

PROCEEDINGS

4th. symposium on small computers in the arts

OCTOBER 25-28, 1984
PHILADELPHIA, PENNSYLVANIA

SPONSORED BY:

IEEE Computer Society
Delaware Valley Chapter ACM/SIGGRAPH
Small Computers in the Arts Network
and in cooperation with ACM/SIGGRAPH



IEEE
COMPUTER
SOCIETY
PRESS

IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

ISBN 0-8186-0610-X
IEEE CATALOG NO. 84CH2087-5
LIBRARY OF CONGRESS NO. 84-81880
IEEE COMPUTER SOCIETY ORDER NO. 610

The papers appearing in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and are published as presented and without change, in the interests of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors, IEEE Computer Society Press, or the Institute of Electrical and Electronics Engineers, Inc.

Published by IEEE Computer Society Press
1109 Spring Street
Suite 300
Silver Spring, MD 20910

Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress Street, Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint or republication permission, write to Director, Publishing Services, IEEE, 345 E. 47 St., New York, NY 10017. All rights reserved. Copyright © 1984 by The Institute of Electrical and Electronics Engineers, Inc.

ISBN 0-8186-0610-X (paper)
ISBN 0-8186-4610-1 (microfiche)
ISBN 0-8186-8610-3 (casebound)
Library of Congress No. 84-81880
IEEE Catalog No. 84CH2087-5
IEEE Computer Society Order No. 610

Order from: IEEE Computer Society
Post Office Box 80452
Worldway Postal Center
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

**1984
Proceedings of the
Fourth Annual
Symposium on
Small Computers
in the Arts**

**October 25-28, 1984
Philadelphia**

**Sponsored by:
IEEE Computer Society
Delaware Valley Chapter ACM/SIGGRAPH
Small Computers in the Arts Network
and in cooperation with ACM/SIGGRAPH**

**Organized and Produced by:
Small Computers in the Arts Network**

Symposium Chairman: Dick Moberg
Proceedings Editor: Donna Mansfield

Organizing Committee: Eric Podietz
Bill Mauchly
John Senior
Kerry Sherin
Trip Denton
Tom Rudolph
Cathy Del Tito
Laura Giannitrapani
Joan Shafran
Alan Datri

INTRODUCTION

We would like to introduce these Proceedings of the Fourth Annual Symposium on Small Computers in the Arts by relating just a bit of the history of the group that organizes it, the Small Computers in the Arts Network. This group is committed to the promotion of small computers in the arts through its monthly newsletter, *SCAN*, concerts, the Symposium, and informal meetings.

This current group emerged out of an earlier one, the Personal Computer Arts Group which held its first computer arts activity, a computer music concert, in 1978. The group hoped to provide a forum for creative people interesting in using computers in the arts.

Computer arts were once the province of engineers and "techies", but now is more widespread. Computer art and music have infiltrated the popular culture; computer graphics can be seen frequently on television, and synthesizers are an accepted part of today's popular music. This Proceedings reflects the universal position of computers in the creative arts today.

We have included articles on computer arts topics: hardware, interface issues, software and tool design, aesthetics, education, as well as descriptions of gallery and performance pieces. This broad range of interests in, and uses of, computers in the arts was what the organizers of the Symposium had hoped to encourage and stimulate from their very first meeting and we are pleased to present a collection of articles so representative of the current interests and applications of the computer in the arts.

To contact the group, please write to:
The Small Computers in the Arts Network
Box 1954
Philadelphia, PA 19105

Table of Contents

Proceedings Committee.....	v
Introduction.....	vii
What Are Good Computer Pictures?..... R.E. Lucas	1
Is Consistent Motion Necessary for Good Computer Animation..... R. Leeman	4
Computer/Art--De-polarization and Unification..... D.M. Palyka	7
Illusion and Technology..... I.V. Kerlow	17
Effectiveness Standards for Computer Graphics..... M. Holynski and E. Lewis	23
Type Principles for Low Resolution Type..... A. Wagner	29
Microcomputer Applications for Graphic Design..... S.E. Metros	37
3-D Computer Generated Animation..... J.E. Sachter	42
A LOGO-Based CAI Program for Producing Representational Art..... J.J. McGinnis	50
The New Studio: The Computer Graphics Workstation..... A. Kaprow and J.K. Shafran	52
Interscore--An Interactive Score Editor for Microcomputers..... P. Prusinkiewicz	58
An Assessment of Microcomputer-Based Art Activities for Secondary Art Programs..... C.M. Naumowicz Zacher	65
Red Network: A Conceptual Art Piece Using Microcomputers..... G.K. Shortess	72
Computers and Poetry..... N. Wilson and W. Wright	76
Digital Sampling and FFT Analysis of Acoustic Sources: A Micro-Computer Implementation..... J. Klapholz	89

Data Compression of Computer Music.....	98
M. Keith	
Computer Music and the Human Interface: Imbedding Performance Knowledge at the Graphics Level.....	103
S. Levine	
Teaching Programming to Musicians.....	114
F.K. Dannenberg, R.B. Dannenberg, and P.L. Miller	
An Art Based Microcomputer Learning System.....	123
C.G. Del Tito	
Using Computers to Write and to Teach Writing.....	125
D. Mansfield	
Animating at Ohio State: Thoughts on a Graduate Program in Computer Graphics.....	130
W.J. Kolomyjec	
Author Index.....	135

WHAT ARE GOOD COMPUTER PICTURES?

Richard E. Lucas

Masters Program / Computer Graphics
Art Education Department
Ohio State University

ABSTRACT

Within the rapidly growing field of computer generated pictures there exist many schools of thought concerning an aesthetic evaluation of such imagery. Presently, and unfortunately, there are no generally recognized criteria for the qualitative evaluation of this art form.

This paper offers the critic and layman alike a more cognizant vantage point from which to seek and identify those qualities I percieve are inherent to computer art.

Once upon a time there was a man called Onk. He and his Cro-Magnon family lived in a cave in central Europe during the end of the last glacial period. One night, after a hard but fruitful day of hunting, Onk returned home with a fresh kill and began to draw a picture on the cave wall of what he had experienced during the hunt. When he had finished, Onk's family came forward and gazed intently at the figures Onk had made.

Twelve thousand years later a young paleontologist named Jill came upon the cave during a spelunking expedition. Discovering the picture in the cave, Jill reached for her camera and proceeded to photograph it for further study at a later date. She eventually wrote a book on her adventures and included her photos with the text.

One day Fred, a computer scientist, found the book in his local library and checked it out. Intrigued with the figures within the photographs, he set about designing a program that would generate the images by way of computer.

After many unsuccessful attempts, Fred finally arrived at a solution and made a picture on his print-out machine. He submitted the image, along with a paper he had written on the subject of computer pictures, to a national science magazine. The magazine published Fred's article as well as the picture he had generated on his computer.

Some time afterward a copy of the magazine found its way into the hands of Margo, who was the curator of a highly respected art museum. Upon reading the article, Margo became fascinated with the technique and irony of Fred's picture. She persuaded the museum to purchase the original print-out and included it in the museum's very next exhibition of modern art. As it happened, Robert, an art critic for the city's newspaper, was invited to the opening of the show. Shocked upon viewing the print-out that had been purchased by Margo from Fred, who was inspired by Jill's photograph of Onk's drawing, Robert bellowed angrily, "Just WHAT in the name of art is the meaning of THIS?!"

Thus is the question of art and computer and their relative significance within the grander scheme of aesthetic analysis.

As early as 1937, Electronics magazine was printing examples of oscillographic design based on the principles of analytical geometry. While these images were generated by a somewhat different method than those later calculated by analog computers they still represent some of the earliest attempts at combining mathematics and electronic machines in a creative endeavor. For over three decades, the field of computer imagery has been growing, changing, and pushing

its way into the already complicated domain of aesthetic pursuit. It has been treated by critics and scholars alike as a bastard child; abandoned at birth, an orphan of the arts, it continues to quest for its true identity. Now, on the brink of maturity, the computer image is making its strongest claim yet to its rightful heritage within the kingdom of art. Where and why this new method of creating images may find its niche in the world of aesthetics depends largely on our perception of it as a true art form.

Historically, the final realization of any new art form requires the general consensus of scholarly analysis as well as passing the test of time, the ultimate proof of its lasting value. The passage of time, of course, takes care of itself and with its passing does a fairly thorough job of weeding out trends and fads. It is the matter of analysis, or more particularly, a system for evaluating that which delineates itself as computer art which represents the real issue at hand. By what criteria might computer generated pictures be assessed as being art in the first place, and, if art, then how may we appraise the merit of that art? I suggest that we begin on familiar ground with a criteria for establishing that a given image produced by way of computer is a definitive work of art.

Generally, the issue of criteria for identifying art fuels much discussion and debate. For my purposes, however, I would like to offer a truncated definition of art that, while open to criticism, is my own and one I find suitable. "Art is that creation spawned from acquired knowledge combined with intuitive feeling, at once informative and emotional in import. Its roots lie deep in the imagination of man and its face is the intentional creation of man's own skilled hands. The only function of art is to provide an aesthetic experience; serving any other purpose distracts it from that intent and removes it from the domain of art."

The term "aesthetic experience" in the aforementioned definition demands further clarification. For this I will rely upon a condensed version of an interpretation by Professor Harry S. Brody of the University of Illinois:

"The aesthetic experience is the understanding of a work of art which, in turn, gives value to experience everywhere. This experience provides a clear intent of that art, and the intent provides ever new experience of it, affecting the viewer with both anticipation and fulfillment. It is the perception of that design containing superior and subordinate elements within it, and the realization that each element makes a valuable contribution to the total design."

As for the critical analysis of computer imagery, the basis for appropriate criteria rests firmly upon the established processes by which formal artistic endeavors are measured. However, it is necessary to expand the boundaries of presently-accepted analysis to include characteristics unique to computer art. The field of computer imagery encompasses a variety of applications most of which fall outside the purpose of art and are not my concern at this time. Therefore, I am speaking only about the computer picture created for the sake of artistic achievement and suggest that there are inherent properties of computer art, as art, that require some changes in the way we evaluate it.

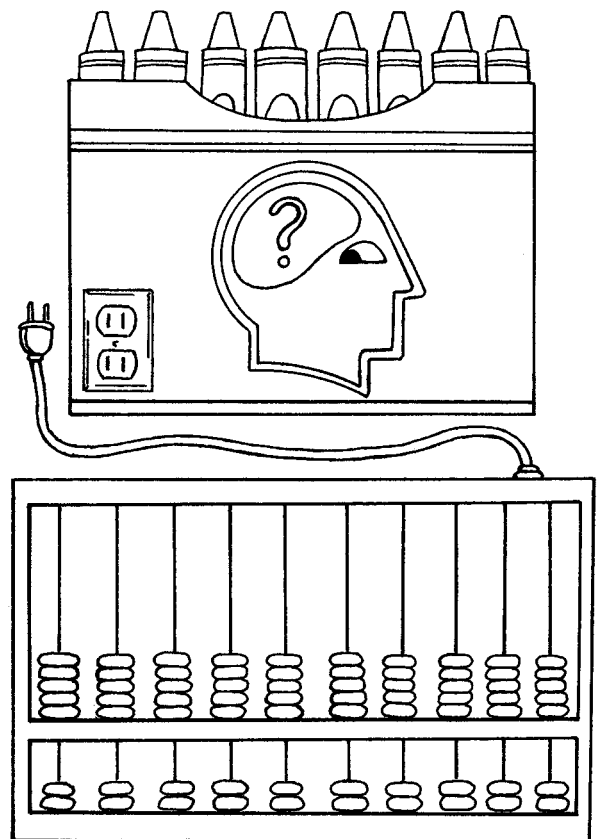
Critics generally rely upon a structured method for analyzing works of art that follows a set pattern of discussion. The four general categories of discourse include description, interpretation, evaluation, and theory. In the areas of description and interpretation, there are often overlapping arguments, matters of point relevant to both categories. It is within these two divisions that certain amendments should be affected to better evaluate computer art.

I will try to explain what I mean in considering the unique qualities of this art form, qualities that justify special attention in the field of critical analysis. These new considerations, of which there are two, I will define as the "philosophy of approach" and the "application of technology." Regarding the first term, I am referring to the intellectual problem solving process by which the artist endeavors to create. Computer imagery, from a scientific viewpoint, is a creative use of mathematical theory, or more simply put, is the science of numbers applied to making pictures. In

this field, the conceptual beginnings of an idea and the execution and realization of that idea are steeped in analytical reasoning involving the skillful use of algebra, trigonometry, calculus, and geometry. More than mere skills, these arithmetic philosophies influence the orientation of aesthetic sensibilities and, in my opinion, affect the artist's insight toward achieving a desired image. It is the nature of programming, I believe, that the creation of art by way of computer is inclined toward a unique level of conscious decision-making. It is this atypical creative process, the philosophy of approach, that must be investigated within the description and interpretation of computer art.

The second term, "application of technology," also represents a condition exclusive to this field of art. I am referring to the state of hardware and the hardware industry responsible for providing the means by which this art form has grown. To quell any immediate protest against this particular point, I would argue that there exists an acute distinction between the development of computer hardware and the development of such hardware as that of, say, the photographic or movie industry. While those two fields of endeavor have been supplied with a continuous flow of updated equipment to improve image quality, that flow represents a series of refinements related to long-standing principles of the physics of optics. Indeed, the situation changes when discussing the manner in which the computer hardware industry is developing. While many of the improvements in this field may be designated technological refinements, the technology itself is in a constant state of flux. There are and will continue to be quantum leaps in research providing the industry with revolutionary breakthroughs and generating entirely new concepts in the development of the hardware used to generate computer pictures. This seemingly endless transmutation of theory affects not only our perceptions of this technology but also the way in which we apply it to the creation of aesthetic imagery. As these fundamental changes influence this art form, so must it influence our description of and interpretation of it. The criteria for analysis should include an understanding of these changes if we are to fully understand the concepts behind the art itself. Our perceptions can only remain clear if our knowledge of the field is complete.

To what extent the aforementioned considerations influence the steps in which computer art is analyzed is the responsibility of the critic. I am only advising that these factors do play a role in the formation of criteria for analysis and should not be overlooked in the development of those criteria. I am not proposing sweeping changes in the established system for evaluating art to suit some lofty notion about computer imagery. Rather, I simply suggest that we stretch the boundaries of those criteria in specific areas to encompass what I feel are characteristics specific to the field of this art form. In all other respects, I am comfortable using the present means by which we assess and experience art. Ultimately, it is that aesthetic experience we receive from an artistic endeavor that determines its level of quality.



Rick Lucas '84

Is Consistent Motion Necessary for Good Computer Animation

Ruedy Leeman

We can look to the visual media of photography, film and video to establish some criteria for the evaluation of computer generated images. Some criteria for good photography are interesting subject matter, pleasing composition, and color and lighting appropriate to the intent of the photographer. These are just a few criteria used to judge artistic photography. The criteria used to judge product, commercial, architectural, and other service type photography are related to the photograph's economic effectiveness. The three previously mentioned criteria would be used to judge service photography but would be relative to the customer's purpose rather than the photographer's purpose. The criteria for good film and video would include the criteria for photography, consistent motion, an interesting plot, supportive music, and a coherent visual concept. These mentioned criteria would be the basic criteria for judging the film and the video media.

Since computer animation uses many of the formal features of photography, film, and video media, the criteria used to judge them are appropriate to judge computer generated animation. I believe consistent motion to be one of the most important criterion for judging computer generated animation therefore, I am limiting my discussion to the use of consistent motion to produce good computer generated animation.

Consistent motion is an important criterion for judging the quality of computer animation since consistent motion will support the overall concept of the computer animation by being an integral part of the animation. Roget's thesaurus describes "consistent" as in keeping with one's expectations, logical agreement among things or parts, and to be compatible or in correspondence. This description of "consistent" can help define what is meant by "consistent motion" in a computer animation. Consistent motion is motion that is in

keeping with one's expectations, motion that has logical agreement among its parts within a time frame, and motion that is not disturbing or incompatible to the mind's eye. Simply stated it is motion that is consistent with the purpose of the animation. If smooth, realistic, or erratic motion are needed for the interpretation of the artist's concept and the motion is producing the correct clues to this interpretation, then the motion is consistent. If the motion lacks consistency relative to the visual concept of the computer animation, then the motion is not transporting the intended ideas of the artist and thus is not successful. For example, if a computer animation is to be set in the "real world", then the motion needs to be consistent with the physical rules of the "real world" if it is to remain believable to the viewer. If the animation is not set in the "real world," then the motion need not use the rules of the "real world," but a consistency of motion is needed to interpret the concept of the computer animation. To keep the movement compatible and believable to the eye in a computer animation, an object must not jump from one position to another position without a smooth transition between the two positions unless this inconsistency of motion is consistent with the visual concept of the computer animation.

Various methods are available to establish a smooth transition of an object between two locations relative to time within a computer animation. This can be done by manipulating the object or by manipulating the position of the viewer. Assuming we manipulate the object, the artist can achieve a smooth transition between two time dependent locations of an object by looking at the established pathway before the first position and after the second position. If the path to the first position is a gradual curve and the path from the second position is very sharp, then the path between the two positions needs to

begin with a gradual curve from the first position and interpolate into a sharp curve going through the second position, to establish consistent motion between the positions. The acceleration and deceleration of the object at the two positions should also be consistent with the acceleration or deceleration of the relative positions of the object. If the object is decelerating on its way to the first position and accelerating when leaving the second position, then the object needs to interpolate from decelerating to accelerating between the two positions of the object. The amount of curvature of the path and the degree of acceleration or deceleration are up to the artist as long as the acceleration-deceleration and path are continuous within a relative radius of the location of the object. In this way, consistent motion can be established by manipulating the object.

Another method to establish consistent motion between two positions of an object is to change the position of the viewer, in other words, to cut to a different view during the position change of the object. The second view must be from a vantage point that displays the object from a completely different angle and relative distance to the viewer. The separate views indicate to the viewer that a change in position has taken place and permits the viewer to perceive the change without confusion. The importance of varying the two views can not be overstated. If the views are too similar, the object will tend to jump to the second position during the cut and the viewer will not accept the motion as real.

Movement in a computer generated animation needs to simulate the movement of objects in the physical world, assuming the animation is set in the physical world. Not all motion in a computer animation, however, needs to use a physical formula for motion control to be accepted as good animation. The physical formula of motion can be a starting point for the simulation of movement in the "real world". The artist needs to be aware of the physical properties of objects in motion. For example, when an object is thrown in the air, it generally moves along a parabolic curve, slowing down as it reaches the top of the curve, the object then begins to accelerate as it proceeds downward along the parabolic curve. Other physical properties are available for harmonic or sine curve movement. Whether the artist uses the exact physical formula or an approximation of the formula to control

motion is not important. It is important for the viewer of the animation to accept the motion to be real and believable. In order to insure that the motion is real, the artist needs to be aware of how objects move in the physical world so s/he does not have objects doing absurd actions that would detract from the appeal and interest of the computer generated animation set in a real world situation. An animation with believable motion is a good vehicle to present the artist's concepts and ideas.

Of course, the artist may find it necessary to completely break the rules of the physical world in his animation to illustrate some other imaginary world. The motion in the animation still needs to be consistent with the "imaginary world" the artist is trying to portray. For example, an artist might establish a world that has contradictory motions relative to the "real world". A ball could bounce higher after each striking of a surface, a small object could easily support a large object, or an object with short transport appendages could go faster for longer than an object with long appendages. If the artist established this "imaginary world" then he would need to be consistent in the motion within this world or the artistic intentions of the animation would be off target. Only if the artist establishes a world that goes between an imaginary and real world can the motion be totally nonsensical to be considered consistent. An object within the confines of an "imaginary or real world" can possess certain powers of motion that are not consistent within that world but this power of motion can not fluctuate in the object if the object is to possess consistent motion. For example, "Superman" comes from another world, he can fly through the air and stop locomotives with a single hand. This gives the author of the story a superhuman vehicle for telling his story. If "Superman" was inconsistent in his super motion, the title of the show would be "The Great American Hero," rather than "Superman". To illustrate, if "Superman" was flying through the air to rescue a damsel in distress and his power of flight was intermittent then "Superman" would be inconsistent with his established character. The audience would be forced to view "Superman" as a clumsy funny guy like "The Great American Hero" rather than the do good save the world boy scout character previously established. The intent of the author's story would go from a light drama to comedy. In any case, the inconsistency of motion of "Superman"

would cause the visual concept of the story to be something completely different from the intent of the author or artist.

In conclusion, without consistent motion relative to an "imaginary world" or "real world," the concept of the artist is attempting to bring forth in a computer animation can get lost or completely misinterpreted. In general, for real world situations if the motion in a computer generated animation is not noticed, the motion is probably believable to the viewer and supportive to the plot and concept of the computer animation. When the motion is noticed, usually something is not believable about the movement and the animation at the very least is distracting to the viewers. With an computer generated animation set in an "imaginary world", the artist intent determines the degree of absurd and irrational motion necessary to transport the artist's concepts and ideas.

Computer / Art -- De-polarization and Unification

by

Duane M. Palyka

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

June, 1984

ABSTRACT

An unnatural duality has occurred in western society that separates individuals into the archetypal roles "artist" and "scientist". Computer graphics/art, besides being a showcase for this duality, can provide a means for resolving it. As an individual well grounded in both roles, the author details his attempts to integrate these roles in his own personality in order to produce unique artwork.

No one can deny the separation of art and science in our western society. This duality is also reflected within the individual (a microcosm of the macrocosm) as a right-brain / left-brain polarization. This social separation, emphasizing the scientific side as the dominant side, also separates the individual from his psyche-- his imagination, his intuition, human factors and humanitarian concerns relegated by left-brain thinkers to the realm of the "irrational".

According to José Argüelles in his book *The Transformative Vision*, "... the problem of specialization did not become globally critical until the development of a mechanistic technology in Europe during the late Iron Age. Accompanying this development was the split of cerebral functions, leading to the creation of the two archetypal roles of artist and scientist ... Insofar as each is rooted in only one side of the human brain-- the scientist in the left, the artist in the right-- each is an incomplete

being."¹

As if separation of man into the "technical" man and the "psyche" man was not enough, the former became attached to the money/power structure and the latter has had to negate his psychic basis in order to survive. "Thus, as the culture develops along profit-making lines that stress novelty, gadgetry, and innovation for the sake of innovation, the artist consciously or unconsciously is swept along."²

Where does this leave the computer artist? Which side of his being forms his work? Is he an artist begging for tools from a technician, or is he a technician making left-brain images? Computer art is a showcase for this duality yet it also offers a rare opportunity to resolve it. My hope is that the individual can transcend society's definition of scientist and artist and function more freely as a human being. Blurring the social roles should make it psychologically easier for the individual to move between the left and right brain hemispheres.

A passive solution to this problem is to dissolve the mechanism that blocks the two. In other words, get out of the way and allow it to happen. The ego is the mechanism that prevents the natural flow between hemispheres. Convert the inner parent/child or master/slave relationship to one of co-operation. On the one hand, the "rational" left-brain cannot do everything itself. And on

¹José A. Argüelles, *The Transformative Vision* (Boulder: Shambhala, 1975), pp. 16-17.

²Argüelles, p. 16.

the other, when the artist swings the other way, there is usually equivalent resistance to getting involved with technical tasks such as programming. Perhaps future generations of artists who grow up programming computers and making computer art will have less rigid brain mappings than we have. Perhaps too, less rigid technical tools will replace programming and make the flow between hemispheres easier. In any case, we should really be interested in developing the whole man, not partials on either side.

Currently computer graphics/art exemplifies this duality and dominance even further by tying the creation of images to commercial production, bringing with it the superficial gadgetry and the "innovation for the sake of innovation" mentioned above. Under commercial emphasis, computer graphic tool development dominates and brings along with it competition and secrecy. The computers used now are expensive and the artist must create on the scientists' terms just to get access to the equipment. Furthermore, if the artist is not a programmer, he must depend upon the scientist to provide the software tools for him to realize his esthetic-- tools which the scientist thinks that the artist should have based upon what he sees as artistic tools of the past.

One of the paradigms of computer graphics/art is the desire and competition to design tools for generating the most "realistic" images. This attitude revolves around the scientist's myopic view of art seen only from the technical side. An artist's work is a balance of "psyche" and "techne", but it is the latter aspect that the scientist relates to and identifies with "art". "Both art and science rely on technique; the difference between them is in the degree of contact with psyche."³ From this viewpoint, the scientist-become-"artist" deals with spaces that have mathematically correct perspective with detailed and rigid object definitions. He becomes concerned with issues of how to make clouds, trees, and mountains look "accurate", without considering that this could be a subjective determination. It appears to him that the "jaggies", the staircasing effect inherent in digital images, serve no esthetic purpose but interfere with the technical (and, hence,

commercial) conception of visual reality. Thus, he forces art to be pulled further from the psyche and into the linear time/space left-brain domain. To be fair, it is a left-brain-based society which demands these tools and the scientist is fulfilling its needs for competitive commercial and military applications. It's frightening to think of videogames as being a training ground to kill people, but military simulators are computer graphics products not unlike videogames.

It's easy enough to point out deficiencies, but it is another matter to show positive alternatives. My own work has been concerned with resolving these issues, and I would like to share with you my personal exploration towards right/left brain unification and art/psyche involvement.

In all of my computer art work to date, I have done my own programming in order to merge my esthetics into the software. In the main body of art works I wish to discuss, I accept standard concepts of object-oriented computer graphics as my basis-- e.g., solids of revolution, polygon tilers, B-spline patches, and polygonal databases. Object-oriented computer graphics involves tools designed to build objects, not spaces. Spaces incorporating objects made with these left-brain tools usually have the feeling that the elements fit together arbitrarily. This is more so in spaces constructed by engineers than by artists since artists are sometimes able to transcend the design limitations of the engineer's tools. In any case, a spatially-oriented computer graphics tool set would promote better holistic thinking-- a right-brain function in the right-brain realm of visual spaces. Later, I shall discuss how spatial tool design relates to my work.

Using these programming concepts, I proceeded to design and build instances of these tools with particular orientation towards my artistic sensibilities. For example, I accept the notion that programming accidents and complex interaction sometimes contribute to interesting visual effects that may relate to my esthetic. I try to be open to creative associations. I then use these tools to build imagery based upon poetic mathematical concepts and my own psychic sense, which I describe as surrealistic in its patterns of free association. Allowing mathematical/programming ideas to merge with visual ones, I attempt to find a

³Argüelles, p. 94.

unique form of personal right/left hemisphere expression-- an expression controlled and sometimes hampered by my initial premise of starting with traditional left-brained computer graphics tools. The final work still reflects a stiff, object-oriented space with mathematically-calculated light sources, shading, and perspective. Frankly, the design of the original computer graphic tools is strong and it is hard to resist using them as a starting point. It is much harder to design personal tools of equivalent strength completely on my own.

In describing my work, I concentrate on artistic esthetic ideas and artistic technical ideas that are unique to my work, and I minimize the description of general computer graphics tools, which can be found elsewhere. Consequently, if the reader is not familiar with general computer graphics tools, I refer him to Andrew Glassner's recently-published book *Computer Graphics User's Guide* and Judy Sachter's Master of Arts thesis for Ohio State University entitled "The Basic Concepts of Three-Dimensional Computer Graphics for Artists". Both are oriented towards the artist as user. Ms. Sachter's thesis hopefully will be coming out in book form in the near future.

In thinking about three-dimensional computer graphics, one must deal with two related spaces: one is the two-dimensional visual space in which the work is seen, and the other is the multi-dimensional mathematical space that contains the numeric information and algorithmic schemes to generate the work.⁴ The visual space, being projected from the mathematical space, can be thought of as a certain way of looking at the latter space. The rendering technique involved takes a snapshot from one point of view of a more complete and complicated space that really has many points of view. This relationship of spaces is similar to the esoteric idea that our own view of experiential reality is being projected for us by a more fundamental and complete "greater reality"; and, as our consciousness grows, we are likely to experience more of the "greater reality" and develop greater understanding of how the lesser realities are projected from it and by it. Hence, in greater realization, we are able to free ourselves from

individual life patterns.

Likewise, in computer graphics, the more we understand the mathematical reality behind the images, the more we are able to understand the projection of the individual images and to see them in a different light. Our added awareness frees us from old patterns of thinking about images. Similarly, once we see how software systems logically work, we can free ourselves from being stuck to one particular software system. Instead of jumping around the branches of a tree trying to familiarize ourselves with each new branch, we should ground ourselves in the roots and see where the branches come from.

The artwork that I'm presenting has esthetic considerations that include more than "meets the eye". Although one can appreciate it on that level alone, if one reaches into the "greater reality" of the work, one gets more from it. This is like understanding the formal logical organization of Rembrandt's "Polish Rider" (The Frick Collection, N.Y., 1655) or the symbolic meaning within Robert Campin's masterpiece "The Annunciation" (The Metropolitan Museum of Art, N.Y., about 1425-28). Except for its deeper mathematical basis, this method of relating an artwork to its "greater reality" is merely an extension of what artists, art historians, and art lovers normally do with works of art.



Figure 1. "Space Carrots".

⁴Judy E. Sachter, "The Basic Concepts of Three-Dimensional Computer Graphics for Artists" (unpublished M.A. thesis, The Ohio State University, 1984), p. 11.

As in nature, my three-dimensional computer artwork has its evolutionary roots based in the "egg". Various parts of the work "Space Carrots" (see figure 1) are "mathematical eggs" formed using the concept of "solid of revolution". Although the term "solid of revolution" may be inferred from the following text, please refer to Sachter for a better description.⁵ A vector that forms the radius of a circle in the "X-Z" mathematical plane grows larger as it moves downward in the "Y" direction towards the center of the form, and then grows gradually smaller again towards the bottom to form an egg. The varying radius, using the vertical egg-like curve to determine its length, changes as it lofts circles in the "XZ" plane. In mathematical space, the circles themselves are really nothing but (x,y,z) point calculations that are deposited equidistantly from each other at the edge of the radius which spins in a circular path. Looking at it from the top down, each set of equidistant points vaguely resembles a circle. To give our egg visual substance, we use our calculated points to attach b-spline patches to our etheric form much as a roofer would use nails to attach shingles to a curved house.⁶

Even though the use of patches may seem to be over-kill to the technically-sophisticated reader for such simple "egg-carrots", they are indispensable for generating the green, amorphous forms in "Space Carrots". These forms are generated simply by adding some restrained random numbers to the radius lengths that form the egg.

Although I could generate a wide variety of interesting forms by varying my egg radii randomly, I decided to utilize solids of revolution for more controlled sculpting effects.⁷ However, instead of using multiple curves to define the object, I used a table-lookup technique similar to that used in texture mapping and bump mapping.⁸ In a frame buffer other than the rendering buffer, I drew an unusual picture of my face using conventional paint system techniques (see figure 2). The drawing's uniqueness comes from the fact that



Figure 2. "Face".

during its execution I visualized the drawing as an unwrapped soup-can label to be wrapped around an egg. I used my imagination to visualize my head as an egg with shades of light and dark representing the spatial differences between my head and the egg. In the "null" case, if I had presented to my rendering algorithm a totally gray drawing in place of "Face", it would have sculpted a faceless Humpty Dumpty.

In generating the egg, each time the radius required a new length, instead of adding some random number to it the program would add to it a number from the frame buffer which contained the face drawing. If the number showed white in the "face drawing" buffer, the radius would move out the furthest, and if the number showed black, the radius would shrink towards the center of the egg.⁹ As you can see by the resultant "Egghead" picture (see figure 3), the experiment was not as successful as one would have hoped. My final result resembles what I look like in someone else's reality projection but not what I look like in my own! Actually, the problems I encountered with this experiment stem from two sources: (1) my preconditioning as an artist in seeing light and shadows in figure drawing and (2) the grossness of the tools I'd developed to do the task. That taught me not to get caught up in the above-

⁵Sachter, p. 19.

⁶Andrew S. Glassner, *Computer Graphics User's Guide* (Indianapolis: Howard W. Sams & Co., Inc., 1984), p. 130.

⁷Sachter, p. 19.

⁸Glassner, pp. 101-104.

⁹Glassner, p. 161.

mentioned "realistic rendering" paradigm!

Notice that I incorporated the Z-buffer into the lower half of the picture as part of the image. A Z-buffer is an extra frame buffer or two used to hold the picture's depth information in order to allow a programming mechanism to resolve depth overlap and intersection on a pixel-by-pixel basis. This mechanism determines how much of the image gets projected from the multi-dimensional "greater reality" to the two-dimensional frame buffer space.¹⁰ Actually, exotic uses of the Z-buffer adds another chance component to the scheme of things. Since I love to incorporate chance factors into my artwork, I use the Z-buffer extensively either visibly or behind the scenes throughout my work.

"Warm-Cool One" (figure 4) is just such an example of one of my "behind-the-scenes" use of a Z-buffer. This picture was created using both the finished two-dimensional frame buffer image of "Space Carrots" and its saved Z-buffer. On a pixel-by-pixel basis, I used the following procedure to modify the image: (1) First, I separated the pixel's color component from its intensity component. This was fairly easy since using an 8-bit pixel, I defined the intensity to be the lower 5 bits and the color to be the upper 3 bits. Here, I had 8 different colors to work with and 32 different intensities

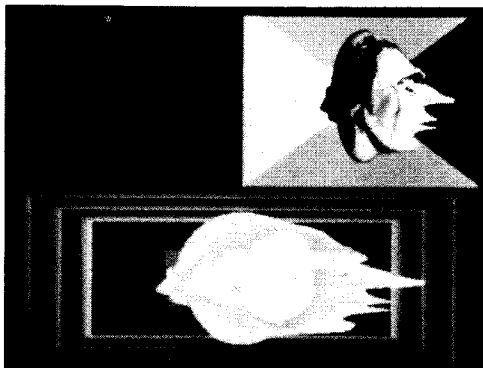


Figure 3. "Egghead".

¹⁰Glassner, p. 169.

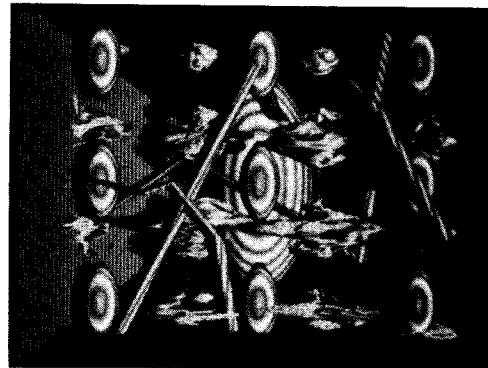


Figure 4. "Warm-Cool One".

for each color. (2) Then, I picked out the pixel's associated Z-buffer value from the same pixel position in the saved Z-buffer. (3) Using this value as an index into a red/green peppermint scheme which I had mapped onto "Z depth", I accordingly changed the pixel's color value. (4) Finally, keeping the old intensity with the new color, I put the new pixel into the old pixel's frame buffer position.

This scheme actually started out as an attempt to make use of artistic warm/cool space to exaggerate depth cues-- a trick which makes a two-dimensional picture look more three-dimensional. Later pictures along this line were more successful but were less visually interesting than this one. In my work, I have the general attitude that the visual image takes precedence over the concept behind it.

Two of the most powerful tools designed for three-dimensional computer graphics are the "polygon tiler" (or "polygon renderer") and the use of a three-dimensional points-polygon database to generate images. In the "greater reality", the points-polygon database contains numeric information on how to render a particular complex form, and the polygon tiler understands this format and renders a view of the form in the visual frame-buffer space. I shall keep the description brief here and again refer the reader to longer descriptions in both the Sachter and Glassner books.

I developed the polygon tiler which I used extensively in the next set of works to reflect my own esthetic interests. I actually used it for the first time back in "Space Carrots". Notice, in "Space Carrots", the randomly formed beam-like form that intersects the space. The sides of the beam, with accompanying aliasing artifacts, were rendered by the tiler.

In "Figure with Aura and Guards" (figure 5), we have three apparent instances, or views, of a female figure defined as a points-polygon database and rendered using my polygon tiler. I say "apparent instances" because the central form in the picture is actually composed of several exploded instances of the figure which are overlayed with various degrees of transparency. The form explosion occurs along the vector direction of the polygon normals, calculated at each vertex of polygonal intersection. In this case, each "normal vector" at a particular vertex is the sum of vectors perpendicular to the surface of every polygon which intersect at that vertex.¹¹ If the normals were rendered along with the polygons, the figure would look like a porcupine. However, the normals are like ghosts in that they are not rendered from the "greater reality" into the preceived reality, but may affect it nonetheless.

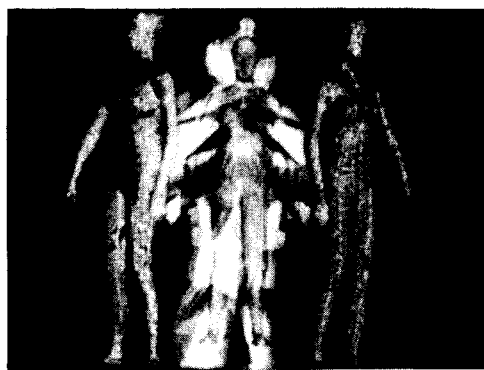


Figure 5. "Figure with Aura and Guards".

In another behind-the-scenes manuever, I have included the "egg" concept as part of the "greater reality" of "Figure with Aura and Guards". The texture on the surface of the figures is a reflection of the egg's presence. The subtle mechanism behind the presence of the egg needs further clarification: The egg, in this case, is a sphere with infinite radius whose center is the center of the female database. Instances of the picture's Z-buffer are randomly placed on the surface of this sphere as part of the following algorithm: Rays from the figure emanate along the normal vectors away from the figure and towards the sphere's surface. They retrieve Z-buffer patterns and colors to be put back on the surface of the figure.

Here, we have a dance which is an interaction between the Z-buffer mapped on the surface of the invisible egg and the figure being rendered at the egg's center. A change in one affects the other. As in a dance of subatomic particles, complex time-space interactions occur which make events appear unpredictable.

In addition, the figure's dance with the egg is symbolized by her auric emanations towards the egg. In fact, the central figure turned out to be so delicate that I just had to put two guards in the picture to protect her.

The remaining pictures in this series were produced by the same algorithmic

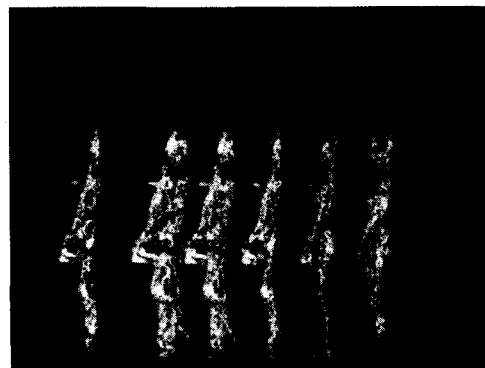


Figure 6. "Sparkling Giacometti Sequence".

¹¹Glassner, p. 68.

mechanisms used in the previous ones, and all contain the dance with the egg. "Sparkling Giacometti Sequence" (figure 6) reflects the use of the "augmented" transparency mode of my polygon tiler. The "augmented transparency mode" renders transparent polygons with sparkles at the edges of the polygons. The "sparkles" really are the result of a bug, or programming mistake, that occurred during the implementation of the transparency feature. Here is a case where a mistake in the code became a "feature" of the program. I kept the bug as part of the program and worked around it to complete the tiler's transparency option.

The four works "Sparkling Giacometti Sequence", "Tall Movement" (figure 7), "Stretched Movement" (figure 8), and "Stretched Movement Zoomed" (figure 9) are products of a certain spatial tension programmed into their "greater reality". This spatial tension causes distortions to figure instances which are placed in a spatial line with each other. The distortion occurs along the direction of the normal component which is in line with the other figures. However, the end figures remain unaffected when rendered. The distortion also varies from figure to figure based upon where a figure is placed relative to the others.

"Picasso Two" (figure 10) is the pinnacle of this series. It is the product of the complex interaction of every element mentioned so far. In this work, the effects created by chance, grouped with those created by unpredictably complex algorithmic interaction, form a visual tension when played against the solid definition of the figure. This tension in visual space is similar to that which I find in many of Picasso's paintings.

Spacial tension is also an integral component of the next series I attempted-- a series much different from the previous one. These new images are the product of a unique attempt to create a form-generating space that resembles, in some sense, our own real star-filled "outer" space. This new visual space has stars in it like in the cosmic space. They create gravitational forces on objects which enter the space. However, that's where the similarities end. The visual space differs from "outer" space in that these stars have no mass and their interacting gravitational forces actually create visual forms. Within each of these

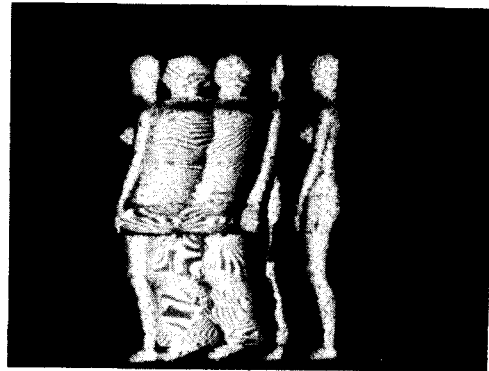


Figure 7. "Tall Movement".

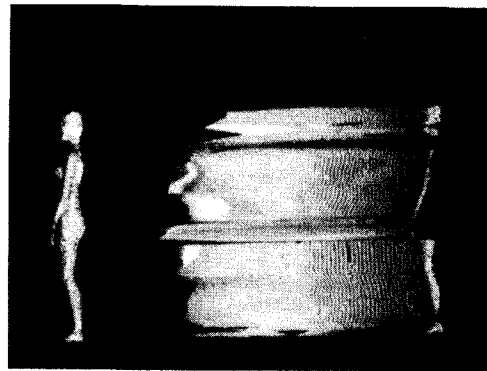


Figure 8. "Stretched Movement".



Figure 9. "Stretched Movement Zoomed".

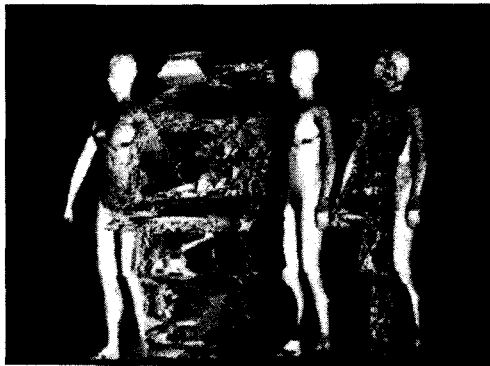


Figure 10. "Picasso Two".

contained spaces are placed several random points, or stars, that exert gravitational forces upon the space. A blobby form is then generated in their midst by the forces created by the stars. "Space One" (figure 11) is a six-by-six matrix of star spaces with their associated blobby forms. "Tex Blob" (figure 12) is one of those space blobs with patterns texture-mapped onto its surface. And "Warm-Cool One Dipped into Gravity Space" (figure 13) is "Warm-Cool One" dipped into one of these gravity spaces. Finally, "Folded Space" (figure 14)-- a freer, more complex result-- is composed of overlapping spaces that create blobby forms which ripple away from their centers.

Earlier, we mentioned the need to create more space-oriented computer graphics tools for more right-brained, holistic methods of design. In some sense, this last series involves a primitive attempt to do just that. Please do not infer that the visual spaces should resemble our cosmic space-- that was simply part of its artistic implementation-- but rather, the merit of its artist-tool design lies in its facility for designing a visual space which, in turn, controls the design and placement of objects within it. This reverses the usual method of spatial design within which objects are arbitrarily placed together. Philosophically, the objects in the real "greater reality" are much more related than what our senses normally lead us to believe. My artwork attempts to suggest that relationship. I would prefer general artistic tools that do the same.

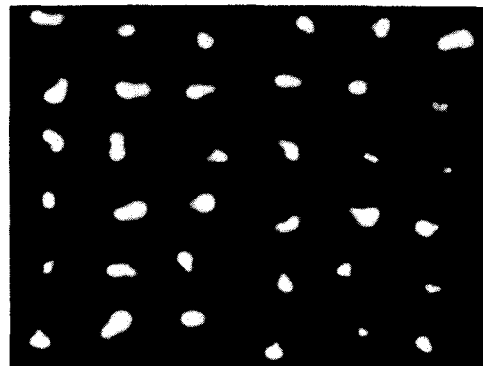


Figure 11. "Space One".

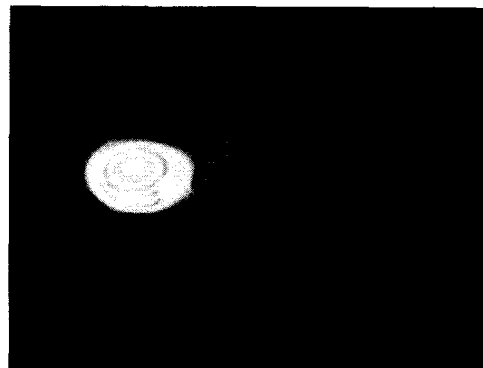


Figure 12. "Tex Blob".

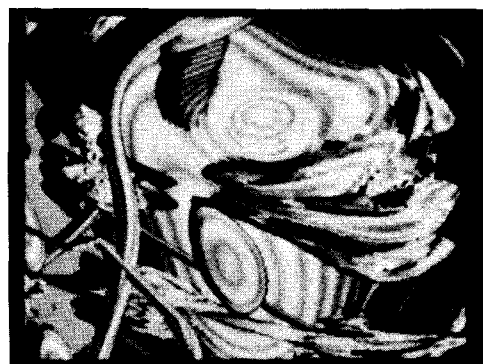


Figure 13. "Warm-Cool One Dipped into Gravity Space".

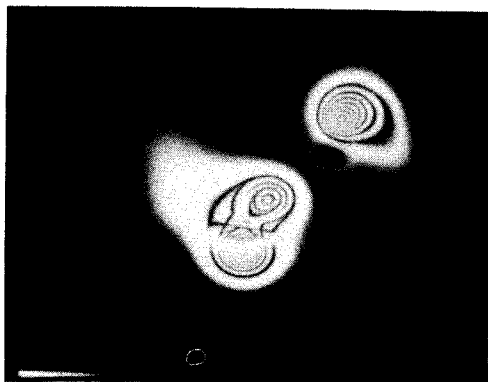


Figure 14. "Folded Space".

After this, for my next work I chose a medium more flexible for realizing psyche/art involvement with less emphasis on mathematics. I used traditional animation techniques with a computer production base. This 22-minute piece called "Living Above the Mouse's Ear" reveals psychic permutations which create their own dreamlike spaces—typical of my conventional painting concerns.

All of my artworks, whether computer-generated or conventional, have a strong surrealistic basis. Working in conditions conducive to a psychic flow causes various interesting images and ideas to emerge on both conscious and unconscious levels. Here one can relate the creative processes and visual products to theories based upon manifestations of the numinous element, or the collective preconscious. J. C. Gowan defines "preconscious" as "... that aspect of the psyche, sometimes, not always, available to the ego. Preconscious insights tend to be expressed through right-hemisphere imagery."¹² Gowan's "collective preconscious" concept is similar to the "collective unconscious" concept of Carl Jung except that its definition is more flexible towards "becoming conscious".

Making computer art through the design and/or modification of programming tools requires great efforts in time and energy. Just

¹²John Curtis Gowan, *Operations of Increasing Order* (Westlake Village, California: By the Author, 1426 Southwind Circle, 1980), p. xvi.

to develop the programming proficiency necessary to work this way, I passed up the opportunity to make a lot of conventional art. However, computers are seductive and fun; I enjoy programming. Time and energy permitting, programming by day and painting by night gives a pleasant right-brain/left-brain balance to my life. It even gives me a means of making a living without compromising my art. However, you will notice that these last two statements are indicative of the social aspect of the problem. I feel that resolving this duality on an individual basis is important enough to warrant the effort. Such efforts further the development of consciousness, both individual and collective.

ABOUT THE AUTHOR

Duane M. Palyka is an artist and computer scientist working and teaching at the New York Institute of Technology Computer Graphics Laboratory in Old Westbury, New York. He received a Bachelor of Science degree in Mathematics and a Bachelor of Fine Arts degree in Painting from Carnegie-Mellon University and a Master of Fine Arts degree in Art from the University of Utah. His computer art has been exhibited internationally since the 1968 "Cybernetic Serendipity" show at the Institute of Contemporary Arts in London. In the early and mid 70's Mr. Palyka was a research associate and systems programmer in Computer Graphics at the University of Utah.

REFERENCES

1. Argüelles, José A., *The Transformative Vision (Reflections on the Nature and History of Human Expression)*. Boulder: Shambhala, 1975.
2. Edwards, Betty, *Drawing on the Right Side of the Brain*. Los Angeles: J.P. Tarcher, Inc., 1979.
3. Glassner, Andrew S., *Computer Graphics User's Guide*. Indianapolis: Howard W. Sams & Co., Inc., 1984.
4. Gowan, John Curtis, *Operations of*

Increasing Order. Westlake Village,
California: By the Author, 1426
Southwind Circle, 1980.

5. Gowan, John Curtis, *Trance, Art, and Creativity.* Buffalo: Creative Education Foundation, 1975.
6. Sachter, Judy E., "The Basic Concepts of Three-Dimensional Computer Graphics for Artists." Unpublished M.A. thesis, The Ohio State University, 1984.

ILLUSION AND TECHNOLOGY

Isaac Victor Kerlow

26 Gramercy Park South, New York, NY 10003

ABSTRACT

Illusion and Technology reviews the ways in which technology, including computer technology, has influenced imaging procedures throughout history. The two main objectives of this paper are to present a methodology for analyzing and classifying computer generated images according to their technical attributes, and to encourage an awareness of the aesthetic issues that exist in their realm.

Categories: Imaging technologies, tools and methods, visual language, computer generated images, relational database.

1. THE EVOLUTION OF IMAGING TECHNOLOGIES

Technology in general can be defined as a set of tools and methods designed for solving a specific task. Imaging technologies are the set of tools and methods designed for the creation of images. Imaging tools are the physical instruments (such as pencils, brushes or computers) that can be used to perform a specific operation. Imaging methods are the set of procedures and operations (such as draw, erase, color, move) that describe the way in which the tools should be used in order to achieve the desired results.

Imaging technologies have been a critical instrument in the development of human consciousness and the communication process. Man has created images since the earliest stages of his life on earth. Visual images, especially those which represent reality, have always had a magical power of evocation: they can create illusions of time and space. The early examples of visual communication were highly symbolic. Nevertheless, some of them like the ancient pictograms created by prehistoric civilizations evoked reality with such power that they could substitute it.

The visual representation of reality and ideas is based on a set of visual elements and operations. The creation of effective illusions extends into the knowledge and control of the imaging tools and the visual language. Successful imaging operations transform images into powerful icons and objects into mythical symbols.

During the Renaissance the medieval barriers that prohibited the representation of the material world were broken with logical thought. Man rediscovered the beauty of nature and embarked in a major effort to develop the technology for representing the world. Renaissance artists, not satisfied with a (typically gothic) symbolic representation of the world, developed a *science of painting* which contained methods and rules to represent all the subtleties of the visual world in a measurable way.

The methods developed during the Renaissance for representing reality remained in use for several centuries. But some imaging technologies developed after them affected imaging techniques and theories. For example, the development of photographic techniques that captured reality without direct human intervention thrust painters into the exploration of alternate methods for representing reality and even for creating non representational images. Years later, the visual flexibility of film and television enriched the imaging environment with movement and time. Today the computer offers new creative alternatives. A more detailed description of the evolution of imaging technologies can be found in reference (4).

After centuries of innovation in the imaging field, visual creators possess not only a collection of imaging tools and methods but also a structured visual language with an extensive vocabulary and precise rules. Visual language, just like written language, has a syntax as well as discrete elements and operations. Mastering the visual language results in the ability to build precise and effective visual communication.

The visual language can be implemented in different technologies. And each technology enriches the visual language by adding new imaging techniques and a vocabulary of its own. The visual language can be implemented today in the computer, the latest technological innovation in the field of imagemaking.

2. THE EVOLUTION OF COMPUTER IMAGING TECHNIQUES

The first computer imaging systems were developed in the late fifties. Most of those systems were based on large computers, ran in non-interactive modes and displayed visual information on monochrome vector displays. Such systems were used to visualize and represent objects and situations in different applications including command and control systems, simulation of real time flight and physics experiments, medical diagnosis and design and manufacturing. The *Sketchpad Drawing System* developed by Ivan Sutherland in the early Sixties marked the beginning of interactive computer graphics.

During the sixties several hidden surface algorithms were developed and implemented on real time color surface shaded animation systems. The first graphics oriented languages were developed during this period and, also, image processing techniques (such as those developed at the Jet Propulsion Laboratory for NASA's Space Program) were incorporated to computer imaging systems.

The seventies and eighties have witnessed the development of specialized graphics hardware and a large variety of realistic rendering algorithms which are of special importance to the consolidation of the visual language of computer generated images.

3. REALISM IN COMPUTER GENERATED IMAGES

The visual language and creative values of early computer generated images were somehow determined by technical limitations. But the development of computer technology (both hardware and software) as well as the expansion of the creative goals and standards lead to the development and consolidation of a visual language unique to computer generated images.

Various strategies have been devised to implement realism in computer generated images, and each one of them generates very different results. Rendering algorithms, data representation techniques and image resolutions are three important factors that affect the style and quality of computer generated images. For example, when four different computer

animation production companies created images for the movie *Tron* (1982) each one of them stamped their own style and visual quality to the sequences each one of them produced.

It is important to understand the technology behind computer generated images in order to understand their true value and meaning. Since the sixties various methods have been devised for the creation of realistic computer generated images and some of them are listed in the following paragraphs. A detailed description of each method is beyond the scope of this paper, but detailed technical descriptions and an extensive bibliography can be found in references (2) and (6).

We find different imaging methods in each of the stages of the process for building a computer generated image. Such methods can be divided in object definition procedures, hidden surface removal algorithms, shading models, and surface definition procedures.

Object definition procedures include blueprint digitization, coplanar elevations, and mathematical or procedural descriptions. Early mathematical descriptions were based on geometrical primitives, more recent methods include regular and irregular polygon meshes, curved surfaces and fractal surfaces.

Methods for the removal of hidden surfaces in polygon meshes include object space algorithms such as Warnock's area subdivision (1969), Watkins' scan line (1970), and Newell's depth sort (1972). Improvements to these methods have been made by several authors during the seventies and eighties. Image space algorithms were first sketched by Appel (1967), and developed later by Whitted (1980).

Methods for hidden line removal in curved quadratic surfaces (sphere, capped cone, ellipsoid, etc.) or parametric bicubic patches of different orders by were developed by Catmull (1975), Blinn (1978) and Whitted (1980).

Early shading algorithms based on Lambert's cosine law determined only visible surfaces with diffuse reflection. Warnock (1969) introduced a reflection model that took into account the object color and the specular component of light (highlights). Gouraud (1971) developed a method of normal interpolation to generate smoothly curved surfaces. Phong (1973) developed a reflection model that included diffuse and specular terms. A model of reflecting surfaces developed by Torrance and Sparrow (1967) was based on illumination engineering theory, and was implemented by Blinn (1977, 1978). Cook and Torrance (1982) developed a method for describing the optical characteristics of a material and, therefore, for rendering more complex and subtle images.

Several shadow algorithms have been reviewed by Crow (1977). Transparencies were made possible by the ray tracing procedures developed by Appel (1967), The Mathematical Applications Group (1968), and improved by Kay (1979), Whitted and Weimer (1980) and Hall (1983).

The creation of visual textures with mapping procedures was introduced by Catmull (1974) and improved by Blinn and Newell (1976). Normal perturbation procedures were elaborated by Blinn (1978) and developed by Haruyama and Barsky (1984). The creation of spatial textures has been executed with fractal procedures by Fourier and Carpenter (1981) and Norton (1982), and with particle systems procedures by Reeves (1983).

4. A HISTORICAL ANALOGY

The search for creating realistic computer images has many parallels with the efforts made during the Renaissance for the realistic representation of reality. During the Renaissance, artists and technicians developed the technology needed to represent reality and to surpass the threshold of perception. Renaissance artists, for example, built pigments, oil-based aglutinants and refined the rules of representation.

Many of the central topics in Renaissance art theories (such as volume, lighting, texture and perspective) are some of the central topics in today's research for the generation of realistic images with computers. Today, though, research is conducted with a different technology. Computers allow for the numerical description of images in three dimensional environments and for their automatic transformation into a two dimensional environment.

The imaging techniques employed by different artists or schools (during the Renaissance and today) mark the style of their works. Styles of pictorial representation of reality have evolved with the imaging technologies and stylistic differences can be found by analyzing the technical procedures for creating images.

Today we can group the images produced by Renaissance artists based on stylistic considerations that arise from technical procedures. For example, the Italian Renaissance painting can be defined in terms of four major schools or styles according to the imaging tools and techniques they employed. Most of the early works dealt with developing the syntax rules and the basic operations of the visual language, while later works dealt with structuring those elements and symbols to depict complex scenes and stories. A brief description based on the technical achievements of the four major

Italian Renaissance painting schools is suggested in the following paragraphs.

The Florentine School painters (i.e. Giotto, Fra Angelico, Boticeilli, Leonardo da Vinci, Michelangelo) were the masters of form and movement. They simulated reality and represented the surface of things with an almost scientific fidelity. They elaborated the principles of perspective, studied the effects of light on volumes and portrayed people with a high degree of likeness and expression. In later years, based on direct observation and study of human anatomy, they mastered methods to represent the human nude in motion.

The invention of a perspective based on mathematical was first sketched by Brunelleschi and later codified and formulated by Leon Battista Alberti in his work *Della Pittura* published in 1457. Leonardo da Vinci introduced at the end of the sixteenth century the *aerial* perspective or simulation of the atmospheric phenomenon by which distant forms lose the sharp definition of their edges and appear dimmer and less distinct to the observer.

Florentine artists considered light as a condition for giving relief to shapes and for making the model *visible*. da Vinci, for example, examined shading subtleties such as transparent objects and the variation of color intensities created by light. For more detail on chiaroscuro and perspective techniques in the Italian Renaissance see references (1) and (7).

The Venetian School painters (i.e. Giovanni Bellini, Giorgione, Titian, Tintoretto, Canaletto, Guardi and Tiepolo) are characterized by the splendour and harmony of color that determines the mood of their paintings. They did not only fill drawings with color but also portrayed atmospheres. Their works are full of profound emotion based on the psychological values of color. Venetian portraits of common people are more than a likeness, they transmit emotions.

The Central School painters (i.e. Duccio, Simone Martini, Piero de la Francesca, Perugino, Raphael) were not obsessed, as the Florentine painters, with naturalism. They were concerned more with the theme than with the formal aspects of the work and developed an intricate sense of composition to illustrate stories.

Most of the Northern School painters (i.e. Pisanello, Mantegna, Antonello da Messina, Coreggio, Parmigiano) remained linked to the principles of the Middle Ages and to the rather gothic Flemish style dominant on the other side of the Alps. With the exception of some influenced by the Florentine and Venetian Schools (i.e. Coreggio), most of these painters were interested in reviving antique techniques and values.

5. A CLASSIFICATION METHODOLOGY

A methodology for analyzing and classifying computer generated images based on their technical attributes can become a powerful tool for evaluating them in the context of contemporary images. The purpose of such a classification is not purely encyclopedic but helps us understand the style and visual value of computer generated images.

The method suggested here classifies computer generated images based on their technical attributes, it analyzes their structure and locates them in a historical perspective. This classification strategy examines the critical steps of the creative process that determine the attributes of the final image. Eventhough this classification model is based on the analysis of technical attributes, its purpose is to classify the images themselves and not the specific techniques (procedures or algorithms) that generated them.

This classification methodology structures information as a *relational database*. Such a database is structured on relations; the entries can be related with each other in different ways and they can be reorganized in different configurations. A more in-depth look at relational databases can be found in reference (5).

Relational databases can be easily understood by users and they facilitate to change the database without affecting the logical structure and applications programs because they represent data in a logical way, independently from the internal representation of the data to the computer. Relational databases keep entries located in a dynamic structure with as many levels of detail as needed. Relational databases are usually represented in a tabular form as a two dimensional array where a field (or domain) is defined by a class of attributes. Several fields define an individual entry or record (table 1).

Data input can be made through on-line data entry forms that include critical attributes (table 2). Such forms can be structured as multiple choice forms or as blank forms. The use of numerical codes makes the implementation of such database possible in a commercially available database package.

A list of entries can be sorted in different ways by using different sort keys. One or several sort keys can be used to select entries in relation to meaningful categories. For example, computer generated images classified from a historical perspective would use date as the primary sort key, using software codes and resolution as sort keys would result in a classification of realistic images,

Table 1. Simple relational database in a two dimensional tabular form.

	<ol style="list-style-type: none"> 1. Entry # 2. Author Name 3. Work Title 4. Date 5. Company / School 6. Country 7. Software Codes 8. Hardware Codes 							
	1	2	3	4	5	6	7	8
	2370	Blinn J	Voyager	1982	JPL	USA	03 07 00 04	01 07 02
RECORD	0732	Greene N	Quark	1984	NYIT	USA	01 05 00 09	03 04 00
	3270	Kawaguchi Y	Growth	1983	Nippon E C	JAP	01 07 04 02	02 05 01
	7320	Porter T	1984	1984	Lucasfilm	USA	02 07 01 02	03 01 03
	FIELD							

Table 2. Database entry form.

ENTRY NUMBER:

TITLE OF WORK:

NAME OF AUTHOR:

Birthdate

Sex

DATE CREATED:

PLACE CREATED:

Company/School

City

Country

HARDWARE DESCRIPTION:

Computer Brand and Model

Computer Category Codes

Memory KBytes

Frame Buffer Depth

Output Device Resolution

SOFTWARE DESCRIPTION:

Data Definition Techniques

2D primitive shapes

3D geometrical primitives

Manual digitizing

Scanning

Mathematical functions

Procedural methods

AI techniques

Rendering Techniques

Hidden surface removal

Shading and lighting model

Texture method

Color space

Image processing

CREATIVE ENVIRONMENT:

2D

3D

4D

ORIGINAL:

Animated Sequence Frame

Installation

Hardcopy

Photography

Hybrid Methods

CREATIVE PURPOSE:

Scientific

Technical

Commercial

Artistic

ADDITIONAL REFERENCE INFORMATION:

Artist involved in software design?

How was the system accessed?

Theme of work?

Etc.

and so on and so forth. Unusual classifications can be defined using more complex sort keys (table 3).

An interesting application of this classification methodology is the selection of computer generated images based on their visual excellence, creative innovation and aesthetic qualities. It is not the purpose of this classification methodology to solve lengthy aesthetic discussions or to explain in detail the changing nature of art and artistic creation. But the model suggested here can help to determine the aesthetic value of a specific work based on its stylistic attributes that derive from a technical analysis.

Computer art can be defined as those works that have been created with the aid of computer-based tools and procedures, and with the purpose of fulfilling an aesthetic need in the creator as well as in the audience. The final database helps to locate a specific work or group of works in a conceptual frame. Nevertheless, without a direct judgement of the image itself, the database does not contain the value judgments to fully judge the aesthetic value of an image.

This methodology can also help to group works in different schools or styles according to the imaging methods used. This analysis and classification method can be especially useful when dealing with works that have been created with integrated software packages that are capable of alternate types of rendering images in more than one way. Some of those software packages (or *test beds*) are described in references (3) and (8).

CONCLUSIONS

Imaging technologies have always been an integral part of the imaging process and, therefore, have always influenced the resulting images. The analysis of the historical development of imaging technologies can help to understand the relation that exists between images and the technologies used to create them.

The analogy between computer imaging techniques (which still are in the developmental stage) and the Renaissance painting techniques can illuminate the importance of technical issues for a stylistic classification.

The organization of computer generated images as a structured relational database can provide valuable information on their style and technological development. The classification model suggested here can be implemented in commercially available database management packages for microcomputers, or in artificial intelligent expert systems.

Table 3. Sorting the database with different primary and secondary sort keys (upper case) generates different reports.

Report A (Author/Historical):

AUTHOR NAME
BIRTHDATE
Company
City
Country

Report B (Historical/Regional):

DATE
COUNTRY
Author Name
City

Report C (Work/Techniques):

WORK TITLE
SOFTWARE CODES
Hardware Codes
Author Name
Company
Country

Report D (Artistic Works):

PURPOSE CODES
ORIGINAL CODES
Work Title
Author Name
Date

REFERENCES

1. Barash, Moshe. *Light and Color in the Italian Renaissance Theory of Art*. New York: New York University Press, 1978.
2. Foley, James D. and Van Dam, Andries. *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley, 1982.
3. Hall, Roy A. and Greenberg, Donald P. "A Testbed for Realistic Images." *Computer Graphics and Applications*, November 1983, pp.10-20.
4. Kerlow, Isaac Victor. "Illusion and Technology." M.S. Thesis, Pratt Institute. New York, 1983.
5. Martin, James. *Principles of Data-Base Management*. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1976.
6. Sutherland, I. E., Sproull, R. F. and Schumacker, R. A. "A Characterization of Ten Hidden-Surface Algorithms." *ACM Computing Surveys*, Volume 6, Number 1, 1979.
7. White, John. *The Birth and Rebirth of Pictorial Space*, 2nd ed. London: Faber & Faber, 1967.
8. Whitted, Turner and Weimer, David. "A Software Test Bed for the Development of 3-D Raster Graphics Systems." *Computer Graphics*, August 1981, pp. 271-277.

EFFECTIVENESS STANDARDS FOR COMPUTER GRAPHICS

Marek Holynski and Elaine Lewis

Boston University

Abstract

This paper presents the development of standards for computer graphics based on the definition of psychologically relevant structural variables. Different methods of obtaining graphic stimuli which illustrate these variables in a visual form are overviewed. A discussion of the creative potential of computer generated imagery is supported by initial evaluation experiments aimed at assessing the relative effectiveness of various structures.

The Need for Aesthetic Standards

Producers of computer graphic art often become so excited by the technical process of creating graphics that they have little or no regard for the aesthetic quality of their finished work. A few masterpieces are produced along with hundreds of dull and boring pictures.

In the past, effective graphic production relied upon the technical skill of the producer. That is, the artist had to become a master of a particular medium, often through years of tedious training and practice. This "apprenticeship-period" allowed an artist to develop a refined sense of aesthetics along with technical skill. Some of the new computer graphics generation systems, however, free producers from many technical aspects of creating images. By offering high quality results through easily learned command languages and flexible input devices, users can become "instant artists." Unfortunately, their aesthetic sense may not have kept pace with their ability to make pictures. Their work is technically proficient without being artistically gifted.

The challenge of producing more pleasing pictures can be met through the development of formal criteria or standards for aesthetic quality. In fact, the computer can assist the selection of superior images without usurping producers' control. That is, a "filter" can select potentially good images from the population of possible images. Ultimately human evaluators will select the best from the potentially good, and initially their judgements will establish the standards which guide this filter. This can allow users to focus on the more intellectual aspects of picture-making. They can spend more time being creative and less time being technicians.

The computer's enormous capacity offers a would-be computer artist creative potential not provided by traditional media. In addition to considering rule systems for design, computer artists must plan for effective use of the unexpected. While they are using various transformations to pattern their elements they can combine these in unexpected ways. The unexpected, defined as the random choice provided by the computer through a random number generator, offers many interesting possibilities. This new creative freedom and the computer's phenomenal capacity for speed allows artists to generate many images. The need for a filter becomes even more crucial.

An Approach

Several scholars have focussed on standards for the computer graphics medium. Some have established categories for structural elements possible through programming techniques [1], [2], [3]. Others have attempted to correlate simple graphics with human evaluation [4]. A problem with these approaches, however, is that they fail to incorporate established principles of design. Aesthetics for a new medium should acknowledge cultural context. Rule systems must build upon those of past and current media. A better approach considers production technique and human response within the context of traditional design principles.

This paper documents three perspectives toward defining computer graphics for empirical testing. The first incorporates fully determined visual structures, fractal patterns, which are often used in computer image generation. The second considers the aesthetic potential of a crude form of artificial intelligence, where the computer structures images; and the third deals with traditional design principles as rules for structuring images.

The First Perspective

One way to define the structure of computer graphics is to use the deterministic algorithms that have produced the images. The simplest images obtained this way are line patterns -- sets of line segments whose endpoints are specified according to a set of rules. The most interesting class of line patterns, called fractal patterns or fractals, falls between the Euclidean definition of one and two dimensional objects. Some consider the design potential of fractal patterns as works of art, while others use fractals as primitives to create more elaborate images. Increasingly they are used to create naturalistic landscapes (river networks, cloud formations) and a range of organic forms (tree branching patterns) [5].

Fractals' relevance as visual primitives plus their precision and flexibility make them desirable as variables. In this perspective, various levels of two fractal patterns are tested for their relation to viewer preference.

Two types of patterns, each using a different fractal equation, were generated. For each type ten examples were produced in order to create a differential, a smooth range of stimuli that vary from a straight line to an extreme curve. Figure 1 illustrates one type, Koch curves, and Figure 2 is an example of the second type, dragon curves.

The algorithm used to produce the Koch curves is based on the regular subdivision of a triangle. The first level is the triangle itself. At the second level each side of the initial triangle is divided into thirds and the middle third is replaced with two sides of a new triangle, one-third the size of the original triangle. Subsequent levels are formed by following this rule and replacing every line segment of the previous level with four new segments, each one-third the length of the previous level.

The algorithm for the dragon curves utilizes the following recursive rule: a level 0 dragon is a straight line, a level 2 dragon is a right angle (two straight lines), a level N dragon is composed of a level N-1 dragons with different orientations.

Two variables were tested in the experiment: dimension and density. The angle of the triangle in Koch curves as well as the orientation of dragon curves were treated as dimension variable 0. In each case 0 values of 0, 20, 40, 60, 70, 75, 82, 86, 88, and 90 were used in order to produce a range of stimuli representing a differential scale with ten levels.

Three levels of density were represented through a different number of line segments within the images. The most simple (DENSITY 1) has 256; the mid-value (DENSITY 2) has 1024; and the most complex (DENSITY 3) has 4096 line segments.

Sixty slides were produced of all levels and densities for both types of fractals. All sixty slides were tested for viewer preference using an open-ended ratio scale. That is, subjects were asked to assign numeric values to indicate their preference for each pattern.

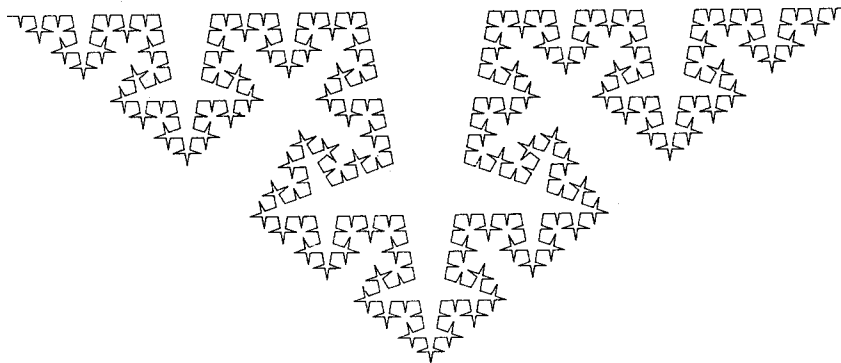


Figure 1: Example of a Koch Curve

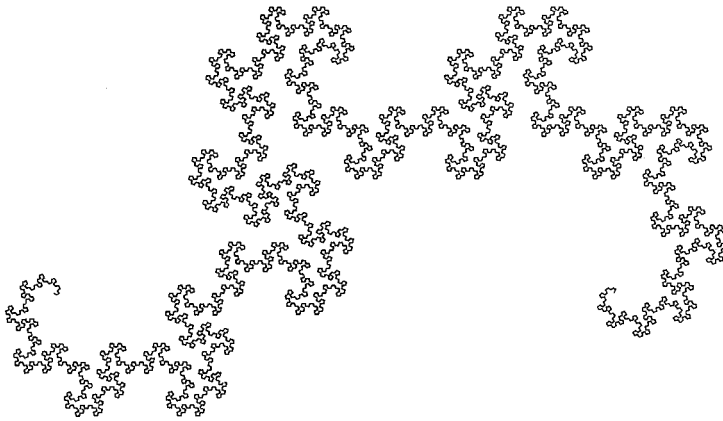


Figure 2: Example of a Dragon Curve

For some densities, a monotonic relationship with a global maximum was shown consistently. In general, subjects preferred the denser, more complex patterns. Figure 3 shows one example of the precise relationships between preference and dimension for all densities of the Koch curve.

It is curious to note that for both types of patterns subjects preferred dimension levels in the upper mid ranges. This suggests a consistent preference for 0 values between 86 and 88 degrees. Perhaps through testing different types of fractal patterns, this characteristic will emerge as one standard for an aesthetic filter.

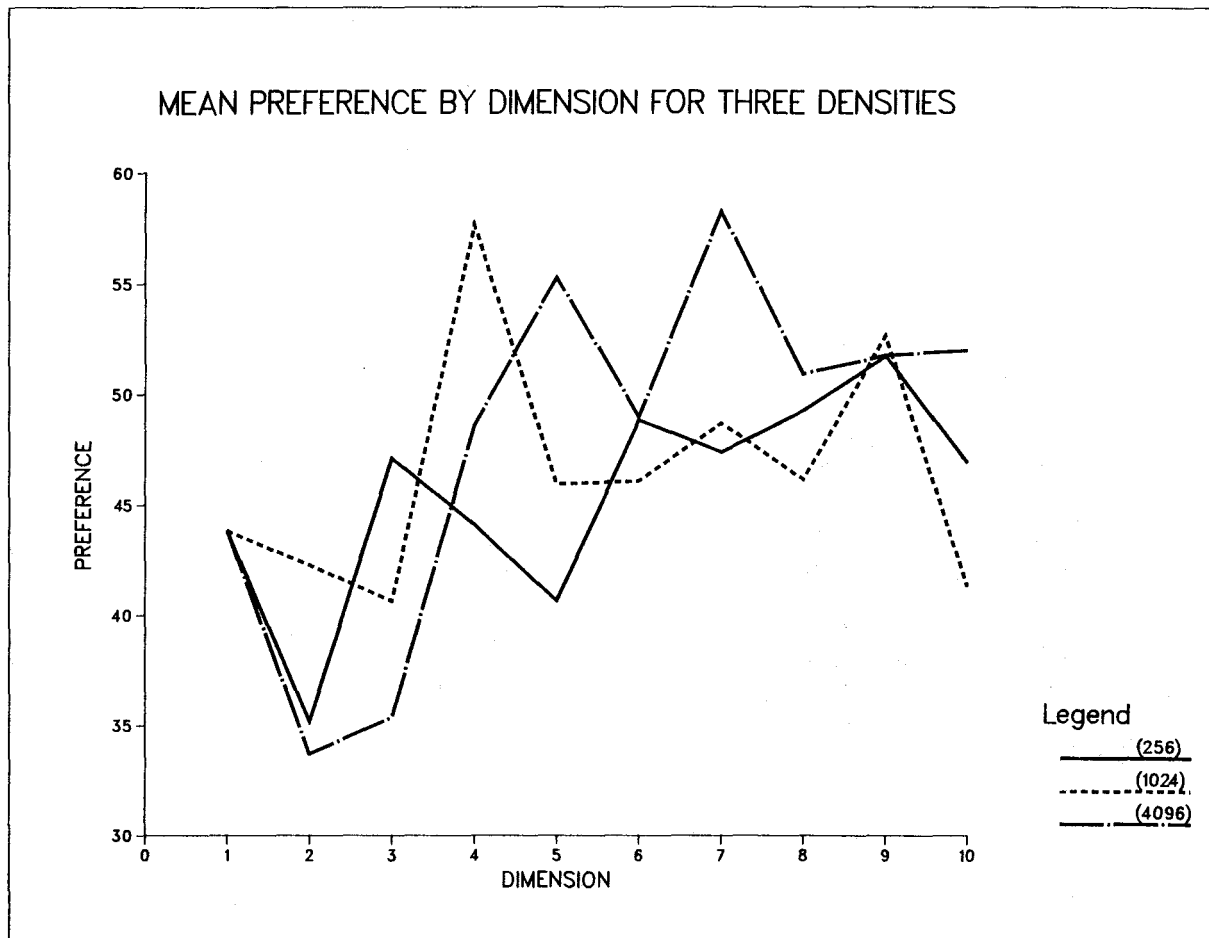


Figure 3: Relationship between Preference and Dimension of the Koch Curve

The Second Perspective

Since generation of computer graphics involves a compromise between structure (algorithms) and chance (random choice), a categorical scheme aimed at image evaluation can be drawn from specific levels of these two parameters. That is, all computer graphics can be classified according to their degree of structure and randomness.

In Arts and Computers [6], Holynski suggests five categories for classifying images. These categories vary according to the amount of structure imposed by the artist balanced by the unexpected offered by the computer. If considered as a formal dimension, these categories are levels which can be described precisely through ordered patterns of a basic element.

The full description of the the experiment illustrating this approach was published in [7]. Here, we present only its brief overview.

Each stimulus pattern is a grid of sixteen basic elements. The manner of determining the orientation of these elements can define the order level of each stimulus. Four orientations of the element are possible and for the least ordered type of pattern -- RANDOM PLACEMENT -- all orientations are equally probable for every element in the grid. Orientation is assigned through reference to a random number generator. An example of this category of stimulus is seen in Figure 4.

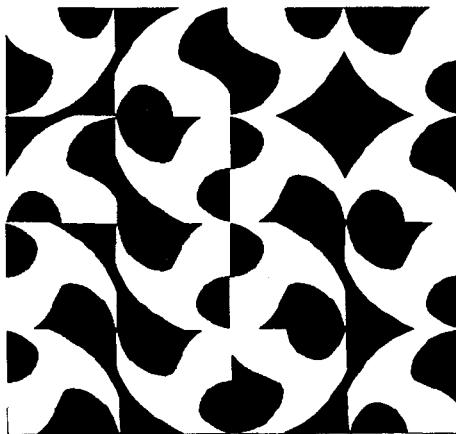


Figure 4: Example of a Random Placement Stimulus

To achieve the two more ordered types of patterns, element orientation was varied in a more systematic way. Both translation and reflection patterns begin with a four element quadrant where each element was randomly placed. Patterns representing the middle level for order can be produced by duplicating -- TRANSLATION -- the original quadrant to fill the remaining three. Figure 5 shows a typical translation pattern. The most ordered patterns were produced by reflecting the original quadrant about both inside edges and the central point. This creates a symmetrical -- REFLECTION -- pattern as shown by Figure 6.

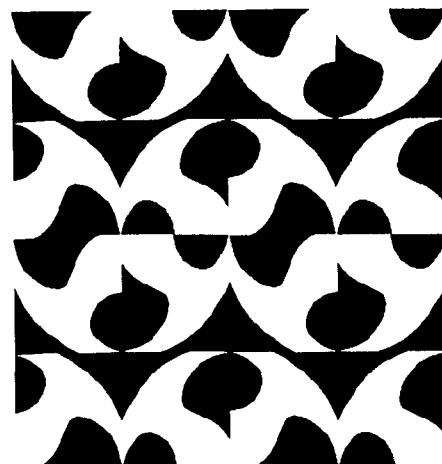


Figure 5: Example of a Translation Pattern

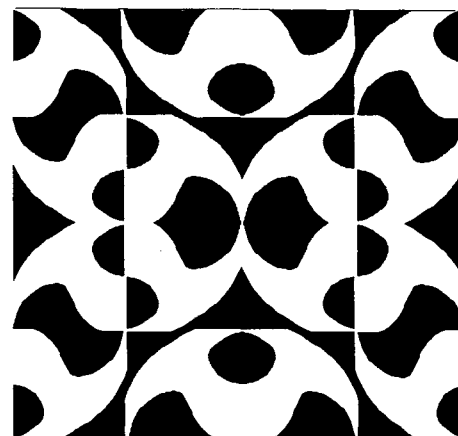


Figure 6: Example of a Reflection Pattern

Twenty-one slides were chosen to represent three levels of order (seven for reflection, eight for translation, and six for random placement). An additional dimension, color, was explored through three hues (nine white, six red and six blue).

The stimuli were rated with regard to preference by assigning a number between 1 and 10 for each pattern. Five represented an average amount of liking with higher numbers indicating greater preference. Mean (average) values were calculated for all preference ratings. When considered as a whole, the average preference rating was 4.829. When considered according to a particular slide's order value, some significant differences appear according to a stimulus' order. A summary table follows:

code	value label	mean	std dev
1	REFLECTION	5.579	0.795
2	TRANSLATION	4.881	0.639
3	RANDOM PLACEMENT	3.886	0.528

n = 44

As this table clearly shows, reflection patterns were most preferred. This suggests that viewers like more ordered patterns better than less ordered or random ones. These results, as expected, are consistent with previous findings [9].

The Third Perspective

This perspective defines three mathematically based variables that relate closely to Swiss Design Theory and grid systems, important concepts in graphic design [10], [12]. From this tradition springs the foundation for variables like complexity, order, and regularity as relevant dimensions of visual form.

These variables, COMPLEXITY (amount of information), REGULARITY (proportional consistency of information) and ORDER (pattern of information), have been empirically tested for their relation to viewer preference [7], [8], [9]. When represented through computer generated stimuli, all are strong predictors. Order is especially significant. In some cases, the structural variables defining a stimulus have predicted sixty percent of the variance in preference. Again, these dimensions of design may provide the basis for an aesthetic filter.

COMPLEXITY is defined as number of pixels per stimulus. Number of pixels varies from 2 to 99 per side of the matrix as complexity ranges from 0 to 16. REGULARITY, which also ranges from 0 to 16, corresponds to systematic variation in width and height where larger values indicate more extreme ratios. ORDER defines pattern variation within a stimulus. As order varies from 0 to 16, 0 indicates a systematic black and white checkerboard and 16 indicates a completely random distribution.

The stimuli for this experiment were forty-three slides which showed both a range of levels for each variable (a visual differential) and all possible combinations of three levels of the variables were produced. Fifteen of the 43 stimuli illustrated three visual differential scales -- simplicity/complexity, regularity/irregularity, and order/disorder -- each with five levels. While these scales were created, only one form variable was manipulated. The other two were held constant. Of the remaining stimuli, twenty-seven were produced to illustrate every combination of three levels for each form variable. Two of these patterns are illustrated in Figures 7 and 8.

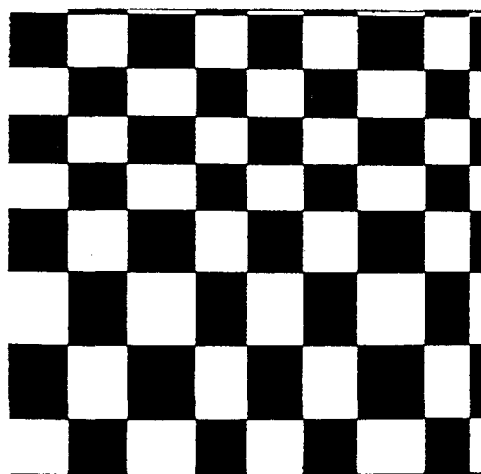


Figure 7: A Stimulus Pattern Showing Low Complexity and Low Irregularity

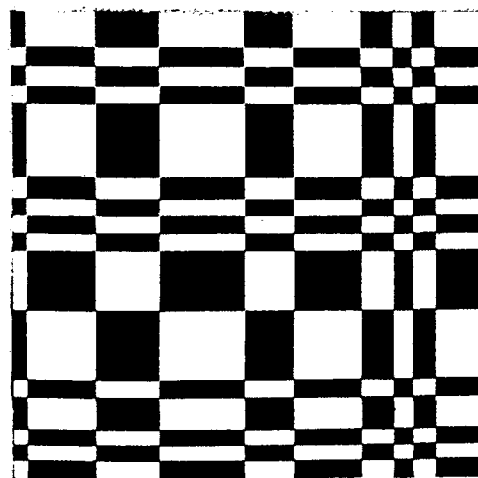


Figure 8: A Stimulus Pattern Showing Low Complexity and Medium Irregularity

The stimuli were rated using an open-ended, ratio level preference scale. When complexity, regularity, and order values were applied as predictors in a regression equation, complexity and regularity are highly significant. Order was not found to be a significant predictor. The summary table appears below:

variable	b	beta	f*
REGULARITY	0.6725061	0.46681	13.767
COMPLEXITY	0.7156979	0.36874	8.654
ORDER	-0.1597848	-0.10748	0.743

* significant at 0.01
n=46

where: beta, b - standardized and unstandardized regression coefficients
f - significance level

These findings imply that complexity and regularity are appropriate variables to consider for an aesthetic filter.

Discussion

By looking at variables for visual form, we can construct a rule system that will free artists from many of the more mundane aspects of production and will allow them to focus on making more intelligent design decisions. Ultimately this will facilitate more pleasing results. A side benefit will allow more economical use of graphic systems.

This paper presents three methods for quantifying visual form, techniques which relate both to image production methods and to the way that people process visual design. All of these measures have been shown to correlate with viewer preference in a variety of ways. Through more extensive examination of the relationship between form and preference, researchers can establish the foundation of theory necessary for an aesthetic filter.

As computer graphics are used more widely in conjunction with other media -- print, film, video and lasers -- or as an independent artform, the need for standards will become more obvious. In the meantime, more basic research is needed to define the direction of future studies. This paper presents a very fertile approach.

REFERENCES

- [1] Jim Fleming and William Frezza, "NAPLPS: A and Graphics, Part 1: Introduction, History, and Structure," BYTE, A New Standard for Text February, 1983, pp.203-254.
- [2] Bruce Olenchuk, "Graphics Standards", COMPUTER GRAPHICS WORLD, August 1983, pp. 56-60.
- [3] Chris Bailey, "Graphic Standards are Emerging Slowly but Surely", ELECTRONIC DESIGN, January 20, 1983, pp. 103-110.
- [4] Reiner Schneeberger, "Experimental Aesthetics with Computer Graphics: Analyses of Viewer's Impression of Computer Graphics", COMPUTER GRAPHICS AND ART, May 1978, pp 20-28.
- [5] Yoichiro Kawaguchi, "A Morphological Study of the Form of Nature", COMPUTER GRAPHICS, vol. 16, no. 3, 1982, pp. 223-232.
- [6] Marek Holynski, ART AND COMPUTERS, Wiedza Powszechna, Warsaw, 1976.
- [7] Marek Holynski and Elaine Lewis, "Experimental Visual Evaluation for Computer Graphics", IEEE PROCEEDINGS, 3rd SYMPOSIUM ON SMALL COMPUTERS IN THE ARTS, 1983, pp. 21-24.
- [8] Elaine Lewis, "An Effectiveness Measure for Visual Communication Media: Toward Definition of Visual Principles", Doctorial Dissertation, Department of Language, Literature and Communication, Rensselaer Polytechnic Institute, 1981.
- [9] Elaine Lewis and Brian Keith, "The Addition of Content and the Consistency of Preference Ratings for Visual Structures", paper presented at the 1983 annual meeting of the International Communication Association.
- [10] J. Muller-Brockman, GRID SYSTEMS IN GRAPHIC DESIGN, New York, Hastings House, 1981.
- [11] A. Hurlburt, THE GRID, New York, Van Nostrand, 1978.

Type Principles For Low Resolution Type

by Annette Wagner

Wagner Design
884 Pippin Avenue
Sunnyvale, CA 94087
408 749-0266

Apple Computer Inc.
10443 Bandley Dr. MS4-N
Cupertino, CA 94087
408 973-2946

Abstract: This paper is a review of the typefaces designed for Apple Computer, Inc.'s Lisa Office System microcomputer by the author. It covers the decisions made through first and second release of the fonts, explains basic type practices and shows how their application can improve the quality of the low resolution fonts in question.

INTRODUCTION

The typefaces discussed in this paper are those designed for Apple Computer's Lisa®. There were two major releases of the Office System software on the Lisa. The first release, June 1983, contained about 40 fonts. For the second release of the software, August 1984, the complete font library was revised and augmented by 20 fonts. This paper covers some of the design questions that arose in the first version of fonts, and how those issues were handled in the revision for second release to further improve their quality.

PROJECT CRITERIA

The Lisa font project started in September 1981 in the middle of the development cycle for the Lisa computer. The original project criteria was for a set of screen fonts to be based on the designs of Courier, Elite and Tile (Illustration 1), which are all daisy wheel type designs. A secondary model was Century Schoolbook for the serif fonts. The character set was the basic ASCII set, later extended to include european and math support.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz...

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz...

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz...

Illustration 1: From the top Courier, Elite, and Tile daisy wheel typefaces.

Modern 24 Point Modern 18 Point

Modern 12 Point
Modern 10 Pitch
Modern 12 Pitch
Gothic 15 Pitch

Classic 24 Point Classic 18 Point

Executive 12 Point
Courier 10 Pitch
Elite 12 Pitch

Modern 24 Point Modern 18 Point

Modern 14 Point
Modern 12 Point
Modern 12 Point 10 Pitch
Modern 10 Point 12 Pitch
Modern 8 Point 15 Pitch
Modern 8 Point 20 Pitch

Classic 24 Point Classic 18 Point

Classic 14 Point
Classic 12 Point
Classic 12 Point 10 Pitch
Classic 10 Point 12 Pitch

Illustration 2: Lisa 1 typefaces at the top and Lisa 2 typefaces at the bottom.

The range of fonts included a 12 pitch, 10 pitch and a typewriter proportional font matched to daisy wheels and 18 and 24 point display fonts for headline use. (see Illustration 2). The fonts were in two sets, serif and sans serif. Myself and another designer, Elizabeth Hall from Elizabeth Hall & Associates, were hired to design the fonts to this criteria.

There were two basic typeface designs used on the Lisa and a variety of type sizes from 8 point to 24 point. Illustration 2 shows the range of sizes, both first and second release. Classic is a serif font similar to Century Schoolbook and Courier. Modern is a sans serif similar to Helvetica and Univers. (There is a glossary at the end of the paper.)

The fonts on the Lisa are stored as bitmaps with width tables and other information. The fonts are both fixed pitch and proportionally spaced. A fixed pitch font is one in which each character is a fixed width; the number refers to the number of characters in a line of text an inch long. A proportional typewriter font is a fixed pitch font with one to five pre-set widths. This is different from typesetting fonts in which every character can be unique in width.

Fixed pitch fonts were carried over onto the computer because of the ease of coding and the mechanical limitations of early printers. One always knew how wide the letters were, no matter which letter one was typing. The use of these kinds of fonts is one of the reasons text on computers is considered to be of poor legibility and readability.

The marketing focus for the Lisa was to sell the machine into a business environment in which compatibility with a daisy wheel was essential. The Lisa marketing group developed this strategy based upon assessing the target user community. This compatibility with the daisy wheel printer influenced many characteristics of the first release font design.

WHAT YOU SEE IS WHAT YOU GET

The software support for the type work was performed by Jeff Parrish who developed the Lisa font editor. The editor allowed one to edit letters pixel by pixel on an enlarged grid (Illustration 3). Among other things, one could alter spacing around the character easily, lay characters over each other, and immediately see the results of any work.

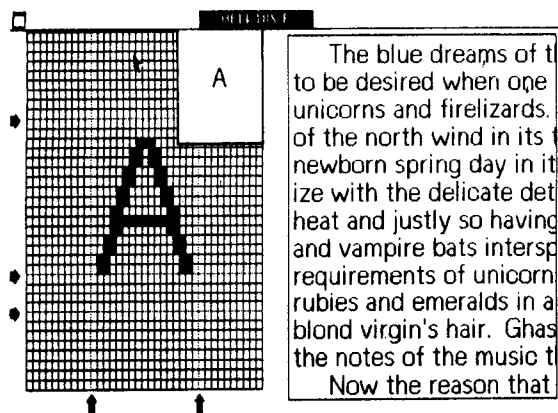


Illustration 3: Font editor screen with bit-editing grid and text panel.

As work started on the screen fonts, printing became a bit more of a reality. Owen Densmore started the print group and began looking at printers. I began designing a group of experimental fonts to put the printers through their paces. As we determined which printers we were going to use we started to develop the printing strategy. Since Lisa was to be a what-you-see-is-what-you-get (WYSIWYG) system, we had to make the printed image the same as the screen image.

We determined we had three devices to match to each other: a dot matrix printer (144 X 160 spots per inch), a daisy wheel (48 X 120 spi), and the Lisa screen (96 X 72 spi). The dot matrix printer also runs in low resolution mode using the screen fonts; this is not counted as a separate device because the fonts were primarily designed for the screen. At this point in the project, we began to design a separate set of fonts for each device. Each set was composed of two parts, serif and sans serif.

Character Width Matching: The different sets of fonts, one for each device, had to be matched to each other. This was accomplished through the use of conversion formulas which converted a number in one resolution to a corresponding number in another resolution. This is a simple task for a computer.

Width tables contained in the font format would be read in and converted using the computer program. For first release this was done with same size fonts only, i.e. an 18 point dot matrix font to an 18 point screen font. For second release fonts were adjusted for family continuity, discussed in Type Rules below, by converting from size to size, i.e. 24 point Modern to 18 point Modern to 14 point Modern. (This part of the matching was not necessary for the WYSIWYG system and was only used as a guide. Parameters in typefaces do not linearly change from large to small sizes; see Type Rules).

The numbers from the dot matrix printer do not convert evenly to numbers for the screen. Roundoff errors occurred all the time; common cases were 5.4, 9.6, 16.8, 10.2. In most cases we would round down. The cases we made exceptions for are uncooperative letters like M, m, w, and W. These letters were more legible with the extra pixels.

Although it is best to match from the top down in a WYSIWYG system, this was not possible with the daisy wheel fonts. For the 12 pitch, 10 pitch, and the 12 proportional we matched from the daisy wheel up to the screen and the dot matrix printer. For the 18 and 24 point fonts we matched from the dot matrix printer down to the screen and the daisy wheel. For second release we were able to add to the 12 proportional width tables and to do some adjusting which allowed a compromise with the daisy wheel for those particular fonts.

We developed eleven fonts per device all matched to each other. Once the bitmaps and matching were completed the fonts were compressed and built into a font library, a component of the Lisa Office System software. I designed the Modern fonts for all devices and the Classic fonts on the daisy wheel and Elizabeth Hall designed the Classic fonts for the screen and dot matrix printer.

EVALUATION OF FIRST RELEASE

After first release of the fonts was done I reviewed the issues that had come up during the development cycle. Some issues were use related, some software related, and some type related. One issue would lead to another. I will explain some examples of the first two here. The last set of issues, type related, will be explained in more detail in the Type Rules section.

Daisy Wheel Impact: The world still revolved around the daisy wheel printer and this caused several problems. One, the type model for the sans serif fonts was Tile and had never been designed as a display font. There were no versions of it in a display size, yet the Modern display fonts had to resemble the daisy wheels. The decision was made, following marketing guidance, to design new fonts based on the daisy wheel model. The fonts were released but were too bold and were a compromise between the typewriter and typesetting world which meant the quality was poor.

Second, since what-you-see-is-what-you-get, all the devices had to have display fonts (18 and 24 point), including the daisy wheel. These fonts were composed of the period on the daisy wheels. A daisy wheel has, as an artifact of the typewriter, an oversize period. The original resolution of the period was 48 X 60 spots per inch. The print group was able to double this resolution in the width direction, giving 48 X 120 spi. This meant the dots overlapped quite a bit in one direction and barely in the other direction. It is a monument to bit editing that these fonts look as good as they do. (See Illustration 4).

Common Usage: Output from the Lisa included the usual letters, memos, proposals and etc. One kind of usage led to re-thinking the way we designed the display fonts. That was the use of overhead transparencies and slides. Overhead transparencies need big type so that everyone in a room can see the information projected on a screen, yet the amount of information on one transparency can be quite large.

We had designed the display fonts on Lisa to have a tight character spacing, as is the current trend in the type field. But by the time the printed output from Lisa was copied, then made into a transparency, the transparency was two generations from the original and suffered image degradations from both the copier and the transparency maker. The image on the transparencies would invariably be blurred and all

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 WXYZ abcdefghijklmnopqrstuv
 wxyz 0123456789 &*!?=--+
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
 WXYZ abcdefghijklmnopqrstu
 vwxyz 0123456789 &*!?=--+
 ABCDEFGHIJKLMNOPQ
 RSTUVWXYZ abcdefghi
 jklmnopqrstuvwxyz
 0123456789 &*!?=--+
 ABCDEFGHIJKLMNOPQ
 RSTUVWXYZ abcdefghi
 jklmnopqrstuvwxyz
 0123456789 &*!?=--+

Illustration 4: Modern and Classic 18 point and Modern and Classic 24 point daisy wheel samples. (48 X 120 spi)

the characters would run into each other. This would be aggravated by bolding the characters. (Illustration 5).

Software Issues: The bold and italic versions of the typefaces on Lisa were done by algorithms in the low level graphics code called QuickDraw. The algorithms take numbers from the font and do calculations on them to modify the font's bitmap in some way.

The bold algorithm takes a bitmap and writes it once and then moves the bitmap over one pixel and rewrites it again. The resulting character is smeared to the right. This method is common on computers and barely acceptable with two adjustments. The amount to which the character is bolded should be commensurate with the size of the type, i.e. an eighteen point type should bold more than a ten point type. So that characters do not run into each other, the amount that a typeface is bolded should be added

FOR YOUR FOR YOUR FOR YOUR	FOR YOUR FOR YOUR FOR YOUR
----------------------------------	----------------------------------

Illustration 5: An example of blurring in a copy. Lisa 1 on the left, Lisa 2 on the right.

to the width of the character to compensate for the smearing that occurs. These two adjustments were made to the second release of fonts on Lisa (Illustration 6). The algorithm did not change; the numbers fed to it did. However, the effect of this change was to make bolded characters wider. One implication is that a fixed pitch font in bold and normal would no longer line up. Fonts that had to line up were handled specially, i.e., LisaTerminal and QuickPort fonts.

**FOR YOUR INFORMATION
FOR YOUR INFORMATION
FOR YOUR INFORMATION**

**FOR YOUR INFORMATION
FOR YOUR INFORMATION
FOR YOUR INFORMATION**

Illustration 6: Modern 18 pt. bold italic, bold and regular on Lisa 1 and Lisa 2.

The italic algorithm takes a bitmap and starting at a specified point on the baseline moves up and over to the right placing dots as it goes; up three, over one, etc. (Illustration 7). For a mathematical formula the algorithm is acceptable though it does leave some blank areas. The angle at which the letters are slanted is a bit steep at the smaller sizes. The major problem with it is the character is not centered in the bounding box when the algorithm is through. The letter hangs out of the bounding box to the right and not to the left. (See Illustration 8) This creates spacing problems when one word is italicized in a line of regular text.

Traditionally, italic fonts are separate designs. With the advent of digital typesetters regular fonts can be mechanically obliqued. Either way, the character is skewed from a point below the x-height line, about the center of the font (Illustration 9). This centers the letter in the bounding box with equal overhang on either side.

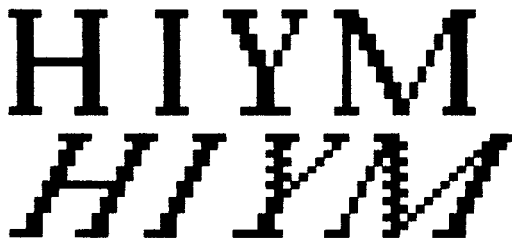


Illustration 7: Example of italic algorithm.

This is an example of an *italic* word in a line of normal text and **what** overhang is.

Illustration 8: Italic example with part of the text selected to show the overhang problem.

A way of fixing the italic algorithm to mimic the typesetting font, is to take the amount of pixels that the letter hangs over on the right, divide it in half, and start writing the character on the device to the left of the normal starting point by that amount. This change was not implemented on Lisa though it may still be at some future time.

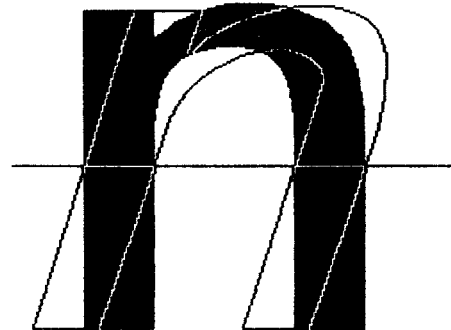


Illustration 9: Correct place to rotate a letter.

TYPE PRINCIPLES & SECOND RELEASE

When the Lisa fonts were designed the first time most attention was paid to the problems of making a WYSIWYG system work. As a result some of the fonts were not as successful as they could have been. In this section I mention some principles that we applied to the fonts as we revised them. The reason these practices were not emphasized in the first release was that, theoretically, they would not have had any impact on the fonts because of the low resolution of the devices. This was found to be untrue when those practices were actually put into use.

In Illustration 10 you can see defined the parts of a letter or letterform. It is the x-height that determines the size of the font. The larger the x-height the easier the font is to read. The x-height is 40 to 60% of the body size of a font. Most x-heights in the Lisa fonts range between 50 to 60% and are considered to be of medium to large x-heights.

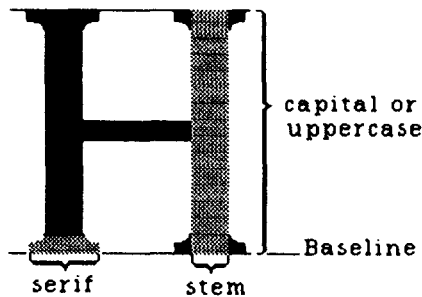
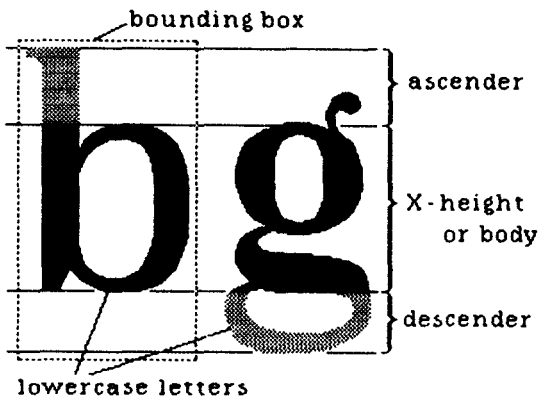


Illustration 10: The parts of a letter.

The relationship between ascenders, descenders, and the x-height changes in a typeface as the point size changes. Display fonts, 14 to 72 point, have a smaller x-height, larger ascenders and descenders and less letter spacing. Text fonts, 8 to 12 point, have smaller ascenders and descenders, a larger x-height and more letter spacing. This progression of change is not linear.

Historically typefaces were cut in metal and a separate font was cut for each size. The changes made accommodated the human visual system though it was done instinctively at that period of time. For a more complete explanation see Bigelow (1983).

The relationship between the vertical stem and the x-height determine the weight of the stem versus the x-height. The relationship is 5 to 6 stems per x-height. If the stem is one pixel wide, then the x-height should be five or six pixels tall. On the Lisa screen the fonts are single pixel stems up to 14 point and increase from there. The ratios, starting at 8 point run: 5:1, 5:1, 6:1, 7:1, 9:2, 13:3.

A family of type is a grouping of all the type sizes and variations of a particular typeface. Variations include roman, italic, light, condensed, bold, etc. (Illustration 12). Across a family of type, items like stroke weight, x-height, capital height, ascenders and descenders are harmonized so that any size and variation of the family is recognizable as a member of that family. A font is one member of a

family, i.e., Modern 12 point is a font in the family of Modern.

This idea of a family was implemented as the Lisa fonts were re-designed so that the various parameters of each font corresponded not only to its counterpart on a different device but also to its own family members. For example, the 24 point parameters for x-height, cap height, ascenders and descenders are the guide for the 18 and 14 point parameters. The way in which fonts change from text to display sizes was also taken into account. As mentioned before bold and italic are done by algorithms in QuickDraw, as well as other variations such as shadow and outline.

ITC Cushing Book	ITC Cushing Medium
WITH SMALL CAPS	<i>ITC Cushing Medium Italic</i>
<i>ITC Cushing Book Italic</i>	ITC Cushing Bold
ITC Cushing Heavy	<i>ITC Cushing Bold Italic</i>

Illustration 12: A type family.

Leading: The space between lines makes for easier readability. The longer the line, the more leading is necessary. Bodoni's rule: For good readability specify one third of the size of text type for leading. Illustration 13 shows normal, tight and loose leading. When leading is too tight or too loose the eye gets lost trying to read across a line of text. On the Lisa the leading is set in the font, but the applications can override it.

Word spacing is the white space between words. A guide to a determination of the correct amount is the width of the lowercase 'r' in the typeface in question. Too much word spacing creates 'rivers'; too little makes the eye work to distinguish the word.

A typeface with large x-height requires more leading than one with small x-height. A general guideline: Type with small x-height, no leading; normal x-height, 1 point; large x-height, 2 points.

A typeface with large x-height requires more leading than one with small x-height. A general guideline: Type with small x-height, no leading; normal x-height, 1 point; x-height, 2 points.

A typeface with large x-height requires more leading than one with small x-height. A general guideline: Type with small x-height, no leading; normal x-height, 1 point; large x-height, 2 points.

Illustration 13: The top paragraph shows normal leading, the middle paragraph tight leading and the paragraph loose leading.

Artifacts of the devices play havoc with this and other practices. The aspect ratio of the pixel on the Lisa screen is 2:3. The vertical dimension is thinner than the horizontal dimension. A supposedly even weight font on Lisa is not truly so. Dots on the screen are written with a writing beam that turns on and off very rapidly. The beam can overshoot causing dot erosion; a single pixel stroke is too light, but a double pixel stroke is too dark.

Physical factors of the devices affect the ability to which one can implement minor adjustments. An issue that came into play from the dot matrix printer was that of dot separation (Illustration 14). Two single stroke vertical bars have to be separated by a minimum of three pixels or they are not recognizable as two separate strokes.

An observation from some experiments I did shows that the more times a vertical line is broken down to form a curve the less distinct the form becomes and the shorter the viewing distance from which one can discern the form. This was determined at a 10 point font size on the Lisa screen (Illustration 15).



Illustration 14: Dot separation. This was printed on the dot matrix at 144 X 160 lines per inch. The lines are separated by 1, 2, 3, and 4 pixels from left to right.

abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz

Illustration 15: Screen resolution fonts printed on the dot matrix printer. The top line shows a rectangular letterform and the bottom shows a curved letterform.

Letter spacing is the white space around a character, sometimes referred to as inter character spacing. When spacing is done correctly the visual space between characters should optically look the same. This creates an even rhythm of dark and light which is very well suited to what our eye wants to see (Bigelow 1983).

To more easily determine the correct spacing we break the letterform into elements (Illustration 16) which are the vertical, horizontal, diagonal, and curve. There is a rule of typography based on these elements that explains how spacing works. It states as follows:

"Letter spacing and word spacing must be consistent. Once you establish whatever amount of letter spacing between two vertical strokes, or a vertical stroke and a slant, a slant and a curve, etc., thereafter whenever that same combination of strokes occurs, the spacing should be the same throughout. Same with word spacing. This rule should never be violated."

— From one of the typography manuals used in a class by Paul Sinn.



Illustration 16: The elements of a typeface.

The original method of spacing letters was to put the same amount of white space between each letter regardless of what kind of strokes the letter was composed of. If we look at the 24 point Modern font for the dot matrix printer, Illustration 17, the first line of text, 'HOHOHH', in the first example, shows all the characters as having 5 pixels between each regardless of stroke type. The spacing was uneven from the right to the left of the letter.

The next example shows three pixels around a curve and four around a vertical; this gives the 6-7-8 example. In the last example the vertical spacing was increased to five pixels on either side of a vertical giving the 6-8-10 example. At the same time the diagonals have been adjusted from three pixels on either side to one pixel on either side.

HOHOHHOOOHHH
HOHOHHOOOHHH
HOHOHHOOOHHH
HXHXXHHAHAHH
OXOXXOOAOAAOO

Illustration 17: The top example shows the characters with 5 pixels between characters. The middle example shows a variation of 6-7-8 in pixel spaces. And the bottom example shows the final spacing of 6-8-10 pixels.

In this way, the spacing for the curve/vertical, curve/curve, vertical/vertical, etc. pairs was produced. From these pairs the rest of the spacing could be determined. A rule of thumb in typography states that if the space between two verticals is X, then the space between a curve and a vertical is 1/2X. That gives a starting point from which the font can be visually adjusted.

Illustration 18 shows various characters and the amount of surrounding white space in pixels. Characters like R, L, E, S, and J went through several steps to determine the correct spacing for both sides of the character. The same basic process was applied to the lowercase letters with the difference being that the numbers were smaller and spacing was therefore tighter than the capitals.

HJHJJHHJOJOJJ
 HRHRRHHRORORR
 HSHSSHHSOSOSS
 HEHEEHHEOEEOEE
 HLHLLHHLLOLOLL
 noonnononno
 nanbncndnenfnngn
 oaobocodoeofogo

Illustration 18: The numbers indicate the spacing for J, R, L, E, and S.

Next the screen version of the 24 point font was matched to the widths of the dot matrix printer 24 point. This approach had the advantage that the spacing was more uniform. But as a result of going to the lower resolution the lowercase letter spacing became the same as the uppercase. The difference in pixels between different kinds of strokes continued to become much smaller; the size of a screen 12 point font yields a minimum of 2-4 pixels between strokes compared to 3 to 10 pixels in the 24 point dot matrix font.

This discussion has focused on the Modern font, which is sans serif. The serif font, Classic, was also adjusted with this method and Illustration 19 shows the change in Classic 24 point for the dot matrix printer. There was a wider range between verticals and curves than with the sans serif font. Dot separation has an impact here on how close the serifs can get to each other.

HOHOHHHOOOHHH
 HXHXHHHAHAHH
 OXOXXOOAOAAOO

HOHOHHHOOOHHH
 HXHXHHHAHAHH
 OXOXXOOAOAAOO

champions
 noonnononno
 nanbncndnenfnngn
 oaobocodoeofogo

Illustration 19: On top is the version one with 4 pixels between curves, 12 between verticals and 0 between serifs. The bottom is 6 between curves, 16 between verticals, and 4 between serifs.

Even with the limitations of the devices the spacing adjustments improved the quality of the typefaces. The space between characters becomes rhythmic and even, which is essential to the readability of the typeface. (This also clears up the copying and overhead transparency issues.)

CONCLUSION

This paper has shown how the fonts incorporated in the Lisa 7/7 software were improved by the application of a variety of type principles. Considering the typeface designs currently used on some microcomputers, these typefaces offer legibility and readability to the best of the device resolution. While low resolution type will never compare with graphic arts quality type, it should be obvious that type designed with these practices in mind is a definite step closer to that quality than fonts which ignore such rules.

The Lisa fonts were designed for their particular devices, but only to the extent that the physical factors that affected the type design were taken into account. They were modeled after typesetting designs to the point that one can recognize a resemblance to a sans serif Helvetica/Univers letterform.

When typefaces are designed for typesetting, practices are followed similar to the ones mentioned here, though typesetting machines have a minimum of 720 lines per inch resolution, and usually much higher. The subtleties of letterform design are therefore much more complex.

In translating type into the low resolutions of the computer, the trained type designer employs his knowledge as a filter to bring the elements of good type to a typeface design within the restrictions of the device. This means that not all the subtleties and details of a design may make it to the printer or screen, but that those which are necessary to make the design a good one will.

Bibliography:

Bigelow, Charles and Day, Donald, *Digital Typography*, Scientific American, August 1983, p. 106 - 119

Bigelow, Charles, *Principles of Structured Font Design for the Personal Workstation*, Byte Magazine, September 1984.

The text of this paper was printed on Imagewriter using 7/7 Office System software on a Lisa computer. The typeface is Modern 12 point at high resolution. The illustrations were done using a Lisa and a Macintosh computer.

Glossary

Ascender: The upper stroke in a lowercase letter: b, d, f, h, k, l, t.

Bitmap: The array of pixels that compose an image whether it be a letter, pattern or picture.

Bold: A typeface variation in which the weight of the character is heavier than normal. Used for emphasis.

Bounding Box: The smallest box within which a character image is contained.

Capital: The large letters in a typeface, also uppercase. Cap height is a measurement of the same.

Decipherable: Text which is read with difficulty, which must be deciphered.

Descender: The downward stroke in a lowercase letter: g, j, p, q, y.

Device: Something an image is output onto.

Display: Fonts from 14 point to 72 points of size.

Family: All the variations of a typeface i.e. regular, bold, italic, condensed, extended.

Font: A font is one variation of a family of type. A complete assortment of any one size and style of type.

Italic: A typeface in which all the characters are slanted to the right. Used for emphasis.

Leading: The space between lines of type. It is measured in points.

Legibility: The speed at which letters can be recognized. Text is read with minor difficulty.

Letter Spacing: The white space between letters.

Lowercase: The small letters in the alphabet.

Pitch: The numbers of characters per inch in a line of text across a page. Fixed pitch fonts have a fixed number of characters per inch.

Pixel: The smallest segment of a raster line that can be discretely controlled by the display system. Also bit. From picture element.

Point: A system of measuring type. There are 72 points to one inch.

Readability: The measure of ease of reading a printed page. Text is easy and pleasurable to read.

Resolution: The number of controllable pixels on a device. In this paper the measurement used is spots per inch.

Sans serif: A typeface with no serifs.

Serif: The short crossline at the end of the main stroke.

Stem: A straight vertical stroke or straight diagonal stroke in a letter with no vertical strokes.

Stroke: A straight or curved line.

Text size: Type from 6 point to 14 points in size.

Typewriter Proportional: On the daisy wheel printer a proportional wheel is one on which the characters have perhaps five pre-set widths to choose from.

Uppercase: Capital letters; it refers to the storage of the letters in hot metal times.

Weight: Refers to the heaviness of the stroke in a font. The stroke can be light, regular or bold.

Word Spacing: The white space between words.

WYSIWYG: What you see is what you get. A term describing visual fidelity between devices.

X-height: The body of the letter minus ascenders and descenders i.e. 'x'. The lowercase 'x' is used to measure this in points.

Lisa is a registered trademark of Apple Computer, Inc. Macintosh is a trademark licensed to Apple.

ITC is International Typeface Corporation, New York.

Microcomputer Applications for Graphic Design

Susan E. Metros

Department of Art
University of Tennessee/Knoxville

Abstract

This paper provides a broad overview of some of the microcomputer applications for the field of graphic design. The author's purpose is to explore applications beyond obvious production oriented tasks. She begins by comparing the past role of America's "Commercial Artist" to the "Designers" role today. In doing so, she demonstrates the need for computer assistance within the changing field of Graphic Design. The remainder of the article describes possible microcomputer applications for typesetting, animation, business graphics, page layout, photography, television, typography design, creative problem solving, education, communication, concept sketching, business services, and image making.

Background

American Graphic Design has undergone an evolution (1). From the turn of the century until the 1950's Graphic Design was the domain of technicians, aptly called Commercial Artists. A successful Commercial Artist was one who could take ideas provided by the client and render them in decorative and stylized ways. This is no longer the case. Today, a "Graphic Designer" plays a multifaceted role focusing on design as a process requiring multidisciplinary professional skills beyond the scope of artistic talent (2).

Presently, the graphic design profession is a primary contributor to modern visual culture. Graphic communication techniques are used for persuasive, informative, and educational purposes. The increased need for design and the complexity of new modes of communication demand new roles requiring new kinds of designers.

- Designers are now translators, interpreting written and verbal information into visual imagery.

- They are teamplayers and coaches working beside professionals like editors, typographers, stylists,

producers, suppliers, illustrators, photographers, printers, marketing statisticians, media buyers, computer consultants, and other specialized designers.

- Today, designers must be literate in areas related to sociology and psychology in order to analyse, predict and control behavioral responses to proposed messages.

- A designer must be well versed in principles governing business. Not only does the design studio have to project a professional attitude, but the designer must also possess the vocabulary to communicate with business clientele.

- Designers must be technically oriented so that they may perform, or direct, the production steps required in preparing a graphic communication piece for publication or broadcast.

- Designers are, of course, artists. An artist manipulates visual elements within a predefined visual field. In the case of graphic design, the image should be distinct, dynamic, mood provoking, and aesthetically pleasing.

- Lastly, the transition from the Commercial Artist of the 1950's to the "Designer of the 80's" means that a designer must possess a strong set of creative problem solving skills. The design process today is no longer solely dictated by the client. Today, a client hires a designer or design team with the expectation that they will carry a project through from concept to final piece. This responsibility includes: defining and researching the problem, creating criteria for a successful solution, generating options for solution, selecting the best of these options, producing the piece, and, oftentimes, even monitoring the project's success in the market.

Introduction

Computer technology is playing an important role in defining the scope of today's designer. A report titled, "The Market for Computer Graphics in the Graphic Arts Industry," published by Frost and Sullivan, Ltd., predicts that electronic paint systems will become universal graphic design tools in graphic design and video studios. The influx of the new technology will be so great that sales of this computerized equipment will top seven billion dollars by 1990 (3).

Print Magazine prefaced the results of a computer graphics usage survey of its readers by stating, "Computer graphics systems are forecast by industry experts as the highest growth segment of the computer industry for the next five to ten years" (4).

However, the computer industry has been shortsighted in limiting its applications for design to those performing production oriented tasks. Production is only a small segment of the total design process. A designer must integrate the social, economic, psychological, and physical attributes of a design problem into a successful and distinct solution. Therefore, if a computer is to be a worthy investment it should address the needs of both creative and practical tasks.

This paper will provide an overview of production applications along with other more creative applications for using computer technology to enhance the total design experience.

It should be noted that a design studio or educational institution need not invest thousands of dollars in order to be part of the "computer revolution." Most of the applications discussed in this paper can be accomplished with a personal computer and add on peripherals.

The system I am using is an Apple II Plus with 64K of memory, two disk drives, a dot matrix printer with a graphics interface, an RGB color monitor, a graphics tablet, a video digitizer, and a communications modem. This equipment costs approximately \$5,000.00. Software, supplies, books, etc. costs an additional \$1,500.00.

Applications

TYPESETTING INTERFACE

A designer is usually provided with written copy for a project. However, he or she is responsible for choosing and sizing a typestyle that will enhance the visual concept without compromising the informational content of the message. Traditionally, a professional typesetter sets the copy to the designer's specifications. Most typesetting is done on a specialized computer. Therefore, it is possible to use a communications modem to interface a microcomputer, located at a remote site, with typesetting equipment. This allows the designer, after entering the correct conversion table, to directly keyboard and specify the desired type and layout. The advantages of this system are a quicker turnover, less chance for errors, cost reduction, and a more personalized involvement with the actual product.

TYPOGRAPHY DESIGN

Many of the graphic packages on the market include a selection of type fonts that the user can manipulate in respect to position, size and weight. Occasionally, the designer will want to create a typeface for a specific use. Most type design routines provide the user with a magnified grid in which existing characters can be altered and new ones created from scratch. The resolution is not nearly good enough to be substituted for professional typesetting, but the characters can be used for rough concepts and do have a certain sensitivity in their own right.

PAGINATION SYSTEMS

One of the most tedious jobs the designer performs is "keylining" a piece. Keylining is the process in which the type and artwork are pasted to a board, colors are separated by "cutting" overlays, and instructions are provided for the printer. Large computer systems have been extremely beneficial in assisting the designer in completing this task. Time Magazine, for example, replaced X-acto knives and waxers with "Vista", a system designed by Scitex Corp. Ltd. This system is comprised of two screens. One of the monitors displays, in symbolic form, the designated story, headlines, art, and pictures. Time's designer can move the elements around and alter size and position. The second monitor shows, in full color, what the actual page will look like. When completed, the page can be outputted fully keylined (5).

Microcomputers are unable to provide the high quality of image required for publication. However, word-processing programs are available that have "cut and paste" routines which can combine type and images together on a single page. In addition, images, copy, grids for layouts, letterheads, etc. can be designed and saved, then retrieved when needed. This technology is useful in the concept stages of the project. In fact, some advertising agencies have been experimenting with "soft proofing" of ads in which they electronically transmit the rough concept to the client for viewing and critiquing (6).

BUSINESS GRAPHICS

One of the more popular applications of computer graphics is creating charts, graphs, diagrams, and information graphics for presentations, training programs, and other situations where complex data is better understood if it can be visually communicated. Many quality programs are on the market that can manipulate data into a variety of graphic formats. The problem with the microcomputer versions is the inferior quality of the resolution and output. However, if the need exists, business graphics can be composed on a micro and then be electronically sent to a professional computer slide production house for printing (7).

PHOTOGRAPHY

Glenn Rand, Director of the Lansing Community College Photographic Technology Program in Michigan, has written a unique program for the Apple that can be used by photographers "to add dimension to the black and white printing process" (8). Dr. Rand's program uses the CRT as a "very controllable" light source. The light pixels are individually programmed to imitate the procedure of "dodging and burning."

TELEVISION

Most of the computer graphics produced for television are created on large, powerful, and expensive systems. However, Chyron Corporation of New York recently introduced a low cost microcomputer peripheral capable of generating both video text and graphics. This system is not as sophisticated as its larger counterparts, but for a designer working with public television or a cable information network it provides good quality graphics for displaying titles and credits, and fulfills other broadcast needs (9).

ANIMATION TECHNIQUES

There are excellent software packages available for micros which provide the "tools" to create two-dimensional animation. "MovieMaker", by Eric S. Podietz, is a fine example of a easy to understand program that allows the user to develop, edit, and review animated sequences. The designer can apply this technology to planning storyboards or producing animatics for television commercials. A computer used for animation will save the designer time by actually drawing many of the frames and allowing for immediate changes.

SERVICE

One of the advantages to using a microcomputer instead of a stand-alone imaging system is that it fulfills the many design related applications previously discussed, and, also can be used to perform everyday office related functions. A wordprocessing program can be used to write letters, advertising copy, articles, briefs, etc. Databases can be tailored to generate class records, client profiles, mailing lists, and any other organizable information. Accounting packages can be used to keep financial records. A communication modem can connect the user to other computers, electronic information networks and services ranging from library circulation databases to letter quality printers.

A software package is available for IBMs and IBM compatible systems that manages the day-to-day functions of an advertising agency. Developed by Gabel Advertising, Inc. in Colorado, to handle their own business's chores, it provides specialized management tools for billing, cost accounting, general ledger entries, payroll planning, invoicing, and check printing.

COMMUNICATION SERVICE

A microcomputer fitted with a communication modem can send and receive information. This makes it an ideal vehicle for communicating timely, design related news and information between students and/or design practitioners.

Currently, I am working on a project to electronically publish a newsletter titled, "Design-on-Line." This publication would feature articles, employment opportunities, legislative news, grant listings, a date-line for conferences and other events, an interactive student message/bulletin board, resource lists, and other design related information. "Design-on-Line" will be shared with other machine compatible design programs and

studios. Eventually, it could grow to serve a larger audience and possibly be self-supporting from subscription and advertising revenue.

EDUCATION

Microcomputers, as an imaging tool, have already made an impact on design education. However, a computer in the classroom can also be used to perform other, diverse, design related tasks. "PFS," by Software Publishing Corporation, is a good example of a database management system that can be very useful in a learning environment. An educator can personalize a form for listing information like bibliographies, slides catalogues, historical data, designer biographies, paper specifications, market statistics, industrial product standards, etc. Once the form is composed a student can enter, access, rearrange, change, or delete information.

The advantages of such a system are the following: Information can be rapidly retrieved from various item headings. Data can be continually updated. Hard copy, even mailing labels, can be easily obtained. The computer provides the user with an interactive session which, in turn, implies a more interesting approach to learning. The information is available to anyone with access to the disk. It can even be sent over modem to other educators and design practitioners.

The computer can also be used, in a learning environment, to provide self-paced educational units on everything from how to perform specific design related tasks to how to use the computer. Students can use a system to review assignments, access hand-outs, or take tests.

Another intriguing use for such a system would be personalized electronic sketchbook/journals. Every student would be required to use a disk to keep notes and sketches. This disk would be a good way to document a term's work and research. In fact, existing databases could be accessed and information pertinent to particular problems and solutions could be saved unto the journal disk for future reference.

CREATIVE PROBLEM SOLVING

Many designers willingly accept the computerization of design production but balk at high-tech interference with the creative, conceptual phases of the design process. However, the microcomputer is an ideal tool for accessing, organizing, manipulating, visualizing, and evaluating information relative to a design problem.

This process, of logically exploring a problem by playfully coaxing the user into manipulating information in ways that new combinations and fresh ideas surface, is called Creative Problem Solving (10).

Last year, at the 3rd Symposium on Small Computers in the Arts, I presented "Electronic Thinking Cap," a paper describing my research on adapting a conventional creative problem solving model to microcomputers. This computer enhanced model consists of a foundation program and numerous application programs. The foundation program is a hierarchically structured, attribute storing, database in which the user can search for, add to, reorganize, and delete information. The application programs are, for the most part, "games" which allow the user to interact with the foundation program in order to generate a quantity of alternative solutions.

The advantages of microcomputer enhanced creative problem solving are numerous. Information can be systematically and rapidly accessed and processed. A quantity of options can be generated, manipulated, and visualized. One-to-one interaction is readily available. An objective appraisal can be used to determine a priority ranking of idea options. The design process can be documented through a history of the user's interaction with the model. Lastly, "Electronic Thinking Cap" can provide the designer with a non-stressful and, hopefully, enjoyable high-tech experience (11).

SKETCHING TOOL

The most creative phase of the design process is the concept stage. Traditionally, the designer uses colorful markers on tracing vellum to present his or her ideas to the client. These are the standard tools of the profession because they allow for subtle to significant reworking of the concept. A microcomputer, used as an electronic sketch pad, is an inviting alternative to markers and vellum. The computer can be programmed to perform an array of visually exciting manipulations once the image has been inputted. For example, using "The Complete Graphics System," written by Mark Pelczarski and David Lubar (Penguin Software), a designer can magnify portions of the image, choose from a palette of 100 colors, paint with over 90 different brush strokes, add text, create mirror images, move selected parts of the image, change colors, and rotate and shrink the image so that multiple views are shown simultaneously. Furthermore, the images can be saved to disk and then retrieved for viewing, reworking, or hardcopy

output.

Models for packaging graphics and other three-dimensional concepts such as exhibits, costumes, stage designs, product prototypes can be built, panel by panel, using the 3-D routines. Once all the points have been visually or mathematically entered, the image can be moved, sized, rotated, and even drawn upon.

IMAGING TOOL

Computer generated art has been slow to gain acceptance by the design community. Owning or buying time on large imaging system is expensive, yet, the smaller microcomputer systems cannot provide the standard of quality needed for professional work. However, the industry is beginning to respond to the needs of illustrators and designers who want to use the computer as a fine art tool but cannot afford a stand-alone imaging system.

Chartpak Company, producers of transfer lettering and graphic art products, recently introduced, "System 640," a graphic system compatible with the IBM PC. This system offers many of the same features of the sophisticated stand-alone systems, at about a tenth of the cost. The advantages of these kinds of "add-on" systems, other than their affordability, are accessibility, ease of operation and the versatility of the personal computer to also be used to perform numerous non-imaging tasks.

Conclusion

This paper has provided a generalized overview of some, but certainly not all, of the microcomputer applications for graphic design. Furthermore, the field is so dynamic, and growing so rapidly, that some of the information presented in this paper will, no doubt, be obsolete by the time this publication goes to press.

There are two important points to keep in mind when considering microcomputer applications for design. The first is that computer technology need not be financially prohibitive. The potential user needs to decide on the priorities for use and purchase a microcomputer system that best serves those needs. Most systems are modular in composition and peripherals can be added as finances permit and the need merits.

Secondly, there is no limit to the microcomputer's potential as a creative addition to a design environment. Massimo Vignelli, an internationally renowned designer stated, "The computer frees the designer to design" (12). If the designer

approaches the machine with an open mind then he or she can go beyond the preconceived notion that a computer is solely a production tool and use it to fulfill a wide range of diverse and creative tasks.

References

1. D. Fahnstrom, "Technology, A Necessity in Design Education," The Design Journal, Vol. 1, No. 2, 1983, p.11.
2. P. Meggs, A History of Graphic Design, New York: Van Nostrand Reinhold, 1983, pp.424-48, p.491.
3. "Artists Will be Among Beneficiaries as Computerized Equipment Purchases by Graphic Arts Industries Top \$7 Billion by 1990," Typeworld, Oct. 28, 1983, p.20.
4. J. Hoffman and R. Auster, "Results of Print's Computer Graphic Survey," Print, March/April 1984, pp.92-93.
5. J. A. Meyers, "A Letter from the Publisher," Time, April 4, 1983, p.1.
6. K. F. Schmidt, "Electronics in Printing," Print, Jan./Feb. 1983, pp.88-89.
7. T. Scannel, "Say it with Pictures," Inc., Sept. 1982, pp.130-132.
8. G. Rand, "The Computer in the Darkroom," unpublished, 1983.
9. K. Lainis, "Microprocessor-based Character Generation," Electronic Imaging, Oct. 1983, p.18.
10. S. E. Metros, "Electronic Thinking Cap," 3rd Symposium on Small Computers in the Arts, Silver Springs, MD: IEEE Computer Society Press, 1983, pp.30-31.
11. D. Koberg and J. Bagnall, The Universal Traveler, Los Altos, CA: Wm. Kaufmann, Inc., 1976.
12. M. Vignelli, Proceedings from ABI Viewpoints Conference, Toronto: Alliance Graphique Internationale, 1982.

3-D COMPUTER GENERATED ANIMATION

Judy E. Sachter

Computer Graphics Research Group
The Ohio State University

ABSTRACT

As computer graphics has become more sophisticated technologically, it has become an obvious tool for animation. While many 2-D and 3-D computer animation systems are based on traditional cel animation concepts and terms, there are a number of new concepts and design criteria the artist must consider when working in this new 3-D medium. Computer animation is a multi-stage process, or animation pipeline. The computer animation environment is described, and each stage in the pipeline is discussed, with special attention to data generation, data manipulation and motion control techniques.

1. Introduction

Animation is the graphic art which occurs in time and conveys complex information through a sequence of images as seen over time. The source of information for the viewer of animation is implicit in picture change; change in relative position, shape, and dynamics. Norman McLaren, a well-known Canadian animator, describes animation:

Animation is not the art of DRAWINGS-that-move, but the art of MOVEMENTS-that-are-drawn.

What happens between each frame is more important than what exists on each frame.

Animation is therefore the art of manipulating the invisible interstices that lie between frames. The interstices are the bones, flesh, and blood of the movie (and) what is on the frame (is) merely clothing [17].

This equilibrium between form and movement characterizes animation and makes it unique as a graphic medium. In animation anything can move in any desired way; this is what makes animation so exciting. The artist can create and bring life to new and imaginary worlds. In this respect animation is one of the most creatively free mediums. Ironically, animation which is so artistically free is also a time consuming, tedious, and expensive medium.

2. USES OF COMPUTERS IN ANIMATION

As computer graphics has become more sophisticated technologically and as computer availability has increased, it has become an obvious tool for animation. The computer can be used as an augments for conventional animation and as an image synthesizer in computer generated animation [14]. As an augments in traditional animation the computer is used to control the moves of the camera, cels and/or the animation bed of the animation stand. One advantage of the computer is its ability to deal with complex relationships precisely and rapidly. The computer is able to repeat mathematical operations over and over again either identically or with small incremental changes. It can be used to compute camera moves, as well as keep the detailed records for the exposure sheets that are necessary for animation organization and production [3].

Computer generated animation can include both 2-D animation and 3-D animation. The approach to 2-D computer animation is very similar to the process of traditional cel animation. The animator need only learn how to use the computer and the specific software available to create, store, and playback the animation [5].

3-D computer animation, in some respects, is similar to clay, puppet, or model animation. In 3-D animation the model, props, and backgrounds are built and/or painted, then the lights are set up to show the dramatic effect desired. A

motion picture camera with single frame capabilities is set up and one frame of film is exposed. The animator moves the model just a little and a frame is exposed. Repeating this process over and over again is the process of creating animation. As a result of the persistence of vision, when these single frames are projected at an appropriate rate they appear to blend into movement.

2.1. 3-D Computer Generated Animation

In 3-D computer generated animation a mathematical model of the characters, props, and background are created by the artist and stored in the computer memory. Many copies or instances of one model can be used. If one tree is digitally generated, it can be used to create a forest. For a scene, these models can be manipulated by changing their size, position, and orientation in space. The color and surface properties are assigned to the models. As with model animation the lights are set up to best show off the form and the action. The "camera" or eyepoint from which this scene is viewed can be placed anywhere and moved at will. The resulting image is rendered on a display device. Then the image can be stored in computer memory and/or recorded one frame at a time on video tape or film.

2.2. Preproduction Planning

The process of planning for an animation is essential. Animation is a visual art form, a form of communication. Therefore, the idea is the first and most essential step of creating an animation. It is what makes the animation work.

The storyboard is the conceptualization of the animation in a visual format. Animation is time consuming and expensive. Through the storyboard, scenes can be edited and rearranged prior to actual rendering, thus avoiding a great deal of time and expense. The storyboard contains the action, timing, sound, motion, transitions, information flow, rhythm, and character development, as well as the lighting specifications and the "camera" shots. There should be enough drawings, usually in comic book form, to get across the key ideas and to be able to clearly follow the action. Even in highly interactive computer animation systems preplanning and storyboarding is essential.

2.3. Movement and Timing

The animator's main concern in 3-D computer graphics is the choreography of objects, lights and the camera. The animator must communicate this information to the computer for each frame of the animation. These changes or transformations are controlled in various ways by the animator, depending on the system being used. Besides specifying the parameters of the objects and the view for a scene, the focus is on the changes of these parameter over time for the desired effect. The reader is referred to the excellent film on timing and motion in animation by Norman McLaren [18].

A common pattern of movement is to hold, accelerate to constant speed, decelerate, and hold. In traditional animation tedious mathematical calculations must be performed by the artist and written on the exposure sheet, and/or incorporated into the drawings or models in order to generate the proper motion for each frame. In computer animation, of course, we can program these mathematical functions so that the animator is free to concentrate on designing motion rather than calculating the numbers to enter on an exposure sheet. While the mathematical functions used in traditional animation can be applied in computer animation to move the models, lights and camera, the computer can keep track of many objects and changes simultaneously, so that the animator can control movement much more complex than in traditional animation. In addition, since the computer, not the animator, evaluates the mathematical functions that generate motion, it becomes practical to incorporate much more complex and realistic models of motion into animation sequences.

Since many 2-D and 3-D computer animation systems are based on traditional cel animation concepts and terms, the reader should become familiar with this field. Preston Blair [2], Frank Thomas [29], Brian Salt [28], Eadweard Muybridge [19,20], Eli Levitan [16], and Harold Whitaker [30] are good sources on traditional animation techniques and terminology, as well as motion and timing.

2.4. Time and Space: Computation Cost

An artist considers the cost and availability of materials and the quality of the workspace when designing a work of art. This is also true in computer graphics. Cost in computer graphics refers to two factors: computation time and memory space.

Modeling is often seen as a process of simplification; yet portraying the world in a manner that is believable to the viewer

requires detail. As a digital image grows more complex and detailed, the more expensive it will be to produce. Conversely, while a simpler image may be easier to produce, it will appear less realistic. Computational cost is related to a number of factors such as the complexity and number of objects in a scene, and the complexity of the light and shading models used. The artist must take these issues into consideration when designing an animated sequence.

The sophistication of the graphics software used to generate images can vary. Programs that try to closely represent the real world may be very complex and require extended computation time. Similarly, mathematical models that try to represent highly detailed objects will take longer and require more computer memory to calculate images. The time it takes to produce one image can vary from a few seconds to a few days!

Realism in 3-D computer graphics requires a fairly powerful computer, considerable mass storage, and a color display device with its own memory. The price of computers and memory has decreased substantially in the last five years, but they are still very expensive. Fortunately for the artist, a new generation of microcomputers is approaching the performance of the mid range computer. Some of these micros are now being tested and used in computer graphics labs. We can look forward to the day when substantial computing power for computer graphics is readily affordable.

3. Artistic Control of Computer Animation

In an analogy with conventional cinema, in computer graphics the animator is able to position and orient simulated light sources and a synthetic camera. Objects can be made larger or smaller or scaled. They can be turned in space or rotated. They can be moved around in space or translated. These manipulations are called transformations. Any parameter that can be controlled for an object can be changed over time during an animation sequence. For example, incrementally translating an object to another position in space over a series of frames will make it appear to move. Varying the distance between moves can either speed up or slow down the motion. If an object is rotated by .5 degrees per frame, it will turn 12 degrees a second, or revolve completely around twice in one minute of animation.

Computer graphics has some of the same problems as traditional animation. If the

size of the move is too great it will create the effect of a visual stutter. In computer graphics the simulation of motion blur is being explored to eliminate this problem [15,23].

Depending on the system, the animator may be able to define a path which an object will follow. A path allows the artist to design complex motion through mathematical procedures.

All of the above operations can also apply to objects that are attached hierarchically. A bicycle can be built out of separate parts. The wheels, the handle bars, pedals, seat, and a rider can be attached to the frame of the bicycle. As the bicycle frame is translated down the street all of the parts come along. The wheels and the pedals can be rotated at a speed appropriate for the distance of the translation, and at the same time the handle bars can be turned. This motion of articulated objects can become extremely complex.

Object attributes such as shininess, texture and color, transparency of an object can change from frame to frame. The background color can change slowly from day to night colors, or a dull opaque object may become shiny and transparent. Lights can also change dynamically. Their range of influence can be scaled incrementally, they can change colors, or even fly around influencing the lighting and color of other objects in the animation.

One feature unique to computer graphics is the ability to arbitrarily and weightlessly move the "camera". Views impossible in other medium can be achieved. The eyepoint and center of interest can be used in animation to simulate filming conventions or techniques, such as a cut, pan, dolly, truck or crane shots. Changing the view angle can simulate the changes of a zoom lens. For example, since the camera is always aimed at the center of interest, for a pan the "camera" is stationary and the the center of interest is moved. In a trucking shot the center of interest and the eyepoint follow parallel paths.

4. The Animation Environment and The Animation Pipeline

An animation environment has both hardware and software components. The hardware for the computing environment includes a computer or computers, terminals, printer, tape drive, and extra memory discs. System software for the computing environment includes an operating system, text editor, drivers for the hardware and

various programming languages. The graphics hardware includes display devices, such as a CRT, vector display or picture-system, a color raster display device, frame buffer, and various input devices such as a mouse, knobs, dials, joysticks, switches, buttons, bit pad and digitizing camera. In an animation environment it is also important to have a way of recording animation on film or video.

The graphics software necessary in an animation environment can be categorized by the purpose of the software: creating and editing data, scene description and motion control, display algorithms or rendering programs, post production (image processing) and a system for saving, filming, and/or recording frames.

Most environments are not made up of one large program. Instead systems are made up of several efficiently designed special purpose programs [7,9]. These programs carry out their designed task and send the results to the next program to implement its special task. This is called the animation pipeline, which looks something like:

```
data generation -> scene description ->
rendering -> opticals -> camera
```

4.1. Data Generation

Data generation software allows the artist to create the mathematical models that will be animated. Some of the methods for building data are projection, lofting, solid of revolution, and combinatorial [27,26].

The biggest concern in designing the data generation system is the ease of use for the artist, its efficiency, and immediate graphic feedback. The artists greatest concern is the ability to create the forms they want. There are some animation systems that combine the data generation with the animation system.

4.2. Data Manipulation

Artists want to be able to refine and adjust an existing object to achieve the best possible form. An interactive system that facilitates these adjustments is extremely helpful. To edit data by hand can be very tedious and time consuming, especially for large objects. For example, if a polygon is defined incorrectly, it may take a while to find out exactly which

polygon it is and correct the problem. To move a point the artist would have to edit the data file and change the values for that coordinate, display the object to see if the change was correct, and so on until he is satisfied. An object editor can allow the artist to change a polygon description, move points, or even add, delete, or assign color to polygons or vertices and immediately see the results [9]. The artist can interactively see the object while he adjusts, edits, and refines the form and may even undo some operations that do not look right [24]. In addition a variety of mathematical techniques are known for cutting, bending, and distorting objects, as well as generating fractals.

4.3. Surface Description

Surface description, such as color, shading model, texture, transparency, or light reflectance of objects are usually carried out in the display algorithm, but the artist has to specify these parameters. In many systems these parameters are designated in the animation script, but in other systems are part of the object description. They are mentioned here, because sometimes special purpose programs are used to create the desired effects. An important example is texture mapping. A texture pattern must be created to be mapped onto an object. This pattern can be created in several ways; through the use of a digitizing camera, through a paint program, a weaving program, procedurally generated, or a scene description created by the artist. This image or a section of this image must be mapped onto the object. Sometimes special software is useful for mapping a texture in the particular way an artist desires.

4.4. Scene Description and Animation

In a scene description the contents of a frame are described. An animation system allows the animator to describe a sequence of images. The types of animation systems currently in use are 3-D keyframe systems, animation languages, systems for controlling articulated figures, systems for simulation or a combination of these.

Interaction between the artist and the computer can be device-driven or language-driven. In a device-driven system the artist interacts with the computer through switches, knobs or dials, bit pad and receive immediate graphic feedback. Depending of the system, the artist can see these changes in real time. In a language-driven system the artist creates a textual script and then 'runs' the script to see the animation. Some systems allow

the artist to interact through either or both methods.

4.4.1. Keyframe Systems

A keyframe is similar to an extreme in traditional cel animation. In a keyframe system the animator specifies the scale, position, orientation, color of objects, and the view parameters for a frame. This becomes a keyframe. Then any desired changes are made in the objects or parts of the objects, the animator specifies how much time has elapsed since the last keyframe, and the new keyframe is set. The computer interpolates the specified number of in-between frames by computing the changes between keyframes. These inbetweens can be calculated linearly (uniform incremental changes) or eased motion can be generated. Some systems give the animator interactive control over the dynamics of motion.

A keyframe system developed at Pacific Data Images, Inc. (PDI) [6], allows the artist to interactively set up keyframes, run pencil tests at various levels and use the same script to fine tune the final animation sequence. This system combines the data generation and the animation into one system. Another example of keyframe systems is ANIMA II [13], a keyframe programming language which uses scripts and produces real-time color shaded animation.

Some 3-D systems use a multitrack or event driven approach, which allows the artist to set times and events independently for each object, rather than all objects in a scene for a keyframe. An example of a multitrack system is MUTAN (Multiple Track ANimator) [11]. This system allows the animator independent tracks for multiple characters, sound, and camera. MUTAN is based on the exposure sheet used in conventional cel animation. TWIXT [12] is an event driven system. Twixt allows the artist to interactively set up events and playback the action in a pencil test to see the motion. Where MUTAN has each object on an independent tracks, TWIXT has each parameter of an object on a separate track. For example, the color of the object could be eased-in while the translation and rotation is eased-out. Twixt also allows the construction of hierarchical relationships among objects, and a shape interpolation routine is built in as one of the tracks. One advantage of an event or multitrack animation systems over a keyframe systems is the ability to create the motion of each object independent of other objects in a scene. The sequences of events can be used to develop an animation script which can create either a realtime pencil test in wireframe or low resolution shaded images, as well as a final finished

animation.

4.4.2. Animation Languages

An animation language is a programming language for animation and graphics, which allows the animator to specify objects and their dynamics and surface qualities as well as camera dynamics. Graphics or animation languages, like other programming languages, allow for variables, iteration, incrementation and conditional statements for controlling animation. Some examples of animation languages are ANIMA II, ANTTTS (Animated Things through Time and Space) [8], and ASAS (Actor/Scripter Animation System) [25]. ANTTTS allows the animator the possibility of hierarchical structures and is able to run on a DEC PDP-11/23 micro-computer. ASAS combines the data generation with the programming language and incorporates advanced programming techniques. ASAS also allows the animator to set up hierarchical structures.

4.4.3. Control of Articulated Motion

Animation of articulated figures is difficult. Even in conventional cel animation, rotoscoping was often used to trace live action film of an actor onto the cels to better portray the motion. Sometimes articulated models were used as a guide to aid the animator [29]. An animation systems focusing directly on this problem of complex motion is BBOP [31], developed at New York Institute of Technology (NYIT). In this system the animator organizes the structure of the articulated figure in a short script. Then the animator interacts with this figure through dials, bit pad, and joystick to control the joint action of the figures. The animator has immediate graphic feedback. The camera may be placed and a sequence of motion designed and played back to test the motion. The animator may specify the motion through graphic means by adjusting a graph for the timing. GRAMPS (GRAphics for the Multi-Picture System) [22], designed for use for scientific research in molecular models and chemical research, deals with relationships of very complex data structures.

4.4.4. Simulation

A simulation is a simplified model of a phenomenon or action. Flight simulators have been used for many years to train pilots. The pilot interacts with the computer through all the normal flight control devices available in a cockpit. These analogue devices control the motion for the real-time animation which is displayed on a TV-screen "windshield". This simulation is augmented by actual pitch and roll of the cockpit for kinesthetic feedback.

Simulation of complex human motion is being studied by Norman Badler [1], David Zeltzer [33], and several others. sa, the skeleton animation system [32]. SA is designed for the artist to interact with the system at several levels. All movements can be set up in an interactive mode, or "skills", such as walking, can be stored and used several times. Since walking and running are cyclical procedures, these can be used and re-used with only slight adjustments to change the quality of the stride. Principles from robotics and artificial intelligence are important parts of such systems.

4.5. Rendering and Display Software

The display algorithm can be implemented in software or hardware. The majority of animation labs and production houses develop their own display software, using a variety of rendering algorithms. Techniques have been developed to remove hidden-surfaces from an image, and for raster displays, to compute surface shading and texture, all without introducing the aliasing artifacts associated with digital imagery. Research is continuing on efficient techniques for generating synthetic images that are nearly indistinguishable from photographic images. In general, mathematical proficiency and programming skills are not required to use such systems. But as with any complicated tool, the user who understands the basic concepts can make the best use of the capabilities of that tool. Foley and Van Dam [10] and Newman and Sproull [21] are comprehensive texts on computer graphics. See [27] for a non-technical introduction to the field of 3-D computer graphics.

In an animation environment the time it takes to generate the final image is important. Some systems with the appropriate hardware allow the artist to view animation sequences in real time. This facility is extremely helpful in testing and debugging the movement and timing of an animation sequence. High resolution shaded graphics are currently too slow to give the animator this necessary feedback until after much time has been spent calculating frames and recording the animation on film or video tape. For this reason many systems allow the animator to choose the appropriate viewing mode for a particular stage of the animation. For example, a pencil test can be displayed in wireframe mode to get a rough idea of a sequence in the early design stages. Next a low resolution, shaded pencil test can be computed to check the color and placement of objects (which may be ambiguous in wireframe views). Lastly, expensive, high resolution

frames are generated. This hierarchy of viewing modes makes the most efficient use of the system. There is no reason to spend days or weeks calculating high quality images until the animator is confident the sequence is correct.

4.6. Post Production

Optical or special effects can be performed electronically. It takes special programs to do fades, wipes, dissolves, mattes, titles, compositing and other image processing. Sometimes 3-D animation is calculated, saved and then used later as 2-D in the same manner as cels. This process is referred to as 2 1/2-D.

4.7. Recording

Animation can be recorded on either film or video. Software and hardware are necessary for this process. An animation camera (16mm, 35mm) must be interfaced with and driven by the computer. Video equipment for recording one frame at a time must also be interfaced and driven by the computer. Equipment for synchronizing sound with the animation may also be available. A number of problems can arise due to the limitations of various imaging devices (see Catmull) [4].

One problem in computer animation is the transfer to hard copy. Transfer to another medium usually causes problems and also moves the product another generation away from the original. In the transfer of an image to film there can be a loss of resolution. If the monitor resolution is less than the resolution of the film then the individual raster lines will be seen on the film. There is also a loss of color quality. This is similar to taking a color print of a stained glass window with the light shining through it; the color loses some of the power of the luminance in the print. If the image is taken by setting a camera in front of the buffer the camera is shooting through both the curved surface of the face of the monitor and the lens, which can reduce the quality of the image. Hardware developed primarily for the purpose of this transfer to film, use a flat screen and usually do three separate passes (either red, green, blue, or cyan, magenta, yellow) which will yield higher quality results. There may be problems with the exposure in areas of high contrast because the exposure has to be longer than the refresh rate. If the final product happens to be a film transfer to video the colors, format, and resolution will change.

If the output is directly to video, the problem is transferring the RGB digital

image to NTSC analogue. Some of the resolution is lost if the video is not a one inch format.

The final form should be known prior to calculating an animation due to the different format size. The edges of the image seen on the buffer may be lost when transferred to video. This video safe area has to be kept in mind while working on the animation.

5. Artistic Concerns

The field of computer animation is still new and there are many problems to still be worked out, but the advantages are great. Once a model has been created it can be used and re-used with many variations. If a scene is not correct the artist does not have to redraw a sequence, but must make some adjustments in the script and regenerate the frames. There is a freedom of form, camera movement, and animation possibilities that no other media provides. Once the artist/ animator moves into this realm of 3-D computer animation, the artistic and aesthetic potential can be realized.

1. Badler, Norman I. "Design of a Human Movement Representation Incorporating Dynamics". Course Notes, Seminar on Three-Dimensional Computer Animation, (July 27, 1982)ACM SIGGRAPH 82.
2. Blair, P. Animation. Walter Foster Art Books, Tustin, CA ().
3. Catmull, E. "The Problems of Computer-Assisted Animation". Computer Graphics, 12, (3) (July 1978)Proc. ACM SIGGRAPH 78.
4. Catmull, E. "A Tutorial on Compensation Tables". Computer Graphics, 13, (3) (August 1979), pp. 1-7. Proc. ACM SIGGRAPH 79.
5. Catmull, E. "Computer Animation State of the Art". Course Notes, Seminar on Two-Dimensional Computer Animation, (July 27, 1982)ACM SIGGRAPH 82.
6. Chuang, Richard and Entis, Glenn "3-D Shaded Computer Animation -- Step-by-Step". IEEE Computer Graphics and Applications, 3, (9) (Dec 1983), pp. 18-25.
7. Crow, F. C. "A More Flexible Image Generation Environment". Computer Graphics, 16, (3) (July 1982), pp. 9-18. Proc. ACM SIGGRAPH 82.
8. Csuri, C., Hackathorn, R., Parent, R., Carlson, W., and Howard, M. "Towards an Interactive High Visual Complexity Animation System". Computer Graphics, 13, (2) (August 1979)Proc. ACM SIGGRAPH 79.
9. Csuri, C., Gomez, J., MacDougall, P., and Zeltzer, D. Beyond the Storyboard: A Tool Set for 3-D Computer Graphics January 1984. Computer Graphics Research Group, unpublished
10. Foley, J. D. and Dam, A. Van Fundamentals of Interactive Computer Graphics. Addison-Wesley, (1982).
11. Fortin, Denis, Lamy, Jean-Francois, and Thalman, Daniel "A Multiple Track Animator System for Motion Synchronization". Proc. ACM SIGGRAPH/SIGART Workshop on Motion, (April 1983), pp. 180-186.
12. Gomez, J. E. "Twixt: A 3-D Animation System". Proc. Eurographics '84, (September 1984)To Appear.
13. Hackathorn, Ronald J. "ANIMA II: A 3-D Color Animation System". Computer Graphics, 11, (2) (Summer 1977)Proc. ACM SIGGRAPH 77.
14. Halas, John Computer Animation. Visual Communications Book, New York (1974).
15. Korein, Jonathan and Badler, N. I. "Temporal Anti-Aliasing in Computer Generated Animation". Computer Graphics, 17, (3) (July 1983)Proc. ACM SIGGRAPH 83.
16. Levitan, Eli Electronic Image Techniques. Van Nostrand Reinhold, New York (1977).
17. McLaren, Norman EXPO 1968, Montreal Canada (). Quote from animation exhibit Cinematique Pavilion.
18. McLaren, Norman and Munro, Grant Animated Motion Parts 1-5. Nation Film Board of Canada, (1979).
19. Muybridge, E. The Human Figure in Motion. Dover, New York (1955).
20. Muybridge, E. Animals in Motion. Dover, New York (1957).
21. Newman, W. and Sproull, R. Principles of Interactive Computer Graphics, 2nd Edition. McGraw-Hill, New York (1979).
22. O'Donnel, T.J. and Olson, Arthur J. "GRAMPS -- A Graphics Language

Interpreter for Real-Time, Interactive, Three-Dimensional Picture Editing and Animation". Computer Graphics, 15, (3) (August 1981)Proc. ACM SIGGRAPH 81.

23. Potmesil, M. and Chakravarty, I. "Modelling Motion Blur in Computer Generated Images". Computer Graphics, 17, (3) (July 1983)Proc. ACM SIGGRAPH 83.
24. Ressler, S. P. "An Object Editor for a Real Time Animation Processor". Proc. Graphics Interface '82, (May 1982), pp. 221-225.
25. Reynolds, C. W. "Computer Animation with Scripts and Actors". Computer Graphics, 16, (3) (July 1982), pp. 289-296. Proc. ACM SIGGRAPH 81.
26. Sachter, Judy E. "3-D Computer Graphics for Artists". Proc. 3rd Symposium on Small Computers in the Arts, IEEE, (October 1983)
27. Sachter, Judy E., "The Basic Concepts of Three-Dimensional Computer Graphics for Artists," Masters Thesis , The Ohio State University , Columbus, Ohio (June 1984).
28. Salt, B. Movements in Animation. Permagon Press, Oxford (1976).
29. Thomas, F. and Johnston, O. Disney Animation: The Illusion of Life. Abbeville Press, New York (1981).
30. Whitaker, Harold and Halas, John Timing for Animation. Focal Press Limited, London (1981).
31. Williams, L. "BBOP". Course Notes, Seminar on Three-Dimensional Computer Animation, (July 27, 1982)ACM SIGGRAPH 82.
32. Zeltzer, David "Issues in 3-D Computer Character Animation". Course Notes, Introduction to Computer Animation, (July 24, 1984)ACM SIGGRAPH 84, to appear.
33. Zeltzer, D. and Csurí, C. "Goal-Directed Movement Simulation". Proc. Conf. Canadian Society for Man-Machine Interaction, (June 1981), pp. 271-279.

A LOGO-BASED CAI PROGRAM FOR PRODUCING REPRESENTATIONAL ART

James J. McGinnis

Temple University

ABSTRACT

A LOGO-based CAI program was developed to facilitate the drawing of realistic scenes on an Apple II computer. The program generates objects that children can use to construct pictures. No two objects are ever drawn exactly alike. A demonstration program that automatically generates entire scenes was also developed.

THE LOGO LANGUAGE

LOGO was developed at MIT over a period of 15 years. A major goal of the LOGO language is to transform the computer into an object to think with.³

LOGO is an ideal language for children because it has relatively few syntax constraints. In addition, most commands can be abbreviated which results in a command set that places minimal typing demands on new users. The simplicity of the language allows children to concentrate on solving problems without frequent digressions to correct typing and syntax errors.

Pictures are drawn on the screen by commanding a "turtle" that resides there to move in the desired direction. When the turtle moves it can, at the user's discretion, leave a trail.

LOGO AND REPRESENTATIONAL ART

In spite of the simplicity of the language, children often have difficulty creating realistic scenes. Young children tend to lose track of the pen status when moving the turtle. They draw lines that were not intended or they accidentally omit lines.

With this in mind, I created a CAI program that facilitates the creation of realistic scenes. The program consists of a set of subroutines that draw scenic

elements such as clouds, mountains, rivers, trees, and houses. The subroutines perform several functions in addition to drawing the object. First, they select the color to be used. Second, they keep track of the pen status. The pen is always lifted at the end of a subroutine so that the child can position the turtle for the next object without leaving a line. The program allows children to use a building block approach to construct realistic scenes while allowing them the flexibility to add their own objects.

The subroutines that draw houses and trees require the child to specify a size. That requirement provides children with a straightforward introduction to the concept of a variable.

The tree subroutine uses a standard binary tree algorithm to draw the branches.¹ The leaves are then placed on the branches using an algorithm that randomizes their location within an area that is determined by the tree's size.

The painting program has been used in several experimental courses that were taught at Temple University. Children between the ages of 7 and 15 used the program to create pictures. It was found that children in these age groups had no difficulty using the subroutines described above.

A LOGO-BASED DEMONSTRATION PROGRAM

An additional program was created for demonstration purposes. The program runs all the subroutines while supplying appropriate constraints on random variables. For example, the program will not allow the base of a tree to be drawn above the horizon. There are no user specified options. The demonstration program takes approximately 4 minutes to run and it never generates the same picture twice.

The program makes use of a monocular cue for spatial perception called interposition. An object that partially obscures another object is perceived as being closer than the obscured object. Clouds are drawn first. Mountains may partially obscure them when drawn afterward. Trees are drawn last so that they will partially obscure mountains and clouds.

The program uses two methods to produce a pseudo-perspective. The size of the random elements that are incorporated into a river is increased as the height on the screen decreases. The same strategy is used to vary the size of trees. Figure 1

shows four pictures that were automatically generated by the demonstration program.

REFERENCES

- [1] Abelson, H. Logo for the apple II. Peterborough, NH.: BYTE/McGraw-Hill, 1982.
- [2] Abelson, H. and Klotz, L. Logo for the apple II: technical manual. Cambridge, Mass.: Massachusetts Institute of Technology, 1982.
- [3] Papert, S. Mindstorms. New York: Basic Books, 1980.

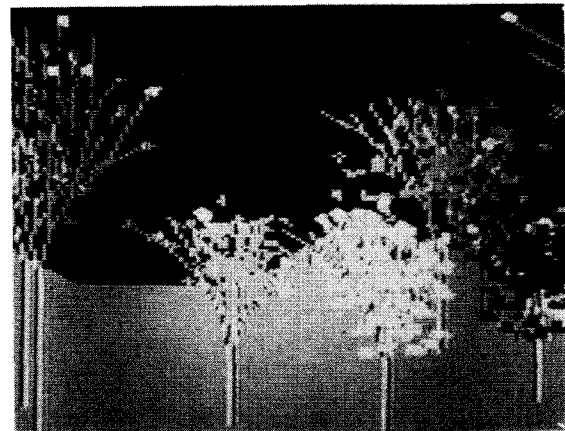
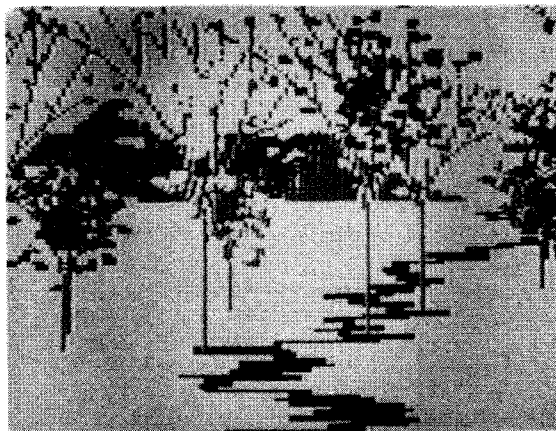
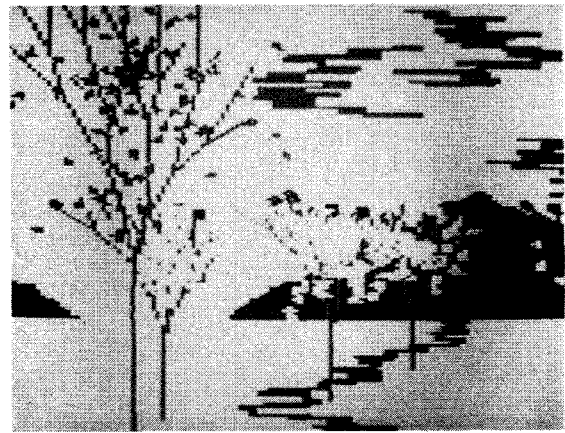


Figure 1. Four Pictures Generated by the CAI Demonstration Program

THE NEW STUDIO: The Computer Graphics Workstation

Alyce Kaprow, Joan K. Shafran
(c) 1984

Consulting Research
Boston, MA

ABSTRACT: This paper will examine computer graphic workstations for artists and designers. It is important to understand the many components of a workstation, what one should "look for" when investing in one and how to evaluate what your goals and needs are when assessing these computer graphic systems.

I. INTRODUCTION

A computer graphics workstation is the artists/designers studio or the equipment that constitutes the user's workplace for the creation of visual and verbal material. It is the single user system, consisting of input, display, and output devices. It can be purchased either "off the shelf" as a complete package, or can be custom made from various components.

In this article we will examine those workstations that are "off the shelf", designed for artists and other users who typically use 2-D design, illustration, and presentation graphics as opposed to true 3-D graphics.*

Before making an investment in a computer graphics workstation, it is important to understand the basic types, the relative costs, what is included (hardware, software), what the state-of-the-market is for computer graphics workstations, and what you need to know about the industry in order to make an intelligent choice. The following pages will discuss those issues and describe the generic types that most designers working now in traditional studios will be using.

II. WHAT IS A DESIGNER'S WORKSTATION?

Although computer graphics workstations are typically made up of similar parts, there are many inherent differences. It is not the individual components that define these differences, but the sum of the parts, or how they are put together to create the total system. When looking at the brochures for many of these systems, they appear to contain identical hardware; but, because of the design of the software, the programming that directs the hardware in its operation, they will work as completely different systems and do completely different things. When looking at computer graphics systems it is essential to understand the basics of hardware, software and

interactive devices.

HARDWARE DEVICES: They are defined as various components of a workstation such as the input, interactive, and output devices and a means of storing information and images. The computer, often referred to as the CPU or central processing unit, can be any size from micro to super. The amount and type of tools, as well as the speed and sophistication, are dependent upon the size of the computer. The ability of one machine to perform operations better and faster is often due to the size of the base system for which it is designed. The structure of the hardware will often allow the use of some tools, but will make it impractical or impossible to use others.

Most state-of-the-market workstations available today operate with computers of approximately the same power and similar characteristics. They are mainly micro systems, which are dedicated to the operations needed for particular design, illustration, animation, and page layout tasks.

The parts of the machine that allow the artist/designer to maintain control and get the job done are the *INTERFACES*. There are three basic divisions, *INPUT DEVICES*, *INTERACTIVE DEVICES*, and *OUTPUT DEVICES*. These interfaces and how well they function perhaps are of the most importance for the designer using a graphics system.

INPUT DEVICES are those which allow the user to place into the computer the initial images, text, or other data that will be elements of the final piece. These can be photographic images, live video, points digitized off a tablet, using a puck, a stylus, a mouse or lightpen, or numbers input through the keyboard. All these inputs become numerical data that will define the drawing.

The two most common input devices are the video camera and the digitizing tablet. The video camera allows the user to scan in any dimensional image or object, or to "grab" a single frame of live video into the computer memory. The original can be either black and white or color. Some machines will accommodate full color, some limited color, and others only black and white within the computer. This capability is usually price dependent. The camera itself is often an ordinary video camera that is linked to the computer through hardware and software, which allows the scanned-in image to be digitized, or read by the computer.

It is a frontend device, controlling the conversion of the image into digital data which the computer can understand. The user need only be concerned with the quality of the lens and the camera, just as in a photographic system, for the image quality is directly dependent upon how good the lens is. Time is also of importance and the user should be aware of how long it takes for the image to be digitized into the computer system. The time varies from machine to machine, and ranges from 1/30th of a second to many minutes.

The tablet, mouse and lightpen are devices that allow the user to draw into the computer, very much like drawing upon a blank sheet of paper. The tablet contains a grid of wires that senses the location on the drawing device and sends a message of the point location to the computer. Depending upon the routine or the process being executed, the location is interpreted and allows the user to paint, draw or edit. The mouse usually operates without a tablet, directly on the surface of a table. The lightpen, another input device, "talks" directly to the computer via light signals. This is used directly on the display screen.

The keyboard, is a keyboard, is a keyboard. The information is put into the computer by the user in the form of alphanumeric or by special function keys designed for a particular system. Any system that relies solely or heavily upon keyboard input, should come with a detailed list and explanation of the function keys and their meanings. Keyboard input is very individualized for each computer and it is impossible to describe any single system as being typical of all systems.

Computer programs can also generate numerical data from calculations and mathematical information that is then converted into an image on the display screen. This is presently the least common way of inputting in the 2-D artist/designer system, although it is quite common in true 3-D and other hybrid graphic systems that depend on calculations for the method of locating picture information.

On the other end of the computer chain are the *OUTPUT DEVICES*. These allow the finished computer graphic to be transferred to "hardcopy" and taken out of the machine. The two major categories that we will be concerned with are photographic and printed output.

Photographic output devices are precisely that. They allow the image on the computer screen to be transferred to film or photo paper in the form of photograph. The format may vary, but it is most commonly a 35mm camera. Other formats are from movie film (animation) and large format sheet film (4x5 and 8x10). Other photographic devices are write-once videodiscs, videotape, and printing plates.

The methods for taking a photograph of computer graphics range from aiming a camera directly at the screen and shooting, or using a fairly inexpensive camera with a hood which blocks all light falling onto the screen (thereby

eliminating reflections); to a more complex photographic system, which produces the image on its own internal monitor directly from the computer memory. Clearly, the range, cost and sophistication of the camera devices are vast and should be considered before investing in any particular style.

Printers and plotters are varied in style of operation and output. The simplest is the single pen plotter which simply records lines. Because almost all art/design systems are capable of far more visual operations and results, the pen plotter is rarely seen in that environment, as the one and only output device. They are mostly used at the architectural and engineering workstation, where clean lines for mechanical drafting, floorplans and elevations are necessary.

Multipen plotters produce a colored image which can vary in complexity and color control. This is dependent upon the number of colored pens and the resolution of the plotter.

In the printer category there are dot matrix printers on the low end and laser inkjet printers on the middle to high end. Dot matrix printers are alphanumeric printers that transfer typewriter characters to paper. Here again, the simplicity of the device will not allow more than letters, numbers, and symbols to be printed, and therefore, restrict the user in the form of output. Dot matrix printers, like single pen plotters, do not have unusual results unless they are under the control of the artist/designer, who clearly sees the advantages and characteristics of the devices and exploits them in a unique manner.

Laser printers employ the use of a very narrow and coherent beam of light which results in precisely focused light. These are often used in in-plant publishing situations, where large amounts of paginated material can be stored, and printed in sequence on both sides of the paper and then collated for delivery. The process is repeated for each copy. The less expensive devices are used as high quality proofing devices, very much like the traditional electrostatic copier, but with greater accuracy, speed, and some say, sharpness.

Other types of output devices will take the form of thermal printers, typographic, for galleys and proofs, videotape, direct broadcast video, film for animation, videodisc, and printing plates. Most output devices can be used with most machines (once the proper "device driver" is installed, if not included with the original package), and some are included in the pricelists of any given system.

The quality of the output device is in the eyes of the beholder and the usefulness of the final result is directly dependent upon the intended purpose of the work. Most workstations will be equipped to allow different devices to be interfaced to the computer. But, by far the major output device for artist/designer workstations is the 35mm camera. Because all other output devices are generally considered to be secondary in quality and desirability by the

manufacturers, this article will limit the discussion of output devices to a simple naming of the generic types.

INTERACTIVE TOOLS: These tools are what allow the artist/designer to perform the work with fluency and efficiency. They are the controls that allow one to work in a manner that is consistent with more traditional tools and art/design interaction. Interactive tools are both hardware and software. It is the design and the integration of the two that define both good and bad interactive tools. Without a doubt, the most difficult and challenging part of designing a computer graphics system is the design of the interface and the interactive devices that will allow the user ease and fluidity of operation, while still maintaining a high level of sophistication in the tasks that are performed. It is important to keep in mind that what appears to be the same level of operation is quite different due to seemingly minor changes. Just as the comfort of a chair is the sum of all the design considerations for human sitting, and one chair that looks almost identical to another can be quite different to the individual who uses it; the design of the interface on a particular computer is also the sum of the parts that are considered during the design phase.

HARDWARE INTERACTIVE DEVICES, many of which have already been mentioned, comprise the parts of the machine that the user directly handles and controls. They are the keyboard, stylus, puck, mouse, lightpen, TSD (touch-sensitive-display), foot pedal and speech recognizers. Clearly, the design and ergonomic considerations are key factors in the usefulness and "friendliness" of these interactive devices.

SOFTWARE INTERACTIVE TOOLS: Once the particular interactive device is chosen by the manufacturer, the manner in which it functions with the workstation is largely a product of the programming written to make it work with the computer. More often than not, the actual hardware is similar and it is the software that is a key factor in the friendliness of a particular station. The issue is complex, and as workstations become commonplace in the design studio, the design of these interfaces will more reflect the needs, tasks, and specifications of the user. The software and hardware engineers of these workstations rarely take into consideration who the end-users of these machines are. They believe that they understand the process of design better than the actual user, a professional artist/designer, who has spent years doing these tasks. It is only recently that computer companies have seen the need to consult with professional artists and designers for help designing the overall interfaces that are appropriate to the marketplace. Within the next several years we will see more and more companies concerned with the "end user" rather than the CPU.

The most common interactive control written for computer graphics is the menu. This is a "listing" of those routines and tools that are part of the "designer's toolbox" of programs that will allow all the art/design functions to happen. An interactive menu will route you through the

system, enabling you to go from one routine to another in a fluid and logical manner. The design of these menus is very different in all cases and quite particular to the system. Easy to read, well laid out menus are an asset, for the user can quickly go from one tool to another. Like a studio of traditional art tools, the ease that one can find the needed instrument is a key factor in the ease in completing the work. Menus that are too cramped and too complex are ineffective, so are menus that are too limited.

Menus can be placed on a tablet (on a paper or acetate overlay). They can be put alongside of the picture space on the monitor, always in view. They can be placed in the picture space but erasable. They can be on a separate screen. Most well designed menus are activated with the stylus, puck, mouse, lightpen, or TSD. Less convenient and "friendly" menus require the user to read the list of choices and then to input the information to the CPU by way of typed commands.

Other considerations for interactive devices are concerned with the idea that the user must communicate the needs in a comfortable way, consistent with design functions. Without this approach to user control, the artist/designer may never feel at ease and in control of the system enough to ever embrace it as a common and useful tool.

III. WHAT IS A WORKSTATION SUPPOSED TO DO?

Like a studio, the graphic workstation functions as a place to perform work, a space in which to do the work, and a toolbox for doing the work. Also, like a single artist's studio, the computer graphics workstation can never supply all the tools for all the tasks an artist/designer may need to use. There is no single machine that will do every job needed to be done. There is no computer on the market that combines all the requirements of illustration, animation, typography, layout and pagination, photographic manipulation, image processing, and graphic design. And there is no system on the market that can replace all the tasks of the designer with the simple press of a button. There is no computer on the market that will ever replace the designer and artist.

What you can expect from a computer is that it will perform tasks easily and quickly, having the ability to make changes and variations with much less effort than traditional tools. Also, computers will allow the user the opportunity to explore ideas that would not ordinarily be done, for their overall complexity in accomplishing them, and because the computer opens up opportunities not necessarily available with traditional tools. In addition, the combination of tools on the computer will allow new ways of exploring visual language, perhaps redefining aspects of that language and the process of design itself.

Computer graphic workstations can be placed in specific categories. Most systems will overlap into two or more, usually with a primary function, often designated by the manufacturers, and secondary ones, which often do not perform as efficiently as the first.

The purpose of this analysis is to allow you to think about major design considerations as decided upon by the manufacturer. What is important in this categorization is to keep in mind the overall purpose of the computer and to be aware that compromises had to be made when designing the system. Because no one computer graphic workstation will do all design/art jobs, the primary category is the one that will be most often followed, compromising on the secondary requirements.

IV. THE FIVE CATEGORIES

BUSINESS AND PRESENTATION GRAPHICS: These systems are designed to prepare graphics for presentations and for consolidation of data in a visual graph and chart format. Attributes of these machines are primarily concerned with their ability to store and interpret data in a visual manner, perhaps through a variety of graph and chart styles once the initial data is entered into the machine. Generally, the choices for color are more limited than other types of systems and in some cases the colors have been pre-chosen into very specific palettes. Often times they are quite bold, allowing ease of presentation when made into slides. In most cases much concern has been given to the design of and the use of type, in presentation formats. Many times the display monitor is simply a rough version of the final graphic, which uses a very sophisticated camera to greatly increase the resolution of the output.

Because many of the graphics are the result of statistical data and must be interpreted into charts and graphs, a common interface is the keyboard as well as the tablet. The systems often use basic shape-making routines that produce circles and rectangles and other polygons to format a presentation, and then manipulate these shapes. In many instances the workstation will incorporate a basic paint system to allow "free-hand" painting onto the screen.

ARTISTS' ILLUSTRATION AND SKETCH SYSTEMS: These workstations are primarily used for paint systems and image manipulations. They have much more sophisticated paintboxes, with many more color and brush choices. As these systems become commonplace and their costs drop, they will likely be the workhorse of the basic art/design studio, for they contain more of the typical tools for artists and designers, with increased flexibility.

It is quite easy to find systems today that have digitizing cameras as part of their basic systems, as an accessory. These allow you to input photographic and live video information and then to do all the painting and manipulation tasks with the digitized image. Typical manipulation routines include scaling, rotating, and distorting. (Keep in mind that these manipulations are all 2-D, on the x and y axis only; i.e. these are not true 3-D systems. But it is the clever use of these functions that allow the

artist to render an illusion of 3-D.)

Other common routines found in this type of system are those having to do with type, grids, color changing, collaging, and cut and paste tasks. The more sophisticated the system the more colors available—up to the ability to display approximately 250,000 colors on the screen at any one time out of a field of approximately 16.8 million colors. The low end of this scale will offer a display of 12-64 colors out of a field of about 500. Most systems are presently in the middle with approximately 256 colors displayable colors from the field of 16.8 million choices.

Output from these systems is usually in the photographic formats noted in the introduction. However, more and more systems are now interfacing to inkjet plotters, which are available for reduced costs.

DIRECT BROADCAST VIDEO AND ANIMATION SYSTEMS: As their names imply, these systems are made for the broadcast and animation needs of telecommunications. They are often found in TV studios and are used to have artwork quickly prepared and then sent directly to the broadcast studios for display on the air. Most of the news graphics that are shown above the left shoulder of the newsreporters are prepared in a studio down the hall and wired directly to the video mixers in the production studios. Requirements for these systems are high-speed storage and retrieval of images, size of the storage library for on-line images, type (headlining, mostly), and image manipulation. It is usually imperative that they be in some way connected to a digitizing camera, so that live video and photographic input can be easily merged with graphics and then manipulated. High resolution is not an issue, for the resolution of TV broadcast is relatively low, and the graphics need not exceed that amount.

PAGINATION SYSTEMS: The pagination systems are designed for page layout and publication design. The components include text, line art, halftones, headings and folios. They were developed initially for the prepress market and are commonly seen in newspaper layout rooms or catalog houses where there are large quantities of layout and type manipulations in need of rapid publication.

Their ability to format and size type is essential. From the low end to the high priced systems within this category, the manipulation capabilities become more and more sophisticated; with the most advanced systems allowing a very rapid manipulation and exact visualization of what will happen in the output. In the lower priced systems only approximations and/or "greeking" of the text is possible. Color has not been a major consideration up to this point because of the immense expense involved, and most systems offer no color at all. It is the black and white nature of these machines that offer the speed of operation.

Pagination systems are usually front ends and offer a large variety of type styles and sizes. In the higher priced systems they are capable of merging the text with images. Output from these systems goes to typesetters, paper proofs, and camera-ready artwork, and some systems can be interfaced directly to plate-making devices at the prepress site.

IMAGE PROCESSING SYSTEMS: These systems allow photographic manipulations to be done directly in the computer that would take hours to duplicate in the darkroom. Sizing and scaling, rotating, cropping, tone and contrast control, and special effects (reversals, solarization, posterization, bas reliefs, etc.) are all part of the manipulation menu available on these machines. Images are scanned into the systems in high resolution, manipulated, and then output to camera-ready copy or directly to plate.

Most systems are black/white only, at the present time. At the highest end, however, there are capabilities for color separations to be made with three and four plate registered output.

V. COSTS AND PRICE VS. PERFORMANCE

Computer graphic workstations are expensive. In the traditional studio, artists and designers wishing to experiment with a new tool are able to go to the nearest supply store and purchase it. With the introduction of high technology equipment into the art/design marketplace, the ability to learn and experiment becomes rare outside the production house. Even within the production house there is little time to experiment on non-production jobs.

Presently, the range of equipment is from approximately \$6000 on the low end to \$1.5 million on the high end. The mean is approximately \$30,000 for the average system, base cost, with no bells and whistles. The costs of purchasing and maintaining equipment still remain very high, and although they are decreasing rapidly, they will still remain outside the budget of the small studio for quite some time.

With regard to the five categories, the least expensive systems usually are those for business and presentation graphics, continuing up the cost scale through image processing systems. As the systems increase in their complexity and add to their toolbox, the price goes up (and up...and up). Thus, the more sophisticated system that allows the user a full range of tools and functions becomes cost prohibitive for all but the largest and busiest production houses. In turn, the cost of operation of these systems is such that the larger and more complex the system, the less time there is for "creative designing" vs. production work. Indeed, this is a dilemma, for the increase of computer graphics in the world of visual communications necessitates all artists and designers to be knowledgeable of (if not experienced in) computers and their capabilities. But the opportunities to get the experience is often the biggest deterrent to the artist.

Clearly, you get what you pay for in computer graphics workstations. The ability to design and manufacture a machine containing all the needed tools of the typical designer and still remain a system that is within the budgets of most users is a problem not yet solved. As each system reaches out to include more and more functions, with better and better interface, the cost of the hardware drops proportionately; but, the cost of the software that implements these tools is continuing to rise.

The result is that systems are limited in their toolboxes and their functions. The manufacturers must make decisions concerning what to include in their systems that will be the most attractive to the potential user, leaving out many options that are technically available, but far too costly to include. Sophisticated tools are costly because of the memory space needed in the computer. The more tools, the larger the space required to house them and to allow them to operate. Without upgrading the extent of memory of the base system, little is left for accessories and picture storage. Compromises must be made by all manufacturers, and those compromises are the very definition of the machines.

As mentioned in the introduction, the inherent hardware of any system is very similar to others in its category. Their differences are in the initial design and implementation of the software. It is that configuration and implementation that define the individual machine and its functions.

VI. CONCLUSION

Every artist/designer using a computer must keep in mind the task to be done and how appropriate that system is for that task. Some tasks demand compromise in the ease of control for complexity of operation. Many factors add to and subtract from the effectiveness of the interface because one function is linked to all others in the chain. Interactive devices should be concerned with the idea that users can communicate needs in a comfortable and consistent way. Using keyboards as a major input device is a good example of how the computer companies must become more sensitive to the issue. The keyboard allows you to input graphic elements which are "plotted" by counting spaces on the keyboard. Although this task is clearly effective for inputting textual material, it is highly inefficient and counter-intuitive for graphics. Because of the demand of the design community, we see fewer examples of systems that have keyboards as their major input device. It is the growing demands of this community that will force the industry to change the methods, and hopefully, produce devices that are effective as well as sophisticated for design tasks. It is not the purpose of this article to debate and discuss the features of all interactive devices, but rather to understand that the computer manufacturer is still in control of the ease and comfort of the machines sold to the public.

It is obvious that the computer, like the camera, is a tool. The workstation for computer

graphics is commonplace in many studios and art departments. Colleges and professional schools are teaching computer graphics as part of their foundations curriculum. Enrollments in all computer graphics courses, no matter how elementary, are well above the space available. It is not simply the "hottest fad" of the year. It is part of the workspace and the vocabulary of visual communications. With this in mind, the workstation is more and more available and its use obvious in all aspects of design and art studios. For the person wanting to enter this world, it is important to understand the initial reason for investing in a computer graphics workstation, for those reasons will be the key factors in deciding what category of system to investigate, and how much money to spend to purchase the minimum equipment needed to get the work done.

*NOTES: In contrast, a true 3-D system is one that can render three dimensional forms from data (numerical). These forms can be objects, environmental drawings and various forms of simulation; taking into account the angle of view, rotation, volume, and other statistical information. A 2-D system is quite different from the 3-D one by the very attributes that define them, and it is important to understand that they rarely overlap in the same system.

There is a "pseudo 3-D" often found in computer graphics workstations, that gives the appearance of dimensionality, but does not have true 3-D characteristics of rotation, variation of angle of view, and scaling. The results of this could be described as the "look" that is part of many graphic systems designed for artists and designers. These pseudo 3-D packages are linked to the 2-D system and should not be compared with the true 3-D systems in terms of performance. The success of these pseudo 3-D tools is really the cleverness of the user, describing the illusion of volume, depth and space much like traditional perspective drawings.

THE AUTHORS: *Alyce Kaprow* and *Joan Shafran* are consultants and researchers concentrating in the area of computer graphics workstations for art and design studios and art schools and departments. They both are active designers and teachers, and have run hands-on workshops using various computer graphics systems. Both have done graduate work at the Visible Language Workshop at the Massachusetts Institute of Technology.

INTERSCORE - AN INTERACTIVE SCORE EDITOR FOR MICROCOMPUTERS

Przemyslaw Prusinkiewicz

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2

ABSTRACT

Since 1970, the GROOVE system has been the classic example of an interactive editor of time functions. It has inspired the design of an interactive score editor for microcomputers, named INTERSCORE. A composer is provided with a wide range of easy to use editing operations, and with both audio and visual feedback (in piano-roll notation). An organ-like keyboard is extensively used as an input device. Time relationships are given particular attention.

INTRODUCTION

Buxton et al. [2] divide the composer's tasks in the context of computer music as follows:

1. Definition of the palette of timbres to be available: This is analogous to choosing the instruments which are to comprise the composer's orchestra. The main expansion on the analogy is that the composer also has the option to "invent" his own instruments.
2. Score definition: Definition of the pitch-time structure of a composition. In conventional music, this task would be roughly analogous to composing a piano version of a score.
3. The "orchestration" of the score: Attachment of instruments to a score.
4. The performance of the material being developed.

The conceptual framework induced by the above classification is observed in this paper. Thus, instruments are defined separately from the score, and the possibility of setting up timbres individually designed for each note is limited. In return, the removal of the instrument definition from the score makes it potentially easier to construct long, complex pitch-time structures.

At the present time, music software available on microcomputers makes it possible to define scores using either one of the following approaches:

1. A score is defined as a sequence of statements which explicitly specify the pitch and duration of each note. Definition of a score is conceptually similar to text editing. It is also possible to edit a score using a symbolic musical notation. The score is then specified, for example, by picking notes of the appropriate duration from the menu and positioning them on the staff [12].
2. A score is thought of as a record on a "pseudo-tape," simulated in the memory of a computer. The sequence of events is defined in real time, by playing an organ-like keyboard used as the input device. The user's model of the system is that of a multi-track tape recorder. Thus, a score is developed by "recording" successive instruments with "playback" on separate "tracks." An unsatisfying track can be "erased." Smaller corrections can be made using a "punch-in/punch-out" facility [8].

The essential difference between these approaches lies in their rapport to time. In the first case the composition time is dissociated from the performance time. Time characteristics of a note are defined as numerical parameters. Consequently, the order in which the notes are specified may be different from the order in which they will be played during a performance. The score can be easily modified by inserting, deleting, or changing appropriate statements. However, since it has to be substantially processed by the computer before the performance, effects of the modifications cannot be instantly evaluated.

In the second case the situation is diametrically opposite. The performance time is a linear function of the composition time. A composer has immediate audio feedback: He hears what he plays. Modifications of the score are made in real time, by rerecording unsatisfactory parts. As a result, the whole composition process occurs under real-time pressure. A mastery of the keyboard is necessary. Small adjustments are difficult to make.

An interactive score editor should provide a composer with a wide range of easy to use editing operations combined with good audio and visual

feedback. This objective was first met in the GROOVE system [10]. The nature of interaction implemented in GROOVE was described as follows:

One of the most important features of GROOVE is the flexible control of "program time" which may be used both to edit and to alter the generation of the output functions.... We may slow down the progress of program time by reducing the frequency of the interrupt oscillator. Or we may stop the progress of time altogether by throwing a switch which essentially tells the computer: "Don't progress time normally at all, but, instead, use the value of a knob to give the current position of time within one disk buffer...." The user may essentially "redraw" any portion of any disk function using any input device he likes, such as the (X,Y) axes of the 3-dimensional wand or a knob value. While he is doing this, not only can he see what he is doing on the oscilloscope display, but he can also observe its effect on the controlled process. So it is quite possible to stop in the middle of a run and "tune up the chord...." Given the appropriate commands, the system will allow any functions of time to be altered in any conceivable manner.

The idea of the GROOVE system has been widely recognized as a model example of interactive score editing [6]. However, the system itself was not portable and is not available since the unique hardware was dismantled.

INTERSCORE is an interactive score editor for microcomputers, inspired by GROOVE. It is written in C, in a modular and portable way. At present INTERSCORE is implemented on the Apple IIe (*), with a 5-octave alphaSyntauri (**) keyboard and Mountain Computer Music System (+) synthesizer boards [12]. Two pedals and a joystick complete the hardware configuration. INTERSCORE makes use of the Syntauri programs Quickwave and Wave to define instruments. INTERSCORE files are compatible with the multitrack recording system METATRAK (**) [8] and the Composer's Assistant (**) [13] can be used to transcribe the score using conventional notation.

This paper presents INTERSCORE from the user's (composer's) perspective. Time problems related to interactive score editing are given particular attention.

GENERAL DESCRIPTION OF THE SYSTEM

From the user's perspective, the editor is embedded in a menu-driven control program. The functions of this program fall into three categories:

1. Selection of the instruments (timbres) to be used for audio feedback. These instruments can be, but do not have to be the same as the instruments used for the final performance.
2. File manipulation. This category includes concatenation and merging. Concatenation of files results in longer scores. Merging increases the number of instruments playing concurrently [1].
3. Transfer of control to utilities, such as programs to define new instruments, the screen dump program, etc.

An example of the screen while editing is shown in Fig. 1. The central area of the screen is thought of as a window, in which the selected portion of the screen is visualized as a plot in pitch-time coordinates (scroll-bar notation [3, 9]). A non-continuous line indicates two or more instruments playing in unison. For comparison, Fig. 2 shows the same piece of music using conventional notation.

Wide lines at the top and the bottom of the score (Fig. 1) show the current position of the time cursor. The notes corresponding to this position are being played by the synthesizer. Under the score, the text portion of the screen is used to display the menu of editing operations and to show the current state of the editor. The selected mode is displayed in inverse video. Special characters warn about particular situations, such as the end of the score being reached. The space in the upper right corner of the text portion of the screen is used to display additional information, for example parameters of editing operations, error messages, etc.

SCORE VIEWING

Even relatively short scores cannot be represented in their entirety on the limited surface of a screen. Thus, an interactive score editor must provide viewing operations which make it possible to select the portion of the score to be seen. In INTERSCORE this selection is thought of in terms of moving a virtual window over the scroll-bar representation of the score. Three special keys ([, |,]) are used to position the window in such a way that the time cursor appears near the left edge, in the middle, or near the right edge of the window. The composer can also specify the resolution of the presentation (number of time units per pixel). Wide scope - low resolution views help in analyzing the general structure of the composition and are particularly useful when browsing through the score. High resolution views are of great value when making fine modifications.

(*) Apple IIe is a trademark of Apple Computer Inc.

(**) alphaSyntauri, Metatrak, and Composer's Assistant are trademarks of Syntauri Corp.

(+) Music System is a trademark of Mountain Computer Corp.

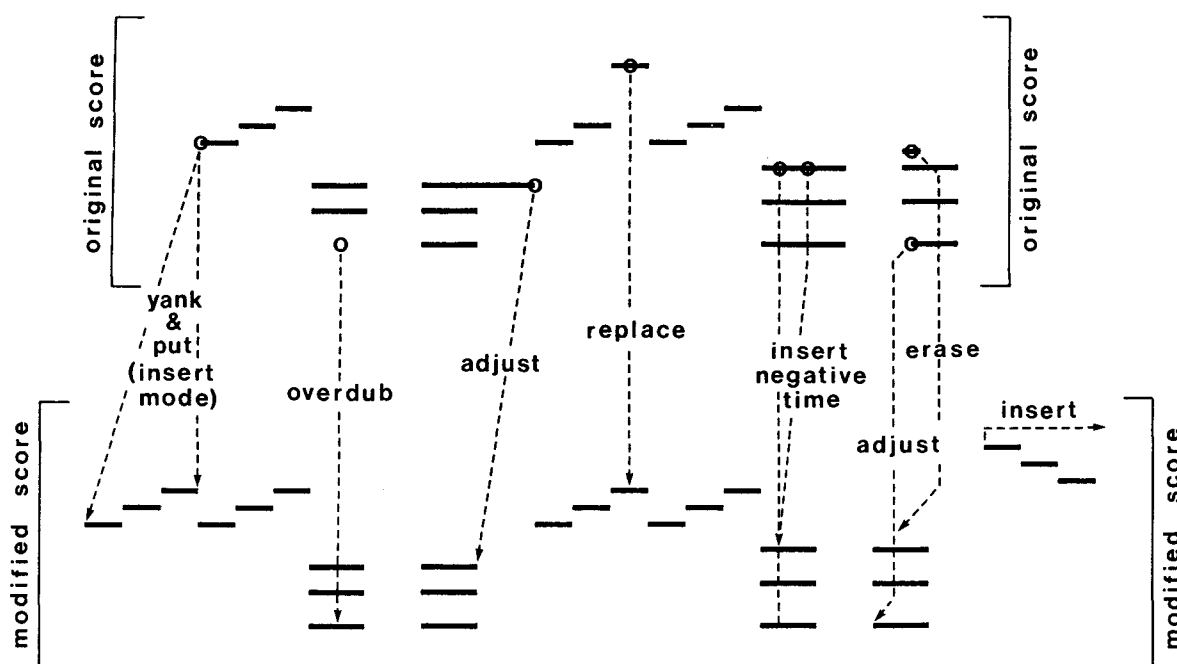


Fig. 3. INTERSCORE editing modes.

pointing to them with the time cursor (x coordinate) and the keyboard (y coordinate). By playing a chord, several notes can be erased simultaneously.

The replace mode is used to change the pitch or the instrument assigned to the selected note.

The adjust mode makes it possible to move the beginning or the end of a note in time.

The remaining two modes: Yank and put are patterned on similar operations found in text editors [11]. Yank fetches a portion of the score, played by the specified instrument, delimited by two positions of the time cursor. Put, in essence, places the yanked portion at another point in the score. This can be done in several ways. First, either the insert or the overdub mode must be selected together with put in order to determine, how the added notes shall be matched with the existing portion of the score. Second, the yanked fragment can be repetitively put, several times in sequence, transposed by an interval specified with the keyboard, and possibly with the instrument changed.

All editing modes are illustrated in Fig. 3.

TIME MANAGEMENT

Time is the essential component of both the process of the composition of a piece of music and its performance. Therefore, the relationship between the composition time and the performance time should be easily manageable by the composer. For this purpose several time modes have been introduced into INTERSCORE.

The simplest relationship between the composition time and the performance time is illustrated in Fig. 4a. During the composition time the keyboard is sampled and the score is updated in equal intervals denoted by D_{tc} . During the performance the score is read and data to the synthesizer are passed in intervals D_{tp} . The ratio D_{tp}/D_{tc} controls the "playspeed." If it is different from one, the performance will be faster or slower with respect to the composition. Absolute values of intervals D_{tc} and D_{tp} control the "time resolution" of the composition. For small values of these intervals (milliseconds) the discretization of time is negligible and the sequence of events entered into the score is repeated with all nuances during the performance (Fig. 4b). Large values of time intervals (fractions of a second) let smaller time differences disappear. This can be used to eliminate some imprecisions which may occur when entering data from the keyboard (Fig. 4c). The

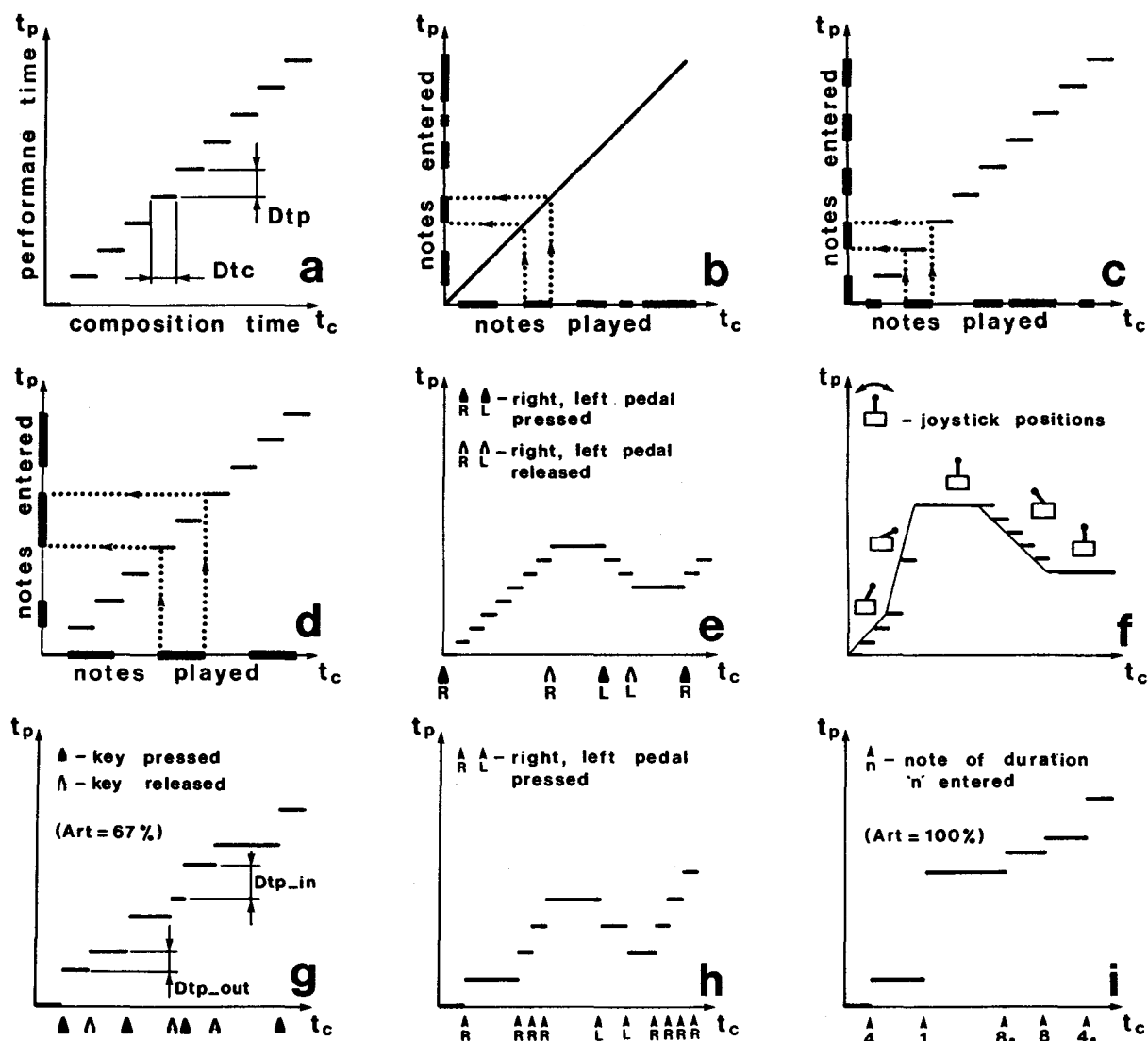


Fig. 4. Composition time - performance time relationship in INTERSCORE.
Explanation in the text.

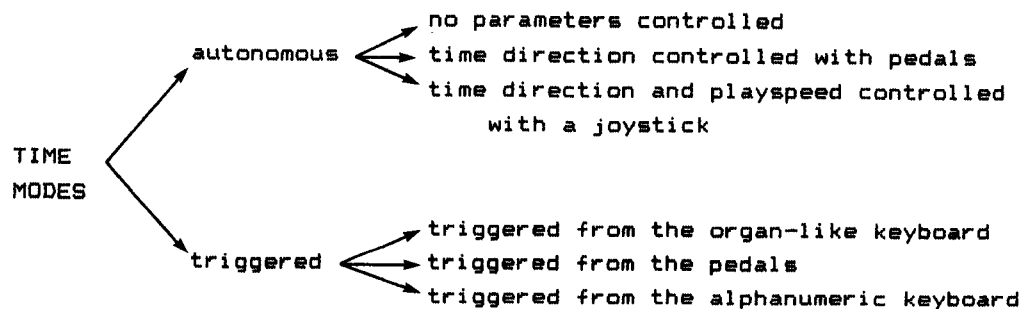


Fig. 5. INTERSCORE time modes.

composer has to enter data in synchronization with a metronome generated by the editor. Otherwise low time resolution may deteriorate rather than improve the score (Fig. 4d).

Abstracting from their finite resolution, the above modes define linear mappings of the composition time into the performance time. A modification introduces a piecewise linear mapping. At any moment the composer can stop the progress of time or reverse its direction by using one of two special pedals (Fig. 4e). While the composition process is still very similar to real-time recording, errors can be instantly deleted before the recording resumes.

When scrolling a score to access a particular fragment, it is convenient to dynamically control the playspeed in addition to the time direction. The device used for this purpose in INTERSCORE is a joystick. See Fig 4f. for an example.

In the modes described so far, time has progressed autonomously. The external devices have only controlled the actual value of parameter Dtp. These autonomous time modes can be contrasted to the triggered modes. In a triggered mode each change of the performance time is directly caused by an external signal. One possible source of this signal is the organ-like keyboard itself. In this case time progresses when a key is being pressed or released. The time intervals associated with the pressed or released keys, Dtp_{in} and Dtp_{out}, need not be the same (Fig. 4g). If

Dtp_{in} >> Dtp_{out},
the entered notes will be performed legato; if
Dtp_{in} << Dtp_{out}
they will be performed staccato. Instead of controlling parameters Dtp_{in} and Dtp_{out} directly, it seems more convenient to specify the overall duration of a note:

Dtp = Dtp_{in} + Dtp_{out}
and its "articulation" (or duty cycle):
Art = Dtp_{in} / Dtp.

Another source of triggering signals are pedals used to increment or decrement performance time by a predefined value (Fig. 4h). While the organ-like keyboard is active all the time, its state affects the score only when a pedal is pressed. Thus, the composer can try, for example, a few possible chords before entering the final one into the score. The duration of this chord will be determined by the number of times the pedal is pressed while the keys are down. Instead of pressing the pedal repeatedly, the composer can also specify the duration of each note or pause by entering appropriate values of the parameter Dtp from the alphanumeric keyboard. A variant of this approach makes use of symbols 1, 2, 2., ... to denote the duration of the whole note, the half note, the dotted half, etc. in a predefined tempo and with a predefined articulation (Fig. 4i). This mode is particularly convenient when entering a score given in conventional music notation.

A summary of time modes provided by INTERSCORE is shown in Fig. 5.

CONCLUDING REMARKS

Design and experimentation with INTERSCORE has yielded some observations:

1. Scroll-bar notation is a convenient means for visual communication between the composer and the computer. The difficulty in perceiving the exact pitch of each note (with no staff) is irrelevant because of the audio feedback. Moreover, it is easy and intuitive to find the pitch of a note by matching it with the displaceable bars corresponding to the keys pressed on the keyboard. Due to the audio feedback, precise positioning of the time cursor (up to a single pixel) is trivial. The visual and the audio feedback are complementary.

2. When a long score of a repetitive structure is edited as a single file, the composer may confuse similar parts of the score. It is therefore preferable to build long scores using separate files, which are concatenated at the end of the editing session.

3. Although the description of INTERSCORE was given in terms of physical devices (joystick, pedals etc.), the software was written in a device independent way, patterned on the methodology developed in computer graphics [5]. Thus, only the appropriate device drivers have to be rewritten when replacing the joystick with a different valuator, or pedals - by any other buttons. Apparently, the organ-like keyboard does not fit into the existing classification and introduces a new class of logical input devices. The problem is, however, that the main purpose of the device independent design (i.e. the possibility of simulating various logical devices using the available physical devices) is questionable in highly interactive musical applications. Substitutions of physical devices (for instance, hand manipulated buttons for pedals, let alone a replacement for the organ-like keyboard) tend to deteriorate the man-machine interface to the point of uselessness, even if they are perfectly feasible from the viewpoint of software design.

INTERSCORE has not yet been extensively tested by musicians, therefore an objective evaluation is not available. However, the main expectation related to its creation seems to be fulfilled: Using INTERSCORE it is easy to quickly create complex, error-free scores. Moreover, if desired, nuances of articulation giving a feeling of "real performance" can be preserved in the editing process.

ACKNOWLEDGEMENT

This research was supported in part by grants from the National Science and Engineering Research Council of Canada.

REFERENCES

- [1] M. Balaban, "The set of tonal-music-strings - a structurally unambiguous representation for tonal music pieces," Weizmann Institute of Science Rep. CS81-28, 1981.
- [2] W. Buxton, W. Reeves, R. Baecker, et al., "The use of hierarchy and instance in a data structure for computer music," Computer Music Journal, vol. 2, no. 4, pp. 10-20, Dec. 1978.
- [3] W. Buxton, R. Sniderman, W. Reeves, et al., "The evolution of the SSSP score editing tools," Computer Music Journal, vol. 3, no. 4, pp. 14-24, Dec. 1979.
- [4] P. Casella, AlphaPlus Tutorial Manual. Palo Alto, CA: Syntauri Corp., 1982.
- [5] J. D. Foley and A. van Dam, Fundamentals of Interactive Computer Graphics. Reading, MA: Addison-Wesley, 1982.
- [6] S. Haynes, "The musician-machine interface in digital sound synthesis," Computer Music Journal, vol. 4, no. 4, pp. 23-44, Dec. 1980.
- [7] H. S. Howe, Electronic Music Synthesis. New York, NY: Norton 1975.
- [8] R. J. Jigour, Metatrak II User's Manual. Palo Alto, CA: Syntauri Corp., 1982.
- [9] G. Krasner, "Machine tongues VIII: The design of a Smalltalk music system," Computer Music Journal, vol. 4, no. 4, pp. 4-14, Dec. 1980.
- [10] M. V. Mathews and F. R. Moore, "GROOVE - A program to compose, store, and edit functions of time," Communications of the ACM, vol. 13, no. 12, pp. 715-721, Dec. 1970.
- [11] N. Meyrowitz and A. van Dam, "Interactive editing systems: Part I and II," ACM Computing Surveys, vol. 14, no. 3, pp. 321-415, Sep. 1982.
- [12] Music System Operating Manual, Scotts Valley, CA: Mountain Computer, 1981.
- [13] K. Reynolds and R. Jigour, Composer's Assistant User's Manual. Palo Alto, CA: Syntauri Corp., 1983.

AN ASSESSMENT OF MICROCOMPUTER-BASED ART ACTIVITIES FOR SECONDARY ART PROGRAMS

Candace M. Naumowicz Zacher

Purdue University
Department of Instrumental Development
West Lafayette, Indiana

ABSTRACT

This study investigated the curriculum concerns for microcomputer-based art activities for secondary art programs. Art supervisors and computer art "experts" were surveyed concerning their opinions for content consideration. Although the participants were generally receptive to microcomputers in the art room, findings indicated several areas of disagreement and indecision as well as a number of impediments to the successful adoption of microcomputers into the art room.

A trend in education toward the acquisition of skills in the myriad uses of microcomputers is evident today as our society begins a shift to a technologically-based society. Being an integral part of the larger educational structure, art education will not likely be excluded from this transition. The urgency and importance of the problem is reflected in a statement by White¹: "It is imperative that the art education profession not become an impotent pedagogical edification due to our unwillingness to accept the responsibility thrust upon us . . . caused by the impact of computers on society and education." In other words, art programs will be expected to provide some type of effective art experience for students on microcomputers.

Only recently has the role of the microcomputer in the secondary art curriculum been critically examined by art educators.^{1,2,3,4} Little research exists examining the new technology and its implication for teaching art on the secondary level. Most existing reports on microcomputers in the art classroom focus on two factors: the present discrepancy and lack of concern for this new medium in the curriculum, and the role microcomputers could serve in the art program. Relatively little is known, however, about what is presently being offered on the microcomputer in art at the secondary level and what will be needed in the near future.

Educational technology as a field has emphasized the integration of new technology into educational experiences. At the same time, educational technology is concerned with the utilization of a systems approach to the development of instructional delivery systems. Needs assessment procedures are a device used in educational technology and

instructional development that could provide the foundation for development of clearly identified aims and goals for a secondary art curriculum in microcomputer-based art activities.

The primary purpose of this descriptive study was to determine by means of a needs assessment the major components and skills that a secondary art curriculum should include in order to facilitate student involvement and competence with microcomputers in art. Writers who support the use of systematic procedures for instructional development have suggested that needs assessment may capitalize on pertinent information and thus facilitate better decisions in the future development of instruction in the use of microcomputers in art for secondary art students.

The evaluation of all potential content areas for microcomputer-based art activities is clearly a critical matter for art education programs. Certain factors may need to be considered by art educators concerning microcomputer-based art content and it would be advantageous to identify these factors so that effective implementation and adoption of microcomputer-based art activities is possible in secondary art programs.

A consensus in agreement of the content concerns for microcomputer-based art activities would seem to be a valid venture to undertake in art curriculum development. The procedures labeled "needs assessment" promoted by Briggs⁵ and Kaufman⁶ suggest that instructional options be subjected to some form of empirical analysis so that art educators would be able to make decisions based on a mutual sense of purpose and agreement. In addition Eisner and Eiler⁷ also indicated the importance of using methods to facilitate change in art so that art educators would not resist change but "employ critical procedures by which wise choices may be made among competing proposals for change." With few exceptions, such data collection and empirical analysis have not been conducted.

A second issue in a needs assessment analysis was to identify the present status of microcomputer use in secondary art classrooms in order to establish the need for the development of related goals. The identification of this present status was secured by investigating the current abilities of secondary art teachers in microcomputer-based art activities.

A major question may be raised regarding how art supervisors see the current abilities of art teachers in the use of microcomputers in the art classroom. Do art teachers have the knowledge of microcomputer applications but not the actual skills to use them? Are the microcomputer skills they possess largely data management? In other words, do they use microcomputers for record keeping and grading of students? Do the art teachers have microcomputers in their schools to use in the art room? Very little evidence is currently available on the present status of microcomputer use in the secondary art classroom.

In this study, the use of microcomputers in the secondary art room was explored as a creative medium and not as a tool for computer-assisted instruction in secondary art programs. The study assessed the current conditions for art teacher preparation in microcomputer-based art activities and the possible content areas to be included in a secondary art program.

Method

Sampling

The study's population consisted of two separate groups. One group consisted of all art supervisors, numbering 369, across the United States who were listed in the American Art Directory, 1982.⁸ The second group was designated as computer art "experts." This group consisted of art educators from higher education. Their selection was based on the following factor: publication of one or more articles on the topic of computer art or computers in art education or one or more courses taught in computer art/graphics or computers in art education.

The sample used in this study consisted of 137 art supervisors and 24 computer art "experts." A stratified random sample of art supervisors was used. The stratification method was utilized in order to reduce variability of the sample. A possible source of error, it was felt, was randomly selecting a sample heavily composed of states with a greater number of art supervisors. This situation could have influenced the internal validity of the study. To offset this possible source of error, a maximum of three randomly-selected art supervisors were chosen from each state unless the state had fewer than three supervisors listed. When the first selection was completed, a further random sample was conducted with the remaining names to achieve the sample size of 137. A major reason for using stratification was to increase the likelihood of a representative sample of the total population.

The justification for a sample size of 24 in the group designated as "experts" was that there are not enough art educators currently who have experience in computer art that would allow for an increase of the sample number. Those individuals who do have the experience are primarily professionals on the post-secondary level.

Objectives

The major objectives of the investigation were to determine:

- (1) the categories of microcomputer-based art skills that are appropriate for secondary art students to develop;
- (2) the present abilities of secondary art teachers to provide microcomputer-based art activities; and
- (3) the present availability and use of microcomputers in art on the secondary level.

Procedures

Data were collected by means of a survey utilizing a mixture of question types in order to secure the most accurate as well as objective information in quantitative form. The survey consisted of three sections: (a) general background data, (b) participants' opinions concerning the potential of specified content regarding use of microcomputers in secondary art education, and (c) present status of secondary art teachers and secondary schools in regard to microcomputer-based art activities.

The reliability of the study was established by a split-half reliability test based upon a pilot study. Calculation of test reliability was conducted on the second section only because this section addressed the primary concern of the study: what topics should be considered for inclusion in a secondary art class on microcomputer-based art activities. In addition, it was the intention of the other section to gather descriptive data which were specific to the participant's situation and did not lend to reliability testing.

After the survey was determined to be a reliable instrument, the survey was distributed by mail to the selected participants of the study. Participants were asked to complete the survey and mail it back to the investigator. A follow-up letter was sent to nonrespondents to motivate them to complete the survey. The total number of responses used in the study was 97; 86 art supervisors and 11 computer art experts.

Data Analysis

Content Considerations

The results of the frequency distributions and the means of the question, "Microcomputers and their exploration as an art medium should be included in a secondary art curriculum," are presented in Table 1. Since the combined percentages of agree and strongly agree was 83%, it would appear that art educators are receptive to the inclusion of microcomputer-based art activities in the secondary art curriculum.

However, it was a major concern of this study to identify more specific areas that art supervisors and computer art experts felt should be included in a secondary art curriculum in the use of microcomputers. Eighteen content areas were chosen for the survey after a review of the literature. Table 2 reports the results concerning these 18 curriculum

Table 1

Responses to "Microcomputers and their exploration as an art medium should be included in a secondary art curriculum"

CATEGORY LABEL	FREQUENCY	PERCENTAGE
Strongly Disagree	0	0.0%
Disagree	4	4.0%
Undecided	12	12.0%
Agree	47	48.0%
Strongly Agree	34	35.0%
N = 97		mean = 4.144

concerns for microcomputer-based art activities. The data revealed that the areas that should be included in a secondary microcomputer-based art curriculum as perceived by art supervisors and computer art experts were: (a) the use of peripheral devices; (b) the evaluation and criticism of computer-generated images; (c) the creation of computer-generated images; (d) the identification of careers in computer graphics; (e) the appreciation of computer-generated images and the ability to assess their use in the environment; (f) the definition of computer art terms; and (g) the selection of appropriate software for personal use. The use of peripheral devices was seen as an important facet of a microcomputer-based art curriculum by 79 of the respondents (81%). Eighty-three of the participants (86%) felt that the ability to critically evaluate computer-generated images was a salient competency for secondary art students. In addition, 81 of the respondents (84%) thought that the creation of computer-generated images was a significant skill for students to exhibit. Eighty participants (82%) from the survey agreed that being able to identify suitable careers in computer graphics was important for art students.

The area of assessing the implications of using computer-generated images in our environment was viewed by 86 of the participants (89%) as a content area to emphasize. Being able to appreciate and value computer-generated images was considered by 84 respondents (87%) to be a worthwhile ability to possess. Seventy-nine participants (81%) of the sample had the opinion that definitions of computer art terms need to be included in microcomputer-based art activities. Finally, seventy-three respondents (75%) ranked the selection of graphic software packages for personal use as a topic to be included in a secondary microcomputer-based art program.

The areas that art supervisors and computer art experts felt should be of least importance or were uncertain about for inclusion in a secondary microcomputer art curriculum were: (a) computer animation, (b) use of commercially-produced graphic software programs, (c) three-dimensional graphics (3D), (d) computer programming, (e) descriptions of hardware and software devices, and (f) two-dimensional graphics (2D). By combining the strongly disagree and disagree categories, the results of the data obtained were that 49 of the participants (50%) felt computer animation should

not be included and 62 participants (64%) felt commercially-produced graphic software programs should not be the major focus. Sixty-seven respondents (69%) felt production of 3D graphics was either not necessary or they were undecided.

Forty-eight respondents (50%) were unsure computer programming was a necessary skill for art students. The sample population appeared to be evenly divided concerning the ability to describe hardware and software devices (mean = 3.39) and the abilities to produce 2D graphics (mean = 3.42).

The data also reveal that of the 81 individuals responding favorably to the question, "Should students be able to produce computer-generated images?", 57% felt they should be able to create 2D designs, 37% felt they should be able to produce 3D designs, and 32% felt the student should be able to create computer animation (Table 3).

Table 3

Breakdown of Responses to Types of Computer-Generated Images Students Should Be Able to Produce By Participant's Response

CATEGORY LABEL	FREQUENCY	PERCENTAGE
2D graphics	46	57%
3D graphics	30	37%
Computer animation	26	32%

Current Status

The frequencies of responses of art supervisors concerning the competency levels of art teachers in microcomputer-based art activities are presented in Table 4. It appeared that there was consistent agreement concerning art teachers' current lack of knowledge and skills in most of the competencies listed. From this sample, it appeared that few school systems have secondary art teachers who possess strong microcomputer skills.

Table 5 presents the number of microcomputers per secondary school while Table 6 reports the number of secondary art teachers who offer some art activities on microcomputers to their students. Comparing these two tables, the results seem to indicate that although 92% of the schools surveyed do have microcomputers, only 27% of the secondary art teachers provide art experiences on them. These data indicated that the majority of schools do have microcomputers available. The data also reveal that of the secondary art teachers who do provide art experience on microcomputers the range is from one art teacher in an entire school system to 10 in a school system. However, 63 art supervisors (73%) responded that none of their art teachers provide microcomputer art experiences.

Problems That Need to Be Addressed in Using Microcomputers in the Secondary Art Classroom

A final open-ended question was provided in the survey concerning immediate problems to be overcome.

Table 2
Summary of Frequencies and Means of Curriculum Concerns
for Microcomputer-Based Art Activities

	STRONGLY DISAGREE	DISAGREE	UNDECIDED	AGREE	STRONGLY AGREE	MEAN
The use of peripheral devices (i.e., graphics pad, light pen) with the microcomputer should be a part of a secondary art curriculum.	3	1	14	40	39	4.144
Art students should be able to evaluate and criticize the quality of computer-generated images.	3	3	8	58	25	4.021
Art students should be able to discuss the historical developments of computer art and identify the important artists in this field.	1	13	25	46	12	3.567
Secondary art students should develop the skill to create and produce computer-generated images.	0	5	11	49	32	4.133
Art students should be able to use the microcomputer as a tool for all areas of art.	7	9	20	40	21	3.608
It is important for art students to be able to identify suitable careers in computer-generated imagery.	1	7	9	54	26	4.000
The ability to create animated sequences on the microcomputer does have to be a skill art students demonstrate.	1	48	22	24	2	2.773
Major focus for a secondary art curriculum should be on the use of commercially produced graphic software programs on the microcomputer.	25	37	18	10	7	2.351
Secondary art students should be able to assess the implications of using computer-generated images in our environment.	0	5	6	63	23	4.072
The ability to produce 3D graphics on the microcomputer is a necessary skill for secondary art students.	7	33	27	23	7	2.897
Art students should be able to program the microcomputer in order to effectively use it as an art medium.	2	24	22	32	17	3.392
An appreciation of computer-generated images in our environment and society is a topic that needs to be included in a secondary art curriculum.	0	4	9	61	23	4.062
The ability to recognize and describe the characteristics of computer-generated images is a skill art students need to possess.	0	9	17	54	17	3.814
Descriptions of hardware/software devices and their functions do have to be included as art activities for students.	0	27	19	37	14	3.392
Static 2D designs in high/low resolution graphics should be a skill exhibited by secondary art students on a microcomputer.	2	15	34	32	14	3.423
Art students should be able to define and use computer art terms.	0	4	14	61	18	3.959
The process of selecting appropriate software packages (i.e., Koala pad, Graforth) for personal artistic use should be a topic in a secondary computer art course.	2	2	20	62	11	3.804
Microcomputers and their exploration as an art medium should be included in a secondary art curriculum.	0	4	12	47	34	4.144

Table 4

Frequency Ratings of Perceptions of Art Supervisors Concerning Art Teachers'
Competencies in Dealing with Microcomputers in Art

COMPETENCY	NONE	WEAK	FAIR	GOOD	EXCELLENT
Computer art terms	16	51	12	3	0
History of computer art	35	43	10	0	0
Recognition of computer characteristics	18	57	13	5	0
Career vocations	31	42	12	3	0
Evaluation and selection of graphic software	39	42	5	2	0
Microcomputer visual problem solving	43	40	5	0	0
Drill and practice	39	42	6	1	0
Tutorial	43	38	7	1	0
2D design on microcomputer	37	40	9	2	0
3D design on microcomputer	50	43	5	0	0
Peripheral device usage -					
graphic tablet/pad	38	39	8	2	1
mouse (tracing stylus)	45	36	6	1	0
light pen	44	38	5	1	0
digitizer	50	34	4	0	0
Programming	48	36	4	0	0
Animation	52	32	4	0	0
N = 86					

Table 5

Number of Microcomputers per High School

CATEGORY LABEL	FREQUENCY	PERCENTAGE
None	7	8.0%
One	3	3.0%
2-5	15	17.0%
5-10	18	21.0%
Over 10	27	31.0%
Lab	14	16.0%
No Answer	2	2.0%
N = 86		

Table 6

Number of Teachers in School System Who
Provide Microcomputer-Based Art Activities

CATEGORY LABEL	FREQUENCY	PERCENTAGE
0	63	73.0%
1	7	8.0%
2	5	6.0%
3	2	2.0%
5	3	3.0%
10	2	2.0%
No Answer	4	4.0%
N = 86		

The problems that were identified were in six categories (teacher training, funding, lack of good software, curriculum development problems, accessibility/availability of microcomputers and space for microcomputers, and the role of the microcomputer in art). Table 7 indicates the frequencies of responses concerning these categories. Twenty-eight of the participants indicated a concern for identifying the role the microcomputer

Table 7

Problems Needing Attention in the Use of
Microcomputers in Secondary Art Classrooms

CATEGORY LABEL	FREQUENCY
Teacher training	8
Funding	15
Lack of quality software	6
Curriculum development	16
Accessibility/availability	11
Role of microcomputer in art	28
No Answer	13
N = 97	

should play in art. According to the participants, the roles included: (a) the microcomputer used only as a tool for other art media, (b) the microcomputer as a separate art medium, (c) the microcomputer used for CAI, and (d) the microcomputer used for teacher management.

The next two highest categories of problems were funding (15) and curriculum development (16). Once the role of the microcomputer has been identified, there appeared to be the need to address the problem of developing appropriate activities for art students. Many respondents felt art educators should determine these activities but indicated that presently most art teachers lack skills in this area. This lack of skill training is reflected in the identified problem of teacher training (8). Secondary art teachers need to be trained in microcomputer uses and skills in order to make intelligent decisions on its inclusion in the art program. Along with these concerns was the knowledge that availability of funds was lacking for curriculum development. Two reasons were often identified for this lack of funding. One reason was cutbacks in areas other than the basics, such

as art, would not allow for what respondents termed "extras" such as microcomputers. The other reason was general lack of awareness by administrators, who provide the funds, of the serious exploration of microcomputers in art and their creative potential.

General Discussion

The findings in this study led to the following conclusions:

- (1) Over three-fourths of the art supervisors and computer art experts agreed that microcomputer-based art activities should be included in the secondary art curriculum. Specifically, they supported the inclusion of the following topics in a microcomputer-based art curriculum:
 - the use of peripheral devices;
 - the evaluation and criticism of computer-generated images;
 - the creation of computer-generated images;
 - the identification of careers in computer graphics;
 - the appreciation of computer images and the ability to assess their use in the environment;
 - the definition of computer art terms; and
 - the selection of appropriate software for personal use.

Although participants were in agreement with the general inclusion of microcomputers in the art program, they disagreed with the necessity of including the following microcomputer activities in the program:

- computer animation;
- the use of commercially-produced graphic software programs;
- production of 3D graphics; and
- descriptions of hardware/software functions.

In addition, they also were primarily undecided on whether the following two areas need to be included in microcomputer art activities:

- computer programming, and
- production of 2D graphics.

- (2) Art supervisors reported that the majority of secondary art teachers lacked knowledge of computer-generated images and skills in art applications on the microcomputer.
- (3) Over half of the school systems surveyed had at least five microcomputers per school with many having lab facilities. The art supervisors, however, indicated that the majority of their art teachers did not provide any microcomputer art experiences.
- (4) The major problems that needed to be overcome so that the use of microcomputers could be successful in the secondary art classroom, in order of importance, were:
 - Identify the role of microcomputers in art;
 - Curriculum development of appropriate activities;

- Funding;
- Accessibility, space, and availability; and
- Teacher training.

The results of this study suggest that secondary art educators are receptive to and see the need for the inclusion of microcomputers in the art classroom. This finding contradicts the earlier held conception^{1-2,9} that newer technology was viewed by art educators with great apprehension and should be avoided in the arts. There is evidence, however, that agreement on certain content areas in microcomputer-based art is lacking. Implications from this study suggest that problems of agreement may be due primarily to lack of knowledge of and skills with microcomputers.

The results are also consistent with the ideas of writers^{10-11,12} in that there appears to be an absence of specific knowledge needed for the use of microcomputers in the art classroom. The successful use of microcomputers in the art class requires not only an awareness and understanding of the creative potential and avenues for exploration with a microcomputer but also an adequate ability to use it. To what extent did the participants in this study provide evidence of these levels of competencies? An answer to this question is provided by the competencies of the secondary art teachers and experiential data. Both competencies and microcomputer experiences indicate weak levels of abilities in the use and knowledge of microcomputer-based art. This means that the absence of these skills in secondary art teachers will influence the art activities that are planned and used in their classroom. The most revealing comparison of this fact is between the number of microcomputers per high school and the number of art teachers offering microcomputer art activities. The majority of art teachers clearly provide no experience although microcomputers are available.

However, many of the areas that the respondents felt should be included in microcomputer-based art activities do not require the actual use of the microcomputer. For example, the evaluation and appreciation of computer art and the identification of careers in computer graphics do not require hands-on utilization of the microcomputer. Although there would be additional information needed by art teachers to provide this content information, it would appear not to be as extensive as actual experience in using the microcomputer. The results suggest that art departments that do not have microcomputers available or funds for their inclusion could provide this type of content without the aid of a microcomputer. These solutions could remedy some of the problems (funding and availability) that participants felt needed to be addressed before microcomputers were included in the curriculum.

The findings of this study also illustrate that computer art may simply be such a new area that inconsistencies do exist concerning appropriate content for art curricula. The research lends support to the contention that art educators lacked understanding of microcomputer-based art in general. An example may illustrate the importance

of acquiring specific skills in microcomputer-based art by art educators. Discrepancies were apparent in responses to the need for students to produce computer-generated images. Although the general consensus was that 3D graphics and computer animation should not be included and that the role of 2D graphics was uncertain in a microcomputer art program, the majority of individuals responded favorably to the creation of computer-generated images. These results are inconsistent with the fact that the only types of images that can be created on the microcomputer are 2D designs, 3D images, and computer animated sequences. A secondary explanation for these inconsistencies may be in the structure of the survey itself. Individuals may have interpreted survey questions differently which could account for this fact as well.

One important result of this research was the participants' identification of areas that needed to be overcome in order to implement microcomputer-based art activities in the secondary art program. Data from the open-ended question revealed the need and importance for educating school administrators about the creative potential microcomputers have in art. Without this awareness and subsequent support by school officials, art educators will have difficulty incorporating microcomputers into the art curriculum.

Another complication to the use of microcomputers in the art program suggested by the data seems to be the lack of accessibility of microcomputers or the space for them in the art room. In many cases, the locations of microcomputers in high schools are in the math and science areas. The open-ended question revealed that art departments do not always have access to them. Results also revealed that art teachers would have difficulty providing space for the equipment in their existing classrooms. Perhaps secondary art educators should consider the possibility of developing a mutually agreeable user schedule for microcomputers in their schools so that they (art teachers and students) could gain access to them.

This study set out to investigate the content areas that should be included in a secondary microcomputer-based art curriculum. What should be clear, however, is that inconsistencies and discrepancies in the respondents' positions on several topics existed, strongly lending support to the contention that lack of agreement on topics may in fact be merely lack of understanding of microcomputers in art. This lack of understanding may be due to a lack of tradition in a new area of art exploration. Thus, art educators must begin to learn how to create microcomputer-based art themselves. In this way, familiarity with and understanding of the medium will develop and the formulation of a tradition in microcomputer-based art will begin.

REFERENCES

- [1] D. White, "Advanced technology, art and art education: Reaching toward the third millennium," *Art Education*, vol. 36, no. 3, pp. 8-10, 1983.
- [2] S. Madeja, "The art curriculum," *Art Education*, vol. 33, no. 6, pp. 24-26, 1980.
- [3] L. Ettinger and M. Rayala, "Computers in art education," *The Computing Teacher*, pp. 24-28, 1982.
- [4] M. DiBlasio, "If and where to plug in the computer: A conceptual framework for computer assisted art instruction," *Studies in Art Education*, vol. 25, no. 1, pp. 39-47, 1983.
- [5] L. Briggs (ed.), *Instructional Design: Principles and Applications*. Englewood Cliffs, NJ: Educational Technology Publications, 1977.
- [6] R. Kaufman, *Needs Assessment: Concepts and Applications*. Englewood Cliffs, NJ: Educational Technology Publications, 1979.
- [7] E. Eisner and D. Ecker, "Some historical developments in art education," in G. Pappas (Ed.), *Concepts in Art and Education*. Toronto Canada: MacMillan Co., 1970 (pp. 12-25).
- [8] Jacques Cattell Press (Ed.), *American Art Directory*. New York: R. R. Bowker Company, 1982.
- [9] V. Lanier, "Newer media and the teaching of art," *Studies in Art Education*, vol. 5, no. 1, pp. 10-19, 1963.
- [10] B. Jones, "Art, music and humanities," Joint issue, Education Board, *Association for Computing Machinery*, pp. 71-75, 1981.
- [11] A. Phelan, "The impact of technology and post modern art on studio art education," *Art Education*, vol. 37, no. 2, pp. 30-36, 1984.
- [12] D. Greh, "Art education in the third wave," *Art Education*, vol. 37, no. 2, pp. 40-41, 1984.

Red Network: A Conceptual Art Piece Using Microcomputers

George K. Shortess

Department of Psychology CU#17, Lehigh University
Bethlehem, Pa. 18015

Abstract

I describe an installation using sound producing microcomputers in a public place and provide an evaluation of the reaction to the piece as well as some comments on the use of microcomputers in environmental/conceptual art.

Introduction

As discussed elsewhere my art work has developed as ways of expressing the formal features of the human nervous system (Shortess, 1983)². Briefly I have used the properties of the nervous system as the basis of my art in much the same way as any artist uses the features and ideas of his/her subject matter to create images in whatever way they are defined by the artist. In my particular approach, I am not trying to create literal nervous systems but

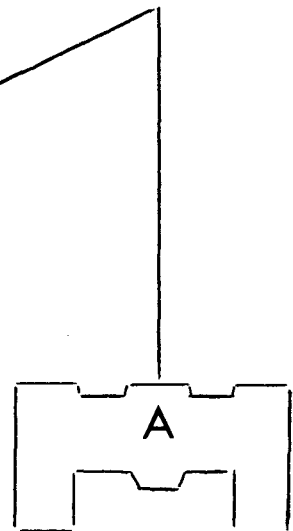
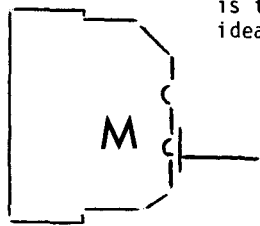
interactive devices that behave in some ways like nervous systems. One effect of these pieces is to create for the viewer the idea that neural activity is an essential part of the art experience. For these purposes the interactive function or behavior of the system is more important than the visual appearance although that too is,

other hand they continue to generate ideas about neural activity, as well as neural activity in the viewer, for as long as the viewers and pieces can interact. They are not so much conceptual in the act of creation (as with much of performance art), but they are conceptual to the extent that they successfully ask questions about the idea that the nervous system is the essential art object.

Within this general context, I will describe my installation piece, Red Network, and provide some comments on a questionnaire that evaluated the exhibition, as well as some general comments on the use of computers in this kind of art.


The Installation

This installation was part of a larger series of exhibitions on Art and Technology sponsored by the Lehigh University Galleries that were in place from September 11 through October 30, 1983 at the University. Galleries in two separate buildings housed the main exhibitions. There were three other components of interest, aside from Red Network: 1) an exhibition of offset prints in Building A, 2) a survey of the history of art and



of course, part of the piece. But it has been the ability of microcomputers to behave interactively that has led me to use them in my work.

In this sense then my work is a form of conceptual art in that it is driven by ideas about the nervous system and asks questions about the nature of art and the role of neural activity in understanding art ideas (see for example, Kosuth, 1969)¹. However, it is not conceptual in the sense that there are physical objects that outlast my creative act. On the

Figure 1. Map of the connecting pathway between the two buildings, A and M.
Scale  100 feet

technology called "The Aesthetics of Technology: From the Crystal Palace to the High Frontier" in Building M, and 3) an exhibition of shells and other natural forms called "Nature's Technology" in Building M as well.

Red Network provided a conceptual link between the two buildings (see Figure 1). For this installation I placed on the floor in the lobbies of both buildings aluminum covered boxes from which extended heavy red cord attached to the ceiling (see Figure 2). All boxes were 15 inches high and occupied from 360 to 650 square inches in various rectangular shapes. The cord also extended outside along the walkways connecting the two buildings, a distance of over 1000 feet. Outside the cord was placed in noncontinuous sections supported by aluminum stakes driven into the ground so that the cord was about a foot above ground level. At the ends of each section, the cord followed the stake into the ground and was anchored below the ground level. This gave the impression that the cord continued underground between the above-ground sections. As the exhibit continued (over a period of six weeks), the number of stakes was reduced, making the connection more conceptual and less literal. In each lobby a small sound generating computer (a Commodore VIC-20) was housed inside the aluminum covered boxes. Pressure sensitive pads were placed under the foot mats inside the doors and connected to the computers. As people entered and left the buildings there were bursts of sound from the boxes. Often a group of people leaving at the same time would produce sequences of the bursts by the chance spacing of their walking patterns. In addition, by alternately stepping on and off the pads, one could

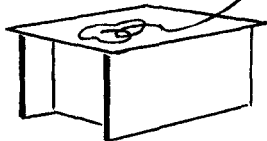
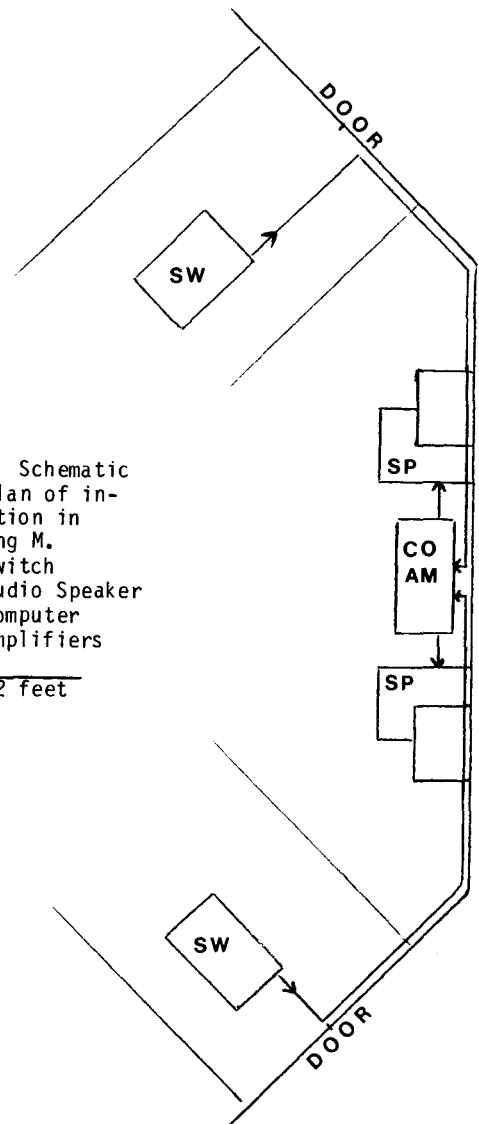


Figure 2. One of the aluminum covered boxes with cord.

Figure 3. Schematic floorplan of installation in Building M.

SW- Switch
SP- Audio Speaker
CO- Computer
AM- Amplifiers
Scale 2 feet



deliberately create sequences of these bursts to form unique sound patterns. These two locations served as the two main nodes with the red cords, as the connecting elements of the total network structure, binding the two exhibition spaces together.

The floor plan for the Building M node is shown in Figure 3. The node in Building A was similar except that there was only one pressure sensitive pad and only one speaker, since the available space was smaller. The joystick inputs to the VIC-20, which is a pair of switches, were simply connected to the pressure switches placed out of sight under the regular door mat runners in the entrances. The connecting wire was run in an unobtrusive way around the edges of the space. The program in the computer kept looping and looking for a change in the state of one of the switches.

When a change occurred in either switch a subroutine was called that generated a sound burst associated with the particular switch that closed. Both switches produced eight notes in succession. A typical sound burst for each switch is shown in Figure 4. The tone for one



Figure 4. Note sequences generated by the sculptures

switch was lower than for the other. The tones for each burst were always the same and were all about 16 msec. in duration but the time intervals between notes of the burst varied each time the mat was depressed. The ranges of intervals were from 20 msec. to 220 msec. for switch 1 and from 20 msec. to 170 msec. for switch 2. The same program, connected to switch 1 only, was used in Building A. The particular sequence of intervals varied in a quasi-random order as determined by the random number generator of the computer. After the burst it again evaluated the switch. If it remained closed the program would not return to the main loop. This required that the participants keep moving around on the mat in order to continue to generate sounds and provided a safeguard against the sound going on continuously if a broken switch remained closed, although the pressure switches continued to operate throughout the exhibit. In fact, there were no reliability problems in the system at all. While the system was looping and looking for a switch closure, it would generate a single note of 16 msec. duration and with the tonal value of high C. The interval between occurrences of this single note was between 0.6 sec. and 18 sec. This served to indicate that the system had a will of its own and was not simply driven by the traffic through the buildings.

The Reactions

The reactions of viewers to the piece included indifference, genuine liking and disliking as well as an unsureness about the work as "art." Some people, I am told, talked to it when they would go by, others played on the mats to create their own sound sequences while others never noticed. However, the computer-generated reactivity seemed to be the most engaging part of the piece.

The outside stakes with the red cord suffered various forms of nonverbal abuse. They

were bent, pulled out, pushed over and taken away. During the daily maintenance I also changed the exact placements and reduced the total number of stakes over the period of the installation partially in response to the abuse.

In addition some of the students got involved with the work in a variety of public ways. There was an editorial in the campus newspaper and towards the end of the show a group of students built a sculpture on the lawn in front of Building M, using pieces of the red cord and stakes of Red Network together with a supermarket shopping cart. It was dubbed "cart art."

After the exhibitions were over, a telephone survey of 110 students was carried out to gauge the impact of the exhibitions for the purpose of improving the exhibition program at the University. I can provide a complete description of the results to anyone interested. Here I will simply summarize them, particularly as they pertain to Red Network. First there was a problem in obtaining a random sample of the Lehigh student population. The sample obtained was overrepresentative of sophomores and underrepresentative of seniors so that uncritical generalizations to the general student population cannot be made. Within that limitation, however, the results from the sample indicate that there were significant differences in student awareness for the exhibitions. The students were most aware of Red Network (65% of the 110 students) while 20%, 28% and 39% of the sample were aware of the other three exhibitions. In addition there were 22 students who were only aware of Red Network and not the other three exhibits while for each of these other exhibits 0, 4 and 6 students respectively indicated awareness of but one of them. This suggests that Red Network was more salient than the other exhibitions. This is consistent with the newspaper editorial and the "cart art" work created by the students, since none of the other exhibitions was responded to in these ways. Whether this salience was due to the interactive quality, the fact that it was located in several areas of the campus or some other reason cannot be determined. Future work may be able to answer these questions about computer controlled interactive art works. One other result emerged. About half the students who were aware of each of the exhibitions reported reacting positively towards it while the other half was either negative or indifferent. There were no differences in this measure across exhibits. Therefore the salience of Red Network did not seem to result from an overriding liking of it, although it should be emphasized that these results are suggestive rather than conclusive.

Comments

I will not attempt an evaluation of the overall success of the piece since I do not have the data with which to do it. Certainly for some viewers the piece did work well. For

others it was something meaningless. As an experiment it was clearly valuable to me as the artist. It demonstrated possibilities for the use of computers in nontraditional forms that allow art, as an open concept, to develop in ways that are at least improbable without the computer. Its value, in addition to its interactive ability, is that it behaves in highly repeatable ways and the artist can specify the rules by which the program works, or change the rules and experiment with the parameters of the program. The microcomputer is also relatively inexpensive, is convenient, fits into a small box and is portable. It is an art form that does not depend on high cost and state of the art equipment.

But when and how does one use a computer, or any other medium for that matter? This seems to me to be one of the most important parts of the process of artistic expression and in the final analysis each artist must answer these questions for her/himself. However, there seems to me to be a guiding principle that is often lost when computers are involved. In the search for higher resolution, more powerful machines and, by some technical criterion, better and improved systems, the technical often drives the creation. For me the making of art should be driven by the art ideas and not the technology, unless of course that in itself is the idea. The technological medium provides the limits of the possibilities but does not provide ideas of art. True, there may be interactions between the medium and the artist that are helpful to the artist in understanding the technology and what it can do, but that is not artistic creation. In the case of Red Network, I wanted to make a statement about connections among a group of art exhibits that reflected the metaphor of the nervous system. In addition, I wanted the piece to be placed in particular public areas without a great deal of security and areas that were not well laid out for connecting a piece into major computer systems. This seemed to be an ideal application of the small VIC-20 microcomputer. But it was the idea of the neural metaphor that came first and then the search for the appropriate medium. It is this principle that sometimes gets lost in the rush to bigger and better machines.

From this it follows that since one of the qualities of the machine is as a processor of information, arts dealing with ideas about information structures, information transmission and other information processing activities find computers a natural medium to use, although certainly not the only one. Since my art work is dealing with ideas about the nervous system, which is an information processing system among other things, the computer can provide a means to express the neural metaphor. In addition, the computer is a powerful medium for looking at the conceptual basis of art. By helping to move art out of conventional molds, it is a way of extending the definition of art. In the case of Red Network, I have combined these two ideas by

asking about the relationship of art to neural responses. In this context then, is the art work the collection of boxes, red cord, computers, wires, switches, etc? Or is it the interaction that occurred while it was in place? Or is it the idea that the essential art work is a pattern of neural activity which exists in some people's neural memory? Different people will answer these questions in different ways and I will not presume a correct answer. As the artist, the exploration of the means of asking such questions is the exciting activity of creation. Microcomputers provide another way of doing just that, but a way that is uniquely suited to the task. In addition to these sound sculptures I have been developing programs, using graphics and text that are more language based. They make statements, ask questions and require viewer responses. This is another of the many potential ways to use the computer medium to express artistically, ideas about neural function and the arts

References

1. Kosuth, J. Art after philosophy. Studio International, 178, 134-137, 1969.
2. Shortess, G.K. Neural Art: Works based on concepts of the nervous system. Leonardo, 16, 306-309, 1983.

COMPUTERS AND POETRY

Nora Wilson
4784 Crestline Ct SW
Wyoming, MI 49509

Walter Wright
School of the Arts
Virginia Commonwealth University
325 N Harrison St
Richmond, VA 23284-0001

In this paper we will discuss computers as a means of presenting poetry and in our discussion address the more general issue of aesthetics and computer graphics.

INTRODUCTION

Nora writes poetry. She doesn't like computers. Computer generated poetry doesn't impress her and, frankly, it doesn't impress me either. My specialty is computer graphics. Approximately 4 years ago I helped design an artists' work station. One of the images I created on this system inspired Nora to write Gretchen Prisoner of Love. More images followed and soon we had our first slide/tape show.

We thought about distributing our creative effort. Traditionally poetry is distributed through the print medium in books and literary journals. Sometimes it is accompanied by illustrations or photographs. We even have concrete poetry which is its own illustration. Unfortunately, people would rather watch TV than read a book of poetry. A slide/tape show is not the most portable of media and not everyone has a slide/tape synchronizer in the den.

Why not make poetry available on TV ? We tried videotaping our creation but this ended in frustration. The sound track was left off our copy rendering it useless.

Why not translate Gretchen Prisoner of Love from slide/tape to floppy disk ? We tried putting it on the computer and herein lies our tale.

AESTHETICS

Gretchen Prisoner of Love is a prose-poem. In the original slide/tape version each of the 88 lines of text corresponds to an image. The relationship between text and image is one of text and context. The image creates a context for interpreting the text. In turn, the text gives meaning to the image. This relationship raises the issue of aesthetics.

"Aesthetic" comes from the Greek for "sense of perception". Webster's defines aesthetic as that which relates to sensuous cognition involving pure

feeling or sensation especially in contrast to reasoning or rationalizing. "Aesthetics" seems to have acquired a related but opposite meaning as a set of rules for judging the worth of a piece of art. This second definition opposes the first because it emphasizes reasoning over feeling. And herein lies the failure of "computer art".

In general, computer art appeals to the intellect not the emotions. We hear something like, "That's a nice piece of computer art because...", rather than, "That piece makes me feel...". This distinction is important. Computer art which illicit the first response will never be anything more than "a nice piece", but that which illicit the second response has real potential. In Gretchen Prisoner of Love we are attempting an immediate sensuous experience; we are attempting to illicit the second response.

To put it another way, poetry appeals to the emotions, this is a given. Computer graphics appeals to the intellect, this is a mistake.

In order to avoid this pitfall, we needed a presentation sufficiently complex so that it defied immediate rationalization. We tried to do this in several ways. An image can be made complex; its relationship to other images in the presentation can be made complex; and, finally, the relationship between the images and the text can be made complex. We have now returned "full circle" to our discussion of text and context. Let's wrap it up by paraphrasing Susanne Langer-art is the appropriate relationship between that which contains (context) and that which is contained (text).

WHAT YOU CAN DO WITH YOUR COMPUTERS

Now for Nora's two cents worth. Anyone who has been paying attention the last few years knows the computer has become a tremendous aid to writers. Word processing makes editing and revising faster and easier, and more and more books and articles are composed and arranged on word processors. There is another kind of writing however, that is still best done by the slower method of pen on paper: poetry and lyric prose.

In well-written poetry and prose-poetry, every word is carefully considered, chosen and placed

for its meaning, impact, and relationship to the whole. As the creative process unfolds, hours or even days may pass before the right word or group of words is found for a particular placement. If a phrase comes to the writer in the middle of the night, she does not get up and power up the system, find the disks and programs she's using, and insert the phrase. She writes it down in the notebook next to her bed.

In the original slide/tape production of Gretchen, the lines were written to accompany the images. The reading, or sound track, was an important part of the presentation. In the computer version, different (though similar, when possible) images were created to accompany the script, which, no longer heard, has to be read off the screen. The movement and placement of the words became an important part of the visual images.

The computer as a means of presenting poetry and images is probably not an obvious application of the technology, but a computer disk is definitely an appropriate vehicle for Gretchen's narrative. Certainly there are worse things you could do with your computers.

THE PRESENTATION

The original slide/tape format uses approximately 90 slides synched to a prerecorded audio cassette. There is a line of text for each slide, an inaudible pulse at the end of each line advances the slide tray. Timing or pacing is determined by the narration. The original images included real photographs, digitized video images and computer generated images with a resolution of 378 x 241 x 16 of 4096 colors.

And we wanted to do this on an APPLE !

The new computer format uses 32 images presented with text on the APPLE's high resolution screen, so-called, having a resolution of 280 x 192 x 6 colors. No real photographs, and the digitized video images are limited to 4 levels of grey. Timing is determined by programming.

The script was reorganized; lines of text were grouped to correspond to available images. In the outline that follows, the images are numbered 1-32. The "&" indicates that the screen is cleared and corresponding lines appear on a blank screen. The "+" indicates that the last lines are cleared before the new lines appear. Because there are only 40 characters per line, some of the original lines are broken. The "/" indicates a pause inserted in order to control the speed at which the line is read, that is, to control pacing for variety and emphasis.

Here is the reorganized script-

1.

Gretchen/

2.

Prisoner of Love/

&

All my life I've struggled/
to find my own identity./
Who am I?/

3.

When I cut through the facade,/
through the layers I present/
to the outside world,/what is left?/

4. (text window)

5.

I am a woman./

+

More than anything,/I feel./

6.

Emotions/are the driving force/
in my life./And why not?/

&

Does anything else really matter?/

7.

We all go through our daily routines/
doing what we need to do/
to get by in the world./

+

We think our activities are important./
We believe we have chosen our paths,/
set our own goals./

8.

We believe we are masters/
of our own fate/because no one/
wants to believe/in a random universe./

&

But do we really make our own decisions?/
Do we really have a choice?/
Perhaps not./

9.

Our life form has evolved/
to its present state/
because our bodies carry/

+

genetic material/whose sole purpose/
is to reproduce itself./

10.

Is that why/I can only identify myself/
in relation to another./

11.

Lowell is a cad./
He takes shameless advantage/
of my love./But I can't give him up./

&

I tried to fight it./
I told myself/
it's useless,/

12.

He'll/only/break/your/heart-/

13.

cut/it/into/pieces./

14.

But he drew me/
to him like/
a magnet,/

+

calling for me/
in my dreams-/

+

GRETCHEN/ (repeat 4 times)

+

I couldn't resist./

15.

Lost in his arms,/nothing else mattered./
His kisses were /like magic,/

16.

his touch/like none I'd known before-/
so gentle,/yet so strong./

&

How can I describe/
what happened in my mind/
when we touched-/

17.

the images/so strange,/colors/so vivid,/

18.

shapes and patterns so real,/
beyond the realm of illusion./

14.

Oh Lowell,/
will you never care for me/
as I care for you?/

19.

It's not not that Lowell is cold/
or unfeeling./He loves his work./

20.

He spends most of his time/
at the computer,/devoting his life/
to information processing./

21.

And this he does/in the name of art-/
because he's also devoted to art./

&

He says,/

+

"Because of the dichotomy/
between art and science,/
artists should take over/
the technology."/

+

Oh I know he's right./

22.

Think of how technical minds/
have misused technology./

23.

How science has abused humanity/
and endangered the future/of our planet./

24.

But Lowell is into/
computer augmented creativity./

+

There are worse things/
a computer scientist can do./

&

Too bad/
computers bore me/
comatose./

25.

My mind goes plaid/whenever he talks/
about the system output/
of his algorithmic strategy./

14.

Can't he see/
what he's doing/
to me?/

+

There must be a way/
to make him notice./

+

A way to get him/
to leave his machine./

&

I needed a/PRIORITY INTERRUPT./

26.

I tried to make him jealous./
I went out with other men./
It didn't faze him./

27.

I flirted blatantly with his co-workers./
He was pleased/
I found his friends so charming./

&

"I want you to be free," he said./
And I want to be free./
but is it possible/

28.

Could sheer will ever harness the energy/
of sociobiologically influenced/
electrochemical synapses/

29.

in a woman born in scorpio sun/
with capricorn rising?/

30.

I do what I need to do/
in my daily routine./
I have chosen my path,/set my goals./

&

But do I make my own decisions?/
Do I have a choice?/

31.

Will I always be/possessed by/passion,/
enslaved by/lust,/a prisoner/of love?/

32. / (final image, no text)

&

Written by/
Nora Wilson/
Programmed by/
Walter Wright/

&

A Lofty Thoughts Production/
Copyright 1984./

STRUCTURE AND COMPLEXITY

Now for Walter's two cents worth. As I worked with Nora's script I became aware of subtleties previously missed, a craftiness lurking between the lines. On the surface Gretchen Prisoner of Love is a simple tale of unrequited love. There are two principal actors: Gretchen who is a woman, and Lowell who is a cad. Gretchen identifies herself in relation to, hopefully, Lowell. He on the other hand seeks fulfillment in his work. But there's more than romance here.

There is the conflict between natural systems such as biology (Gretchen is something of an amateur sociobiologist), and man-made systems or technologies: between feelings and the machine. And there is the conflict between art and science. So we have several conflicts interwoven in the script. Each a simple confrontation of opposites; but together a complex structure which approximates reality. For example, Gretchen and Lowell may not be seeing eye to eye but, at least, they're both for art as opposed to (mis)applied science.

The writer weaves together a number of simple themes to produce a complex structure. Nora has done this not only on the level of plot, but with voice, grammar, and syntax. Here is Gretchen (the amateur sociobiologist) pondering her predicament:

"Our life form has evolved to its present state because our bodies carry genetic material whose sole purpose is to reproduce itself. Is that why I can only identify myself in relationship to another?"

Or, consider this gem:

"...shapes and patterns so real, beyond the realm of illusion."

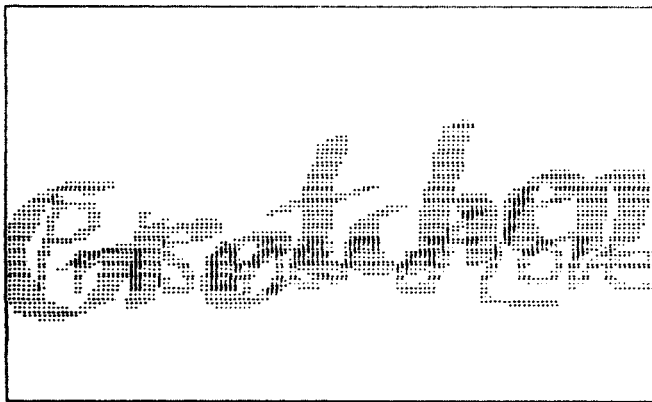


Figure 1 Titles

The visual artist must also weave together a number of simple themes to produce a complex structure in support, or sometimes in opposition, to the text. I used three types of images corresponding to the three voices used in the text; real (digitized video) images corresponding to the narrative voice; illustrative in response to the use of cliches; and abstract images in response to the more poetic passages.

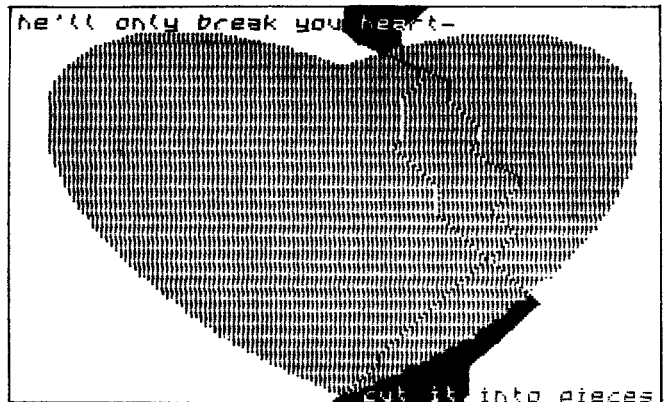


Figure 2 A Broken Heart

The title- "Gretchen/Prisoner of Love"- is an example of illustration, Figure 1. The title is spelled out directly, an obvious one to one correspondence. Further, the phrase "Prisoner of Love" is imprisoned inside "Gretchen"; indicating that the prison isn't outside Gretchen, the prison is Gretchen. Another example of illustration is the breaking heart, Figure 2.



Figure 3 Gretchen



Figure 4 Gretchen and Lowell



Figure 5 Lowell



Figure 6 Gretchen Thinking

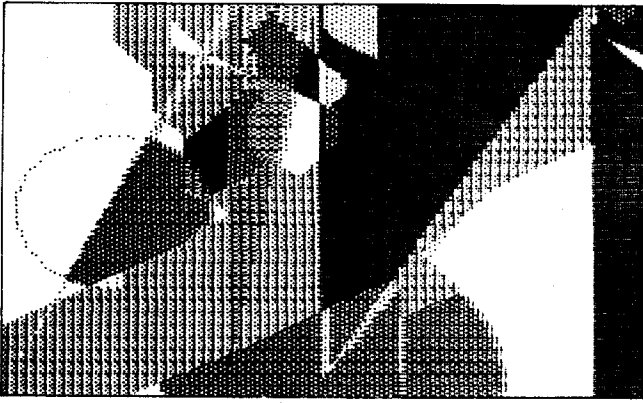


Figure 7 Emotions

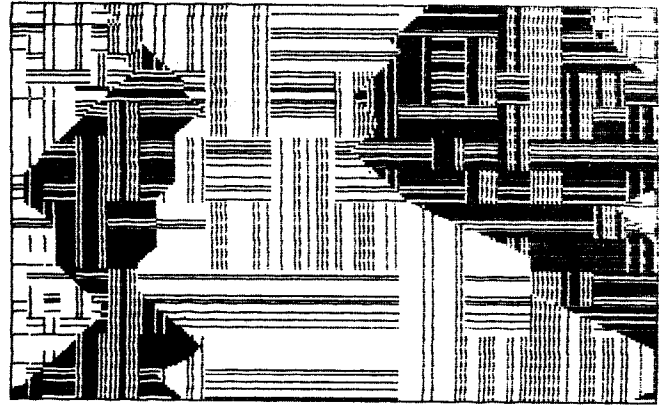


Figure 8 Our Daily Routines



Figure 9 A Random Universe

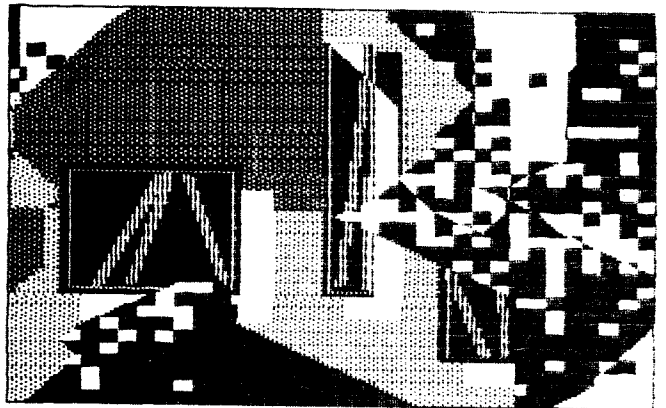


Figure 10 In The Name of Art

Real images, Figures 3-6, are used to introduce Gretchen- "Who am I?"- (and there she is) and Lowell who first appears in a composite image- "...in relation to another."- a visual pun, and then on his own (the cad). There is one particular image of Gretchen, she's in the lower half of the frame, used at three points in the script where the narrative becomes highly introspective. Her thoughts appear cartoon-like above her head. And at one point Lowell's voice appears (in inverse type) to summon her away. Here we have something like counterpoint; the real images become a cliché.

Abstract images, Figures 7-10, are used to convey feelings and concepts. The first pure abstract appears with the line- "Emotions are the driving force in my life" . A knife-like triangular (negative) shape cuts the picture and overlaps the previous image of the body. It points to the text for added emphasis. Another type of abstract is used with the phrase- "...our daily routines..."- and appears again with- "...my daily routine." In this image groups of orderly horizontal and vertical lines are torn apart into areas of

contrasting color indicating the usual outcome of well laid plans. Some abstract images are (or contain) visual puns; for example the image which accompanies the phrase- "...a random universe."- is made from random pieces of images used elsewhere in the presentation, and with- "...in the name of art..."- is an image containing, what appear to be, little framed canvasses.

Many images are combinations of these three basic types- real, illustrative and abstract. The statement- "I am a woman."- resulted in a composite image made from two real images, Figure 11. The dominant figure is assertive like the statement; the second figure appears to be emerging from tears in the first and changes the original assertion to a question- "I am which woman?". These same two figures are combined again near the end of the presentation; they are in better balance. The final image, Figure 12, combines illustration, realism, and abstraction. Gretchen, as she first appeared, is combined with the unbroken heart, however, she is still a prisoner as suggested by the vertical bars which break up the image. She is still a prisoner of love.



Figure 11 I Am a Woman

OK, I've suggested ways in which these three types of images were woven together to form a complex structure which supports through correspondence and opposition the equally complex structure of the text. This structure is one of expression; and expression is only one dimension of visual imagery. Just as the writer (bless her heart) works with grammar and syntax, so does the visual artist. Composition is a second means of structuring visual imagery, a second dimension. Again taking the three types of images- real, illustrative and abstract- let's look at the order in which they occur in the presentation.

Group 1: the titles constitute a single illustration, followed by one real image, four abstract images, and another illustration. This group of seven images introduce Gretchen and the concept of genetic determinism.

A bridge: two realistic images serve to introduce Lowell and form a bridge to the second major grouping of images.

Group 2: again this group starts with an illustration- the breaking heart- followed by one real image, four abstract images, and the same real image repeated. In this group of seven images Gretchen driven, we suppose, by genetic determinism pours out her heart.

Group 3: immediately we launch into the third and last major group of images. Four abstract images, one real image, one abstract image, four real images, and four abstract images. This group is obviously different; there are fourteen images- twice as many as groups 1 and 2; the order of subgroups established in 1 and 2 is reversed. We are introduced, in the first subgroup of four images, to the themes- feelings versus the machine and art versus science. In the second subgroup of four images Gretchen confronts Lowell directly, attempting to lure him from his machine. This is stage two of the standard "hollywood plot" which has three stages- crisis/ conflict/ resolution. But there is no resolution, in the last subgroup of four images we have a recapitulation of the original romantic conflict; and the question-

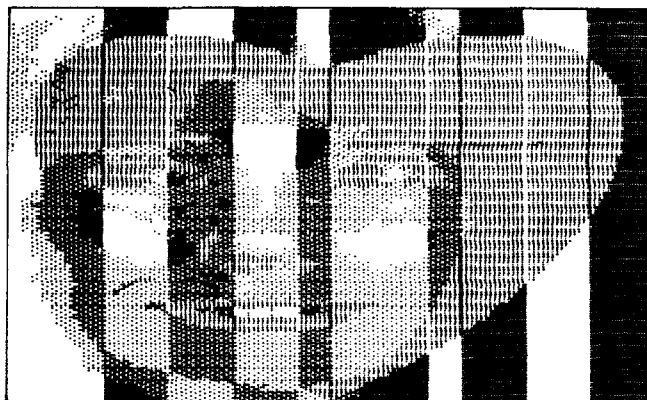


Figure 12 A Prisoner of Love

"Will I always be...a prisoner of love?"- is left unanswered.

A summary: the final image combines illustration, realism, and abstraction. The unanswered question reverberates through this image.

Let's look at the order of images in its simplest forms- "I" represents an illustration, "R" a real image, "A" an abstraction, and "/" divides the major groupings-

IRAAAAI/RR/IRAAAAAR/AAAAARRRRRAAAA/I

And as numbers-

1141/2/1141/41144/1

OK, here we are at the opposite end of the structuring continuum- from expressive to mechanical; informal to formal; or, as Gretchen might put it, from feelings to the machine. The question we must ask now is-

Is this a suitable structure, an appropriate container (context) for that which is contained (text) ?

And, of course, I think it is. Here's why. The major groupings of images correspond directly to the development of themes within the plot, as outlined above. The ubiquitous rule of threes is basic to the "western aesthetic". Parts 1 and 2 are identical, in a formal sense, leading us to predict Part 3. But no, Part 3 inverts our expectations; elaborating and extending the structure of Parts 1 and 2. Literature, hollywood films, and pop music- just to mention a few diverse examples- use similar means of structuring and form(al) development. Not wishing to belabor what is meant to be but one example of structuring, I have a final observation. The number sequence above, if transposed into musical notes and intervals in a minor scale, would make a nice lament...

The next several sections are technical in nature. They describe the hardware and software used in bringing Gretchen Prisoner of Love to the computer screen. We will look at assembly language utilities, higher level language programs, and other graphic systems used in producing and orchestrating text and images. These are the building blocks for the presentation system. Discussing them not only satisfies a programmer's compulsiveness but will help to define the term "complexity".

GRAPHIC UTILITIES

VCUtilities is a package of assembly language graphics routines developed for an advanced computer graphics class at Virginia Commonwealth University, School of The Arts. It includes -

LAD - calculate line address.
 SAD - calculate screen address.
 & - fast clear.
 % - zap phase bits, all black bytes cleared.
 S! - clear screen to 1 of 32 color patterns.
 P! - draw a pixel.
 X! - draw a pixel in XOR mode.
 V! - draw a vector.
 HL - draw a horizontal line.
 VL - draw a vertical line.
 R! - draw a rectangle.
 A! - draw a filled rectangle.
 T! - text drawing in XOR mode.

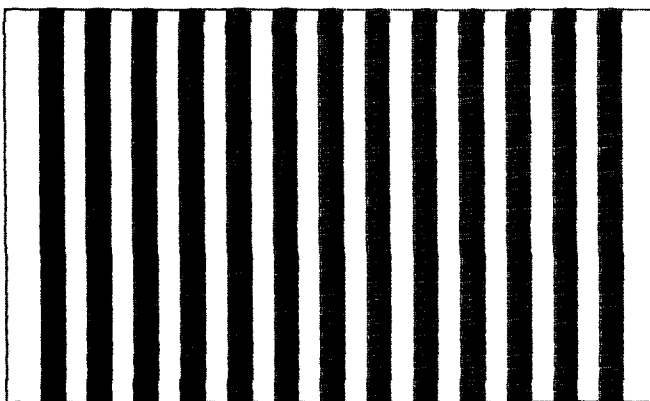


Figure 13 Vertical stripes on the current screen.

I< - digitize a video image.
 S> - save image from current screen to alternate screen.
 S< - load image from alternate screen to current screen.
 P> - pack current screen to buffer.
 P< - unpack buffer to current screen.

My APPLE has a 294K Synetix RAM card and a MicroWorks video digitizer. The RAM card gives me 32 alternate high resolution screens. This is handy because I program in TransFORTH which leaves only 1 screen on the APPLE. My version of VCUtilities uses the 1 available screen on the APPLE as the current screen and the additional 32 screens on the RAM card as alternate screens. Images are loaded from alternate screen in REPLACE, XOR, OR or AND modes. Consider for a moment two images, vertical stripes on the current screen and horizontal stripes on an alternate screen. Figures 13-17 shows the current screen before and after loading the alternate screen using the 4 logic modes.

The XOR mode is used in text drawing because it allows text to appear over any background and because text can be erased by redrawing. Understanding these 4 logic modes is important to any graphics programmer because they contribute in large part to the unique character of computer generated images. More later.

VCUtilities includes packing and unpacking routines. Packed images take up less space in memory and on the disk. All 32 images used in Gretchen are stored on the back side of the program disk. Finally, I should mention that VCUtilities was developed using the ALD II system written by Paul Lutus. And while we're on the subject, Paul Lutus also wrote the TransFORTH language used to develop the presentation system that made Gretchen possible.

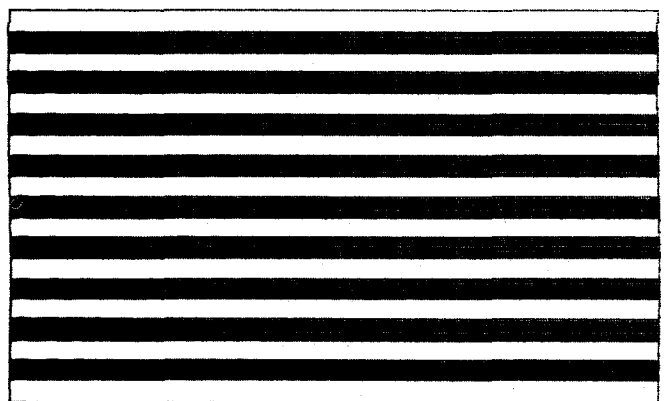


Figure 14 Horizontal stripes loaded in REPLACE mode.

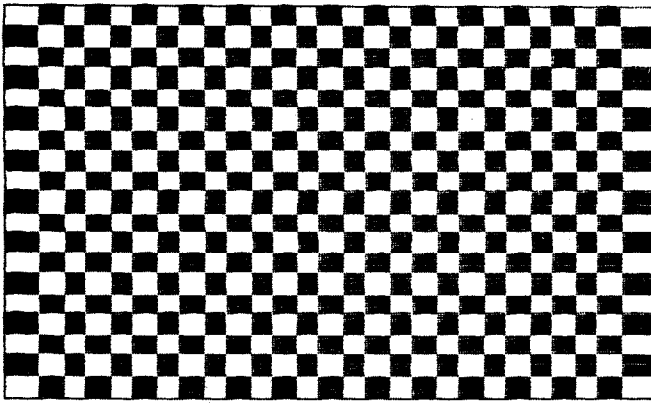


Figure 15 Horizontal stripes loaded in XOR mode.

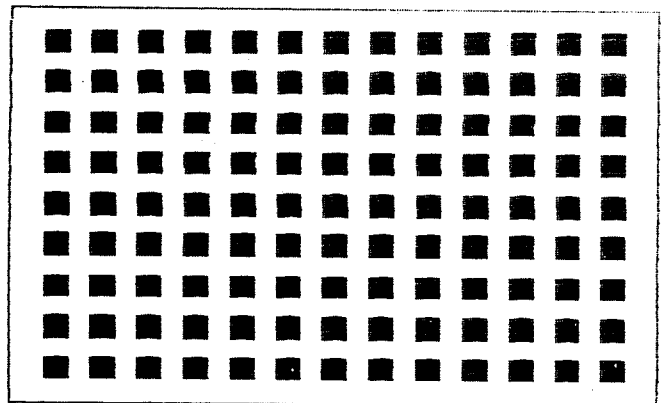


Figure 16 Horizontal stripes loaded in OR mode.

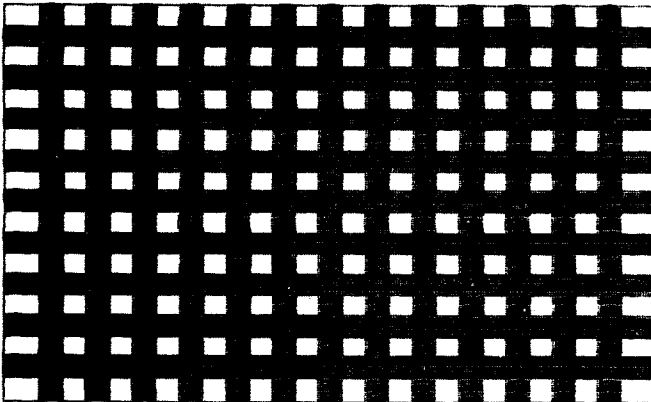


Figure 17 Horizontal stripes loaded in AND mode.

GRAPHIC PROGRAMS

First a brief explanation of the FORTH language. FORTH routines are called words and each word is added, in turn, to the system library. New words are defined using words already in the library. A FORTH system is built up or extended for each application. Therefore each application is its own FORTH system with its own library of specialized words. FORTH uses a data stack, parameters used by a word are placed on this stack before the word is invoked. For example, the following line of code

```
2 2 +
```

adds two and two. The syntax is called Reverse Polish Notation, the parameters are "2" and "2", and the word is "+". The result "4" is placed back on the stack.

The words or programs described below were added to the TransFORTH library to make the special graphics system used in producing Gretchen. These additional words or programs include-

HGR - display high resolution page 2.

- C! - set color pattern, takes 1 number (0-31) from the stack.
- & - fast clear.
- % - zap phase bits, all black bytes cleared.
- S! - clear screen to 1 of 32 color patterns.
- DATA - set xy coordinates.
- P! - draw a pixel, takes 2 numbers (X0,Y0) from the stack.
- X! - draw a pixel in XOR mode, as above.
- V! - draw a vector, takes 4 numbers (X1,Y1,X2,Y2) from the stack.
- R! - draw a rectangle, as above, where X1,Y1 define the upper left corner and X2,Y2 define the lower right.
- A! - draw a filled rectangle, as above.
- T! - text drawing in XOR mode, takes 2 numbers (Column,YO) from the stack and retrieves a line of text from the TransFORTH buffer PAD.
- I< - digitize a video image.
- L! - set logic mode, takes 1 number (0-3) from the stack.
- S> - save image from current screen to alternate screen, takes 1 number (0-31) from the stack.
- S< - load image from alternate screen to current screen, as above.
- D> - save current screen to disk, asks for file name and adds the prefix "PAC." indicating a packed image.
- D< - load current screen from disk, asks for file name as above.
- @ - wait, takes 1 number from the stack which determines the delay until next word executed.

In addition to the routines listed above, words were developed to create and manipulate images. These additional words are used to pack images created using other graphic systems, to transform images, to collage images together, and to move groups of images between disk and the 32 alternate screens. None of these words are compiled with the system, rather they are stored on disk to be used as required. Briefly, they are-

PACK.KOALA - takes images created with MicroIllustrator, packs them and stores them on disk.

PACK.CEEMAC - takes images created with CEEMAC, packs them and stores them on disk.

INVERT% - flips the phase bits, changes green to orange and violet to blue.

INVERT - flips all bits, inverts the image.

- an integer random function, takes 1 number from the stack and places a random number back on the stack.

TP - "Torn Paper" creates a mask useful in collaging images together. There are 4 additional words TP0, TP1, TP2 and TP3 corresponding to the 4 logic modes.

VB - "Vertical Bars" creates a mask as above. Again 4 additional words VB0, VB1, VB2 and VB3 correspond to the 4 logic modes.

SQ - "Squares" creates a mask as above. Again 4 additional words SQ0, SQ1, SQ2 and SQ3 correspond to the 4 logic modes.

TSAVE - saves alternate screens to disk, takes 2 numbers from the stack indicating the first and last screens to be saved.

TLOAD - loads alternate screens from disk as above.

Computer generated images achieve a unique character through the use of masks. Complex images are built by collaging together several basic images in combination with masks using the 4 logic modes mentioned earlier. Photographers and filmmakers are more familiar with the term "matting" and videomakers with the term "keying". Essentially, we're talking about the same thing. Figures 18-22 show various masks produced by the Torn Paper, Vertical Bars, and Squares words.



Figure 18 Torn Paper mask.

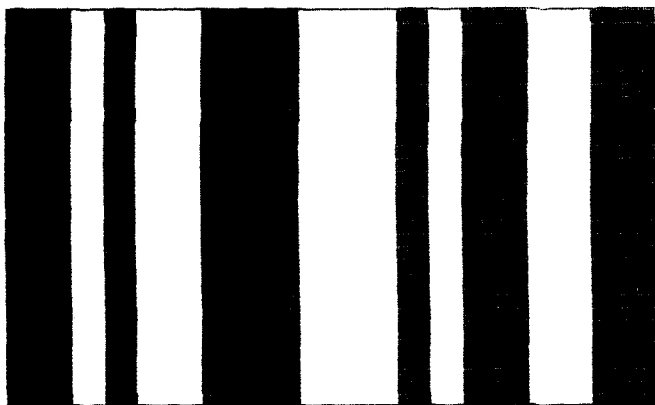


Figure 19 Vertical Bars mask.

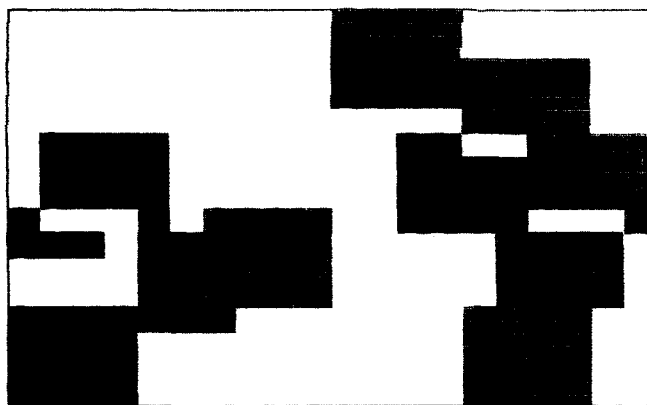


Figure 20 Squares mask.

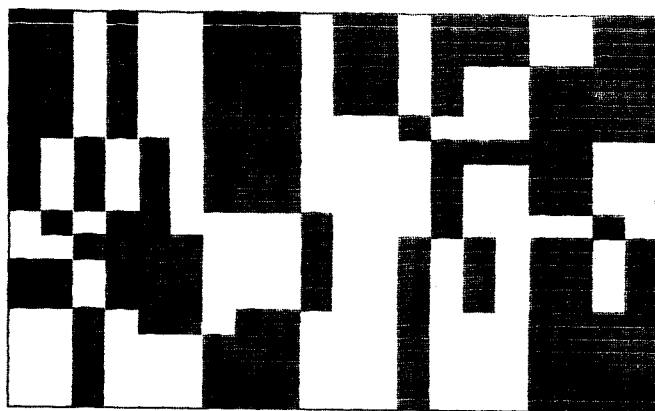


Figure 22 Combination of Vertical Bars and Squares.

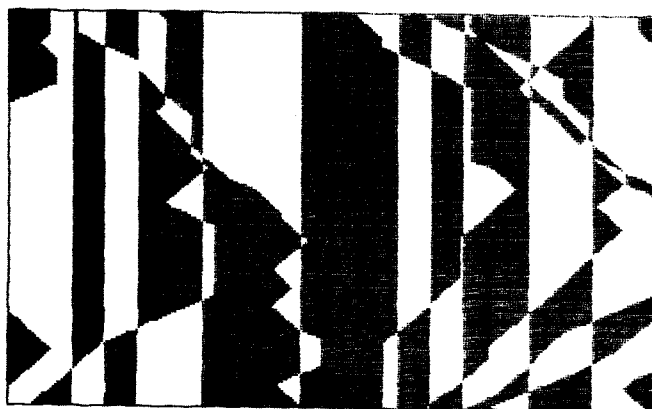


Figure 21 Combination of Torn Paper and Vertical Bars.

GRAPHIC SYSTEMS

Besides the VUtilities graphics system, both CEEMAC and MicroIllustrator were used to create images for Gretchen.

CEEMAC is a system for visual composition. The user writes a score which results in a dynamic visual display of dots, straight lines, curves, and predefined shapes. The user can interact with this display in "real time" and can direct the composition according to his/her whim. An animator friend of mine calls this "metaprogramming"; the user defines the rules which structure the image but doesn't define a particular image. Alternating between "execution" and "editing" the user refines

the rules until satisfying images result. These images can be "frozen" and saved on disk. CEEMAC was written by Brooke Boering and is available from Vagabondo Enterprises, 135 Stephen Rd, Los Aptos, CA 95003. Figures 23-24 are images from one such CEEMAC score.

MicroIllustrator is a "paint system" which comes with the Koala Pad. It allows the user to draw freehand or with simple shapes. There are 8 brushes available and the user can select from a palette of 18 colors. A fill routine and a magnify routine are included. Images can be saved and loaded from disk. I use this system to create original images and to 'clean up' digitized video images. Figures 25-26 were created with MicroIllustrator.

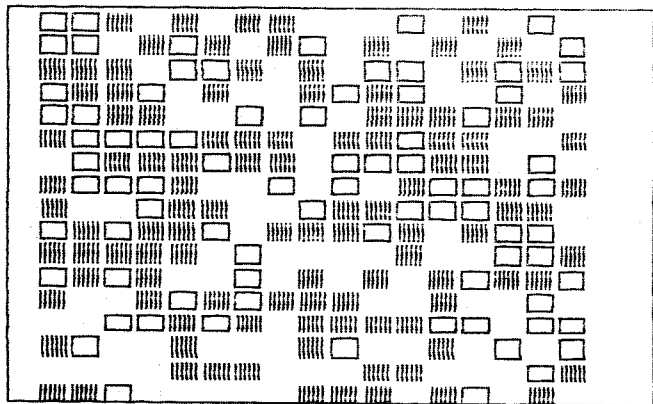


Figure 23 CEEMAC Image

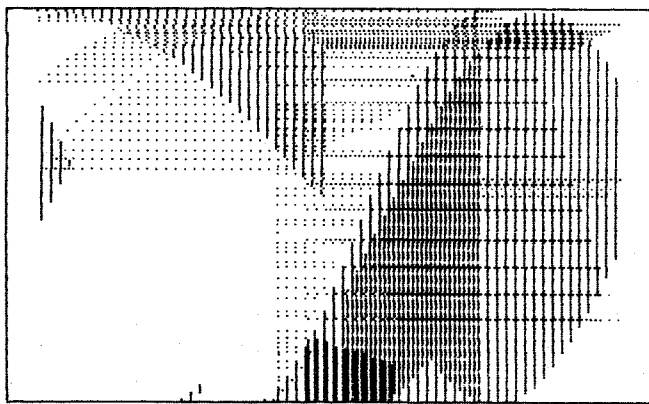


Figure 24 CEEMAC Image

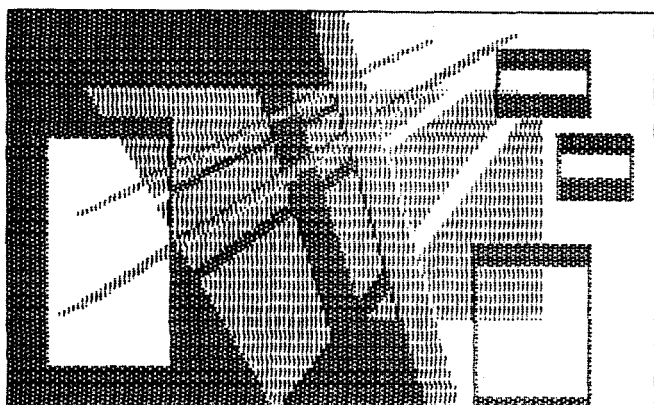


Figure 25 MicroIllustrator Image

Each of these graphics systems- my own TransFORTH system, CEEMAC, and MicroIllustrator- produce unique identifiable images. My own system can digitize a video image and produces geometric and mathematically derived images. CEEMAC images are also geometric but are more fluid and more random than my own. MicroIllustrator produces those cartoon-like graphics tablet images that easily identified as "computer graphics". Many of the images in the presentation collage or combine

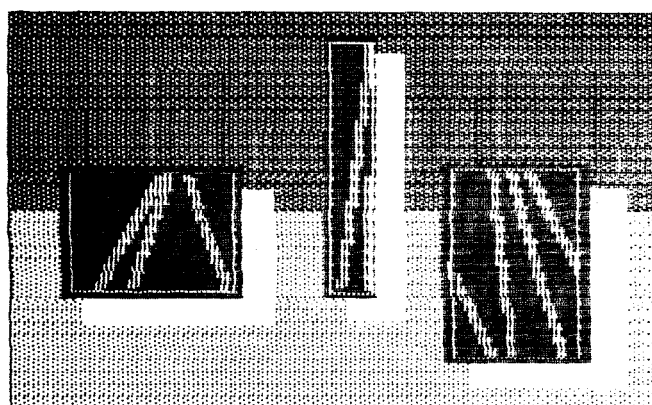


Figure 26 MicroIllustrator Image

images from the same system or images from different systems, Figures 27-30.

These images are the result of overlaying simple images; sometimes of similar types (complementary) and sometimes different (contrasting). The juxtapositions which result create visual tensions which are resolved structurally using balance and proportion, design techniques familiar to all graphic artists.

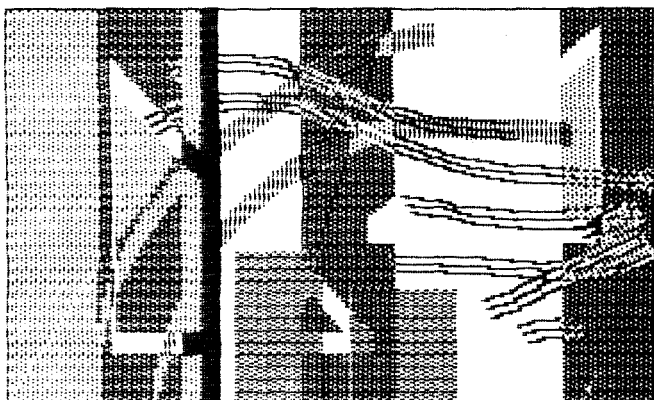


Figure 27 Shapes So Strange combines two Koala Pad images.

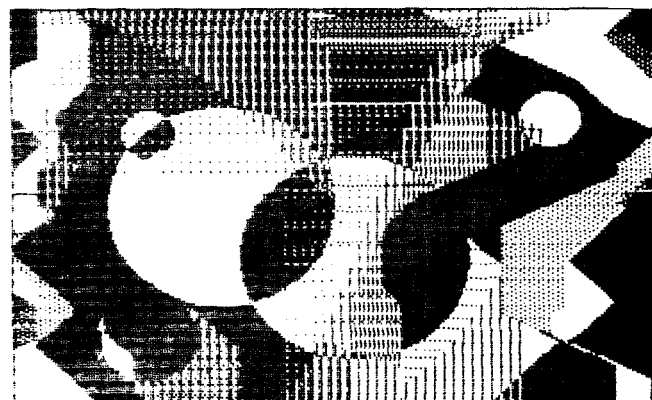


Figure 28 Shapes and Patterns collages images from CEEMAC and my own system.

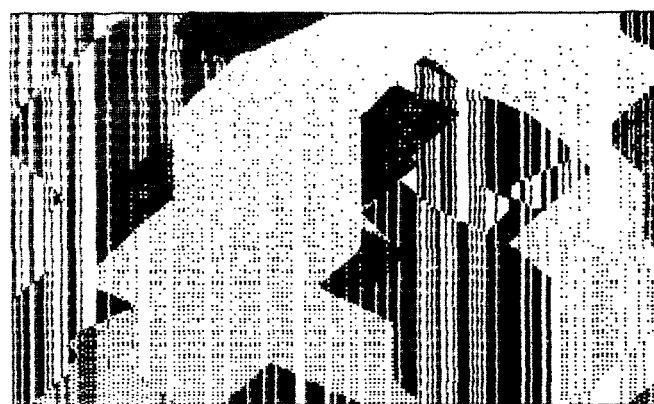


Figure 29 Computer Augmented Creativity uses images from CEEMAC and my own system.

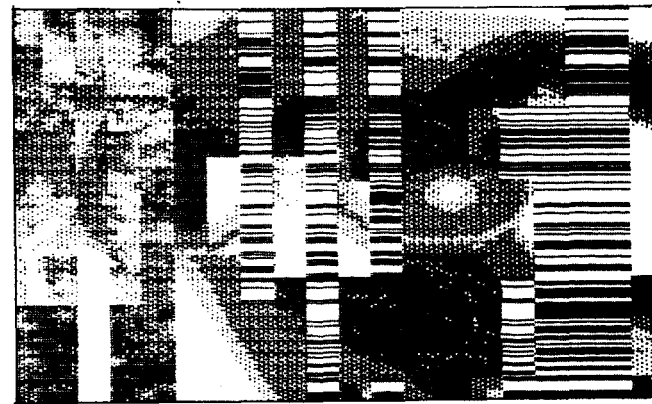


Figure 30 Possessed by Passion combines digitized and geometric images.

COMPLEXITY AND ART

In art then, as in life, complexity is produced by overlaying simple images or patterns, resulting in tensions and, therefore, changing the way in which the elements which make up the images or patterns are perceived. And, we hope, changes the viewer in some small way as well. We have all experienced disruptions to our daily routines which produce tensions and demand change, either in our routines or in ourselves. When things become too much, we seek to relieve the tensions, to restore harmony, to return to a simpler way life. The same can be said of art. The complex rhythms and conflicting melodies introduced into a piece of music must be resolved by the composer. The tensions produced by contrasting color and form must be resolved by the painter. No, I don't mean resolution as in conflict/ crisis/ resolution, the hollywood plot; I mean resolution as in dynamic balance. Nora does not provide a solution to Gretchen's problem; this problem may be insoluble; this is real life. The painter and the composer do not resolve conflicts between elements by keeping the good ones and

eliminating the bad; but by creating dynamic balance, a structure in which conflicting elements not only co-exist but co-exist to each other's benefit. Complexity, therefore, is an important attribute of the structure which organizes a work of art.

CONCLUSION

This past summer, We installed Gretchen Prisoner of Love in a group show at Virginia Commonwealth University's Anderson Gallery in Richmond. The presentation was programmed in TransFORTH, using the routines outlined above, and ran on an Apple IIe. The images were stored on my Synetix RAM card, for fast recall, and appeared simultaneously on three monitors. The program cycled continuously, during gallery hours, for two weeks and drew a small but appreciative audience.

In an effort to achieve broader distribution, I have reprogrammed the presentation in assembler to run on any 48K Apple II. Gretchen Prisoner of Love is available at minimal cost; it can be copied free of charge.

DIGITAL SAMPLING AND FFT ANALYSIS OF ACOUSTIC SOURCES: A MICRO-COMPUTER IMPLEMENTATION

Jesse Klapholz

Klapholz Technologies
3730 Lankenau Road
Philadelphia, Pa. 19131

ABSTRACT:

An FFT spectrum analyzer, implemented as a hardware plug-in card for the AppleII computer, with its associated software, is described. The FFT process is explained, and a brief background of acoustical analysis is given. Several applications of the hardware/software in musical acoustics are presented.

0 INTRODUCTION:

Audio and acoustical analyzers are no strangers to the musical instrument world, nor is digital synthesis. However, what is new is a highly sophisticated hardware/software digitally-based analyzer system for the AppleII computer. Until a few years ago, computerized FFT analyzer systems were available only to large research organizations. This paper will describe how the IQS 401-1 FFT Spectrum Analyzer system can now be affordably used by acousticians, sound engineers, musicians, musical instrument designers, and/or builders. The system will be shown to be useful in both analytical and digital sampling/manipulation applications of any analog signal(s).

1 BACKGROUND:

The famous German physicist Hermann von Helmholtz (1821-1894) invented perhaps the most classic analyzer; this was used in experiments forming many theories on which modern acoustics and psychoacoustics are based. Even before the era of electronic Fourier spectrum analyzers, Helmholtz used a large set of his acoustic resonators to verify the existence of harmonics in complex tones. By holding successively smaller (i.e. higher frequency) resonators to his ear with a musical note playing into the resonators large opening, Helmholtz heard an increase in the amplitude of

any frequency that was present in the harmonic structure of the instrument. Thus, he could roughly determine the Fourier spectrum of the note.

The invention of the vacuum tube in 1915 marked the beginning of modern analysis advancing the field from mechanical analyzers to electronic machines. The solid-state and computer era provided the means to advance the implementation of Helmholtz's and Fourier's theories, and brought them into our laboratories and studios.

The FFT technique is based upon Fourier's theorem which states that any periodic function (of frequency f) can be expanded as a linear combination of the sine and cosine functions of frequencies $f, 2f, 3f, \dots$. Therefore, once one set of values is known, the other may be computed, i.e. transform time domain values to frequency domain values and visa versa. The Fourier transform technique became popular in the 60's when researchers developed a mathematical shortcut, the FFT (Fast Fourier Transform), minimizing the computation time.

The technique of the FFT starts with a digitized signal waveform, which we can consider to be a list of numbers. We multiply the numbers on this list by those on other lists representing digitized sine and cosine waves. In doing so, we use multiplication to transform the time-oriented list made from the original signal into a second list of numbers, one for each of the frequencies we have analyzed. All of this 'number crunching' is carried out by a micro-computer, and the resultant list of numbers is then displayed on a CRT in the form of a graph.

2 THE FFT ANALYSIS SYSTEM:

The IQS 401-1 FFT Spectrum Analyzer is an extremely powerful AppleII based

system that allows complete analysis using the Fast Fourier method. This laboratory grade instrumentation system affordably provides all the tools necessary, both hardware and software, to accomplish FFT analysis of physical systems or analyze arbitrary signals for power spectrum, phase, or group delay characteristics. System facilities provide for test signal generation and the acquisition, analysis, storage, and plotting of real-time wave forms and spectra in either the time or frequency domain or three-dimensionally in both domains simultaneously. System software features extreme versatility and ease-of-use while clear presentation of data is provided by its high-resolution graphics.

2.1 THE HARDWARE: The basic system architecture is outlined in figure 1. The preamp (figure 1b) is gain controlled by a DAC used as a digitally programmed attenuator, via software operation. In the feed-forward compensation mode, the preamp has an extended gain bandwidth-factor of ± 0.2 Db 5Hz to 40KHz.

The next stage consists of the anti-alias low pass filter, which is programmable in order to accommodate multiple sample rates, and has a 100 Db/octave rolloff. The output of the filter routes to the sample and hold amplifier, where digital data is strobed out onto the data bus via an ADC (audio to digital converter).

The test signal generator circuit consists of a DAC (digital to audio converter) that can receive instructions directly from software. This allows simple software driver routines to create a wide variety of waveforms for many applications. For example, it can be used to provide impulses that are software controllable in amplitude and width for impulse testing. Another application is to 'play back' waveforms at any speed from memory which may have been previously sampled, brought off disc, or synthesized by inverse FFT. Such waveforms could consist of gated sine tone bursts, pre-recorded sounds, sections of speech, etc.

Referring to the system block diagram in figure 1, the digital portion is represented by the lower 2/3 of the diagram. It provides interface, memory, timing and logic control functions. The SYSTEM CONTROL LOGIC block determines

how each module interacts with one

another and links the overall system to the Apple II bus structure.

In order to maximize efficiency, many of the most often used machine code subroutines (including the FFT program) are placed in read only memory (ROM). Three ROM sockets (ROM BANK 1-3) are incorporated in the hardware. A single 4 kilobyte ROM is installed in the present version, thus allowing future expansion for additional signal processing functions.

2.2 THE SOFTWARE: Written in BASIC, the control program can be easily modified (or a new version written from scratch) to accomplish specialized tasks. For instance, establishing whether or not an electric piano conforms to an accepted standard curve might require a sequence of steps from the original control program. Maximizing the use of software plays a major role in the control of system costs; this is one feature that makes the IQS an economical package.

There are two basic modes of the control program, the ACQUISITION MODE and the ANALYSIS MODE. All of the commands are invoked by a single key-stroke, such as A = AMPLIFICATION, V = VOLUME, P = PLOT etc. Pressing a single key such as 'S' would bring up the 'sampling rate menu' onto the screen. There are eight different choices of sampling rates available; these are shown in figure 2. When the sampling rate is selected, the ANTI-ALIAS filter and test pulse signal width are set. All of the major commands are shown in figure 3.

3 DATA ACQUISITION:

Upon 'booting up' (loading the control program into the computer) the IQS software, some of the variables are in a 'default' setting; i.e., they will be set to predetermined values unless you elect to change them. These are the (A)mplification of input, (C)hange number of averages, (S)ample rate, and (V)olume of test signal commands.

Any analog signal may be analyzed regardless of its waveshape (sine, cosine, square, etc.). It may be voice, low frequency vibration, the brain's alpha waves, musical instruments, or any analog signal.

Pressing the space bar will initiate a test pulse and then acquire one sampling of 4,096 points. While data is being sampled, the levels and time posi-

tioning of the analog input are displayed as a waveform on the monitor, similar to an oscilloscope display. The 'return key' can be pressed at this time to display the first 256 points of time data on the monitor. At this point you may save the waveform to disc, or proceed with data processing.

4 DATA ANALYSIS & MANIPULATION:

If one desires to use some signal conditioning, rather than only using the 'raw' input data, DC offset can be removed from the signal. It can also be digitally filtered, resampled to improve the low frequency resolution, the spectrum may be smoothed out, the signal may be windowed (weighted), etc.

At any point during a measurement session, any waveform or graphics display may be stored to disc for future use. Similarly, any waveform or display may be sent to a graphics printer or plotter. With a waveform in memory or one retrieved from disc files, FFT's can now be computed. There are four different lengths of FFTs: 128, 256, 512 or 1,024 points. The time length will be dependent upon the selected sampling rate and how many points in the FFT. Once an FFT has been computed and the power spectrum displayed, the phase response and group delay may also be computed and displayed.

All FFT analysis, since it is a mathematical process, is performed in linear frequency scales. The addition of a movable cursor allows for an exact readout of both frequency and level. The advantage of using linear frequency scales is that one can easily determine the bandwidths of notches or peaks in the spectrum. Since we all respond logarithmically to linear physical changes of external stimuli, viewing spectral information in the frequency domain in log scales is essential. Therefore, displaying spectral information in log frequency scales is accomplished through a built-in graphics subroutine which can be executed in a matter of seconds with the (L)og command.

Music and natural sounds have more than just steady state conditions. Primarily, they are attack, steady state, and decay, and they are characterized by pitch, duration, articulation, loudness, timbre, etc. A very useful feature of microprocessed FFT spectrum analysis is that any part of the captured waveform can be analyzed,

i.e. the initial attack/transient, the steady state, and the decay tail or any combination of these.

Data may be presented as a fundamental frequency and its harmonics, and may also show how all of these components take place in time. This frequency domain information is a direct reciprocal of a time domain waveform. Through FFT analysis we can view sampled sounds 'jumping' from one domain to the other, gathering information that can be used for investigative purposes, fine tuning of instrument construction, or building up 'wave shape tables' for digital synthesis of natural sounds. All of these features become invaluable tools for 'before' and 'after' pictures, various comparisons between stored-on-disc information and device(s) under test, etc.

5 APPLICATIONS:

Before commencement of any analysis, the device(s) being tested should be clearly understood in terms of how they operate and interact with their environment. Now that the basics of the FFT analyzer have been presented, some practical applications shall be considered.

The FFT and its companion, the inverse FFT, find extensive use in the production of electronic music. Using the IQS IQSynth Wavepak I software, one is able to perform waveform sampling, storage, synthesis, and comprehensive editing of sampled and created sounds.

With a reference microphone connected to the analyzer, a sample of a musical instrument waveform may be digitally recorded. The wave sample can then be played back through the 401-1 or polyphonically through any synthesizer using a 256 by 256 waveform table. If it is desirable to alter the harmonic structure, one can view the waveform in the frequency domain, and edit the overtone structure (up to the 128th harmonic, both odd and even). Performing the inverse FFT of the spectrum will yield a new time domain waveform, which may be played back or archivally stored for use in future musical performances, etc.

As with the above process, through the editing functions, one may start with a blank frequency screen and plot his/her own fundamental/harmonic spectrum structure, then perform the inverse FFT 'creating' a new waveform.

Conversely, the amplitude and time values of any waveform may be modified.

To display energy information as a function of frequency and time, the traditional 'waterfall' display is used. The 401-1 does this with a **Spectral Decay** contour program. Figures 4 and 5 are examples of spectral decay contours taken from the waveform data acquired from a loudspeaker's response to the pulse 'played back' through the 401-1 analyzer. The major differences between the two displays are a result of the 'windowing' (weighting) used in figure 5. Figure 4 used a "rectangular" window (flat or unity weighting), and the FFT's in figure 5 were computed with a cosine function window. As can be seen, figure 5 is much easier to analyze, i.e. pick out resonances and so on.

When an FFT is computed, it is looking at a 'window' of the time domain data, that is, a 'slice' of the continuous 'periodic' waveform. The FFT joins the beginning and end of this slice of time, forming an assumed periodic function. Tapering the ends of this slice of time by various 'windowing' functions can thus reduce errors that would normally be introduced by joining ends that are continuous (since, after windowing, both the beginning and end would now have zero energy). Further, selecting various windowing functions allows control over the trade off between frequency and time resolution.

Once the time domain data is gathered, these or other programs may be run to display the information in many different views, giving us much more insight than was previously possible.

A new program, TimeSpectrum v. allows us to view frequency/time changes of physical systems in a unique and unprecedented way. The TimeSpectrum display eliminates the distortions normally encountered when isometrically displaying a three dimensional plot in a two dimensional graphics medium. TimeSpectrum does this by showing the energy amplitude with a greyscale intensity; the more energy, the darker the display point becomes. The frequency and time scales are now shown in the x and y axis without distortion, and 'clumps' of frequency defined 'energy packets', displaced in time now form a crystal clear picture.

Figure 6 is the loudspeaker analyzed in figures 4 and 5, the darker

areas of the display correspond to the

peaks in amplitude of the spectral decay plot. Comparing the identical data in these two plots, one can easily see the build-up, steady-state and decay characteristics of the d.u.t. (device under test) all in one plot. This observation can be made without having to resort to the 'front' and 'rear' view of the more conventional 'waterfall' plot.

Figure 7 is a 'voiceprint' plot; three discrete 'energy-packets' can easily be seen displaced in time as a function of frequency. It is also interesting to observe in figure 7 the 'repetitiveness' of these 'energy-packets'.

Figure 8 and 9 are plots of the same note played identically on two different pianos. The strong 'ridge' in figure 9 is the actual note being played, showing a piano with a much better 'voicing'. This application of Time Spectrum demonstrates the enormous amount of information attainable for analysis of musical instruments.

6 SUMMARY:

The IQS 401-1 FFT Spectrum Analyzer hardware plug-in card for the AppleII computer, with its associated software was described. The FFT process was explained, as well as a brief background of acoustical analysis. Several examples of analytical and digital sampling/manipulation applications of analog signal(s) were presented. The system was shown to be useful to acousticians, sound engineers, musicians, musical instrument designers, and/or builders.

7 CONCLUSION:

Once the basic techniques of FFT analysis are mastered, the applications are endless. In this paper, we have touched upon only basic techniques. Based on early practice, the application of computer technology has led us to a whole new view of electro-acoustic analysis. The current state of the art has given us 'cameras' to take 'pictures' of electro-acoustical phenomena more efficiently, accurately, and affordably than older methods.

8 ACKNOWLEDGMENT:

The author wishes to thank Walter Holland and William Elder of IQS Inc. for their input and constant support.

IQS 401 QUICK REFERENCE GUIDE

ACQUISITION MODE

CONTROL OPERATION

SPACE = SAMPLE data (4096 points)
 E = ERASE screen
 B = BLANKING of waveform (toggles)
 A = AMPLIFICATION
 C = CHANGE # of averages - in powers of 2
 D = DELAY before sampling (1 = 0.6 msec)
 P = PLOT screen to printer w/Grappier
 S = SAMPLE RATE
 V = VOLUME of test signal
 I = INITIATE -collect & average data (2048 points)

ANALYSIS MODE

CONTROL OPERATION

RETURN = READY/DISPLAY FIRST 256 points of time data
 1 = 128 point FFT
 2 = 256 point FFT
 3 = 512 point FFT
 4 = 1024 point FFT
 SPACE = RETURN to ACQUISITION MODE
 D = DIFFERENCE - current less stored spectrum
 B = PHASE and then GROUP DELAY
 ! = CONVOLVE - current with stored spectrum
 Q = SPECTRUM DECAY PLOT from current waveform
 OPTIONS:
 <freq. scale> 1 = LOG scale
 2 = LINEAR scale
 <size of FFT> 1 = 256 point
 2 = 512 point
 <# of spectra> 32
 50
 <delta points> 1 - 80
 <vert. reduct> 3 - 4 for 32 spectra
 5 - 7 for 50 spectra
 <start point> 0 - 2048

IQS 401 QUICK REFERENCE GUIDE

ANALYSIS MODE, Cont'd

DATA MANIPULATION

H = SHIFT data left
 O = OUT DC - removes DC from time data
 R = RESAMPLE
 S = SKIP 256 points in time data
 T = TRUNCATE time data
 U = NEWSTART - select new start in data
 W = WINDOW - cos wgt to first 256 points
 Y = DIGITAL FILTER - moving avg. smooth
 X = EXCHANGE spectrum in temp. storage
 Y = DIGITAL FILTER - moving avg. smooth
 % = SMOOTH spectrum

DISPLAY MANIPULATION

L = LOG display
 P = PLOT screen to printer w/Grappier
 RIGHT => = VIEW right half of 1024 point FFT
 LEFT <= = VIEW left half of 1024 point FFT

DISC STORAGE AND RETRIEVAL

ACQUISITION MODE:

G = GRAPHICS SAVE
 N = FILE RETRIEVE & RETURN TO ANALYSIS

ANALYSIS MODE:

G = GRAPHICS SAVE
 K = WAVEFORM SAVE

TO EXIT PROGRAM - you must be in Analysis Mode, then
 press Control Key and E simultaneously.

Rev. 9/82

FIGURE 3.

SPECTRAL DECAY OF FULL RANGE LOUSPEAKER USING
RECTANGULAR WINDOW

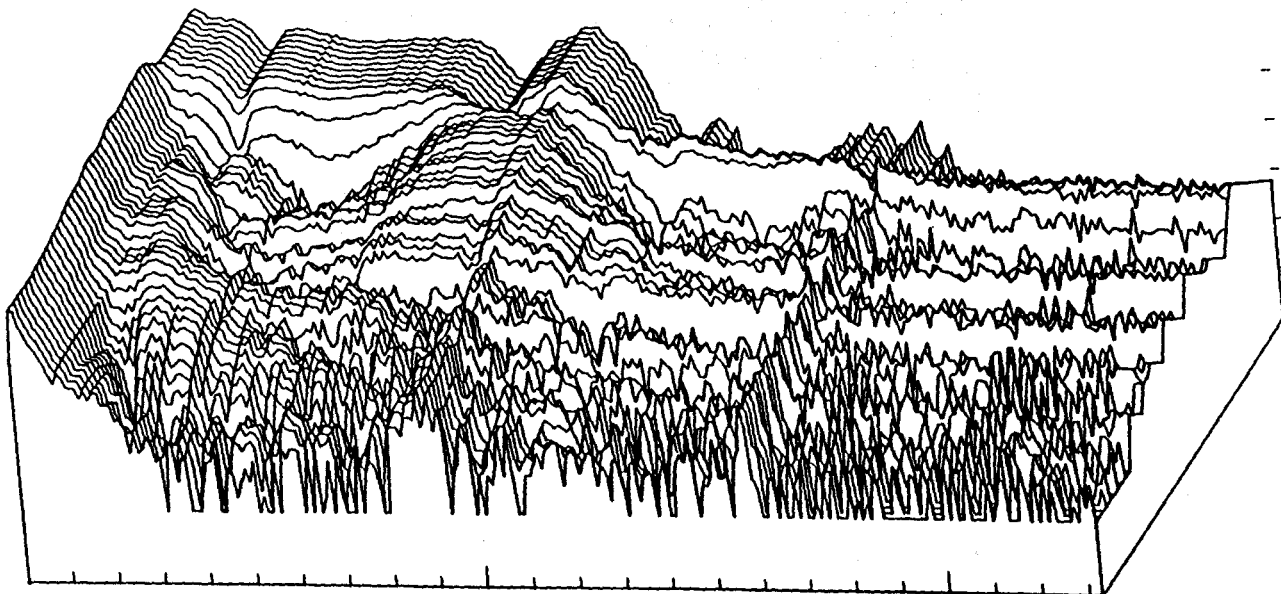


FIGURE 4.

SPECTRAL DECAY OF FULL RANGE LOUSPEAKER USING
COSINE WINDOW

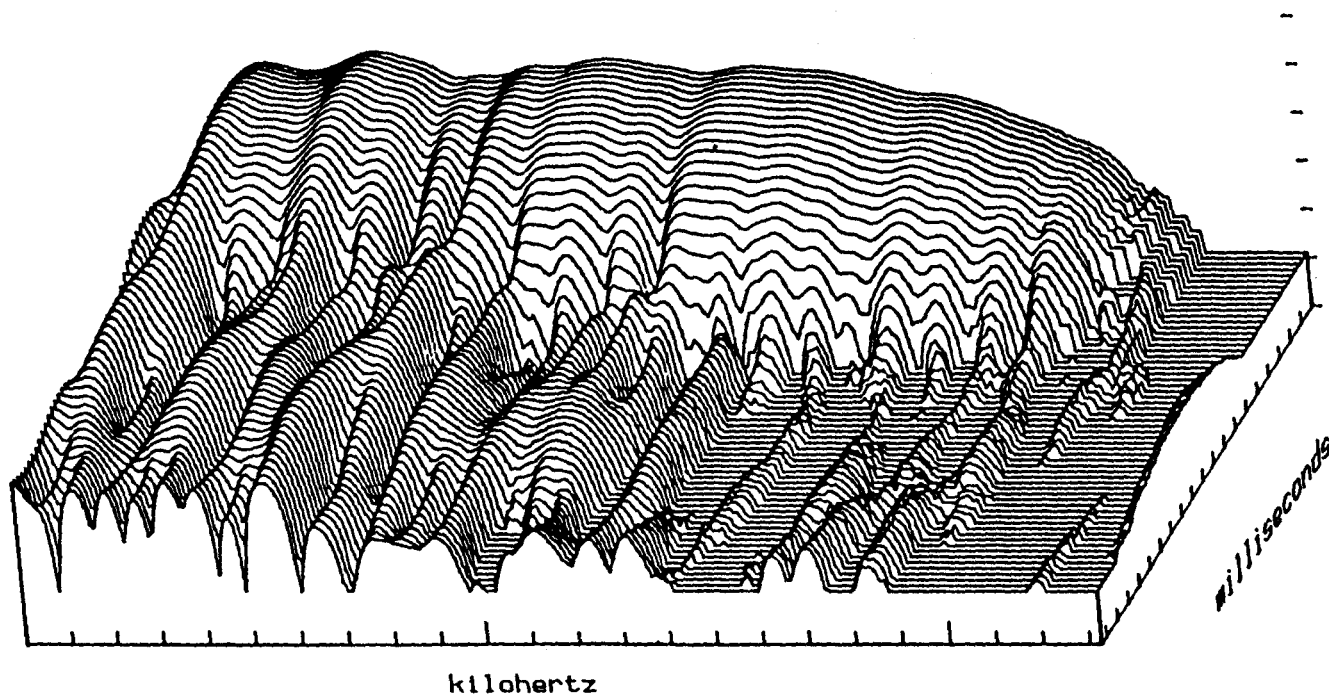
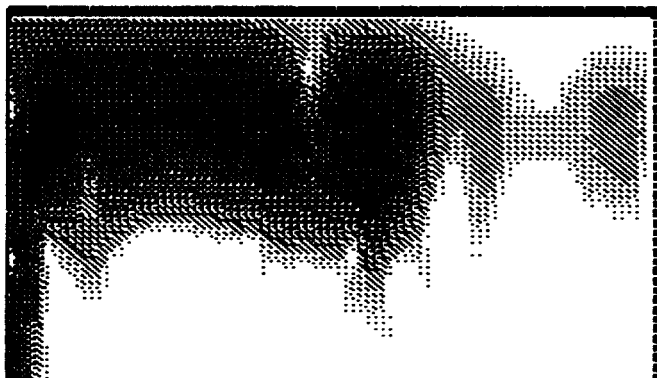


FIGURE 5.

TH67 335 WI 128 OR24793DL3 15434



IQS inc

FIGURE 6.

TH175 425 WI 256 OR128 DL16 1543

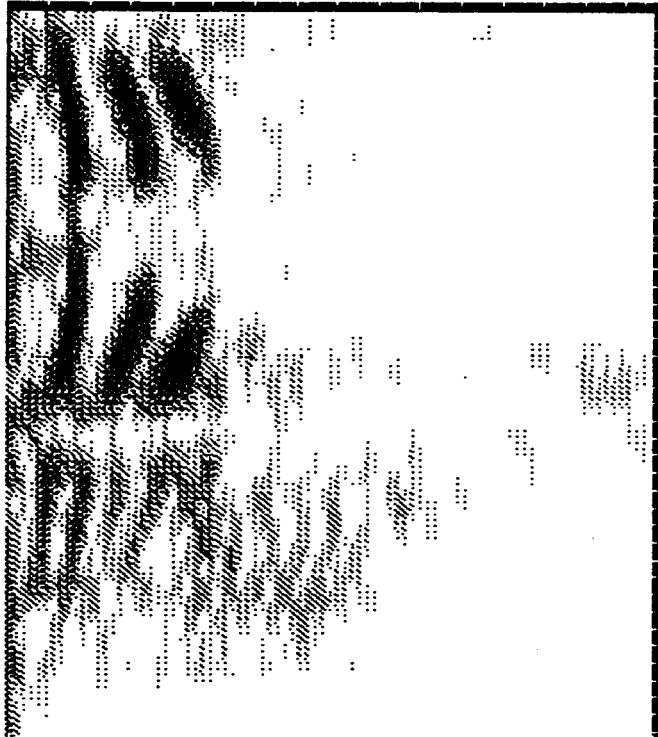
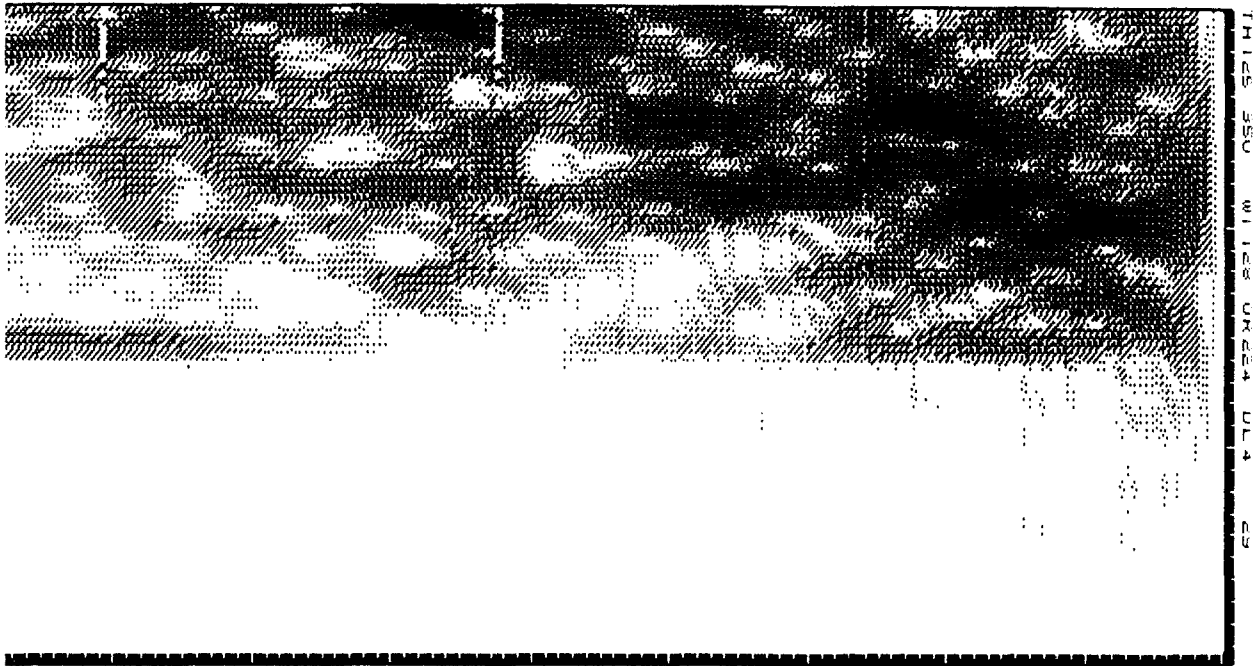
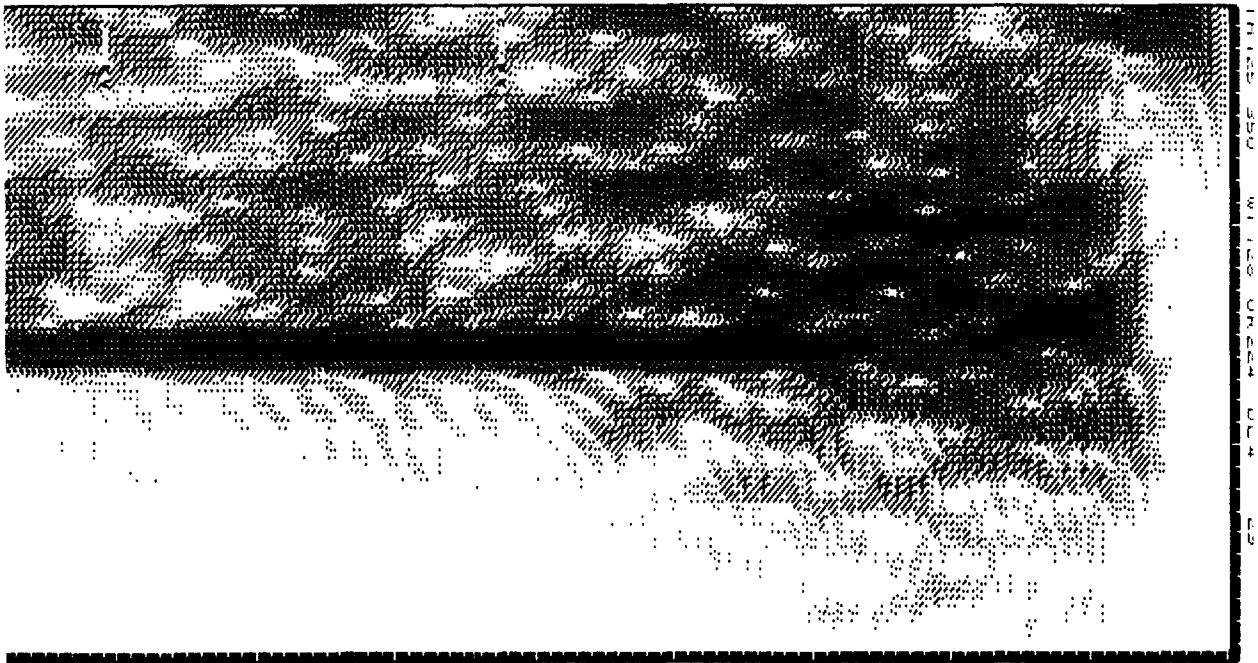


FIGURE 7.



IQS inc c

FIGURE 8.



IQS inc c

FIGURE 9.

Data Compression of Computer Music

Michael Keith
D46 Abbington Drive
Hightstown, NJ 08520

Introduction

This paper will discuss the application of data compression techniques to the storage and manipulation of digital music data in a computer. The goal of this endeavor is to take music files from an existing microcomputer music system (e.g., ALF, MMI, Mountain Computer) and compress them into smaller files with the following properties:

- (1) Compressed files are significantly smaller than the original (on the order of three times smaller is desirable), and
- (2) The compressed files can be decompressed very quickly (a few tenths of a second is desirable). This allows programs to play compressed music files with essentially no apparent delay to the user.

Unfortunately, these are conflicting goals; in general, to achieve greater compression requires a more complicated algorithm which yields a slower decompression program. We will show, however, that it is possible to achieve significant compression, by presenting an algorithm with a compression factor of 3.3 (for ALF-format music) that also satisfies property (2).

First Steps

The initial motivation for this research was the monumental work by W. F. McGee of Ontario, who entered all 371 four-part chorales of J. S. Bach into the ALF music system on the Apple II (using the ALF music editor program). These pieces were entered in an efficient way (using subroutines for repeats, etc.), but still require 5 Apple diskettes to store all the music.

The immediate challenge which presented itself was to try and fit all 371 chorales on a single Apple diskette. Although this might seem to require a compression factor of about 5, in fact less is required, because there is some overhead storage space used up by each separate file on an Apple diskette. After compression, we can store, say, 10 chorales per file (rather than one per file) and save a lot of this overhead. We will see shortly an estimate of the compression needed to accomplish this task.

The basis of all data compression algorithms is to *exploit the statistics of the data*. For example, in compression of digitally-encoded pictures (bit-maps), adjacent pixels are often identical in value. Therefore, run-length encoding, which encodes runs of identical pixels by storing the value and number of identical pixels, can be used to compress data possessing this type of statistics.

In order to exploit the statistics, we first have to *obtain* the statistics, by analysis of the data. In Figure 1 we give a partial listing of one of the Bach chorales (number 14), printed by a program that "disassembles" ALF songs. Hex values are shown in the left column, and the meaning of each code is shown in the right column. The first byte of a song denotes the number of voices. The next 2*(number of voices) bytes are pointers to the start of the data for each voice. Following this are 3-byte groups encoding the actual music data. At the end of the file is one byte giving the suggested playing speed, and finally 160 bytes (four 40-character lines) encoding the title of the song.

As a basis for computing compression factors, we need to know the total number of bytes in all 371 chorales. Analysis of the chorales produced the following information:

Lengths of the Chorales	
Largest	3794 bytes (Number 205)
Smallest	683 bytes (Number 130)
Average	1146 bytes
Total	425K

Note that the capacity of an Apple II diskette is 560 sectors = 140K bytes. Thus we need a compression factor of about 3 to fit all the chorales on one diskette. Actually, since some file-storage overhead is unavoidable, a compression of about 3.3 is required. This will be our design goal.

Now let us begin compressing. Referring to Figure 1, the first thing we realize is that the last 161 bytes (suggested speed and title) are not used when playing a song (only when editing). Since we are only concerned with preserving the information necessary to play the music, we can throw these 161 bytes away.

We also observe that there are two fundamentally different types of data triples: those encoding notes and rests, and those encoding commands (such as TEMPO, TIME, CALL, etc.) The commands can always be distinguished from notes because their first byte is always C1 through FF hex. In addition, we see that the statistics of the commands are significantly different from the statistics of the notes. So we decide that our encoding will have two "modes" - one for encoding commands and one for encoding notes, with some as yet unspecified means of switching modes. We will call these two modes "byte mode" and "bit mode", since it turns out that commands are most conveniently encoded in bytes and notes are best encoded as a bit stream.


```

FILE CHORALE 14

04
35 01 E0 01 94 02 51 03

FE FE FE RESERVED
FE FE FE RESERVED
FE FE FE RESERVED
FC 81 00 KEY
5C F0 00 NOTE: G -3 TIME: 4
5C F0 00 NOTE: G -3 TIME: 4
5A F0 00 NOTE: F#-3 TIME: 4
56 F0 00 NOTE: E -3 TIME: 4
52 F0 00 NOTE: D -3 TIME: 4
5C F0 00 NOTE: G -3 TIME: 4
60 F0 00 NOTE: A -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
64 7B 00 NOTE: B -4 TIME: 2
66 3C 00 NOTE: C -4 TIME: 1
6A 3C 00 NOTE: D -4 TIME: 1
66 F0 00 NOTE: C -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
60 7B 00 NOTE: A -4 TIME: 2
64 7B 00 NOTE: B -4 TIME: 2
66 F0 00 NOTE: C -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
60 E0 01 NOTE: A -4 TIME: 8
5C D0 02 NOTE: G -3 TIME: 12
CA 0A 00 RETURN

---- (3 more similar subroutines) ----

C8 22 04 CHANNEL
FC 81 00 KEY
FD 04 04 TIME
FB F0 00 QUARTER
C1 14 00 GAP
C2 00 FE TRANSPOSE
C3 00 20 ATTACK
C4 19 00 DECAY
C5 D8 D6 VOLUME
C6 00 00 SUSTAIN
C7 DC 05 RELEASE
C0 F0 00 NOTE: <REST> TIME: 4
C0 F0 00 NOTE: <REST> TIME: 4
C0 F0 00 NOTE: <REST> TIME: 4

07 0A 00 CALL
C9 0A 00 CALL
5C F0 00 NOTE: G -3 TIME: 4
60 F0 00 NOTE: A -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
5C F0 00 NOTE: B -3 TIME: 4
5C F0 00 NOTE: G -3 TIME: 4
60 F0 00 NOTE: A -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
5C F0 00 NOTE: G -3 TIME: 4
56 F0 00 NOTE: E -3 TIME: 4
60 68 01 NOTE: A -4 TIME: 6
5C 78 00 NOTE: G -3 TIME: 2
5A F0 00 NOTE: F#-3 TIME: 4
5C 78 00 NOTE: G -3 TIME: 2
5A 78 00 NOTE: F#-3 TIME: 2
56 E0 01 NOTE: E -3 TIME: 8
52 F0 00 NOTE: D -3 TIME: 4
52 F0 00 NOTE: D -3 TIME: 4
5C F0 00 NOTE: B -3 TIME: 4
5A F0 00 NOTE: F#-3 TIME: 4
56 F0 00 NOTE: E -3 TIME: 4
52 F0 00 NOTE: D -3 TIME: 4
5C F0 00 NOTE: G -3 TIME: 4
60 F0 00 NOTE: A -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
60 F0 00 NOTE: A -4 TIME: 4
6A 68 01 NOTE: D -4 TIME: 6
66 7B 00 NOTE: C -4 TIME: 2
64 F0 00 NOTE: B -4 TIME: 4
60 7B 00 NOTE: A -4 TIME: 2
64 7B 00 NOTE: B -4 TIME: 2
66 F0 00 NOTE: C -4 TIME: 4
64 F0 00 NOTE: B -4 TIME: 4
60 E0 01 NOTE: A -4 TIME: 8
5C D0 02 NOTE: G -3 TIME: 12
C0 F0 00 NOTE: <REST> TIME: 4
C0 F0 00 NOTE: <REST> TIME: 4
C0 F0 00 NOTE: <REST> TIME: 4
C0 F0 00 NOTE: <REST> TIME: 4
CB 00 00 STOP

----- (3 more similar parts) -----

FF 00 00 END

FF SUGGESTED SPEED

CHORALE 14
O HERRE GOTT, DEIN GOETTLICH WORT
J S BACH
31 AUG 1983

```

Figure 1: A "disassembled" listing of the ALF music data for one of the Bach chorales (number 14).

For reference, the average size of 1146 bytes per chorale can be analyzed further as follows:

Category	Average No. Bytes
Notes	748
Commands	237
Speed + Title	161
Total	1146

Having disposed of the 161 title bytes, we will now discuss compression of each of the other two categories.

Compression of Commands (Byte Mode)

In byte mode, we notice the following three things that yield compression:

(1) Triples containing three FE's (of which there are many) can be encoded by a single byte. These FE's are reserved bytes which I believe are used by the ALF real-time software as variable holders.

(2) Quarter, Time and Key commands (FB, FC, and FD hex) are not needed when playing a song (only when editing). So we can encode each of these by one byte which will be decompressed into a 3-byte NOP instruction. The NOP's are necessary to keep the length of the decompressed song the same as the original, which is essential since CALLs use relative addresses to point to subroutines.

(3) The 21-byte sequences beginning with a C1 are always the same. We can encode these 21 bytes by a single byte which will be decompressed into these fixed 21 bytes. This sequence appears once for each part, and so yields a significant savings.

All other bytes in byte mode will be transferred directly (with no compression).

To be more specific, we use the following byte values to encode these compression sequences:

Byte Value	Interpretation
0	21-byte C1 sequence
1	FE FE FE
2	Switch to bit mode
3	End of data marker
4	FC FC FC (NOP, replaces FB, FC, FD commands)

On the average, the combination of deleting the last 161 bytes plus the byte mode compression saves about 300 bytes per song. In other words, the approximately 400 bytes in the original are reduced to about 100.

Compression of Notes (Bit Mode)

Since the amount of command and title information is essentially a constant (and significantly smaller than the amount of note information) the compression of the actual notes and rests of the music is clearly the most crucial in obtaining good overall size reduction. In ALF music, any triplet with a first byte of C0 or less represents a note. The first byte of each triplet represents the pitch (actually, only the high-order 7 bits are significant) and the second two bytes represent duration (from 1 to 64K units of time).

This can be improved immediately by realizing that there are never more than 16 different durations of notes in a single chorale. We could thus encode note durations by storing a table of the 16 (or less) durations used in the piece, and then encoding the duration of each note by storing a pointer into the duration table. This would require only 4 bits per note. In combination with the 7 pitch bits, this gives a total of 11 bits, as compared to 24 bits (3 bytes) in the original ALF data. This of course means that the compressed data in bit mode will be a bit stream which will not be aligned with byte boundaries. Note also that even though byte mode compressed data is always composed of bytes, it may also not be aligned on byte boundaries, due to the fact that bit and byte mode are freely mixed. In general, only at the beginning of the compressed file are things aligned on byte boundaries.

The combination of all the compressions so far yields an average compressed length of 442 bytes, for a compression factor of 2.6. Although this compression is substantial, it is still not enough to fit all the chorales on one diskette. We are forced to look for an even better scheme.

Looking at the data of Figure 1 again, we notice the following fact: most of the notes in the chorale are close to the preceding note (usually differing by only a few half steps). If we arrange our duration table properly, it also seems likely that the duration pointer for each note will also be close to the one for the preceding note. For example, after a quarter note, an eighth note, half note, or another quarter note is more likely than a thirty-second note. This leads us to examine the statistics of the *first-differences* of the notes; that is, the difference between each note and its predecessor.

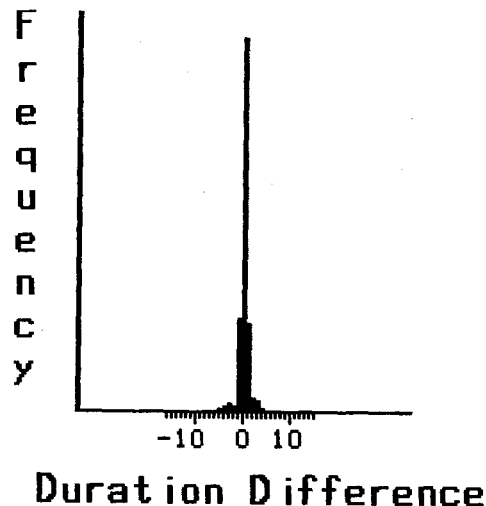


Figure 2: Distribution of the first differences of the duration pointers for the notes in a subset of the Bach chorales. Note the predominance of the values 0, +1, and -1.

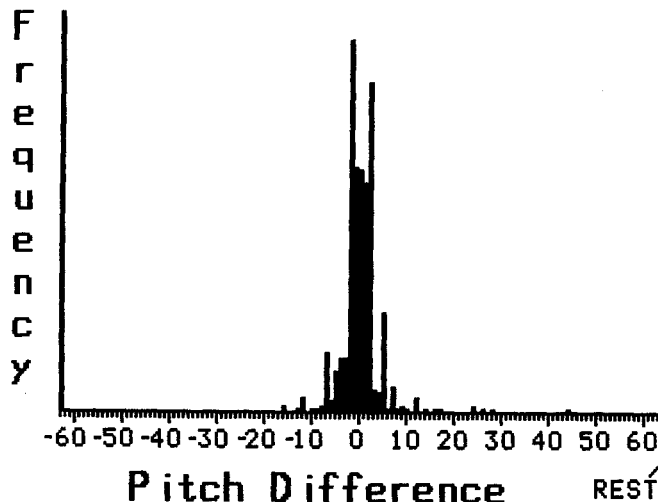


Figure 3: Distribution of the first differences of pitch.

Figures 2 and 3 show the distributions of the first differences for both the pitch and duration of the notes in the Bach chorales. These statistics were compiled from a subset of the 371, but we can safely extrapolate the results. The duration table used for this analysis was composed by the following rule: take all durations not equal to a sixteenth, eighth, quarter, half, or whole note, sort them, and then tack the aforementioned five durations on the end. This has the advantage of keeping the common (non-dotted) durations adjacent.

These graphs confirm our suspicion that the first differences are not evenly distributed, being weighted in favor of the smaller absolute values. Such a distribution can be efficiently encoded by a *variable-length code*. Such a code uses shorter codes (fewer bits) for the common values and longer codes for the rarer values, with the codes chosen so that no code has a bit pattern equal to the high-order bits of a longer code. This is necessary for decoding to be possible.

Since the duration values of 0, +1, and -1 are very common, we will encode these with the two-bit codes 00, 01, and 11. Note that these are precisely the two's-complement values (to two bits) for these numbers. Larger duration values will be encoded by the code 10dddd, where ddddd is the two's-complement value for the duration value. We call this a "2+5" code, since it uses two bit groups for short codes and 2+5 bit groups for long codes. Note that, although it is not apparent from the Figure 2 (since it reflects only a subset of the 371 chorales), there are some very rare occurrences of differences greater than 8 in absolute value. This is why a 2+5 code must be used rather than a 2+4.

For pitches, two bits is not sufficient to handle the common cases, so we will use 3 bits. Larger values will be encoded as 100pppp (values up to 4 bits), and even larger values will be encoded as 1001000pppppp. Thus for pitches we have a "3+4+7" code.

Each note will be encoded duration first, then pitch. A few special cases need to be handled. A duration value of -8 will switch to byte mode (and no pitch information will follow). A pitch value of 7 will denote a rest.

Putting this all together, when we compress all the chorales we get an average size of 344 bytes, for a compression factor of 3.33. We have actually implemented this algorithm exactly as described, and have indeed succeeded in fitting all 371 chorales of J.S. Bach on a single diskette (albeit just barely - we had to remove DOS from the diskette in order to make them fit).

```

No. of Exceptional Durations
Duration 1
.
.
Duration K
(whole, half, etc. are implicitly
a part of the duration table)

Pitch of first note
Duration pointer for first note

No. of Voices
Pointer to voice 1 (2 bytes)
.
.
Pointer to voice N

<MUSIC DATA>
(See Text for Description)

```

Figure 4: Format of a compacted data file

Summary

The format of the entire compressed music file is shown in Figure 4. Note that, since the encoding works with the first differences of the note pitches and durations, we have to store the pitch and duration of the first note, in order to get the decoding algorithm started. The data is encoded using the two-mode bit and byte compression scheme described above, with the 2-4/3-4-7 code for notes.

The decoding algorithm is summarized in Figure 5. This algorithm has been implemented in 6502 assembly language, and does indeed satisfy our speed requirements; decompression never takes more than a few tenths of a second.

```

duration table = exceptional durations + {1,2,4,8,16}
copy #voices and voice pointers to output

last note = pitch of first note
last dur = duration pointer of first note
mode = byte

[†] if (mode = byte)
{
    byte = next 8 bits from input

    if (byte = 0)
        output 21-byte C1 sequence
    if (byte = 1)
        output FE FE FE
    if (byte = 2)
        mode = bit
    if (byte = 3)
        quit /* end of data */
    if (byte = 4)
        output FC FC FC (= nop)
    if (byte > 192)
        copy it and next 2 bytes to output
}
else /* mode must be bit */
{
    dur = next 2 bits from input
    if (dur = -2)
        dur = next 5 bits
    if (dur = -16)
    {
        mode = byte
        goto [†]
    }
    note = next 3 bits
    if (note = -4)
        note = next 4 bits
    if (note = 7)
        note = 192 /* ALF value for rest */
    if (note = -8)
        note = next 7 bits
    if (note <> 192)
    {
        note = note*2 + last note
        last note = note
    }
    dur = dur + last dur
    last dur = dur

    output "note" and 60*duration table[dur]
}
goto [†] /* repeat until end of data */

```

Figure 5: The decompression algorithm. This algorithm is best implemented in assembly language due to the speed requirements and the extensive bit manipulation.

Possible Extensions

The compression factor of 3.33 achieved thus far is still not the best possible. In fact, the variable-length code used above, which was chosen mainly for decoding simplicity, is only one of a large number of possible encoding schemes. The question of which one of a multitude of variable-length codes is optimal was answered long ago [1]. Such a code is called a *Huffman code*, after the discoverer of the algorithm for its construction.

Given the statistics in Figures 2 & 3, we computed a Huffman code for encoding pitch and duration first-differences. Although we have not implemented it, we calculate that this code produces an average compressed chorale length of 304 bytes, for a compression factor of 3.77 and a total size of 113K. Each note requires only 5.26 bits of compressed data.

Even though these results were presented for a specific music source format (ALF), the general techniques should be applicable for any system. They should also work for other types of music besides the Bach chorales, although the compression factors will vary depending on the degree of "well-behavedness" in the music.

Yet further refinements are possible. Encoding of the *second-differences*, at least for pitch, might yield further compression, due to the commonness of scales in music, for which the second differences are small.

Finally, note that all these are "dumb" compression schemes. A "smart" compressor would recognize repeated phrases, correlation between the voices, and other high-level concepts which could dramatically increase compression still further. These possibilities are left for the mavens of artificial intelligence.

Acknowledgements

The author would like to thank W. McGee of Ontario for the music data that made this research possible, and R. Mercuri for posing the original "fit-it-all-on-one-diskette" challenge.

References

- [1] D. Huffman, *Proc. IRE*, 40 (1952), pp. 1098-1101.

Computer Music and the Human Interface:

"Imbedding Performance Knowledge at the Graphics Level"

Steve Levine

Software and Systems
P.O. Box 621 Snyder, New York 14226

Abstract

This paper presents an illustrated discussion of the key considerations surrounding the issue of "Built-in User Friendliness" with respect to computer music systems. The central theme will focus on the use of pointing device input and color graphics on a microcomputer to control the composition and performance of music. A comparative study will be made of past, present and future trends that contribute to the overall awareness of the music community with respect to technological approaches to this issue.

Scope

For the sake of clarity and space, this paper will focus on performance control. Once we develop an awareness of the required system, we can begin to create a metaphor, using computer graphics that will simplify our interaction with the system.

Here come the "toys"

With the advent of highly available integrated computer graphics and built-in music synthesis systems on personal computers, the computer arts hobbyist can now explore modes of abstraction heretofore unaccessible. Inexpensive touch tablets such as the "Koala-Pad" and "Chalkboard" have made it quite affordable and simple to add pointing input capability to graphic based applications.

A typical home computer that has graphic and sound capability can be purchased for less than \$1000.00 in most cases. This includes from at least a 256X192 to 640X200 pixel graphic resolution with up to 16 colors at once. This was not possible when I started fooling with computers. By going with any one of a half dozen popular models, the average user is able to purchase inexpensive add-ons such as memory, interface cards and music synthesizers. Software of many different categories can also be acquired for very nominal prices.

Mass Discovery

The effect of readily available low cost tools such as these has had a profound evolutionary effect on the proliferation of software and hardware. Many more individuals are staying up late and writing programs or building boards that

make music and art. We have accumulated a lot of knowledge in recent years to feed the movement at large.

User-Friendliation

Since there is more humanity involved in developing user-interfaces there is bound to be more user-friendly software awareness in the air. I would not imply that this software will be user-friendly by nature, but there is a good chance that more of it will be found.

Nice programs come about by programmer awareness of the way in which people do things most naturally. Machines have been, by and large, quite user hostile. This is due to several factors, some being economic, others physical constraints and just plain ignorance. The "average user" concept applies pretty much across the board. What is desired is for a computer or even a car for that matter, to anticipate in some way, the operations that can be automated and present the remaining variables in a form that is comfortable to humans. This task requires some study of the particular problem being solved by the computer and the group of users that will interact with the system.

Some basic archtypes

Let's consider some of the devices within the context of this paper, first in a general realm and then in the musical realm and try to examine their relationship to the user.

Knobs, Levers, Buttons and Switches

These objects have been with synthesizers since the beginning, around 1950. Most electronic composers felt that these were representative of a fairly humanized control schema and were willing to work with them. There were of course, the mavericks who felt that there was definitely a better way to control many things at once. There was an implied accessability in being able to reach out and change one parameter out of a panel of a hundred. Many schemes have been devised since the early days which attempt to simplify the task of controlling many variables in a performance session.

A knob, for instance, connects to a rotary control which normally provides 300 degrees of

rotation. This rotation is translated into continuous control signal and applied as a varying parameter to some kind of music or sound context. If there are 4 channels of sound and there is one rotary control for each, then to mix four music signal's loudness using these controls, all one must do is rotate the respective knob for each channel whose loudness needs to be changed. This action is very straight forward when the correct legend is printed on or near the knob in question.

Switches have similar demographics. If a switch is designated to effect an action and it's function is clearly marked, then again the user must just select the switch and press it. A good switch is one which has some kind of status indicator associated with it's current state. There are switch schemes in use that use a momentary switch, one that is spring-loaded and only makes contact as long as it is held down, to cycle through several states, even though the switch only has two states itself. With these, there is definitely a need for a status indicator.

Indicators

Simple indicators are used throughout all of society. Warning lights, beepers, buzzers and so on. They prove their effectiveness when they either succeed or fail to inform us in time to take the appropriate action. A whistling teapot tells us that the water is boiling for our tea. An alarm clock naggs us to wake up. Floor indicators in an elevator show us which floor we are zooming past and generally rings when we come to the floor that was selected.

A good example of a volume indicator is when the pumping gas into an automobile. The driver may be running the engine while the passenger (in a self-service station) pumps the gas. If the amount of gas desired is not the same a full tank, then the person pumping usually watches the dollar amount on the rolling "This Sale" display on the pump. At the same time the driver may be watching how much (or little) gas he or she is receiving. If it's the driver's money, then it usually won't do him much good to try to estimate the dollar amount based on the tank full indicator. It would be best in this case to simply say "it's at 3/4 full, that's enough." There are many ways in which to indicate things, some prefer a volume indication with no more than 1/4 full resolution. Others require graduations on a finer scale, such as a numeric indicator along with a volumic one. This is where the situation requires a human factor analysis for the particular task.

Graphic Metaphores

The number of variables of interest in a situation such as ours urges the use of efficient demographics. It may be possible for the military to train soldiers to memorize the function and position of a hundred controls or a nuclear reactor operator for that matter. For the musician or composer, these things tend to compete for his or her creative time and energies.

Using computer graphic techniques, it is possible to create a set of "Icons" and "glyphs", which

are just environment-specific symbols which represent various functions and controls. These Icons may be optimally designed to take full advantage of the standard nomenclature used by the synthesis community. They may also be re-configurable to suit the individual needs of a particular composer.

A good example of well designed metaphores is that which can be found in the Macintosh computer, by Apple. The "MacPaint/MacDraw" programs make use of an icon set which represents actions and devices commonly used by artists and paste-up persons. For example, there is a little spray-paint can icon that represents air-brush technique. For the non-artist, first time user of the package, this will not be obvious but it will foster some immediate curiosity and inevitable experimentation. After one try, it will become obvious to the user what this symbol means. For the experienced artist, this should represent something that they are intimately familiar with, if only by one level of abstraction removed. There is also an eraser symbol which when selected, causes the images that are swept over it by mouse movements, to be erased. A lasso icon is used to effect a "cut and paste" operation on sections of the image.

Again, these symbols may not make a lot of sense to everyone, but with minimal familiarization they become as recognizable as a hammer or screwdriver in a tool box.

Based on this approach, it is apparent that we can make the most efficient use of iconographic representation if we consider the issue of symbolic familiarity.

In the context of computer synthesis applications, we may be creating a new dialect for symbolic representation. This doesn't mean that we should abandon familiar metaphores in the design of the new ones. It will make it much easier to gain acceptance by both the synthesis community and new devotees.

Pointing

One metaphore that has gained wide acceptance by computer users is the notion of "pointing". This comes natural to anyone who has been asked to select a teddy bear at a carnival, after knocking over 3 cupie dolls with a baseball.

The notion is simple. There are several items on a list in front of you. You are asked to select some. You must indicate which ones you have selected. You may not speak the choices. You have a pointer symbol that is shaped like an arrow which you may move with your hand. By moving this arrow to the position of the items of choice you may then select items by pressing a "Yes, this one please" button.

There are several implimentations of this metaphore currently in use.

The Joystick

The joystick is used extensively in video games to position gunsights. They usually contain buttons to "Fire" at the enemy ships. The joystick however, is not the best for picking things on a graphic screen. It is made up of a 2 or 3 inch handle connected to 2 variable resistor controls. When it is moved, it produces a horizontal and vertical signal which can be related to X and Y movements.

Digitizing Pads

Digitizing pads come in a few different shapes and sizes. There is the older more common "Bit Pad" which uses a pen-like stylus connected by a wire to the main surface and related electronics. This pen is used just like a pen; it is drawn over the surface or just placed at specific locations on the surface to effect pointing operations. There is usually a spring loaded switch in the pen body and tip that allows "Yes" functions to be accomplished by just pressing the pen down a bit harder than usual. An alternate pointing instrument that goes with the bit-pad is a "Puck" device. This is shaped like a round puck that fits between the thumb and forefinger, and is moved around on the surface by sliding. There is a plastic protrusion which contains a cross-hair as a reference for accurate placement of the pointing apex.

Touch Pads

Another type of digitizing pad is a "Touch-Pad". There are several of these available, but they all work pretty much the same way. The user may simulate the action of the digitizer pad by using only a finger or a special wooden stylus which is not connected to anything. It helps to use the stylus if a greater resolution of choice is desired. The finger works well for most music synthesis applications. The touch pad surface is made to represent the working area of the screen. By placing the finger in different areas of the pad, a pair of coordinates are generated just like a joystick, but they are represented by a 2-D flat device. There are usually a couple of buttons on the side of these units to facilitate "yes" functions.

Touch Screens

There is a device called a "Touch Screen" which allows the user to use a bare finger to point right at something on the video screen. This uses special electric field effects to determine the coordinates of the finger on the glass of the screen. A set of squares or circles are first drawn by the software. Then labels are written inside the shapes. These become the choice symbols that you point to. Instead of moving some kind of pointing icon on the screen to the button of interest, the finger against the drawing of the box or circle causes the computer program to know which area has been touched, therefore lighting up the "button" image on the screen. This is the quickest way to point to something on a computer. Unfortunately, this is rather expensive and turns out to be low resolution in most cases.

Mice

The last pointing device I should mention is the "Mouse". This is the pointer that is used with the Macintosh computer. It is a puck-like device that has a ball mounted on it's underside. The mouse is rolled on a table or any surface that is dry and free of debris. As the mouse is moved on the table, a similar movement takes place on the computer screen with the graphic pointing icon. The mouse contains mechanical and electronic devices within it's puck to generate data corresponding to 4 directions of movement;

up,down,left and right. With this information coming to the computer as the puck is moved on the work surface, a "heading up and to the left" or "down to the right" kind of thing is derived from the mouse.

The author prefers a touch pad of at least enough resolution to cover the full screen points. It is also very nice to add pressure sensitive devices beneath the pad in order to get a "Yes" from pushing down real hard on the surface.

Graphic Considerations

I will now discuss some of the computer dependant considerations we must explore when implimenting a graphic based human interface for music.

The first thing that comes to mind when considering a particular graphic environment is that of display resolution. This is a measure of how many dots may be controlled by the computer display interface. These dots are usually referred to as "pixels", meaning picture elements. This parameter is a function of hardware, specifically memory. Of related importance is the number of simultaneous displayable colors. Again, this is a function of memory.

In order to represent musical symbols comfortable, that is with adequate resolution to discern between similar symbols, we need at least 512X200 pixels. It is nice to have at least 8 different simultaneous displayable colors to choose from.

Black and white with some grey scale control can be used as well. The Macintosh screen is very good in this respect, as it has 512X342, which is better than 512X200.

A second issue is display speed. This is almost as important as the former attributes. We all would like to have fairly speedy computer displays and computers for that matter. In the case of our music application, we may at times, require the display to scroll or update some graphic information while it is in the midst of a synthesis operation. We definitely don't want the display operations to slow up the main processor when it is sending information to the synth!

An acceptable display speed should be based on the requirements of the interactive portion of the software. For example; if there are a number of controls represented on the display and you go to position the pointing icon over one of these, the icon should move smoothly in conjunction with the pointing device movement. Certain actions such as screen clear, line and circle draws etc. should be reasonably fast, no more than a half second at most. A good measure of display speed is how fast it takes to redraw the entire screen with geometric symbols being generated in software.

Graphic Software

I prefer to write graphic programs in a high-level language such as Pascal or 'C'. I use basic only to experiment with ideas, not to produce efficient, well-structured programs. Assembler is used to write the low level routines that are extremely time sensitive. I should

point out that most video games are written entirely in assembler for the simple reason that they have to be hand optimized for memory usage and speed. Some of the fanciest utilities are imbedded in these assembler written games, which can't be done with a higher level language. The amount of movement on the screen is very crucial in games, an area where a substantial amount of graphic expertise is required.

Many personal computers contain Rom subroutines for performing primitive graphic functions. These routines can be accessed from the basic interpreter or from some compiled languages.

Language Constraints

As was mentioned just above, graphic operations have to be coded as efficiently as possible in order to avoid uncomfortable delays in the update of the screen. The most time critical areas are those which take place with respect to an input action. A good example in our application is that of envelope function display. Refer to figure 1.

When the line drawing of the figure is being changed, there shouldn't be a noticeable delay in the time between drawing segments. The value of the envelope segment being worked on has to be shown in the little window as well. This may turn out to be a function of the most primitive operation in the computer roms, such as the line routine. Most personal computer manufacturers implement this in such a fashion as to keep it in a standard form that will connect other programs to it the same way each time. Most of the time this is not the most compact and fast way to accomplish it. If we want to optimize this line drawing function for instance, we could write our own in assembler or 'C', Pascal etc. Doing it in Basic will most likely use the built in system routine that we just spoke of. For those of you that wish to implement their own line and circle drawing algorithms, along with a host of other primitives for graphics, see the reference Foley, Van Dam, [1982] Chapter 11. for an excellent tutorial on interactive computer graphics.

In most cases, there are function libraries that are written for popular compilers that include efficient algorithms of this sort. The advantage of writing them yourself is that they can be optimized to suit your application.

Structured Programming

Another issue that comes heavily into play is that of interactive development. This becomes obvious when you attempt to write software for a graphic application and the time in between getting a changed piece of code going and the last piece of running code becomes quite long. Here is where interpreted Basic becomes tempting. This is fine for just playing around and testing algorithms, but when it comes to finished programs, Basic falls on it's face in most cases. The high level structured languages such as 'C' or Pascal, Fortran etc. provide advanced and powerful data structures to specify and organize your tasks in a clear way. If one takes the time to learn one of these languages, code eventually starts looking like the original algorithm and flowcharts become unnecessary. Programs become extremely readable, easy to fix and find bugs in and they become very Transportable.

This last item is of great importance in computers in music and arts. After all, there aren't a lot of companies supporting these kind of efforts and most computer music and art software is being developed by individuals! So, wouldn't it be nice if these efforts could be pooled? As I stated earlier in this paper, we got this far from the grass-roots movement anyhow. Program portability is the way that we can share our work with other enthusiasts.

The System Approach

When I speak of systems with respect to computer music, I naturally accept the fact that we need an adequate environment in which to work. First, the computer must have moderately quick disk drives. I cannot be satisfied with serial (9600 bps) drives, such as on the Commodore 64 or Atari. These machines are great for music experimentation but due to many hardware constraints, they remain in the semi-serious realm for me.

A decent system should have more than 64k of memory. The reason is that there has to be some way to allow most, if not all of the program to reside in main memory at once. In addition to the program, there has to be room for some data, such as notes, and graphic screens to be displayed at different times. Some micros have the provision for storing a whole floppy disk worth of data in memory, in order to simulate a disk. This permits extremely quick file input/output and allows the program to be very large, without experiencing the delay of loading from disk. (overlays).

As I pointed out before, program speed is important to music development. The most crucial aspect of working with a computer to develop music is to be able to hear and see the changes that the user makes, as soon as possible after they are entered into the computer. This closes the loop between conceiving the piece and hearing it, which leads inevitably to changing it again, and so on. This phenomenon applies to computer graphics just as importantly.

The Operating System

The operating system is also important. We would like it if there were features built in to the O/S that enabled our programs to execute with ease, rather than having to create strange code that takes the place of the normal operating system features. An example of this is the ability to call the system with a function number and request it to perform some kind of input or output operation, without us having to write it ourselves. If we want to hang a Bit-Pad or a mouse on the computer, we don't want to hack up an assembly routine just to address the individual serial port. It would be much cleaner and easier if there were built-in device independant system calls or standard device drivers that handled the low-level I/O. This would allow the transporting and adapting of our programs without having to worry about the primitive details.

Other O/S considerations are, of course standardization. At the moment, Apple-II's have the easiest, most portable I/O configuration around. All you have to do is stick a serial card in a certain slot, such as slot #1, then you can refer to it by typing "PR#1" and forget it forever! At the moment, IBM-PC's have the most

well structured and documented O/S in the micro world. Cheap cards can be purchased which pop right in and perform some function. The binding to the O/S is very straight forward. A person can write their own device driver or high-level program without worrying if it will work on someone else's machine. There is plenty of information around to assist the do-it-yourselfer. The Macintosh has some catching up to do before this can be said about it.

One last consideration of the system, is that of availability. If one is to take a serious direction with computer music development, then it would be nice if this were a popular enough computer that there were plenty of cheap software and hardware available for it. There are only a handful of companies that are developing or offering music systems or cards. Most of these are only able to focus on the personal computers that have sold well. This isn't bad, though. The high volume of sales drives the music system prices down, making it easier for you to spring for that extra add-on above the cost of the computer.

Think hard and do some investigation before you decide on the P.C. of your choice.

Music Application

After discussing some key issues with regard to our subject, we can now take a close look at using computer graphics to represent performance knowledge.

It should be pointed out that the system being described refers to a hypothetical software and hardware arrangement. The concepts apply to analog and digital synthesizers interfaced via the Musical Instrument Digital Interface (MIDI) and built-in digital synthesizers, which are available for some of the popular personal computers.

In order to adopt a methodology which is suited to several situations (other than performance) we must consider what modes of commonality each of these will share. The best place to begin is at the topmost level of the "hypothetical computer music system". Refer to figure 2.

As you can glean from the figure, we have a few function levels that may want to effect or communicate with each other.

The "Score" level can have many meanings. William Buxton from the University of Toronto Computer Science Research Group, who has been responsible for many developments in the area of the human interface, has stated simply that "A score is a score is a score.....". What is being said here is that in musical performance, all of the parameters concerning the composer that contribute to the final sound output can be organized into some kind of score file. Scores that have many items in it, such as the waveform specs, envelopes, performance parameters such as vibrato, glissandi and the like and the music notation itself are not uncommon in some computer music pieces. If every parameter can be written with respect to its occurrence in time during the playing of the piece, then why not place it all in the score when composing the final phase of the work?

The way in which I have chosen to relate the different functions at the editing or top level is with a separate operation for each. So, the Score editor handles music notation entry and modification, the voicer handles sound analysis and specification, with preview and the player handles performance. This is not to say that data won't be relayed between the three functions.

The commonality of the three top-level functions is seen in two ways;

First, the editors may be invoked at any time in another editor's life. You may be editing a score and you desire to edit a voice, all that is required is to invoke the voicer and edit voices while the score editor remains in its last state, semi-visible on the screen, to be returned to at a later time.

Second, some variables parameters that are derived from one editor may be fed to another to provide some kind of function. Obviously, the score editor may be limited in the type of data it can provide to the other editors. Conversely, using the voicer one may create an ascending staircase function that can be plugged into a composition algorithm which generates an ascending series of notes based on a rule which determines the intervallic spacing. Also, the play editor, which relates more to the real-time aspect of performance, would be more than happy to associate such a staircase with some live parameter that effectively sweeps pitch. A live control from the play editor could be used to "perform" a function into the voicer or an envelope can be developed with the voicer that is shown in the score following the onset of a note.

With this approach, we can flexibly associate parameters between functions without having a total merging of the three editors. In this way we retain a separate but powerful hierarchy which has been found to be effective for newcomers to computer music.

An approach to defining the interconnection scheme for the various edit functions and their parameters may be derived in the following way;

- Segment the system as was done in the generic block diagram (or to your preference)
- Define the list of parameters that will be associated with each segment and their individual limits, uses etc.
- Propose a relation strategy for these parameters by introducing them to each other, ie; Live Pitch, meet Envelope function (will that work or not?) Timbre Control, meet Note Score (wah).

Once the interconnect scheme has been outlined, we develop scaling and ranging black-boxes that can be used to massage the variables when they are going to be used across editors. This may be needed when, for instance, the values coming from one generator are too large or quick for another function input. This is essentially a simple form of signal processing, not unlike the metaphors of patching analog synthesizers.

Windowing

The term "Window" should be familiar to anyone who has seen the Macintosh. It uses a system of "pull-down" windows or screen segments that contain separate objects of information, in text or graphic form, that can represent a function or job that is running at the moment while the job that is covered by the new window is still continuing underneath. This may not seem straightforward at first, but consider the next example as pertaining to our music application.

Suppose we have score editor and it's purpose is to allow us to write a musical passage with conventional music notation. See figure 3.

What if we have developed a certain amount of the score, and while still in the score editor, we now desire to hear an instrument in order to make up our minds about the next score entry. It would be like fine-tuning the work in progress, as we go along. Wouldn't it be nice if we could just pull down a totally different editor menu for making instruments, while still being able to see and effect what was going on in the score editor? This sounds pretty high tech, doesn't it? I should point out that this type of function, known as "Window Management" is becoming quite commonplace. It makes a world of difference for the user interface portion of a system.

Figure 4 illustrates pull-down multiple windows. The commonality I spoke about should begin to surface from a using standpoint. Clearly, there are several combinations of routings for the different system functions when developing or performing some music. This shows itself in the score where the envelope, which is normally an amplitude vs. time function is being changed by the envelope editor. Although the old envelope is not shown here, it represents the shape of the envelope in the second bar of the score. The way that this would be used is by the definition of the envelope in the envelope editor, the tagging of the finished one and the placement of it into the score.

Windows are not entirely necessary. If they are available in the computer you are working with, then use them. A very useful system can get along without them as well. Many of the digital

syntheizers that use graphics, adopt a "Page-Turning" concept. This is done by setting up several individual screens where the editing work will take place and allowing the user to request a new page or return to an old one at will. Once the page is turned, the new page becomes the scene of focus. The old page may be returned to with the previous work left intact.

One other thing that is bothersome is the time that often ensues between page turning. This method fails to leave something on the screen while the next editor is being overlaid, such as in windowing, so the interest is held and it doesn't seem like a long Compile or something.

The Recording Studio Metaphor

Let us examine a recording studio situation that we would like to computerize. In this discussion we will combine the synthesis, composition and record/playback functions together into one system. We will try to define the areas which lend themselves to graphic modeling and then devise a philosophy for codifying them.

In our simple home recording studio we have these devices;

- 8 channel computer interfaced digital synthi
 - o timbre control
 - o note pitch
 - o amplitude control

- Personal computer
 - o Black and white window graphics
 - o Floppy disks
 - o 128k memory
 - o touch pad

Figure 5, the system diagram shows the various components.

The next block diagram, figure 6, shows software already created.

Using this configuration, what will we want to accomplish in our recording studio?

The Tape Recorder

First, there has to be a "Tape Recorder". This can be a software function that allows us to record the work that we create, into memory and then floppy disk. There is a fast forward and rewind that allows random access to certain positions on the tape. We would like to record different types of information next to each other on separate tracks. These are parallel areas on the tape that contain sound or control information that is recorded or played-back together in time. On a real tape recorder, you can edit the sound on the tape by "Splicing" pieces of tape containing the sound portion that you wish to join with another. You can also "Punch-in" which is just recording at a precise time and position on the tape and "Punch-out" or stopping at the exact moment that the time window ends. These functions are simply file operations.

The Mixer

Next, we would like to have some sort of mixer. This device normally allows the musician/producer to add together several sound tracks coming from tape, live instruments or voice into a multi-channel output such as stereo or quad. The mixer is primarily used to set the individual amplitude levels of instruments. In this way, you can "Mix" the levels down into a composite using your taste for combining them.

The Synthesizer

As stated above, the synthi has a certain amount of controllable parameters such as pitch, timbre and amplitude. Each of these have several sub-parameters that are used to perform or compose with. These sub-parameters are derived by supplying time, frequency and amplitude varying functions to the variable sections. Some of the sub-parameters are;

- Vibrato depth
- Vibrato speed
- Timbre X (+/- 2048)
- Timbre Y (+/- 2048)
- Timbre Z (+/- 2048)
- AmplitudeEnvelopeBreakpoint (rate,value)
- Portamento speed

- Glissando speed
- Glissando interval

These parameters exist for each channel of the 8 voice synthesizer.

The heuristic device

Earlier in this paper we discussed the psychology of symbolic familiarity. This is very important when devising a methodology for iconographic design. In our recording studio metaphor, there has to be a general understanding of the devices, controls and movements involved in making music this way.

Most of what is known about recording and synthesizing has been acquired by professionals who make music for a living. There are many individuals who also design instruments and recording equipment who have to be aware of the professional heuristics that come into play when a non-technical person interacts with a machine. Some of the components of the modern studio require quite a verbose manual that is normally not that simple to get through.

Is "Less" beautiful?

The current wave of inexpensive electronic instruments are making use of low panel real-estate, or in other words almost no front-panel controls. Many manufacturers are providing just 1 or 2 controls and a stacked selector /mode switch scheme. This requires that the musician keep a good copy of the operational aspects of the system in their head. Most of these instruments have one very small (2 - 16 digit) display, one or two slider controls and many cheap membrane-type switches, like the ones on new microwave ovens. Some include a printed road-map of the patch possibilities of the synth, in order to simulate a menu-driven system. This is not as bad as it sounds. This scheme allows manufacturers to make instruments at a much lower cost than before. They can eliminate much of the expensive and un-reliable hardware that is associated with knobs, buttons, and faders etc. This keeps the cost down to where the average person can afford one. These instruments are usable but are more prone to error.

Do it like the 'Pro's'

If the system is going to be easy to learn, modify and work with, then it has to take into consideration how a musical piece is made. How do we know this before we have a system to try it on? As I said earlier, much of the producing knowledge already exists. If we adhere to some of the time-honored traditions of the professional recording business, we will have an easier time building in this performance knowledge.

The following is a short list of operations for the creation of a musical piece;

- 1 - Write musical phrases (score)
- 2 - Make a simple instrument (voicer)
- 3 - Play the phrase with the simple instrument (play)
- 4 - Revise the phrases
- 5 - Make more instruments
- 6 - Try multi-part playing
- 7 - Revise the instruments (add effects)
- 8 - Mix /Play the levels of the different instruments/parts

- 9 - Record on multi-track (in the computer)
- 10 - Edit tracks, Punch-in/out
- 11 - Do final mix down

Using the basic operations as a guide, we can study the things that require built-in knowledge.

A detailed discussion of the score editor is beyond the scope of this paper, therefore we will confine ourselves to voicer and player. Many good examples of score editing have been developed in recent years. There is a fine paper by Mercuri, R [1981] and Buxton, W [1979] cmj(3,4) The evolution of the SSSp score editing tools]

Voicer Smarts

To incorporate some knowledge about voicing a synthesizer into our "Voicer" software we have to consider what important functions are necessary. The voicer screen shown in figure 7, contains only a small portion of what can be implemented. Our voice making editor should allow us to create arbitrary functions consisting of line segments. It should also allow us to graft together different types of waveforms such as the top 4 waveforms in the figure. Not only would we like to paste and cut, we would also like to manipulate them once they are on the screen. How about a bending function? What would we do with it? Suppose you have an envelope function such as the 5th function down on the left. Amplitude is represented in the vertical (Y) axis and time in the horizontal (X). (see figure 8)

If the function was actually a pipecleaner and we grabbed it at the middle of the top line as in the next figure, then bent it down towards the bottom, it would turn out looking like figure 9.

The "Bend" function is a rubber-banding line routine that knows how to draw a triangle using the new coordinate that you point as the apex and redraws the two sides everytime you move the apex a little. This saves the step of having to erase and redraw the lines by hand. This example is a very simple one, but represents the kind of things that can be done using built-in graphic functions.

Response Curve Editing

Another kind of operation that provides a very useful function is that of response curve editing. This makes it possible to create a function that will be used as the response for a control input. For each value that is read from a joystick or fader control, the response curve is used to treat the value according to the shape of the curve.

Using the Voicer editor, you can draw a response curve and patch it to the joystick in the Play editor. When you move the joystick, the response curve that is patched in will be used as a "Filter", in other words, Joystick movements in ----> | X | ----> transformed values out. This is essentially what a filter does. This can be done with a simple table look-up operation. The weight of the joystick value is used as an address for the response curve, which is stored in the table, the values in the response curve table are then read out in place of the joysticks.

An example of this is an inversion function. If a line that goes from max to min value is put into the table starting at the lowest location going to the highest, then when a low value of joystick is used to look up the response curve, then a high value will be read out, and so on. (See figure 10)

Teaching the System about You

Most computer programs are not set up to learn from it's users. Programmers, on the other hand, have been teaching computers how to do things their way. There is an obvious distinction here. On the one hand, programmers build-in algorithms into their programs that usually perform a set of operations according to a set of input data, usually provided by the user. On the other hand an artificially intelligent, or "Expert System" program is set up to remember inputted knowledge which comes from the user and later make deductions or inferences based on this set of rules, when queried in more abstract ways than usual. In the music application, why not program the computer to store and compress performance expertise as we gain it?

This is, of course, not as involved as some of the advanced efforts going on in A.I. (Artificial Intelligence). We are only trying to sample real-time performance input and modification sequences as we perform in the Play editor.

The MIMIC function

If all the parameters on the screen are variable when you go to perform with the system, at the play level, then why not store the progressing movements of the faders, or the adding of a new voice during the performance? Any of this can be considered live once it has been set to some timing. If you speed up the tempo the whole stored performance should speed up, including the fader movements etc.

Implimenting MIMIC is just a matter of using an internal clock to synchronize and time tag the events of interest. At the tick of each clock, all the desired parameters are sampled from the system and a history of their movements are taken for a finite amount of time. This can get very bulky, data wise, if it's done for too long. A way to reduce the parameter storage is to simply store the deltas, or changes.

Many sequencers store keyboard and velocity information which is played in by the musician. Very few try to capture performance parameters as in MIMIC.

Music objects and PLAY

Many diverse textures, sequences, timbres and rythms can be derived from a modern digital music synthesizer. With the addition of a real-time interactive graphics computer and some of the components that we discussed above, a whole world of new possibilities open up.

It is quite an experience to be able to manipulate an enormous patch panel of synths in's and out's on a computer screen, each one having some kind of immediate effect on the sound. The

great part, is that all of the best patches can be stored, transferred and manipulated by the computer.

When we work with highly detailed systems after a while, we tend to organize things into macro-things. These macros are made up of combinations of the smaller items which we find work well together. There are of course, nearly endless combinations of smaller things yielding more macros.

Say we want to work with single small functions such as the ones in figure 11.

When this thing is done we may want to call it "Warble" for instance and save it as an "object". We could just take the LFO and VCO combination and call it "Warblosc" or something. These two sets of low-level functions are objects themselves, but may be contained in larger objects as well.

Other objects may be related to performance methods. We can devise a particular play sequence made up of inputs and their associated response curves, routings and interconnect conditioning.

Icons revisited

In dealing with object oriented structures, we will have to design the ideal set of icons to represent these objects. In order to get a sufficient amount of iconographic data on the screen, there has to be a compact philosophy adopted.

How do we compress graphic information sufficiently to contain it on the screen? At the same time we would like to remember what these symbols mean without having to look them up each time.

First, I suggest a smaller window onto the larger world. This is illustrated by figure 12.

With such a system, you may slide left or right, up or down, by moving into the north/east/south/west box on the lower right hand portion of the screen.

Second, the idea of families of symbols. When a set of icons which represent various manifestaions of the same family of objects take on similar graphic attributes they become easier to group, visually and contextually. This is the key to icon design in a crowded situation.

Symbols do it

Figure 13 is a symbol breakdown showing the different attributes that distinguish one member of the family from another. Also in the figure is the different family types. The figure relates to the PLAYER and VOICER screens.

The symbols shown in the last figure are tagged with a specific function on the "Voicer" or "P layer" screens. In this way, the object oriented editor can connect or corral these together and call them a "Macro" object, give it a new name or symbol and store it on disk. They now seem more like a set of building blocks with pictures on the faces.

Figure 14 shows a typical arrangement of objects and macro-objects in a performance situation.

Conclusions

This might appear to be a "patch editor" of sorts, but is truly an "Object - Oriented" one at best. By defining the inputs, outputs and limits of these objects, they may be treated as subroutines from the computer's standpoint and physical blocks from the user's.

Developing an object oriented editor which allows graphic icons to be "patched" together is not a simple task. I have found that using some of the methodologies mentioned herein simplifies the tedium of designing the data structures. It also helps to study the important functions in terms of their real-world analog application.

A very good discussion of this subject may be found in Buxton, W., et al [1982] and Kowalski, M.J. and Glassner, A [1982].

Once these objects are associated with functions, they may be connected visually on the screen and their effect can be experienced at once.

It is hopeful that synthesizer technology will keep up with the rapid growth of software knowledge which is increasing with leaps and bounds.

Bibliography

Foley, J.D and Van Dam, A [1982], "Fundamentals of Interactive Computer Graphics, Addison Wesley. pp. 431-476

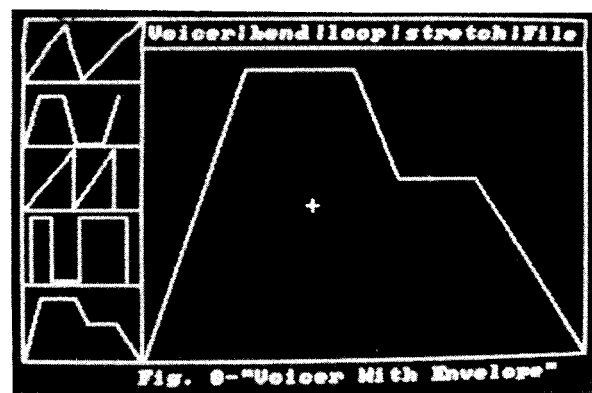
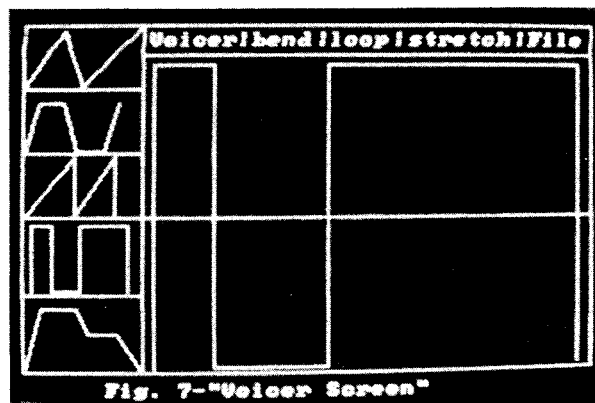
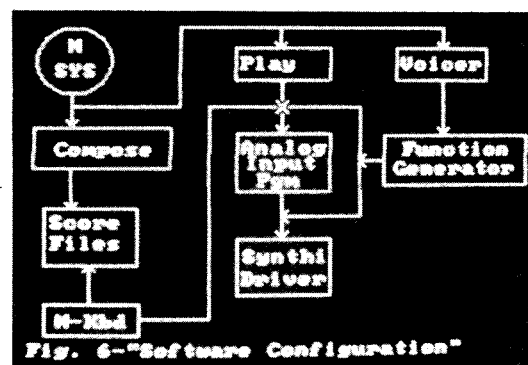
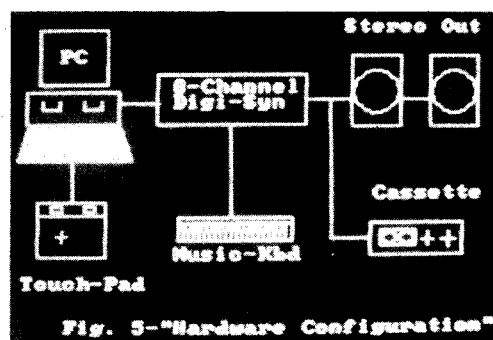
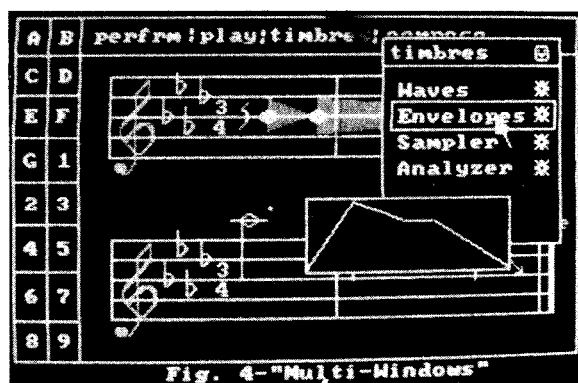
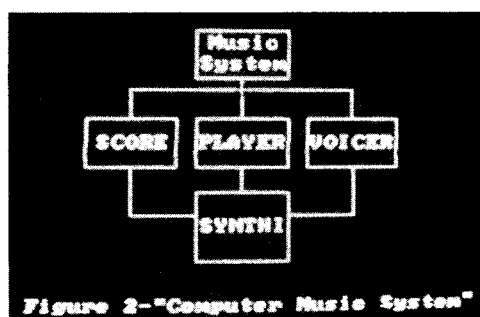
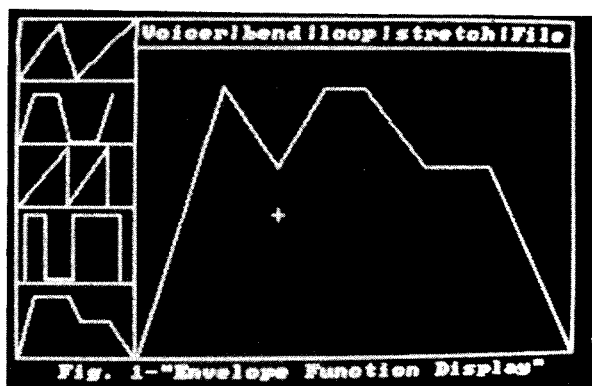
International MIDI association, 8426 Vine Valley Drive, Sun Valley, California, 91352

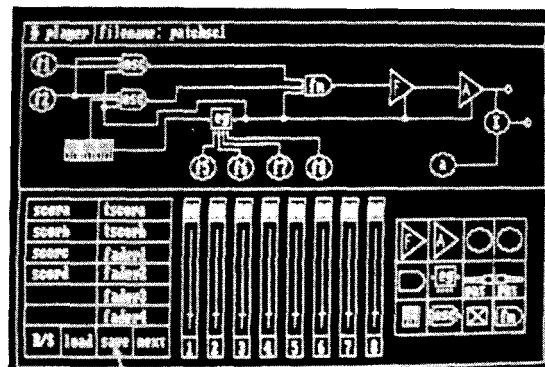
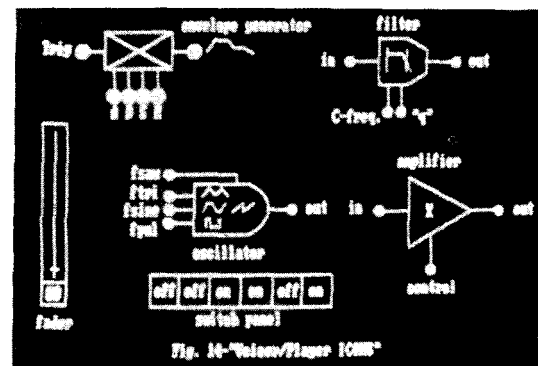
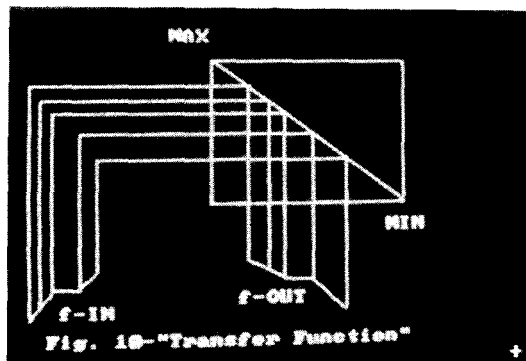
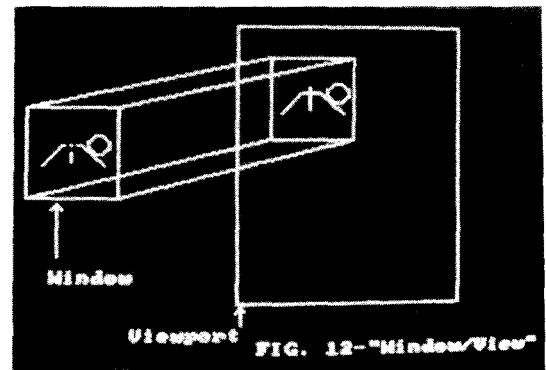
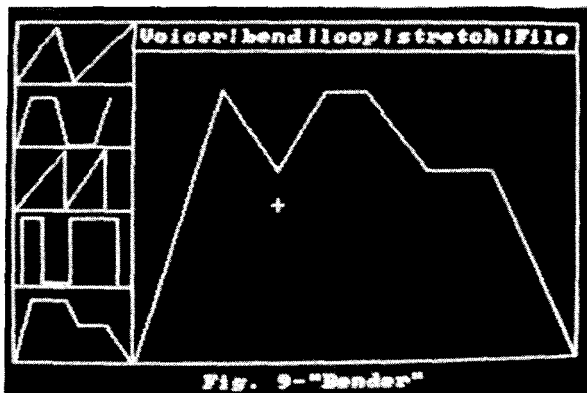
Mercuri, R. [1981], "Manuscript: Music Notation for the Apple II". IEEE Proceedings of the Symposium on small computers in the arts. pp. 8

Buxton, W., et al, [1979], "The evolution of the SSSP score editing tools", Computer Music Journal, 3,4, pp. 14

Buxton, W., et al, [1982], "Objed and the design of timbral resources". Computer Music Journal, 6,2 ,pp 32

Kowalski, M.J. and Glassner, A ,[1982] , "The NYIT digital sound editor", Computer Music Journal, 6,1 ,pp 66.





Teaching Programming to Musicians

Frances K. Dannenberg, Roger B. Dannenberg, Philip L. Miller

Computer Science Department, Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

A new approach has been developed for teaching programming to musicians. The approach uses personal computers with music synthesis capabilities, and students write programs in order to realize musical compositions. Our curriculum emphasizes abstraction in programming by the early introduction of high-level concepts and the late introduction of programming language details. We also emphasize abstraction by relating programming concepts to musical concepts which are already familiar to our students. We have successfully used this curriculum to teach Pascal to children and we are presently using it in a university-level course for composers.

We have developed a new methodology, designed especially to teach programming to music students. Students are challenged to produce artistic works through programming skill, using personal computers with music production capabilities. A key feature of our curriculum is that it allows students to use their existing musical knowledge as a basis for understanding computer programming. We have used this approach successfully to teach Pascal to children ranging in age from 9 to 16 years, and we are now using the curriculum as part of a college-level computer music course.

This approach is unique in several ways: First, it is designed specifically for artists. Programming is viewed as a means of creative expression rather than an abstract skill whose utility may be difficult to justify to an artist.

Secondly, our approach is an inherently multi-media one. We have found that "listening" to a program's execution while reading the program is helpful in learning and debugging. Finally, we build upon existing musical knowledge. Musicians are familiar with the concepts of sequence, repetition, conditional selection, and procedural abstraction from the domain of music. We make use of analogy to teach the corresponding programming structures.

In Section 1, we present the origin and goals of this project. Then, in Section 2, we describe some earlier work and experience that guided our curriculum design. In particular, we wanted to teach what we call the *abstractionist methodology*. Section 3 then describes our specific curriculum design for teaching programming to musicians. Our experience with this curriculum is discussed in Section 4, and we present our conclusions in Section 5.

1. Background

We began with the goal of designing the curriculum for a "Computer Arts Summer Program." The program was to be held at the American Center in Paris, and was aimed at 12- to 16-year-olds. We planned to include computer music and computer programming instruction and to provide every student with a personal computer in the style of many "computer camps" held in the United States. It was also decided to integrate the music instruction as much as possible with computer programming.

We considered two approaches to the use of computers for music. First, we could present fixed, menu-oriented programs for drawing, composing, and computer-aided instruction. Rather than writing their own programs,

students would manipulate parameters in existing programs. Alternatively, we could write interfaces to graphics and sound synthesis devices so that students could create music by writing their own computer programs. We decided to concentrate on the latter approach: that is, teaching students how to program in order to produce music.

2. Previous Work

We know of no work that addresses the needs of teaching programming to the musician in particular. However, there is a wealth of literature concerning programming methodology and pedagogy in general. Of particular interest are papers by Perlis¹, Dijkstra², and Hoare³, which discuss the importance of various forms of abstraction to programming. We call the general approach advocated by these authors the *abstractionist methodology*. Because of its importance to our curriculum for teaching musicians, we describe it here in some detail.

2.1. Abstractionist Methodology

We recognize three principal levels of programming abstraction: the control structure level, the procedure level, and the data structure level.

Abstract Control Structures. The most familiar level is that of control structures⁴. This is essentially the structured programming movement of the 1960s, with *do-while*, *if-then-else*, etc. In contrast to the *goto*, which may be used to create arbitrary flow of control, control structures should have single points of entry and exit, and they should indicate the programmer's intention, for example, to iterate a sequence of statements.

Procedural Abstraction. The second level of abstraction is abstraction at the procedural level. The idea is that problems are too complex to be thought about all at once, so we think about them hierarchically. To illustrate, we will borrow from a textbook⁵. Consider the task of grocery shopping. This high-level task can be divided into smaller tasks (subtasks) in many ways. Let's say we decide to decompose it into two subtasks: *generating a shopping list* and *buying all the items on the list*.

We continue now with the subtask of *generating a shopping list*. Likewise, it may be decomposed in a variety of ways. Assume our solution is first to obtain a pencil and paper, followed by examining the kitchen cupboards for some idea of what is needed, and finally, to consult the spouse for a contribution to the list. Generating the shopping list could be done in other ways. For example, one might simply delegate the task to one's spouse, cook, maid, etc.

With our shopping list firmly in hand, we can consider the subtask of *generating the shopping list* completed. We now turn our attention to the other major subtask, *buying the items on the list*. This, of course, could also be done in a variety of ways. Let's say we wish to do it in the following manner: go to the grocery store; collect the items on the list; pay the cashier; and, finally, return home with the groceries.

This leaves out many details of the acquisition phase. For example, we named a subtask *collect items on list*, however, we have said nothing about how this is to be accomplished. How are we going to search the store for the items on the list? Are we going to use a shopping cart,

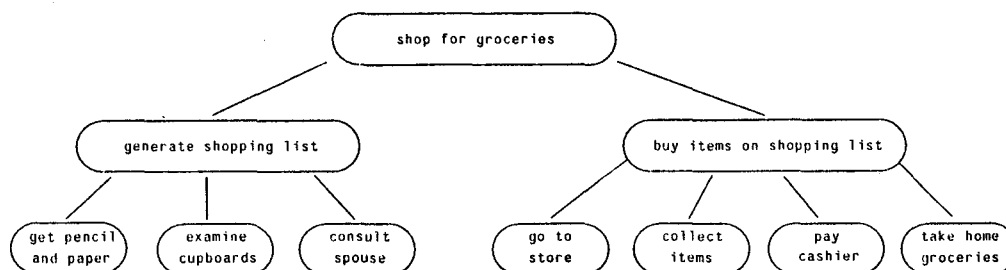


Figure 2-1: Figure Showing The Tree Structure Of Problem Decomposition

or perhaps just a shopping basket? We named another subtask of *paying the cashier*. This might be done with cash. It might also be done with a personal check, or with manufacturers' coupons, a charge card or some combination of these methods.

We see that this technique generates a hierarchical solution to the overall task. Although we've expanded only two levels, it gives enough of the idea for you to see how it is done. In programming, tasks are similarly divided into components which are then written separately.

Data Abstraction. Data abstraction is the business of thinking of a problem as a set of objects and the operations that are performed on those objects. Consider a payroll system. The problem is to maintain information on the employees of a company. The information that is kept on each employee includes such items as name, rate of pay, hours worked, whether or not the employee is participating in the company's group insurance plan. We can think of this information as an index card of information. Because the company has a number of employees, the index cards are arranged alphabetically into a shoe-box of index cards. Thus the abstract structure for the payroll problem is a shoe-box of index cards.

The second aspect of data abstraction concerns the operations on the structures. There must be a way to thumb through the cards, searching for a particular one. There must be way to copy information from a card and a way to change the information that is on a card. A card must be added when a new employee is hired. A card must be removed when an employee is terminated.

In programming, the box of index cards would be represented by a data-structure. It is desirable, in order to reduce program complexity, to confine the details of this data-structure to only a small part of the overall program. This is accomplished by writing procedures for each of the desired operations. If an operation is complex, it might be implemented by a package of procedures.

We find these three levels of abstraction to be at the core of good programming practice. They facilitate design of software that is at once verifiable, implementable, debuggable, and extendible. They dovetail neatly with the ideas of information hiding and strong typing. They represent current thought in software engineering.

2.2. Abstractionist Pedagogy

Recently, a few educators have begun to adopt a pedagogical style that is designed specifically to teach the abstractionist methodology. This style does not have a name in computer science, but we will call it the *abstractionist pedagogy*. Among the people with whom we are familiar, Bob Floyd is credited with the idea, which has since been applied in several textbooks^{6, 7, 5, 8}.

Among the key features of the abstractionist pedagogy are the early introduction of high-level concepts and the late introduction of programming language details. This encourages a hierarchical approach to problems, beginning with the highest level. Programming instruction begins with the introduction of a handful of pre-written procedures. The student writes his first program simply by calling these procedures sequentially. Next, the student is given a technique for writing new procedures, built from sequences of the primitive procedures mentioned above. Control structures are then introduced, and finally, a full programming language is presented to the student.

Standing head and shoulders above the rest in successfully executing this pedagogy is Richard Pattis. In the marvelous little book, Karel the Robot: A Gentle Introduction to Programming, students learn to manipulate a robot, Karel, using primitive procedures such as *Move* and *TurnLeft*. The robot is simulated on a standard CRT. Tasks are designed for the student, such as programming Karel to step over a hurdle or to escape a maze. As the student learns more powerful techniques of programming, successively more general and elegant programs that control the robot are written.

We believe that this task domain, one that is visual and tactile, is a good one for introducing programming methods. Unlike the domain of numeric calculations (the unfortunate standard fodder for beginning programmers)

the Robot world introduces no intellectual barriers to the student. It provides an environment that is at once intuitive and rich with analogies that can be exploited for introducing and fixing the rudiments of sound programming methodology. The book and the approach are now being adopted in a number of high schools in the U.S. and abroad, in part due to the positive recommendations of the College Board's Advanced Placement Computer Science Development Committee⁹,¹⁰.

2.3. Assumptions and Prejudice

We designed our curriculum for musicians with several assumptions in mind:

- The first is that the abstractionist methodology is sound and should be taught to beginning programmers.
- The second assumption is that there is a best way to teach this methodology. The abstractionist pedagogy has been used successfully at Carnegie-Mellon and elsewhere in programming courses.
- The third assumption is that musicians can learn the programming methodology. It is sometimes held that the mathematically-oriented students (engineering and science) are able to learn programming methodology, while artists are either unable or much less able to do so. It is clear, however, that some musicians are excellent programmers. Some are respected computer scientists. We decided that the best approach was to assume that for the purposes of programming, musicians as beginning programming students are no different from any other group of beginners.
- The fourth assumption, as suggested in our description of the course, is that the best way to introduce the concepts of programming methodology is to tie these concepts closely to a knowledge base that is familiar. In introducing a new concept, a successful teaching method is often to explain it by its analogy to some more familiar concept. In our course, we apply this to teaching programming abstraction, explaining programming structures to students by analogy to similar hierarchical organizations in music, with which they are familiar.

Thus far, we have introduced a number of important ideas. We have specified a programming methodology as the correct one to teach. We have talked about how to teach that methodology in terms of subject matter, texts, and software. How this all manifests itself in terms of teaching programming to musicians is the subject of the next section.

3. The Abstractionist Approach in a Musical Setting

When we began to design our programming course for musicians, we looked for musical analogues to the concepts we wanted to teach: sequential execution, procedural abstraction, and control structure abstraction. We were quite pleased to find musically meaningful analogies for all of these concepts. Below, we describe how each concept was presented to our musician/programmers.

For the introduction of procedural and control abstraction, we wanted to keep programs as simple as possible, avoiding issues such as parameters, input/output, and synthesizer interfaces. Taking Karel the Robot⁸ as a model, we defined a set of parameterless procedures to play the notes of an octave scale and to produce silence¹. A few more procedures were added to produce sound effects, and an *include file* mechanism was used to hide the definitions of all of these procedures. The use of personal computers made it possible for each student to have a machine that could edit, compile, and execute programs using these procedures. Each machine could also synthesize appropriate sounds.

3.1. Sequence/Melody

The first programming lesson consists of a simple melody and an explanation of how to translate the melody into a program. For example, the following melody:

¹The procedures are *PlayDo*, *PlayRe*, *PlayMi*, *PlayFa*, *PlaySol*, *PlayLa*, *PlaySi*, *PlayDo2* (an octave higher than *PlayDo*), and *Rest*. The names of these procedures were chosen to avoid a clash between the Pascal reserved word "do" and the solfège syllable "Do".



would be translated to:

```
program Melody;
{include definition file here}
begin
    PlaySol;
    PlayLa;
    PlaySi;
    PlayLa;
    PlaySol;
    PlayLa;
    PlaySol;
    Rest
end.
```

Students are encouraged to compose their own melodies and to program the computer to play them.

3.2. Procedures/Phrases

For the next lesson, an example is chosen that includes several occurrences of a musical phrase. The example is translated into Pascal, and it is observed that the program contains a duplicated sub-sequence of commands. Students are shown how to build a named procedure from the sub-sequence. For example, the following melody:



could be rendered as follows, using a procedure to implement measures 1 and 3:

```
program Lesson2;
{include definition file here}

procedure DoSiLa;
begin
    PlayDo2;
    PlaySi;
    PlayLa
end;

begin
    DoSiLa;
    PlaySi;
    Rest;
    Rest;
    DoSiLa;
    PlaySol
end.
```

3.3. Loops/Repetition

After programming a composition using procedures, we turn to basic control constructs. The loop is the first construct considered; its musical analogue is the *repeat*. At this point, we consider only the **for** loop. The next example illustrates the use of the **for** loop to program a musical repeat:



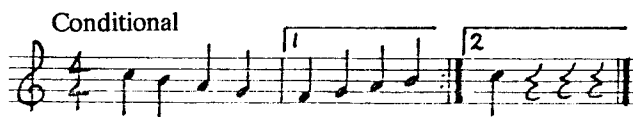
```
program Arpeggio;
var i: integer;
{include definition file here}
begin
    for i := 1 to 3 do
        begin {repeated measure}
            PlayDo;
            PlayMi;
            PlaySol;
            PlayMi
        end;
    PlayDo {the last note}
end.
```

For this lesson, Pascal requires a declaration for the loop control variable. Since variables have not yet been introduced, we avoid the issue by describing the declaration as a “magic incantation” to be explained later. A loop construct that implicitly declares its control variable¹¹ would be preferable for teaching purposes.

Perceptive students will realize that a musical repeat can be implemented by programming the repeated music as a procedure and calling it several times. An interesting work to discuss at this point is *Vexations*, by Erik Satie, which consists of a short musical statement to be repeated 840 times!

3.4. Conditionals/First and Second Endings

The next lesson concerns conditional execution. The analogue in music is the first-and-second ending notation. Consider the following example:



```

program Conditional;
var i: integer;
{include definition file here}
begin
  for i := 1 to 2 do
    begin
      PlayDo2;
      PlaySi;
      PlayLa;
      PlaySol;
      if i = 1 then
        begin {first ending}
          PlayFa;
          PlaySol;
          PlayLa;
          PlaySi
        end
      else
        begin {second ending}
          PlayDo2
        end
      end
    end
  end.

```

Students should be encouraged to experiment with conditionals in non-traditional musical structures. For example, conditionals could be used to introduce variations at several points in a repeated note sequence.

3.5. Parameters

Until now, no procedures have been parameterized. This simplifies the presentation of control constructs and procedures, but imposes rather severe limitations on the variety of sounds that can be programmed. In the next several lessons, students are taught how to call parameterized procedures and how to declare them. By this time in the course, students recognize the need for more subtle control over sound, and welcome the introduction of parameters.

Predefined procedures called Note and Rest are used to introduce parameters. The Note procedure takes arguments for frequency, amplitude, and duration; for example Note(440, 100, 50). The Rest procedure has one argument, duration; for example Rest(90).

Students are then taught how to define their own parameterized procedures. At this point, they have the programming skills necessary to create interesting pieces.

Advanced students will want more direct access to the sound generation hardware than that provided by the Note procedure. In our case, we use a fairly sophisticated synthesizer interface capable of independent time-varying frequency, amplitude, and waveform control over 16 oscillators. The synthesizer interface illustrates data abstraction. Procedures are used to manipulate some underlying structure (the synthesizer) in order to hide irrelevant details of the structure. Students are encouraged to develop their own data abstractions at the next higher level in order to obtain a control interface that is appropriate for their composition. For example, a procedure named Gliss could be written in terms of primitive frequency controls in order to implement a musical *glissando*.

4. Results and Discussion

We taught a 15-day course, where students had a total of 4 hours per day for instruction and access to computers. Although the course was intended for 12- to 16-year-olds, the actual range was 9 to 16 years. All of the students were able to develop programs that used procedural abstraction, loops, and conditionals. For example, one student, who had no previous computing experience, wrote an 83-line program to perform a piece with the structure *ABA* (see the appendix). The *A* section was implemented as a procedure with an internal structure of the form *abaca*. This was accomplished using a **for** loop to iterate 3 times, with a conditional to insert *b* after the first iteration, and *c* after the second iteration. This program used a variation of the Note procedure to give control over the rate of attack and decay of each note.

4.1. Music as a Concrete Programming Task

As expected, students understood the programming tasks immediately since they came from familiar intellectual territory. This allowed students to concentrate on the solutions to the problems rather than trying to understand the problems themselves.

4.2. "Listening" to Program Behavior

As with the domain of Karel the Robot, which can be simulated on a CRT, we found music to be attractive for programming because it was possible to follow program behavior quite closely. This was true in part because program behavior was slowed to a musical pace. Also, one could hear the result of each program step; consequently, one did not often need to deduce a program's behavior from its final output. Rather, the entire program execution was transparent, and problems could be isolated without a painful debugging process.

In addition, we found that the music domain has specific advantages over Karel. First, music is an ideal medium for transmitting large amounts of information about program behavior to our musicians. It is also possible to read a program listing visually while simultaneously following program execution aurally. This was valuable in helping students to learn the association between program statements and their actions.

4.3. Motivational Factors

Our students discovered that making music with computers is also fun and exciting. As students completed their assignments, they would perform their pieces for the class, often receiving applause and compliments. Students were highly motivated to finish their assignments!

4.4. Extension to Other Domains

Based on our experience, we feel that other domains could serve as an excellent basis for the abstractionist pedagogical style. The style is appropriate for various types of music synthesizers², but it might also be considered for the new, low-cost speech-synthesis devices. Another interesting domain is that of computer graphics. The "turtle graphics" interface is an example of an appropriate set of primitives¹². In another application, Harry Holland at Carnegie-Mellon University is using our approach to teach Pascal to artists. His students use a color graphics display and program in terms of primitives like Box, Circle, and Line. Architectural drawing is another possible domain. Finally, a mechanical robot is being constructed at Carnegie-Mellon University, based on Karel, to make the programming task more exciting.

²For this reason, we do not describe our lowest level synthesizer interface in greater detail here.

At Carnegie-Mellon University, the programming pedagogy is reinforced not only by the Pattis text and the Miller and Miller text, but also by software that was written with an eye to the abstractionist methodology. GNOME software is built so that procedural and control abstraction are the natural form of program construction. Details of syntax and some details of semantics (e.g. the order of procedure declaration) are issues for the programming environment, not for the programmer.¹³

We are currently using the abstractionist pedagogy as part of a computer-music course for college students. In this course, however, we introduce parameterized procedures at the beginning so that students have more music-making capabilities from the start.

5. Conclusions

We have presented our view of the proper pedagogical style for teaching the abstractionist methodology. The approach has been used successfully at Carnegie-Mellon University and elsewhere.

It was gratifying to discover that the approach can be adapted quite well to the musical domain and that musicians can indeed learn to program with the abstractionist methodology. In fact, music has specific advantages, including familiarity with the domain, program behavior that is audible, and a strong motivation to "compose" programs.

It is interesting to compare our experience teaching grade-school level students to that of teaching university students. Our goal with the grade-school students was primarily to teach programming, while in the university course, programming skills are primarily a means of realizing a composition. One conclusion is that there are limits as to how far one can integrate the teaching of music and programming. For example, the programming tasks described in this paper have little musical value to a university-level course in computer music, but the programming concepts are an important foundation for more sophisticated tasks. The problem is that a "toy" domain like Karel the Robot is ideal for teaching programming, but toy music domains are not attractive to serious musicians. We believe part of this problem can be solved by a better choice of synthesizer interface, and we intend to experiment further in future courses.

The number of musician/programmers is small, but the field of music has already felt their impact. It will be interesting to watch what musicians do with programming skills as they become more widespread.

6. Acknowledgements

It our pleasure to acknowledge a number of people and organizations whose contributions made this project a success. Raj Reddy deserves credit for the concept of a computer arts camp for children. Colette Wilkins was indispensable in teaching as well as in a multitude of other tasks essential to the success of the course. Judith Pizar, Henry Pillsbury, Alex Mehdevi and the staff at the American Center obtained equipment and handled innumerable problems in preparation for the course. Computer equipment was loaned to us by Atari France and Apple. Carnegie-Mellon University's Summer Studies funded the courseware development, which was programmed by Linda Isaacson, Richard Sean Keegan, Robert Rose, Peter Shell, and Mark Wilkins.

References

1. Perlis, Allan, "A First Course in Computer Science," 1965.
2. Dijkstra, E.W., *Structured Programming*, Academic Press, 1972, ch. Notes on Structured Programming.
3. Hoare, C.A.R., "Proof of a Program: FIND," *CACM*, Vol. 14, No. 1, January 1971, .
4. Dijkstra, E.W., "GOTO Statement Considered Harmful," *CACM*, Vol. 11, No. 3, March 1968, .
5. Philip L. Miller and Lee W. Miller, *Computer Science, The First Course*, Random House, 1985.
6. D. Cooper and M. Clancey, *Oh Pascal*, Norton, 1982.
7. Arthur Keller, *A First Course in Computer Programming Using Pascal*, McGraw-Hill, 1982.
8. Pattis, R., *Karel the Robot, A Gentle Introduction to the Art of Programming*, John Wiley and Sons, 1981.
9. The Advanced Placement Computer Science Committee of the College Board, "Advanced Placement Course Description: Computer Science," 1984.
10. The Advanced Placement Computer Science Committee of the College Board, "Teacher's Guide to Advanced Placement Courses in Computer Science," 1984.
11. William Wulf, D.B. Russell, and A. Nico Habermann, "Bliss: A Language for Systems Programming," *CACM*, Vol. 14, No. 12, December 1971, .
12. Harold Abelson and Andrea diSessa, *Turtle Geometry: the computer as a medium for exploring mathematics*, MIT Press, 1980.
13. David B. Garlan and Philip L. Miller, "GNOME: An Introductory Programming Environment Based on a Family of Structure Editors," *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, ACM, May 1984, Published as SIGPLAN Notices 19(3) and Software Engineering Notes 9(3).

Appendix

Listing of a Student Program

```

program Param;
var
  {note: this is the interface include file: }
  (*$ID2:INTER.DEF*)

  procedure Note(Pitch, Attack, Decay,
                  Amplitude: integer);

  const Voice = 0;
  begin
    WaitVoice(Voice);
    FDelay(Pitch, 0, Voice);
    ARamp(Attack, Amplitude, Voice);
    ARamp(Decay, 0, Voice)
  end;

  procedure Bizarre;
  var C: integer;
  begin
    for C := 1 to 3 do
      begin
        Note(700, 100, 100, 100);
        Note(750, 97, 97, 97);
        Note(800, 94, 94, 94);
        Note(850, 91, 91, 91);
        Note(900, 88, 88, 88);
        Note(950, 85, 85, 91);
        Note(1000, 100, 100, 90);
        Note(1250, 90, 90, 100);
        Note(1500, 100, 110, 100);
        Note(2000, 90, 89, 79);

        if C = 1 then
          begin
            Note(850, 75, 77, 75);
            Note(825, 78, 77, 78);
            Note(800, 76, 79, 78);
            Note(4000, 70, 120, 110);
            Note(4700, 120, 70, 110);
            Note(800, 89, 70, 75);
            Note(730, 95, 84, 77);
            Note(888, 100, 110, 95);
            Note(2540, 127, 71, 120);
            Note(1700, 120, 120, 90);
            Note(990, 110, 127, 75);
            Note(4000, 90, 89, 97);
            Note(4500, 85, 90, 100);
            Note(4700, 90, 110, 100);
            Note(5000, 100, 75, 89);
            Note(500, 120, 75, 100);
            Note(700, 90, 90, 100);
            Note(1000, 85, 90, 86);
          end
        end
      end
    end
  end

```

```

else if C = 2 then
  begin
    Note(5000, 100, 100, 90);
    Note(9000, 127, 127, 90);
    Note(8500, 127, 127, 90);
    Note(8000, 127, 127, 90);
    Note(7500, 127, 127, 90);
    Note(7000, 127, 127, 90);
    Note(2700, 95, 90, 100);
    Note(2000, 90, 99, 110);
    Note(900, 85, 90, 90);
    Note(700, 85, 80, 95);
  end
end;

begin
  MusIni;
  Bizarre;
  Note(350, 65, 65, 70);
  Note(325, 65, 65, 74);
  Note(300, 65, 65, 77);
  Note(375, 127, 127, 90);
  Note(400, 120, 120, 95);
  Note(385, 100, 100, 85);
  Note(300, 95, 95, 85);
  Note(250, 90, 87, 94);
  Note(215, 70, 78, 66);
  Note(207, 50, 56, 90);
  Note(200, 80, 76, 88);
  Note(189, 79, 87, 85);
  Note(206, 80, 75, 76);
  Note(200, 76, 47, 69);
  Bizarre;
  ARamp(0, 0, 0)
end.

```


AN ART BASED MICROCOMPUTER LEARNING SYSTEM

CATHERINE G. DEL TITO

Presentation Graphics 703 S. 3rd St. Phila. 19147

Abstract

Microcomputer learning outside of traditional classrooms and traditional student groups is essential in the growth of a technology based culture such as we are experiencing today. Computers reek of intimidation to the average lay person. Somehow this must

be altered, and forceful means are not necessary. The Computer Learning Center of Cleveland addressed this audience, drawing students ages 6 to 66 to individualized learning situations using the arts as the teaching medium.

"What are these boxes, that masquerade as typewriters and contain cloned brain cells of deceased geniuses? Why are they running my bank, brainwashing my kids, beating me at chess...." This is a question often heard (in so many words) from the braver members of that vast majority of humankind who see the computer as an alien life force positioning itself for takeover of the planet. How do we assuage this fear, appease the hostility, open the mind, even better, open the hearts of those who feel the enemy is us.

We took a simple approach. Use the computer for that which separates us from the from the other animals (and possibly the alien life forces), art. Use the computer to write poems, draw pictures, play songs and maybe it will become a friend instead of the enemy. The Computer Learning Center of Cleveland did just that. Armed with 20 microcomputers (Apple IIe, Ibm PC, Commodore 64, Trs-80 Model III, Atari 800), assorted peripherals, an energetic director, and a staff of artists, musicians and writers, we set out to bridge the gap.

The physical constraints were next to idyllic. A computer for every student. One teacher for not more than 6 students so as to approach individual attention. An easily understandable set of books, Thinking, Learning, Creating, as a core curriculum in BASIC. An array of reference books for LOGO, graphics, sprites, music and word processing.

Classes were one and a half hours each week for 6 weeks. Each student was master of his destiny. The teacher and student decided where to begin, depending upon the student's background, and where to head, according to the student's interests. The approach of the TLC books was straightforward, DO THIS and see what happens, THIS IS WHY it happened, CAN YOU MAKE IT DO THIS? This approach fostered creativity in asking the student to take a learned concept and

use it, e.g. make the computer print a poem you have composed. Because of the individual approach, it was hard to get lost; you could go into reverse, back up, and try again. If the student was familiar with material, they could move on readily.

We had a few rules. It behooved us to discourage canned software in favor of self-creation, though paint programs and music systems had their place. We outright outlawed games unless keyed in by the student or obviously created by the student. We also discouraged chewing on disks, kicking computers and asking the teacher questions about intricate assembly language routines. We did encourage any and all semi-plausible ideas. A group of students worked on translating the board game Clue into the computer. Page flipping on the Apple was used to make dogs bark and King Tut wink. Sound was used for the barks as well as the theme from Star Wars. The following "student profiles" highlight some of the work done. (I will refrain from educational analysis - the reader is free on his own there)

Jamie (age 7) had only one small problem, containing his enthusiasm. When he had the turtle in Atari LOGO make a SQUIREL (an endlessly varying geometric spiral), he would have to applaud. But when he then had it change colors, he had to let out a "Oh, wow, neat" so people would come and look.

Jamey (age 15) had a favorite response to any question, "I guess so". He started by plotting a hot air balloon (in Basic) on an Apple. But balloons are supposed to fly. We simplified the balloon to lines (vectors) which we translated to hexadecimal numbers (even if he did not fully understand them). We entered the hex codes in a shape table and wrote a program in Basic that had the balloon lift

off the ground and fly around (using draw and undraw of the shape table). The balloon looked pretty dumb on a blank screen so Jamey used Micropainter to draw the Alps for the background. Unfortunately the Apple resolution prevented a passenger in the balloon so it was good that the Basic program was steering.

Kim (age 15) was one of those students that try as you might to confound them, they keep coming back for more. By her 7th session she had taken things into her own hands. She took a basic plot routine written by one of the teachers for the IBM PC and developed it into a drawing program which enabled one to plot lines, circles, fill areas, select colors, erase areas, save pictures and catalogue a disk. We showed Kim and her work off proudly to all visiting dignitaries.

Some of the best students were the parent/child duos where traditional roles became blurred. Parent and child became a worthy team, taking turns asking the right kinds of questions and taking their respective guesses. Each brought a different perspective to the learning that complimented each other.

We pitted students against each other in battles of wit. We set 2 brothers to work designing a quiz program to see who could come up with the silliest answers. We challenged 2 schoolmates to translate a magazine program for their respective mach-

ines. We had tales of ghosts and Reagan coming from the word processor. Yes, we had fun, but not at the expense of those who chose more conventional modes of learning. The environment was created specifically to accommodate the range of learning styles.

Enough of anecdotes. Summary please. There is a cavernous gap between those of us who pioneer technology and write papers and those of us who do not understand why the computer keeps getting their bills wrong. Our plan was to cross that gap. It worked. Kids came back for more (voluntarily) and adults left unafraid of alien beings. It makes good sense that learning takes place when the mode of learning is personal and human.

So, why you may ask, is this paper written in the past tense? The learners who came, for the most part conquered. But how do we reach those huddled masses who are ignoring the enemy COMPUTER and hoping it will go away? How do we convince them it is safe to give it a try? These questions we couldn't answer as easily. All the marketing analysts in the land of OZ didn't seem to have the answer. The school is merging with another center which teaches business software using a similar individual approach. The original concept, teaching microcomputer use with art as a medium is very powerful. Now, how do we make it stay?

USING COMPUTERS TO WRITE AND TO TEACH WRITING

Donna Mansfield

Much has been written about the use of computers in art and music, somewhat less attention has been directed to the use of computers in professional writing and the training of writers. Books on "computers and writing" tend to be about using a word processor, or describe how a writer can adapt to using a computer. This paper will be about using computers to write and teach writing, but it will not be a tutorial on word processing. Rather, it will be an attempt to make some sense of the effect computers have on an individual's writing style and ultimately, on the product itself.

Some writers resist computers because they see them as the ultimate Twentieth Century expression of gimcrackery and believe that this "souless" machine will interfere with the magic of their art. Others have taken to computers with the fervor of the zealous convert; paper is no longer a word in their vocabulary, hardcopy is the term of choice.

Within this exaggerated and extreme range are writers who use computers regularly to write. For them, the computer is a tool which helps them to compose and construct documents. Neither hated, nor loved, the computer is merely an electronic pencil.

Writers in this group use the computer in different ways. Some cannot compose at the computer and continue to write their outlines and first drafts in longhand. For them, the computer is an electronic typewriter, but yet something more, because they can revise and polish a draft into a document without needing to retype each version.

Others use the computer for everything. They type the outline on the computer, keep notes on it and compose the first draft right at the keyboard.

This reflects the different styles that writers bring to their business. The computer facilitates a variety of styles. The writer can do little at the keyboard, or everything at the keyboard.

The ultimate question, of course, is how this tool affects the final product, and there seems to be no definitive answer to that. At this point there seems to be a "wish list" of what a computer could do for a writer or student of writing. This list includes some of the most repeated current "wishes":

- ensure that the writer produces grammatically correct sentences
- organize documents better
- check style
- write more 'naturally'

We will examine each of these to try to determine both how a computer does this and how this affects the writer and the product.

Checking the writer's grammar and spelling

There are a number of programs that check for basic grammatical problems such as subject-predicate agreement, spelling, punctuation and transistional phrasing. These are certainly a convenience for the student or the writer who has mechanical problems, but they will not cause a frog to write like a prince. (Imagine running a program like that on *Finnegan's Wake* or *Clockwork Orange*).

However, there are some students and would-be writers who believe that mechanical problems with the language prohibit them from putting anything on paper. This group would be greatly encouraged by the presence of a non-judgemental critic. The computer will help them become aware of the particular problems they have with the language so that they can work on resolving these difficulties.

Many professional writers use spelling and grammar checkers as a proofreading aid, but this has little effect on the ultimate product.

We can probably conclude that programs like this are an aid to the uncertain and reluctant writer and a convenience for the professional. They will enable the writer to at least produce mechanically correct writing and perhaps, to pinpoint the areas in which he has difficulties with the language.

Organizing a Document Better

One of the biggest problems facing a writer is how to organize his

document. Individual sentences and paragraphs, as well as the entire document must be well-organized for the final product to be successful.

One traditional method for organizing a paper has been to write important quotes and points on index cards and arrange and re-arrange them until a coherent, logically argued point of view is expressed. While this method has some value, the writer is only arranging subsets of prose, not entire passages, this system is, in effect, a relatively sophisticated outlining procedure. The final document still may or may not be well organized.

The computer shines at this point in the writing process because it enables the writer to see several different ways in which the material can be organized. The computer facilitates experimentation and also allows the writer to evaluate the different structures as he sees them in print. Seeing differently organized versions is obviously, quite different (and better) than guessing what the final document will look like.

The computer has real potential for resolving and refining the organizational problems of writers and students. And because this is one of the most important aspects of writing, the computer here has some very real benefit for the writer.

Writing "Naturally"

Starting a project is for most writers and students the most difficult part of the writing process. Even professional writers report that it is frightening to look down at a blank piece of paper knowing that within some deadline he is supposed to produce a polished, organized and clever document.

For students of writing, choosing a topic is itself a slippery process. "Choose a narrow enough topic to write about", the teacher says. This presupposes that the student has been able to think of anything at all, and is probably despairing at the thought of writing anything longer than two paragraphs.

Pre-writing programs

Some educators have designed computer programs to help students choose and refine a topic. These are frequently called "Pre-writing" programs. Most often they are designed as dialogues. The student answers general questions about his interests and then the dialogue concentrates on one interest hoping that the "discussion" will jog some likely paper topic in the student's mind. Once he has settled on a general topic, the dialogue with the computer helps him refine the topic down to some manageable level.

Programs like this get the student over the often insurmountable hump of choosing a good topic. The dialogue with the computer, it is hoped, will crystallize random thoughts into a likely topic.

Presently though, the dialogue in many of these programs is so general that it is hard to see how much help they can really be. The student can type whatever nonsense he chooses into the machine and the computer will still respond "Thank you, Johnny. What interesting projects you have chosen!" Perhaps, it is just that these programs are at a very early stage of development. With a little refinement, programs like this can be a real aid to the beginning writer. And as for its effect on the final product, well, there can be no final product without a beginning.

Compose programs

There are some writing theorists who believe that some people are so blocked by the initial process of writing that they can't begin at all. "Compose" programs have emerged as a solution to that problem.

In this sort of program, the student sits down at the computer and just types out the first 100 words that come into his head without worrying about grammar, punctuation or correct word choice. Out of this "spilling your guts" is supposed to come a topic and the first step toward beginning a project.

While this is an interesting idea, there are two obvious, immediate problems. The first is that typing for some people unfamiliar with keyboards of any sort is often as frustrating as beginning to write.

I taught a "Technical Writing" class composed of mainly engineering students. The class was part of a research project to measure the effect of using computers on the students' feeling towards writing. The investigator allowed the students a choice of using the computer or not. The investigator's hypothesis was that the students' writing would improve because they would prefer writing with computers (to not using computers). In fact, only two (out of 25) students used the computer; the rest did assignments in longhand. I believe that this was because the students found it very difficult and unpleasant to type. One could conclude that the "spilling your guts" method of writing was a good beginning technique for writers, but that it can be done just as well using a pencil and paper as a computer. In this case, working with an audience who found it foreign and unpleasant to type, using a computer may inhibit

and frustrate students more than it frees them to write. It seems likely, that once this group is experienced and comfortable at a keyboard, programs like this may greatly ease a writer into his project.

The other problem with this method is that some people cannot "spill their guts" on paper. Their natural style is to polish and write precisely, but then, one might say that these people do not need the help of a "compose" program to begin.

Conclusion

It may seem taken in toto, that the computer has a limited effect on the writer and his final product, but I believe, in fact, that the computer is a great tool for the writer because of the flexibility it allows. A writer does not have to change his style to use a computer, but rather can use the computer to enhance his already established style. If a writer composes quickly, right at the keyboard, he can use the grammar and spelling checkers to correct the mechanical part of his first draft. If he is worried about the organization, he can cut and paste sections in a variety of different ways until he finds the one that does the job best.

For some of us, the best part of using a computer is the speed and ease with which you can revise and polish. You simply do not have to begin at the beginning everytime you want to change something, you just change that part and the rest of the document stays the same.

Educators are still finding their way with computers. Many of the educational programs that are being put on a computer today were theoretically outdated five years ago, but some people seem to feel they will

have new life if they are put on a computer (or people will not recognize the same, deadly drills in a new format).

I believe that much of the future success in using computers to write or to teach writing will come with the realization that the writer can use the computer the way he wants it. Spelling programs are nice, grammar programs are nice, but what is wonderful is that you can develop and refine your style using a computer. You do not have to adapt to someone else's style. And there lies much of the difficulty in writing anyway. Some teachers believe there is a right way to write (their's) and a wrong way to write (your's). I am not talking about grammar and spelling, a writer must have these internalized to succeed in his field. But I am talking about how you begin a project, whether to outline or not, whether to polish all the way through or only at the end. You can do whatever is best for you when you write with a computer, you can do this with a pencil, too, but you can do it so much faster and easier with a computer. Flexibility, I believe is the computer's greatest aid to a writer.

Animating at Ohio State:
Thoughts on a Graduate Program in Computer Graphics.

William J. Kolomyjec, Ph.D, M.F.A.

Departments of Engineering Graphics and Art Education
The Ohio State University - Columbus, Ohio

Abstract

An evaluation of a state-of-the-art graduate program in computer graphics/animation in the Department of Art Education, The Ohio State University by a participating faculty member/ computer artist. Observations are made concerning computer graphics/ animation graduate programs in general and two courses in the OSU program specifically. Discussion is intended to illuminate aspects of providing computer graphics/ animation instruction at the post-secondary level.

Introduction

I was privileged to attend the first two courses of a three course sequence in computer graphics/ animation offered by the Department of Art Education, The Ohio State University. These courses are the heart of a graduate program in computer graphics/ animation. To add a historical perspective: The program is the fruit of much labor by Dr. Charles Csuri, long time computer artist and arts educator. Dr. Csuri has spent over twenty years in academics working to make the Ohio State program one of the finest programs of its kind in the world. He is also a partner in Cranston Csuri Productions, a major commercial computer graphics/ animation venture.

The academic home of the animation 'group' faculty is the Department of Art Education but the computer graphics facility, called The Computer Graphics Research Group (CGRG), falls under the office of the Dean of the College of the Arts. Graduates of the program receive MA or Ph.D degrees in Art Education. Computer Science and Engineering students are affiliated with the program through their research efforts with the CGRG and may receive graduate degrees in Computer and Information Science (CIS). The program is unique because both student types utilize the facility and are constantly interacting

with each other. The program is jointly administered by Dr. Csuri and Dr. Thomas Linehan. Dr. Linehan is a professor of Art Education and the associate director of the CGRG. As a testament to the demand and reputation of this program, last year there were over 400 applicants to the Art Education graduate program in computer graphics/ animation, 12 were admitted.

This article represents some of my thoughts based on my experiences as an observer and participant in the program. Last academic year, 1983-84, I petitioned for and was granted a joint (no salary) appointment in the Department of Art Education at Ohio State. Although I have a full-time appointment in the Department of Engineering Graphics at OSU, I have, for a number of years, been an active computer artist (Kolomyjec 1976, 1981, Peterson 1983). Receiving the appointment in Art Education has allowed me unique participation in the program. It has allowed me to make the transition in computer art from two dimensional static imagery to three dimensional computer graphics/ animation. I am indeed grateful to my colleagues and friends Drs. Csuri and Linehan.

General Observations

Before I make specific observations about computer graphics/ animation courses, I would like to make several general observations about the subject matter and the University. My first observation is that computer graphics/ animation has its own identity. It is a force that must be recognized and reckoned with. Over the years it has grown as a synthesis from many traditional programs at the University. Computer graphics/ animation no longer belongs exclusively to the sciences or to the arts, although those within it must take care not to forget where its roots lie.

Fifteen years ago you had to be an engineer or scientist to gain access to the medium. Today the medium can be found outside of the technical milieu. More and more computers are found in education and art colleges. Granted, most of computing in these areas is of the microcomputer variety, nevertheless, I find this very significant. In fact, using the CGRG as an example, computer graphics/ animation is the envy of students of engineering, computer science and the arts. In many ways, I feel this graduate program represents the next level of computer aesthetic instruction at the post-secondary level.

Two, there are very few programs in computer graphics/ animation to be found in higher education. I doubt that nationwide there are no more than five bonafide programs (I know of three, and one is in Canada). I would define a bonafide program to be one that offers either an undergraduate or graduate degree in computer graphics/ animation. As a consequence, very little courseware exists. Curriculum concerns such as educational goals and objectives, and content of instruction are still evolving. Of course, this tends to frustrate University and program administrators, which in turn slows down program development.

Any institution, college or department that is interested in providing instruction of computer graphics/ animation will ultimately have to deal with four major issues: One, the immense cost of the medium, i.e., hardware, software etc., and the issue of where it belongs in the framework of the institution. Second, when these primary issues have been resolved and programs are being put together, curricular concerns are a new frontier. Just look at discipline areas, the ingredients of a successful program, that are encompassed by computer graphics/ animation: funding and administration, computer science, various forms of engineering, aesthetics, conventional animation, film, video. Putting together an academic program is a nightmare of vested interests, territoriality, and new thought in direct confrontation with old ideals.

Three, there is a need for qualified teachers. There are very few faculty who have the skills, the experience and the academic credentials to meet traditional University requirements. Very few individuals can be found (or even identified) who embody the synthesis of science, technology and art who are qualified to teach computer graphics/ animation and have the qualifications to be given graduate school appointments. Very few qualified individuals are willing to tolerate the hassles of University reward

systems, much less ignore the big money to be made in this area outside academe. As a consequence, I fear very few quality academic programs in computer graphics/ animation will be forthcoming from higher education for a number of years.

Lastly, there is a well-known need for standardization in computer graphics and teaching systems ought to be built to conform to these general standards. The personal computer can help a lot by providing basic competencies in computer graphics. But, no one is in agreement about what type of hardware is better or what operating system is preferable. At the next level, computer graphics/ animation media requirements go beyond personal computers. The present trend would seem to be: 32-bit processors, megabytes of storage, Unix operating systems, vector and raster display algorithms, multiple bit frame buffers and some kind of image recording device, such as a 16mm camera system with animation motor. Never forget that any state-of-the-art computer graphics/ animation system will require support from systems programmers and maintenance contracts.

The Courses

At the 1983 Symposium on Small Computers in the Arts a paper was given that presented information explaining the hardware configuration, software and the outline of a course to be given to a beginning computer animation class around the Digital 11/23 microcomputer system (Schweppe, 1983). Twelve entering graduate students and myself comprised the class. Below is a summary of the syllabi of the first two computer graphics/ animation courses. It will serve to illuminate discussion in the remainder of the article. The length of each course was ten weeks.

The First Course.

The content of the first course consisted of: Introduction to Cranston Center and the computer facilities. Initial discussion of the hardware, operating system and the text editor. Data construction and the use of data generation software. Introduction to 'scn-assmblr,' vector and raster scene description software. Concepts related to three dimensional primitives and their use. Introduction to storyboards and animation principles. Introduction to the 'C' programming language. Use of 'C' to produce 'scene files' for the scene description software. Topics related to programming animation.

The Second Course.

The content in the second course consisted of: Readings in conventional animation techniques. Intermediate and advanced tutorials in the text editor and the 'C' programming language. Readings involving three dimensional computer graphics. Discussion of timing in animation. Production of more elaborate storyboards. Programming pencil tests of motion in vector. Hand-building complex data. Writing and producing 'complete' computer animated sequences. Oral presentations of assigned reading in a variety of areas related to computer graphics/ animation.

Specific Observations

Let it be known that I am clearly biased, the two term experience I had was truly wonderful and personally self-actualizing. I do not wish to be critical of the Computer Graphics/ Animation Program in the Department of Art Education at The Ohio State University. Rather, I prefer to speak in a friendly witness. Furthermore, I make these specific observations in light of the general observations stated earlier.

Observations related to the courses.

To say that the first course was intensive and fast paced is a slight understatement. The first task was to learn the idiosyncrasy of the computer system that was used in the course, a PDP 11/23. Initially, this meant learning the Unix(TM) operating system and a text editor called 'ed.' Later, we were introduced to some C program utility software, followed by an introduction to the syntax of the programming language C. Since most individuals met the two programming course prerequisites, as a whole, it went fairly well. Help from C programming experts was generally available.

Since three dimensional computer graphics is the brand of computer graphics/ animation taught at OSU (as opposed to two dimensional paint system animation taught elsewhere), the data building assignment is essential. It is my observation that most artists can "see" in three dimensional space but very few know how to use the formal graphical language of engineering graphics. I think some individuals would have benefited from a review of orthographic projection, all would have benefited from a presentation of the fundamentals of descriptive geometry and exercises in solving problems in three space.

The first course was not without technical problems, but this is to be expected in any developing course (or program). Using state-of-the-art software is both a blessing and a curse. Data generation software is a great idea. It is intended to eliminate the drudgery of hand-building data. However, since the data generation software we were asked to use was under development it contained some bugs. Complete segments would not work and when data was produced, it would not work in conjunction with other software. Most individuals enjoyed using the software but became frustrated when it did not perform properly. In all fairness to the program, it is better to work with experimental tools than not to have any tools at all.

On the other hand, we used some great, reliable, software, software that was well supported and maintained. Furthermore, systems programmers were able to improve and enhance software based on our suggestions. A program called `scn_assmblr` is particularly noteworthy. It was used to generate single frames of three dimensional vector and raster imagery. (See Schweppe, 1983 for a description of `scn_assmblr`.) Scene descriptions were initially generated in a "local mode" by the keyboard input of a few commands. A beginning exercise of generating several scene descriptions in this manner was an excellent way to become familiar with the syntax and capabilities of `scn_assmblr`.

`Scn_assmblr` is a wonderful piece of software and it worked very well. No individual student had any major problem with using it. However, the power of the `scn_assmblr` program (and the underlying power of Unix) lies in its ability to execute scene files. Scene files were the output of C programs written to generate a series of commands for `scn_assmblr`. (Scene files were essentially "printf" statements of `scn_assmblr` commands.) Under the control of a scene file, `scn_assmblr` could be programmed to generate an image and signal the animation motor of a 16mm camera. Thus, by providing these instructions repeatedly in our scene files we were able to create our animations.

Once the mechanics of animation were understood, the process called animation had to be addressed. This in itself is an art. Storyboarding is the procedure we used to visually describe the intentions of our animations. By providing essential keyframes, as in a comic strip, we planned our ten second sequences.

Ten seconds is not a long time, at twenty-four frames per second it amounts to 240 frames of animation. Depending upon the complexity of the scene a single frame can take several minutes to "render." In

our first animations we were limited to the use of two "objects." Frames were rendered in less than a minute of time. Thus a typical 10 second animation would take about 4 hours to film. However, most of us quickly discovered the notion of 'trade-off' in our second animations. In my own case, after I was able to get my C program to run, it was taking over 8 minutes per frame to render a scene. Since the end of the term was approaching, we were limited to seven hours of film time per person, my problem was to somehow cut a 32 hour animation down to 7 hours. This was a typical situation and here's how most of us adjusted. First, you redesign your three dimensional data (objects) to consist of the fewest number of faces (polygons) possible. Also, if you are only going to see the object from one direction you remove its back side (like the trick of using a building facade in Hollywood movies). The fewer polygons involved, the faster the display algorithm can render a scene. Next, you have to double or triple click (record multiples) of a single scene. Using a combination of these techniques I was able to cut my filming time down to a little over 7 hours.

In the first course you have to learn a lot to do a little. Much of it is discovery learning, that is learning by doing. Also, information must be shared between individuals, it is impossible for the instructor to anticipate every situation and prepare each individual student for it. However, good instruction is the key to the overall success of the course. By an organized presentation of essential materials (concepts and principles) and supportive material (reference articles and manuals) the motivated students can be successful.

It was my opinion that the second course was a little disappointing in that it was not as well organized as the first. This comment is not to be misconstrued as negative, merely an observation. I prefer to call it constructive criticism. After a very well taught and thoughtful presentation of material in the first course, the second course fell apart at times. For a variety of reasons, it took the first couple of weeks to rekindle overall motivation.

The second course began by superficially trying to reinforce concepts learned in the first course. This attempt should have been better organized. There seemed to be a big gap between terms. I feel some attempt should have been made to exploit discoveries made the previous term. A discussion or review of each individual animation from both an aesthetic as well as technical point of view would have been

appropriate. I think programming techniques themselves could have been discussed. This could have been a valuable experience, and it was lost.

In the second term a single animation assignment was made. Each individual was given free-reign (all things to be considered) to produce a complete animation of unspecified length. Each student was to conceive, storyboard, program pencil tests in vector, provide a sound track for and film in raster his/her animation. I have mixed feelings about this, on one hand it's wonderful to have all this creative latitude, on the other hand, people (especially creative people) tend to bite off a bit more than they can chew. In the final analysis, this approach worked well for some and it was deadly for others, namely those students whose programming ability got in the way of their creative undertakings.

Other aspects of the second course are noteworthy. Assigned readings were an important component. If nothing else, you begin to appreciate the range, breadth and diversity of the knowledge areas that computer graphics/ animation encompasses. Equally important were the class discussions of those readings. Readings were assigned from far-ranging areas; from traditional animation (writing by Thomas, Johnston and Halas) to articles about other computer animation software (MUTAN, Gramps, ASAS etc). Also, the project involving the design and construction of a complex object was a useful and practical experience for many.

Observations not related to the courses.

If you want a quality graduate 'arts' program then individuals selected to participate in the program must have documented aesthetic skills. A careful scrutiny of portfolios, as well as a demonstrated ability to perform well in at least two prerequisite computer programming classes forms the minimum standard of acceptance. Good programs are made up of good people. At OSU, it would be fair to say that selection criteria is heavily weighted (as it should be) in favor of the individual who has demonstrated an aesthetic sensibility. Much credit should be given to the directors of the program for this emphasis. The virtue of this philosophy was visually evident in the quality of each student's work at the end of each quarter.

On the issue of an undergraduate program. At Ohio State, the present thinking about an undergraduate program in computer aesthetics is that it is an important but secondary concern. There is

a general consensus of thought that the primary effort should be to develop a sound graduate program. Once in place the graduate program can play a major role in the development of the undergraduate program. I feel that this is a logically sound approach, especially when you think about the resources, particularly of the human variety, that would be readily available to the undergraduate program. Moreover, it would provide the possibility as an income source for graduate students, and provide a training ground for future professors of computer aesthetics.

Other observations: When twelve people are, at times, forced to use one system due to any number of disastrous situations, they'd better be able to get along. This means that the program administrator must go out of his/her own way to facilitate a friendly working environment and maximize avenues of communication. This of course goes for persons delivering instruction and systems people. Also, due to the variety of the skills and backgrounds of the people affiliated with the program, it is to be expected that people need to find a common language. Artists should not be isolated from computer programmers or systems people and vice versa. Interaction should be encouraged.

Some final comments about hardware and film unique to computer graphics/animation. Whether the system is a stand alone workstation or a multi-user system with several terminals, a comfortable number per workstation/terminal is 5 or 6, ten or more is too many. Individuals can and should maintain their own files on a regular basis. Hard disk cartridges, floppies or tape should be purchased and used by the students to back up their own files. Not all terminals need to be graphics terminals. Entering, editing and executing practice programs to learn language syntax can be done on less expensive alphanumeric terminals, if possible on a system with non-graphics peripherals.

Filming can tie up a system for days, make sure the bulk of a class or group's programming is completed before filming. Timesharing systems can and do crash while someone is filming. This can often be avoided by not doing program development and editing on the same machine/ system while it is being used for filming. Filming requires cameras with animation motors which can be software activated. Film requires sources for purchase of materials and processing, as well as editing and projection equipment.

Summary

Computer graphics/ animation has emerged as a discipline area and rightly deserves its own identity. Presently, there are but a few educational programs to be found. In general, the major obstacles that must be overcome by any group, college or institution interested in providing a computer graphics/ animation program are: cost, curriculum, finding capable faculty, and an overall lack of computer graphics standards in terms of both hardware and software.

The courses under development at The Ohio State University are unique and serve as a model to others interested in developing computer graphics/ animation programs. Their graduate courses in computer graphics/ animation are intensive. Course content initially focuses on the use of hardware and software. Strong arguments can be made for taking a three dimensional approach but perhaps a stronger emphasis on three dimensional concepts ought to be included. As competencies are acquired more traditional instruction in programming techniques and animation principles is necessary. Quality instruction and motivated students are essential to the success of the program.

Computer graphics/ animation instruction based on a microprocessor is time consuming and requires inventiveness on the students' part. Perhaps smaller assignments with more constraints would have been better in the courses to optimize overall use of the equipment and to provide a more positive (successful) experience for all students. A bibliography supported by a library and a good sized quantity of reference and resource material is an essential ingredient to the program.

In any developing program there will be problems with the use of new tools but the benefit far outweighs the consequences of not having any tools. Having a parallel research effort nearby makes all the difference! *ScnAssmblr* is an excellent piece of software for doing computer graphics/ animation. Unix(TM) and the C programming language provide a workable environment. Good organization and technical support are essential to the success of the program.

Due mainly to the issues of funding and institutional politics there will probably never be more than a few academic centers for the instruction of computer graphics/ animation in the United States. Development of programs will be hampered by the lack of qualified faculty more so than by a lack of equipment. Although

associated hardware requirements go beyond what is presently available in personal computers, this probably will be less of a problem with new developments in hardware in the near future.

The graduate program in computer graphics/ animation in the Department of Art Education at The Ohio State University is an important focal point of post-secondary instructional activity. In computer aesthetics education it represents the next level. The OSU graduate program in computer graphics/ animation has been in place but a couple of years. It is a program that strives to keep current with an evolving technology. Thus, the curriculum has to be flexible and state-of-the-art. It is growing, but not without growing pains. Yet, I have watched it evolve into a better program every year. I find the program fascinating and I am grateful to be associated with such a fine group.

References

Kolomyjec, W.J., "The Appeal of Computer Graphics," Artist and Computer, ed. Ruth Leavitt. Harmony Press. New York. 1976.

Kolomyjec, W.J., "Thoughts on Computer Aesthetics and the Future Role of Small Computers," Proceedings, First Symposium on Small Computers in the Arts, Philadelphia, PA. 1981.

Peterson, D., Genesis II: Creation and Recreation with Computers. Reston Publ., Reston, VA. pp. 61-64.

Schweppe, M. "Developing a 3-Dimensional Animation System for the Digital 11/23 Microcomputer System," Proceedings, 3rd Symposium on Small Computers in the Arts. Philadelphia, PA., 1983.

Author Index

Dannenberg, F.K.	114
Dannenberg, R.B.	114
Del Tito, C.G.	123
Holynski, M.	23
Kaprow, A.	52
Keith, M.	98
Kerlow, I.V.	17
Klapholz, J.	89
Kolomyjec, W.J.	129
Leeman, R.	4
Levine, S.	103
Lewis, E.	23
Lucas, R.E.	1
Mansfield, D.	125
McGinnis, J.J.	50
Metros, S.E.	37
Miller, P.L.	114
Naumowicz Zacher, C.M.	65
Palyka, D.M.	7
Prusinkiewicz, P.	58
Sachter, J.E.	42
Shafran, J.K.	52
Shortess, G.K.	72
Wagner, A.	29
Wilson, N.	76
Wright, W.	76

PROCEEDINGS

**4th. symposium on small
computers in the arts**

ISBN 0-8186-0610-X
IEEE CATALOG NO. 84CH2087-5
LIBRARY OF CONGRESS NO. 84-81880
IEEE COMPUTER SOCIETY ORDER NO. 610

IEEE COMPUTER
SOCIETY
PRESS