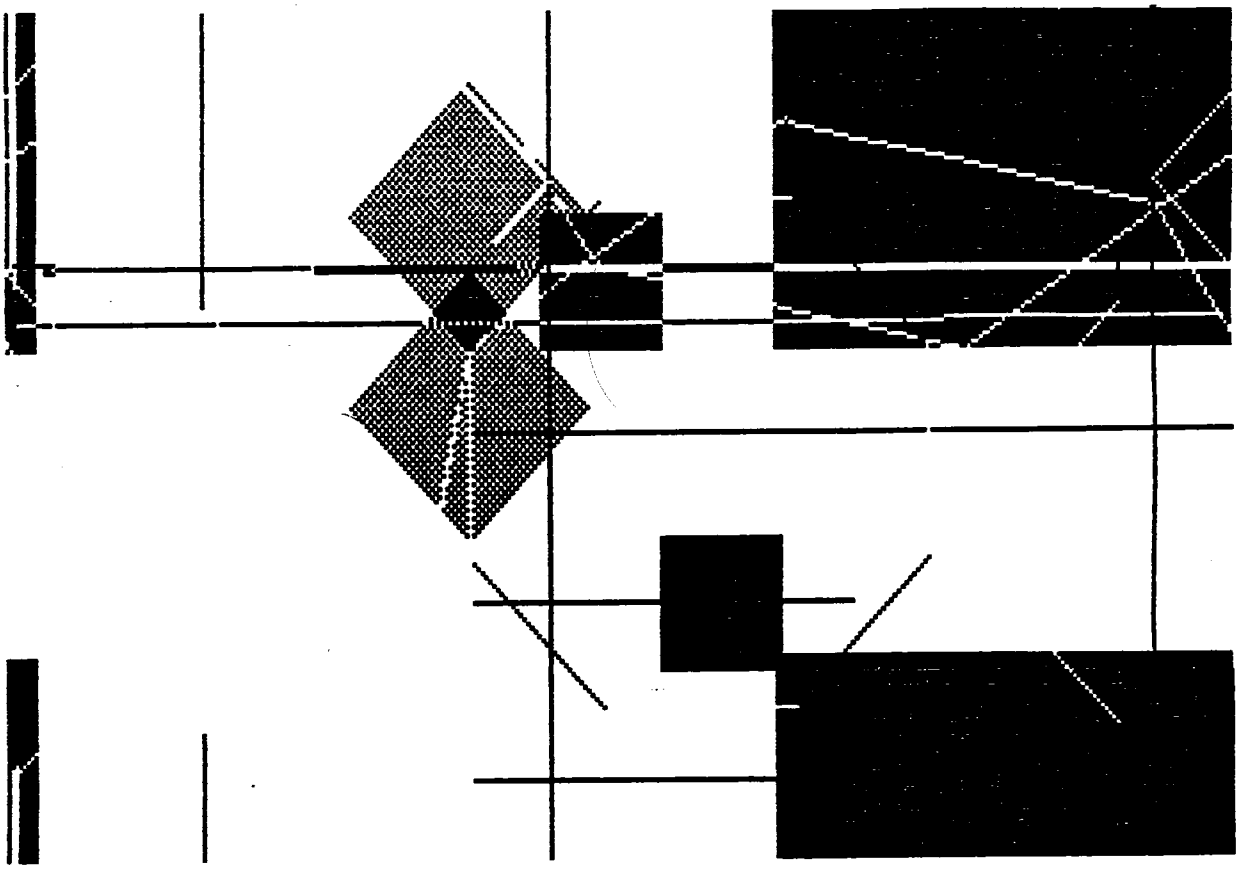


Plate 3

A Frame From "Memories: Things Past"



Listing 1

```
/* turtle graphics routines
 *      forward(n) move forward n units, drawing line
 *      right(a)   turn right 'a' degrees
 *      left(a)    turn left 'a' degrees
 *
 *      Global variables:
 *      int     X               current turtle x coordinate
 *      int     Y               current turtle y coordinate
 *      int     theta           current turtle direction in degrees
 *      int     WRAPFLAG        if non-zero, lines wrap around edge of screen
 */
#include <math.h>
#include <stdio.h>
#include "video.h"
#define MX maxxdot
#define MY maxydot

int X, Y, theta, WRAPFLAG;
extern int MX, MY;

forward(len, color)
int len, color;
{
int nx, ny;
int keybd;
double phi;
if(len == 0) return;
phi = atan(1.0)*((double)theta)/(double)45.0;
nx = X + (int)(((double)len) * cos(phi));
ny = Y + (int)(((double)len) * sin(phi));
keybd = v_draw(X, Y, nx, ny, color);
X = nx; Y = ny;
#ifdef DEBUG
        printf(" %d, %d, %d, %d",X,Y,nx,ny);
#endif
return(keybd);  /* true if keyboard was hit */
}

right(a)
int a;
{
theta -= a;
theta %= 360;
}

left(a)
int a;
{
theta += a;
theta %= 360;
}
```

Listing 2

```
/* video.c
 *
 */
#include <dos.h>
#include <conio.h>
#include "video.h"
#define void int

#define BIOS_VIDEO 16
#define SETMODE 0
#define SETCOLOR 11
#define SET_CURSOR 2
#define WRITE_PIX 12
#define MX maxxdot
#define MY maxydot
```

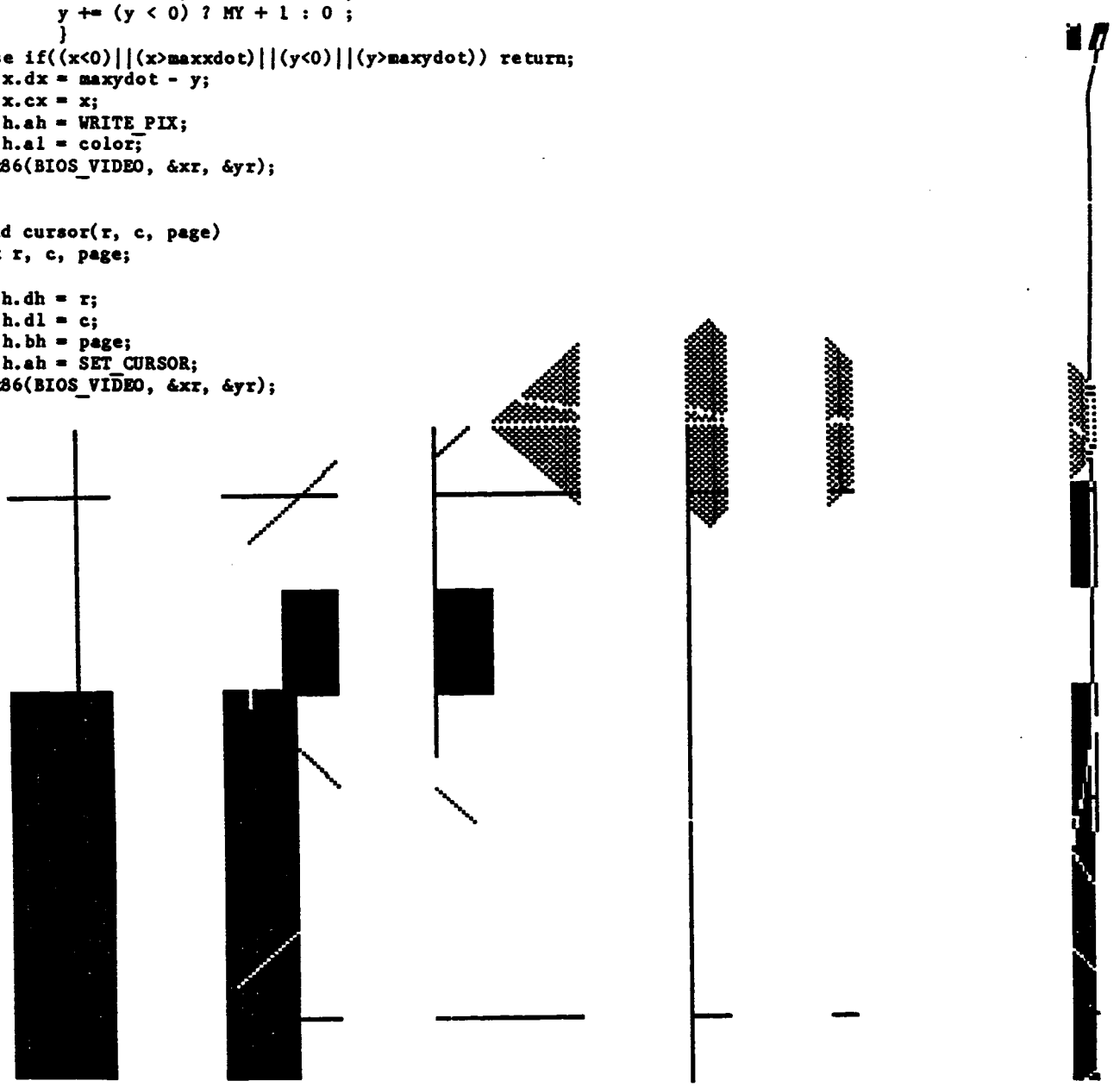
video.h

```
#define ALPHAMODE 2
#define COLORMODE 3
#define LORESMODE 4
#define HIRESMODE 6
#define EGAMODE 16
#define ATTMODE 64
#define MAXROW 24
#define MAXCOL 79

#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define WHITE 7
#define GREY 8
#define LTBLUE 9
#define LTGREEN 10
#define LTCYAN 11
#define LTRED 12
#define LTMAGENTA 13
#define YELLOW 14
#define HIWHITE 15
```

```
static union REGS xr, yr;
int maxxdot=319;
int maxydot=199;
long SLOW=1;
extern int WRAPFLAG;


int v_init(mode, bgcolor, palette)
int mode, bgcolor, palette;
{
xr.h.ah = SETMODE;
xr.h.al = mode;
int86(BIOS_VIDEO, &xr, &yr);
if(mode == ALPHAMODE) return;
if((mode == HIRESMODE)||(mode == ATTMODE)||(mode == EGAMODE))maxxdot = 639;
if(mode == ATTMODE) maxydot = 399;
if(mode == EGAMODE) maxydot = 349;
xr.h.ah = SETCOLOR;
xr.h.bh = 0;
xr.h.bl = bgcolor;
int86(BIOS_VIDEO, &xr, &yr);
xr.h.ah = SETCOLOR;
xr.h.bh = 1;
xr.h.bl = palette;
int86(BIOS_VIDEO, &xr, &yr);
return(maxxdot);
}

int v_draw(x1, y1, x2, y2, color)
int x1, y1, x2, y2, color;
{
int dx, dy, incr1, incr2, incr3, d, x, y, xend, yend;
dx = abs(x2 - x1);
dy = abs(y2 - y1);
if(dy <= dx) {
        if(x1 > x2) {
                x = x2;
                y = y2;
                xend = x1;
                dy = y1 - y2;
                }
        else {
                x = x1;
                y = y1;
                xend = x2;
                dy = y2 - y1;
                }
        d = 2 * dy - dx ;
        incr1 = 2 * dy;
        incr2 = 2 * (dy - dx);
        incr3 = 2 * (dy + dx);
        putdot(x, y, color);
        while(x < xend) {
                x += 1;
                if (d >= 0) {
                        if (dy <= 0) {
                                d += incr1;
#ifdef DEBUG
printf("A");
#endif
                                }
                        else {
                                y += 1;
                                d += incr2;
#ifdef DEBUG
printf("B");
#endif
                                }
                        }
```

```
                else {
                        if (dy >= 0) {
                                d += incr1;
#ifdef DEBUG
printf("C");
#endif
                                }
                        else {
                                y -= 1;
                                d += incr3;
#ifdef DEBUG
printf("D");
#endif
                                }
                        }
                putdot(x, y, color);
                }
        }
else {                  /* abs(slope) > 1 */
        if (y1 > y2) {
                y = y2;
                x = x2;
                yend = y1;
                dx = x1 - x2;
                }
        else {
                y = y1;
                x = x1;
                yend = y2;
                dx = x2 - x1;
                }
        d = 2 * dx - dy;
        incr1 = 2 * dx;
        incr2 = 2 * (dx - dy);
        incr3 = 2 * (dx + dy);
        putdot(x, y, color);
        while(y < yend) {
                y += 1;
                if (d >= 0) {
                        if (dx <= 0) {
                                d += incr1;
#ifdef DEBUG
printf("E");
#endif
                                }
                        else {
                                x += 1;         /* ??? */
                                d += incr2;
#ifdef DEBUG
printf("F");
#endif
                                }
                        }
                else {
                        if (dx >= 0) {
                                d += incr1;
#ifdef DEBUG
printf("G");
#endif
                                }
                        else {
                                x -= 1;
                                d += incr3;
#ifdef DEBUG
printf("H");
#endif
```

```
                        }
                    }
                putdot(x, y, color);
                }
        }
#ifdef DEBUG
printf("%d, %d, %d, %d, %d, %d\n",dx,dy,d,incr1,incr2,incr3);
#endif
return(kbhit());
}

void putdot(x, y, color)
int x, y, color;
{
int i;
for(i=0 ; i<=SLOW ; i++) /* do nothing */ ;
if(WRAPFLAG) {
        x %= MX + 1 ;
        y %= MY + 1 ;
        x += (x < 0) ? MX + 1 : 0 ;
        y += (y < 0) ? MY + 1 : 0 ;
        }
else if((x<0)||(x>maxxdot)||(y<0)||(y>maxydot)) return;
xr.x.dx = maxydot - y;
xr.x.cx = x;
xr.h.ah = WRITE_PIX;
xr.h.al = color;
int86(BIOS_VIDEO, &xr, &yr);
}

void cursor(r, c, page)
int r, c, page;
{
xr.h.dh = r;
xr.h.dl = c;
xr.h.bh = page;
xr.h.ah = SET_CURSOR;
int86(BIOS_VIDEO, &xr, &yr);
}
```



Listing 3

```
TO HIL :SIZE :LEVEL :PARITY

IF :LEVEL = 0 THEN STOP
LT :PARITY * 90
HIL :SIZE :LEVEL - 1 :PARITY * 1
FD :SIZE RT 90 * PARITY
HIL :SIZE :LEVEL - 1 :PARITY
FD :SIZE
HIL :SIZE :LEVEL - 1 :PARITY
RT 90 * :PARITY
FD :SIZE
HIL :SIZE :LEVEL - 1 :PARITY * -1
LT :PARITY * 90
END
```

Listing 4

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include "video.h"
extern int X, Y, theta, WRAPFLAG;
extern long SLOW;

hil(s,l,p)
int l;
float s, p;
{
extern int forg;
if (l <= 0) return;
left((int)(90*p));
hil(s,l-1,-1.0 * p);
if(forward((int)(s),forg)) exit(0);     /* abort if a key is typed */
right((int)(90*p));
hil(s,l-1, p);
forward((int)(s),forg);
hil(s,l-1, p);
right((int)(90 * p));
forward((int)s,forg);
hil(s,l-1,-1.0 * p);
left((int)(90 * p));
}
int forg=BLUE;

main(argc,argv)
char *argv[];
int argc;
{
int mode=LORESMODE;
int waitflag=0;
int initflag=1;
```

```
int palette=0;
int back=RED;
int i, argcnt;
int l=5;
float parity=1.0;
float size=5.0;
argcnt=argc;
WRAPFLAG = 1;
X = 310;
Y = 100;
theta = 0;
for(i=1 ; i<argc ; i++) {
        if(argv[i][0]=='-') {
                argcnt--;
                switch(argv[i][1]) {
                        case 'l' :
                                l = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 's' :
                                size = (float)atof(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'm' :
                                mode = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'w' :
                                waitflag++;
                                break;
                        case 'x' :
                                X = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'y' :
                                Y = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'p' :
                                parity = (float)atof(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 't' :
                                theta = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'b' :      /* backround color */
                                back = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'g' :      /*palette choice*/
                                palette ^= 1;
                                break;
                        case 'f' :      /*forground color*/
                                forg = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'n' :
                                initflag ^= 1; /* toggle initflag */
                                break;
                        case 'd' :
                                SLOW = atoi (&argv[++i][0]);
                                argcnt--;
                                break;
                        default :
                                goto usage;
                        }
                }
        }
```

```
if (argcnt > 1) {
usage:  fprintf(stderr,"turtle: usage: [-x X] [-y Y] [-l len] [-t angle] [-m mode]\n");
        exit(1);
        }
v_init(mode, back, palette, initflag);
#ifdef DEBUG
        printf ("s=%g,l=%d,p+%g",size,l,parity);
#endif
hil(size,l,parity);
if(waitflag) {
        putchar(7);
        getch();
        printf("%c[=2h",0x1b);
        }
exit(0);
}
```

Listing 5

```
/* line.c just draws one line of length s at coordinate x,y
 * and angle t
 * delay d
 */
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include "video.h"
extern int X, Y, theta, WRAPFLAG;
extern long SLOW;
int forg=WHITE;

main(argc,argv)
char *argv[];
int argc;
{
int mode=LORESMODE;
int waitflag=0;
int initflag=1;
int back=BLUE;
int i, argcnt;
int size=5;
argcnt=argc;
WRAPFLAG = 1;
X = 160;
Y = 100;
theta = 0;
for(i=1 ; i<argc ; i++) {
        if(argv[i][0]=='-') {
                argcnt--;
                switch(argv[i][1]) {
                        case 's' :
                                size = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'm' :
                                mode = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'w' :
                                waitflag++;
                                break;
                        case 'x' :
                                X = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'y' :
                                Y = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
```

```
                        case 't' :
                                theta = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'b' :
                                back = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'f' :
                                forg = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        case 'n' :
                                initflag ^= 1;
                                break;
                        case 'd' :
                                SLOW = atoi(&argv[++i][0]);
                                argcnt--;
                                break;
                        default :
                                goto usage;
                        }
                }
        }
if (argcnt > 1) {
usage:  fprintf(stderr,"line: usage: [-x X] [-y Y] [-l len] [-t angle] [-m mode] [-d delay]\n");
        exit(1);
        }
if(initflag) v_init(mode, back);
#ifdef DEBUG
        printf ("s=%d",size);
#endif
forward(size,forg);
if(waitflag) {
        putchar(7);
        getch();
        printf("%c[=2h",0x1b);
        }
exit(0);
}
```

Visual Metaphors Used In Teaching "C" Programming Concepts to Artists

Duane M. Palyka

Computer Graphics Lab
New York Institute of Technology
Old Westbury, New York 11568

September, 1987

ABSTRACT

This paper is a follow-up paper to the paper I presented two years ago entitled "Teaching 'C' Programming to Artists", and is based upon material which I developed for my class *Introduction to "C" Programming with Graphics Applications*, taught to artists at New York Institute of Technology. I developed these "visual metaphors" simply because the students, being visually oriented, needed visual stimulation to help them learn programming concepts. Since this paper uses examples from widely-spaced classes without much technical explanation, it is helpful to know some "C" before reading this paper.

If your brain catches on fire because of insufficient insulation by levels of language exposure, remember that you have been properly warned! This paper is oriented towards teachers who know programming and wish to teach it to artists. Artists wishing to wade through on their own should cover themselves with a light coat of BASIC and learn to skim and hit the good parts without crashing into any hard-core concepts.

As usual, I imbed the metaphors with my own personal iconography. In fact, the last metaphor in this paper also reflects my continued exploration of the relationship between art, the unconscious mind, and personal mythology.

The Class

Briefly, the class consists of about 15 students who group themselves into threes on five or six Mindset computers (IBM-PC clones with better and faster graphics).

The class centers around three basic problems: The student is required to write (1) a program which draws three rectangles in a composition (like Joseph Albers, Ad

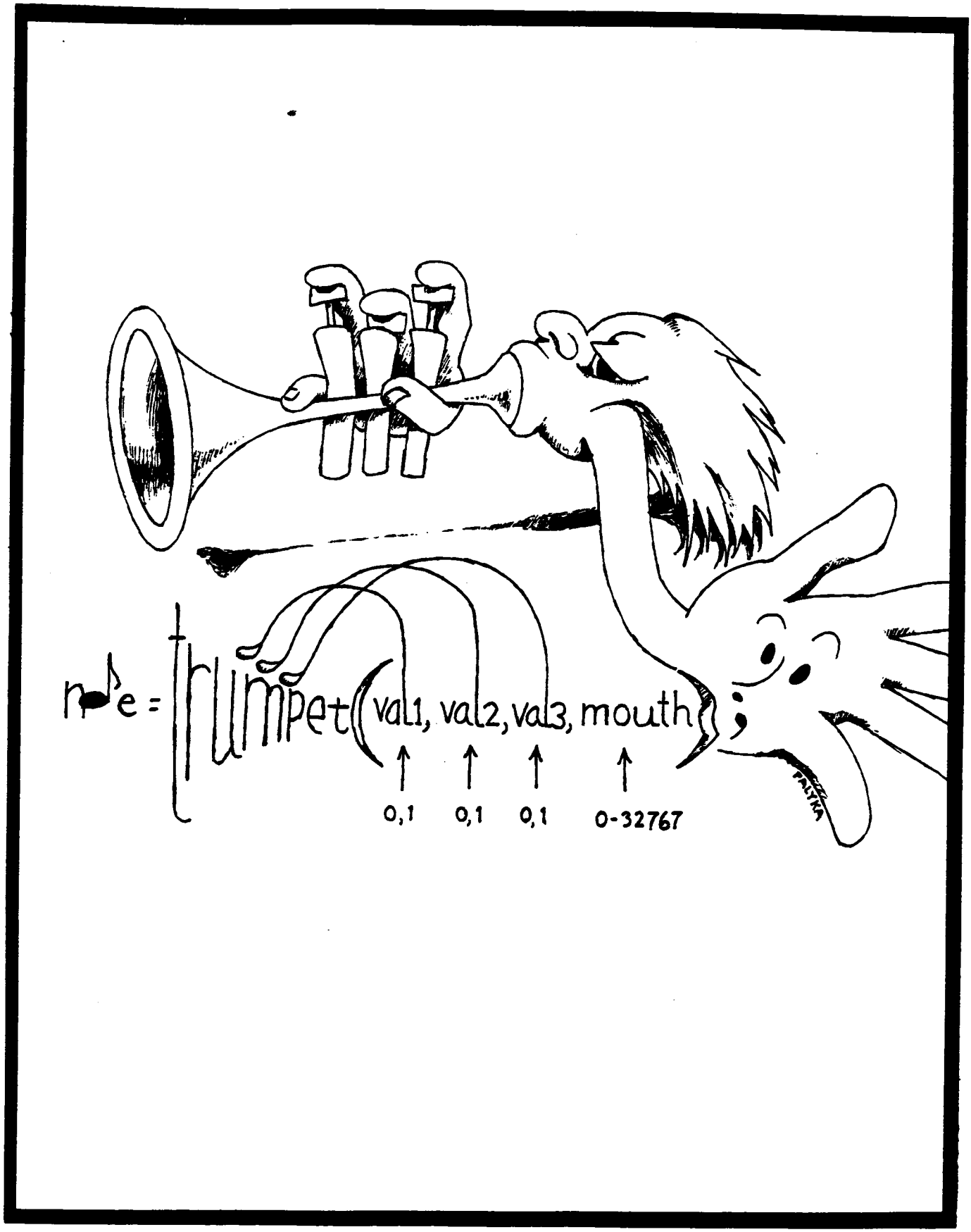


Figure 1.

Reinhardt, or, perhaps, color-field artists), (2) a program which takes two drawings by the artist and, using them as key frames, generates a near real-time sequence of 2D animation, and (3) a program which takes one drawing, extrudes it into 3D space, and performs various 3D transformations upon it in near real-time.

Besides the experience of programming, manipulating images in 2D and 3D, the students are required to make images work esthetically in near real-time.

This paper, however is not concerned with these three problems. They were the subject of the last paper, and may be included in future papers. Instead, this paper concentrates on the in-between material that helps the students to learn the concepts which enable them to write the above programs. Using this material, I attempt to ease the trauma of entering unaccustomed territory.

PAINT PROGRAM PANIC 101: The Motivating Anxiety State

There happens in each computer artist's life a moment where, usually after an exhausting day of heavily poking pixels in a paint program, out of a deep sleep he awakens in a cold sweat with the intense realization that he CANNOT and WILL NOT allow himself to be a slave to computer programmers and system designers any more! He ABSOLUTELY MUST learn the "C" programming language NOW! It is time to gain control of the medium!

After his body stops shaking, the distraught artist usually calms himself down by busying himself with some mundane task like (1) turning the light switch on and off several dozen times, or by (2) herding all the chickens he keeps in his bedroom out into the hall (chickens are interesting creatures to draw and they help with breakfast as well), or maybe by (3) clipping his toenails with an electric hedge trimmer-- a convenient noisy, destructive device handy for herding chickens and for drowning out cackling sounds.

As reported to me, a few artists, on their third or fourth attack, even reach desperately for their convenient bedside "C" Programming "bibles". It appears that they carefully, but futilely, put these books next to their bed in hopes that some information might seep into their brains by osmosis during the sleeping process. Seeing streams of unintelligible gibberish imbedded in the pages, they, being experts on intuition, feel that there must be something important there! They know, deep inside, that they could do wonderous artistic things with it if they could just execute some kind of book-to-brain "block transfers"!

Clutching the books to their chests, they conclude that visual people need VISUAL METAPHORS to learn programming language concepts! (Trumpets blow here!)

So... Within the scope of this paper I would like to present three examples that I use in my discussions of C-Programming constructions. I developed them because I could no longer stand the silence and blank looks in the classroom during my verbal elaborations of these concepts.

For those students who have had some exposure to BASIC, "C" shows that you do not have to live with "goto's" and "gosub's". You can actually create modular black boxes that can be isolated from the rest of your code and kept in working condition, impervious to changes you may make in other parts of your program. And you can even call these "black boxes", or functions, reasonable names that relate to their functionality.

In the first "visual metaphor", I use an analogy of "function" to A MAN BLOWING A TRUMPET! (Insert more trumpet blows!)

BLOWING A TRUMPET-- The First Visual Metaphor

To add variation to the computation within a function, specific values are input through the argument list to figure into the desired calculation, and often a single value is returned as the "return value" of the function, which gets associated with the function name itself. In the statement: "note = trumpet(val1, val2, val3, mouth);" the man in figure 1 blows the trumpet by removing from his memory the values he wishes to use for the arguments "val1", "val2", "val3", and "mouth", and by substituting those values in place of the above arguments, he creates the note that he wants to play. In this case, the value of the note is the return value of the function. It is received from the function after it has completed and is placed into a memory location called "note".


```
EatCorn()
{
        int angle, ear_dist, got1, kernel_dist;

        PositionEar();
        do {
                ear_dist = 0;
                while (SeeKernel()) {
                        EatIt();
                        for ( kernel_dist = 0;
                                kernel_dist < 100 && !SeeKernel();
                                        kernel_dist++ )  MoveRight(1);
                        ear_dist += kernel_dist;
                }
                MoveRight( -ear_dist );
                for ( angle = 0;  angle < 360 && (got1 = SeeKernel());
                        angle++ ) RotateEar();
        } while (got1);
        DisposeOfEar();
        WipeFaceNSmile();
}
```

Figure 2.

So, a debugged and working function is like a trumpet in the sense that you don't have to know how the trumpet makes its notes, you just have to know how to use it.

An important point that beginning students tend to miss is that you don't have to redefine the trumpet for every note. There is no need for "trumpet1(), trumpet2(), or trumpet3()". The same function "trumpet()" is used for every note, and the returned value is different depending upon which values are used for the input arguments. In our case, either 0 or 1 can be substituted for any of the three valves, giving you 8 possible combined valve values, and some number between 0 and 32767 can be used to select some combination of volume and embouchure.

Although this example is a simplified model of a trumpet, it is useful as a good metaphor to explain some basic "C" programming concepts.

EATING CORN--
The Second Visual Metaphor

As a step up from the simple structure of "Blowing a Trumpet", the second metaphor "EatCorn" illustrates the use of functions, "while's", "do-while's", "for" statements, and nested loops in a simplified "real-life" model. Here we require the student to carefully trace through the code for proper comprehension.

Whereas in BASIC, we have a "for-next" concept, in "C" we have much more flexible alternatives in "while", "do-while", and "for" statements. Although in class I describe their functionality in detail, since the scope of this paper is to talk about visual metaphors, I'll just summarize these basic concepts in pseudo-jibberish. I'll also leave it to the reader of this paper to wade through on his own other nuances of syntax-- like discussion of local and global variables, "define" constructions, etc.

The construction "while(<test>) <statement>;" continuously performs <test> and executes <statement> as long as test is "true" (i.e., computes to some non-zero number). Conversely, the computer declines to execute <statement> and leaves the loop when the test is "false" (i.e., equals zero). So if the <test> is never "true", the statement is never executed and we never loop at all; and if the <test> is always "true", the statement is always executed and we never leave the loop.

The construction "do <statement>; while(<test>);" loops by executing the <statement> and then does the <test> to see if it should execute it again. Here the <statement> is always executed at least once.

The construction "for (<initialize>; <test>; <increment>) <statement>;" works like: "<init>; while(<test>) { <statement>; <increment>; }" where "{" and "}" put groups of statements in place of one, and ";" placed at the end of statements delimits them. However, several statements <statement1>, <statement2>, <statement3> separated by commas (as above) can replace <init> or <increment>. Also, as we'll see later, it is syntactically correct to omit <init>, <test>, <increment>, or <statement> from the construction if they are superfluous to the logic involved.

Returning to our second example, several "black-box" functions used in the "EatCorn" metaphor are:

(1) PositionEar()-- which puts the ear of corn in the correct position for eating.

(2) SeeKernel()-- reports back "true" (=1) or "false" (=0) if the ear is positioned properly so that a kernel can be easily seen.

(3) EatIt()-- a sense-organ gratifying black box which actually allows the user to eat one kernel as found.

(4) MoveRight(num)-- moves the ear right "num" corn units, where several corn units will get you to the next kernel (if a negative number is substituted for "num", the ear will move left instead).

(5) RotateEar()-- rotates the ear of corn one degree about the cob axis.

(6) DisposeOfEar()-- allows the user to throw away the presumably finished ear of corn.

(7) WipeFaceNSmile()-- allows the corn eater to clean his face afterwards.

Even though I cagedly avoid defining the insides of these black-boxes, I do take pains to describe the workings of the "EatCorn()" function.

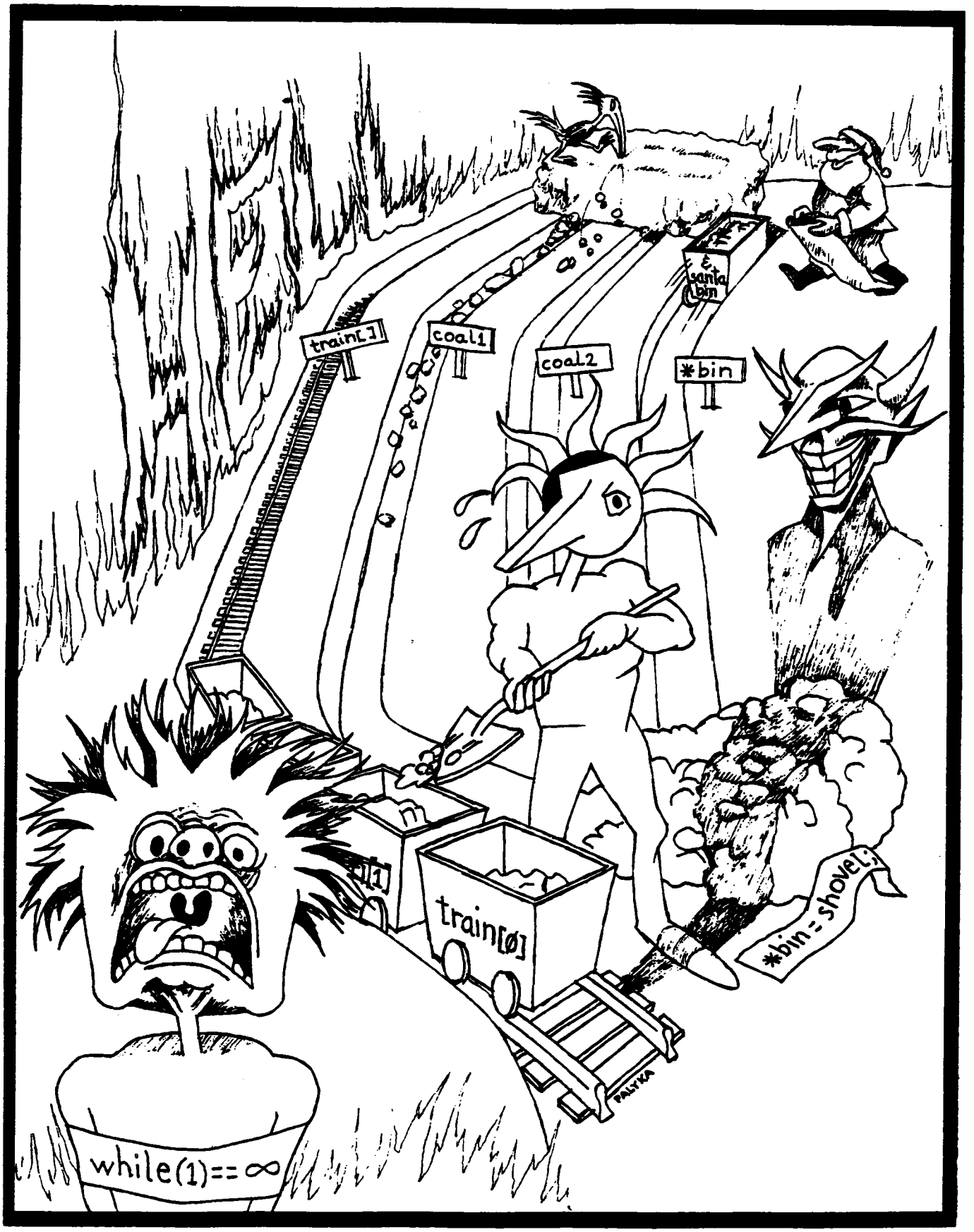

train[1]
coal1
coal2
*bin
santa bin
train[1]
train[ø]
*bin=shovel;
while(1)==∞
PALYKA

Figure 3.

So within the function "EatCorn()", the first thing the creature in figure 2 does is position the ear of corn for eating by calling "PositionEar()". After that he loops in a "do-while" loop until all the kernels are consumed-- indicated by a false "got1" returning from a fruitless search for a kernel, which is left-justified on the ear. Starting at the far left (like on a typewriter), the creature keeps track of how far he goes to the right with the variable "ear_dist". Notice that "ear_dist" is always set to zero at the start of the line. The consumption of a line of kernels continues in a "while" loop as long as the creature can "see a kernel" as he moves to the right.

When "SeeKernel()" returns "true", the corn eater gobbles it up in "EatIt()" and proceeds to hungrily search for the next one by using a "for" loop which moves right one corn unit. The variable "kernel_dist" keeps track of how many corn units it takes before he sees a new kernel. Within this "for" loop, (1) "kernel_dist" is initialized to zero, then (2) a test is made to see if the value of "kernel_dist" is still less than 100 and if any kernels have been seen yet. This arbitrary number "100" was chosen as some reasonable test to make sure we don't run off the end of the ear. If we haven't run off and if we haven't found a kernel, then we (3) move right by one corn unit, (4) increment "kernel_dist", and do our test again (which includes a kernel scan).

Finding a kernal exits the "for" loop and causes us to add the distance to the kernel to the distance measured from the left end of the ear. If we "max out" on kernel_dist, then we will also die when we loop around again on the "while" and try to see the kernal again. This means that we have to move left the number of corn units contained in "ear_dist" (MoveRight(-ear_dist)) and rotate the ear of corn to try to find another row to munch on. (For typewriter enthusiasts, this is equivalent to a "carriage-return" followed by a "line-feed".)

Within the next "for" loop, we see the corn eater rotating the ear one degree at a time until he either sees a kernal, or finds himself back at the spot he was when he began the rotates. If he sees a kernel, then "got1" is "true" and he loops in the outer "do-while" to eat it, otherwise he leaves the loop, throws away the ear, cleans his face, and smiles a smile of gluttonous satisfaction.

That's a lot of work just to eat an ear of corn, but remember that computers are basically stupid and must be told every detail of complex actions. It is roughly equivalent to analyzing a "piece of business" that an actor may perform to keep an audience entertained over dull periods in a script-- like this past section in ours.

HELL--
The Third (and Last) Visual Metaphor

After "EatCorn", we have the ultimate in complex metaphors "Hell", which approaches such mythological epic proportions that even Dante would roll over in his grave if he knew about it. Here, not only should the student carefully trace through the code, but he should relate the code to the pictures (figures 3 and 4) as well. "Hell" consists of functions, "while" loops, "for" loops-- and also introduces "if-else" statements, "switch-case" statements, and (uggh) indirect addressing(!)-- composed in the form of a complex machine with a bizarre mythological storyline. In fact, "indirect addressing" may be the most hellish aspect of the entire scenario. "C" is sufficiently "low-level" that it is almost impossible not to talk about this concept at early stages of the course, so pains were taken to give examples of indirect addressing within function arguments on the function side of the interface; and, similarly on the calling side, discussing the passing of arguments by address instead of by value. Actually, "C" is designed such that it helps to think like the compiler to understand the language. So, if you envision yourself as a compiler in hell, you are taking the correct approach to understanding this program.

The story consists of a mechanically compulsive coal-picking bird (at the top of figure 3) who walks back and forth across a large lump of coal using his beak as a pick-ax to dislodge chunks of the coal as he goes. Each picked coal chunk will then either fall into one of the two chutes labeled (in figure 3) as containing either "coal1" or "coal2". Where each chunk will fall is determined by which chute the bird is closer to at this point in his coal chunk walk. Complementing the task of the coal-picking bird, it behoves the poor trapped souls in hell, placed at the bottom of the chutes by the devils, to "earn their

keep" by evenly distributing the chunks between the cars of the train and Santa's special bin.

You may ask at this point (assuming you've gotten this far): "What does Santa Claus have to do with hell?" Well, as all little boys and girls know, if you are naughty, Santa will fill your stockings with soot and ashes instead of with goodies. In figure 4, we see the devil's kids, who are especially naughty, eagerly awaiting coal chunks from Santa.

Upon examination of the code, in the main function "main()" we see that since the "while(1)" can never be exitted, hell is an endless loop which goes on forever. This is consistant with the Christian concept. Proceeding further, within the endless "while" we have a "for" loop for the bird picking his way across the coal. Using the variable "bird_step", it takes him 100 steps to get across the chunk.

Notice that the "for" loop doesn't initialize anything-- in particular, it doesn't initialize the main stepping variable "bird_step". However, this is fine since "bird_step", a static variable, is initialized to zero at load time, and is returned to zero later by the "BirdReturn()" function.

For each step the bird takes, we call the function "BirdPick()" and pass it (1) the current step number and (2) the locations of "coal1" and "coal2", which gives the bird someplace to put the coal that he picks. Inside of "BirdPick", we see that the arguments which contain the addresses of "coal1" and "coal2" are actually called "chute1" and "chute2" instead. I tried to make the names describe (as well as I could) the indirect process used by them.

In standard bird-picking tests exhaustively performed in super-heated environments (in a Cray II with the cooling system turned off), it was determined that our particular coal-picking bird cannot pick more than 1000 coal-units for each step he takes, so we defined "PICK_MAX" as 1000. In fact, after much picking, the remaining coal block may be too small to even do that! He can then only pick up to the size of the remaining coal block. Thus, we have designated a variable called "pick_max" which normally contains "PICK_MAX" as its value unless the coal_block is too small, then it is set to contain the current size of the reduced coal_block.

Now, the actual amount that the bird picks is somewhere between zero coal units and "pick_max" coal units. So the random number generating function "random()" (with two arguments designating the range) should return a value within this range and put it into "picked_coal". Since the "picked_coal" was removed from the block, we have to subtract it from the "coal_block" with the statement "coal_block -= picked_coal;" For those of you below "C" level, that's equivalent to "coal_block = coal_block - picked_coal;"

The next two statements in the code simply perform a linear interpolation on the picked coal to determine how much to put into "chute1" and how much goes into "chute2". The "*" before each of these indicates that "chute1" and "chute2" really contain the addresses of where to put the coal, so again we have to go indirect to the addresses.

Leaving "BirdPick()" and contemplating entry into "Hell()", we see that since the addresses we gave in the "BirdPick()" chutes are for "coal1" and "coal2", upon leaving it we now have the correct coal units in each of these two variables. Hence we are ready to dump them into "Hell()" to be shoveled into the "train[]" and also into the "santa_bin". Since the "train[]" is an array of cars returning from "Hell()", we must give "Hell()" the address of that array as well as the number of cars in the train. And since the coal is just dumped into hell without regard as to where it came from, only the number of coal units needs to be deposited through the "coal1" and "coal2" arguments. The "bin", however, actually returns from hell, so we must give "Hell()" the address of "santa_bin" so that the trapped shovelers know where to put the coal.

Upon entering the function "Hell()" we notice that to shovel coal efficiently into the cars, "shovel" can be allowed to handle an entire bin-full of coal at once. After all, weight becomes an arbitrary factor in a situation where physical reality doesn't exit as we know it. (Parenthetically, this has always been a problem with computer graphics as well.)

To complete the "Hell()" function, the first "for" loop shovels the coal into the train, and the last statement after the loop gives Santa his share.

The next function, "SantaDump()", gives Santa the means to dump his bin into

```
/*
HELL -- a computer C-itcom by Duane M. Palyka.
*/

#define     COAL_BLOCK_SIZE   1000000L
#define PICK_MAX     1000L
#define NUMSTEPS     100
#define NUMCARS           20

int bird_step, bucket, coal1, coal2;
int sack, santa_bin, stocking[3], train[NUMCARS];
long coal_block = COAL_BLOCK_SIZE;

char TestSize();


main()
{
      while (1) {
            for ( ; bird_step < NUMSTEPS; bird_step++ ) {
                  BirdPick( bird_step, &coal1, &coal2 );
                  Hell( train, NUMCARS, coal1, coal2, &santa_bin );
                  SantaDump( &santa_bin );
            }
            BirdReturn( &bird_step );
      }
}


BirdPick( which_step, chute1, chute2 )
int *chute1, *chute2;
{
      int picked_coal, pick_max;

      pick_max = (coal_block < PICK_MAX? coal_block : PICK_MAX);
      picked_coal = random( -1, pick_max );
      coal_block -= picked_coal;
      *chute1 = (NUMSTEPS - which_step) * picked_coal / NUMSTEPS;
      *chute2 = which_step * picked_coal / NUMSTEPS;
}


Hell( train, n, coal1, coal2, bin )
int train[], *bin;
{
      int i, shovel;

      shovel = (coal1 + coal2) / (n + 1);
      for ( i = 0; i < n; i++ )
            train[i] = shovel;
      *bin = shovel;
}
```

stockings of the little devils. Again we must concern ourselves with passing the address of "santa_bin" because "santa_bin" gets cleared within this function after it is dumped. In "SantaDump()", we see that the coal is dumped into a sack, the bin is cleared, and the sack is tested for its size with "TestSize()".

Within "TestSize()", we have a series of "if-else's" to test for sack size. If the sack contains from 0 to 5 coal units, it is sized as "Puny", 6 to 10 is designated "Small", 11 to 25 is "Medium", 26 to 50 is "Large", and anything over 50 is called "X-tra large". Like our previous "trumpet()" function, "size" is returned as a "return value" for the function.

Now that Santa knows the size of his sack, he can use a "switch-case" statement to parse the proper amount of coal into the three stockings. If he has a large sack, he starts at "case 'L'" and puts half of the sack into the first stocking, half of the remainder into the second, and the rest into the third. In the code, the "break" statement gives him a chance to stop and eat cookies and milk.

If he has a medium-sized sack, he skips the first stocking, and starting at 'M', fills the second and third stockings with all the coal. If Santa happens to have a small load, he dumps all his coal into the third stocking. the entire sackful of coal. And, finally, if he has a size other than small, medium, or large, instead of trying to sort it all out, he just dumps it into the bucket.

To complete the outer loop of "main()", "BirdReturn()" allows the bird to return to his starting position on the block after he does his 100 coal-picking steps. Notice that the address of "bird_step" is passed as an argument to "BirdReturn()" and it is referred to as "shoe" within the function itself. Of course, everyone knows that you look into a "shoe" to find a "step"... and so we wait for "step" to step down to zero.

Questions

Now that we have traced through the entire program step by step, I would like to pose three questions to you that can only be determined by tracing through and seeing what the program does, not by only looking at the options available: (1) When Santa sizes up his sacks of coal, which sack-fuls miss the stockings entirely and end up in the bucket? (2) Are the trapped souls in hell ever saved? If so, how? and (3) Which little devil gets the most coal?

Answers

(1) Logically, by looking at "TestSize()", you might say that both "X-tra large" and "Puny" will miss the stockings. However, one has to trace through the logic of the program to see if both can occur. The largest number that the random number generator in "BirdPick()" can give is "PICK_MAX" (or 1000). "PICK_MAX" coal units distributed into 20 cars and one Santa bin gives us less than 50 units for Santa's sack. Since "X-tra large" is defined to be anything over 50 coal units, "X-tra large" is eliminated as an answer. Now we test for "Puny": Starting again at the random number generator, we see that it is possible to have 21 units of coal picked, If these 21 units are divided evenly between all the bins, Santa gets 1 coal unit. "TestSize()" certainly tells us that 1 coal unit is "Puny". Since there are no other options to test for, the answer is that only "Puny" coal units will end up in the bucket.

(2) Again looking at the random number generator, we see that the "-1" argument, which we cleverly stuck in as the lowest possible value, allows the possibility of negative coal to occur. Following that through as we did with question #1, we see that Santa will end up with the negative coal, causing "TestSize()" to freak when it tries to size it up. Whenever the coal size is negative, "TestSize()" figures that the universe must have reversed itself causing a "cold day in hell", so it allows all trapped souls to escape from the system by executing an "exit()", which takes us immediately to the upper command level of the system. Contrary to what we said earlier and what the fellow in the lower left of figure 3 seems to believe, our hell is not really a Christian hell (an infinite loop), but is more like a Buddhist hell-- existing only until the effect of the sooty deeds that brought the soul here have worn off (or, in our case, are picked away). The soul currently shoveling (in the picture) realizes this fact and has confided in me that he is grateful for the help of Santa, who has the ability and the desire to save him from his condition.

(3) This problem is here as an incentive for the reader to implement the program and find out for himself.

D P$X

2) yes - picked out

```
BirdReturn( shoe )
int *shoe;
{
        while( *shoe > 0 ) *shoe -= 1;
}


SantaDump( bin )
int *bin;
{
        char size;

        sack = *bin;  *bin = 0;
        size = TestSize( sack );
        switch( size ) {
                case 'L': stocking[0] = .5 * sack;  sack -= stocking[0];
                case 'M': stocking[1] = .5 * sack;  sack -= stocking[1];
                case 'S': stocking[2] = sack;  break;
                default:  bucket = sack;
        }
}


char TestSize( coal )
{
        char size;

        if ( coal >= 0 ) {
                if ( coal <= 5 ) size = 'P';            /* puny. */
                else if ( coal <= 10 ) size = 'S';
                else if ( coal <= 25 ) size = 'M';
                else if ( coal <= 50 ) size = 'L';
                else size = 'X';
                return( size );
        } else {
                printf( "What!!!!  Negative coal???0 );
                printf( "It must be a cold day in hell!!!0 );
                printf( "All trapped souls escape from system!!!0 );
                exit();
        }
}
```

Conclusion

Whatever one thinks of the above examples, utilizing the literary qualities of the language to stimulate the visual, story-telling imagination should be used more in presenting abstract computer programming concepts to visually-oriented students. Artists enjoy this more than having the teacher step through an algorithm that coldly calculates a Fibonacci sequence. However, if a Fibonacci sequence is shown to be related to the golden mean and translated to visual terms, we may have something else indeed. The idea should be to present the subject matter to the students in the way that they understand the most.

Another factor involved (particularly in the third metaphor) is the attempt to integrate various aspects of human thought processes through the use of personal mythology. Here, in a small way, I attempt to integrate logical, rational thought processes, literary storytelling, and visual, architypal images which try to touch upon deeper parts of the unconscious mind-- all with a taste of humor to make this concoction palatable.

In contrast to developing a personal mythology, the usual method of trying to adopt a general mythology on an individual basis (which conventional religions promote) often inhibits individual unification and growth rather than helping it. For the mythology to be effective it us reach us on an unconscious level and help us to develop further as individuals-- not mold us into preconceived images of what we should be. It should encourage us to explore our individual psyches further. Ideas of this nature flow through much of my work and is particularly evident in my animation "Living Above the Mouse's Ear".

Although some of my iconography (like Santa and the devils) is borrowed from general mythology, I modify and distort them for my own subjective mythology. Specifically, the devil in figure 3 is represented by a shadow to reflect Carl Jung's concept of "shadow"-- the unconscious part of one's personality which we deny in ourselves but readily see as flaws in others. This trait alone leads to prejudices, alienation, and hellish states of mind. Although not reflected in the drawings, bringing the "shadow" to consciousness and accepting ourselves for what we are, we can even find that the dangerous "shadow" can take a helpful form leading us to further integration.

As a final perspective, this last metaphor of hell reflects my youth growing up in steel town Pittsburgh, where I spent one hot summer inside and under furnaces shoveling hot cinders. At one point in my youth I thought that that's all that life offered. In Pittsburgh, the blast furnaces lit up the sky to obscure any stars that might shine through.

ABOUT THE AUTHOR

Duane M. Palyka is a senior research scientist at the New York Institute of Technology Computer Graphics Laboratory. He received both a BS in mathematics and a BFA in painting from Carnegie-Mellon University. During his employment as a research associate and systems programmer in computer graphics at the University of Utah, he received an MFA in painting from their art department. His computer art has been exhibited internationally since 1968. For the past four years, Mr. Palyka has taught the following courses in NYIT's Art Department: *Media and Art Applications of Computer Graphics,* and *Introduction to "C" Programming with Graphics Applications,* and has assisted Peter Voci and Tim Carlin with: *Microcomputer Graphics,* and *Images I and Images II* (using NYIT's paint system).

REFERENCES

1. Campbell, Joseph, *Creative Mythology-- the Masks of God.* New York: Penguin Books, 1968.

2. Jung, Carl G., *Man and his Symbols.* New York: Doubleday & Co., 1964.

3. Zimmer, Heinrich, *The King and the Corpse-- Tales of the Soul's Conquest of Evil.* Princeton: Princeton University Press, 1948.

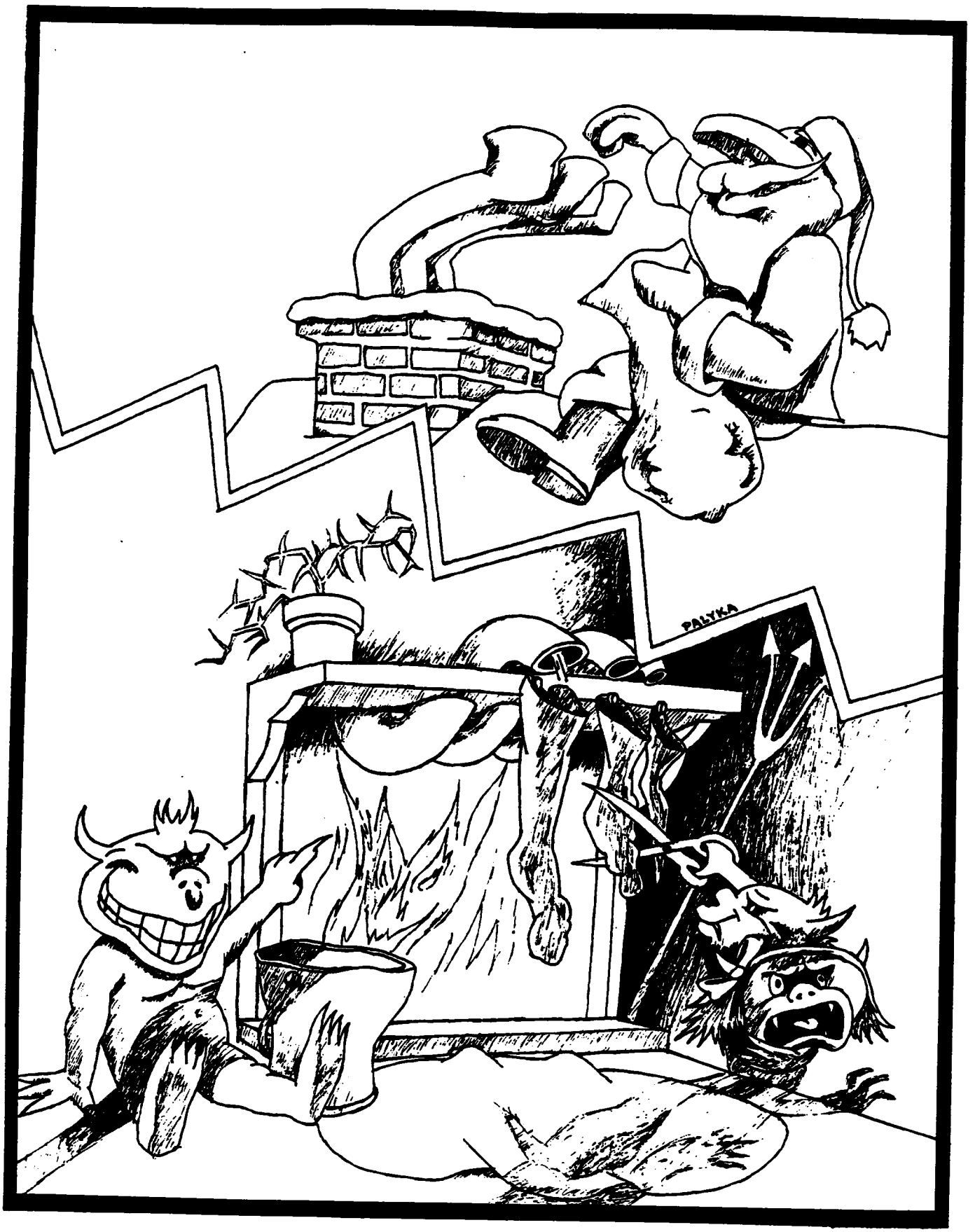


Figure 4.

INSECT MYTHOLOGY INSECT TECHNOLOGY

Paul Rutkovsky
Associate Professor, Dept. of Art, Florida State Univ.
Tallahassee, FL 32306
Research Fellow, Center for Advanced Visual Studies, MIT
Cambridge, MA 02139

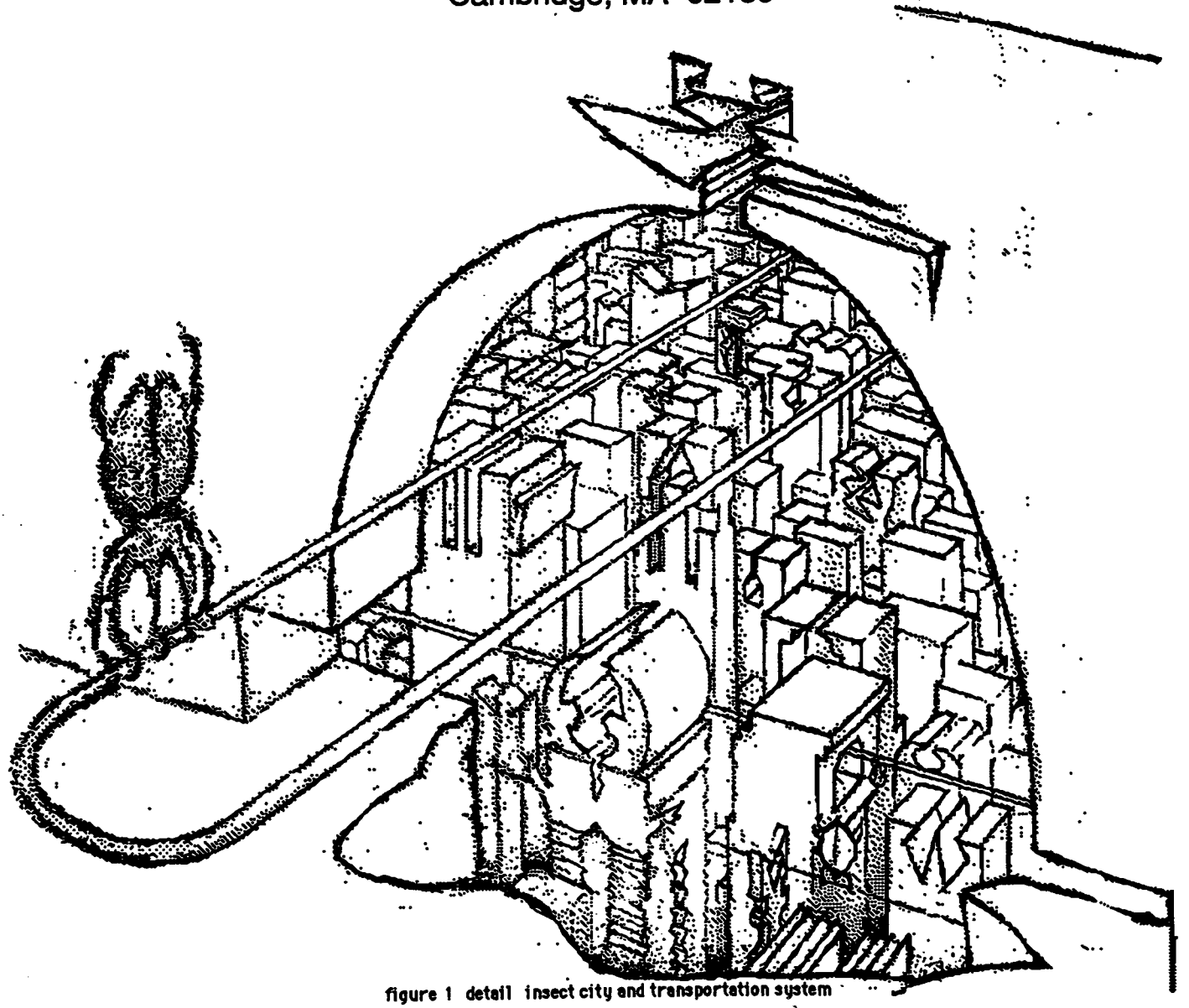


figure 1 detail insect city and transportation system

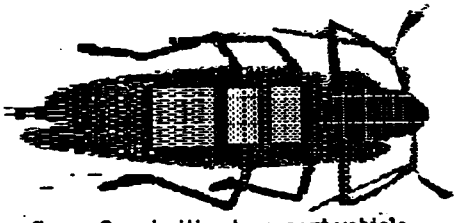


figure 2 primitive transport vehicle

Crandon Park, Key Biscayne, Florida

***INSECT MYTHOLOGY INSECT TECHNOLOGY** is a fabricated civilization created as a specific site proposal for the Metro-Dade Art in Public Places Trust in Southern Florida. In developing a pseudo history and visual representations of the insect's architecture, personal computers were utilized to generate graphic images. Four locations were chosen as "simulated" excavation sites where insects and their structures would be revealed.*

Each site represents a different geological time in the insect's civilization, beginning two million years ago and leading to its decline a million and a half years later.

The following is a brief narrative of INSECT MYTHOLOGY INSECT TECHNOLOGY.

An ancient civilization of insects indigenous to Florida thrived on Key Biscayne over two million years ago. They inhabited almost every square foot of the Key but dominated the area that later became Crandon Zoo. On the old zoo site, the insects built an advanced civilization that has mystified and baffled Entomologists, Anthropologists, Archaeologists, and Scientists ever since evidence was first discovered in 1979, shortly before the zoo closed. Three major excavation sites, plus one yet to be completely uncovered directly beneath the main gate, are located in the old zoo, where specialists have unearthed bizarre evidence of a once highly sophisticated civilization of insects that apparently constructed their own living and working environments, not unlike our 20th century cities. The insects, which also included mites, spiders, and centipedes, developed an amazing ability to harness their own physical energy (possibly two million years ago) and a capacity to utilize the Key's natural resources. These insects not only built their own cities but also constructed an empire around a high-technology culture. It is believed that less than one million years ago they began to harness a form of energy that was similar to electricity. This allowed them to build extraordinarily large and complicated structures (by insect standards).

At one point in their evolution it is believed that the insects metamorphosed into a resemblance of their own technological creations.

Four insect excavation sites are located in Crandon Park (old zoo). The excavation sites are simulated and appear to be recently uncovered; digging activity is still apparent on at least one site. Equipment and other materials are on the active site.

Approximately ten to fifteen feet below the surface two giant Goliath Beetles were discovered in the old zoo site. Archaeologists speculate that they were placed there at least a million years ago as a ritual burial of some unknown origin. There is one male and one female beetle methodically placed side by side with the male facing the main entrance gate (east) and the female facing the main service gate (west). It is believed that the males's large thorax collapsed about a thousand years ago, creating large depressions in the earth that later filled with water to form the lakes around the Central Garden area. As a result of this collapse, giant antennae and legs of both beetles have been pushed up through the ground surface in various locations in the park to be revealed as large sculptural structures.

figure 3 giant male Goliath Beetle

As an integral part of educating and developing an accurate awareness of the park to the public (specifically the Insect Civilizations), an ongoing series of press releases describing the gradual discovery of various sites in the park will be instituted. These press releases will be pseudo-press releases in that they will be announcing the discovery of fabricated sites. In order for this to properly work the discovery of the insect sites should be treated as an unexpected "real" find. The press releases will be written by the Metro-Dade Art in Public Places and the artist to maximize interest in the development of the park and the insect excavation sites.

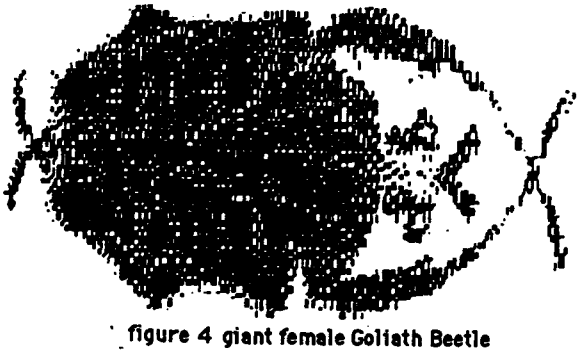
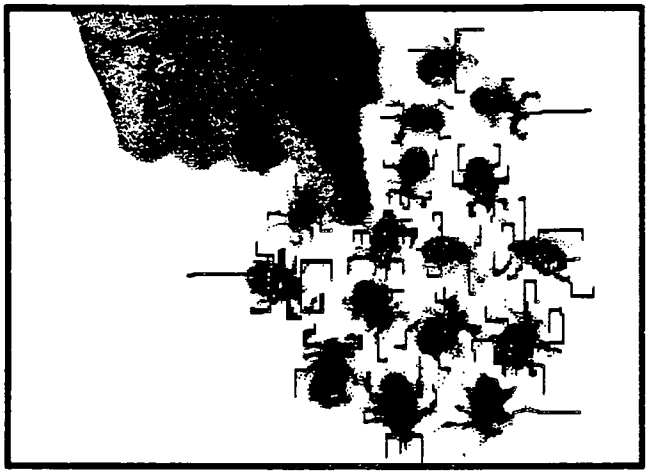


figure 4 giant female Goliath Beetle

SITE #1 The Main Entrance (two million years old)

This site has not been explored to any great extent because it was discovered while the entrance was being constructed. However, some relics and evidence have been saved. This site seems to have functioned as a meeting area and included several structures that might have been of some religious significance.



SITE #1 An Entomologist pointing to highly developed fossilized insects found imbeded in sand near the main gate.

figure 5

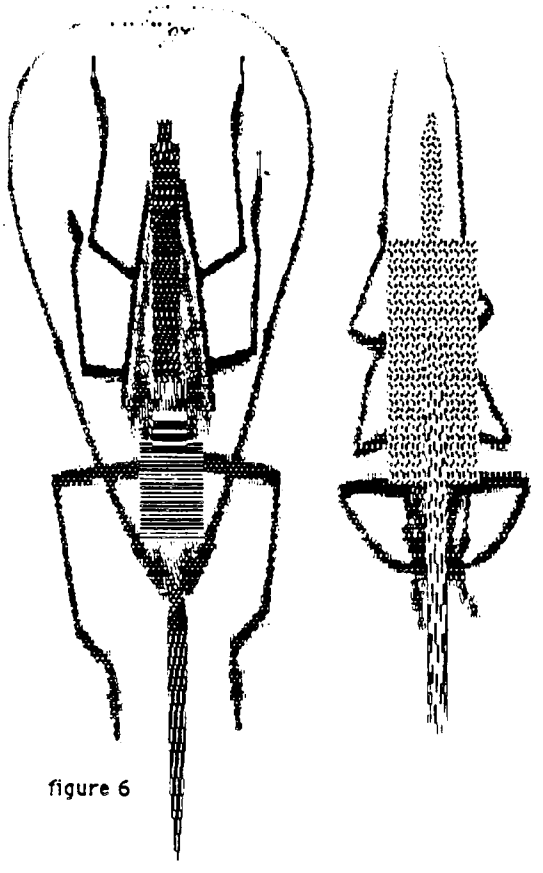
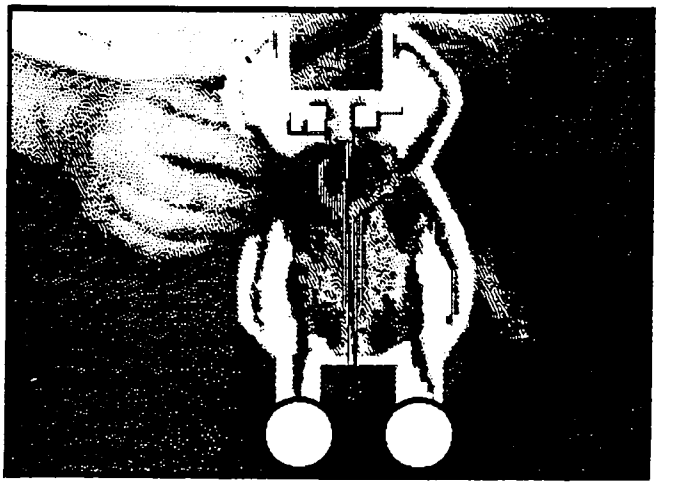


figure 6

VOTIVE FIGURES FROM SITE #1

SITE #2 The Turtle Pen (two million years old)

It is believed that this was the site of a once major metropolis for the insects, where they had the ability to harness energy in some form and to miraculously exploit the area's natural resources to build unusually large structures. These architectural giant structures were apparently living and working quarters for the insects and are more primitive when compared to the streamlined structures of site three. Some evidence of a transportation system as well as indoor plumbing has been found.



SITE #2 This is the best preserved example of an early transportation system, believed to be approximately two million years old

figure 7

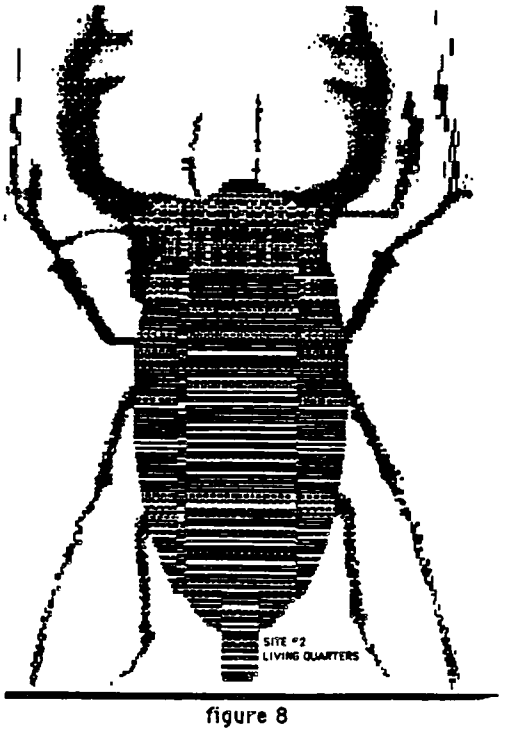
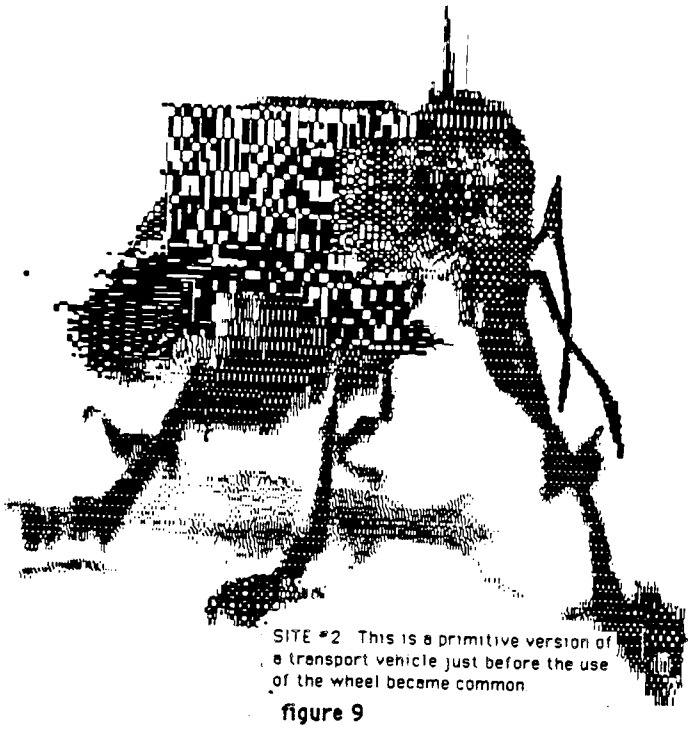


figure 8

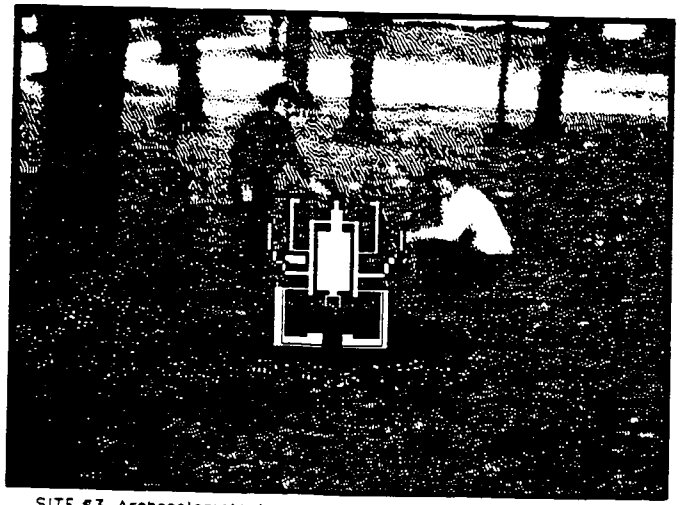


SITE #2 This is a primitive version of a transport vehicle just before the use of the wheel became common

figure 9

SITE #3 The Small Alligator Pit (one million years old)

A highly evolved version of the Goliath Beetle is believed to be responsible for the phenomenal success of this site. The area excavated represents a thriving community of insects that had the ability to capture and use electric power to their advantage. They managed to build vehicles that allowed them to traverse to the southern tip of Key Biscayne in less than a day. Their architectural creations were such that they were almost indistinguishable from themselves. They began to look and function like their creations. The metamorphosis was so complete that the city apparently became the living insects and the insects functioned as a self regulating, governing civilization.



SITE #3 Archaeologists discover a highly evolved Goliath Beetle not far from site number three. This specimen is probably a million years old and was capable of generating electricity

figure 10

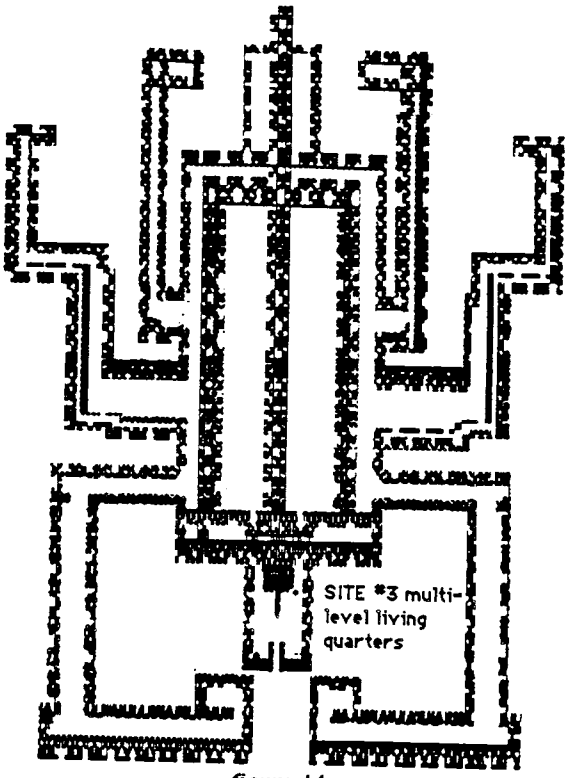


figure 11

SITE #4 Audad Hill Feature (750,000 years old)

Evidence at this site indicates a declining insect civilization. The success of site three did not apparently continue here, and the artifacts and major structures seem to suggest a catastrophic disaster took place soon after a flood. Flooding was a common occurrence on the Key but something more mysterious than the Atlantic Ocean destroyed this civilization. The remaining structures are devoid of any sophisticated understanding of elemental engineering. The decline was most likely gradual but complete--the insects lost their ability to maintain a highly structured and advanced society.

SITE #4 Archaeologists find the remains of a building or insect not far from site #4. Evidence shows that this site was in a decline approximately 750,000 years ago

figure 12

Electronic Imagining: Myth or Fiction

Mark W. Scott
Spiznet Enterprises
PO. Box 30125
Philadelphia, Pa. 19103

I intend to describe my personal experiences with small computers as a medium for artistic expression.

Spending much time around home computers back in the 8-bit processor days of 4-16 colors and 64K RAM, I abandoned any hope of realizing personal graphic visions or pursuing serious efforts to exploit these machines' gross level of representation. During this time (employed as a programmer) I stopped producing graphic art, a process that had been a continuous part of my life from age three onward, although never formally guided.

That was destined to change. During the past two years, entirely without the aid of electronic media, I have produced a large body of work ranging from comic books to wall-sized murals to biblical calligraphy, working with the traditional tools of hand, pen, and brush.

At the beginning of this year I re-examined the possibilities that exist for a reasonable quality, low-end computer graphics product and was very impressed. Though still limited in the size, color, and texture of output, the actual creative environments of the Mac Plus and Amiga 1000 are friendly and large enough to stimulate even PC-shy artists.

The industry, too, seems on far more solid ground than it was three years ago, I will only mention the level of corporate PC infiltration, huge price/performance advances, and increased reliability of products as justification of this perception.

I resolved, though, that my computer graphic efforts would not simulate what I could obtain with other media, and that they would instead be well-considered extensions of my organic work made desirable for several reasons:

First for the purpose of greater editability and duplicity (sic) of images.

Second, to keep track of my symbolic universe's characters, powers, histories, etc. Once I did this I felt so relieved that I did a coldstart of my cerebellum.

Third, I knew that one day (or more likely late one night) I would be able to provide my characters with a program that would give them the ability to move around, lo, even to be artificially intelligent!

Plate 1: Spiznet philosopher with students.

Plate 2: Untitled, brush and ink.

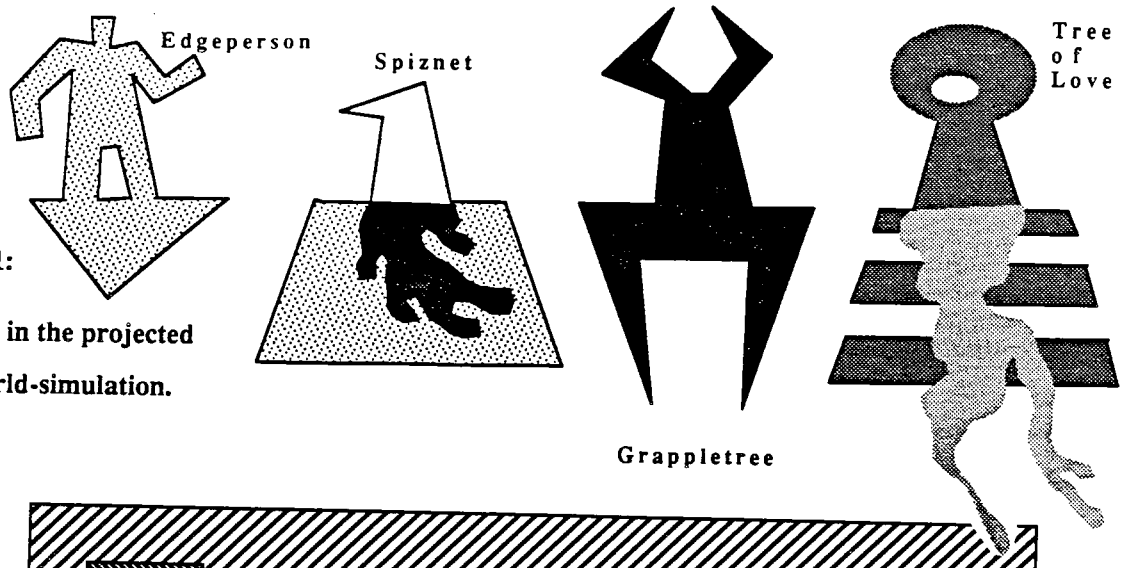


Diagram 1:

Four actors in the projected Spiznet world-simulation.

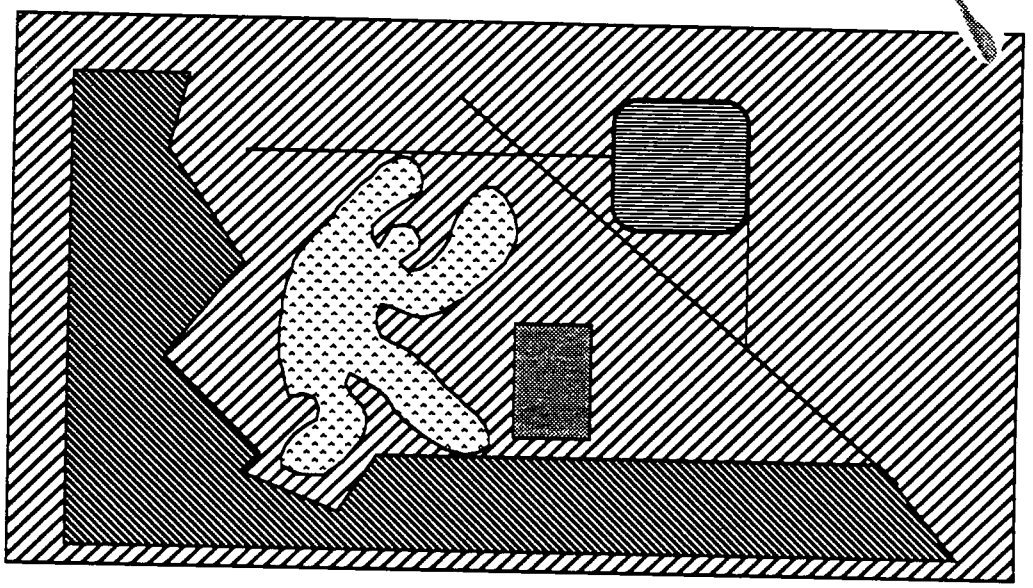


Diagram 2: Edge person watching television.

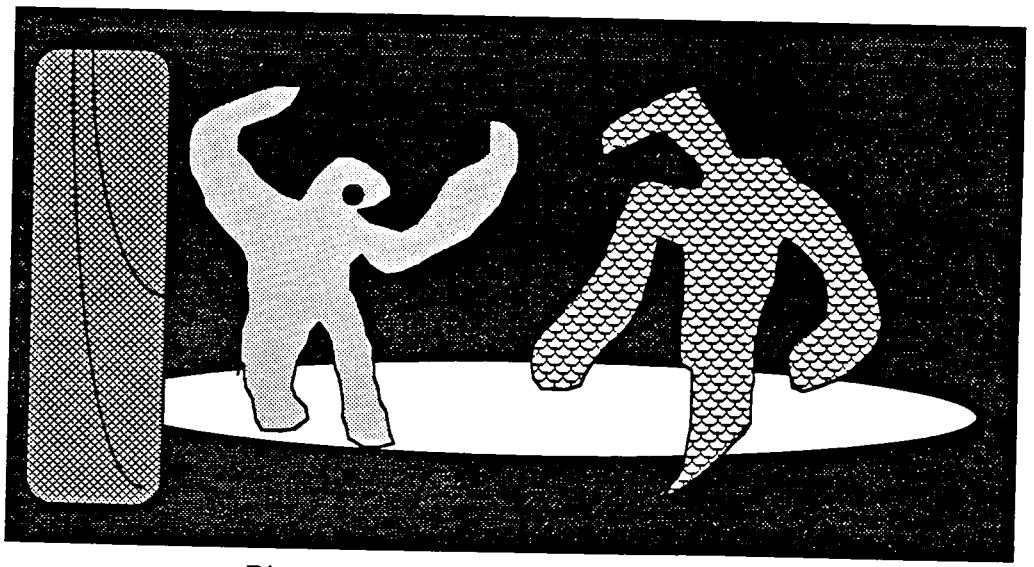


Diagram 3: Edge person vs. adversary.

Plate 3: Untitled with text from medieval Book of the Pious.

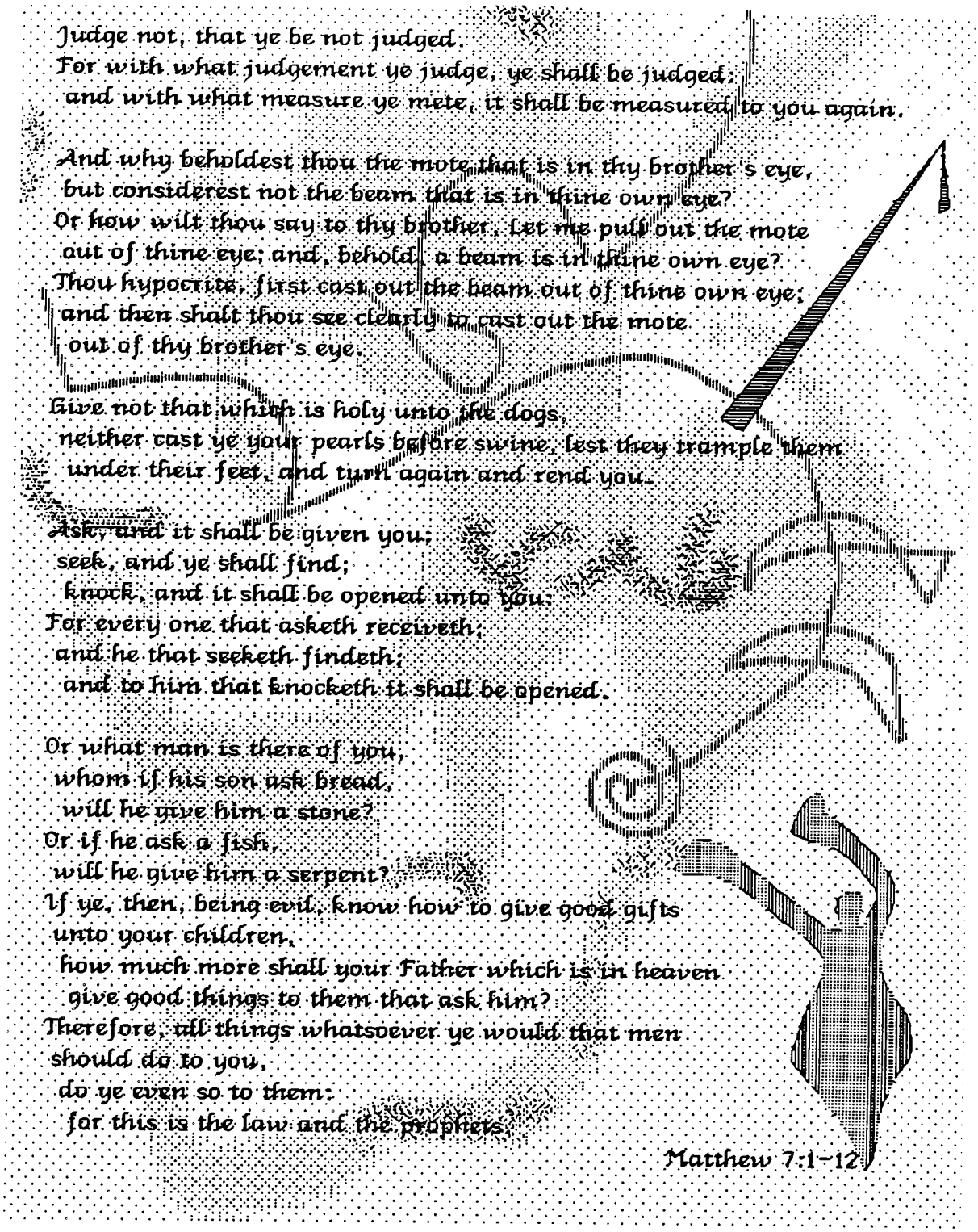
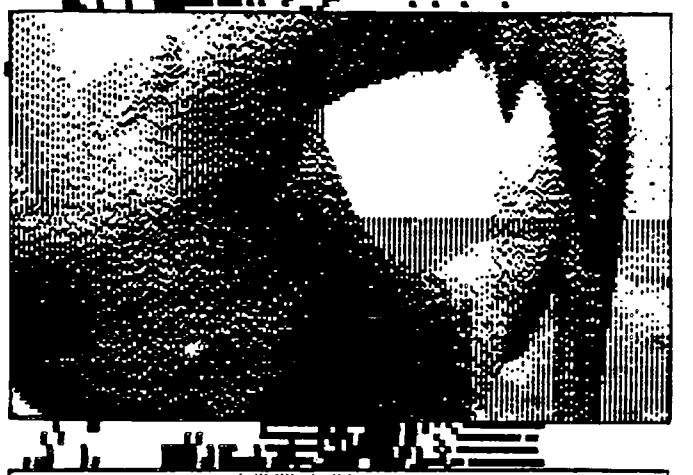


Diagram 4: Macintosh text/graphic w/ Bible verses.

Graphics Primatives, Sound Primatives and Fundamental Concepts for the Electronic Arts (an autobiographical survey)

Walter Wright
Virginia Commonwealth University
School of the Arts
325 N Harrison St
Richmond, VA 23284
(804) 257-1709

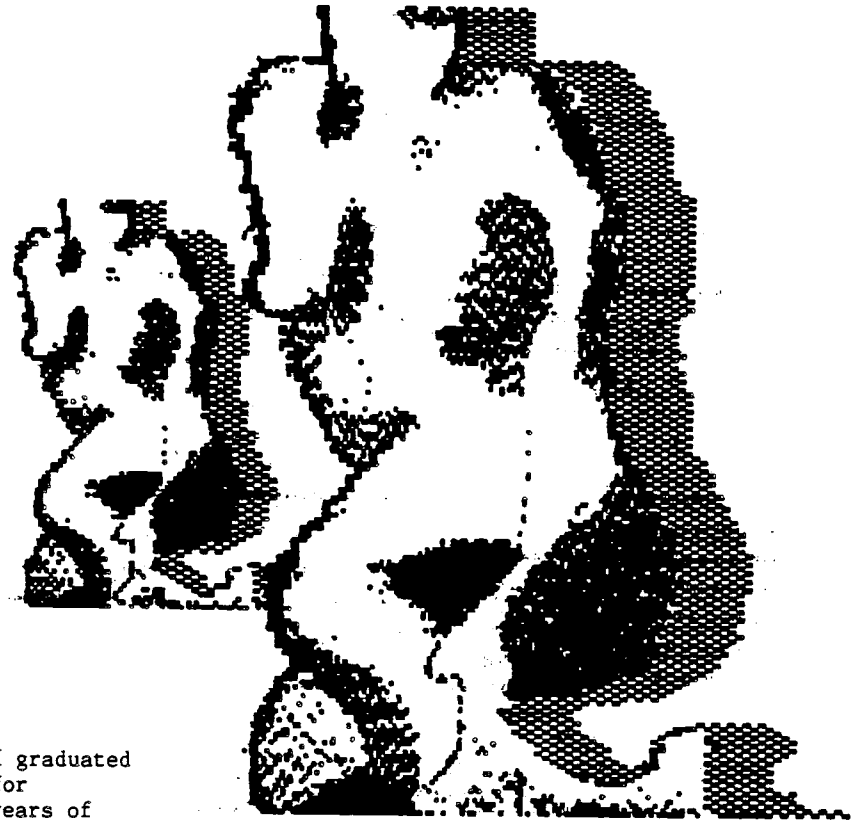


INTRODUCTION

The easiest way to put my paper in perspective is to tell you how I got where I am. That would put things in a framework and showing the relationship between several areas which may seem to be seperate but really aren't, like video and computer graphics. Computers are being used to create images for video animation and digitally controlled video effects.

This paper is part of a three part sequence. David Cook will talk about new tools for artists, in particuar computer graphics artists. He's will talk about AT&T's new Vista board. I'm supposed to talk about the basic tools that go into the electronic artists' toolbox. I've decided to be very general and include graphics primatives, sound primatives and some of the fundamental ideas that should be part of any electronic artists toolbox. Dr Bill will have the final word in this sequence. He will talk about algorithms for electronic art: how to use the tools; the process of creating a work of electronic art. Even though the material is technical, this should not intimidate artists at all, they're just as smart as anyone else. It shouldn't intimidate the art appreciator either. After all, in order to properly understand a work of art one has to understand the things that go into it. It's impossible to appreciate electronic art and electronic music without some appreciation of compositional techniques, tools and instruments, and a little bit of history... it certainly helps anyway. What is electronic art? It's a pigeonhole in which we place art using technology and, in particular, electronics including video, computer graphics, electronic sound, and even interactive environments. Electronic art is a category, it's not art, just a category in which to place these works. Art, I think, is the experience of the work, the thing the artist makes, the final result. It's not the object itself, the program or the print. It's not the recording. It's the actual experience of hearing and seeing the work. That's an experience shared by the artist and the audience alike. Artists put together the basic elements in ways that evoke or create meaning for the artist and the viewer or listener. Hopefully this experience brings insight. Perhaps the artist is trying to share some particular insight he/she has in relation to the medium or the subject of the work. The final result of this experience will be a deepening of understanding of the medium and of life itself. I would guess that the purpose of art is to enhance the quality of life and therefore one should come away from the experience both entranced and enhanced.

By the way, I wrote, or rather, dictated, this paper while driving cross country from Sheridan, Wyoming, to somewhere in Ohio.



WATERLOO

My background is in architecture. When I graduated in architecture in 1965, I was hunting for something to do other than repeat five years of architecture school in two years for a masters degree. The only exciting thing happening at that time was computer graphics. It was happening only at a few places, and only on very large machines. I decided that I should go to one of those places with a large machine. I was reading about people like Ivan Sutherland who had developed Sketchpad in 1963. I went to the U of Waterloo, Ontario, to study systems design. I was interested in computer graphics, and they had a large IBM mainframe.

At Waterloo I saw a film from Boeing in Seattle which showed airframes flapping their wings up and down in an exaggerated manner. I suppose it was intended to depict stress. On the same piece of film was a sequence shot from the cockpit of a new commercial airliner landing at Seattle (?) airport. All of this work was line drawing, in vector mode, as wireframe models. Each frame was plotted on a standard plotter. This was the heroic or brute force method of computer animation.

At that time the only computer graphics was vector graphics. Things were drawn, not on a video monitor or television screen, but on something more like an oscilloscope, a vector graphics display. Each line was drawn individually. The kind of mathematics used to draw these pictures was simple analytical geometry with a little bit of trigonometry thrown in. All this was familiar to students of engineering drawing, a class which I taught at the U of Waterloo. It was my pennance for access to the computer graphics facilities. At this point in time, graphic primatives were fairly straightforward; a computer, data storage, usually a tape drive, and a plotter. The plotter used a couple of simple commands; move with the pen up and move with the pen down. Computer graphics was black and white, or green and white on the vector display screen. The basic format was to move the pen from where it was to wherever you wanted it next. You either lowered the pen and moved to a position which created a line or vector, or you lifted the pen and moved to a position without drawing a line. Computers were used for engineering and architectural drawing. This meant drawing rectangles, circles, arcs, and doing various kinds of shading or hatching as well as various symbols and lettering. The graphic primatives were built on the available hardware including the plotter and the vector display. They dealt with pens up and down and simple geometric contructions such as lines, rectangles, circles and arcs (curve fitting came later).

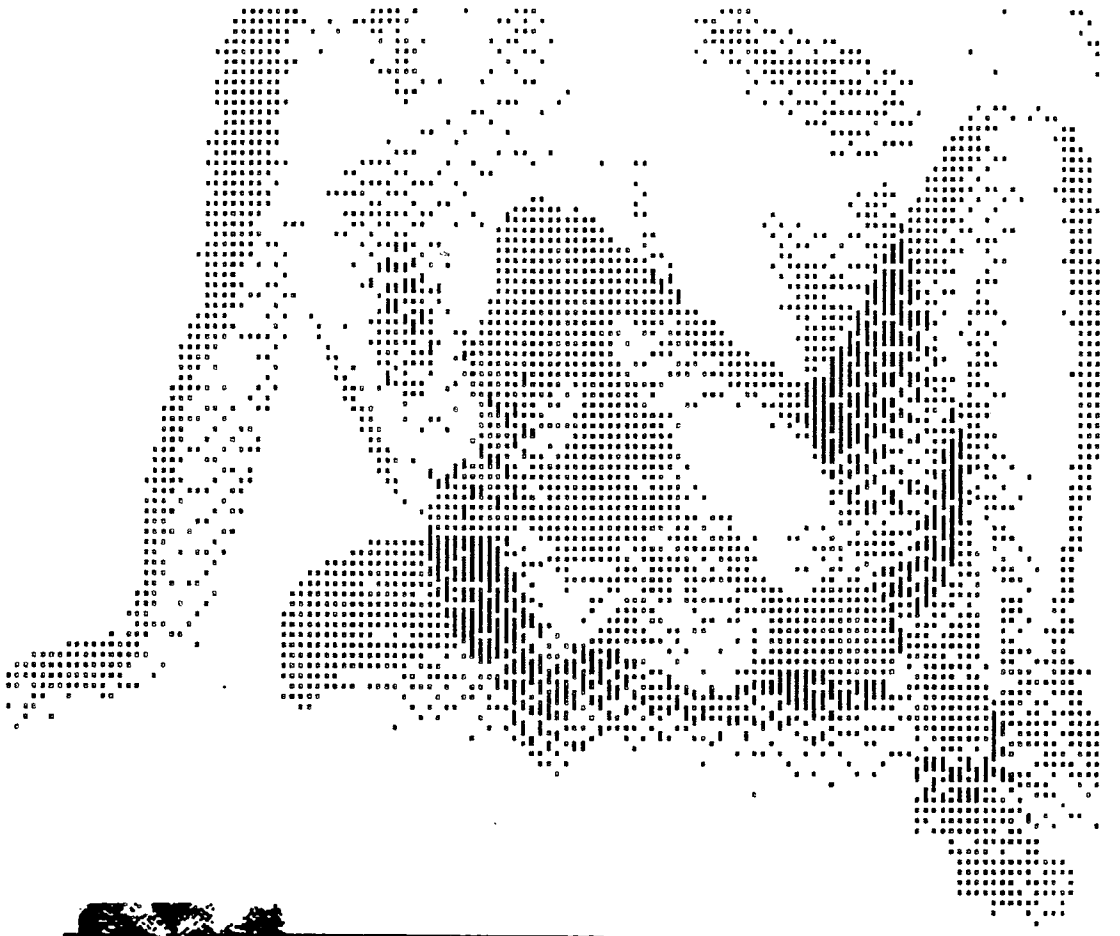
I should say a little about the kinds of mathematics that go into vector drawing. There's two dimensional, 2-d, and three dimensional, 3-d, drawing. There are two coordinate systems, cartesian and polar. In the cartesian coordinate system a point is defined with x and y coordinates in 2-d and x, y and z coordinates in 3-d. For polar coordinates it's a direction and a radius in 2-d, and two angles and a radius in order to locate a point in 3-d. Polar coordinates become spherical coordinates in 3-d. Once a point is located, equations can be used to describe a line, and a line segment or vector can be determined by start and end points. As the plotter steps along, it moves in the x direction or in the y direction, the jaggies.

A software routine is required to draw sloped lines. A line can be described by its equation which is handy in determining intersection points with which you can clip and create windows. There are routines to draw circles and arcs. In order to do this the data is transformed from one coordinate system to the other, from polar to cartesian and visa versa, using sines and cosines. There are formulas for circles and for ellipses and, by modifying these formulas, you get a generalized routine that will draw an arc in any direction and from any starting angle to any ending angle. With these tools you can do 2-d line drawing. With some additional trigonometry it's possible to extend this to 3-d such as Boeing's flapping airframe which travelled through a 3-d world as seen out the cockpit window, landing at a database representing an airport. This requires perspective as well as routines for translation, scaling, rotation and clipping.

I mentioned Ivan Sutherland and Sketchpad. I saw that film and other films by John Whitney, then artist-in-residence at IBM in New York City. He was creating computer graphics on the new 2250 vector display unit. The graphics were shot directly on a microfilm plotter. He was producing abstract films based on rhythms of geometric shapes, sequences of kaliedoscopic images. We saw several of these films including Permutations. At that time a student from Los Angeles joined our group at Waterloo. John Stehura had worked with John Whitney's sons on a computer graphics film that was generated in the meteorological labs at UCLA using their IBM mainframe and microfilm plotter. It was optically printed in a number of passes to add color. Essentially it was the same old thing, a plotter with pen up and down to join the points. John introduced me to the concept of metaprogramming, another item for the electronic artists' toolbox. It's an idea upon which programs can be built. Metaprogramming is to program the rules for creating the image but not to program the image explicitly. To put it simply... when creating an image I may use a number of lines, circles, rectangles. The choice of what I'm going to use and the number of each is determined by the program using a process of random selection. John's idea was to create complete animation sequences. He would set rules such as how dense they should be, colors to be used, speed with which shapes should move, whether they should zoom up or back, whether they should move in a particular direction or various directions... these sorts of things. He determined the structure but not the individual frames of the sequence. John called that metaprogramming; Dr Bill calls it Algorithmic Art.

About this time a salesperson appeared at the U of Waterloo with the latest thing in the field of music education, the Moog synthesizer. Robert Moog had designed a modular music system which he was making in Trumansburg, NY, only a days drive from Waterloo. I went to visit the factory. The Moog was not unlike John's idea of metaprogramming. It got me thinking about another important idea and about hardware and with software. The new idea was the modular system. Bob Moog had broken his machine into a series of modules that could be patched together and that communicated with each other using standardized signals. In the original Moog there were a number of signals: trigger pulses, control voltages and the sound signals themselves. The modules included filters, equalizers, mixers, amplifiers and oscillators. They were voltage controllable and could use control signals from other modules. the oscillators could be used as both sound sources and control voltage sources. A keyboard controller, a ribbon controller, and even a computer could be interfaced; although that hadn't been done at the time.

I proposed that Waterloo buy a Moog but, until that happened, I experimented with that idea of modularity in software. I made an oscillator subprogram, another subprogram that was an amplifier, etc. Then I sent these subprograms a parameter list representing the front panel controls and control signals, and patching information. I started programming my own model of the Moog synthesizer in FORTRAN and used it to generate John Whitney like images. I wrote subprograms representing the modules, then a main program, a metaprogram, which turned things on and off, determined what things should be drawn in a frame, decided on frequencies, size or amplitude... things like that. My program was software and the Moog synthesizer was hardware. This parallel interested several people at Waterloo who were thinking about digital to analog conversion and analog to digital conversion. They wanted to control an analog instrument with a computer. So here are some more concepts: digital versus analog, and hardware versus software. Both are important to the electronic artist, who should be able to move back and forth between hardware and software, digital and analog. More on this subject later.



NEW YORK

After graduating from U of Waterloo in 1967, I went from Toronto to New York. I wrote an architectural drafting system for a firm called Environetics, a large interior design contract firm. It generated architectural floor plans, electrical and mechanical drawings, furniture layouts... all the required contract drawings. I did this on an IBM 1130 using a Xynetics flatbed plotter. It was 1970, I paid my dues in exchange for access to this machine, I wore a tie for two years and went to an office on Madison Ave. Since I wasn't fulltime after the first year or so, I contacted another firm a few blocks over called Computer Image Corporation. They had Scanimate, an analog machine that did raster animation. This was my introduction to raster images. Up until this point I'd been working exclusively with vector images and with plotters or vector display screens. I became interested in video. Down in the Village there was an electronic arts theater started by Woody and Steina Vasulka called The Kitchen. It was in the Mercer Arts Center and it featured video, electronic music, almost any electronic art form that one cared to bring in and share. Woody and Steina were working with a 1/2" b&w portapak and simple video special effects devices. One of the first people to exhibit at The Kitchen was Nam June Paik. He had developed the first video synthesizer. I believe that would be a few years after Lee Harrison III developed Scanimate. Mr Harrison started sometime in the 1950s with the idea of making the job of animation a little easier. He wired up an actor or dancer who would then go through the kinds of action that you were interested in animating. This was Animac; an exoskeleton that the actor or dancer wore, his/her movements were plotted on an oscilloscope as a series of balloon shapes, kind of like a learn-how-to-draw-a-figure book you would find at your local art supply store. Animac proved a little unwieldy so Scanimate was designed to animate things like copy or text. Animating copy for television advertising was a big job for traditional cell animation but with Scanimate it was cost effective and quick. Scanimate used a b&w high resolution television system, a 1000 line system broken into five segments and with analog control moved those pieces of the raster around on a high resolution display. That in turn was rescanned by a regular format television camera, a 525 line system, the grey levels were quantized and colors assigned.

I was pleasantly surprised to find that the analog control signals used to move the raster were generated by a series of modules including oscillators, amplifiers, triggers, and ramp generators. All of these things were familiar to me as software that I'd written for animation, and as modules on the electronic music systems I'd played with. After I was trained and working at Computer Image Corporation, I began to make experimental tapes late at night and on weekends. I took a John Cage approach to my video compositions, since I'd been reading about his work. I rescanned images off live network TV and ran them through Scanimate electronically assigning color and distorting the images using the modules provided and then put control signals on the tape as a soundtrack. I showed the results of my experiments at The Kitchen. I would talk with people there about the kinds of equipment that they had access to and the kinds of equipment they were building, things they were doing in the developing area of video art. Perhaps I've strayed slightly from computer graphics but a lot of this information and these devices showed up later in computer graphic systems.

I can add a few more hardware primatives, from my experiences with Scanimate, including ramp generators, triggers, and delays. I've mentioned the concept of a modular system, meaning that signals can be interchanged module to module; control modules, processing modules and signal generating modules can all be linked together and exchange signals of various kinds. A few more modules; quantizers, attenuators and control voltage sources should be added. Later all this was sorted out by Serge on his modular music systems where sound, triggers, and control signals all became the same and could be plugged back and forth to one another. All of the signals became interchangeable. I can introduce some video modules related to the area of video art and the kinds of things that were happening at The Kitchen: the keyer, and the colorizer; and I should talk some more about Nam June Paik. He had built the first video synthesizer. I should add video mixers to the list of hardware primatives. The Paik/Abe Video Synthesizer (Shuya Abe assisted Paik in its design) was a large mixer with six b&w camera inputs; color, almost like watercolor, was washed over the b&w images. All the color signals were mixed together to a single output. The PAVS was a specialized video mixer, similar to the voltage controlled amplifiers and mixers used in the sound synthesizers.

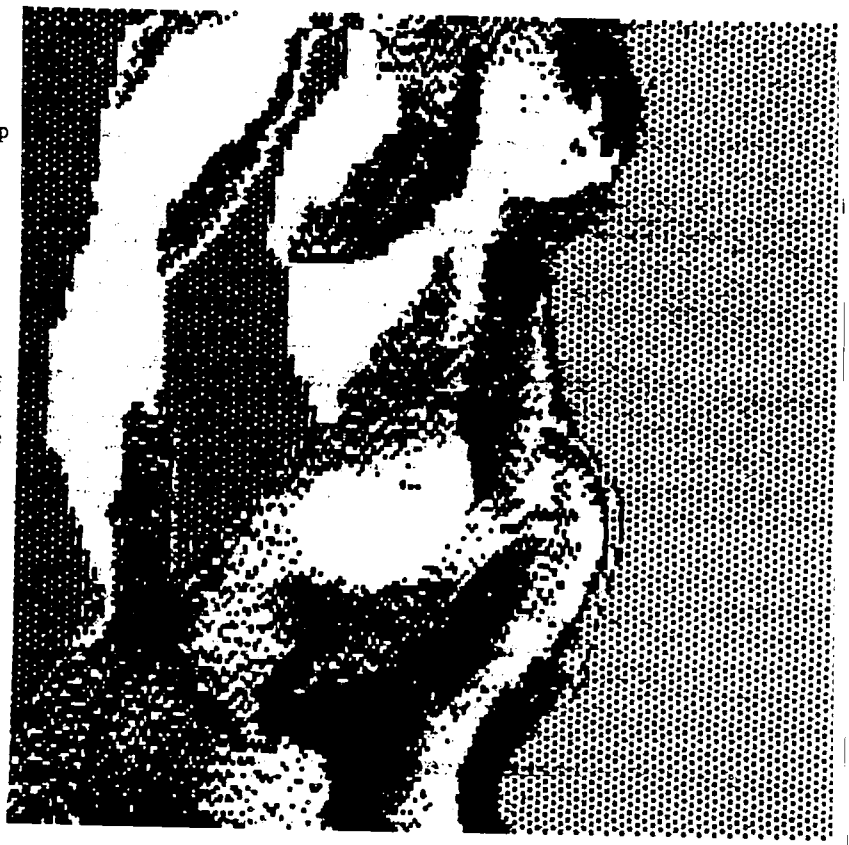
So far I have identified some basic or fundamental ideas; now for some definitions. Analog versus digital: an analog system models something such as sound or the brightness of an image by using a continuously varying voltage. In other words, the continuously varying voltage becomes an analog for the sound: the air pressure which moves the membrane in our ear back and forth, or as created by the vibration of an instrument. In video it represents the brightness levels across the picture. Digital systems do not represent the information as a continuously varying voltage but rather as numbers, as a series of on/off pulses or digits, 0 and 1, which taken as a group form numbers. Put the numbers together and the result is a representation of something like an event, a color,a position, a level or an amount of something. Remember the idea of modularity, as in the Moog synthesizer. Its interesting to note that Scanimate and the PAVS were self-contained or closed systems. They were nonmodular or at least not very modular. In music it's slightly different; the keyboard controllers were set up to play traditional scales, envelope generators simulated traditional instruments with controls for attack, sustain, decay, release. Later the envelope generator was taken apart, broken down, by Serge and in his system we find ramp and slew generators as in Scanimate.

Here is another fundamental idea or concept which requires definition: vector versus raster. Vectors are easy to deal with digitally, they're quantifiable. The plotter deals in steps, a given number in the x direction and a given number in the y direction. A raster is defined by time, a regular scan, the image is created by scanning rather than drawing vectors in a random manner. Raster scanning is predictable, going from top to bottom of the screen at a fixed rate. Vector drawing occurs as fast as the device can go, the more vectors the longer it takes to draw a frame, a distinct disadvantage because, as the image gets more and more complex, it takes longer and longer to render. The frame rate drops and the image starts flickering and the illusion of smooth motion is lost; it fails as realtime computer graphics. Of course if the device is a microfilm recorder or plotter you can draw frame by frame and project the result at a normal film rate thus preserving the illusion of smooth motion. In a raster system, since the frame rate is fixed, it's a problem of getting all the information to the screen in a reasonable time. There is only a short period of time to update the frame; during the vertical interval while the beam is flying back to the top of the screen, or by "pingponging", that is, loading up one image and displaying it while loading up a second image, bouncing back and forth between the two screens. I'm getting a little ahead of myself because the raster systems I started with weren't digital they were analog; Scanimate and the PAVS. They worked with realtime inputs, they didn't calculate or remember information. They modified information in realtime using various hardware modules similar to those used in music or sound synthesizers. The realtime instruments were analog and the instruments that worked outside of realtime were digital.

I'm in New York maintaining my computer-aided drafting system for Environetics, working part-time as a video animator on the second machine at Computer Image Corporation/Dolphin Productions and making tapes to be shown in the evenings at The Kitchen on the weekends. The TV Lab at Channel 46 invited various artists to work there; Woody and Steina and Ed Emshwiller. I'd seen Ed's film Relativity several years before. He had become interested in video and turned up at Computer Image Corporation one day wanting to work on the weekends and make a video piece for WNET. He was the first one to put in drawings on the spot; he drew we shot. He recorded directly, mixed his material live and later edited it to a sound track. The result was called Scapemates; the first and possibly the last piece of video art run on Channel 46 that was created on Scanimate.

BINGHAMTON

In 1972, I met several people from the New York State Council on the Arts and Ralph Hocking from the Experimental Television Center in Binghamton. Bored and disillusioned with Madison Ave, I decided to become artist-in-residence at the Experimental Television Center in upstate NY. So I left behind the big computers and all the expensive equipment like Scanimate, the Xynetics plotter, the 2" video equipment and all that stuff. I went up to the ETC which had a Paik/Abe Video Synthesizer, a bunch of portapaks and a 1/2" video editing system. Here we find another of those fundamental ideas that was important to the development of electronic art: small format video-- technology that's accessible. Up until this point, people that worked in electronic arts had diffcult time gaining access to big machines and facilities that were owned by corporations or universities. They worked on weekends or after midnite. The portapak, the small video camera and recorder, made this technology available to individuals. An artist could make his/her own videotapes, edit them, add sound, and so forth. Paik was the first artist to do this. He picked up his portapak and on the way to a performance, taped the Pope's motorcade which he passed on the way, and showed the tape that evening. Off to Binghamton I went, to use the portapak and to demonstrate Nam June's video synthesizer at non-profit institutions around the state. Small format video facilities were available to individual artists through a number of public access studios in NY state. People borrowed a portapak, made a tape, edited it, colorized it (if they were working at the ETC), and finally, played it back on their own televisions as their own home video art. This was a great idea; I still think it's a great idea. It could make electronic art a household word. The newer 1/2" editing equipment incorporated an important technological advance, capstan servo editing that allowed both assembly and insert edits. The first person I knew to take advantage of this development was Gary Hill of Woodstock Community Video. In his tape Rock City Road, Gary did things with his 8650 that had only been possible with very expensive 2" editing equipment at real TV stations. The combination of the small video recording machines, the portapacks, and the development of the capstan servo editor forms another fudamental or generative idea in the field of electronic art.



What was going on in Bingamton? The ETC thinking about using computers, the small computer wasn't out yet but there were kits available and there were older machines available to non-profit organizations like the ETC. David Jones, engineer-in-residence, began to develop electronics so that eventually all the Center's equipment could be controlled with a computer. One of the first things David built was a extension of the keyer and quantizing colorizer; the Jones' Colorizer. This device was different than the PAVS in that it allowed you to mix color, using RGB components, something that should be familiar to computer people. It was an analog system but

it as a controller for David's modules. Things were becoming digital.

There's one thing that isn't obvious but should be mentioned; most artists had little experience with computers and computer programming. Being able to program a series of events, to store a series of control signals for audio or video; this was nothing new to me or to the people who had been working in computer graphics. I don't remember thinking much about it, but I do remember teaching people at the ETC programming. I do remember meetings at Media Studies in Buffalo where the hot topic for discussion was... should electronic artists be learning programming... and the conclusion was: Yes, they should. Progamming is a whole new way of thinking about things.

I'm on the road to DeMoines Iowa, and just passed through Omaha Nebraska. It smelled like banana bread.

About this time, or a little before, The Kitchen collapsed, literally not figuratively. The Broadway Central Hotel fell down. Woody and Steina were looking for something to do. The Kitchen rose from the rubble and reconstituted itself as a legitimate art gallery. The Vasulkas worked at the TV Lab at Channel 46 for awhile then moved to Buffalo, NY. This was probably bad for them but it was good for me because Buffalo was accessible. They were at the Center for Media Study at State

University of New York. Woody and Steina both had teaching jobs and all their savings went towards equipment. Woody became interested in digital image generation and computers. He started to teach himself digital electronics and programming. He bought himself a SOL20. As I've mentioned, the ETC was interested in that sort of thing, David Jones was designing colorizers, keyers and various little boxes, and thinking about computers. It was through Woody and Steina that I first heard about Dan Sandin out in Chicago. Dan built the video version of the Moog synthesizer. His Image Processor included oscillators, comparators, differentiators, mixers, a quantizer and a colorizer. The Moog synthesizer was put together using discreet components, the circuits boards used lots of small parts. Sandin's Image Procesor was the result of a development in the electronics field, the op amp. This is an integrated circuit called an operational amplifier; voltage controllable, it eliminates the need for lots of small parts. Most of Sandin's modules, in order to make them inexpensive and easy for artists to build, were based on a single high frequency op amp suitable for video. David Jones at the ETC took a slightly different route, he designed his own op amp out of discreet components. Later, when I got my Serge synthesizer, I found that Serge had designed his modules based on a single op amp. An important feature of Sandin's IP is that the video signals and the control signals are the same. They can be used interchangeably, video can be used a control signal and control signals can be used as video. David Jones' systems will accept control signals as video but not visa versa.

I'm trucking the PAVS and Jones' Colorizer around NY state. Gary Hill at Woodstock Community Video became interested in frame buffers. He bought a Serge, one of the first Serges produced, a very impressive looking thing with lots of patchcords. Gary was interested in interfacing his Serge to some custom designed video equipment. He asked David Jones to help him develop a colorizer, keyers and a frame buffer all controlled by the Serge. It was an integrated analog and digital system, it wasn't a computer but it was moving in that direction. The frame buffer depending on high speed analog to digital, A/D, conversion uses part of a computer, the memory. Woody was working on the same thing, a framebuffer, and he expanded his frame buffer so that when images were read out from memory they passed through an arithmetic logic unit, ALU, which is also part of a computer. In fact, Woody's frame buffer was controlled by an LSI-11. The ETC got itself an LSI-11 and developed modules were seperated out. Four b&w input images were passed through four seperate channels. Each channel had two keyers, one that worked from the top down and another that worked from the bottom up, so that you could isolate areas of the images. To these areas you could assign color and the result went to an output mixer. The Jones' Colorizer was essentially analog but it made extensive use of keying which is, in a sense, digital, on/off, 0 or 1. It was somewhat modular, it was designed to be voltage controlled. Along with the colorizer David built a bank of oscillators, ramp generators and sequencers.

GRAND RAPIDS

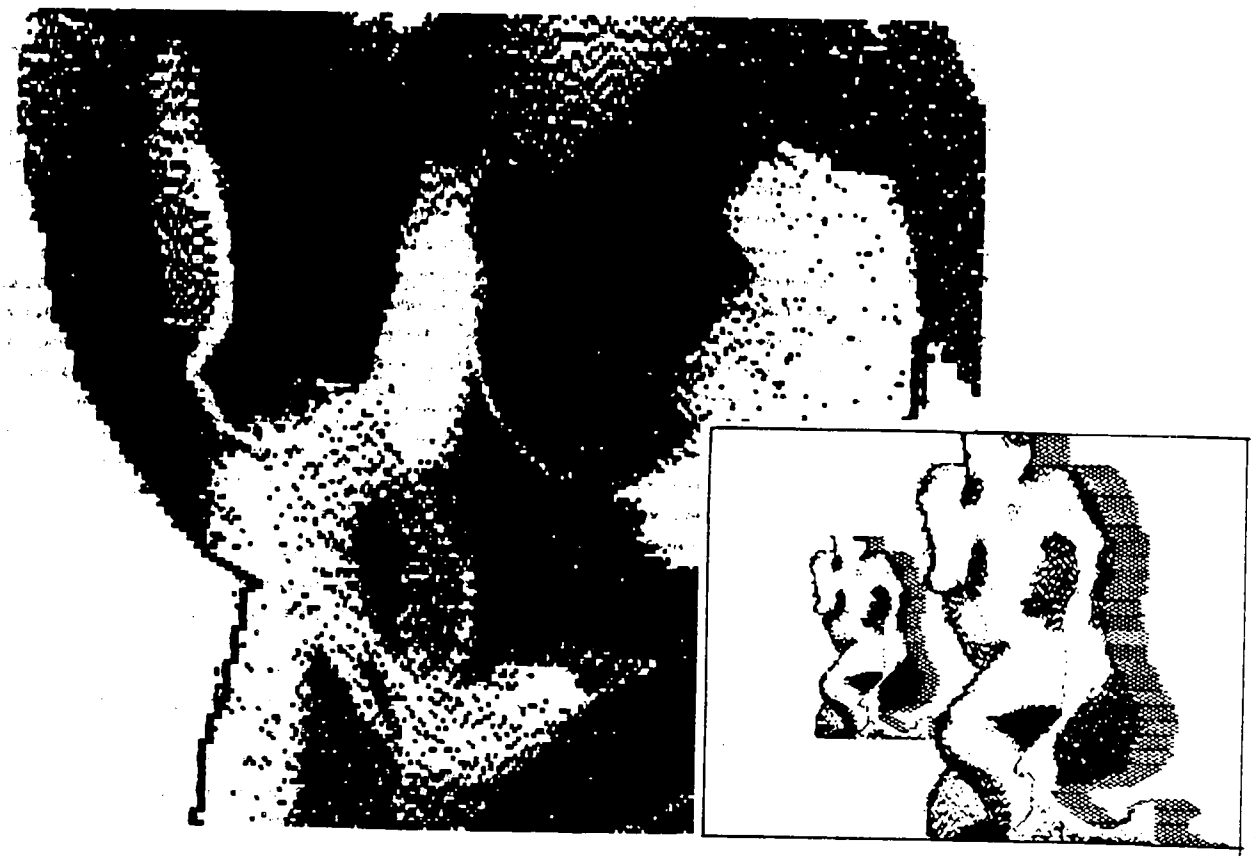
In 1976, William James College in Grand Rapids, Michigan, wanted a visiting video artist. They called Nam June Paik. That would have been wonderful, Nam June Paik a resident of Grand Rapids. Unfortunately, Nam June wasn't available but fortunately, I was. I went to William James College as a visiting artist for a semester in their Arts and Media program. I took along the PAVS from the ETC and I took along a bunch of modules I'd gotten from Serge. I started to put together my own Serge Modular Music System. My semester vist turned into a fulltime position. I was close to Chicago so I wrote to Dan Sandin, got the plans for the Image Processor and with a couple of students started to assemble modules. I had a chance to look into electronics, build some IP modules and put together a Serge Modular Music System of my own.

About that time the Apple computer appeared. This gives me chance to return to a recurring theme, accessibility and affordability. Here was a computer that was accessible. It was possible for an individual artist to own one. Even though I had to get a loan from the bank, it was possible. I didn't need a state grant. I didn't need to suck up to some corporation or institution to get time on their precious mainframe at 2 o'clock in the morning. You could have a computer in the livingroom. You could own a computer which could be programmed from top to bottom, allowing you to gain a deeper understanding of the tool or medium. Not only can you just program, you can program in assembler. You can build circuit boards and put them in your machine and watch or listen to them run. The idea of the small computer as a personal appliance or as an entertainment unit has perhaps reached its peak, or saturation point. But it's still important to remember that the small computer can be used by an artist, it fits in an artist's studio and it's affordable. It's possible to be seduced by more and more computer power, by higher and higher resolution, by greater speed, by thousands of colors, by multi-tasking, by desktop video, any of these things. It's just a little bit more money, just a little bit more. If I worked a little harder for a few more hours, if I got this grant or that grant, I could do it. But there's a threshold point, a point of diminishing returns. Perhaps, the computer companies in their efforts to sell more machines by raising the capabilities and the price of their machines are putting them beyond the reach of artists. They don't represent a very big market but, maybe, artists ought to hold out for their own machines.

I put the Apple in my livingroom. I was interested in seeing what kind of things I could develop on it. I should talk a little bit about programming languages. These languages contain some of the graphic and sound primatives I'm supposed to be discussing. In Applesoft BASIC, the pen commands have disappeared and in there place are these graphics commands; COLOR= and HCOLOR= which allow you to define the drawing color, PLOT and HPLOT commands which set a location on the screen to a preset color, HLIN, VLIN and HPLOT TO which draw line segments. The sound primatives are POKEs for pitch and duration, clunky at best. I worked in CEEMAC, a graphics language written by Brooke Boering which includes a number of graphic primatives; the COLOR= command, ADOT turns on a point, BLINE draws lines, ABOX draws rectangles either outlined or filled, SPLINE draws curved lines. CEEMAC also has; ANOTE that sets both pitch and duration for a sound. CEEMAC has some additional effects; RASTER which allows you to move the image up and down, PULSE which flips or inverts certain colors on the screen, etc.

Here's a graphics primative I haven't mentioned; it goes back to the old vector displays. Vector displays had a display buffer, a precursor to the frame buffer, and in the display buffer was the display list. The display list was broken down into groups of vectors or line segments that defined a shape. In Applesoft BASIC there is a shape command, DRAW, and an absolutely annoying, completely ridiculous method of coding shapes. In CEEMAC there is a shape primative, SHAPE, and a much more manageable, much more enjoyable way to code shapes. A shape is a collection of vectors, or a group of lines. CEEMAC and Applesoft BASIC are limited to 2-d shapes. But I've worked with GraFORTH. There were a number of people interested in GraFORTH at one time. It has a shape primative which s 3-d. It acccepts 3-d data and allows you to build up 3-d shape tables that can be translated, scaled, rotated, and shown in perspective. You can do short 3-d animation sequences on you Apple using GRAFORTH. It is an extremely clever language and fun to use. It never became popular. I suppose because it is a programming language that requires some background in programming and in graphics; most people struggle along trying to learn BASIC. GraFORTH has a sound primative NOTE similar to the primative in CEEMAC, in addition, there is a primative for TIMBRE.

Add metamorphosis to the shape primative. Metamorphosis is an animation technique; the changing of one shape into another. You can do this with shape tables. There has to be the same number of vectors in each table so that the points will map one to the other, often in a clockwise order . There's linear interpolation and sine wave interpolation; Dr Bill showed some examples of this at SCAN '84. Animation is important, in order to use the new desktop video systems, it helps to understand traditional cel animation. There are a number of interesting things to be considered when it comes to animation. One can think about the difference between the expensive chromosaurus animation as opposed to Will Vinton's singing raisins; it shows you what a real animator can do as opposed to a programmer.



Back to my story, I didn't program immediately in CEEMAC on my Apple and I didn't create 3-d animation in GraFORTH. In fact, when I got my Apple I was running off a cassette tape; not a very happy experience. I got a disk drive. Like everybody else who had a background in programming on plotters, I was unfamiliar with raster graphics. The tools that Apple gave you were, at best, rudimentary. So it was necessary to build up a series of small programs in BASIC that were artistic or, at least, along those lines. Something that embodied creativity. Something that wasn't just blowing up spaceships. Essentially trying to discover what the artistic potential of the machine really was. Granted, as in video, working on the first Apple my expectations shouldn't, in fact couldn't, be all that exalted. The machine had limits; very low resolution with a few colors... the "high" resolution mode was really 140 x 192 x 6 colors including black and white. That's not much and the colors themselves weren't all that hot. So there I was stuck with this thing, programming in Integer BASIC on a machine with limited memory. Things did not look all that promising but, as I'm an optimist, I saw potential in my homely machine.

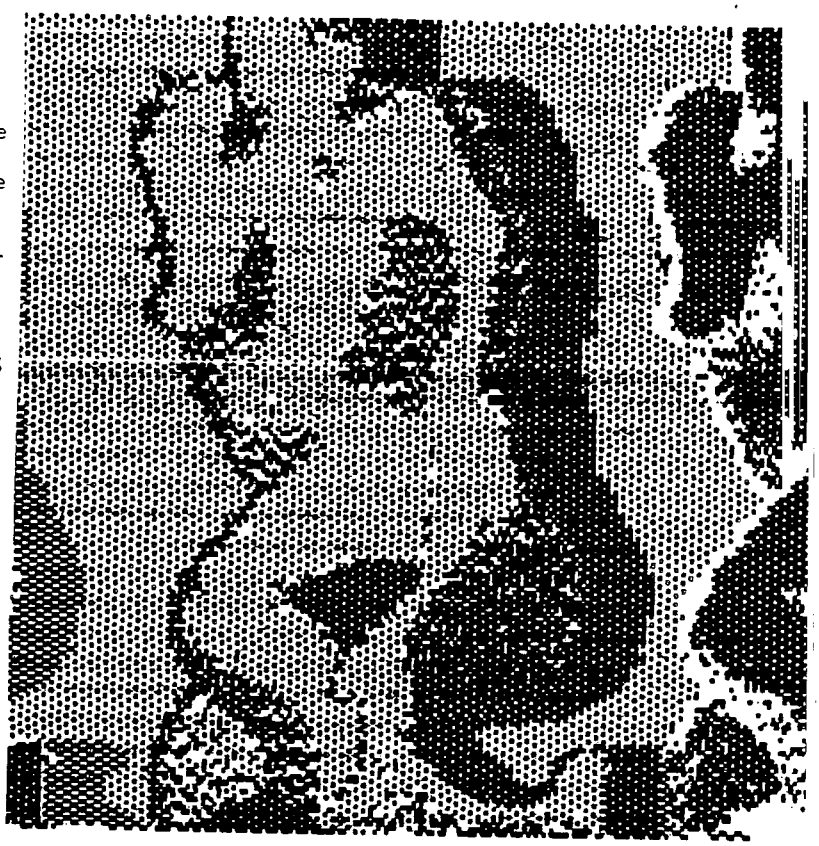
I should mention a couple of books. Expanded Cinema by Gene Youngblood, that was the book that got me; confirmed my wildest dreams and suspicions about electronic art. It's about computer graphics, video and experimental film. Expanded Cinema includes John Whitney and John Stehura and his film Cybernetik III. A second book became important to me, Point and Line to Plane by Wassily Kandinsky. What interested me about Kandinsky's book was his approach to the problem of visual art, in his case painting and design, and the idea of design principles. How was I going to determine a direction to take with this new computer? Now I had to tackle the raster, to deal with points, lines, texture, color, contrast,

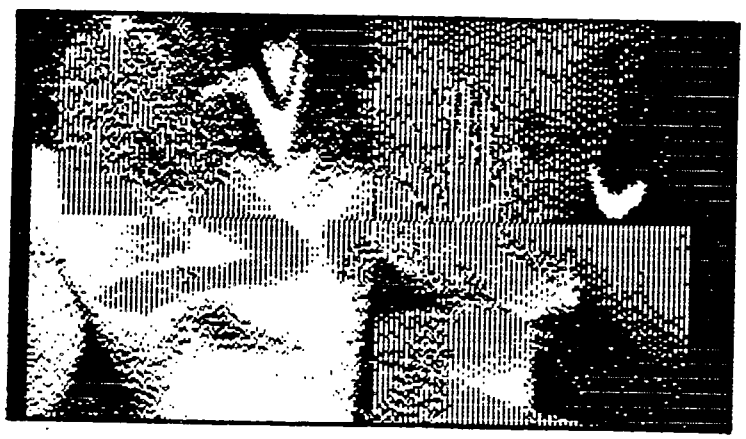
rhythm...all those basic things that are called design principles. Another generative idea is these graphic design principles. They are useful because they can be embodied in a program. In other words, you can develop a program that will look at or work with a certain set of proportions, not fixed but flexible; one proportion could depend upon the next and so on. Design principles can be programmed.

The west side of DeMoine is one large traffic circle, one of those that seems to go 360 degrees around and you're sure you're headed back the same way you came. The signs however assured me I was going east. Then they disappeared and for miles there were no signs saying where I was. But I've recovered and I'm on I80 east headed for Davenport. There is another book I thought of while driving through DeMoines; Cybernetic Serendipity by Jasia Reichard. It was a pivotal book, which along with Expanded Cinema, layed out the possibilty for a whole new area involving art and technology.

But back to my story. I was teaching at William James College. I brought my computer to school and set it up for a while, a couple of students experimented with it. They used it in some films and videotapes. The school got an Apple II+ and there were two computers, enough to teach a course in BASIC programming and computer graphics. I ran across the Fire Organ disk by Brooke Boering and sent for the CEEMAC language. I experimented with CEEMAC, and maybe the first CEEMAC course ever taught was at William James College in Grand Rapids. It was a success.

My studio now included the Serge Modular Music System, the Apple complete with disk and color monitor, CEEMAC and FORTH. I'd been working a couple of years on developing programs for it and various ways of dealing with design and stuff like that. Another advance in technology became available, the Number Nine Graphics System. Brooke told me about the board, it used a new graphics chip, it was its own self-contained graphics system that could be put inside the Apple computer and it had a resolution of 512 x 416 x 16 of 4096 colors. Well that was pretty amazing, I had to have one, I took my tax refund and purchased a NNGS board and an extended memory card for storing images. I taught myself Apple assembly language, it seemed an obvious thing to do since the software that came with the NNGS was deplorable. I was introduced to a brand new set of graphics primatives and found an application for some old ones. Once in assembler, you can use logic functions, and when working in raster graphics you can take portions of the image or whole images and combine them in a variety of ways. Not only can you combine them by replacing or copying, but you can perform arithmetic and logic operations on them at the same time. Images can be AND, OR and XORed together. They can be ADDed and SUBtracted together. They can be averaged and here the concept from video pops up again, keyed. That is, given the color information at a point or pixel one image can either replace or not replace another, 0 or 1, producing a key or matte.

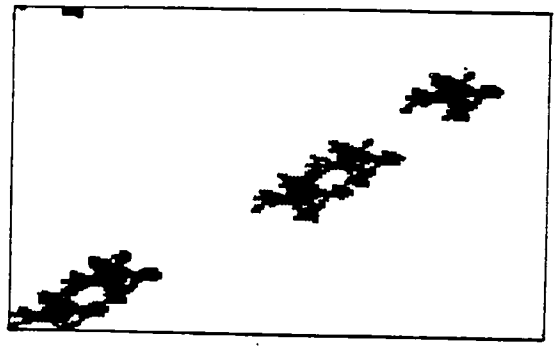




I met Dr Bill at SCAN '83, he had started his career next door in Lansing, at Michigan State. He'd graduated with a masters degree in computer graphics (he says). Then he'd gone to Ohio State and became Dr Bill the computer graphics art educator. He'd started with a mechanical drawing kind of background or reverted to that as a means of gaining access to the university's mainframe computer and plotter. His background was in vector graphics and plotter graphics as was mine. We'd both taught mechanical drawing as penance to gain access to our university facilities. Dr Bill had contacted some of the people I'd read about in Cybernetic Serendipity, those Europeans who'd done interesting graphics using plotters. I associate Dr Bill with another fundamental idea, the random number generator. Again this is familiar to programmers. It's probably not as familiar to people from traditional fine arts. The Europeans were into random numbers. There's quite a bit to be said about them. Applesoft BASIC does not contain a very good one; CEEMAC contains an excellent assortment; in FORTH you can construct your own. I noticed in a new music language by Allan Strange, based on FORTH, he had constructed some special random number generators for music composition. I've experimented with white noise, brown noise and, most recently, with fractal random number generators. I began experimenting with controlled randomness and probability; using random number generators to determine the probability of events. These probabilities might vary over time, or over space. You could change the probability of an event given a location on the raster.

About this time, I got together with David Cook, a student at William James College. David was an accomplished musician and I wasn't. Nevertheless, we decided to play together I on my Serge and David on the college's Moog. We did Tuesday nite gigs at the Biermeister, electronic sound improvisation. Later, David and I started Digital Image Corporation. David developed an artists' workstation based on the Cromemco. It had better resolution than my Apple, 378 x 240 x 16 colors out of 4096. He programmed the paint system for DIC, which contained the standard set of graphic primatives and a whole new set based on the unique qualities of a pixel oriented graphics system. He developed the idea of using logic functions and arithmetic functions to generate texture and pattern, simulate transparency, etc. There were additional graphic primatives, related to logic functions and usually referred to as image processing; smoothing, blending, averaging and edge detection. David included keying in his paint system; keying on a color or colors. He developed some quite sophisticated methods of matting; combining two images using a third image as the matte. You could cut out pieces of two images and recombine those in order to make a third image. You can get quite good at combining logic functions in order to generate things like cutouts for lettering, shadows, and special effects. David included skewing the raster, scanning it in different directions, and the idea of a brush, ie. cutting out a portion of an image to be used as a brush. These primatives should be part of every paint system.

To bring things up to date, 1982. Woody and Steina tired of Buffalo moved to Santa Fe. The ETC moved further up the river from Binghamton to Owego. David Jones was developing his framebuffers and the studio system is now a complete set of modules both audio and video all talking to one another. A couple of David's framebuffers were floating around; Ralph had one, Peer Bode the ETC's artist-in-residence had one. There were several in New York city. Nam June Paik was by now an international celebrity.



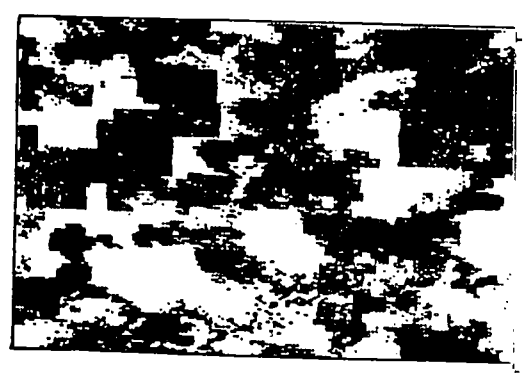
RICHMOND

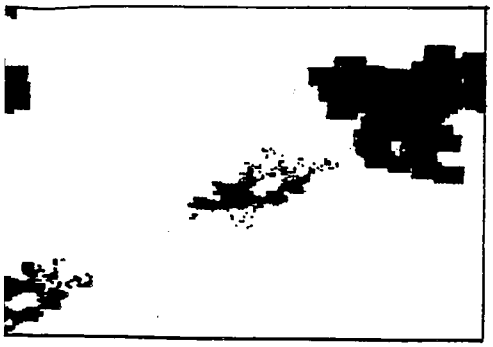
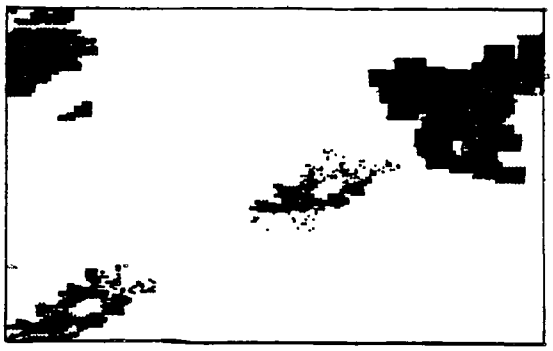
Not to sure where I am but there goes a rest stop. I'm on my way to Davenport then south on I74 to avoid the black hole or at least paying for the black hole on the turnpike.

Things were moving backwards in Grand Rapids, a resurgence of conservatism, our sister college Thomas Jefferson had been closed. My days were numbered. Luckily a position opened up at Virginia Commonwealth University. I joined the faculty of the School of the Arts at VCU in the Media area to teach computer graphics and video. I met Dr Bill again at SCAN. Our curricula were alike and each included computer graphics, video, and electronic music or sound. When I left Michigan I sold my Serge, my tape recorders and all that. I kept my Apple and the NNGS board, my S8 movie camera, and moved into a house in the country. A couple of more books: Myers book on Apple graphics and Foley and VanDam's Principles of Computer Graphics, which gives programming techniques for lines, clipping, circles, fills, world coordinates, 3-d, etc. It should be in every electronic artist's library. VCU has a large design department, my department. I teach video, computer graphics and sound communication. It's a much bigger program than William James College and it's more commercially oriented. The students weren't the alternate education types I was used to. They were interested in artists' workstations, desktop publishing, making slides, etc. I introduced CEEMAC and talked about sequencing images. I'd been teaching film and video production and film theory at Willliam James. By the time I got to VCU, I'd added a new set of primatives to my toolbox: cinematic codes, the techniques of filmmaking. Computer graphics isn't 2-d or even 3-d but 4-d. It's a timebase art like music, dance, theater and film. It moves in time; it has a lot to do with animation.

With my tax refund I bought a new Serge with A/D and D/A conversion and interfaced it to an old Apple. The same year at SCAN I discovered a digital keyboard called the Mirage. The next year's tax rebate turned into a Mirage and an Amiga. My studio now consists of my old Apple with the NNGS board and VCUtilities (the subject of my paper for SCAN '85), another Apple controlling the Serge Modular Music System, my Mirage and an Amiga 1000. The Amiga will interface to the Mirage and to David Jones' newest framebuffer.

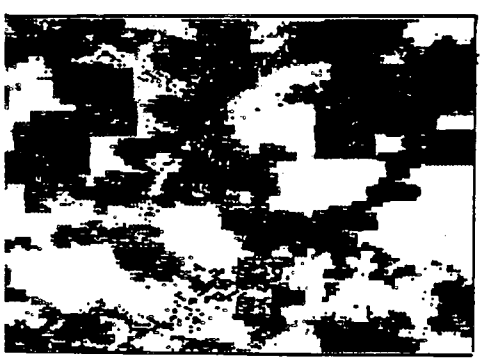
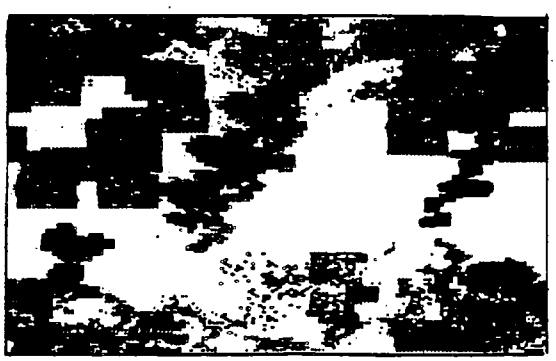
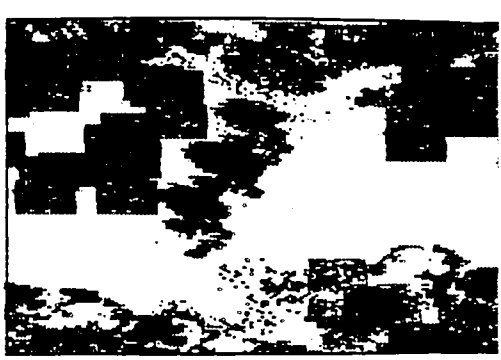
There are two types of raster systems. The NNGS board is a bitplane system where each word in memory represents a group of pixels on the screen, a 16 bit word represents 16 adjacent pixels on the screen but only represents 2 colors. If you want 4 colors on the screen you require an additional bit plane, two words of memory would have to be used together to generate a number called the color code for each pixel and that code would be used to look up the color. Bitplane systems use a colormap, a single plane can show 2 colors, two bitplane combine to show up to 4 colors, three bitplanes 8 colors, four bitplanes 16 colors, and five bitplanes, the limit of my Amiga, up to 32 colors. In the five bitplane scheme all five bits representing the pixel are combined to form a number, 0 through 31. That number, the color code, is used to look up a value in the colormap. The colormap usually contains a 12 bit number... 4 bits for red, 4 bits for green and 4 bits for blue, a total of 4096 possible colors including black and white. But there's another way of doing it. That is, for each pixel on the screen there's a unique memory location. In the AT&T boards we have a 16 bit word reserved in memory for each pixel; that works out to 1 overlay bit which controls keying and 15 bits for color... 5 bits for red, 5 bits for green and 5 bits for blue... so any pixel on the screen can be live video or one of 32000 colors. Given enough locations on the

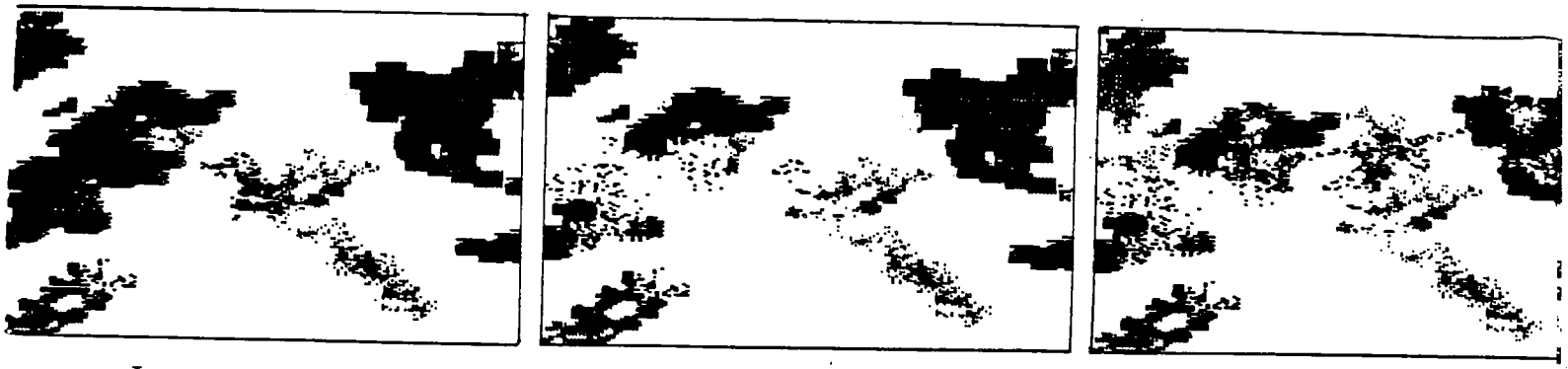



screen, you could display all 32000 colors at once. It's an easy scheme to work with; having programmed on my NNGS board, I know that a bitplane system can get incredibly messy. In order to pick out the value of a pixel when doing a logical transfer, for example, one has to access four seperate bit planes on the NNNGS board. You have to fetch four 16 bit words from memory, mask out the bits you're after, and put them together in order to find the color code. It's a time waster; on the AT&T board you can go directly to a single memory location and that tells you exactly what's there. If a modification needs to be made it can be performed directly on that word. It's a much faster and more direct way of dealing with a raster.

It's time to consider sound primatives. For the Moog synthesizer and the Serge Modular Music System the sound primatives were hardware modules. To talk about sound primatives one should talk about the modules themselves and about waveforms; sine, triangle, sawtooth and square or pulse are the common waveforms that appear on most oscillators. Some oscillators provide a means of mixing between the various waveforms; some provide a way to modify the width or symmetry of the waveform. Another type of waveform, commonly used for control, is the ramp; up then down, attack and decay. A more complex version is the Attack Sustain Delay Release, ASDR, or envelope generator in which the attack and release are set, a sustain level is set, and the attack and release are triggered. This waveform can control the amplitude of a sound or the sweep of filter. There are two basic ways to create sound; additive versus subtractive synthesis. In additive synthesis the sound is built up from simple waveforms such as sine, triangle and pulse waves's. In subtractive synthesis it is created by taking away parts of a complex waveform like white noise or its cousin pink noise, as in grapefruit. White noise is composed of random frequencies equally distributed over the audible range; pink noise is balanced to fit the human ear. Besides the sound generating modules or primatives we find processing primatives; filters and voltage controlled amplifiers, VCAs. Input to a processing module might come from an oscillator and the control from something like an ASDR or envelope generator. The trigger for the envelope generator and the pitch control for the voltage controlled oscillator, VCO, could come from a keyboard or sequencer. The processor, say a VCA, then creates a sound which can be made to approximate real instruments like a piano or violin, or it can be completely electronic. The Serge is especially good a that, generating sounds that are very unlike acoustic instruments. For a long time, Serge didn't offer a keyboard. He felt that if you needed a keyboard you should play the piano, why bother with a synthesizer. Other processing modules include waveshapers, which add complexity to raw waveforms, wave doublers, delay units both analog and digital, filters of various sorts, high pass, low pass, bandpass and notch filters (filter resonance can be controlled), resonant filters and resonant mixers. There are phasers and ring modulators. There are sequencers which play back a sequence of voltages accompanied by a trigger pulse at each step. On the Serge the sequencers were 4, 8 and 16 steps, with as many as 4 seperate control voltages at each step. These devices could cycle forward or backward, the duration of the steps could be varied. As it turns out, the sequencer function sounds like it would be an ideal thing to digitize and, in fact, when the computer was interfaced to the synthesizer through A/D convertors, it was to take over the for the sequencer. Large numbers of steps could be generated, a 1000 or more steps, and if you had a 16 channel D/A convertor you could generate 16 seperate voltages at each step.

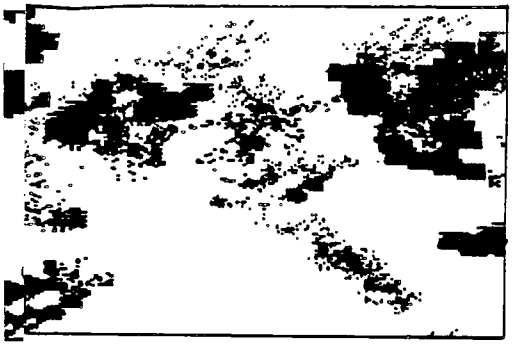
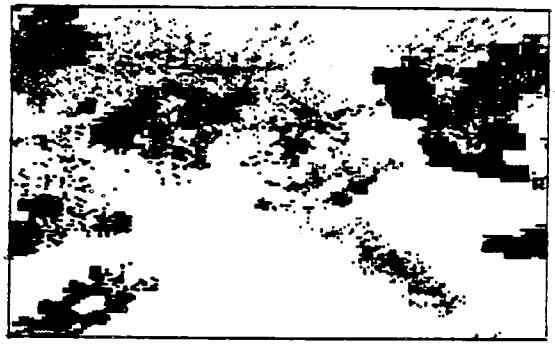
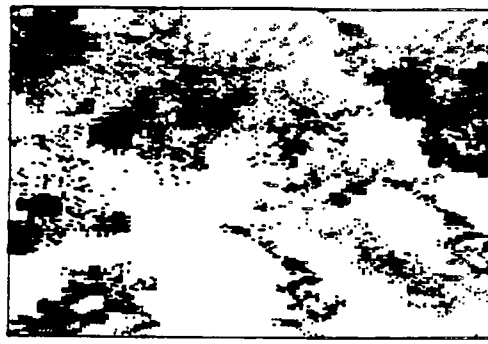






In summary, there are waveforms produced by modules like the voltage controlled oscillator, VCO, and the noise generator. We have the envelope, ramp and slew generators, the filters, the keyboard controller, voltage controlled amplifiers, VCAs, and in some cases voltage controlled mixers. The Serge has a variety of mixers with voltage controlled level and pan, even quad mixers which rotate the sound. Concepts relating to sound primatives include additive and subtractive sythesis; building up a sound from its raw or simplest components or filtering out portions of a complex waveform to create a sound. Percussive sounds can be produced by filtering down white noise whereas a flute sound is easier to build up additively.

Recent developments have clouded this straightforward system of modules. We have digital oscillators with the waveform represented as a sequence of numbers in memory. You could use any sequence of numbers as an oscillator... therefore you have the Mirage and other sampling keyboards. These keyboards combine digitized sound sources with some of the traditional analog processing modules, envelope generators, filters, and mixers. But the oscillator or sound source itself is digital. Its a very powerful system. Another idea which is very important to electronic sound is MIDI. SCAN always has a number of workshops on MIDI. It moves control parameters from one instrument to another to allow one keyboard to play a number of other modules, to add to the number of voices, etc. Once those signals are made available and once the keyboard is enabled to read those signals, the signals from the keyboard can go directly to a computer for processing and storage; they could be exhibited as a score. Conversely the computer can generate MIDI information for the keyboard, old information or new information that could do things that couldn't be done manually from a keyboard. MIDI is a very powerful idea, a generative idea in the area of electronic sound; it could be used to control events, musical as in a score, or one could invent new forms of composition based on sounds that don't exist in traditional music.

Briefly, let's consider the DX7. It is not a modular instrument like the Moog, it is performance oriented and uses some advanced techniques. The controls are digital, the sounds are analog. Recently, in my Mirage newsletter I note that someone is developing programs on the Apple which model the process of additive synthesis used at Columbia Princeton and the technique of FM synthesis used by the DX7. The sounds are produced on the Apple as data which is downloaded to the Mirage. Comparing the Moog, the Buchla and the Serge to the newer keyboard synthesizers; the limited interconnection of modules isn't as flexible as that which can occur on a modular system. I suppose that every Serge assembled is different and is tailored to each artists quirks. Any number of modules connected together in a fixed system gives a single possibility whereas if a dozen modules can be patched together in a flexible system you have many more possibilities. A patch panel system allows you to compose by recording your patches, composing by fixing the process by which the sound was created rather than working with individual sounds or notes in a more traditional way. In electronic art the emphasis in composition should be on process. One should look for new methods of composition appropriate to the medium rather than just "porting over" the traditional methods of composition. That applies in the area of electronic sound and also in the area of electronic images and computer graphics. In the area of computer graphics, why make the computer duplicate a photograph? It's not art. Artists must find those kinds of images and processes that are appropriate to the medium. Electronic art is about process, investigation and experimentation.

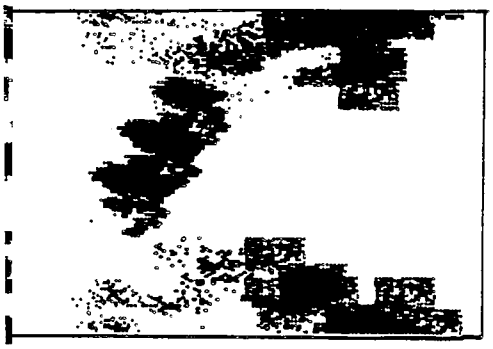
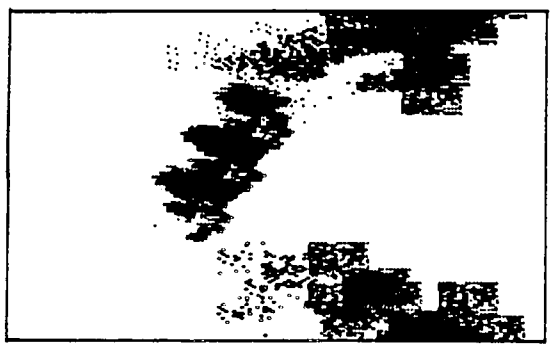




Now for the present, 1987. David Cook has been working as a consultant for EPICenter, programming on the ICB, Targa and Vista boards. He's also programming workstations for the fashion industry. He's making significant progress in the area of raster graphics. He's discovered a couple of new books: Procedural Elements for Computer Graphics by David F Rogers that gives you all the techniques for raster graphics; sorting out shapes, 3-d rendering , etc. These things are becoming more and more important. Last year's buzzword was desktop publishing; this year's is desktop video. Video is raster graphics. Desktop video is happening on the AMIGA. A number of interesting programs are emerging which integrate the production of images, animation and sound. These programs enable compose images and animation sequences with a soundtrack to accompany them. It's not a new idea but it's becoming more of a reality.

CONCLUSIONS:

Its time to look for gas, I'm somewhere around Iowa City and I presume that if they sell cold beer on the interstate at 3 AM they sell gas.

These primatives and ideas are all from own experience, these are my tools, things I've found useful. I keep modifying them, trading them in as appropriate. As the hardware changes the tools change; keying went from video hardware to software. You can fashion tools to fit your method of working. You can take any one of these elementary tools, focus on it and develop it for your own use. Last year one of the graduate students, Mitchell Lee, took the idea of pattern and tried to write a simple Applesoft BASIC program to produce as large a variety of patterns as possible. He learned a lot about pattern, how to use random number generators, what patterns remained or became interesting and what patterns became boring after awhile. I decided experiment with pattern. I experimented with variations in color. I used a repetitive series of locations, changed the color and repeated the pattern. I'd never thought of how to control the random number generator until I started to experiment with pattern. And that led to a series of "painting" programs. I took what I'd learned from the pattern generators and tried to generate a series of paintings by defining a random brush of varying density. The series of locations repeated in the pattern generator programs became the brushstroke. The painting process was metaprogrammed so that brushes of different sizes were chosen. A large brush made fewer brushstrokes and the brushstrokes contained a lesser number of points than a finer brush which made more brushstrokes and used a longer series of points. I had to repeat short brushstrokes a larger number of times in order to fill up the space. That's as complicated as the paint programs got. I let them run and watched


them develop over time. I haven't bothered to go in and modify the random number generators with probabilities over time but that will be the next stage. And I'd like to set the probability of certain colors appearing at certain places on the screen or, perhaps, certain densities in certain areas of the screen. You can create your own tools. Those that suit your style or your methods of working. If you're from a programming background then they should fit your style of programming.

Artists who use computers should program and, not only should they program, they should be able to read other artists' code from the assembler level right up through higher level languages. I think that artists will develop their own styles of coding, just as they develop their own styles of imagery. All this complicates things for those people who want to identify a work of art. There are a number of products that are the result of the process of creating electronic art; the final product which may be a print, an animation on videotape, an audiotape, a slide, or printout. But there's also the program that was used to create the product, that's as much a work of art as the final result. In fact, one program can produce many "works of art". As we move towards realtime digital and analog systems we find things like the Fairlight Computer Video Instrument and the Mirage that are realtime performance instruments and at the same time are programmable. Is a programmed performance a work of art?

I mentioned the AMIGA and this brings up an important point for discussion. It's the idea of SMALL computers in the arts that's important; equipment small enough and inexpensive enough that an individual can have it in his/her studio. I give this conference credit for sticking with that idea. It has resisted and, hopefully, will continue to resist the usual consumer penchant to assume that bigger is better, things must grow, to solve bigger problems throw more money at them.

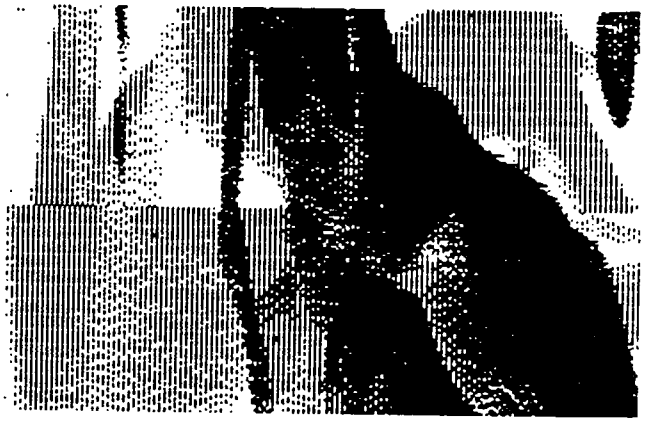
We're stopped dead on my way to Springfeild OH. I thought of two more books; Suzanne Langer's Philosophy In A New Key, and Godel Escher Bach by Douglas Hofstader.

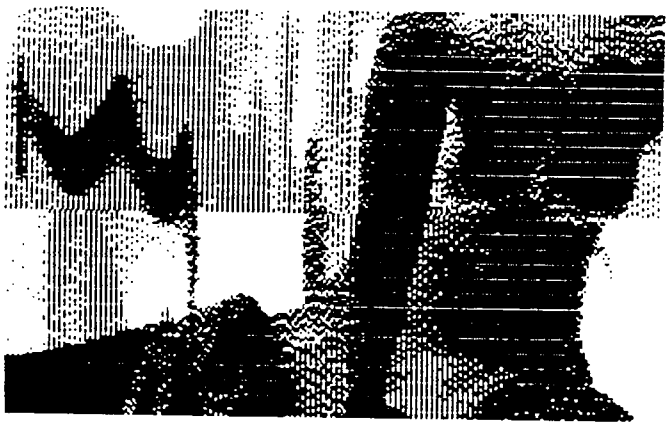
I mentioned modularity, the Moog had a series of modules that could be patched together and offered a variety of interrelationships that would be unavailable in a fixed or static system. I didn't talk about "systems" but it's been implicit in our discussions. There's a modular music system, a video system, a computer system, analog systems, digital systems, etc. A system is a series of parts more related to each other than to anything else, and a part of the definition of a system is the boundary. It can't exist without internal parts or without a clear boundary. The boundary is often a definition; like in an analog system the parts share the idea of a continuously varying voltage as representing information, or in a digital system all the parts use binary numbers. I had a marine biologist friend who used to say that if you wanted to find the really interesting things you would look at the boundaries between systems. He was always poking along the shoreline, in tidal flats, in places where the water met the forest. In architecture school I was told to pay special attention to how and where things joined together; another example of looking at the boundaries. Steina Vasulka said that in her videotapes the things she left out were just as important, if not more important, than the things she chose to include. And in music the silence between notes and between phrases is just as important as the note or phrase. At the interface between two systems one finds a very productive area for artists to explore. I'm interested in the interface between analog and digtal systems. That interface could be the sound created by a computer controlling an analog synthesizer. The interface between a preprogrammed system such as a sequencer and manual system such as keyboard results in a whole variety of interesting and expressive

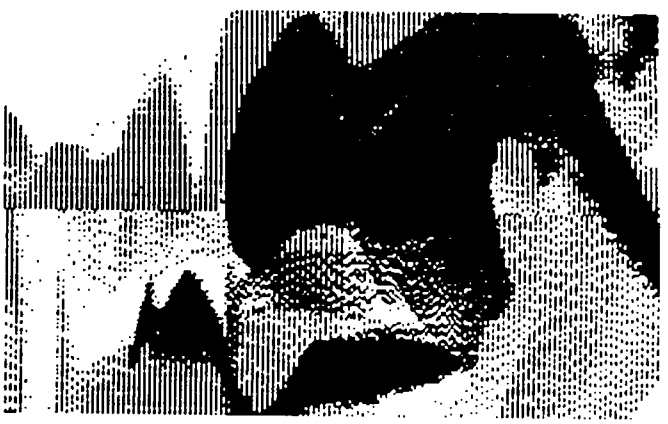
effects. You can take advantage of the tension between the exact regularity of the sequence and the lack of regularity in the manual system. The same thing can exist in an image; you can combine a digitized image with a hand drawn overlay. Or if you extend this idea a little bit further, computer output combined with more traditional techniques; a computer printout with a drawn image on a single print. Used in a traditional process, the feel of the computer can be used to provide tension and contrast to the more traditional forms.

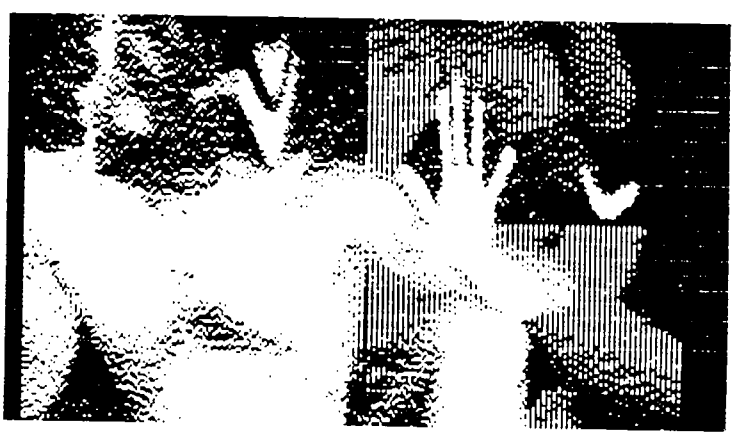
What's the difference between amateurs and professionals in the field of electronic art? Alot of people I know remain amateurs and refuse to participate in the business of art. I don't think they would reject some of the advantages of being professional like recognition, money and all that. So why do the remain amateurs? Because a professional requires big bucks, expensive machines, money for computer time, money for display devices, money for a few minutes of 35mm film. They have chosen to remain amateur for a reason.

I'm glad that SCAN supports amateurs. A number of electronic artists teach. It gives them enough money for and the time to experiment and explore. Gene Youngblood says in Expanded Cinema that the role of the artist is not to create a product but to create structure, to create new processes. I think that's a given in the electronic arts, all the artists' energy goes into the design of the system. The creating of structure, the development of notation, reapplying ideas in new ways... the results are often preliminary but this doesn't make the work of less value. The work must be looked at in a different context. The emphasis should be on experimentation. A word of caution when dealing with experiments. One can expect a large number of experiments to go awry, at least in terms of the desired result. Most experiments are destined to fail. Not doomed to failure, rather, it is an advantage that they fail. When dealing with new machines and new possibilities that have never been plugged together before, one isn't seeking verification but rather definition of the territory, as Woody Vasulka used to say. Artists are mapmakers. There will be false starts, but in the process one make's fairly accurate maps of the territory; maps that can be used later to enhance and fine tune the product. At this point in time electronic arts is so large a territory that artists must spend their effort on exploration and experimentation.









ORPHEUS IN THE UNDERWORLD: A CLASSICAL MUSICIAN ENCOUNTERS THE NEW TECHNOLOGY

DAVID A. LAWRENCE

835 Sycamore Drive, Palo Alto, CA 94303

One classical musician gives an account of his journey into the world of computers, synthesizers and sampling keyboards, and what he comes to like and dislike about it once he gets there. This paper is intended as a beginner's guide for others who haven't yet ventured into this territory, as well as a source of insight for the computer-initiated into the peculiarities of how classical musicians think.

Introduction

Given the absolute primacy of microelectronic technology in the world of pop music (my beloved Steinway would now be referred to as an "acoustic piano" in these dark realms, as it is no longer a mainstream instrument), one would suppose that the conservatories and colleges entrusted with the task of passing on musical culture to the next generation would have taken some notice. But there is little evidence that this is so. Most music departments have an electronic studio of some sort, but these seem to exist primarily to allow composers to play games with new sounds. And in any case, one mustn't assume naively that the very existence of such studios assures that some amount of idea content will leak through the walls to the counterpoint class next door.

The usual explanation advanced for this revolves around the generally hidebound character of academics, as well as a particular reluctance to embrace computer technology, which is perceived as dense and difficult. But I am not convinced that this image--the horse-and-buggy classical musician in a motorcar world--is really the crux of the problem. For one thing, computer phobia is overrated; it is easier to learn basic tasks like word processing on a computer than it is to learn touch typing. For another, there are some legitimate reasons why violinists and oboists may choose to revile the microchip, which I shall try to articulate here.

It may be quite possible to synthesize a note that sounds like a flute (more on this later), but this is a radically different musical experience from playing the flute. It would be hard to overestimate the significance of the tangibility of tone production in the history of classical music-making. The love of the string player for the particular qualities of his instrument, the obsessive quest of the wind player for the perfect reed--these images are at the very heart of musicianship in the classical tradition. This is the endless struggle: take more bow on the C#--no, too sweet, less vibrato--decrescendo to the D--glide up--make it breathe--make it sing! If one removes the musician from physical contact with the vibrating membranes and air columns that produce the tone, if one replaces this process with keystrokes and electronic circuitry, then there cannot be quality musicianship.

The other important reason for the disdain of the classical musician is that synthesized sound is perceived to be intrinsically ugly. Now, that sounds like a strong statement, but consider the following: all of the traditional orchestral instruments produce a tone that is round and full yet extraordinarily clear, even the low brass. Synthesized sounds, by contrast, particularly from the analog instruments that have been around for a long time, are very fat and fuzzy. This is something of an oversimplification, but not much. I can see no other reason for instance, why the saxophone has been denied admittance into the orchestral family (except for the very occasional and discreet guest appearance) all these years--the sound is too flabby--it doesn't have that quintessential classical clarity. It's important to realize, though, that in addition to it being an acoustic prejudice, this is only a partial argument: synthesized sound is a tiny fraction of what the microchip can do. Nonetheless it is an argument with an understandable historic origin, in the home electronic organ, early synthesizers, and fuzzed-out electric guitars: if it plugs into the wall it's going to produce an ugly tone.

Indeed I share portions of these prejudices. In the past two years, however, I have become convinced that microelectronic technology is going to have a revolutionary impact on classical music-making. Why I came to this conclusion, and how I survived the perilous journey in search of information from this other world, are the topics of this essay.

Can my sequencer win the Tchaikovsky Competition?

Every kid who ever wanted to play a Chopin Etude faster than his fingers would allow has no doubt tried tape recording himself at a slow speed and playing back the result at a faster one. It doesn't work, of course, since as the tempo is doubled, so is all the other information, such as the

frequencies of the tones. Well, sequencers work like tape recorders, except that instead of recording analog data, they record ones and zeroes. In this digitized form, then, any variable (such as tempo) can be isolated and altered to whatever specifications one desires. I would imagine that this is a somewhat more startling development in the classical than the popular sphere, in the light of how extraordinarily central the virtuoso performance tradition is to the history of all the classical instruments.

This actually means that velocity, the most basic of all the technical skills in performance, is, in theory, irrelevant now, at least in the making of studio recordings. Of course it isn't quite that simple, but it is nonetheless an absolutely stunning technology. One of the first things I did after obtaining my sequencing software was to enter Scriabin's Opus 42 #5 into my computer at a lento tempo. The result, speeded up, was as good as most recorded performances, and better than some.

So we seem to have an ethical and musical dilemma of the first order here. Of course the problem is not entirely new. There has always been a certain amount of career-saving in the recording business through tape-editing tricks. One can substitute a high C by an apprentice for the note the aging soprano can no longer reach. But more than the barest minimum of this and you are hip-deep in one-inch long pieces of tape, and on the road to chaos. With sequencers there are no such limitations; anyone who can play at all can play at any tempo. Accuracy is not a problem either. Did you miss a few notes? Just go back afterwards in edit mode and change them.

Does this mean, then, that as soon as beginner pianists worldwide understand the implications of sequencing technology, we'll be awash in faux Appassionatas? I don't think so. For one thing, the cheap-thrill phase passes quickly (I notice I haven't sequenced any other Scriabin Etudes). For another, the performance, in the end, is going to be judged by the same standard as any other performance--does it make a valuable interpretive contribution. It seems safe to argue that it will be at least as difficult to have mature insight into a Beethoven sonata at a slow tempo as at allegro molto. And if, in fact, it comes to pass that some interpretive genius can sequence better than he can play, are we not culturally richer for this fact?

The crisis, of course, comes from the way in which the centrality of the virtuoso tradition has been celebrated in the culture. Technical exercises on an instrument are like the prayers of the novice; one must serve time, and one must suffer, before the status of membership in the order is conferred. This is the largely unstated but absolutely essential truth of the virtuoso tradition. All of the instruments are extremely difficult to play, the piano perhaps least of all, and this makes the virtuoso performer, with his Olympian technical skills, a certified martyr to the demands of high art. When this concept begins to erode, then, cultural conflict will no doubt ensue.

The miracle of keyboard access

As a composer I am occasionally asked by non-musicians whether microtechnology helps me create music. I have learned that this question comes from a mental image of a machine so finely attuned to the nuances of my style that it could complete an entire sonata from the opening phrase. Lacking that skill as it does, the microchip nonetheless possesses other skills that are equally miraculous. Most significantly, it can create instrumental sounds or timbres (called "patches") either synthetically, or by sampling (recording and reprocessing) of existing sound sources. This technology, with its ambition of putting the entire palette of instrumental colors under the control of a keyboard mechanism, is conceptually an extension of the great cathedral organs, with the patches being an electronic incarnation of the organ stops. But the ability of some contemporary keyboards to replicate the sounds of the orchestral instruments has already surpassed, by far, the finest creations of the great organ builders.

What this means, in effect, is that I now have the capacity to play an oboe at the keyboard. And given how difficult it is to learn the proper embouchure to make even a single decent note on an oboe, I have some fears that when word really gets out about this, all those oboists who have been practicing for 20 years just to be able to play their instrument at all are going to buy guns and shoot themselves, or us. Keyboard access has already caused absolute havoc in the popular music business. The keyboardist as one-man band is now the norm in studio recording, and a lot of studio musicians are out of work.

On the other hand, this technology is going to be terrific for people like me. Whether for the realization of performances, or simply the joy of playing other instruments, anyone who can play the piano can now pretty much play anything. And the effect on composers generally is going to be one of radical democratization of opportunities. Unless one is wealthy or famous, or unusually charming, performers are not lined up at one's front door waiting for a chance to play each new composition as it emerges. And in time, one learns to adjust compositional goals to a realistic expectation about existing performance potential. I know, for instance, that there's not much point in my writing a wind quintet, no matter what the muse says, if I only know two wind players.

Well this has all changed now. Not only is there more or less unlimited potential for instrumental performance via keyboard access, but, as has always been the case historically, the presence of opportunity stimulates the creative mind. Just as Brahms' friendship with the clarinetist Richard Mühlfeld brought forth the clarinet chamber music masterpieces of his later years, so too the purchase of a particularly evocative trombone patch (for a few dollars!) may move some contemporary keyboardist to write a new trombone sonata. Not to mention the fact that in terms of the mechanics of performances, keyboard patches have it all over

real people: they don't need to be cajoled or have their egos massaged, they don't have (wrongheaded) ideas of their own about interpretive issues, they aren't late for rehearsals, they don't miss cues, and they don't spill coffee on the rug.

But what, I hear you asking, about the joys of ensemble music making with other human beings? Well, no one is against that. In fact, the scenario I have just described is perhaps ideally understood as a rehearsal scenario. As stated, mechanical music making is never ultimately going to replace human performers, because, even from the audience's perspective, there is something absolutely palpable about the exhiliration of witnessing the struggle of man and instrument to create the sounds. On the other hand, most of us are in less than ideal circumstances. And the encouragement and self-reliance that keyboard access tends to foster will be very significant compositional aids.

Notation and printing

Many musicians, even professionals, are painfully ignorant about how music has traditionally been printed. The first thing one needs to know is that most scores, including all the wonderful ones we conservatory students pored over in our school days, were engraved by hand--there is no workable musical equivalent of a Linotype machine. The second important reality is that this industry pretty much died in Europe and the U.S. after World War Two. Many "new" editions now are photocopies of old engravings. What little legitimately new work is done comes mainly from Asia, where there is still some cheap skilled labor. This also explains why there are price stickers on top of other price stickers on the sheet music in your local store.

The musical typewriter, like the musical Linotype, is an idea whose time never came: musical notation, unlike the written word, is highly vertical as well as horizontal, and therefore any machine that works on a linear principle is ill-suited to the task and highly inefficient. And to top off the bad news, music cannot really be written by hand effectively. Performers reading music, particularly if they are sight-reading, rely on a peripheral, gestalt sense of the shapes of note groupings; one rarely has the time to identify each note individually. As a consequence, the performer's eye must be accomodated by a very rigorously geometric and exact formation and placement of the notes and symbols. Therefore, even a copyist with a very compulsive and steady hand is inevitably asked whether a certain note is a D or an E ("It's a little below the line; I can't tell"). And absolutely nobody has a musical hand that can duplicate the sturdy and lucid geometry of the old engravings. Inevitably, then, hand-drawn work bespeaks amateurism, and the consequent unfavorable comparison with professional printing.

Again the microchip comes to the rescue. Since the end product is machine-made, it has the desired exactness of appearance. Although I have never seen a computer-printed manuscript that's as beautiful as the best engravings, some of them are pretty good. Additionally, there is some hope of using the computer's vast processing power and speed to notate directly from performance. Ideally, one does not want to hand-enter every note and beam--that's no different from the inefficiency of the musical typewriter. A number of software systems claim that they can transfer files directly from performance to notation. And they do, sort of. The problem is that the machine is not going to compensate for any inexactness on the performer's part, since it's a literal beast. Metric inexactness is particularly exasperating in this respect, since there's so much of it in classical performance (if you need proof that your playing is not strictly metronomic you will get it when you transfer your files to a notation program). There are ways of getting around most of this difficulty, but there will probably be at least some hand editing in the music printing process by computer--there are a lot of variables.

Meanwhile, though, through the performance miracles of sequencing, keyboard access, and notation programs for printing music, a huge portion of the obstacles that have always kept creative musicians from realizing their ambitions in the past have been removed. In a way, it is hard to grasp how truly liberating this technology is going to be: a pianist composes a string quartet, plays all the parts himself, assembles them into a recording, makes a copy of the score without having to write it out--there must be some catch! Well, there are a few; read on.

Does it really sound like an oboe?

That depends. First of all, there is much more computer-related product out there than there is buying public, so advertisers, salespeople, magazine reviewers and that crowd tend to lie like crazy in order to move merchandise. Just as graphics on computer games have been oversold (I have a friend who calls them all "Attack of the Flying Semi-Colons" because of their tedious sameness), so too synthesized and sampled patches are praised out of all proportion to their real worth. Most patches are simply awful, and almost everyone, if pressed hard, will admit this. If a keyboard comes with a hundred factory patches, there might be five that are of any use.

The acoustic instrument replications are even harder to find, since the classical marketplace is perceived (correctly) as a relatively limited one. Additionally, they are harder to produce. For one thing, they are not single sounds. If you freeze a wave from a trumpet note on an oscilloscope, and then reproduce that wave at the desired frequency, you are not going to get the sound of a trumpet. Astonishingly, to early researchers, the middle of any number of instrumental notes (at the same frequency) sound rather alike--a trumpet like a violin like a flute. In order to get the characteristic trumpet sound, one needs the trumpet attack evolving into the trumpet decay--particularly the attack. Whether one builds this shape synthetically, or modifies a sample of it, it's a lot of work.

Secondly, if your aim is a specific replication, serendipity works against you. This is an

important point, as it sharply differentiates the classical musician's concerns from those of others. One of the most enduring new-technology studio preoccupations is tinkering with old patches to get new ones. By changing any parameter one achieves something new; if that something new isn't pleasing, no harm--simply change it again. This exercise, so dear to the hearts of synthesizer programmers, is the opposite of what you want to do. What is necessary, rather, is the most intense and highly-engineered kind of effort to identify the characteristics of every facet of the instrument in question, as well as a commitment to persevere as the sound closes in, ever so slowly, on that goal. This is, in a word, best left to professionals.

The scoresheet, as of this writing, is about what one might expect: the more complex and idiosyncratic the sound, the harder it is to reproduce. The percussion family has been replicated very well. So too the guitar and the harpsichord. Most of the brass sounds have been captured reasonably well. Some winds, particularly the flute, are very good; others are only fair. Most string patches are poor. Ordinarily the piano would be at the bottom of this list because of its complexity--waves are bouncing all over the place inside a grand piano. Because of its centrality in the music-making culture, however, several engineers have made it a special priority, and as a result there are a couple of rather good grand pianos.

Be warned, however, that I am talking only about the very top of the product market, and the bottom is absolutely immense. There are entire lines of keyboards whose instrument-replication claims are total falsehoods. Additionally, there is the deep problem of the inability to achieve variations in musical expressiveness from a single note, that I mentioned in the very beginning: even if one found the perfect violin tone, one wouldn't want to hear exactly that tone every time, in every musical situation. I have some fears that this may be an insoluble problem at this stage of the technology; after all, that's what bow arms are for--it's hard to imagine how that same degree of subtlety could ever be achieved by depressing a note on a keyboard, no matter how configured.

Lastly, there is a real danger that this entire discussion will continue to be held from the point of view of the listener only. There have been some famous instances of tests of acoustic-instrument substitutes like the Kurzweil 250 piano, to see if audiences could be fooled (they could, sometimes). This misses an important point, though: the future of the (acoustic) piano is not really up to audiences, it's up to pianists. What they want to play is what will endure. Yes, the Kurzweil is a rather magnificent hunk of engineering. But if the purpose is just to play some Chopin, not to process MIDI data or some such thing, then, with all due respect, any pianist who would prefer the Kurzweil to a quality acoustic instrument is a certifiable lunatic.

Will digital sampling really reproduce any sound?

Not really, although that's certainly what all the ads claim. Sampling, which is currently something of a fad, is limited by, among other things, the quality of the microphone and your skills as a recording and sample-editing engineer. The part about microphone technology is just common sense, but that doesn't mean that a lot of people haven't overlooked it. The idea of using a home machine to sample in the field, as in "I think I'll sample the great organs of Europe," is a complete joke. Even a total pro would need years of work and studio time to pursue that kind of project.

On the other hand, sampling technology, per se, is a very exciting development. By and large, more quality acoustic instrument replications have been developed using sampling, and the modification of samples, than through the synthesis of sound. One might want to consider building a library of sounds by buying samples, rather than making them.

Will the computer usher in a golden age of music?

It hasn't so far. First of all, this is very new, by the standards that music historians use. I am always amazed, and a bit horrified, to realize that a pop style that's a few years old is not just dated, it's sort of camp. The rhythms of change in that world are grotesquely foreshortened, as on some planet in a science-fiction novel. Even in modern times, by contrast, it took serialism about 60 years to wax and wane. So it would not surprise me if we are still dubious about this technology for another decade or two, nor would that necessarily be a mistake.

Secondly, however, there is always the problem of how technology shapes aesthetics, and that is a very deep issue. If one thing is absolutely crystal clear about these new gadgets, it's that they are wonderful at creating new timbres--that's what they do most naturally and easily. Accordingly, most of the serious musicians who are playing with them are compulsive timbre-makers. I have been in a number of classroom situations in which the entire aesthetic discussion consisted of timbrel considerations; not one word was spoken about melody, harmony, rhythm, counterpoint, or form.

In the pop world, there's this preoccupation with finding one's new sound also, without much regard to whether or not the music is any good. But that's not inconsistent with the way popular musics have always been conceived. I find it much more troubling in the classical realm, in that it seems dangerously ahistorical, among other things. The concern for timbre is, by and large, a 20th-century phenomenon, so it is a relatively recent development. Of course there were always composers who were better at clothing their work in effective colors than others, but the central aesthetic concept, from plainsong through Webern, has been the relationship of the notes. Bach's Art of the Fugue, perhaps the quintessential European piece, can be played by any number of instrumental combinations and its aesthetic content remains the same. So too can the great symphonies, at least through Brahms, be played at the piano without losing their significance.

In this light, then, the concern for timbre

alone seems rather like judging the quality of architecture by the color of the exterior paint--it's a factor, but not a very important one. It also puts these composers at odds with music history and aesthetics as they are currently being taught. The study of the relationship of the notes is still the conceptual basis of the music curriculum, and there is no reason to think that should be changed. Since it is always argued that composers lead the way into new aesthetics, with theoreticians and historians lagging behind, perhaps even the ahistoricism of the timbre-makers is amply precedented. But only up to a point. There is always the danger, however small, that classical composers, like their pop-music comrades, will actually forget where music is buried--just as every so often some pop composer will come up with a "new" chord progression that has actually been in common usage since Debussy. That's the kind of thing that makes academics snobby about popular music, and with some reason.

Will I be able to understand the new technology?

All of it? Absolutely not. But don't feel bad; most of the other musicians using it won't either. I was amazed, and somewhat relieved, to discover that even the technically astute are mystified by the behavior of their studios a fair portion of the time. In fact I've seen computer wizards stare at their screens with total incomprehension--that's normal, it seems. But more importantly, the proliferation of the gadgets and their complexity have created a technical maze that is so dense that it constitutes a major impediment to music-making. In this context, the need to keep goals clear and means simple is especially urgent. It is all very well to have a studio that looks like the inside of a spaceship, and I suppose it confers feelings of power and prestige for some, but let there be no mistake about the fact that it interferes with creativity. It is not at all necessary to have a cross-section of the available synthesizers--only those that enable you to do specifically what you want to do.

It is also a mistake to be greedy about acquiring computer hardware and software. Programs are not like novels, to be kept on the shelf for a rainy evening; with the possible exception of computer games, they are tools for doing work. Each one requires a period of apprenticeship before you master it, and the more complex the tool, the longer the apprenticeship. Therefore, the more tools you use, the more hours of apprenticeship there are and the fewer hours of creativity. You want to find one sequencer that you like, and one transcription program, and so on, and then learn to use them well. In the same way, it is not necessary to master all the applications of a given program, as if it were a territory to be explored. These are tools; they exist to serve you. This all seems self-evident to me, but a great many people in this field nonetheless own rooms full of hardware and software, most of it going untouched. I understand that the lure of the gleaming dials and the computational power is very strong, but one has to understand that these situations are minefields, waiting to drain your energy away from the music. You might, if all else fails, want to remember that all of the great composers in history, combined, didn't own a single MIDI cable.

That said, however, this is a wonderful time to be in music, perhaps the best there ever was. If we could manage to keep the thought in mind that the gadgets are enabling tools, that give us options, not aesthetic, cultural or personal imperatives, then the proliferation of resources would be entirely positive. We will probably never have this many options again, as the quantity of music hardware and software products cannot be explained by market factors alone. I assume that it has to do in part with the number of computer types who have musical avocations or interests. This quantity and quality of programs and machinery is even more inspiring in light of how many of the touted uses for the personal computer are actually silly and unnecessary (the "recipe file" syndrome). By contrast, there are some music applications that are not only useful, they're earth shaking.

PC-BASED COMPUTER ANALYSIS AND SYNTHESIS OF AN ACOUSTIC PIANO NOTE

K.S. Gugel* and R. Sudhakar**

* International Business Machines Corporation/Entry Systems Division
** Florida Atlantic University/Electrical and Computer Engineering
Boca Raton, Florida

ABSTRACT

An IBM Personal Computer AT system was used in conjunction with a professional data acquisition unit to digitize and store several cassette deck recorded piano notes. These digitized notes were then visually analyzed both with an AT monitor and a high resolution plotter. Fourier and Walsh Transformations were then performed on the digitized data to yield further information. Upon completion of this study, several types of data reduction and waveform synthesis methods were formulated. The actual test performed on the experimental synthesis methods consisted of recreating the piano note and then subjectively comparing the audio performance of the synthetic note versus the original note.

INTRODUCTION

Computer-controlled digital music synthesis will profoundly affect popular music in the coming decades. The number of possible control and sound generating structures in digital synthesis is enormous. Advances in solid state circuits is another motivating factor for real time synthesizers. With currently available signal processing chips, it is feasible to design inexpensive synthesizers of sufficient tonal quality and richness which can be down loaded and controlled by home computers.

A piano was selected for the study and synthesis because it is a multi-timbre instrument and is considered to be among the most difficult musical note to synthesize [1]. Thus, if an adequate method could be derived to synthesize a piano note, the method should also show the capability to synthesize other musical instrument notes. The IBM PC AT was chosen as the base processing system because of its low cost and availibility. Several high performance commercial music synthesizers such as the New England Digital Synclavier [2] and Fairlight CMI III [3] are very similar in memory size and processor power to the IBM PC AT and IBM Family II personnel computers. However, these music synthesizer systems cost in excess of 60,000 dollars [4]. Our goal is to create a PC based digital music synthesizer with similar capabilities as the high performance machines, but at a fraction of the cost. The synthesizer can be realized with a digital signal processing microprocessor and 128 kbytes of random access memory.

Some of the popular systems to date use techniques such as subtractive synthesis, Digital phase distortion synthesis, Digital FM synthesis and Additive Synthesis [1]. We chose to experiment with the Fourier synthesis and direct waveform synthesis. Fourier synthesis in principle is similar to additive synthesis and exploits the excellent analysis capability of Fourier transform. Direct waveform synthesis with data reduction was selected because of its ability to accurately synthesize a musical note.

DATA ACQUISITION AND HARDWARE CONSIDERATIONS

The piano sampled was a 1972 Baldwin Acrosonic Spinet. A TEAC V-800X high fidelity cassette recorder was coupled with a Shure PE-588 studio quality microphone to record the initial piano notes. The recorded cassette tape was then played back into a Data Translation DT2821-F-8DI data acquisition unit. This unit resided inside the AT computer. The system also contained three megabytes of Dynamic RAM which was allocated as RAM Disk for the acquisition unit. Upon completion of the acquisition, the acquired data was dumped to diskette or hardfile storage. All acquisition was performed at either 50 or 65 kHz. The sampling resolution was 12 bits; however, because of the Disk Operating System file format, two bytes of file storage was exhausted per sample.

The maximum time span for a recorded note was approximately 11 seconds. Eleven seconds of sampling at 50 kHz (2 bytes per sample) created a 1.1 mega byte file. This large file barely fit on the system's 1.2 MB high density diskette.

Results were displayed on the system's Princeton Graphics high resolution color monitor and were also plotted on an IBM 7385 high resolution/high speed plotter. Audio results were obtained by converting the processed files back to the analog domain with the DT2821's Digital/Analog Subsystem. The D/A conversion information was then recorded onto cassette tape for future listening.

ANALYSIS OF THE PIANO NOTE

Although various other piano notes were experimented on, we are considering only Middle C (261.63 Hz) in this paper. Middle C was digitized under two different dynamic conditions. These conditions were termed the "short note" and "long note". The short note or staccato was created by striking and releasing the key as quickly as possible. The long note on the other hand was created by striking the key and holding it down until the sound could no longer be heard.

Figures 1 and 2 provide the time plots of the short and long notes of Middle C. Similar plots were made for other notes also (not shown). From the plots, one can observe that the low notes contain a large amount of high frequency content. Similarly, the upper notes exhibit low frequency amplitude modulation. These observations can be attributed to the fact the struck note causes resonances in other harmonically related notes.

In the inset of Figure 1, waveform peaks can be observed shifting slowly up and down. The waveshape is continuously changing from one period to the next. It is also noticed that the waveforms differ for the same note for different octaves. In the case of the long note (Figure 2), the envelope of the waveform appears to decay in a linear fashion. This gradual envelope decay has been exploited by several synthesizer designers such as the Kurzweil 150 and 250 synthesizers [5]. Such synthesizers approximate the original signal with a base waveform which consists of a large number of waveform segments. The envelope of the segments can be altered by the user for special effects or new sounds.

Frequency/Sequency Analysis

The next analysis procedure consisted consisted of observing the frequency domain characteristics of the notes. To this end, Fast Fourier Transforms (FFT) and Fast Walsh Transform (FWT) were performed on the digitized notes. While the Fourier Transform was calculated to generate harmonic magnitude and phase information, the Walsh Transform was calculated to determine its desirability for waveform synthesis. Because the Walsh Transform is based on square wave sequences, it is inherently a much faster transform than the Fourier.

Transform lengths such as 256, 512, and 1024 were implemented in the FFT/FWT study. Because the note is much longer than any of these lengths, it was subdivided into blocks (segments) having the same length as the transform length. Thus, a note containing several hundred thousand samples would be divided into contiguous, non-overlapping segments of length 256, 512 or 1024. Here a 1024-point FFT was run on the segments of the long note; the magnitudes of the first 128 harmonic numbers are plotted in Figure 3, as a function of the segment number. Another plot was made for the short note (not shown). The plot for the short note is similar to that of the long note except that the number of segments is much smaller.

The long note file had approximately 550,000 samples or equivalently 537 segments each of 1024 samples. In figure 3 only the first 150 segments are shown. It was found that the remaining 387 segments also yielded harmonic magnitudes which vary smoothly and linearly enough to permit accurate extrapolation from the first 150 segments. The magnitudes decay rapidly during the first second (50 segments) to a small fraction; thereafter the magnitudes decay slowly. Another observation is that most of the excitation energy appears to be in the lower 64 harmonics. Thus the smooth decay of the magnitudes along time as well as along the harmonic number can be exploited in data reduction.

In addition, the unwrapped phase plots were also obtained for different transform lengths. However, unlike the magnitude plots, the phase plots did not reveal any smooth trends. This inconsistent behavior of the phase is due to the fact that the segments contain partial periods of the waveform. Smooth behavior can be expected if a segment is made to contain full periods, but this leads to the difficulty of identifying full periods and then computing Fourier values at correct frequencies using non-FFT methods. In summary, phase does not lend itself very easily to data reduction techniques.

Results of the FWT were not encouraging. Though the sequency magnitudes decay with time, there is no definite trend along the sequency numbers. FWT was not used for any subsequent synthesis methods.

SYNTHESIS METHODS

In the following sections two of the data reduction resynthesis techniques attempted, viz., Fourier Synthesis and Direct Waveform Synthesis are presented:

Fourier Synthesis with Magnitude Approximation

One popular method of data reduction is to approximate the harmonic magnitude function along the harmonic numbers or along segment numbers (Figure 3); the approximated functions are then inverse transformed to obtain the synthesized note [6,7]. Here we consider two methods of approximation of the magnitude function along the harmonic number. They are labeled as (i) Harmonic magnitude truncation and (ii) Harmonic magnitude linear interpolation.

Magnitude Truncation: It consists of setting the small magnitude values of higher harmonics to zero. This also automatically eliminates their phase values. A 1024-point FFT was executed on the Middle C short note and of the unique 512 harmonic values (due to symmetry), only the first 64 harmonic magnitude and phase values were preserved. All others were set to zero for each segment. This corresponds to roughly an 8:1 data reduction. The inverse FFT was then performed on the reduced file and played back. The truncation produced a high frequency glitch which was audible as a clicking sound, as a result of the waveform mismatch at the segment boundaries (see Figure 4).

Smoothing was then tried at the segment boundaries (of the synthesized waveform) to quiet the clicking sound. Two methods of smoothing were tried. In the first method, a small set of samples (say 6 samples) on each side of the segment boundary was replaced with the interpolated values obtained from the neighboring samples of the set. Linear as well as cubic polynomial interpolation of the samples were tried. However, the clicking sound persisted, even though the boundaries after interpolation appeared smooth and continuous as shown in Figure 5. The smoothing method was repeated for cases where 128 and 256 harmonic magnitudes were preserved (data reductions of 4:1 and 2:1); it could only weaken the click sound.

Another method of smoothing was to replace and patch in a small set of samples on each side of the segment boundary with original samples (which were saved for this purpose). On experimentation, it became apparent that a large number of replacement points (such as 128 samples) were needed to eliminate the click, thus severely minimizing the data reduction.

Magnitude Linear Interpolation: In the linear interpolation of magnitudes, first 64 and 128 magnitude values were preserved and the remaining values upto the 512-th value were calculated from a straight line approximation. However, the reduction produced a poor note which sounded like noise superimposed on the original. When the preserved values were increased to 256, the reproduction was indistinguishable from the original, but the data reduction was insignificant.

The experiments were then repeated for approximating the magnitude values, as a function of segment numbers. Again the segment boundary mismatch problem was encountered when the data reduction was significant. Thus, the Fourier synthesis method appears inadequate in providing the required data reduction for the piano note, though it has been shown to be successful in synthesizing less complicated waveforms [8].

Direct Waveform Synthesis with Data Reduction

The approach here was to save the first second of the original long note file as it is, (because of the complexity of the waveform during attack and sustain) and to apply data reduction only to the remaining 10 seconds of the waveform ("tail"). If an adequate synthesis method could be devised to reduce the tail to a few kilo bytes, the 1.1 Mbyte long note could be compressed into 128 k bytes of memory. The synthesis techniques tried on the tail waveform are: i) Repetition with linear envelope ii) Differential template synthesis and iii) Differential waveform coding.

Repetition With Linear Envelope: In this method, the first full period of the tail waveform (say of P0 samples) was saved as the template, as shown in Figure 6. The number of repeated periods (say N) in the tail is estimated as total number of samples/P0. Now the synthesized note is produced by repeating the template with a gain factor linearly decreased from 1 (for the first period) to zero (for the N-th period). The note sounded similar to that of a sawtooth waveform and was too simplistic for acceptable reproduction.

In order to moderate the high tonal nature of the synthesized waveform, the number of samples in the repeated periods was made to vary randomly over a small range; this range was obtained from a visual inspection of the original waveform. The period variation was achieved through the use of PC AT's random number generator and another computer program for period normalization. The normalization program obtains the desired period (say P1) from the random number generator and recalculates the new sample values at 1/P1 spacing from the original P0-long template, by linear interpolation. For example, the value of the n-th new sample is obtained by linearly interpolating the m-th and (m+1)-th sample values of the template for a fraction f, here m and f are the integer and fractional part of nP0/P1. The period variation method did not yield acceptable results. The note sounded like a sawtooth with superimposed noise.

Further improvements were attempted by choosing a block of 8 consecutive original periods as the template instead of the single period; the template was repeated for N/8 cycles with decreasing gain factors. Again the synthesized note did not provide any appreciable improvement.

Differential Template Synthesis: Here the first period out of every block of 20 consecutive periods is chosen as a template. Twenty five such templates were extracted from the waveform tail. The aim was to obtain a 20:1 data reduction. With a sampling rate of 65 kHz the template periods varied between 244 to 258 samples. All the templates were then normalized to a period of 256 samples using the period normalization program. The number of samples in every 20 period block was kept the same for convenience. The sample values of the in-between 19 periods in each block are obtained by linearly interpolating from the two bounding templates. For this, (256 sample long) difference waveforms are created by obtaining the difference between the values of the corresponding samples in every pair of consecutive templates and dividing the difference by 20. Then the samples values of the first repeated period of the block are obtained by adding the difference waveform to the template; continuing in this manner, the r-th repetition period waveform within the block is obtained by adding the difference waveform to the (r-1) -th repetition period.

The synthesized note sounded very close to the original but contained a slow vibrato or frequency modulation. This vibrato may be due to the choice of constant number of samples in every 20-period block. Also because of the changing nature of the waveshape (in the original signal) from period to period, it is very difficult to locate the beginning samples of the templates (which are 20 periods apart) exactly. Here the number of samples in the 20-period block was solely based on the visual inspection of the first two blocks. The beginning samples of the periods can be determined more accurately by performing an autocorrelation of the waveform and locating the peaks. Since the periods are approximately known, the correlation need to be computed only for the lag values near

the period. Thus with improved period detection, differential template synthesis yields good results.

Differential Waveform Coding: This method was performed on the 50 kHz sampled data. Unlike the previous methods the entire waveform, except the first 1050 samples (of the attack region), was subjected to the data reduction. The data compression was achieved by reducing the number of bits required for representing each coded sample. To this end, first, an autocorrelation was performed on the entire note to determine the beginning samples of the several thousand periods. The periods ranged between 193 to 195 samples and were normalized to 194 samples using the normalization program. Next, the samples of the r-th difference waveform are calculated by taking the difference of the sample values of (r+1)-th period and r-th period. The normalized waveform and the difference waveform are shown in Figures 7 and 8. Then another data file (called DIFF2) is created from each difference waveform by calculating the difference between the consectutive sample values of the particular difference waveform. Only this new data file is saved. Since the difference between the samples are small, the DIFF2 samples can be represented with fewer number of bits as compared to the original waveform. This is illustrated in Fig. 8. The procedure is termed as Differential waveform coding [9].

Table 1 illustrates the memory savings anticipated for different stretches of the long note. Beyond the 250,000-th sample, the consecutive samples of differential waveform differs only by one bit. Though DIFF2 samples have varying bit-widths (from 8 to 1), they can be stored serially in the memory by placing special code words within the bit stream to indicate change of bit-width. The scheme can be easily implemented in hardware or software. The overall data reduction is about 6·2:1. Currently the method is being tested out for its subjective performance.

The DIFF2 samples can be made smaller by a power normalization procedure [10]. It involves determining the averaged power per sample for each period by summing the squared values of the samples within a period and dividing by the numbr of samples. The power normalized version of a particular period is then obtained by multiplying the sample values of the period by the square root of the power ratio of that period and its preceding period. The difference waveforms are determined from the power normalized periods; corresponding scale factors are also saved for synthesis.

Further data reduction could be attempted by approximating the DIFF2 samples in the 5 to 11 second range with smoother functions. Additive synthesis with sinusoids or linear predictive coding could be used to determine the smooth functions. A small number of coefficients may be sufficient to synthesize the smooth functions.

CONCLUSION

The piano note has a quasi periodic waveform which is complex and continuously changing. The complexity can be attributed to factors such as the mechanical nature of generation, hammer strike on multiple strings and resonances of other notes. Because of this inherent complexity, Fourier synthesis methods when attempted with significant data reduction, resulted in segment mismatch problems and had limited appicability. Differential Template synthesis and Differential waveform coding methods showed promising results. An optimum approach may be to go for a hybrid scheme. For example, the first 1 k samples of a long note may be stored directly, the next 100 k samples may be subjected to Differential Template Synthesis and the remaining 450 k samples may be reducted using Differential Waveform Coding. The data reduction can be increased form the present value of 6·2:1 by incorporating additive sine wave synthesis or linear predictive coding into the Differential Waveform Coding.

A spin-off from the analysis was the generation of neighboring notes by changing the sampling rate of the basic middle note. Thus three notes above or below the basic note can be reproduced, either through software calculation or through D/A conversion rate change. With this 7 note range (±3) for synthesis, an 88 note piano keyboard will require about 13 basic notes. Assuming a 10:1 data reduction of the hybrid synthesis scheme, the 13 notes representing the entire piano range can be stored in a single high density 1·44 Mbyte diskette. Thus an IBM PC hard disk will be able to store the compressed files of several such musical instruments.

REFERENCES

[1] Howard Massey, A Synthesist's Guide to Acoustic Instruments, New York, New York: Amsco Publications, 1987.

[2] Richard W. Schramble, "The Hills are Alive with the Sound of Music," IEEE Potentials, p. 12-17, May, 1984.

[3] D. Goldstein and Simon Trask, "Future Shock," Music Technology, p. 42-46, September, 1986.

[4] Tim Tully, "Choosing the Right Sampler," Electronic Musician, p. 26-34, December 1986.

[5] C. Morgan, "The Kurzweil 250 Digital Synthesizer," Byte, p 279-288, June 1986.

[6] J. Moorer, "Signal Processing Aspects of Computer Music: A Survey," Proceedings of the IEEE, Vol. 65, No. 8, p. 1108-1137, August, 1977.

[7] S.C. Bass and T.W. Goeddel, "Some Examples of Discrete Time Musical Sound Synthesis Using window Function Waveforms," 1983 IEEE International Symposium on Circuits and Systems, Vol. 3, p. 1087-1089, 1983.

[8] G. R. Charbonneau, "Timbre and the Effects of Three Types of Data Reduction," Computer Music Journal, Vol. 5, No. 2, p. 10-19, 1981.

[9] Hal Chamberlin, Musical Applications of Microprocessors, 2nd Edition, Indianapolis, IN: Howard W. Sams and Company, 1987.

[10] M. Le Brum, "Digital Waveshaping Synthesis," Journal of the Audio Engineering Society Vol. 27, No. 4, p. 250-266, April, 1979.

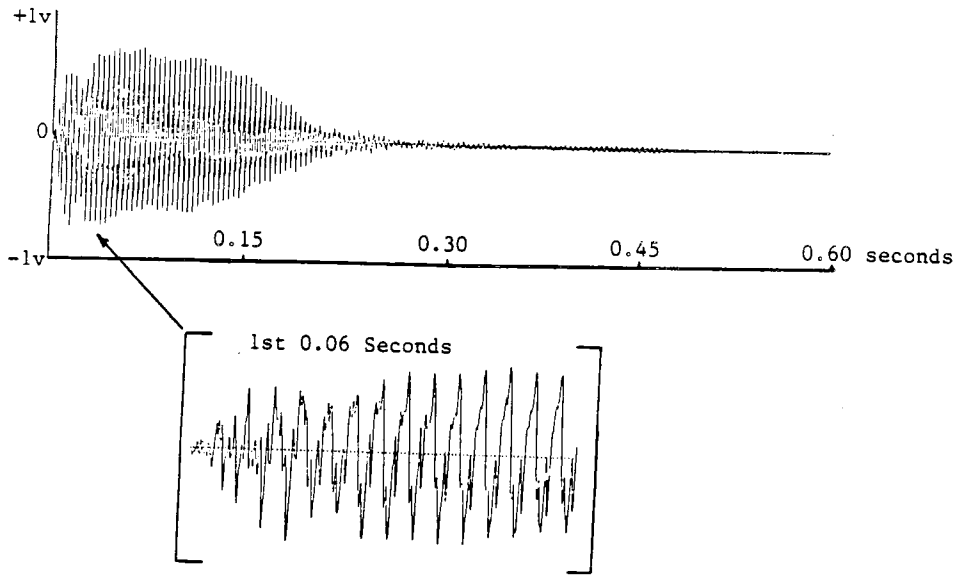


Figure 1. Middle C Short Note Envelope and Waveform Plot.

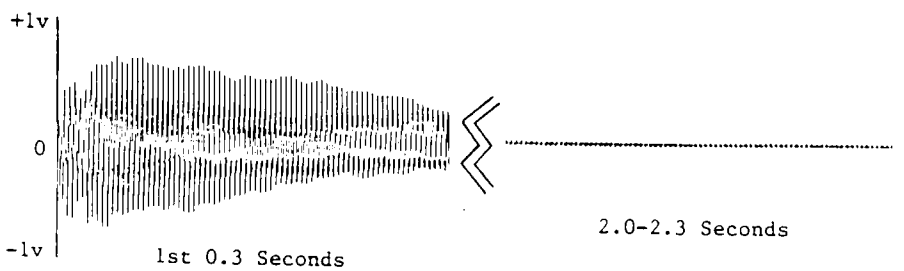


Figure 2. Middle C Long Note Envelope Plot.

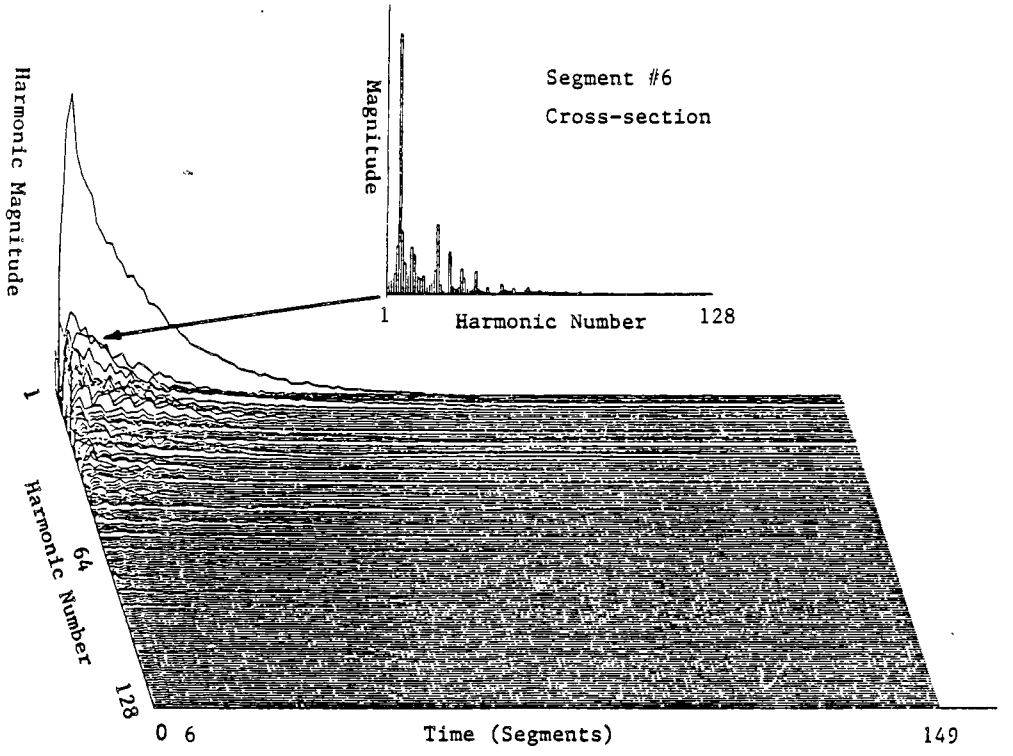


Figure 3. Fourier Magnitude as a Function of Harmonic Number and Time (Segment Number) for Middle C Long Note (1024 FFT).

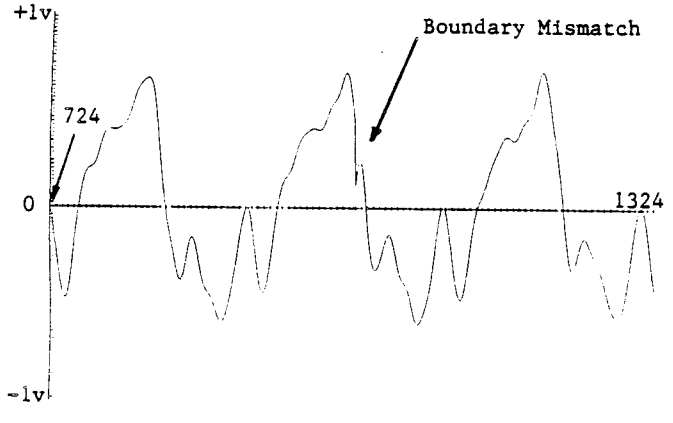


Figure 4. Effect of Magnitude Truncation on Segment Boundary.

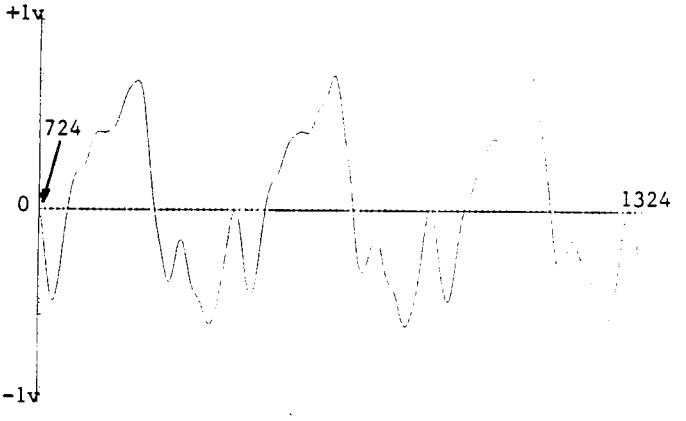


Figure 5. Result of Boundary Smoothing Using Polynomial Interpolation.

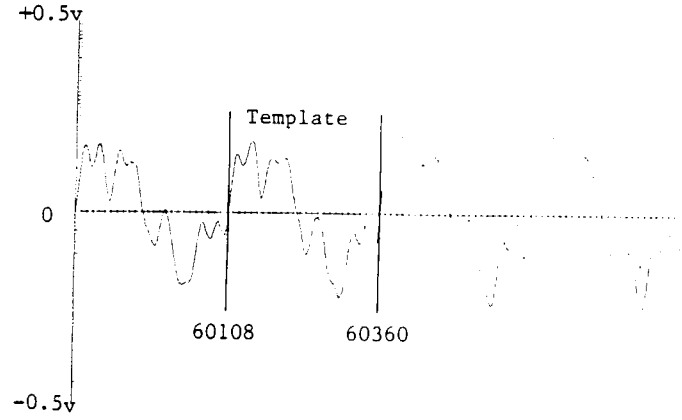


Figure 6. Template for Single Period Repetition.

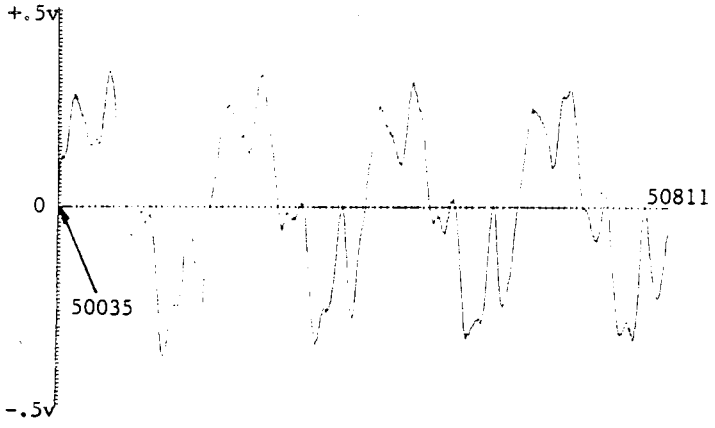


Figure 7. Four Periods of Frequency Normalized Waveform.

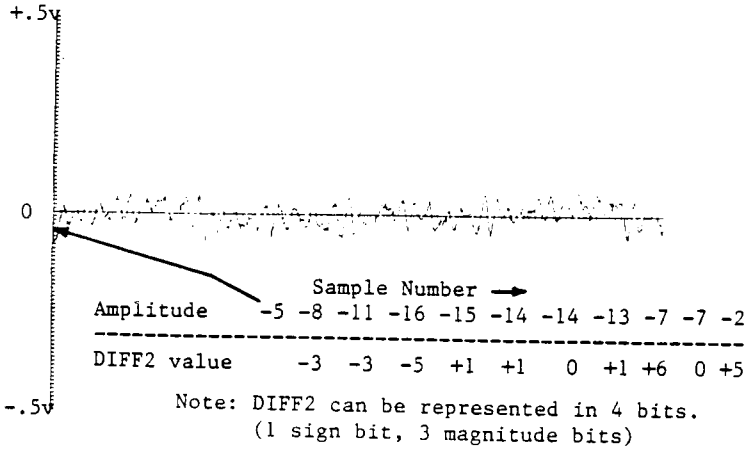


Figure 8. Difference Waveform Corresponding to Figure 7. First 11 values and their differences listed below.

Table 1. Bit Reduction and Memory Requirements for Middle C Long Note.

Sample Range	Bits Used	Reduction	Memory
1-1050	16	none	2K
1050-2000	8	2:1	1K
2000-20000	6	2.67:1	13.5K
20000-40000	5	3.2:1	12.5K
40000-100000	4	4:1	30K
100000-250000	3	5.33:1	56.3K
250000-500000	2,1	8:1,16:1	47K
		Total Memory =	162.3KB
		Overall Reduction =	6.2:1